HANSON'S AUTOMATED MARKET MAKER

Henry Berg and Todd A. Proebsting[*]

I. INTRODUCTION

Abstract

From Hanson's "market scoring rule," we derive all the necessary formulae to implement a corresponding automated market maker for a prediction market. The market maker has many desirable qualities and always stands ready to trade, thus providing liquidity to markets. The formulae cover all transactions for buying and selling market contracts. In addition, we address practical concerns like how to correctly treat rounding errors and how to prevent errors that allow traders to cheat the market, and provide a practical numerical example. We have used Hanson's automated market maker to run many markets at Microsoft.

Motivation

Prediction markets usefully aggregate individual predictions into simple prices. Many studies demonstrate their accuracy in diverse applications, from predicting printer sales to predicting outcomes of political elections.

A prediction market works by creating contingent securities that represent the mutually exclusive possible outcomes of a future event. In its simplest form, the contingent security is worth some set amount if the outcome is realized, and it is worthless otherwise. For instance, a security contingent on the Green Party winning the next election might be worth $1 if the Greens win, but would be worthless if they lose.

Once the contingent securities have been created, market participants are free to trade those securities amongst themselves for some currency. Security prices reflect predictions of the likelihood of the contingent event—high prices indicate high estimated probability and low prices indicate low probability.

Two popular mechanisms exist for matching buyers and sellers in prediction markets: continuous double auctions (CDA) and automated market makers (AMM). In a CDA, the market maintains a list of Bids and Asks for each security. The Bids represent individual commitments to buy some number of shares of the security at given prices; Asks represent commitments to sell shares at a given price. Anybody wishing to purchase shares would

[*] Microsoft Corporation (henryb@microsoft.com, toddpro@microsoft.com)

check the lowest Ask price and either accept that price, or post a Bid to buy a lower price. In thinly traded markets, the spread between the lowest Ask and the highest Bid can be great—often too great to encourage trading. The complexity of the Bid/Ask process can also discourage traders from entering the market.

Market makers are traders that offer to buy or sell shares at prices with small spread. Market makers can exist with or without a CDA behind the scenes. Humans can set market maker prices, or computer algorithms can drive pricing. In either case, changes in trading behavior will drive market maker prices higher or lower.

An AMM allows traders to place all orders with the AMM, and the AMM determines the cost of each transaction and adjusts the prices of the securities. AMMs present many advantages to those wishing to implement a prediction market. An AMM can stand ready to trade at all hours, it can react instantaneously to changes in trading, and it can eliminate the unfortunate spreads found in thinly traded markets. To be useful, an AMM should have the following properties: the AMM algorithm should not be vulnerable to becoming a money pump to a clever trader; the AMM's potential losses should be bounded ahead of time; and the AMM should have tunable properties (e.g., how will a $1,000 bet affect prices?).

Fortunately, Robin Hanson invented the basis for an automated market maker that meets all these requirements when he invented "market scoring rules" (Robin Hanson, "Combinatorial Information Market Design" [2003] *Information Systems Frontiers 5:1* at pp107-119. Available at http://hanson.gmu.edu.).

There are many organizations that offer prediction markets both to the broader web community and privately. Some of the ones offering automated market makers of various kinds include:

Consensus Point (http://www.consensuspoint.com/)
Inkling Incorporated (http://inklingmarkets.com/)
Hollywood Stock Exchange (http://www.hsx.com/)
Hubdub (http://www.hubdub.com/)
Media Predict (http://mediapredict.com/)
Nosco (http://www.nosco.dk/)
Popular Science Predictions Exchange (PPX) (http://ppx.popsci.com/)
ProTrade (http://www.protrade.com/)
Qmarkets (http://www.qmarkets.net/)
Shuugi.in (http://shuugi.in/)
Spigit (http://www.spigit.com/)
Washington Stock Exchange (http://www.thewsx.com/)

Xpree Inc (http://www.xpree.com/)
Zocalo (http://zocalo.sourceforge.net/)

We go on to describe our internal implementation of Hanson's market scoring rule.

II. AUTOMATED MARKET MAKER FRAMEWORK

Hanson's Market Scoring Rule

Hanson's market scoring rule is based fundamentally on "proper scoring rules", which create an incentive-compatible way to reward accurate predictions of future events. Predictions take the form of probability estimates, and higher correct estimates are rewarded more generously than lower correct estimates. There are many proper scoring rules, all of which have the property that a rational estimator would maximize his expected return by revealing his true beliefs. This requires the score received to increase as the estimate of probability of the outcome (r_i) increases, such that the maximum possible score is received by truthfully revealing the probabilities of each outcome.

Hanson's market maker derives from the logarithmic scoring rule (Hanson 2003, p. 109):

$$score_i = b \log (r_i)$$

Here, r_i represents an estimate that event "i" will occur, and $score_i$ represents the score if that event happens. b represents a simple scaling factor. The highest possible score is 0 (when something is predicted with certainty $r_i = 1$), and the scores can be arbitrarily lower. Moreover, the logarithmic scoring rule is local, so the score of a given prediction can be computed based only on that prediction, independent of the predictions of competing events.

An Automated Market Maker

This scoring rule is easily turned into an automated market maker. A market consists of a set of mutually exclusive outcomes. Shares of the outcome that actually occurs are worth 1, all other shares are worth 0. The market maker sets the price of a given security based on the net amount of all securities outstanding. The more shares of an outcome that are outstanding, the higher the price of that outcome. If we assume that all securities start at the same price, and if we let s_i be the net amount of a given security that has been sold, then the price of security i is:

$$P_i = \frac{e^{s_i/b}}{\sum_k e^{s_k/b}}$$

It's trivial to see that these prices sum to 1, which we would require to eliminate arbitrage opportunities. From this formula, we can derive formulae for any transaction that the market maker might want to support: buying or selling securities, determining how much the price will move after a given transaction, etc. The market maker has many desirable qualities:

- The market maker stands ready to buy or sell an unlimited amount of any security, although the price of that security may be driven vanishingly close to 0 or 1.
- The market maker only risks losing a bounded amount of money. (All market makers risk losing money since they stand ready to buy or sell any security and only one of those transactions will make money.)
- The market maker cannot be turned into a money pump through clever sequencing of Buy and Sell transactions.

Unfortunately, creating a market maker from the price formula is not as simple as it might appear. That is because the prices change continuously as shares are traded—every fraction of a share that changes hands affects the price of the next fraction of a share.

The rest of this paper presents commonly needed formulae for implementing the market maker implied by this formula.

III. PRACTICAL FORMULAE FOR HANSON'S AMM

Formulae

The following symbols will be used for the various formulae:

a. s The stock vector. s_i represents the market's holdings of security i.

b. r The vector of "reports" (i.e., estimates) of a participant.

c. b The elasticity constant for the market. The greater b is, the less market prices change with each security purchase.

d. P The vector of prices of the securities prior to a transaction. P_i represents the price of security i. Prices are between 0 and 1, exclusive.

e. P' The vector of resulting prices of the securities after a transaction.
f. Q The quantity of a transaction (number of shares).
g. K The total cost of a transaction (amount of virtual currency).

Setting Initial Prices

This price formula above differs slightly from Hanson's by dropping the "a_i" offsets from the exponents. The purpose of those offsets is to set the initial prices of the securities. A simpler, equivalent way to set initial prices is to have the market patron make an initial "purchase" of securities that will drive the prices to the desired levels. This can be done with:

$$s_i = b\log(P_i) + X$$

where P_i are the desired initial prices, and X is any arbitrary constant. It is simpler in practice to start with no shares outstanding (s_i = 0), and all prices set equally. This creates an advantage for early traders, which in practice may be a desirable incentive to give traders to participate early. A third option is to allow traders to auction the right to enter the market first, or otherwise select preferred traders to fill this role.

Simple Transactions

What is the total cost of a transaction that would move the price of a security from P to P'?	$K = -b\log\frac{1-P}{1-P'}$
What is the total cost of buying Q shares of a security whose initial price is P?	$K = -b\log\left(P(e^{Q/b}-1)+1\right)$
How many shares of a security must be bought/sold to move the price from P to P'?	$Q = b\log\frac{P'(1-P)}{P(1-P')}$
How many shares of security can be bought/sold for a total cost of K?	$Q = b\log\left(\frac{e^{\frac{-K}{b}}-1}{P}+1\right)$
What will be the resulting price of a security after buying/selling Q shares of that security with an initial price of P?	$P' = \frac{1}{1+\frac{1/P-1}{e^{Q/b}}}$
What will be the result price of a security with an initial price of P after a transaction with a total cost of K?	$P' = 1-\frac{1-P}{e^{-K/b}}$

The table above relates how prices, quantities and total cost are related in simple transactions. All signs are from the perspective of the market trader (units of currency and shares of stock received by the trader are positive, units of currency and shares of stock given out by the trader are negative).

Loss Limit

Market makers can lose money. Imagine that the correct security's original price is P and that traders buy as much of that security as they can and nothing else. The market maker would have to pay off each security for $1, but would have sold each for less than $1. The total difference represents the market maker's loss. The loss limit is:

$$L = b \log P$$

In some prediction markets, there is a set amount of money available for trading (*K)*. In this case, the market maker's loss limit is:

$$L = Q - K$$

$$= b \log \left(\frac{e^{\frac{-K}{b}} - 1}{P} + 1 \right) - K$$

The greatest risk to the market maker derives from the lowest priced security initially.

Composite Bets

This market maker conveniently supports buying equal amounts of competing securities in a given transaction. To do this, simply use the combined prices of the individual securities as P in the preceding formulae. Of course, when shares of multiple securities are bought, the stock vector, s, must be updated for the appropriate constituents. If we let E be the set of securities that are to be bought together in equal amounts, then:

$$P_E = \frac{\sum_{i \in E} e^{s_i/b}}{\sum_k e^{s_k/b}}$$

Conditional Bets

This market maker conveniently supports conditional bets of the form, "if any event in W occurs then I win, if any event in L occurs then I lose, but if anything else (C) happens refund my money." Effectively, rather than winning and losing outcomes there are also neutral outcomes. These bets are conditional in the sense that they are of the form "I bet that something in W happens given that it was something in $W \cup L$ that happens."

To create such a conditional bet, it is necessary to have a transaction that includes shares of securities in W to pay off a correct prediction, and shares of securities in C that will return the cost of the whole bet in case something outside of $W \cup L$ happens. To buy Q shares of this conditional bet, you must buy Q shares of W and just enough C shares to refund the total cost, K. In other words, there must be exactly K shares of C, so that if C ends up including the correct outcome, the trader receives just enough of a benefit to offset his initial cost K.

Fortunately, it is easy to determine the price for this conditional bundle of shares. Use the standard formulae above with the following price:

$$P_{W|W \cup L} = \frac{\sum_{i \in W} e^{s_i/b}}{\sum_{k \in W \cup L} e^{s_k/b}}$$

*How to Set **b**?*

The elasticity constant b controls how much prices change for a given transaction size (measured in shares or cost). Setting b is a vexing problem: set too low, the market prices will swing wildly on any trade, and set too high, the market may not move enough reasonably reflect aggregate opinions.

The simplest rule of thumb is to determine how large a bet should move the market to a given price. For instance, if \$1,000 is bet on some security and nothing is bet on any other, it may be desirable for the price of that security to move to \$0.99. (I.e., a \$1,000 bet would only be made if the bettor had 99% confidence in the underlying event.) This leads to:

$$b = \frac{-K}{\log \frac{1-P}{1-P'}}$$

Note that the original price, P, for this computation is typically the starting price for all N securities, which is $1/N$.

*Adjusting **b** When the Market is Open*

For some markets, it is impossible to know how much money will be wagered, which makes it difficult to pick an appropriate amount of elasticity that works throughout. In this case, it is helpful to be able to pick a small b (high elasticity) and then adjust as more money is brought into the market.

Adjusting b presents two challenges. First, changing b without changing anything else will change all of the prices. (Recall that prices are a function of the stock vector and b.) Second, the elasticity affects market maker risk—increasing b will increase the loss limit for the market maker.

To change b without affecting market prices, the market maker must purchase enough of each security to maintain the pre-change prices. This can be accomplished using the same formula used to set initial prices:

$$s_i = b\log(P_i) + X$$

With this formula, simply set X to minimize the number of shares that the market maker must buy, and then buy the shares needed.

The new risk of market maker losses is determined by the loss limit at the current market prices minus the market maker's net balance from previous sales.

IV. PRACTICAL CONSIDERATIONS BEYOND MATH

While the formulae that drive the market maker are essential to any implementation, there are other practical concerns.

Stock Vectors

Any market maker implementation must keep an accounting of the state of the market. This accounting is simple:

- The market maker must keep track of its net position in each security. I.e., how many shares of each security are outstanding.
- The market maker must know the elasticity constant, b.

Strictly, speaking the market maker could keep track of prices rather than shares, but that can often present problems with rounding numbers, since in practice it is easy to constrain the number of shares to an exact value, but harder to constrain the price changes.

Prices

In our experience, many people do not feel comfortable with prices ranging from 0 to 1, and are much more comfortable with 0 to 100. To accommodate prices in a range $(0, c)$ it is necessary to replace amounts of currency with their scaled equivalents, i.e. P with P/c, and K with K/c in all of the preceding formulae. See the practical numerical framework section below for the modified formulas.

Short Sales

In our experience, many people wish to enter into *short* sales—selling shares they don't own, but presumably believe are overpriced. The market maker supports short sales.

Although technically supported, there is one practical reason why some markets disallow short sales. Short sales create a liability on the part of the seller to cover that sale some day. This adds complexity to the overall trading platform, and creates the need for more policies to govern the degree to which short sales must be "covered."

To avoid short sale issues, we encourage presenting traders with the ability to "bet against" a security by buying a bundle of an equal number of all other securities.

Rounding

If not handled properly, the rounding of floating point numbers on a computer can create possible ways to pump money out of the market maker by creating a sequence of trades that round in the trader's favor.

Avoiding these errors is straightforward. First, the market must define the precision at which shares are traded. For instance, whole shares, shares to two decimal places, etc., can be supported. Similarly, the market must define the precision at which money is traded. The internal precision at which shares and money are traded may be greater than that displayed to the trader, in which case the trader sees a truncated or rounded representation of the actual value. (There is a risk of trader confusion when actual and displayed precision differ, however.)

Once the precision has been decided, the following rules must be obeyed:

- Use share quantities as the fundamental units for any transaction.
    - For purchases from the market maker, round quantities down to the precision.
    - For sales to the market maker, round quantities up to the precision.
- Once share quantities have been computed, compute total costs.
    - For purchases from the market maker, round costs up to the precision.
    - For sales to the market maker, round costs down to the precision.

Share prices need not be rounded up/down if the market maker charges a “transaction fee” that exceeds the precision of money in the system.

Turning Currency Into Something of Value

Sometimes prediction markets employ virtual currency in lieu of real currency, which may subsequently be turned into something of value. Whatever scheme is employed, it is important to preserve *incentive compatibility*, i.e. a trader's incentive should always be to tell the truth to maximize his reward. One hazard is the inadvertent creation of *tournament incentives*, where a trader is rewarded for making risky predictions that do not necessarily represent his true beliefs. Rewarding the top N traders in a market creates tournament incentives that are not incentive compatible. Since the primary goal of prediction markets is to make accurate predictions, we strongly advise caution in this area.

An incentive compatible alternative is to convert market currency into raffle tickets, and then award prizes by lottery. However, in our experience this can be quite frustrating to traders when prizes are won by traders with small balances. We sometimes compromise by offering two prizes per market: a large prize awarded by lottery, and a smaller prize that goes to the "top trader" in the market. A small tournament incentive is present, but the larger lottery prize discourages risky behavior and preserves overall incentive compatibility. Similarly, simply publishing rankings creates tournament incentives among competitive traders.

Setting and Adjusting Market Maker Elasticity

Determining a desirable elasticity for the market maker is difficult. If a market is too elastic, traders will observe wild price fluctuations. If a market is not elastic enough, traders are frustrated by the relatively static prices.

As discussed above, our approach has been to make an estimate of the participation we expect for a market, then to set the market maker elasticity so that if all active traders predict the same outcome with all of their currency, the price for that security is driven to a fixed value. Unfortunately, estimating the participation is difficult for new markets.

We are currently deploying markets that adjust their elasticity as trader capital enters the market. We limit the size and frequency of the changes to avoid creating the perception among traders that the market is changing radically. When the elasticity changes, the market maker adjusts the stock vector to keep prices from changing.

Encouraging Participation

We often observe large spikes in trading activity right after a market is created or a reminder sent, followed by low trading activity until the next reminder. To increase participation (without creating perverse incentives) we now reward visits to the market web site itself, regardless of resulting trading behavior.

V. PRACTICAL NUMERICAL FRAMEWORK

When implementing an automated market maker, we need to take everything discussed above into account, including the inaccuracies of floating point representations and computations. We add in the scaling factor *c*, which represents the maximum price for a security, most commonly 1 or 100 (both have been used at Microsoft). Adjusting the math for this results in:

What is the total cost of a transaction that would move the price of a security from P to P'?	$K = -bc \log \frac{c-P}{c-P'}$
What is the total cost of buying Q shares of a security whose initial price is P?	$K = -bc \log\left(\frac{P(e^{Q/b}-1)}{c} 1\right)$
How many shares of a security must be bought/sold to move the price from P to P'?	$Q = b \log \frac{P'(c-P)}{P(c-P')}$
How many shares of security can be bought/sold for a total cost of K?	$Q = b \log\left(\frac{c(e^{\frac{-K}{bc}}-1)}{P} + 1\right)$
What will be the resulting price of a security after buying/selling Q shares of that security with an initial price of P?	$P' = \frac{c}{1 + \frac{c/P-1}{e^{Q/b}}}$
What will be the result price of a security with an initial price of P after a transaction with a total cost of K?	$P' = c - \frac{c-P}{e^{-K/bc}}$

We'll set prices scaled to *c*:

$$P_i = \frac{ce^{s_i/b}}{\sum_k e^{s_k/b}}$$

Running a market requires us to pick a value for b, which we can do using the method described above for a market with N possible outcomes (and thus N securities). Given a total expected flow of capital into the market of K, and a desired upper price target if all of this flows into one security of Pupper, we will use:

$$b = \frac{-K}{c \log \frac{N(1 - \frac{P_{upper}}{c})}{N-1}}$$

VI. PRACTICAL NUMERICAL EXAMPLE

We present a market example, do all the math to six digits after the decimal point and use the natural logarithm. Consider a market designed to estimate the probability of four possible outcomes for sales of product X in calendar year 2009:

A. Sales < 1,000,000 units
B. Sales ≥ 1,000,000 units but < 1,500,000 units
C. Sales ≥ 1,500,000 units but < 2,000,000 units
D. Sales ≥ 2,000,000 units

We refer to each contingent security here using the letter assigned to it (A, B, C or D). The four possibilities (N = 4) are mutually exclusive, and in practice should be anchored to a real-world event, which in this case might be the official sales result report released in January, 2010. In this example, we scale our prices so that securities float between \$0 and \$100 (c = 100). After the real-world event occurs, each share of the security representing the outcome that occurred is worth \$100, and each share of the other securities is worth \$0.

Each trader will start with \$10,000 in virtual dollars. The only thing left to determine is the value of b to select for the market. Make b too small, and the market prices will move too easily. Make b too large, and traders will be frustrated by their inability to influence the market prices. In this example, we are expecting 20 active traders from the pool of invitees. We set b so that if all 20 traders invest everything in one outcome, the price of that security will rise to \$99:

$$b = \frac{-K}{c \log \frac{N\left(1 - \frac{P_{upper}}{c}\right)}{N-1}}$$

$$= \frac{-(20 * 10000)}{100\log\frac{4\left(1-\frac{99}{100}\right)}{4-1}}$$

$$= 463.232312$$

We now open our market with all prices set to *c/N*, which in this case is 100/4 = \$25, so P_A = \$25.00, P_B = \$25.00, P_C = \$25.00 and P_D = \$25.00. Now our first trader asks to purchase \$5000 worth of security B, so we need to calculate how many shares that is, where *K* represents the total change to the trader's cash balance (-\$5000):

$$Q_{\boldsymbol{B}} = b\log\left(\frac{\mathrm{c}\left(\mathrm{e}^{\frac{-K}{bc}} - 1\right)}{P_B} + 1\right)$$

$$= 463.232312\log\left(\frac{100\left(\mathrm{e}^{\frac{5000}{463.232312*100}} - 1\right)}{25.000000} + 1\right)$$

$$= 174.004846$$

To prevent rounding errors from causing cash-positive transactions that should not be ("cash pumps"), we round the number of shares awarded down as discussed above, in this case to 6 digits after the decimal point. Since the amount of currency is exactly \$5000, we do not need to round it up. Note that there are many transactions supported by this model, including multiple simultaneous purchases, and that we are illustrating only a simple single-security purchase and sale here. We then calculate the new price for each security given the outstanding stock sold by the market maker so far ($s_A = 0$, $s_B = 174.004846$, $s_C = 0$, $s_D = 0$):

$$P_A = \frac{ce^{\frac{s_A}{b}}}{\sum_k e^{s_k/b}}$$

$$= \frac{100e^{\frac{0}{463.232312}}}{e^{0/463.232312} + e^{174.004846/463.232312} + e^{0/463.232312} + e^{0/463.232312}}$$

$$= 22.442099$$

$$P_B = \frac{ce^{\frac{s_B}{b}}}{\sum_k e^{s_k/b}}$$

$$= \frac{100e^{\frac{174.004846}{463.232312}}}{e^{0/463.232312} + e^{174.004846/463.232312} + e^{0/463.232312} + e^{0/463.232312}}$$

$$= 32.673702$$

$$P_C = \frac{ce^{\frac{s_C}{b}}}{\sum_k e^{s_k/b}}$$

$$= \frac{100e^{\frac{0}{463.232312}}}{e^{0/463.232312} + e^{174.004846/463.232312} + e^{0/463.232312} + e^{0/463.232312}}$$

$$= 22.442099$$

$$P_D = \frac{ce^{\frac{s_D}{b}}}{\sum_k e^{s_k/b}}$$

$$= \frac{100e^{\frac{0}{463.232312}}}{e^{0/463.232312} + e^{174.004846/463.232312} + e^{0/463.232312} + e^{0/463.232312}}$$

$$= 22.442099$$

So the purchase transaction moved the market prices of A to \$22.44, B to \$32.67, C to \$22.44 and D to \$22.44. Note that although we may display the prices rounded to the nearest penny, internally we always compute them from the stock vector, which is an exact representation of the number of shares outstanding, and maintain higher precision for internal calculations. In this way we avoid cumulative rounding errors, which can be a big problem over large numbers of transactions.

Now suppose the same trader wishes to sell his stake of 174.004846 shares of B. Since the number of shares is exactly 174.004846, we do not need to round it up. We compute the amount of money this stake is worth, rounding the amount of currency awarded down to prevent cash pumps as before:

$$K = -bc \log\left(\frac{P_B\left(e^{\frac{Q_B}{b}} - 1\right)}{c} + 1\right)$$

$$= -463.232312 * 100 * \log(\frac{32.673702\left(e^{\frac{-174.004846}{463.232312}} - 1\right)}{100} + 1)$$

$$= 4999.999916$$

Note that our rounding to avoid cash pumps intentionally destroyed \$0.000084 of currency for the trader with this purchase and subsequent sale. If instead we allowed minute fractional positive cash flow, then we would be subject to scripted attacks that generated cash. Just like share prices, the logical thing to do is to display the rounded value of \$5000.00 to the user, meaning that the loss of minute fractions of cash would only become visible to a trader after many such transactions. We have not seen this in practice.

Finally we compute the new prices of each security given the outstanding shares of each security (s_A = 0, s_B = 0, s_C = 0, s_D = 0) in the same way as above, which results in a return of the prices to P_A = \$25.00, P_B = \$25.00, P_C = \$25.00 and P_D = \$25.00.