Automated Trading in Agents-based Markets for Communication Bandwidth

Nir Vulkan[1], Chris Preist
Trusted e-Services
HP Laboratories Bristol

HPL-2000-24
9th February, 2000*

Email: N.Vulkan@Bristol.ac.uk
cwp@hplb.hpl.hp.com

electronic commerce, internet auctions, electronic data submissions

Automated agents are increasingly being used by organisations and individuals trading in electronic markets. Agents are particularly useful in markets where trade might not have been possible otherwise, for example because a lot of information must be processed quickly, or because employing human traders in 24-hour, small transactions markets is not cost-effective. Markets for communication bandwidth, where organisations trade the rights to transmit data over a network, are one such application. Because demand fluctuates considerably every few seconds agent-based spot markets provide extra liquidity.

This paper considers the design of agents which automatically trade in a k-double auction market for communication bandwidth. We suggest criteria and a general framework for building adaptive agents based on ideas from statistical decision theory. In particular our agents are designed to differentiate stable from unstable market conditions and to best-respond to these changes.

[1] Department of Economics, University of Bristol, 8 Woodland Road, Bristol BS8 1TN.
* Internal Accession Date Only
© Copyright Hewlett-Packard Company 2000

(1) Introduction

With the increasing spread of the Internet, more and more commercial transactions are taking place electronically. Consumers purchase books and CDs, and businesses conduct electronic auctions of supply contracts. Electronic double auctions, such as Fastparts (http://www.fastparts.com/), allow buyers and sellers to negotiate with each other. Some goods, such as software, can also be delivered via the net. The Internet, particularly the World Wide Web, is emerging as one of the most efficient media for carrying out business-to-consumer and business-to-business transactions. The competitiveness of the internet is likely to further increase: *Agent technology* could become for e-commerce what Windows was for PCs - a relatively simple and user-friendly way of using the new technology. In this framework individuals and organisation interact via the network using *software agents.* A software agent (also *automated,* or *autonomous* agent) is a program that acts independently on behalf of its user and in of its interests. The main characteristic of these interactions is that the user *delegates* the authority to search, match, and even to transact business to the agent. See [11] Vulkan(1999) for a discussion of the economic implications of this.

An interesting recent development in electronic commerce is the buying and selling electronically of the right to transmit data over the net and the telecommunications infrastructure. Organisations like Band-X (http://www.band-x.com/), RateXchange (http://www.ratexchange.com/) and Min-X (http://pulver.com/min-x/) provide bulletin boards and double auctions to buy and sell bandwidth and connection time. Currently, these transactions are wholesale, between large operators. However, we could imagine such transactions taking place far more frequently.

If negotiation were cheap and easy, and appropriate billing infrastructure was in place, re-negotiation of connection contracts could take place every few seconds. This would allow sources of traffic (such as local Internet service providers) to dynamically switch between long distance carriers in response to price fluctuations. In this way, a spot market for bandwidths could develop. For an overview of the current and future state of Internet bandwidth markets, see [3] Lehr and McKnight (1999).

One of the keys to making this happen is the ability to negotiate automatically. If negotiations are to take place constantly it would be expensive for people to perform the task. Furthermore, they will not always be able to react fast enough to changes in market conditions. For this reason, communications bandwidth has inspired work on automated negotiation from its inception [6] Rosenschein and Zlotkin (1994).

One approach to automate negotiations is to consider the structure of the market game-theoretically, and treat the choice of what bid/offer to make at any given time as the selection of a strategy (for example, using the framework of Harsanyi, [8] Rustachini et al. (1994), study the equilibria of k-double auctions. Game theory and economic theory provide us with tools that allow us to analyse the relationship between market structures and efficiency of outcomes. The design of an automated market for self-interested agents is that respect no different from any other market. Companies which set up electronic markets, on the Internet or elsewhere, want them to maximise their future profits through efficiency and competitiveness. Similarly, organisations want agents which maximise their profits, given the communication protocol (or "the market design", as an economist will put it).

Accumulated knowledge from economic theory and mechanism design can be applied to these goals. Over the last three decades, rapid progress in implementation theory (mechanism design), had brought the subject to an engineering-like state, where a large number of well understood mechanisms can be prescribed for participants with a given set of preferences. However, human agents often find ways to outsmart the designers of these mechanisms. Markets for automated agents can therefore be a much more suitable application for implementation theory. An automated agent is a pre-programmed algorithm, very much like the game-theoretical concept of strategy. In this sense, game theory (and mechanism design) seems more suitable for automated agents than it is for humans.

For agents to be fully autonomous, they must be able to learn and adjust to changing circumstances. Agents will choose their bids <u>on-line</u> based on current information from the user (e.g. maximal willingness to pay, priority of achieving trade) and on past observations. Designing adaptive agents which trade in spot markets is a difficult task -

to some extent this is similar to trying to quantify the work of financial traders - but one for which a practical solution must be found in order for such applications to be implemented.

There is a large literature on learning in games, and in auctions in particular.[3] Experimental game theory, behavioural economics, and evolutionary economics all focus on the learning process and on the effects it might have on behaviour in the steady state. Many models in which an individual learner is faced with an uncertain environment have been developed recently (*e.g.* [5] McKelvey & Palfrey (1995), [3] Fudenberg & Levine (1997), [1] Erev & Roth (1997), [7] Roth & Erev (1995)). These are motivated by question such as: Under what conditions will players converge to an equilibrium? What degree of rationality should players posses in order to reach an optimum? And which rules best resemble how real people behave?

Although we are also interested in the adaptive process and its possible affects on outcomes, our motivation is different: We look for conclusions that could be used by the engineers who design artificial traders. Moreover, we are not interested in equilibrium as such, because in the market for communication bandwidth there is no "stable" stage game: The number and identity of players change constantly. In fact, the number of players at any trading period will typically not be known in advance. Moreover, exogenous shocks to demand and supply will be indistinguishable from changes in aggregated behaviour, which complicates the prospect of formal analysis even further. Even the agents that remain in the market for a number of periods may change their utility model because their users' requirements change.

Our concerns about rationality are also different from those of experimental economists in that complexity should only matter if it is on-line. Off-line, the agent can be as complex as necessary, as long as it is able to quickly and efficiently process the available information on-line and decide its bid. In this sense complex algorithms, which seem very unlike real people's decision rules, can be considered as long as they

[3] There is also a large literature on learning in financial markets. However, this literature focuses on the transmission of private information via the price mechanism. Since we treat bandwidth as a perishable good, this issue is not relevant in our setting.

(a) perform well, and (b) require only a reasonably limited computational effort to produce a bid at period *t*, given the history at *t-1*, and the new reserve price.

It should be clear from our description so far that the bandwidth-trading environment requires new conceptual thinking about learning. The first contribution of this paper is to normatively express the requirements of a "good" learning algorithm in this setting. Because there is no "game" as such, learning takes place in a stochastic environment, where each action (i.e. a bid) is associated with a success probability and a payoff. If these success probabilities are fixed over time the environment can be described as a multi-arm bandit (where each bid value correspond to an arm). If the environment is stochastically stable, then there exists one strategy (i.e. a function mapping from reservation prices to bids), which statistically dominates all other strategies. If this is indeed the case, then a "good" learning algorithm will converge to playing this dominant strategy. However, the environment will not always be (stochastically) stable, and we require that the learning algorithm will be responsive to such changes in the underlying structure. A "good" learning algorithm will quickly adapt to such changes.

We provide a high-level specification for an on-line algorithm (an agent) for trading in communication bandwidth. The algorithm uses the tools of statistical learning theory to test on-line whether the trading environment is consistent with the agent's model of the world. If the environment is stable and is consistent with the agent's model, then the agent best responds to its beliefs (i.e. chooses the bid which maximises the user's expected utility). If the trading environment is not stable then the algorithm switches to its transitory mode where the next period's bid is selected. Finally, the algorithm allows for new models of the environment to replace old ones. We show that our algorithm achieves the two requirements described above, by testing its own success in predicting what happens next at any given stage, but only changing its learning mode when sufficient evidence exists to suggest its learning is not consistent with recent outcomes.

We show that the suggested algorithm satisfies both requirements of convergence and responsiveness. We show also that it combines rapid convergence with maximum responsiveness to changes. We provide a set of simulations which demonstrate the performance of the agent in a number of trading environments.

(2) Double Auction Markets for Automated Agents

In this section, we introduce some economic terminology, and describe the k-double auction marketplace that is used subsequently in this paper.

Buyers and sellers meet to trade goods and services in a market. Buyers may bid to buy a good at a given price, and sellers may offer to sell a good at a given price. The market consists of a clearing mechanism, which determines how bids, offers and other messages can be exchanged to determine a trade. In a k-double auction, buyers announce their maximal willingness to pay, and sellers the minimal price they are willing to accept. Sorting out the buyers bids in increasing order, and the sellers in decreasing order, denote by $[a,b]$ the interval between the buyer's bid with the highest index (in the sorted list) which is still smaller than the corresponding seller's bid with the same index.[4] A k-double auction mechanism selects the clearing price $p=ka+(1-k)b$, where *kÎ(0,1)*. All buyers who bid at least p trade with all sellers which bid no more than p. All those who trade, trade at the clearance price, and all others do not trade. Note that, since the number of bidders is finite, there is a positive probability that a given buyer or seller enters the marginal bid (i.e. a or b). Hence, bidders can gain from setting their bids strategically (i.e. for a seller this is the real value under which it is not worth trading, and for a buyer a price above which trade is no longer profitable). For the case where the number of buyers and sellers, and the distribution of reserve prices are commonly known, the exact formulae for optimal bidding strategies can be found in [8] Rustachini *et al* (1994).

In this paper, we consider a repeated k-DA for trading instantly perishable goods. A good is instantly perishable if it ceases to be useable soon after it becomes available for sale. The right to transmit on a network at a given time is an example of such a good; a

[4] Alternatively, bids can be aggregated into supply and demand curves. The interval [a, b] now correspond to the intercepting section of the two graphs.

network provider can offer connection time now, but if it is not used now, it ceases to be available. You can't store up connection time for sale later. (Though, of course, new connection time becomes available.)

Such a good suggests a marketplace where trades are taking place continuously, and buyers and sellers are continuously adjusting their price in response to the activities of others. Each buyer and seller has a certain good (or need for a good) at any given trading round, and announces the price they are willing to trade at. In the next round, any goods/needs from the previous round expire, but they receive new goods/needs. Hence goods are constantly flowing into the market, and traders are modifying their bids/offers to attempt to trade successfully.

At each trading period users communicate privately to their trading agent their reservation price, i.e. the maximal (alt. minimal) price it is willing to pay (alt. accept) without making a loss. These reservation prices originate from fixed marginal costs (for sellers), or from contracts with end users (for ISP providers, who are buyers in this setting). In addition to these constraints, users may choose to communicate to their agents a high priority of trading in the next period (for example continuous bandwidth may be required when transmitting voice or video, as opposed to transmitting data, which could wait until the ISP provider gets a better deal on bandwidth rates). There are a number of ways to implement priorities. First, the priority of trade can be taken into account by the program that sets the reservation price: A seller who desperately needs to trade will accept lower prices, and similarly, the buyer will pay more. A second possibility is for the agent to be given a general function of two variables, the reservation price and priority (that is, when the user install its agents she is asked to express her utility from combinations of price and priority). However, there are many difficulties in characterising an optimal strategy in multidimensional settings (the first-order conditions are not always well defined in this case, especially if the utility function is not a straight forward weighting of the two parameters).

A third possibility is for the user to indicate a high priority using the weight attached to the *probability* of trading. Note that by entering a bid, the user is effectively participating in a lottery: He trades with probability less than one. A user with a high priority of trade is therefore more *risk averse* in economic terms. We propose a method where this risk aversion is explicitly entered into the agent's decision choice. To illustrate how this can be done, consider a seller with a reservation price (excluding any priority considerations) r. The monetary payoff for this seller from trading at price p is p-r (Similarly, the payoff for a buyer is r-p). Remember that since the number of bidders is finite, p will depend on posted bids i.e., agents are not price takers. Denote by $P(b)$ the *ex-ante* probability of trading in the next period as a function of the posted bid, b. A reasonable objective function for a risk neutral agent with no special priority is therefore likely to be of the form $P(b)\times(p-r)$. A higher bid can increase the payoff from trading (because there is a positive probability that this agent will happen to be one of the two agents whose bids determine the clearance price), but decreases the probability of trading. A lower bid does the converse, meaning the agent is more likely to trade but will make less profit. We can generalise the risk-neutral objective function to include a measure of risk, α. Given α, the agent chooses the bid which maximises $P(b)\times(p-r)\times(1-\alpha P(b)\times(1-P(b))$. If $\alpha=0$, the agent is risk neutral. If α is positive, the agent is risk averse, and prioritises getting a trade over making a large profit. If α is negative, the agent is risk seeking. This method of representing priorities has several advantages: First, it allows for a separate representation of reservation prices and priorities. Reservation prices are normally associated with the physical costs and constraints from fixed-price contracts with end-customers of telecommunication services, while priorities result from shorter-term consideration like the nature of current transmission. On the other hand, agents still maximise a single dimension function, hence they can compute the (typically unique) optimal bid quickly.

(3) Learning in a double auction for bandwidth

In this section we consider the normative aspects of trading in double auction markets.

A history-independent strategy (HIS)[5] is a function from the set of reservation prices to the set of possible bids. Normalising bids to *[0,1]*, an HIS for the buyer is therefore a function *B:[0,1]® [0,1]*, and *S:[0,1]® [0,1]* for the seller, where *B[r]* returns the bid corresponding to a reserve price of *r*. We restrict attention only to strategies which do not use dominated (loss-making) bids, i.e. for sellers *S(r)³r*, and for buyers *B(r)£r*. An additional reasonable assumption is that *B'(r)³0* that is, bids are not a decreasing function of the reserve price (and similarly, *S'(r)£0*). Notice that even if the number of possible bids is finite, the number of bidding strategies is $O(n^n)=O(2^n)$, where *n* is the number of possible bids. Learning algorithms that consider and compare all possible strategies are therefore not feasible.

Suppose that the number of bidders is first drawn according to NBIDS, and that then all these bids are independently drawn according to a distribution BIDS, with support on the unit interval. Suppose both distributions were known. Then it is possible to compute the optimal bid, given the reservation price. For the case when the number of bidders in known, the exact formula determining the optimal bid, see [8] Rustachini et al (1994).

Denote now by *Opt(r)* the function which, for any given reservation price, returns the optimal bid (in a stochastically stable world). Let *L* be a learning algorithm, where $L(r^t, h^{t-1})$ returns period t's bid given a history. Following (but slightly abusing) the terminology of [2] Friedman and Shenker (1998), we can now introduce the following definitions:

Optimality: Suppose that the world is stochastically stable. Then we say that a learning algorithm *L* is optimal if and only if $lim_t|L(r^t, h^{t-1})-Opt(r^t)| = 0$.

Responsiveness: Suppose that a stable stochastic structure starts at period t'. Then L is responsive if $lim_t|L(r^t, h^{t-1})-L(r^t, h^{t-1/t'})| = 0$, where $h^{t-1/t'}$ denotes the history from t' to t-1.

[5] That is, a strategy in the one-shot double auction game.

Degree of Convergence: We say that L1 converges faster than L2 if $|L1(r^t, h^{t-1})-Opt(r^t)| < |L2(r^t, h^{t-1})-Opt(r^t)|$ almost always.

Degree of Responsiveness: We say that L1 is more responsive than L2 if $|L1(r^t, h^{t-1})-L(r^t, h^{t-1/t'})| < |L2(r^t, h^{t-1})- L(r^t, h^{t-1/t'})|$ almost always.

(5) High Level Spec of the On-line Trading Algorithm

The procedures used by the main algorithm are explained in sufficient details for a programmer to create a working code, although "fine tuning" of some of the parameters (e.g. the degree of statistical confidence with which the Null hypothesis are accepted) is deliberately left unspecified. A trail-and-error process is required to set the actual values of these parameters. We come back to this point towards the end of this paper.

The top-level algorithm is defined in pseudocode as follows;

```
For each time period t:
IF model of the world is stable
THEN
    Generate bid by maximising the given utility function subject to the
    current estimates of the model
    Update the estimates of the model



ELSE
    Use transitory mode to generate a bid
    Update the transitory bidding rules (optional)
    IF model of the world was stable at time t-1 THEN re-initialise estimates
```

At t=0, model of the world is unstable.

Model of the world becomes stable at time t if observed data is continuously consistent with the model of the world.
Model of the world becomes unstable at time t if observed data is continuously inconsistent with the model of the world.

(5.1) Components of the model

The design consist of the following 4 components:

1. The model of the world and updating
2. Consistency test for observed data
3. Test for continuous inconsistency
4. Transitory bidding rules and updating

The Model of the World and Updating

If the world is stochastically stable, in the sense that the relevant underlying are fixed over time, then the agent can estimate these probabilities using available data. If the agent can observe the actual bids, then it can estimate the distributions NBIDS and BIDS. For example, NBIDS is estimated by an array $NBIDS^t[\cdot]$, initialised as zero everywhere, and updated using: $NBIDS^t[i] = NBIDS^{t-1}[i]$ for all $i \neq n^t$, and $NBIDS^t[i] = NBIDS^{t-1}[i]+1$ for $i=n^t$, where n^t denotes the number of bids observed at period t.

If bids cannot be observed directly, then the agent can estimate $Trd(\cdot)$, the probability that a given bid is smaller than the market clearing price. The empirical distribution $Trd^t(s)$ is used to measure the proportion of time, in the previous t trading periods since Trd^t was last initialised, where the bid s would have been accepted. Denote by p^t the equilibrium price at period t. This implies that a seller bidding any price $s \leq p^t$ would have succeed at trading at period t. (Similarly, any buyer bidding $b \geq p^t$ would trade). Sellers start off with $Trd^t(s)$ initiated as zero everywhere and updated using:

$$Trd^{t+1}(s) \;=\; \frac{t \cdot Trd^{t}(s)+1}{t+1} \quad \text{if } s \le p^{t}$$
$$Trd^{t+1}(s) \;=\; \frac{t \cdot Trd^{t}(s)}{t+1} \quad \text{if } s > p^{t}$$

One can imagine more sophisticated forms of beliefs: For example, beliefs about possible patterns in the market (e.g. “market are bearish in January”). That is, the distribution of trading prices is stable, but time dependent. In any case, estimating and updating of the model will take a similar form (only the agent will be updating more sophisticated data structures).

The agent then solves its maximisation problem replacing the real distributions with its empirical estimates. Since, in a stochastically stable world, these empirical distributions converge to the real distributions (according to the law of large numbers), the outcome of this maximisation problem will also converge to *Opt(r)*.

Consistency Tests

At any stage of trading, except the first few rounds, the agent is able to compare its current model of the world (e.g. the empirical distribution $Trd^{t}(s)$), with recent data. This can be done by defining and testing the appropriate Null hypothesis. That is, the agent should be able to conclude that “There is, or is not, sufficient evidence with which to reject the hypothesis that the data is consistent with the model”. This test is then carried out to a pre-specified error measure.

The agent does not have a pre-conceived idea about the nature of the distribution it is estimating. In practice, these distributions can take any form. Therefore a non-parametric test should be carried out. There are a number of such tests: For example, the Kolmogorov-Smirnov test considers the absolute value of the difference between the sample and the model. It turns out that this difference is distribution-free (see, for example, [9] Silvey (1975)). The null hypothesis is then accepted if the difference is below a pre-specified threshold, and rejected otherwise.

There are many other non-parametric tests. In our simulations, we use convergence of means. Here, the null hypothesis is expressed in terms of the difference in means between the sample and the model. We come back to this issue in more details, when we describe our simulations. Whichever test is used, the user must specify (a) the size of the buffer which defines the sample size, and (b) the acceptance threshold.

Tests for Continuos Inconsistencies: Detecting Structural Breakdowns

If the Null hypothesis is rejected once, then this could be because of "bad" sample. Of course, the larger the sample size, the less likely this is to happen. Still, our simulations suggest that even for finely tuned learning parameters, bad data will occur. We therefore take the approach that a small number of rejections should be tolerated by the algorithm before past history is thrown away.

More specifically, we check whether the null hypothesis is consequently rejected a fixed number of times. Only if this happens, we conclude that a structural breakdown is likely, and therefore re-initialise beliefs (effectively throwing away old data). In general, the user may specify a different test to determine when to throw old data away. When data is discarded in this way, the model of the world is considered unstable, and the algorithm moves to its transitory mode. It is considered to return to stability when the null hypothesis is accepted a fixed number of times, and the algorithm becomes belief-based once more.

Transitory bidding rules and updating

The purpose of this learning mode is to choose bids in those cases where the agent's current model of the world is not consistent with recent data. Since the world is not stochastically stable, and the number of players is not known, there is no theory of optimal trading in these cases. Moreover, reinforcement-type learning, where all strategies are played with positive probability, and where these probabilities are updated according to some measurement of "fitness" (i.e. how well each strategy performs), are not a realistic option either. As we explained above, the number of strategies (i.e. functions from reservation prices to bids) is exponentially large.

Reinforcement learning theories were originally introduce to capture specific features of *human* learning (and in particular “bad learning”, like the probability matching affect, see [10] Vulkan (1998)). Hence they provide very little insight to the design of robot-traders. However, some of these models (e.g. Q-learning, Generic Algorithm) proved successful is unstable environments, where the predictability of any theory based on past-performance is normally very low. Since the model never converges (in finite time) to a probability mass of one (i.e. to playing a single rule), any of a number of rules can be selected. Moreover, the accumulated weights placed on these rules provide the agent with a reference base to those rules that have proven successful in the past, under similar unstable circumstances.

There are several possibilities on how to design the transitory learning mode. First, one could follow the approach taken in financial markets, where technical rules are used to generate bids when fundamental trading (i.e. where economic reasoning is used to generate expectation on what is likely to happen next) fails. However, since little is known on the behaviour of spot markets for bandwidth, there is very little to base these rules on. Second, it is possible to specify a small number of rules-of-thumb, S, and use a reinforcement mechanism to select amongst these rules. Here are a few examples of such rules for a seller agent:

1. Maximal Likelihood Rule: Ask for the reservation price, $s(r)=r$. This rule maximises the likelihood of trade, but at the expense of abandoning profits.
2. Greedy Rule: A rule which asks for the highest possible price, $s(r)=1$. Profits are maximised, but the likelihood of trade is minimised (the equivalent rule for the buyer is $b(r)=0$).
3. Linear Rules: Any combination of the above two rules: Asks for the average between the maximal price and the reservation price, $s(r)=qr+(1-q)$ for *qÎ (0,1)* (equivalent rule for buyer agent is $b(r)=qr$).
4. Constant bid rules: Use the same bid always, except in those cases where this bid is lower than the reservation price: e.g. $s(r)=max(0.75, r)$.

5. Decreasing Surplus Rules: A rule whereby the demanded surplus (i.e. the ask price minus the reservation price), decreases with r. Any rule which satisfies the following three constraints: (1) *s(1)=1* (2) *s(r)³r* for all $r \in$ (0,1), and (3) *s'<1*.[6] For example, $s(r) = \sqrt{r}$. (or, $b(r) = 1 - \sqrt{(1-r)}$ *)* for the buyer agent).
6. History Dependent Rules: For example $s = f(eq^{t-1})$, or $s = f(eq^{t-1}, eq^{t-2})$ and so on.[7]

Since S is incomplete, the choice of rules in S could, in principle, affect the agents' learning outcome, and payoffs. To minimise the risks from such an ad-hoc approach it is possible to add a rule which returns a random number between r and 1, so that the reinforcement mode returns any value in the interval [r, 1] with positive probability That is, although the set of strategies is incomplete, all (undominated) outcomes are supported.

(5.2) Analysis of the learning algorithm

For a certain class of environments, we can prove useful properties of our algorithm.

Staticity: We say that a trading environment is *static at the time interval* $[t_1, t_2]$ (where $t_2 >> t_1$), if for every $s \in [0,1]$, the Probability($s^t < p^t$), and the payoff from bidding s^t, assuming the bid is accepted, are fixed for all $t \in [t_1, t_2]$.

Semi-staticity: We say that a trading environment is *semi-static* if these exists a set $\{t_1, t_2, ...\}$ such that the trading environment is stable at the intervals $[t_i, t_{i+1}]$, for $i=0,1,..$

Denote by L^* any algorithm which fits the above high-level spec. Denote by α the probability that the model becomes unstable when the trading environment is stable (i.e. that the null hypothesis is continuously rejected.) Denote by β the probability that

[6] That is, *s(r)-r* is a decreasing function of *r*.
[7] An underlying assumption of the belief model is that the events eq^t are i.i.d. (although they are allowed to be time dependent). If, however, eq^t is dependent on previous outcomes, say on eq^{t-1} then this assumption is clearly violated. The purpose of this class of rules, is therefore to detect and therefore capitalise on such dependencies.

the algorithm fails to re-set beliefs when the underlying structure changes (that is, the null hypothesis is not continuously rejected immediately after a change in the underlying probabilities). As we explain later, the tests employed by the algorithm should be chosen so to minimise both α and β. We are now able to prove the following results:

Proposition 1. *L** is optimal.

Proof. Suppose that the trading environment is stable over $[t_1, t_2]$. Then the null hypothesis is accepted with probability 1-α at any stage in that interval. Hence, with probability 1-α *L** will not re-set its empirical distributions. Because of the law of large numbers the empirical distributions will converge to the real distributions, and $L^*(r^t, h^{t-1})$ will converge to $Opt(r^t)$.

Remark: Notice that the above algorithm has also a fast rate of convergence. First, note that the tests are chosen to minimise α, hence the probability of counting data which "counts" is maximised. Second, *L** does not keep old history (like many other learning algorithms used by game theorists, like fictitious play), which is likely to be irrelevant to the current underlying structure. Hence, with a large probability, the estimates are based on all, and nothing but, relevant observations. Hence, the rate of convergence is maximised. The importance of throwing away old observation is highlighted in our simulations (and is nicely illustrated in Figure 4).

Proposition 2. *L** is responsive.

Proof. Let the environment be stable from period t'. With probability 1-β L* will throw away observations made before period t'+c (where c is a small constant depending on the test used for re-setting beliefs). Hence, with probability 1-β $L^*(r^t, h^{t-1}) = L^*(r^t, h^{t-1/t'})$, for $t>t'+c$. Hence, *L** is responsive.

Proposition 3. In a semi-static environment, there is no other learning algorithm that is more responsive than the above algorithm.

Proof. Note from the proof of Proposition 2 that with probability 1-β, $L^*(r^t, h^{t-1}) = L^*(r^t, h^{t-1/t'})$, for $t>t'+c$, or $|L^*(r^t, h^{t-1})-L^*(r^t, h^{t-1/t'})| = 0$. Since the difference is already zero, and since β is minimised, L^* maximises responsiveness.

The responsiveness of L* is clearly demonstrated in our simulations, and is illustrated in Figures 1-3 below.

(5) Simulations

In this section we report the results of preliminary simulations we carried out to test the performance of agents using our top level algorithm.

In these simulations, the agent trades in a semi-static environment, where the stochastic nature of the world remains stable for a period, before changing to another stable structure.

To simplify matters, we assume a one dimensional distribution from which the trading price is taken. If our selling agent bids anything above this price, then it does not trade. However, if the bid is below the current price, the agent trades *at its bid* (this is somewhat similar to a 0-double auction). This provides sufficient structure on the agent's decision problem (so that choosing a higher bid increases expected payoff, but decreases the chances of trading), while allowing us to simulate a relatively simple process of generating a single price (and not all the other bids). Such an assumption is justified when the number of agents is large, because the agent can neglect the effect of its own bid on the clearance price. When the number of traders is small, this is no longer a good approximation of the actual clearing price.

We assume the agent is risk neutral. Hence it computes: $Arg \max_{s \in [r,1]} P(s) \cdot (s-r)$

The agent's model of the world then takes the form of a function *P(s)*, which returns the probability of the bid *s* being accepted. Of course if the world is stochastically stable then *P(s)* is fixed over time for any given value of s. We start with, $P^0(s) \equiv 0$ and the agent updates its beliefs using:

$$P^{t+1}(s) = \frac{t \cdot P^t(s) + 1}{t+1} \quad \text{if s} \leq p^t$$

$$P^{t+1}(s) = \frac{t \cdot P^t(s)}{t+1} \quad \text{if } s > p^t$$

At each trading round we test whether P is consistent with the last 8 observations. Specifically, we compute, at time *t*, the values x_1 and x_2, such that $P(x_1)$»0.4 and $P(x_2)$»0.6. We compute the average equilibrium price over the last 8 periods of trade, A_t. P is considered consistent if $x_1 < A_t < x_2$. Otherwise P is inconsistent.

If P has been inconsistent for 5 consecutive rounds, then P is re-initialised. In other words, the agent detects a structural breakdown of the underlying probability distribution of equilibrium prices. Old information may be harmful in that it will outweigh new, relevant information. Hence, the agent starts learning P from scratch.
P will not be reinitialised again until it has regained stability; in other words, until the agent detects 5 consecutive periods where P is consistent.

We present results in two environments. In each environment, there are 1000 trading periods, split into 5 blocks of 200 periods. In each block, the environment is static, with the equilibrium price p^t being drawn each round from a uniform distribution over a certain range (maximum range, [0,1]). Hence, overall, the environment is semi-static. The distributions used are as follows;

Rounds:	1-200	201-400	401-600	601-800	801-1000
Environment 1:	[0.9,1.0]	[0.5,0.6]	[0.9,1.0]	[0.5,0.6]	[0.9,1.0]
Environment 2:	[0.5,0.6]	[0.6,0.7]	[0.7,0.8]	[0.8,0.9]	[0.9,1.0]
Environment 3:	[0.9,1.0]	[0.8,0.9]	[0.7,0.8]	[0.6,0.7]	[0.5,0.6]
Environment 4:	[0.4,0.6]	[0.3,0.7]	[0.2,0.8]	[0.1,0.9]	[0.0,1.0]
Environment 5:	[0.0,1.0]	[0.1,0.9]	[0.2,0.8]	[0.3,0.7]	[0.4,0.6]

In environments 1 to 3, the distribution range is always of size 0.1. In environment 1, it alternates between two locations. In environment 2, it slowly drifts upwards, while in environment 3 it slowly drifts downwards. In environments 3 and 4, however, the mean of the distribution remains fixed at 0.5. In environment 4, the distribution becomes increasingly wide, while in environment 5 it becomes increasingly narrow.

FIGURE 1: Graphs for environment 1

Figure 1a: Cumulative profits

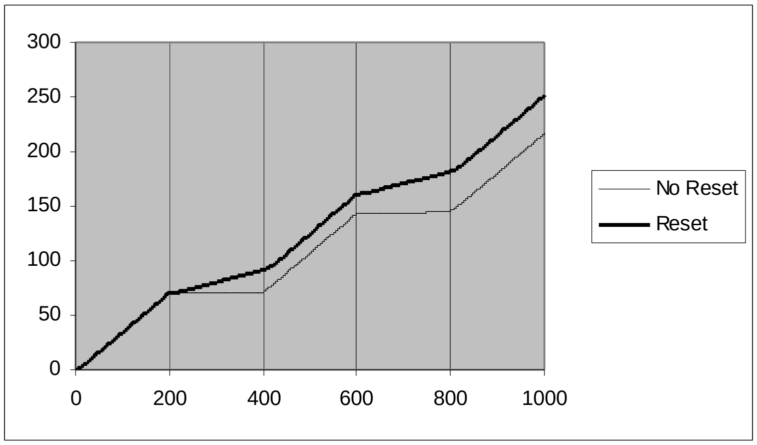


Figure 1b: Deviation of actual bid from apriori optimal belief algorithm plus reset

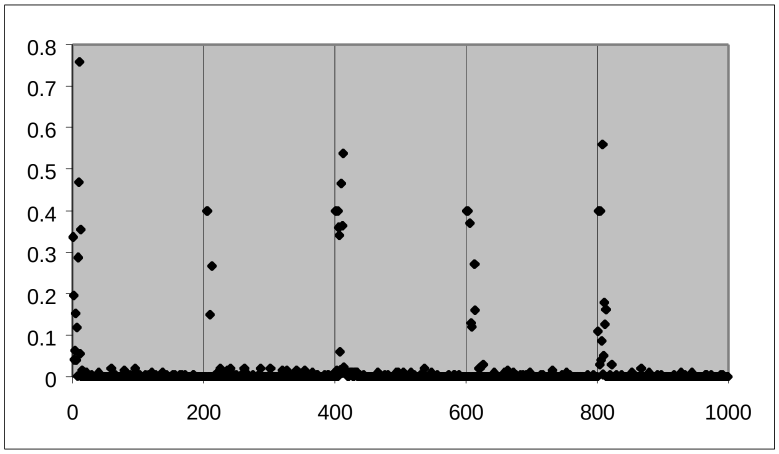


Figure 1c: Deviation of actual bid from apriori optimal belief algorithm without reset

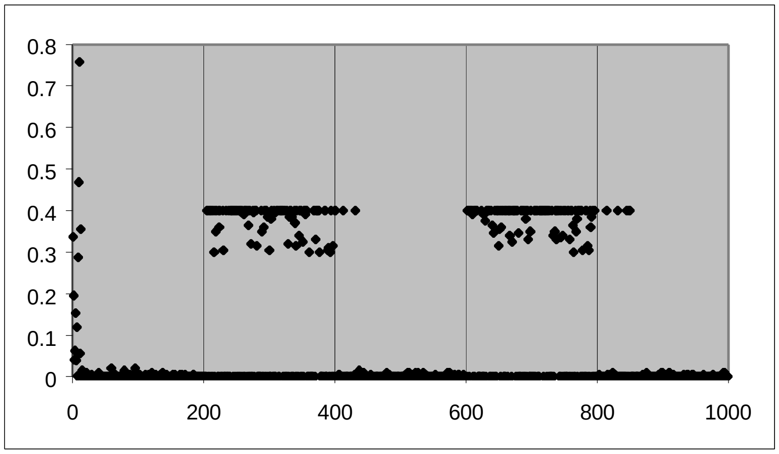


FIGURE 2: Graphs for environment 2

Figure 2a: Cumulative profits

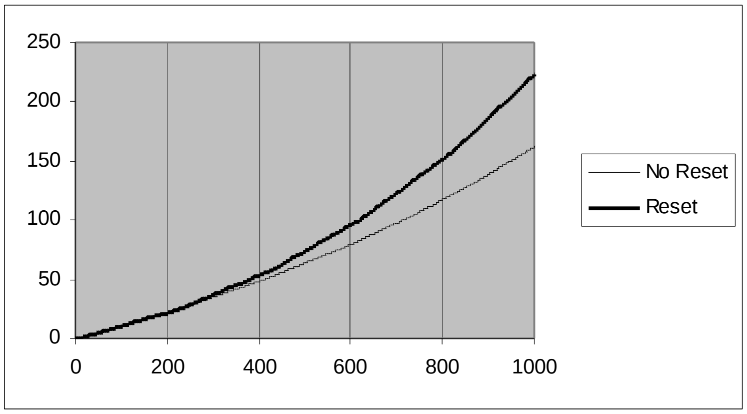


Figure 2b: Deviation of actual bid from apriori optimal - belief algorithm plus reset

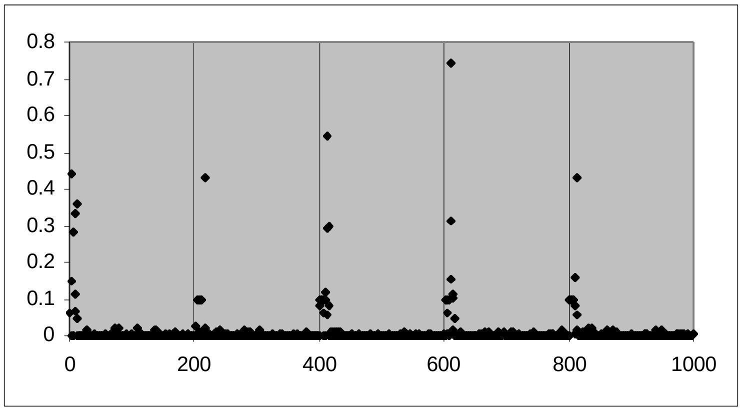


Figure 2c: Deviation of actual bid from apriori optimal - belief algorithm without reset

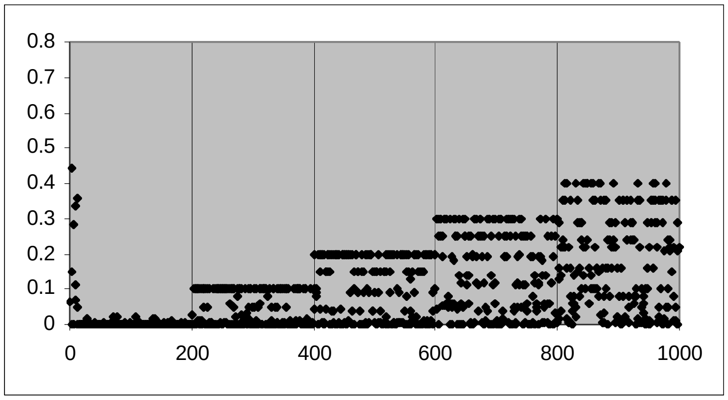


FIGURE 3: Graphs for environment 3

Figure 3a: Cumulative profits

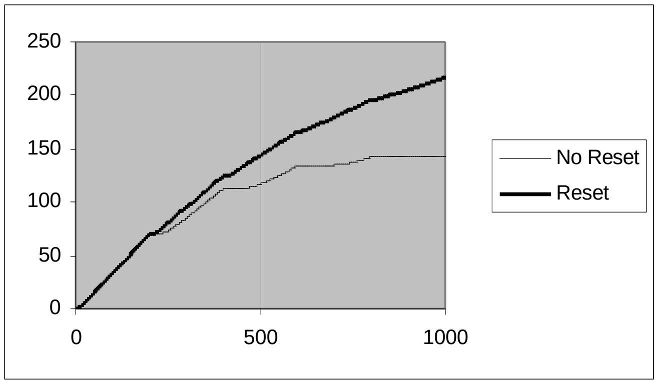


Figure 3b: Deviation of actual bid from apriori optimal - belief algorithm plus reset

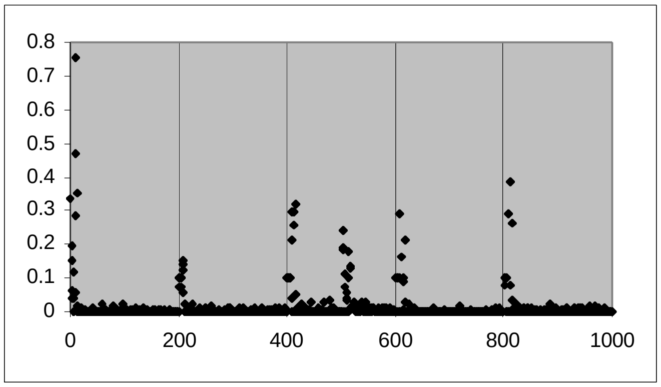


Figure 3c: Deviation of actual bid from apriori optimal - belief algorithm without reset

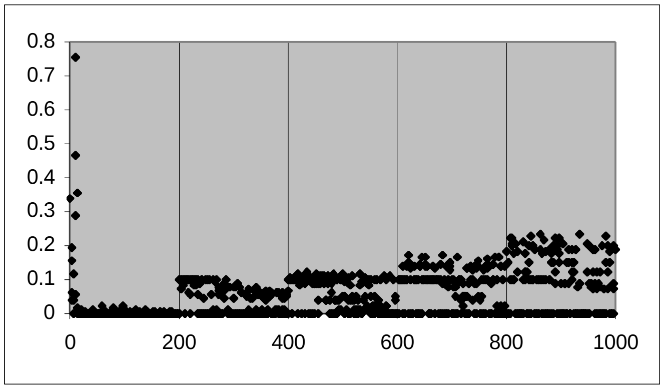


FIGURE 4: Graphs for environment 4

Figure 4a: Cumulative profits

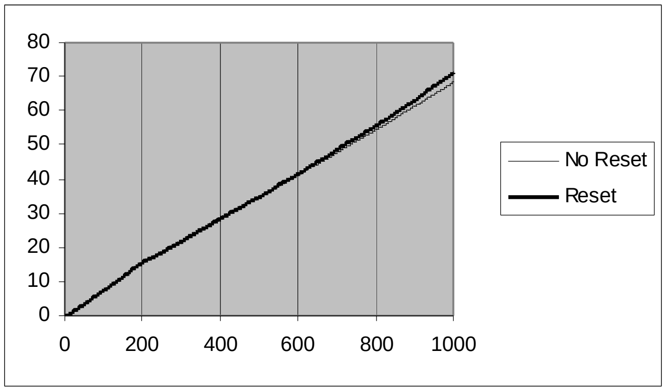


Figure 4b: Deviation of actual bid from apriori optimal - belief algorithm plus reset

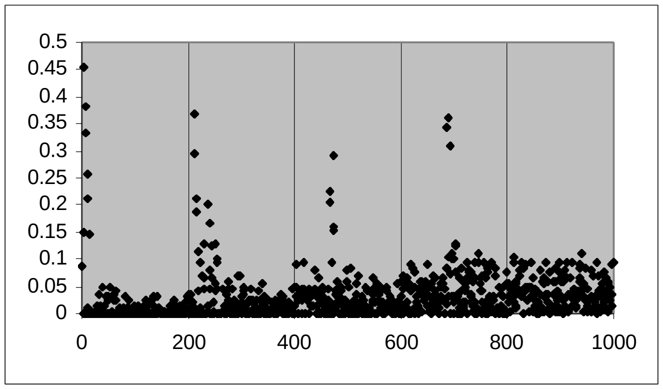


Figure 4c: Deviation of actual bid from apriori optimal - belief algorithm without reset

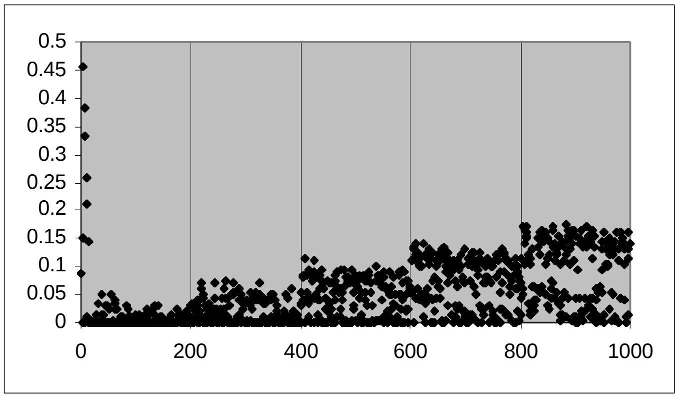


FIGURE 5: Graphs for environment 5

Figure 5a: Cumulative profits

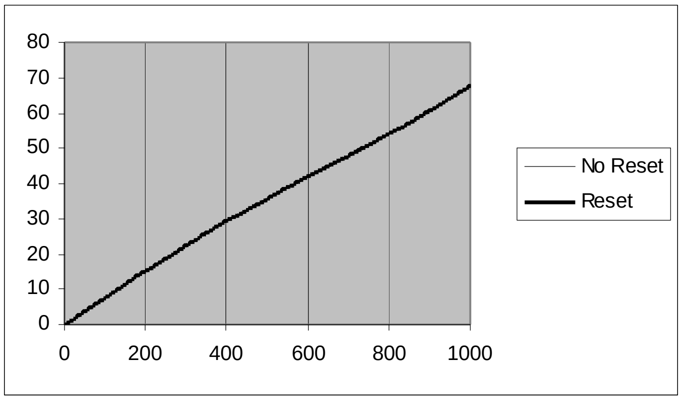


Figure 5b: Deviation of actual bid from apriori optimal - belief algorithm plus reset

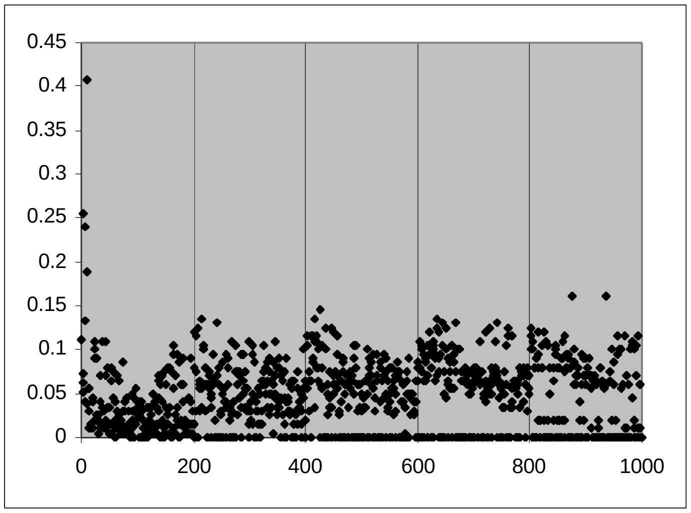
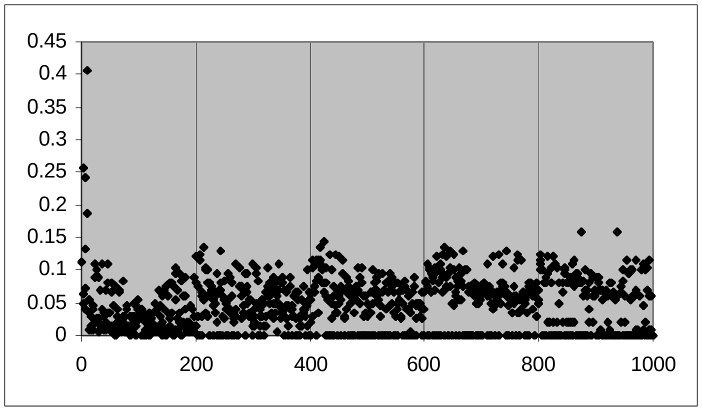


Figure 5c: Deviation of actual bid from apriori optimal - belief algorithm without reset



Figures 1 to 5 give the results for these 5 environments. In each case, graph (a) plots cumulative profit against time, averaged over 100 runs, for the algorithm with reset, and a purely belief-based algorithm. Graphs (b) and (c) plot the absolute difference between *Opt(r)* and the agent's actual bid[8] . In graph (b), the agent is using reset and switching, while in graph (c) it is using belief-based learning only.

(7) Discussion and Future Research

Figures 1-3 show that the belief resetting algorithm significantly outperforms the non-resetting algorithm in environments 1-3. Graph (b) in each case shows that the algorithm correctly identifies changes to the underlying distribution of equilibrium price, and resets the beliefs. As a result of this, the algorithm rapidly returns to make bids close to the apriori optimal, while the algorithm without resetting (graph (c) in each case) is unable to do this. In the run shown in 3(b), the algorithm also incorrectly identified a change of distribution, and reset at round 508. It is inevitable that this will occur on occasion, due to the law of probabilities. However, as graph 3(a) shows, the algorithm performs well even taking into account these occasional misidentifications.

In environments 4 and 5, the resetting mechanism performs almost identically to the pure belief –based approach, as figures 4 and 5 show. As the mean of the distribution does not change, only the standard deviation, it is harder for the algorithm to identify

[8] For example, when p^t is uniformly distributed over [0,1], *Opt(r)=(1+r)/2*.

that a change has taken place. In environment 5, where the standard deviation decreases, it cannot identify the change at all. Hence, both algorithms perform identically. In environment 4, where the standard deviation increases, the resetting algorithm is sometimes able to detect the change, though very late in the block. Because of this, it performs marginally better than the standard belief-based approach.

It may be appropriate that the algorithm does not detect a structural breakdown when there is a small change in mean, or only a change in standard deviation. In such circumstances, adapting may be more efficient than resetting. Alternatively, it may be that better performance could be obtained by adjusting our test for stability to be stricter. However, by doing this, it increases the chance of false positives, and resets occurring during a stable period, which is clearly undesirable. Further work will be necessary to determine this.

The algorithm can also be improved by the addition of appropriate rules of thumb to use at times when a structural breakdown occurs. Such rules of thumb may be based on experience from bandwidth trading data (the distribution of supply and demand schedules), and human trading expertise.

The specific details of which statistical test to use and which rules of thumb to use will need more experimentation to answer. However, we believe that the general algorithm design outlined in this paper provides a promising insight into the way to design trading agents which behave optimally in statistically noisy environments.

References

[1] Erev I. and Roth A. E. (1997), "On the need for low rationality, cognitive game theory: Reinforcement learning in experimental games with unique, mixed strategy equilibria", mimeo, University of Pittsburgh.

[2] Friedman E. J. and Shenker S. (1998), "Learning and Implementation on the Internet", mimeo, Rutgers University.

[3] Fudenberg D. and Levine D. (1997), "Theory of learning in games", mimeo, University of California, Los Angeles.

[3] Lehr, W. and McKnight, L. (1999), "Next Generation Internet Bandwidth Markets", To appear in *Communications and Strategies.*

[5] McKelvey R. and Palfrey T. (1995), "Quantal response equilibria for normal form games", *Game and Economic Behaviour*, **10**, 6-38.

[6] Rosenschein J., and Zlotkin G. (1994), *Rules of Encounter*, The MIT press, Cambridge, USA.

[7] Roth A. E. and Erev I. (1995), "Learning in extensive-form games: Experimental data and simple dynamic models in intermediate term", *Games and Economic Behaviour*, **8**, 164-212.

[8] Rustachini A., Satterthwaite M. A., and Williams S. R. (1994), "Convergence to efficiency in a simple market with incomplete information", *Econometrica*, **62-5**, 1041-1063.

[9] Silvey S. D. (1975), *Statistical Inference*, Chapman and Hall Publishers, London.

[10] Vulkan N. (1998), "An economist's perspective on probability matching", Forthcoming, *Journal of Economic Surveys*.

[11] Vulkan N. (1999), "Economic Implications of Agent Technology and E-Commerce", *The Economic Journal*, **109-453**, F67-F90.

Affiliation of author

Footnotes

[1] Department of Economics, University of Bristol, 8 Woodland Road, Bristol BS8 1TN. E-mail: N.Vulkan@Bristol.ac.uk

[2] Agent Technology Group, Hewlett Packard Labs, Filton Road, Bristol BS12 6QZ. E-mail: cwp@hplb.hpl.hp.com

3 There is also a large literature on learning in financial markets. However, this literature focuses on the transmission of private information via the price mechanism. Since we treat bandwidth as a perishable good, this issue is not relevant in our setting.

[4] Alternatively, bids can be aggregated into supply and demand curves. The interval [a, b] now correspond to the intercepting section of the two graphs.

[5] That is, a strategy in the one-shot double auction game.

[6] That is, $s(r)-r$ is a decreasing function of r.

[7] An underlying assumption of the belief model is that the events eq^t are i.i.d. (although they are allowed to be time dependent). If, however, eq^t is dependent on previous outcomes, say on eq^{t-1} then this assumption is clearly violated. The purpose of this class of rules, is therefore to detect and therefore capitalise on such dependencies.

[8] For example, when p^t is uniformly distributed over [0,1], $Opt(r)=(1+r)/2$.