Proposal for a Reinforcement Model of Learning that Incorporates Beliefs and Signaling

Eleftherios Gkioulekas

1 Introduction

This is a proposal for a reinforcement learning model of non-cooperative games. In a non-cooperative game, players are in a situation where they have to choose a strategy at the same time. No player can wait to observe what the others did before making his move. This makes these games very interesting becaus every player, to play the game well, needs to get into the other players minds and outsmart them. As a consequence, when the game is repeated and information about the game's outcome becomes available to the players, they acquire experience that enables them to make better choices. This is the effect of learning, and it is this process of learning that we want to model.

An example of a non-cooperative game that we will study in this paper is the so-called "beauty-contest" game. In this game we have n players and they are supposed to pick an integer between 0 and 100. When all players turn in their choices, the average is computed, and then we multiply the average by a factor p which is $0 < p < 1$ to obtain the *winning number*. The player who is closest to the winning number wins a payoff and everybody else loses. When you are asking players to choose a number, what you are asking them in reality is their beliefs about the rationality of other people. To see why this is so, suppose that everybody playes 100. Then the winning number is going to be $100p$. It follows that if everybody chooses a number at random, then if there are players that chose a number below $100p$, then the players that chose a number over $100p$ are guaranteed to lose. The only players that will ever choose a number over $100p$ then are the players that chose the number at random. These are what we will call $level - 0$ types. Smarter players will assume that everybody else chooses at random, and to beat that they will choose a number bellow $100p$. By a similar argument, if everybody choses $100p$ then the winning number will be $100p^2$ and if some wiseguys pick a number bellow $100p^2$ then the ones that picked a number x between $100p^2 < x < 100p$ will all lose. We call these somewhat sophisticated losers level-1. As for the smarter guys that assumed that everybody was a level-1 type and chose a number bellow $100p^2$, we call these level-2. Experiments suggest that when players play this game repeatedly, they choose smaller and smaller numbers and eventually converge to zero. This tells us that previous experience affects the decisions of players, and this effect we call *learning.*

Simple reinforcement models were introduced in this field by Roth and Erev to study learning. In their model, the players have inclinations towards the various strategies, and when a strategy that they play wins the inclination to play that strategy again is reinforced. This model is very simplistic and inadequate because it completely ignores the fact that people are aware that other people are also playing the game, and that people will take their beliefs about what other people will do into account before deciding on their own strategy. We believe however that the motif of reinforcement can be reused to establish not only the inclinations of players towards their strategies but also their beliefs about other player's inclinations as well as their beliefs about other player's beliefs about their own inclinations. Then, their inclinations and their beliefs, and beliefs of beliefs can be combined into making a decision. Stahl has studied the beauty contest game using a reinforcement model that essentially reinforces levels of rationality. While the ideas in Stahl's model carry more truth than the Roth and Erev reinforcement model, one has to wonder where these strategies are coming from. We would like a more general model in which game specific cognition would be introduced by itself, and not ad hoc. In this paper, we will present our proposal for such a model.

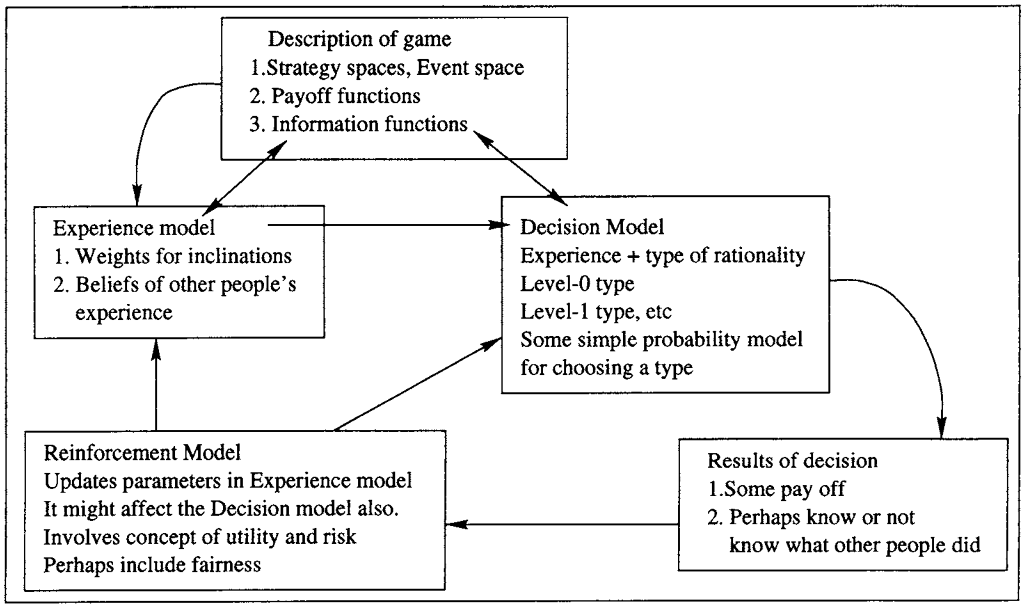

Description of game
1.Strategy spaces, Event space
2. Payoff functions
3. Information functions
Experience model
1. Weights for inclinations
2. Beliefs of other people's experience
Decision Model
Experience + type of rationality
Level-0 type
Level-1 type, etc
Some simple probability model for choosing a type
Reinforcement Model
Updates parameters in Experience model
It might affect the Decision model also.
Involves concept of utility and risk
Perhaps include fairness
Results of decision
1.Some pay off
2. Perhaps know or not know what other people did

Figure 1: An outline of the components that make up our model.

2 Model Overview

Before entering the details, we will give a brief overview of our model. This is necessary because there are many components put together in our model, and each component can be seen as a *submodel* if you will. In figure 1 we sketch out these components and their dependencies on one another.

- **Description of game.** In this model component, we quantify the game in a rigorous manner. This involves writing down the *strategy sets* for each of the players (which in some cases is not quite straightforward, as in ultimatum games), the *payoff functions* which tell what each player earns depending on what happens at each iteration, and finally what I call *information functions.* Information functions reflect an aspect of games that has been overlooked by current research, which we believe is very fundamental to the process of learning. When a player chooses his move and reaps the rewards (or the "pain") of his decision he learns by asking himself "what would have happened if I had done this instead?". In other words he thinks over the strategies he could have chosen and ponders on them. But to actually tell what would have happened had he chosen some other strategy, he needs to know what other players did. [1] In some games, everything is out in the open and everybody knows what everybody else did. In other games, everybody is being secretive and the only clue you have is your payoff, as there is only a certain set of events that would lead you to have such payoff. Finally, there are games where partial information is given. One such game is the beauty contest game we described earlier. In that game, in addition to knowing your payoff, you also know the winning number, which restricts the space of possible events that might have happened even further. The purpose of the information functions is to encode the information players have about what the other players did so

[1]He also needs to know what the payoffs are. In modern game theory experiments, that's a given. Early in the days of behaviourism the subjects didn't know the payoffs, and they were expected to learn how to play same way rats learn how t get food by hitting levers.

that we can incorporate this aspect of the game to the other components of our model.

- **Experience model.** This part of my model is more or less taken from Roth and Erev's ideas about reinforcement models. In Roth's model, the strategies that are tried out are rewarded according to their payoffs. Moreover, old strategies' contributions are gradually dwindled by using a *forgetting factor*. For every iteration, the players have an idea of what worked in the past and how well, which we quantify by assigning weights over the strategy spaces for each player. In addition to these inclinations, players have beliefs about other player's experience. This is the part where our model deviates from Roth. A player that has managed to dupe his opponent by pulling some sort of slick trick will avoid using the trick after a while, because he believes that with every iteration the other player is less and less likely to fall for it. Roth's prediction is that this won't happen. The trickster player will keep exploiting his player and will only stop doing that only when the other player stops falling for it. We think that this is a very simplistic description of human experience, and that in making decisions players consider their beliefs about other player's experience. This makes experience recursive, bc the experience of one player includes beliefs about the experience of other players. And the belief about experience of players involves beliefs about the beliefs of other players about the experience of other players. Of course, this recursion is supposed to stop somewhere. Where it stops determines how rational people are (or paranoid). In our model we will assume 2 levels of rationality, that is we will go as far as beliefs about beliefs of other player's experience. Nothing prohibits us however from going deeper into more and more paranoid levels of rationality. We think that in multiplayer games, 2 levels of rationality are enough. In two-player games however, it may turn out that we need more levels of rationality.

- **Decision model**: The decision model translates player experience into a final assignment of probabilities to the strategies in each players strategy space. We have already talked about levels of rationality. Current research has shown that humans do not have a fixed type of rationality. Stahl has come up with a nice generalized way to think about this. He believes that players are not thinking in terms of the actual strategies they have but instead they think in terms of *decision rules* and they choose a different decision rule every time, tending to choose the ones that work better. [2] A *decision rule* is an algorithm which maps experience to probabilities assigned to the strategies in the player's strategy space. A strategy is then pulled using a randomizing device and these probabilities. Stahl introduced a very specialized model like this, for the beauty contest game, that seems to work pretty well. The problem is that it works because it is specialised and optimized to work only on the beauty contest game. It is not clear how the same model can be applied to other games without generalizing it first. This was the primary motivation for my model. In Stahl, experience is represented by the previous winning numbers and decision rules are gaussians distributions around appropriate centers. And then there are inclinations associated with these decision rules. In our model, our decision rules will be a set of *levels of rationality*. In this paradigm we start out with a naive type of player that has a very simple minded way of making decisions. We call him a *level-0 type* and call his decision rule a *level-0 rule*. With such a starting point a more sophisticated person will assume that everybody else will use that level-0 rule, and will use a more complex decision rule which is meant to best respond to that assumption. We call this a *level-1 rule*. Generally speaking, we can define a *level-n rule* as the decision rule that best responds to the belief that everybody is using the *level* $n-1$ rule. Notice here that the starting point (what we use for a level 0 rule) affects a lot what the higher levels are. Also notice that higher level rules can be built on top of any simple rule we decide to start out with at level-0. This nice generality allows us to introduce many variations to the model, and creates many questions that need to be addressed with intensive research. There are two candidates to level-0that

[2]But why stop here? Why not recurse and say that a player may start having beliefs about other player's rationality as well as experience?! Recursion seems to be the motif of paranoia, and it is interesting to study in the future how much recursion we need to interpret experimental data.

I have come up with. One candidate is Stahl's *level*-(-1) rationality. In this rule, the player will just randomly choose a strategy and experience is completely ignored. A better candidate in my opinion is to use a small variation of Roth's reinforcement model. In this model, players repeat the strategies that are vindicated a posteriori. So the experience of what worked in the past is being put to use here. We argued earlier that this model by itself is inadequate. No smart player will keep tricking a sucker player with the same trick a lot, even if the trick seems to be working, because he wants to stop before the other player learns. However, by adding levels on top of this simple behaviorist paradigm we may get an adequate set of decision rules. With our set of rules set in stone, when the time for the decision comes, the player decides which rule to use. He uses a set of probabilities that he builds up as part of the experience model to make a stochastic choice. Then uses the decision rule to assign probabilities to his strategy space. Then he chooses at random with these probabilities. As you will see, it is very easy to object that this model is too complex and humans can't possibly have all of it in their heads. True. We believe that by randomizing at the end, we will account for mistakes and such failings of the human nature. A combination of a complex model and randomization could be a good recipie. We will have to see how it turns out in practice.

- **Results of Decision.** The main result of a player's decision is the *payoff.* They all get a payoff at the end of every iteration. The rest of the result is lots of thinking about what happened and what strategies got vindicated a posteriori. We need to explain the meaning of the word *vindicate* here. A strategy is vindicated depending on how much payoff the player would get had he played that alternative strategy instead. The more that hypothetical payoff is, the more the strategy is vindicated. Roth's model is has a more simplistic philosophy. It only evaluates the strategy that was actually played. And, on a rather ad-hoc basis, he also has the player think about strategies that "neighbor" the strategy that was actually played. We think that the idea of "neighbor" makes assumptions about the structure of the strategy space, that are not good assumptions to make. The approach of vindicating all strategies eliminates this problem, and seems to be more intuitive. This is the point where the *information functions* we mentioned become useful. In order to claim an expected payoff for your alternative strategies, you need to know what actually happened; what other players did. Roughly speaking what an information function does is take 3 arguments: a player, the event that actually happened, an event that might have happened and return the probability that that player gives to the occurace of the hypothetical event based on the information he has at the end of the iteration. This information is reflected indirectly through the second argument, the event that actually happened. What we can then do is for every hypothetical strategy, find the (vindication) payoff given what other players did and look at the sum of these payoffs over what other players can possibly do, weighted by the information functions. At the end we end up with a number which is the expected payoff for the alternative strategies. Alternatively, as possible future research, we could consider working with probability density functions here instead of stripping the expected payoff from it's distribution and just taking the average. We could set up a utility system in which narrow distributions are prefered to wider. People just prefer certainty over uncertainty. We could we could..., at least it is nice that we are able to say this.

- **Reinforcement model.** This is the model where given the results of the decision, we update our experience. This is a very crucial part of the overall model, because it is the heart of the learning process. Experience consists of our inclinations towards the *decision rules* (i.e. levels of rationality) as well as a wealth of information that is inputed to these decision rules. This information in our model is inclinations that every player has towards his strategies, as well as beliefs about the inclinations of others. We reinforce the strategies based on the vindicated payoff that we described before. In addition to this, we add a Roth term which reinforces just the strategy that was played depending on the payoff that was actually earned. We do this because, there is a difference between hypothetical strategies and the one that was actually played. The hypothetical ones are vindicated based on an *expected* payoff. [3] The one that was actually played is vindicated by an actual payoff which is well-known to the player.

There are two things that one can do with such a model. One is to run simulations in which robots that follow the decision and learning rules of the model play against each other, and see what happens. Another is to try to fit the model to experimental data. That is, we have real people play the game among themselves, and then try to fit the model to the data and see how well it fits compared to other models in the literature.

3 The Gory Details

In the previous section we went through a qualitative overview of the model that we propose. In this section we are going to introduce the mathematical concepts of our model, and will go over the model in thorough detail.

3.1 Description of Game

Roth and Erev went out of their way to say how their simple model was able to capture structural differences between various games. In our discussion the structure of the game is captured by the *strategy spaces*, the *payoff functions* and the *information functions*. In this section we introduce these terms and discuss them in detail:

- **Strategy Spaces.** Let n be the number of players in the game. No-one says that all players have the same strategies available to them, therefore we have as many strategy spaces as we have players. Let $S_1, S_2, \ldots, S_n$ be the strategy spaces of players $1, 2, \ldots, n$ respectively. When all players play a strategy $a_i \in S_i$, the ordered set $(a_0, a_1, \ldots, a_n)$ represents the *event* that took place during that round. We call the space $\mathcal{E}$ of all events in the game the *event space* of the game and we define it as the cartesian product of all the strategy spaces:

$$\mathcal{E} = \prod_{k=1}^{n} S_k$$

For convenience in writing out mathematics, we introduce the following additional notation. We denote the set of integers from 1 to n by

$$[n] = \{1, 2, \ldots, n\}$$

We also introduce the *dual strategy spaces* $\mathcal{C}_i$ which are defined by

$$\mathcal{C}_i = \prod_{k \in [n]-\{i\}} S_k \quad \forall i \in [n]$$

The dual strategy space $\mathcal{C}_i$ is the space of all the things that *the other players* can do; other players as in everybody except player i. From the point of view of player i, S_i is the space of all the things that he can do, $\mathcal{C}_i$ is the space of all the things that everybody else can do, and $\mathcal{E}$ is the space of all the things that can possibly happen.

For example, consider the beauty contest game with 3 players in which each player has to choose a number between 0 and 100. In this case the strategy spaces are all

$$S_1 = S_2 = S_3 = [100]$$

[3] A more accurate representation is in terms of a payoff distribution, but let's start simple

While in this example they are all the same, it doesn't have to be true that they will be the same in all games. The event space in this case is

$$\mathcal{E} = S_1 \times S_2 \times S_3 = [100]^3$$

If player 1 chooses $34 \in S_1$, player 2 chooses $12 \in S_2$ and player 3 chooses $56 \in S_3$, then we say that the event $(34, 12, 56) \in \mathcal{E}$ happened. Moreover, from the point of view of player 1, everybody else played $(*, 12, 56) \in \mathcal{C}_1$. From the point of view of player 2, everybody else played $(12, *, 56) \in \mathcal{C}_2$ and from the point of view of player 3, everybody else played $(34, 12, *) \in \mathcal{C}_3$. Note how in these examples we use $*$ in the place of the missing strategy, in order to distinguish the dual strategies that belong to different dual spaces.

We have then three types of objects to juggle with: *strategies, events* and *dual strategies.* To manipulate these objects mathematically we define the following operations:

- **Merge Operation:** Let $x \in \mathcal{C}_i$ and $y \in S_i$. If $x = (a_1, \ldots, a_{i-1}, *, a_{i+1}, \ldots, a_n)$ then we define $x + y$ by:

  $$x + y = y + x = (a_1, \ldots, a_{i-1}, y, a_{i+1}, \ldots, a_n)$$

  In other words, the + operator is overloaded to combine what a player did and what other players did to an event. In our previous example we would have

  $$34 + (*, 12, 56) = (*, 12, 56) + 34 = (34, 12, 56)$$

  $$12 + (34, *, 56) = (34, *, 56) + 12 = (34, 12, 56)$$

  $$56 + (34, 12, *) = (34, 12, *) + 56 = (34, 12, 56)$$

- **Event to Strategy projection:** As we said S_i is the set of all the strategies that are available to player i. We overload the same symbol to have a different meaning, under a different context. In particular, if $e \in \mathcal{E}$ is an event and $e = (a_1, \ldots, a_i, \ldots, a_n)$, then we define $S_i e$ by

  $$S_i e = S_i(a_1, \ldots, a_i, \ldots, a_n) = a_i$$

  So S_i becomes an operator which when it operates on an event e it returns the strategy that player i played in that event. For example:

  $$S_1(34, 12, 56) = 34 \qquad S_2(34, 12, 56) = 12 \qquad S_3(34, 12, 56) = 56$$

- **Event to dual strategy projection:** In a similar manner, $\mathcal{C}_i$ is overloaded to be the operator that takes an event $e \in \mathcal{E}$ and returns the corresponding dual strategy. In particular,

  $$\mathcal{C}_i e = \mathcal{C}_i(a_1, \ldots, a_i, \ldots, a_n) = (a_1, \ldots, a_{i-1}, *, a_{i+1}, \ldots, a_n)$$

  For example:

  $$\mathcal{C}_1(34, 12, 56) = (*, 12, 56) \qquad \mathcal{C}_2(34, 12, 56) = (34, *, 56) \qquad \mathcal{C}_3(34, 12, 56) = (34, 12, *)$$

- **Dual Strategy to Strategy projection:** Similarly to *event to strategy projection* if $c \in \mathcal{C}_i$ and $i \neq j$, then $S_j c$ is the strategy that player j played in c. Notice that $S_i c$ has no meaning because the dual strategy c contains no information about what player i did.

A nice thing about this notation is that it can be combined to produce more complex operations. One such operation is ***substitution***. Let $e \in \mathcal{E}$ be an event and $x \in S_i$ be a strategy available to player i. Say that player i wants to ask the question: "what would have happened if I had played x instead of whatever I played in e?". The hypothetical event that player i is thinking about is the one in which the i'th strategy in e is substituted by x. This operation we will call ***substitution*** and denote it in terms of a ***substitution function***

$$\sigma_i : (e, x) \in \mathcal{E} \times S_i \longrightarrow e' \in \mathcal{E}$$

The point now is that the σ_i notation is not really needed because we can rewrite the same thing as:

$$\sigma_i(e, x) = x + \mathcal{C}_i e$$

We will prefer the second way of writing things because it cuts down on the parenthesis we have to use, and makes our equations look less cumbersome and more intuitive.

- **Payoff functions:** The structural element of a game is the payoff matrices. In an n-player game it makes more sense to talk about ***payoff functions***. A payoff function is a mapping $\pi_i : \mathcal{E} \to (-\infty, +\infty)$. If all players together play the event $e \in \mathcal{E}$ then player i will win a payoff equal to $\pi_i(e)$. It is reasonable to believe that players are happier when they get higher payoffs. Payoff functions imply that certain strategies are the best responses to what other people do. Let's look at things from the point of view of player i. If he knows that everybody else is going to play $c \in \mathcal{C}_i$, then the best response is to choose a strategy $x \in S_i$ that maximizes his payoff. The set of all such maximal strategies is:

$$\forall c \in \mathcal{C}_i : r_i(c) = \{x \in S_i \mid \forall y \in S_i : \pi_i(x + c) \geq \pi_i(y + c)\}$$

If player i could actually know what c is, his life would have been very simple. Of course, our players don't have this prior knowledge when they are choosing their strategies. So, to make the right choice, they need to think backwards. They need to ask themselves: "if I play this strategy $x \in S_i$, what $c \in \mathcal{C}_i$ should other people do in order for strategy x to give me a maximal payoff?". The set of what we want other people to do, if we are going to play $x \in S_i$ is:

$$\begin{aligned} R_i(x) &= \{c \in \mathcal{C}_i \mid x \in r_i(c)\} \\ &= \{c \in \mathcal{C}_i \mid \forall y \in S_i : \pi_i(x + c) \geq \pi_i(y + c)\} \subseteq \mathcal{C}_i \quad , \forall x \in S_i \end{aligned}$$

Let's look at an example. Consider a 2-player game with strategy spaces:

$$S_1 = \{a, b\} \qquad S_2 = \{a, b\}$$

and payoff

$1 \searrow^2$	a	b
a	2,2	0,1
b	1,0	1,1

This is the traditional way of representing the payoffs in 2-player games in the literature. The column on the left lists the strategies of player 1 and the row at the top lists the strategies of player 2. In this case, they are the same thing. The cells in the table correspond to events and in each cell $e \in \mathcal{E}$ we list the payoffs in the order $\pi_1(e), \pi_2(e)$. The event space in this game is

$$\mathcal{E} = \{(a, a), (a, b), (b, a), (b, b)\}$$

and the dual spaces are:

$$\mathcal{C}_1 = \{(*, a), (*, b)\}$$
$$\mathcal{C}_2 = \{(a, *), (b, *)\}$$

Also, reading the payoff functions off the table, we can write:

$$\pi_1((a,a)) = 2 \quad \pi_1((a,b)) = 0 \quad \pi_1((b,a)) = 1 \quad \pi_1((b,b)) = 1$$

$$\pi_2((a,a)) = 2 \quad \pi_2((a,b)) = 1 \quad \pi_2((b,a)) = 0 \quad \pi_2((b,b)) = 1$$

Finally, the response sets are:

$$\begin{array}{ll} R_1(a) = \{(*, a)\} & R_2(a) = \{(a, *)\} \\ R_1(b) = \{(*, a), (*, b)\} & R_2(b) = \{(a, *), (b, *)\} \end{array}$$

One important thing to note is that the response sets are not disjoint. That is for any two strategies $x, y \in S_i$ it does *not* follow that $R_i(x) \cap R_i(y) = \emptyset$. We will need this fact in the future.

- **Information functions:** The final item that determines the structure of a game is the information that is accessible to players about what the other players do. We, the intellectuals who aspire to mode this game will know everything: what the players think, what they did, what they believe. However, the players themselves don't have to have all this knowledge. This makes learning a little harder. Say that after an iteration, a player wants to look back to his strategies and see what payoff he would have had if he had played a different strategy from the one he did. Let $e \in \mathcal{E}$ be the event that actually happened in the iteration. The dual strategy (i.e. what everybody else did) is $\mathcal{C}_i e$, so the payoff that player i would have earned had he played strategy $x \in S_i$ instead of what he did, is $\pi_i(x + \mathcal{C}_i e)$. What if player i however doesn't know what the other players did? Then, the player can't possibly compute $\pi_i(x + \mathcal{C}_i e)$. Instead, he will need to compute an *expected* hypothetical payoff:

$$\Pi_i(e, x) = \sum_{c \in \mathcal{C}_i} \mathcal{I}_i(e, c) \pi_i(x + c)$$

So, $\Pi_i(e, x)$ is the hypothetical payoff that player i would *expect* to earn if he were to play $x \in S_i$ instead of what he played. On the right hand side we sum over all the things that other players could have done. For each such thing $c \in \mathcal{C}_i$ we weight the corresponding hypothetical payoff $\pi_i(x + c)$ that player i would have had with a probability $\mathcal{I}_i(e, c)$, which corresponds to how likely it is that other players did $c \in \mathcal{C}_i$, *according to player i's opinion!*. That opinion depends on the *information* that is accessible to player i about other people's strategies, and that information depends on what actually happened (the event $e \in cE$). Therefore, the probability $\mathcal{I}_i$ should also depend on e as we indicate in the equation above. This concept of the information function is a very important one, and we feel that it has been overlooked by current research. It captures another aspect of how the game is administered. Usually, when players play a game, they know their payoff, and the know what they did. Sometimes they are told what other players have done, and sometimes they are not. It is clear that depending on the case, learning will take place in a different way; namely if everything is done in the open, players will learn faster. The information functions are meant to capture precisely this, so because of their importance, we state their definition all by itself:

Definition 1 *The function $\mathcal{I}_i : (e, c) \in \mathcal{E} \times \mathcal{C}_i \longrightarrow p \in [0, 1]$ is called an* information function *and it represents the likelyhood that players other than player i played the dual strategy $c \in \mathcal{C}_i$ in the opinion of player i, given the information that he would have if the actual event that happened is $e \in \mathcal{E}$.*

For example, let's look at the information functions for $e = (b, a)$ in the same game as before. To remind, the payoffs for the game are:

$1 \searrow 2$	a	b
a	2,2	0,1
b	1,0	1,1

In addition, let's assume that players only know their payoffs and that they don't know what other players did. Since $e = (b, a) \Longrightarrow \pi_1(e) = 1 \wedge \pi_2(e) = 0$. Player 1 knows $\pi_1(e)$ but doesn't know e. Similarly, player 2 knows $\pi_2(e)$ but doesn't know e either. Now, player 1 knows that having played $S_1 e = b$, the only way for him to have a payoff $\pi_1(e) = 1$ is if the other player plays one of $\{(*, a), (*, b)\}$. It follows from this that

$$\mathcal{I}_1((b,a),(*,a)) = 1/2 \quad \mathcal{I}_1((b,a),(*,b)) = 1/2$$

Similarly, player 2 knows that having played $S_2 e = a$, the only way for him to have a payoff $\pi_2(e) = 0$ is if the other player plays one of $\{(b, *)\}$. Since there is only one way for this to work out, player 2 knows with certainty that player 1 played b and:

$$\mathcal{I}_2((b,a),(a,*)) = 0 \quad \mathcal{I}_2((b,a),(b,*)) = 1$$

Now, let's do a more general case. Say we have an n-player game with payoff functions $\pi_1, \pi_2, \ldots, \pi_n$. Say that during an iteration, the event $e \in \mathcal{E}$ takes place and that the only thing that every player knows is how much money he earned. Look at player i. He has earned $\pi_i(e)$. The set of things that other players could have done then is:

$$A_i(e) = \{c \in \mathcal{C}_i \mid \pi_i(e) = \pi_i(S_i e + c)\}$$

and

$$\begin{aligned} \mathcal{I}_i(e,c) &= \begin{cases} 1/|A_i(e)| & \text{if } c \in A_i(e) \\ 0 & \text{if } c \notin A_i(e) \end{cases} \\ &= \frac{a_i(e,c)}{|A_i(e)|} \end{aligned}$$

where we define

$$a_i(e,c) = \begin{cases} 1 & \text{if } c \in A_i(e) \\ 0 & \text{if } c \notin A_i(e) \end{cases}$$

This example gives us a nice special way to represent the actual information that becomes available to the players when an event $e \in \mathcal{E}$ happens; player i knows that other people did a dual strategy in $A_i(e) \subseteq \mathcal{C}_i$. We will call the set $A_i(e)$ a ***restriction set.*** Restriction sets are ***not*** the most general way to represent information flow in a game. The most general way is through information functions. If you really want to think in terms of "sets", you can use ***fuzzy sets.*** [4] However, in most games that we have to deal with, this special representation is all we really need, including the beauty contest game.

[4] A fuzzy set is a generalisation of our ordinary notion of sets. In ordinary sets, an element either belongs in the set or doesn't. Black and white, in other words. In a fuzzy set, each element is mapped to a degree of membership which is a real number between 0 and 1. In other words, in a fuzzy set, instead of stating membership, we state degree of membership.

Let's say something further about these special cases. Assume that player i has more than one piece of information, and that every piece of information can be represented in terms of a restriction set. Let these restriction sets be $A_i^{(0)}(e), A_i^{(1)}(e), \ldots, A_i^{(m)}(e)$. Also define the membership functions:

$$a_i^{(k)}(e,c) = \begin{cases} 1 & \text{if } c \in A_i^{(k)}(e \\ 0 & \text{if } c \notin A_i^{(k)}(e) \end{cases}$$

Then, all these restrictions can be combined to one restriction by:

$$\begin{aligned} A_i(e) &= \bigcap_{k=1}^{m} A_i^{(k)}(e) \\ a_i(e,c) &= \prod_{k=1}^{m} a_i^{(k)}(e,c) \end{aligned}$$

and write the information function as:

$$\mathcal{I}_i(e,c) = \frac{a_i(e,c)}{A_i(e)}$$

As you will see, we will use this fact later, to compute the information function for he beauty contest game.

3.2 Experience and Decision Models

In their model, Roth and Erev represent the player's learning as a set of "inclinations" (they use the term "propensities") towards the strategies in their strategy space S_i. Players are inclined to play strategies that have worked in the past. Roth tries to incorporate what he calls the *power law of practice* according to which *learning curves tend to be steep initially, and then flatter.* To do this, instead of dealing directly with probabilities, every player is assigning weights $w_i : S_i \to (0, \infty)$ to his strategy space. Then a strategy is chosen stochastically using probabilities given by:

$$p_i(x) = \frac{w_i(x)}{\sum\limits_{y \in S_i} w_i(y)}$$

After the decision is made and the payoffs are earned, the weights are updated to reflect the learned experience. The simplest way to do this is to just add the payoff earned, to the weight of the strategy that was actually played. That is:

$$w_i(x, t+1) = f\, w_i(x,t) + \delta(x, S_i e)\pi_i(e)$$

where the delta-function is defined by:

$$\delta(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

The f factor is called *forgetting factor* and it makes the old reinforcements to the weight to decay and give way to the most recent ones.

The weights $w_i(x)$ and are a simple *experience model* because they represent the internal information that players use to make decisions. The first equation is a simple *decision model* because it combines experience to yield probabilities with which a decision can be made. And the second equation is a simple *reinforcement model* because it says how experience is learned in every iteration. Roth's model is so simple that this distinction may seem pedantic. In our model however, it is very important because each of these three pieces is quite non-trivial. In this section we will discuss the experience and part of the decision model. In the next section we will discuss the decision model in more detail, and in the section after that, the reinforcement model.

In the overview we pointed out that player experience also includes their beliefs about the experience of other players. To take this into account, we are extending Roth's experience model as follows: Players do not only use the weights $w_i(x)$ when they make a decision. They also maintain ***beliefs*** about what other people's weights are. Player i's belief of what player j's weight is, we denote with $w_{ij}(x)$. Player i's belief about what player j believes about the weights that player k uses is $w_{ijk}(x)$. To organize this mess we introduce the concept of ***knowledge contexts***. In simple terms, knowledge contexts are just the subscripts that we put next to the weights. For instance, let's say that players have a silly property that we generally denote with the letter a. For instance, say that a is the IQ of the players. Then, a_i will be the IQ of player i, a_{ij} will be player i's belief of what player j's IQ is, a_{ijk} will be player i's belief of what player j belief is about player k's IQ and so on. In this case, we are applying the knowledge contexts to the weights. In time we will apply them to other concepts also. If you want, you can think of the indices as just indices. It is more constructive however to think of them as mathematical objects in themselves. Single indices,like i,are just integers from $[n]$. Double indices, like ij, can be thought of as ordered pairs (i, j) from $[n] \times [n]$. In general an arbitrary number of indices $i_1 i_2 \dots i_k$ can be thought of as an ordered set $(i_1, i_2, \dots, i_k)$ in $[n]^k$... but not quite. There is a restriction: What does a_{ii} mean? The belief of player i about what a_i is? But player i *knows* what a_i is. In other words, ii and i are the same knowledge context. If we will represent these things with ordered sets, we need to make sure that adjacent elements are not equal. Generally speaking then, a knowledge context is an ordered set of integers from $[n]$ with arbitrary size and such that adjacent elements are not equal. The set of all knowledge contexts is:

$$L_n = \{\emptyset\} \cup \bigcup_{k=1}^{+\infty} \{(i_1, i_2, \dots, i_k) \in [n]^k \mid a_i \neq a_{i+1} \, \forall i \in [k-1]\}$$

For reasons that will become clear soon, we choose to include the empty set $\emptyset$ as an element in L_n.

We say that contexts with only one index (such as i) are *level-0*, contexts with two indices (such as ij) are level-1, etc... We will usually use the letter λ to refer to knowledge contexts, and $|\lambda|$ to denote the level of the context. So, if $\lambda = ijk$, then $|\lambda| = 2$. The first indice of λ indicates the *owner* of the knowledge or belief and we call it the owner of λ and write: own(λ). For example own(ijk) $= i$. Likewise, the last indice refers to the player to whom the associated property refers too. For instance a_{ijk} is player i's belief about player j's belief about a_k; about a property of player k that is. So a_{ijk} discuses a belief about player k, therefore we say that k is the *target* of the knowledge context $\lambda = ijk$, and write trg(λ) $= k$. Level-0 knowledge contexts have the same owner and target. Of course, as the case is with every recursion, we need to terminate it at some point. Usually, we terminate it at level-2, however, in order to be general, let's say that we will terminate at an arbitrary level ℓ. We will denote the restricted set of contexts with level up to ℓ as:

$$L_n(\ell) = \{\emptyset\} \cup \bigcup_{k=1}^{\ell+1} \{(i_1, i_2, \dots, i_k) \in [n]^k \mid a_i \neq a_{i+1} \, \forall i \in [k-1]\}$$

To manipulate knowledge contexts we introduce the following definition: Let $\lambda, \mu \in L_n$ be knowledge contexts. Then $\lambda\mu$ is the combined context in whose indices are the indices of λ followed by the indices of μ. For example:

$$\lambda = iji \wedge \mu = jk \Longrightarrow \lambda\mu = ijijk$$

$$\lambda = iji \wedge \mu = ik \Longrightarrow \lambda\mu = ijik$$

we do this is because in games with negative payoffs there is the danger that the denominator becomes zero. Moreover, setting the weights to zero is basically saying that we are indifferent towards all decision rules, which is nice; in Roth's approach, we can't even set them all equal to zero. Finally, there is the variable μ which controls the pace with which players are learning. Roth controls the same thing by initializing all weights to an arbitrary positive value. The smaller that value, the easier players learn. However, our approach is better because rather than hide this important property in the initialization we have it right out there in the open

When we go through choosing a decision rule and then a strategy, the effective probabilities with which strategies are chosen are:

$$\mathcal{P}_i(x) = \sum_{\rho=1}^{\ell} \Theta_i(\rho)\mathcal{P}_i^{(\rho)}(x)$$

but there is more going on. Not only is a strategy $x \in S_i$ selected. A rule $\rho \in [\ell]$ is selected as well. So, when we reinforce the weights $w_\lambda(x)$ we also reward the decision rule itself depending on how well it worked out. The details of this we will discuss in the reinforcement model, in the next section.

For now, to complete this discussion, we need to tell you how $\mathcal{P}_i^{(\rho)}(x)$ are actually computed. The $\rho = 0$ case is easy. This is the level-0 decision rule in which players tend to repeat what worked in the past, without taking any beliefs into account. This is the type of players who would pull the same trick on their opponent again and again without sitting down to think that if they overdo it, their opponent may figure out how to outsmart them. In math, that rule is:

$$\mathcal{P}_i^{(0)}(x) = \frac{e^{\mu w_i(x)}}{\sum_{y \in S_i} e^{\mu w_i(y)}}$$

Notice that in writing down this equation, we are making an implicit assumption. We are using the same μ we used in the rules for $\Theta_i(\rho)$. So we are assuming that people learn strategy inclinations with the same pace as decision rule inclination.

Given this rule as a base rule, we build the rest of the decision rules in the following way: The $\rho = 1$ decision rule is best responding to the assumption that everybody else uses the $\rho = 0$ rule. The $\rho = 2$ rule is best responding to the assumption that everybody else uses the $\rho = 1$ rule. And in general the ρ-level rule is best responding to the assumption that everybody else is using the $(\rho - 1)$-rule. This mean that the way we will derive the high level decision rules, is actually indepedent of what we use for a 0-level rule. This means that in the future, the decision model can be modified a great deal simply changing the 0-level rule. In the $\rho = 1$ decision rule, the player responds to the assumption that everybody else will use the $\rho = 0$ rule. So, when player i is thinking about one of his strategies $x \in S_i$ he will try to answer the following questions:

- What would other people do if they were all 0-level? That is, what is the probability that they play the dual strategies $c \in \mathcal{C}_i$? We denote this probability with a function $P_i^{(0)} : c \in \mathcal{C}_i \longrightarrow [0, 1]$ which is equal to

$$P_i^{(0)}(c) = \prod_{j \in [n] - \{i\}} \mathcal{P}_{ij}^{(0)}(S_j c)$$

where $\mathcal{P}_{ij}^{(0)}(x) : x \in S_j \longrightarrow [0, 1]$ is the belief of player i of what $\mathcal{P}_j^{(0)}(x)$ is. To compute this belief, we simply take the 0-level decision rule and use the weights in the ij context:

$$\mathcal{P}_{ij}^{(0)}(x) = \frac{e^{\mu w_{ij}(x)}}{\sum_{y \in S_j} e^{\mu w_{ij}(y)}}$$

By the same token, we define:

$$\mathcal{P}^{(0)}_{\lambda j}(x) = \frac{e^{\mu w_{\lambda j}(x)}}{\sum_{y \in S_j} e^{\mu w_{\lambda j}(y)}} \quad \forall \lambda \in L_n(\ell - 1)$$

This is nothing but the 0-level decision rule, where instead of using the experience, we use the appropriate beliefs of experience (beliefs, or beliefs of beliefs, or beliefs of beliefs of beliefs, etc.).

- What are the chances that other players play a dual strategy for which $x \in S_i$ would be a good response? Earlier, we introduced the response set as the set of all dual strategies for which a strategy x is the optimal response.

$$R_i(x) = \{c \in \mathcal{C}_i \mid \forall y \in S_i : \pi_i(x + c) \geq \pi_i(y + c)\}$$

The sensible thing to do is to prefer those strategies $x \in S_i$ for which a dual strategy in $R_i(x)$ is more likely to occur. The probability that such a strategy is chosen is given by the following sum

$$W^{(1)}_i(x) = \sum_{c \in R_i(x)} P^{(0)}_i(c)$$

and it is sensible to prefer x for which this probability is largest. You may be tempted to set $\mathcal{P}^{(1)}_i(x) = W^1_i(x)$ but that wouldn't be quite right. Remember that the response sets $R_i(x)$ are not disjoint. If we sum $W^1_i(x)$ over all $x \in S_i$, some dual strategies will be overcounted and the sum is not going to equal 1. To obtain $\mathcal{P}^{(1)}_i(x)$ then, we normalize the weights by dividing them with that sum:

$$\mathcal{P}^1_i(x) = \frac{W^1_i(x)}{\sum_{y \in S_i} W^{(1)}_i(x)}$$

For a 2-level rules, following a similar line of reasoning we obtain:

$$\mathcal{P}^{(2)}_i(x) = \frac{W^{(2)}_i(x)}{\sum_{y \in S_i} W^{(2)}_i(x)} \quad W^{(2)}_i(x) = \sum_{c \in R_i(x)} P^{(1)}_i(c) \quad P^{(1)}_i(c) = \prod_{j \in [n]-\{i\}} \mathcal{P}^{(1)}_{ij}(S_j c)$$

$$\mathcal{P}^{(1)}_{ij}(x) = \frac{W^{(1)}_{ij}(x)}{\sum_{y \in S_j} W^{(1)}_{ij}(x)} \quad W^{(1)}_{ij}(x) = \sum_{c \in R_j(x)} P^{(0)}_{ij}(c) \quad P^{(0)}_{ij}(c) = \prod_{k \in [n]-\{j\}} \mathcal{P}^{(0)}_{ijk}(S_k c)$$

$$\mathcal{P}^{(0)}_{ijk}(x) = \frac{e^{\mu w_{ijk}(x)}}{\sum_{y \in S_k} e^{\mu w_{ijk}(y)}}$$

Note that the 2-level rule, once it goes through the recursion mess, it all boils down to using the $w_{ijk}(x)$ weights.

In general, a ρ-rule can be obtained by the following recursion:

$$\mathcal{P}^{(\rho)}_{\lambda j}(x) = \frac{W^{(\rho)}_{\lambda j}(x)}{\sum_{y \in S_j} W^{(\rho)}_{\lambda j}(x)}$$

$$W^{(\rho)}_{\lambda j}(x) = \sum_{c \in R_j(x)} \left\{ \prod_{k \in [n]-\{j\}} \mathcal{P}^{(\rho-1)}_{\lambda jk}(S_k c) \right\} \quad , \forall \lambda \in L_n : |\lambda| + \rho \leq \ell$$

We begin the recursion for $\lambda = \emptyset$ and $k = \rho$ and work our way down. When k is reduced all the way to 0 we use:

$$\mathcal{P}^{(0)}_{\lambda j}(x) = \frac{e^{\mu w_{\lambda j}(x)}}{\sum\limits_{y \in S_j} e^{\mu w_{\lambda j}(y)}}, \qquad \text{for } |\lambda| = \rho - 1$$

to terminate the recursion.

Notice that it is not possible to evaluate $\mathcal{P}^{(\rho)}_{\lambda}(x)$ for just any $\lambda \in L_n$ and ρ because after you go through the recursion you may end up with weights $w_\lambda(x)$ with $|\lambda| > \ell$. The way to handle this is to assume that

$$|\lambda| > \ell \Longrightarrow w_\lambda(x) = 0$$

Since with every recurrence, we subtract a unit from ρ and add a unit to $|\lambda|$, and since initially $|\lambda| = 0$ and $\rho = \ell$ it follows that the restriction we need to watch out is:

$$|\lambda| + \rho \leq \ell$$

Obviously, these decision rules are kinda complicated. In fact you may well argue that humans can't possibly be computing these probabilities in their heads. In fact, if they could, a ρ-level player would just go in compute all $\mathcal{P}^{(\rho)}_i(x)$ and choose the $x \in S_i$ that maximizes that probability. Well, they can't, so instead of doing this, the player choose his strategy stochastically using these probabilities; introducing thus the typical errors that humans usually do.

3.3 Reinforcement model

The decision model, uses experience to make a decision. Experience, as we have seen is represented by weights $w_\lambda(x), \forall \lambda \in L_n(\ell)$ and $\theta_i(\rho), \forall i \in L_n(0)$. A decision is represented in part by the event $e \in \mathcal{E}$ which contains the strategies that players actually play in the iteration. The other part of the decision is the choice of decision rules to use. We denote this with a vector $\rho_i, \forall 0 \leq i \leq \ell$. The reinforcement model now, deals with two things:

- Initializing the experience representation
- Updating experience after a decision is made and the payoffs are distributed.

We start with the second part first. To update θs we use the following rule:

$$\theta_i(\rho, t+1) = f\theta_i(\rho, t) + a_i \delta(\rho, \rho_i)\pi_i(e_t)$$

where e_t is the event that happens at time t, $f \in (0,1)$ is the forgetting factor, a is a scaling constant, and $\pi_i(e_t)$ is the actual payoff earned by the player. The purpose fo the forgetting factor is to exponentially decay the reinforcements that happened in the past, and it is supposed to model the fact that people forget what happened before. Another fact that the forgetting factor can model is the fact that people don't always follow the power law of practice in the experimental data. The power law of practice says that learning curves are at first steep and then they flatten out as the weights pile up and become larger and arger. A small f would slow down this flattening of the learning curve appropriately enough to fit the data. In the second term, we are reinforcing the rule that worked with the payoff earned. The other rules are not reinforced at all, but the forgetting factor is applied to them. We choose to make our rationality reinforcement simple because the experience reinforcement will be more complicated.

The strategy weights will follow a rule that has a seemingly similar structure:

$$w_\lambda(x, t+1) = f w_\lambda(x, t) + E_\lambda(e_t, x) \qquad , \forall \lambda \in L_n(\ell)$$

Again, for the same reasons, there is forgetting. The actual reinforcement is in the second term, $E_\lambda(e_t, x)$. For the case $|\lambda| = 0$ we use the following form:

$$E_i(x,t) = b_i\delta(x, S_ie_t)\pi_i(e_t) + \gamma_i[1 - \delta(x, S_ie_t)]\Pi_i(e_t, x)$$

The first term applies to the strategy that was actually played and reinforces it with the payoff that the player actually earned. The second term reinforces the rest of the strategies with the expected payoff that would have been earned had one of those other strategies been played. That payoff we denote with $\Pi_i(e,x)$ and earlier we showed that it can be computed from the information functions by the equation:

$$\Pi_i(e,x) = \sum_{c\in\mathcal{C}_i} \mathcal{I}_i(e,c)\pi_i(x+c)$$

Notice that we are putting different constants in front of the first and second terms. The reason we do this is because it takes a different mode of thinking to reinforce the actual strategy that was played from reinforcing hypothetical strategies that weren't played and players will probably have different inclinations to think in those two different terms.

The effect of this second term is that it will reinforce strategies that look like they might have worked well, had they been chosen instead. This is what seems to be meant by Roth and others, when they talk about reinforcing "neighbourhing" strategies. For example, in the beauty contest game, if 42 won the game, it is quite likely that had the player chosen 43, that would have also won. If we are going to reinforce 42 then, shouldn't we also reinforce 43? I claim that the fundamental reason why it seems that we should is not because 43 is right next to 42 but because it would have also been a winner strategy had it been chosen.

In the cases where $|\lambda| > 0$, there is a slight change in the form of the reinforcement term; namely:

$$E_{i\lambda j}(x,t) = \sum_{c\in\mathcal{C}_j} \gamma_i\pi_j(x+c)\mathcal{I}_{i\lambda j}(e_t,c) \qquad \lambda \in L_n(\ell-2)$$

This reinforcement is an attempt of player i to guess how the other players reinforce their own inclinations and beliefs. It is explicitly assumed of course that our players know the payoff functions of everybody in the game. One thing that is not well known is what other people do which we explained, is addressed by the information functions. In this case however, we want to use the *belief* of what other players information functions are. In other words we need $\mathcal{I}_{ij}$, $\mathcal{I}_{ijk}$ and in general $\mathcal{I}_{i\lambda j}, \forall\lambda \in L_n(\ell-2)$. We will say more about this in detail in a little while. Once we have them, $E_{i\lambda j}$ is just a multiple for $\Pi_{i\lambda j}$ and it is the belief equivalent of the second term in the E_i equation. Notice that we include no belief equivalent for the first term. The reason we do this is because it is a little complicated. Notice that while player i doesn't know the event e_t that actually takes place, he does know what S_ie_t is and uses it in the first term in E_i. Other players will do the same for their first term. Player i however can't know what they are using for their S_je_t, and it turns out to be fairly complicated to accomodate for that. So, we choose to ignore this in our present model.

Next, we discuss the beliefs of information functions. First off, let me note that it may turn out that we are overestimating our players. Whether we should be using beliefs of information functions is something that we need to decide on after studying lots of data. Until we do, it is useful to see how we would go about it, **if** using beliefs of information functions is actually the way to go. Let's start with $\mathcal{I}_{ij}(e,c)$, given $\mathcal{I}_i(e,c)$. Firstly, to avoid confusion notice that

$$\mathcal{I}_{ij} : (e,c) \in \mathcal{E}\times\mathcal{C}_j \longrightarrow p \in [0,1]$$

That is, the c in $\mathcal{I}_{ij}$ is $c \in \mathcal{C}_j$ (**not** $c \in \mathcal{C}_i$). $\mathcal{I}_{ij}(e,c)$ itself is player i's belief of what $\mathcal{I}_j(e,c)$ is. Recall that to compute the information functions player j uses S_je explicitly. [5] We have the same problem as before.

[5] Review the example we gave earlier in this section, to recall

The variable ℓ controls how many levels of rationality we acknowledge in our model. Given ℓ our experience model says that player i experience is represented by the $w_{i\lambda}(x)$,$\forall\lambda \in L_n(\ell-1)$. This is $|L_n(\ell-1)|$ variables. Since we would expect that there is some upper limit to the number of variables a player can keep track of, we should expect that as the number n of players increase, the optimal ℓ that will fit experimental data best should decrease. It is useful then to enumerate the set $L_n(\ell)$:

$$\begin{aligned}
|L_n(\ell)| &= 1+\sum_{k=1}^{\ell+1} |\{(i_1,i_2,\ldots,i_k) \in [n]^k \mid a_i \neq a_{i+1}\, \forall i \in [k-1]\}| \\
&= 1+\sum_{k=1}^{\ell+1} n(n-1)^{k-1} = 1+\sum_{k=1}^{\ell+1}(n-1)^{k-1}+\sum_{k=1}^{\ell+1}(n-1)(n-1)^{k-1} \\
&= 1+\sum_{k=0}^{\ell}(n-1)^k+\sum_{k=1}^{\ell+1}(n-1)^k = \sum_{k=0}^{\ell}(n-1)^k+\sum_{k=0}^{\ell+1}(n-1)^k \\
&= \frac{(n-1)^{\ell+1}-1}{(n-1)-1}+\frac{(n-1)^{\ell+2}-1}{(n-1)-1} = \frac{(n-1)^{\ell+1}+(n-1)^{\ell+2}-2}{n-2} \\
&= \frac{n(n-1)^{\ell+1}-2}{n-2}
\end{aligned}$$

From this enumeration it follows that the weights that a player has to remember for n players and with ℓ-level rationality are $O(n^\ell)$.

For our decision model we follow the footsteps of Stahl: The players select a *decision rule* first, and then they select a strategy using the decision rule. A decision rule is an algorithm that maps a player's experience to a function that assigns probabilities to each strategy in a player's strategy space. We will denote these functions with the symbol $\mathcal{P}_i$ for each player i. Say that we have a model in which there are $\ell+1$ rules in use. Then there are two things to keep track off:

- The probabilities that these rules assign to the strategy spaces:

$$\mathcal{P}_i^{(0)}(x), \mathcal{P}_i^{(1)}(x), \ldots, \mathcal{P}_i^{(\ell)}(x)$$

- The probabilities with which these rules themselves are selected in the first place:

$$\Theta_i(0), \Theta_i(1), \ldots, \Theta_i(\ell)$$

The weights $w_\lambda(x)$ determine the $\mathcal{P}$s. The Θs in turn are determined by corresponding weights $\theta_i(\rho)$ with $0 \leq \rho \leq \ell$. Those weights are another part of our experience model. The dependence of Θs on the weights is part of the decision model. We choose to use the following relation as part of our decision model:

$$\Theta_i(\rho) = \frac{e^{\mu\theta_i(\rho)}}{\sum_{k=1}^{\ell} e^{\mu\theta_i(k)}}$$

This looks like the Roth formula we presented at the beginning doesn't it? Then, you know what we intend to do with the little $\theta_i(\rho)$. They are weights, they represent experience and we actually intend to reinforce them in the same way Roth does. The difference between the w weights and the θ weights is that the w weights refer to the inclinations of players towards their strategies. The θ weights refer to the inclinations of players towards rationality. One difference from Roth is that we have exponentiated the weights. The reason

Player i does not know what $S_j e$ is. We fix it in a similar way we fixed the first term in $E_{i\lambda j}$:

$$\mathcal{I}_{ij}(e,c) = \sum_{x \in S_j} p^*_{ij}(x)\mathcal{I}_j(x + C_j e,\, c)$$

where $p^*_{ij}(x)$ is the probability that player j actually played strategy $x \in S_j$ according to the beliefs of player i. This probability can be obtained by using the probabilities that player i assigns to the dual strategies (namely the $\mathcal{I}_i$ functions); what player j actually did, is in that dual strategy somewhere:

$$p^*_{ij}(x) = \sum_{c \in C_i} \mathcal{I}_i(e,c)\delta(x, S_j c)$$

The delta function filters out from all dual strategies wrt player i, the ones in which player j did $x \in S_j$.

In a similar fashion $\mathcal{I}_{ijk}$ is player i's beliefs about $\mathcal{I}_{jk}$ so:

$$\begin{aligned} \mathcal{I}_{ijk}(e,c) &= \sum_{x \in S_k} p^*_{ijk}(x)\mathcal{I}_{jk}(x + C_k e,\, c) \\ p^*_{ijk}(x) &= \sum_{c \in C_j} \mathcal{I}_{ij}(e,c)\delta(x, S_k c) \end{aligned}$$

Note that $p^*_{ijk}(x)$ is player i's belief about what player j believes about the probability with which that player k chooses $x \in S_k$ Therefore the information function that needs to be used here is $\mathcal{I}_{ij}$ and the delta function will of course filter the $c \in C_j$ for which $x = S_k c$.

Taking this one last step into generality, we obtain the following recursion for the beliefs of information functions on an arbitrary knowledge context.

$$\begin{aligned} \mathcal{I}_{i\lambda j}(e,c) &= \sum_{x \in S_j} p^*_{i\lambda j}(x)\mathcal{I}_{\lambda j}(x + C_j e,\, c) \\ p^*_{\lambda jk}(x) &= \sum_{c \in C_j} \mathcal{I}_{\lambda j}(e,c)\delta(x, S_k c) \qquad \forall \lambda \in L_n(\ell - 2) \end{aligned}$$

With this said, we now know how to reinforce experience after an iteration. The final issue is how do we initialize the weights in the first place? Roth's approach was to ignore this problem. Our approach is as follows. We inherit the $0 \le \theta(\rho) \le \ell$ from previous game experience and we do some work to initialize the w's. An interesting phenomenon observed by Camerer in beauty contest games is *transfer of learning.* The observation here is that players who acquire experience in one game, seem to be applying that experience in a different game that they play next. We speculate that inheriting θs from game to game, will account for it. In fact, θs are the only part of our experience model that is transferable between different games (i.e. indepedent of the structure of the strategy spaces). So, let's say that the θs came from somewhere. We could think of them as parameters of the model that are to be fit to the data.

With the θs known, it follows that $\Theta_i(\rho, 0)$ is also known at time $t = 0$. Now, to initialize the w's observe first that different rationality-level people would think differently about it. Say that $w^{(\rho)}_\lambda(x)$ are the initial values an ρ-level type of thinking would come up with. Then, we initialize $w_\lambda(x, 0)$ by:

$$w_{i\lambda}(x,0) = \sum_{\rho=0}^{\ell} \Theta_i(\rho, 0) w^{(\rho)}_{i\lambda}(x) \qquad \lambda \in L_n(\ell - 1)$$

So what is this thinking? Let's look at it case by case first:

- **Level-0 types**: These guys are completely indifferent towards their strategies, and they believe that other people also think the same way, since it never occurs to them that they may not. It follows then that they will be indifferent to the strategies they choose so they will set:

$$w_i^{(0)}(x) = w_{ij}^{(0)}(x) = w_{ijk}^{(0)}(x) = 0$$

  and in fact:

$$w_\lambda^{(0)}(x) = 0, \forall \lambda \in L_n, |\lambda| \geq 0$$

- **Level-1 types**: These guys believe that everybody else is a level-0 type, so

$$w_{ij}^{(1)}(x) = w_{ijk}^{(1)}(x) = 0$$

  and in fact

$$w_\lambda^{(1)}(x) = 0, \forall \lambda \in L_n, |\lambda| \geq 1$$

  They themselves however are not completely indifferent to their strategies. Depending on the game, if you expect everybody else to make a move at random, you will choose the strategy that is more likely to earn you payoff. Then, we can set up the wights equal to the expected payoff of every strategy on the condition that the other people choose randomly:

$$w_i^{(1)}(x) = \frac{1}{|\mathcal{C}_i|} \sum_{c \in \mathcal{C}_i} \pi_i(x + c)$$

- **Level-2 types**: These guys believe that everybody else is a level-1 type. Since level-1 types would believe then that everybody else is a level-0 type, it follows that

$$w_\lambda^{(2)}(x) = 0, \forall \lambda \in L_n, |\lambda| \geq 2$$

  As for the level-1 types themselves, level-2s would expect them to select the following for initial weights:

$$w_{ij}^{(2)}(x) = \frac{1}{|\mathcal{C}_j|} \sum_{c \in \mathcal{C}_j} \pi_j(x + c)$$

  This is the same equation that we wrote down for $w_i^{(1)}$. Finally, when the level-2 types initialize their own weights they take into account the fact that the other players, being all level-1, will *not* be indifferent in selecting a dual strategy $c \in \mathcal{C}_i$. In fact, earlier (decision model) we wrote an expression for the probability with which a other players would choose one of the dual strategies if they were all level-1:

$$P_i^{(1)}(c) = \prod_{j \in [n]-\{i\}} \mathcal{P}_{ij}^{(1)}(S_j c)$$

  Well, if this is how dual strategies will be chosen, then the expected payoff according to i with respect to his strategies is:

$$w_i^{(2)}(x) = \sum_{c \in \mathcal{C}_i} P_i^{(1)}(c) \pi_i(x + c)$$

The general pattern should be clear now. One rule is that:

$$w_{i\lambda j}^{(\rho)}(x) = w_{\lambda j}^{(\rho-1)}(x)$$

You see this pattern when you compare $w_{ij}^{(2)}(x)$ with $w_i^{(1)}(x)$ in the examples. From this rule, it follows that if you can evaluate $w_\lambda^{(\rho)}(x)$ for the cases where $|\lambda| = 0$ then you can evaluate them in general. In particular, we have:

$$w_{\lambda j}^{(\rho)}(x) = w_j^{(\rho-|\lambda j|)}(x)$$

The level-0 cases can be handled with a recursion given by the following rules. First of all, level-0 types are completely indifferent to their initial strategies, so as we said earlier:

$$w_i^{(0)}(x) = 0$$

A level-$(\rho+1)$ type, assumes that everybody else thinks in terms of level-(ρ), so they assume that the dual strategy is going to be chosen with probability:

$$\wp_i^{(\rho)}(c) = \prod_{j\in[n]-\{i\}} \Omega_i^{(\rho)}(x)$$

where $\Omega_i^{(\rho)}(x)$ are the probabilities that correspond to the $w_i^{(\rho)}(x)$ weights, that is:

$$\Omega_i^{(\rho)}(x) = \frac{e^{\mu w_j^{(\rho)}(x)}}{\sum_{y\in S_j} e^{\mu w_i(y)}}$$

Therefore, they will initialize their weights $w_i^{(\rho+1)}(x)$ by the expected payoff that they would get if they played strategy $x \in S_i$ and everybody else played dual strategy $c \in \mathcal{C}_i$ with probabilities $\wp_i^{(\rho)}(c)$. The weights then will be initialized to

$$\begin{aligned}
w_i^{(\rho+1)}(x) &= \sum_{c\in\mathcal{C}_i} \wp_i^{(\rho)}(c)\pi_i(x+c) \\
&= \sum_{c\in\mathcal{C}_i} \pi_i(x+c)\left\{\prod_{j\in[n]-\{i\}} \Omega_j^{(\rho)}(S_jc)\right\} \\
&= \sum_{c\in\mathcal{C}_i} \pi_i(x+c)\left\{\prod_{j\in[n]-\{i\}} \frac{e^{\mu w_j^{(\rho)}(S_jc)}}{\sum_{y\in S_j} e^{\mu w_j^{(\rho)}(y)}}\right\} \\
&= \frac{\sum_{c\in\mathcal{C}_i} \pi_i(x+c)\left\{\prod_{j\in[n]-\{i\}} e^{\mu w_j^{(\rho)}(S_jc)}\right\}}{\prod_{j\in[n]-\{i\}}\left\{\sum_{y\in S_j} e^{\mu w_j^{(\rho)}(y)}\right\}}
\end{aligned}$$

Putting everything together, the weights are being initialized by

$$w_\lambda(x,0) = \sum_{\rho=|\lambda|}^{\ell} \Theta_i(\rho,0) w_j^{(\rho-|\lambda|)}(x)$$

where

$$i = \mathrm{own}(\lambda) \qquad j = \mathrm{trg}(\lambda)$$

3.4 Concluding thoughts

This initialisation model is by no means the last word on the matter. One important factor that is completely left out is *focal points*. For example, in the beauty contest game, experiments sugget that the number 50 tends to attract players and they select numbers around that number. There is nothing rationally special about choosing number 50 other than the facty that it is right in between the tinterval $[0, 100]$. In that sense we say that 50 is a *focal point* of the game. Modelling focal points is a difficult bussiness and we will just leave them out of our model altogether.

Another factor that has been left out completely from this treatment is *fairness*. We have always assumed that players reinforce their strategies so that they maximize their own profits. However, if a player is in a situation where one strategy will give him some payoff and will hurt a lot the other player, and another strategy will give him a little less payoff without hurting the other player, it is quite possible that he will choose the second strategy, to be *fair*. Experimental evidence suggests that players are more fair in certain games, than we have been lead to believe by traditional game theory. Adding fairness to our model would involve hacking on our reinforcement model. Everything else can be left the way it is. But we can basically switch from reinforcing by payoff to reinforcing by utility which utility would be a combination of payoff and fairness. Rabin describes the detail of such an approach in a paper of his.

Finally, let's say something about *risk*. Suppose that a player is confronted with two options. One in which he wins 100 for sure, and one in which he wins 200 with 0.5 probability and 0 with 0.5 probability. Our approach with reinforcing strategies that weren't played with hypothetical expected payoffs will look at these two options as equivalent. Both have 100 expected payoff. However, it is appearent that they are not the same, and that risk aversive people will prefer the certain 100 than an uncertain 200. Is risk incorporated in our model? It is in an indirect way. Remember that to obtain probabilities from the weights $w_\lambda(x)$, we end up exponentiating the weights. In doing that, the weights are mapped to a curved line. Since we use positive coefficients in exponentiation, the curve is the opposite of a risk aversive curve. In other words, our model makes players who like to take risks. This is probably realistic. Players seem to be attracted by strategies in which cells there are big payoffs, without noticing that they may not win those payoffs. On the other hand, we may add more flexibility to our model and allow people to be risk aversive. This can be done by tweeking the exponentiation of weights to a different function.

4 Example games

In this section, we analyse two games to show how our model is applied. We also show some preliminary results. These games, are the beauty contest game and the El Farol game.

4.1 Beauty contest game

As we have explained in the introduction, in the beauty contest game, players choose a number between 1 and 100. We compute the average of the numbers chosen and multiply the average by a factor p to obtain a *winning number*. The player who chose a number closest to the winning number wins. If two players are equally close to the winning number and also no other player is closer than they are, then these two players share the payoff.

To keep it general, let's say that players are supposed to choose a number between a and b. Then, the strategy spaces for each player are:

$$S_i = \{a, \ldots, b\} \quad \forall i \in [n]$$

The event space is

$$\mathcal{E} = \{a, \ldots, b\}^n$$

and has cardinality $|\mathcal{E}| = (b-a)^n$. This is unfortunate because it says that the size of the event space increases *very fast*! So in this preliminary study we will only be able to state results for small instances of the game. If $e \in \mathcal{E}$ is the event that actually happened, then the winning number is

$$\eta(e) = \frac{p}{n}\sum_{i=1}^{n} S_i e$$

and the set of players that are winners are:

$$\begin{aligned} W(e) &= \{i \in [n] : \forall j \in [n] : |S_i e - w(e)| \leq |S_j e - w(e)|\} \\ &= \{i \in [n] : |S_i e - w(e)| = \min_{j\in[n]} |S_j e - w(e)|\} \end{aligned}$$

The payoff is shared by the players in $W(e)$, therefore it is given by

$$\pi_i(e) = \begin{cases} 1/|W(e)| & \text{if } i \in W(e) \\ 0 & \text{if } i \notin W(e) \end{cases}$$

As for the base information functions $\mathcal{I}_i(e,c)$, they can be written in terms of a restriction set $A_i(e) \subseteq \mathcal{C}_i$:

$$\mathcal{I}_i(e,c) = \frac{a_i(e,c)}{|A_i(e)|} \qquad a_i(e,c) = \begin{cases} 1 & \text{if } c \in A_i(e) \\ 0 & \text{if } c \notin A_i(e) \end{cases}$$

where $A_i(e)$ contains all the dual strategies which combined with player $i's$ strategy, will give the same winning number and the same payoff to player i. That is:

$$A_i(e) = \{c \in \mathcal{C}_i : \eta(S_i e + c) = \eta(e) \wedge \pi_i(S_i e + c) = \pi_i(e)\}$$

4.2 The El Farol coordination game

Suppose that there are some number of people, who like to attend an event on Thursdays. However, if too many people or too few attend, their enjoyment abates. Specifically, there is a utility function $U(k)$ which tells us how good the even is when n players show up. Given this set up, people have two choices: They either attend the event, or they don't. If they don't show up, they earn a penalty payoff p. If they show up, they earn a payoff equal to $U(k)$. In our study, we have been using utility functions of the form:

$$U(k) = A - (B-k)^2$$

This is a parabola that has a peak at $k = B$, and this means that the optimal number of people to show up is B. A controls the points at which this utility function becomes negative. If A is really large, then there will always be a positive utility no matter how many people show up. So, then the optimal strategy would be for everybody to show up. If A is small, then there will be some cases where going to the event will earn the players a negative payoff, and they would have been better off if they hadn't shown up.

We label the strategies in this game with 0 and 1. 0 for *not attend*, 1 for *attend*. This means that

$$S_i = \{0,1\} \Longrightarrow \mathcal{E} = \{0,1\}^n \Longrightarrow |\mathcal{E}| = 2^n$$

Right away we see an advantage of this game over the beauty contest game. The event space grows exponentially still, but at a much slower pace.

Just as in the beauty contest game, a critical piece of information was the winning number, here the critical piece of info is the number of people who actually attended the event. This number we denote $N(e)$ and we can formally define it as:

$$N(e) = \sum_{i=1}^{n} \delta(S_i e, 1)$$

The payoff functions are:

$$\pi_i(e) = \delta(S_i e, 1) U(N(e)) - (1 - \delta(S_i e, 1)) p$$

As for the information functions, we can make two assumptions. One assumption is that the player i knows precisely who showed up and who didn't. Another assumption is that player i only knows how many people showed up. In the first case, the information functions are rather trivial:

$$\mathcal{I}_i(e, c) = \delta(S_i e + c, e)$$

In the second case, they are given by a restriction set $A_i(e)$ by

$$\mathcal{I}_i(e, c) = \frac{a_i(e, c)}{|A_i(e)|} \qquad a_i(e, c) = \begin{cases} 1 & \text{if } c \in A_i(e) \\ 0 & \text{if } c \notin A_i(e) \end{cases}$$

$$A_i(e) = \{c \in \mathcal{C}_i : N(S_i e + c) = N(e)\}$$

We don't have to specify a payoff restriction in this set since, if we don't show up to the event, we earn a fixed penalty. And if we do show up, the payoff we earn only depends on the number of other people that show up.

Usually restriction sets are being enumerated by brute-force counting. We go over the dual strategy space and we count how many dual strategies are in the set. In this game however, it is very straightforward to enumerate the restriction set with a combinatorial argument. Let n be the total number of people, and $N(e)$ be the number of people that actually show up. There are two cases:

- **Case I:** Player i doesn't show up, that is $S_i e = 0$. Then we need $N_i(e)$ out of the other $n-1$ players to show up. There are

$$\binom{n-1}{N(e)}$$

  ways to do this and each such way corresponds to a dual strategy, therefore

$$|A_i(e)| = \binom{n-1}{N(e)}$$

- **Case II:** Player i does show up, that is $S_i e = 1$. Then we need $N_i(e) - 1$ out of the other $n-1$ players to also show up in order to have the right count of players. Likewise, it follows that there are

$$|A_i(e)| = \binom{n-1}{N(e)-1}$$

  ways to do this.

For both cases, we write down the following enumeration equation for the cardinality of the restriction set:

$$|A_i(e)| = \delta(S_i e, 0) \binom{n-1}{N(e)} + \delta(S_i e, 1) \binom{n-1}{N(e)-1}$$

Since we only need the choose factorials for a total of $n-1$ players, we only need to store n values in order to be able to compute the $|A_i(e)|$ for any event. Recall that the choose factorials are defined by

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

However using the recursion

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

is a more efficient and more accurate way of computing them.

Because the El Farol game is a smaller problem, I analysed it first. Note that the model we propose is a fairly complicated one and that it will take a prolonged research to fully explore it and to try to fit it with currently available experimental data. So, for now, we will only state a few preliminary results on the effect of rationality levels and forgeting.

Figure 2 showes the probability with which initial decisions are made. As A increases, so does the probability to attend. Notice however that when players think with rationality level 2 as well, the probability is not strictly increasing with A. This is because for level 2 players there are two competing forces. One force is the fact that higher A means higher payoffs in general. The other force is that higher A means that all the level 1 players will tend to attend, and that will bring the payoffs down. What we see in the plot with rationality level 2 players is the presense of these competing forces.

Figures 3, 4 and 5 show how populations vary as we vary forgetting and rationality. Every pair of graphs corresponds to a different forgetting setting. On the top graph players only act with rationality level 0 or rationality level 1. On the bottom graph, players act with rationality levels 0, 1 and 2 as well.

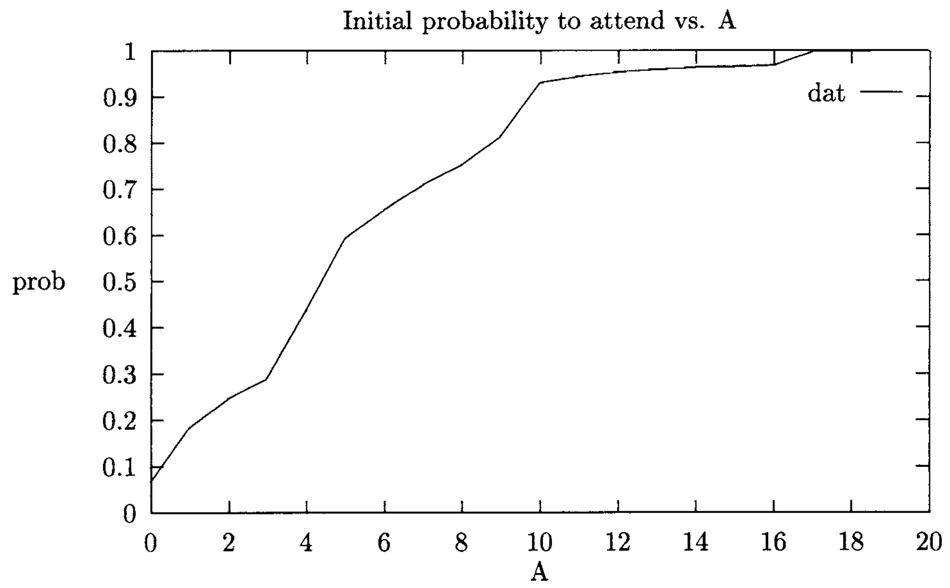
It seems that there is a point of equilibrium at 3 players attending the event. When the forgetting is minimal (i.e. when f is large, as in Figure 5) the population stayes on the equilibrium position for long periods of time. Occasionally people make a different decision and then a perturbation follows, but these perturbations come and go. When the forgetting increases, perturbations happen more frequenty and make the system become more and more chaotic. In addition to forgetting, rationality seems to also play an important role. Without rationality 2 types, the population fluctuations seem to be larger whenever they happen.

Overall then it seems that we have two competing forces in this game. One of them is the force to lock-in: Some players learn to always go and others learn to always not go, and they are happy that way. The other one is the force to forget the lock-in. To do something different! When the first force is dominant, forgetting manifests itself as pulses that shake the population up and down a little bit but leave it some time to converge. When forgetting is dominant, the population behaviour looks like random noise. We can see this effect in the probability plots also (Figures 6 and 7). In fact one interesting thing to see in the figure for $f = 0.75$ is that that particular player is locked in to not attending the event and a few perturbations lock him in to the opposite state.

One could use standard deviation to measure stability. Then one could see the effects of all the model parameters individually and draw mant interesting conclusions.

Probability vs. A graphs

The following graphs show the probability of people to attend the event on their first iteration, as a function of A in the utility equation $U(n) = A - (n-1)^2$. There are 5 players. $\mu = 1$, and the reinforcement parameters $a_i = b_i = \gamma_i = 1$.



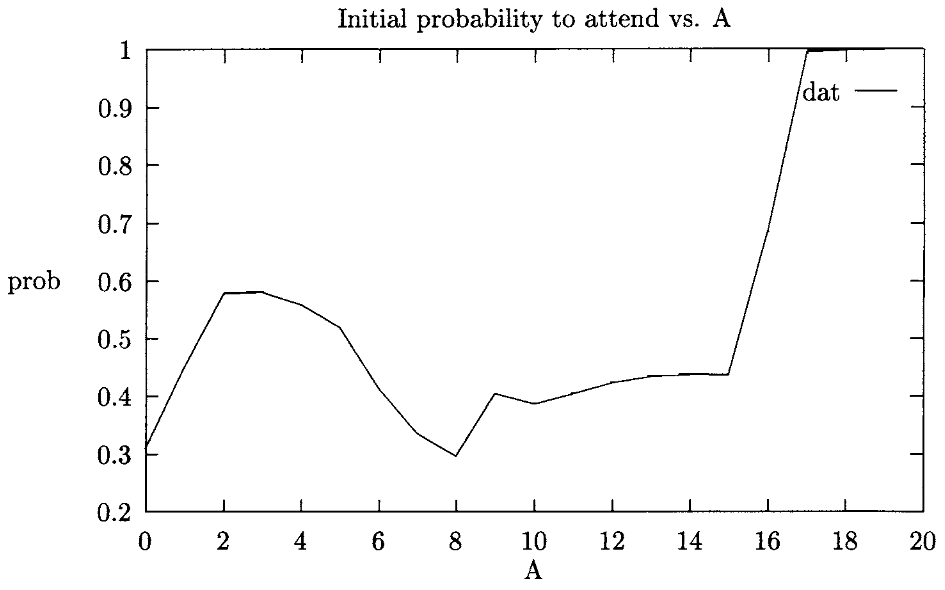
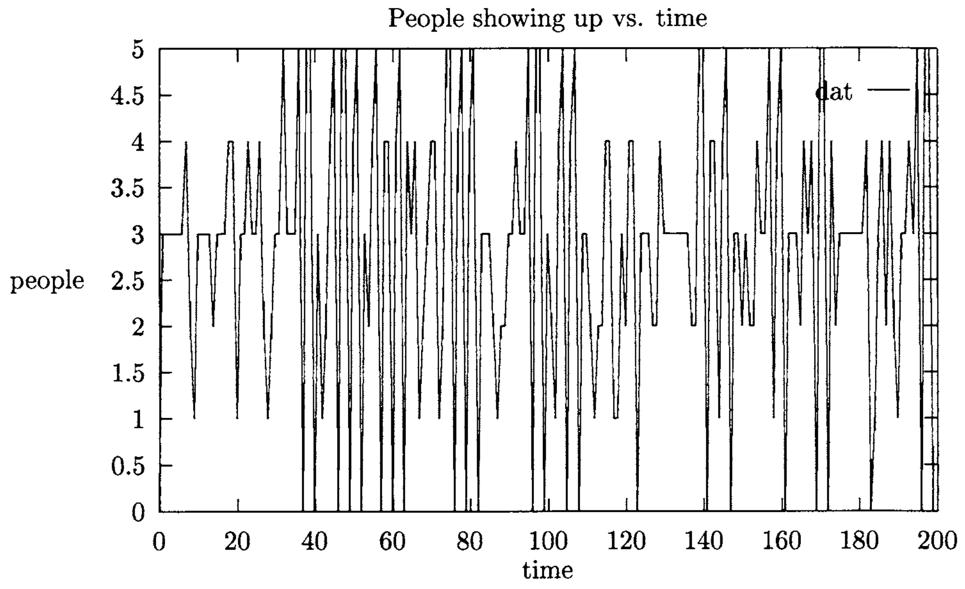


Figure 2:**(up)** Rationality 0 or 1.**(down)** Rationality 0 or 1 or 2

Population Graphs

The following graphs show the population of people that attend the event in each iteration. In all plots $U(n) = 5 - (n-1)^2$, $\mu = 1$, and the reinforcement parameters $a_i = b_i = \gamma_i = 1$. There are 5 players.



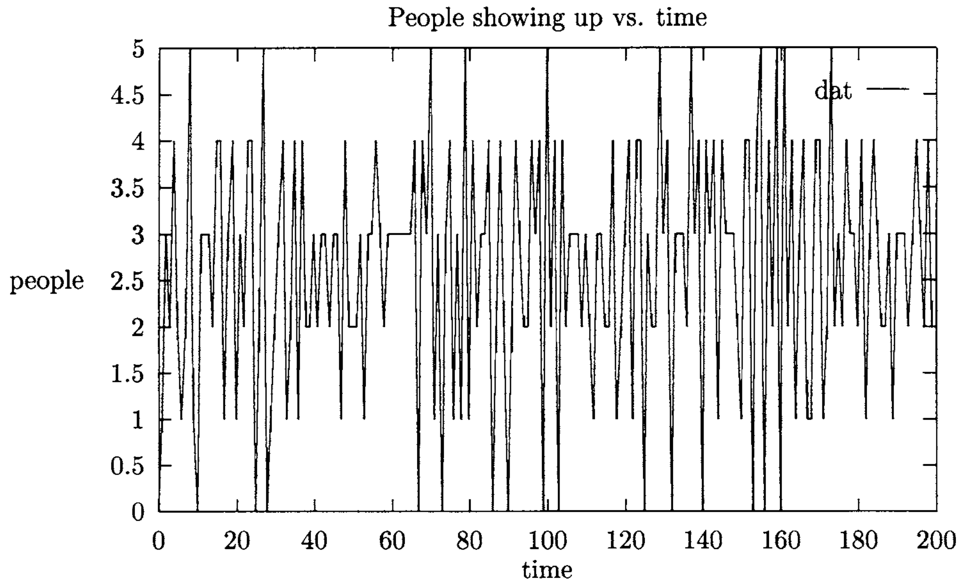
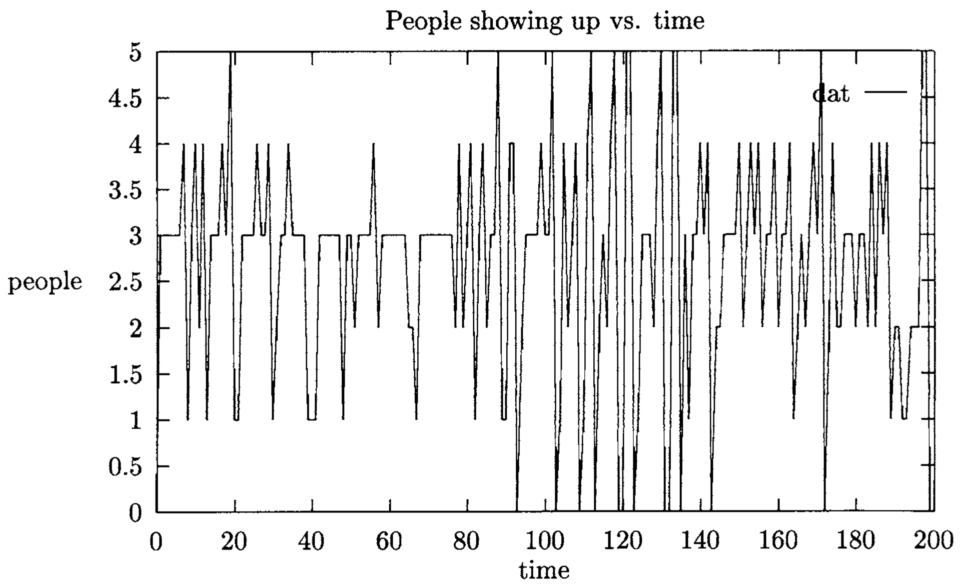


Figure 3:**(up)** Rationality 1**(down)** Rationality 2.. Forgetting 0.25



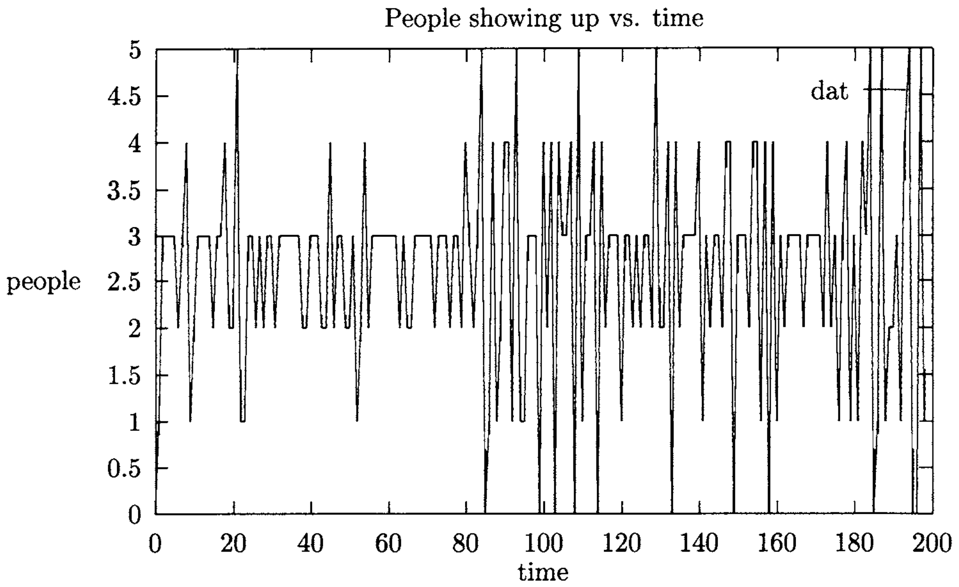
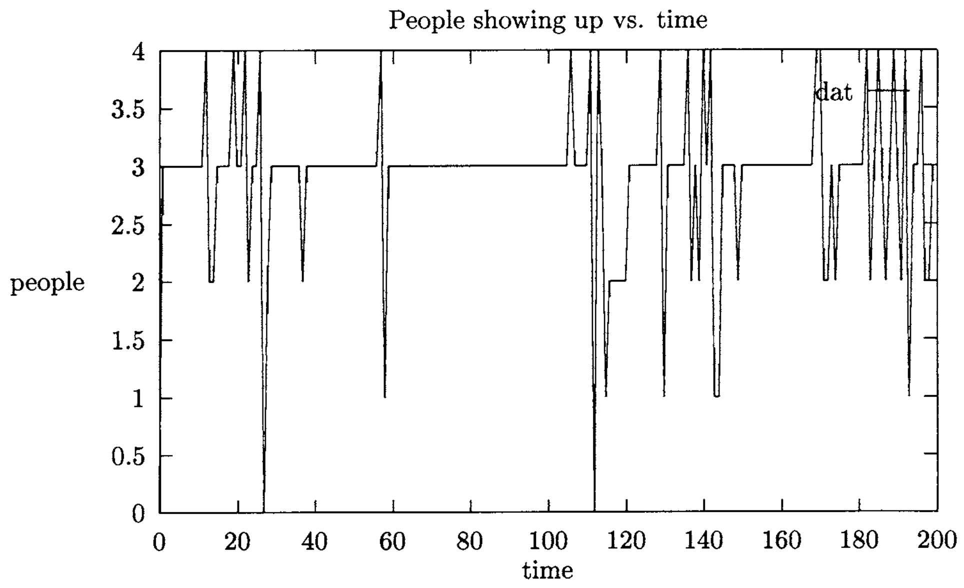


Figure 4:**(up)** Rationality 1**(down)** Rationality 2.. Forgetting 0.5



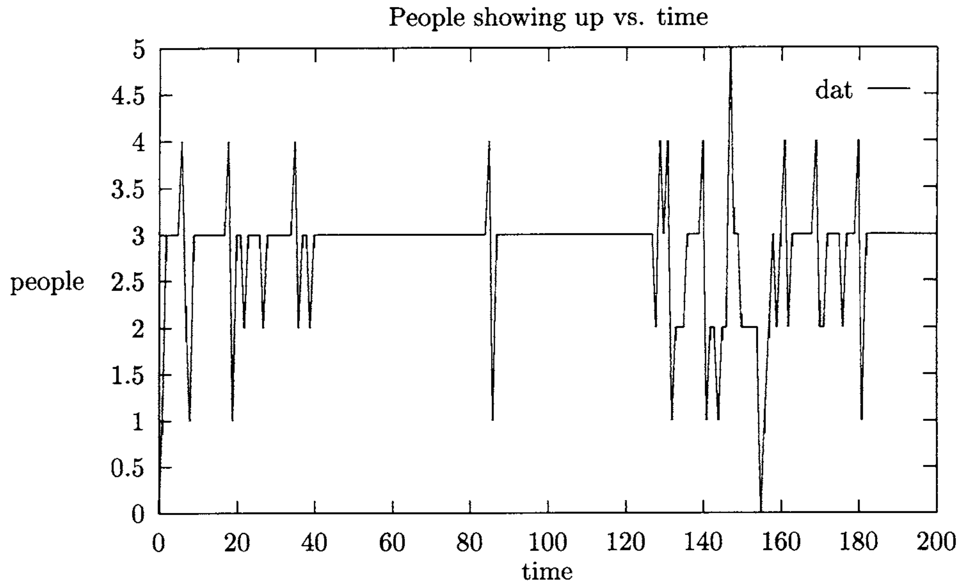
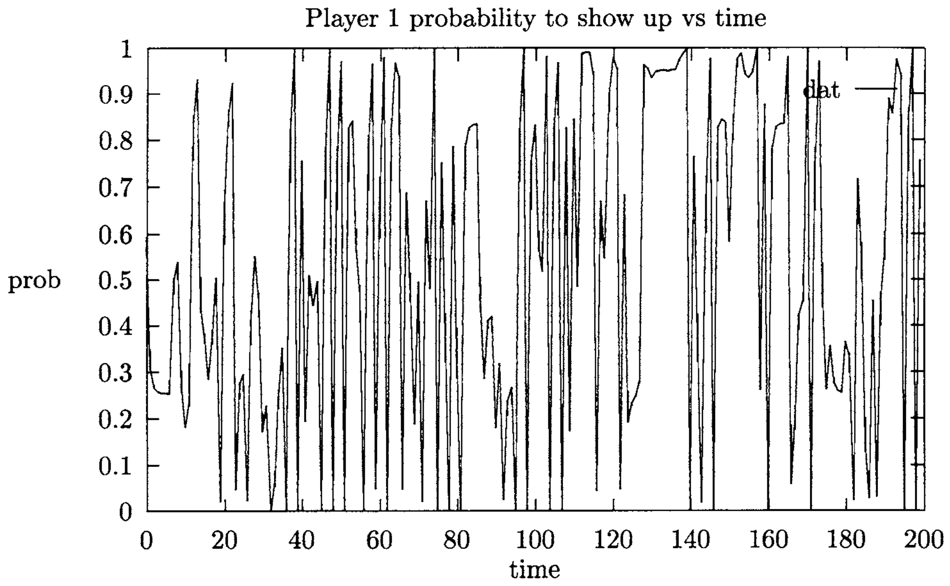


Figure 5:**(up)** Rationality 1**(down)** Rationality 2.. Forgetting 0.75

Decision Probabilities

The following graphs show the probability with which one player made decisions over the course of the simulations. In all plots $U(n) = 5 - (n-1)^2$, $\mu = 1$, and the reinforcement parameters $a_i = b_i = \gamma_i = 1$. There are 5 players. In sequence the plots correspond to forgetting parameters 0.25, 0.50,0.75, 0.9 and players have rationility level 0 or 1.



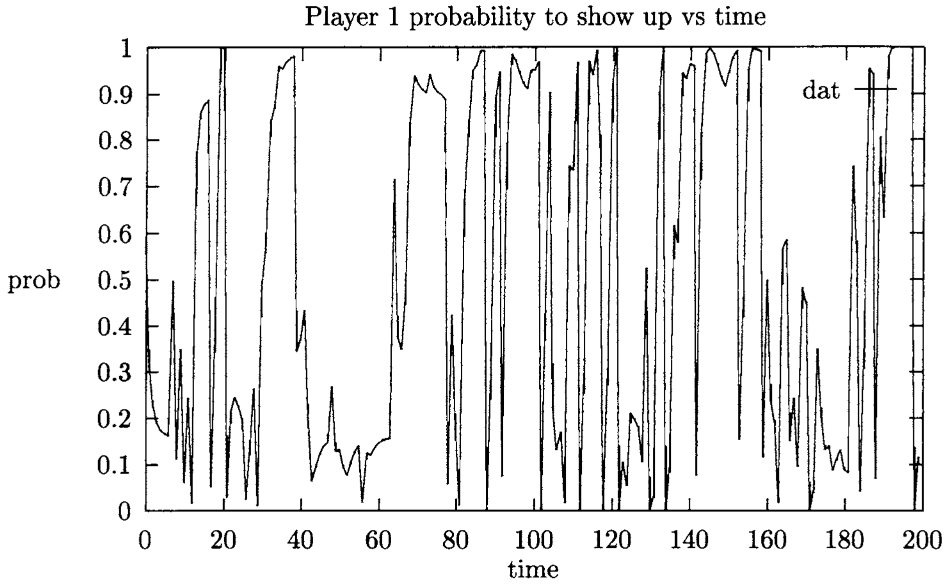
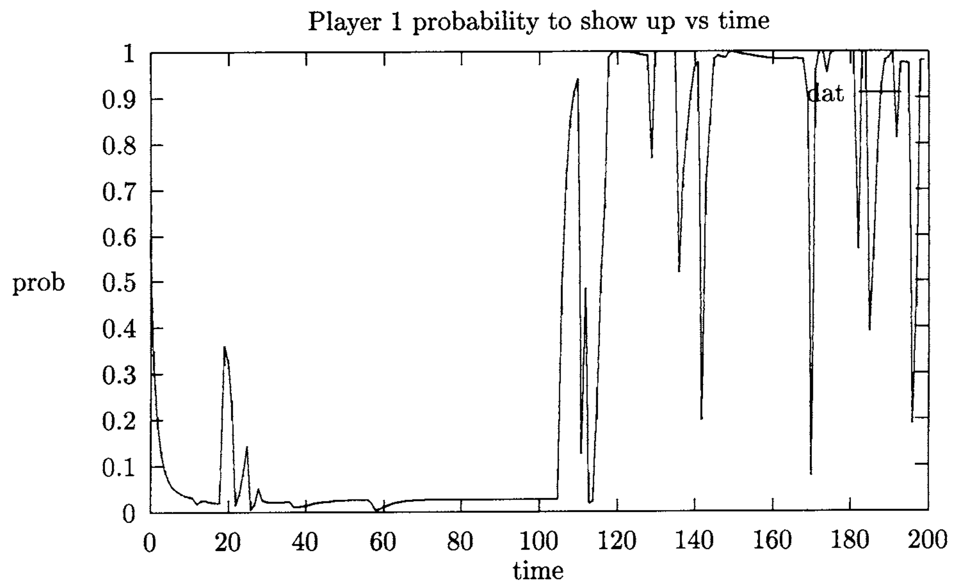


Figure 6:**(up)** Forgetting 0.25**(down)** Forgetting 0.5



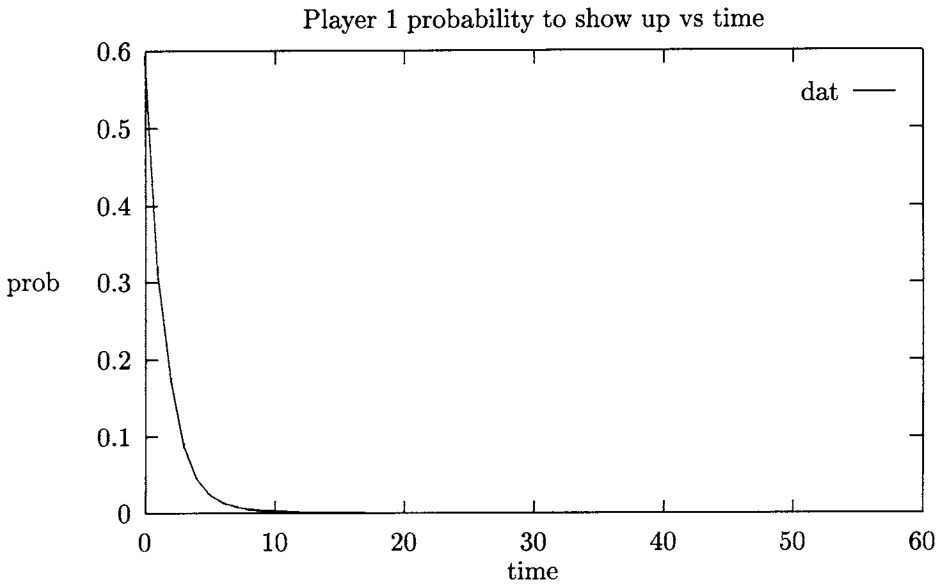


Figure 7:**(up)** Forgetting 0.75**(down)** Forgetting 0.9