

Gossip Based Prediction Market Simulation

Paul Constantine

Wednesday, February 1, 2006

1 Introduction

In this paper I present a software tool for simulating a dynamic parimutuel market (DPM) based on an underlying model for the spread of information through a community. I use a standard gossip model for the information flow through the community. Typically these models are used to model network congestion, but I have taken a more literal interpretation of the “gossip”. The typical scenario for this simulation involves a rumor that spreads through the community; the rumor gives some information about the content of the DPM auction. When an individual hears the rumor, he will place a bet in the DPM in response to the content of the rumor. The prices of the assets in the DPM change according to the selected price function. The simulation tracks both the change in prices of the assets and the spread of the information through the community. At the end, the simulation ranks each of the individuals in order of influence on the community using a PageRank type metric.

In the first section, I present a terse theoretical background on the primary components of the simulation: gossip models, DPM's, and PageRank. Following the theory, I give detailed documentation of the simulation including where to access it and how to use it. In the third section, I give some typical experiments the simulation can perform. And in the final section, I give some preliminary conclusions and ideas for future work and development of the simulation.

2 Background

The first step to understanding the simulation is to understand the theoretical concepts and algorithms that comprise it. Therefore in this section, I give a brief treatment of the gossip models used to simulate the flow of information through the community. Then I summarize the concepts of a dynamic parimutuel market. Finally I touch on PageRank - the algorithm I use to rank the individuals at the end of the simulation.

2.1 Gossip Models

The model I use for simulating the flow of rumors through the community is a standard gossip model. These models are typically used to model network congestion and the information flow in large networks such as the Internet. In the simulation, the rumors act as catalysts for individuals to buy or sell assets of some outcome of the DPM. For example, suppose some individual i holds shares in outcome A of the DPM. In theory, he holds these shares because he has some information which has convinced him that outcome A will occur (more on this in the next section). When the simulation begins, s may call another individual t and pass his information to t . If t does not already know this information, then the new information may incent him to invest in outcome A and perhaps sell off his shares of some other outcome.

2.1.1 Push Algorithm

The gossip models come in two flavors; I analyze each in turn. The first flavor is called the *push algorithm*. Given a set of n individuals, suppose that one individual s knows a rumor. During the first round, s will randomly choose another individual t , call him, and tell him the rumor. At the next round both s and t will each randomly choose another individual and spread the rumor. Note that there is a positive probability

that during this round s and t will call each other and the rumor does not spread. At each round the set of individuals “in the know” each randomly choose another individual to spread the rumor. This process continues until all n individuals know the rumor. It is natural to ask how long it will take for the rumor to spread through the community. This is given in the following proposition.

Proposition 1 *The number of cycles required for a rumor originating from an individual s to spread through a community of n individuals is $\mathcal{O}(\log n)$.*

Proof: This proof is from [1]. We will make use of the Chernoff bounds for a random variable X which is a sum of independent Bernoulli random variables. We state the bounds without proof for $\delta > 5$:

$$\Pr[X > \delta E[X]] \leq e^{-\delta E[X]}, \quad \Pr[X < E[X]/\delta] \leq e^{-\delta E[X]}$$

Now split the spread of the rumor into three phases: During the first phase, the rumor spreads from s to $\log n$ individuals. During the second phase, the rumor spreads from $\log n$ individuals to $n/2$ individuals. And during the final phase, the rumor spreads from $n/2$ individuals to the rest of the community. I analyze spreading time of each phase in turn.

1. Fix an individual v , and let us examine the probability that v will hear the rumor from s in the first $6 \log n$ rounds. The probability that s will call v in each round is $1 - 1/n$, therefore v does not hear from s in the first $6 \log n$ with probability $1 - (1 - 1/n)^{6 \log n}$. Now if X denotes the number of individuals which receive the message after $6 \log n$ rounds, we compute

$$E[X] \geq n \left(1 - \left(1 - \frac{1}{n} \right)^{6 \log n} \right) \geq 6 \log n - \mathcal{O} \left(\frac{\log^2 n}{n} \right) \geq 5 \log n.$$

The second inequality follows from the Taylor expansion in real variables, i.e. $(1 - x)^n \leq 1 - nx + \frac{n^2 x^2}{2}$ when $nx < 1$, and the third inequality holds if n is large enough. Now we apply the Chernoff bound to conclude that the probability that $X > \log n$ is at least $1 - 1/n$.

2. For the second phase, we consider a single round k . Let $T(k)$ be the number of informed individuals at the beginning of this round so that $\log n \leq T(k) \leq n/2$. In this round, the probability that the uninformed individual receives a message is $1 - (1 - 1/n)^{T(k)}$. Then we can compute the expected number of individuals that receive a message for the first time during this round:

$$\begin{aligned} \mathbf{E}[|T(k+1) - T(k)|] &= (n - |T(k)|) \left(1 - \left(1 - \frac{1}{n} \right)^{|T(k)|} \right) \\ &\geq \frac{n}{2} \left(1 - \left(1 - \frac{1}{n} \right)^{|T(k)|} \right) \\ &\geq \frac{|T(k)|}{2} \left(1 - \frac{|T(k)|}{2n} \right) \\ &\geq \frac{3}{8} |T(k)| \end{aligned}$$

where the first and third inequalities follow from $T(k) \leq n/2$, and the second inequality follows again from $(1 - x)^n \leq 1 - xn + \frac{x^2 n^2}{2}$ when $xn < 1$. With this we can choose a sufficiently small constant c such that the Chernoff bounds imply that at least $T(k)/c$ uninformed individuals hear the rumor in round k with probability at least $1 - 1/e^{2T(k)}$. Since $T(k) \geq \log n$, we see that this event happens with probability at least $1 - 1/n^2$.

Now if the number of informed individuals increases by a factor of $1 + 1/c$ during each round, then after $\mathcal{O}(\log n)$ rounds, at least $n/2$ individuals will hear the rumor. The probability that the number of informed individuals grows by such a factor in the first $\mathcal{O}(\log n)$ rounds is at least $1 - \mathcal{O}(\frac{\log n}{n^2})$ which is $1 - 1/n$ for sufficiently large n . Therefore we conclude that this phase ends in $\mathcal{O}(\log n)$ rounds with high probability.

3. We are now in the homestretch; $n/2$ individuals are already informed. With the same trick, we fix an uninformed individual v . Note that probability that v does not receive the message in the next $4 \log n$ rounds is at most $\frac{1}{2^{4 \log n}}$, since in any round at least half the individuals are informed. This quantity is at most $1/n^2$, so applying a union bound for all the uninformed individuals, we see that everyone will hear the message with probability at least $1 - 1/n$ in $4 \log n$ rounds.

Therefore we have shown that with probability $1 - \mathcal{O}(1/n)$ everyone will hear the rumor in $\mathcal{O}(\log n)$ rounds.

It is also possible to show that for the standard push algorithm, the number of messages used in passing the messages through the community is $\mathcal{O}(n \log n)$.

2.1.2 Pull Algorithm

The other flavor of gossip algorithm that I implemented is called the *pull algorithm*. Again, given a set of n individuals, suppose that individual s knows a rumor. In the first round *every* individual randomly chooses another individual and asks “Do you have a rumor?” The set of individuals that chose s in the first round now know the rumor. Note that this set can be empty. This process continues until the rumor has spread through the community. Following a similar three stage proof as above, it is possible to prove the following proposition.

Proposition 2 *The number of rounds required for a rumor to spread through a community of n individual with the pull algorithm is $\mathcal{O}(\log \log n)$.*

One can also show that the number of messages required to spread the rumor is again $\mathcal{O}(n \log n)$.

2.2 Prediction Markets

I built the prediction markets component of my application based entirely on [2]. Therefore in this section, I summarize the motivation, concepts, and price functions in [2]. As in [2], I assume in my analysis that there are two exhaustive and mutually exclusive outcomes A and B in which individuals may trade shares. Denote the total money in A as M_A and similarly the money in B as M_B . Next denote the number of shares outstanding in A as N_A and likewise the number of shares outstanding in B as N_B .

2.2.1 Dynamic Pari-Mutuel Markets

A dynamic pari-mutuel market (DPM) is a hybrid of a standard pari-mutuel market (e.g. wagering at a horse racing track) and a continuous double auction or CDA (e.g. the New York Stock Exchange). The DPM retains advantages of both types of markets: (1) Like the CDA, the DPM allows for the continuous flow of information, similar to the way corporate activity affects stock prices in real time on the NYSE. In this sense, the share prices in the DPM reflect the information in the investor community. (2) Like the standard pari-mutuel market, the DPM poses limited or no risk to the financial institution since it merely redistributes wealth among the participating investors. (3) Similar to the standard pari-mutuel market, the DPM provides infinite liquidity in the market, i.e. traders may trade options in any outcome at any time without waiting for a willing trade partner. In practice a DPM may function as a *prediction market* where traders buy and sell shares in the outcome of a future event. The prices of these shares adjust in real time as investors trade; an individual investor may sell his shares before the event passes to cash in a profit or cut losses, or he may retain the shares until the event passes and then receive the final payoff.

For a living example of a DPM, visit

<http://buzz.research.yahoo.com/bk/>.

2.2.2 Price Functions

A key component of the DPM is the price function, i.e. the way the price of the asset changes as demand changes. In [2], Pennock describes two variants of a DPM which affect the details of the price functions. The variation occurs in how the prize money is distributed to the winning wagers; the difference is subtle. In

the first variant, the initial investment is refunded entirely to the winning bidders, and then the remaining prize money is distributed equally among the winners. Symbolically, the payoff per share at the end of the auction is

$$\mathcal{P}_A = \frac{M_B}{N_A} \quad \text{or} \quad \mathcal{P}_B = \frac{M_A}{N_B}$$

If outcome A occurs, then traders holding shares in A receive their initial investment plus \mathcal{P}_A per share; the case for B is similarly. In the second variant, *all* of the money is redistributed equally to winning bidders, i.e.

$$\mathcal{P}_A = \frac{M_B + M_A}{N_A} \quad \text{or} \quad \mathcal{P}_B = \frac{M_B + M_A}{N_B}.$$

I have only implemented the prices functions derived in the context of the first variant, therefore I summarize those in what follows.

The first price function derives from the reasonable constraint that *the price per share of A is equal to the payoff of per share of B and likewise for B* , i.e.

$$p_A = \mathcal{P}_A \quad p_B = \mathcal{P}_B$$

It suffices to derive the first price function for A since the derivation for B is entirely symmetric. If we let $m(n)$ be the price of purchasing n shares of A then we can write $m(n) = \int_0^n p_A(n) dn$. Substituting these into the constraint equation we have

$$\begin{aligned} p_A &= \mathcal{P}_A \\ \frac{dm}{dn} &= \frac{M_A + m}{N_B} \\ \frac{dm}{M_A + m} &= \frac{dn}{N_B} \\ \int \frac{dm}{M_A + m} &= \int \frac{dn}{N_B} \\ \ln(M_A + m) &= \frac{n}{N_B} + C \\ m(n) &= M_A \left(e^{n/N_B} - 1 \right) \end{aligned}$$

Note that this function requires that the initial quantities N_A and N_B be positive. This can be accomplished with an initial ante from traders.

The second price function is derived from the constraint that *the ratio of the prices must equal the ratio of the money wagered*, or symbolically

$$\frac{p_A}{p_B} = \frac{M_A}{M_B}.$$

Using a slightly different derivation involving market probabilities of A and B , Pennock arrives at the second price function

$$m(n) = M_A \left(e^{2\sqrt{\frac{N_A+n}{N_B}} - 2\sqrt{\frac{N_A}{N_B}}} - 1 \right).$$

These are the two prices functions that I have implemented in my application.

2.3 PageRank

PageRank is the canonical example of an applied eigenvalue problem; I used [3] and [4] as references. In general terms, it ranks the nodes of a given graph according to their ‘‘importance’’. The importance of a given node i is related to the number of edges connected to i . I will explain the PageRank algorithm in its typical context - ranking the pages in the graph of the world wide web where the each page is a node and each link from one page to another is an edge.

To define the importance of a page fairly, we consider its relation to the importance of other pages as well as an average importance given to all pages just for existing. Let x_i be the PageRank of page i . Then we can compute

$$x_i = \alpha \sum_{j \in B_i} \frac{1}{N_j} x_j + \frac{1}{n}$$

where $\alpha < 1$ is a scaling constant, B_i is the neighborhood of node i , N_j is number of outlinks from j to i , and n is the number of nodes in the graph. This formulation generates the following matrix problem: find a vector \mathbf{x} that satisfies

$$\mathbf{x} = \alpha \mathbf{P}^T \mathbf{x} + \frac{1}{n} \mathbf{e}$$

where \mathbf{P} is the adjacency matrix of the directed web graph and \mathbf{e} is an n -vector of 1's. Rearranging the terms of this equation gives us the following linear system.

$$(I - \alpha \mathbf{P}^T) \mathbf{x} = \frac{1}{n} \mathbf{e}$$

Solve this with your favorite solver to get the PageRank vector.

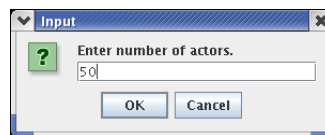
3 Software

In this section I describe the software that I have developed to run the prediction market simulation. I wrote the application entirely in Java. You must have the latest Java Runtime Environment installed in order to use this application. To download and install the JRE, go to <http://java.sun.com>. The application is available as a Java Web Start application at

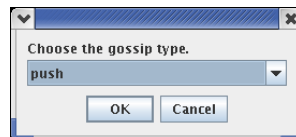
http://ill-conditioned.stanford.edu/GOSSIP_Web/

Click the link to get started.

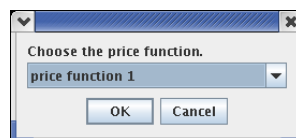
Your browser will ask you if you trust content from me. Click "Yes"; trust me. The application screen should appear. Click 'INITIALIZE' to set the options of the simulation. A small dialogue box will appear asking you to enter the number individuals in the simulation. Please enter less than 500. This is a demo - not a high performance computing tool.



Once you click 'OK', another dialogue box will appear with a drop-down of available gossip algorithms. The algorithm you choose will determine how information flows within the simulation. Choose one and click 'OK'.

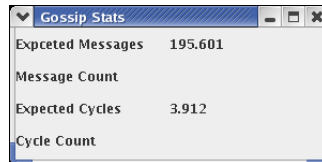


One final dialogue box will appear with a drop-down box of available price functions. Choose price function 1 or price function 2 to be used throughout the simulation.

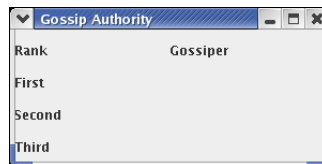


You have now entered all the parameters of the simulation. Each individual is initialized with 10 shares of both outcomes and pays an ante of 20 to enter the simulation. One individual is chosen at random and receives a message that he should sell his shares of A and purchase shares of B . Throughout the simulation, the uninformed individuals are colored red and the informed individuals are colored green. The originator of the rumor retains a green circle around him just for reference purposes.

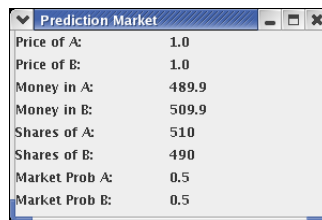
Three more windows will appear with statistics for each of the components of the application. The “Gossip Status” window lists the order of the number of messages passed in the simulation and the order of rounds required for the rumor to spread through the community. This window also has trackers for the actual value for each of these quantities, so keep your eye on this window.



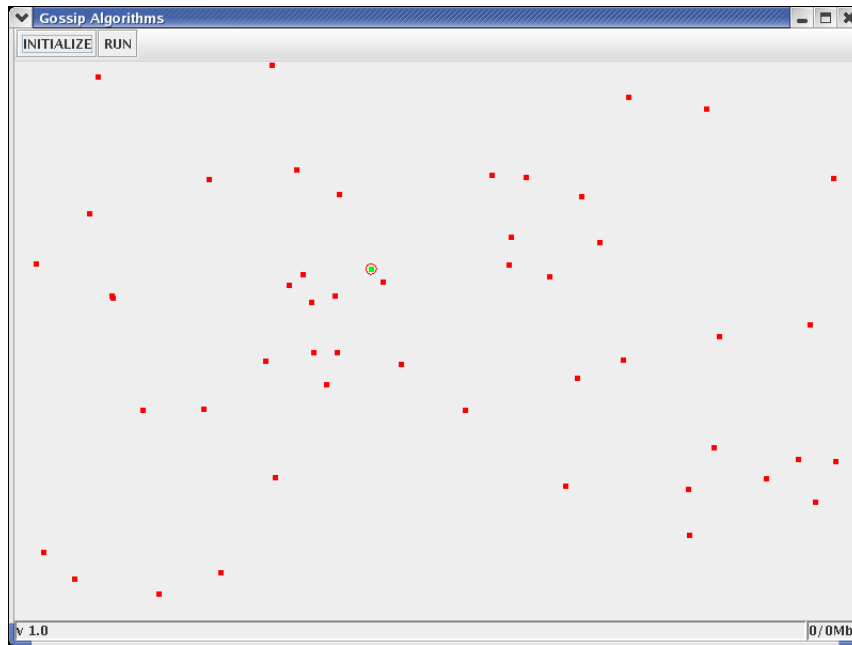
The “Gossip Authority” window appears useless, but it will measure something very important. As the rumor disperses throughout the community, the simulation builds a graph that has a directed edge from node i to node j if j first heard this message from i . Therefore the graph represents which individuals influenced other individuals within the community. With this graph, the simulation computes the PageRank vector for individuals. The top three individuals according to this ranking are listed in the “Gossip Authority” window. If the listed individual was also the originator of the rumor, then an asterisk appears next his listing in the window.



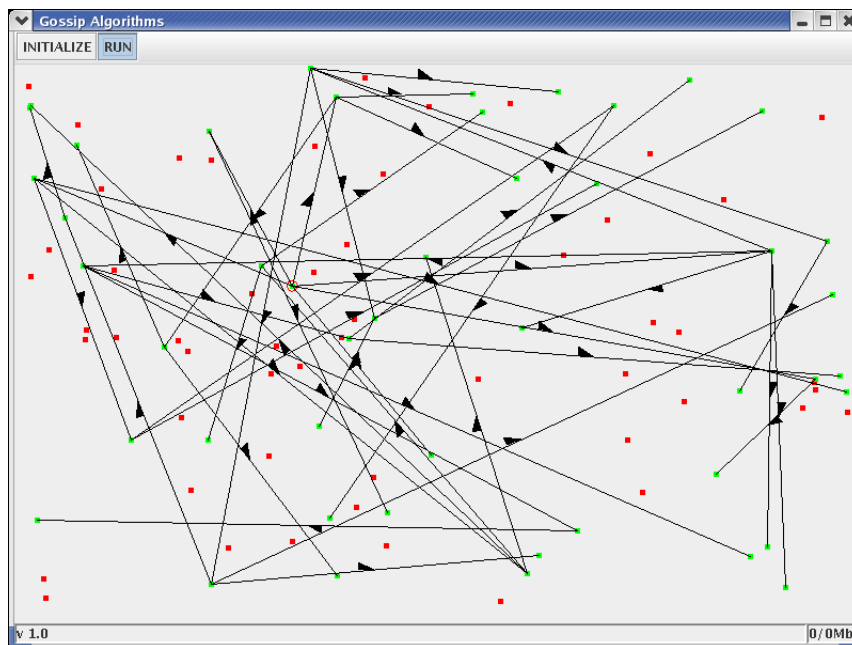
The “Prediction Market” window contains a number of quantities of interest when tracking the prediction market. The quantities are mostly self-explanatory: the prices per share of outcomes A and B , the total money invested in outcomes A and B , the number of outstanding shares in outcomes A and B , and the market probabilities of outcomes A and B . The market probabilities measure the market’s overall perception of the likelihood of events A and B . This is a good measure of the total information in market.



The main window now displays the individuals distributed at random. As the simulation proceeds, this window will visualize the graph of influence based on who spreads the rumor as described above.



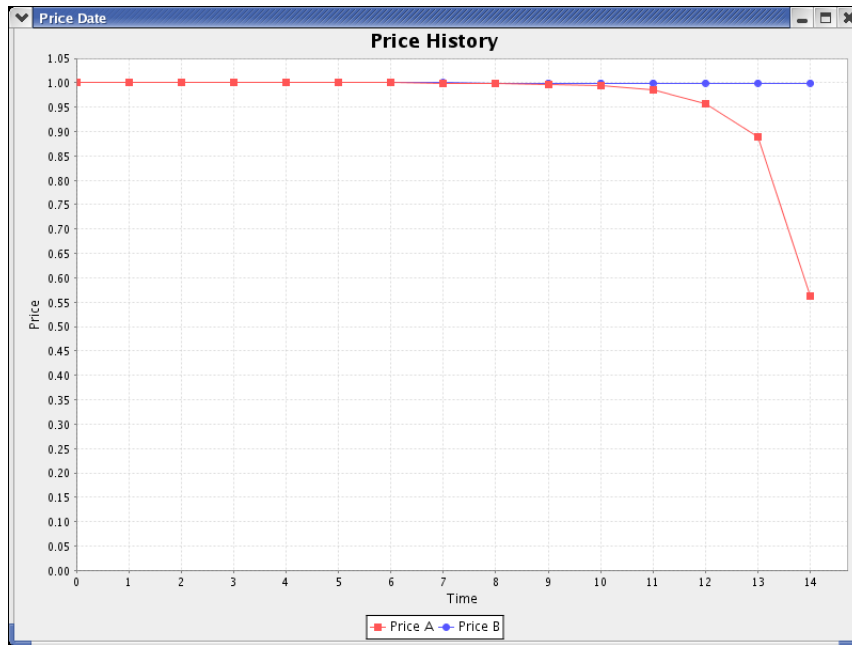
Press 'RUN' to see it go!



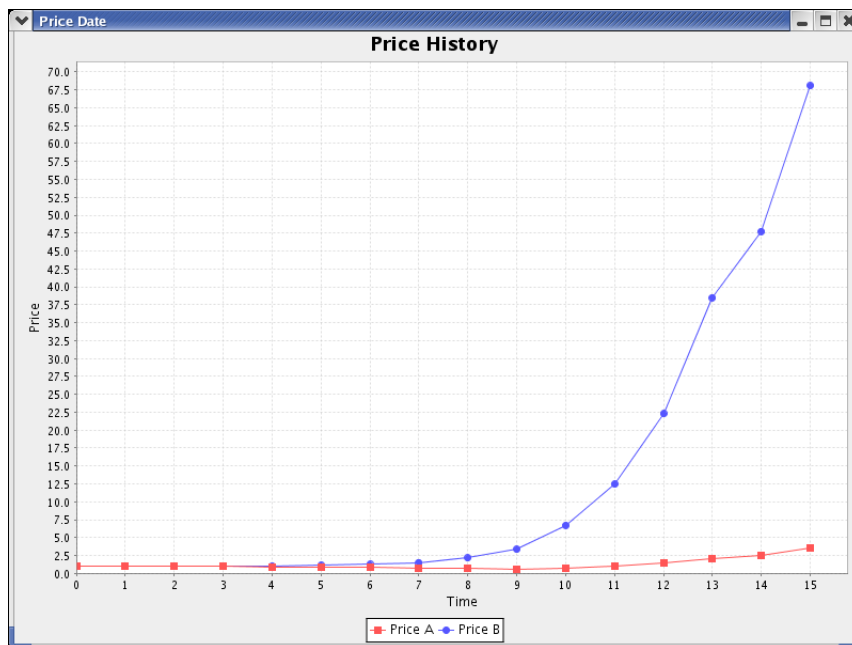
At the end of the simulation, another window will display a chart of the prices per share of outcomes A and B .

4 Experiments

The most interesting comparison with this tool is between the two price functions. I ran the simulation with 500 individuals and the push algorithm varying the price function. The resulting plots are interesting. The next chart shows the first price function.



The next chart shows the second price function.



Notice that in the first plot, the price of the undesirable stock A declines as the simulation continues while the price of B remains relatively constant. But in the second plot, the price of the desirable B increases dramatically while the price of the undesirable stock remains relatively constant. Therefore the second price function creates dramatically more total money in the market than the first price function.

Another interesting experiment is to visually verify the phases described above in the section on gossip algorithms. The total time to spread the rumor once the size of the informed set reaches $n/2$ is noticeably faster in the pull algorithm than the push. The following table lists some results for the number of rounds required for the rumor to spread with the two different gossip algorithms. Here the theory is verified; the pull algorithm outperforms the push algorithm in terms of number of rounds required to spread the rumor.

n	Msgs (Push)	Rounds (Push)	Msgs (Pull)	Rounds (Pull)
10	30	6	50	5
100	511	12	600	6
1000	6072	16	10000	10

5 Conclusions and Future Work

The concepts in this first draft of the simulation are elementary, and the simulation behaved mostly as I expected. But I believe the idea has potential for creating new avenues for exploration in the relatively new field of prediction markets. There are commercial applications for prediction markets as evidenced by companies like Newsfutures. In a subsequent draft of the simulation, I would include much more complexity in both the prediction markets themselves as well as the mechanism for rumor spreading. For example, the simulation should have multiple rumors that start at random times through out the simulation. Also, there are many variants and hybrids of the push and pull algorithms that I could implement.

In the prediction markets, I would of course like to include more outcomes for individuals to bid on, which would make room for a wider variety of rumors spreading. The mother of all enhancements would be a move to designer compound markets, where individuals could customize their own outcomes and bet on them. That would be cool. This would require total reworking of price functions. But the price functions may need to be reworked anyway. The two price functions I have implemented were derived from reasonable constraints, but there may be other reasonable constraints. I think this is one area worth exploring.

The user interface for the application also needs tweaking. There are too many windows and not enough visual distinction between the data; it looks like a bunch of numbers changing randomly. And there are lots more relationships to mine with the nice chart package I used. If you have made it to the end of my paper, I'll give you \$1.

6 Bibliography

1. Kumar, Amit. *CS909 Topics In Algorithms, Lecture 9*, <http://www.cse.iitd.ernet.in/amitk/teach/lec9.ps>.
2. Pennock, David M. *A Dynamic pari-mutuel market for hedging, wagering, and information aggregation*, ACM Conference on Electronic Commerce, May 2004.
3. Gleich, David. *CME200 Linear Algebra with Applications to Engineering, Computing PageRank using the Jacobi Method*, Class lecture notes.
4. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Stanford CS tech report SIDL-WP-1999-0120.
5. Arcaute, E., Is the Best, ICME Press 2005