Journal of Economic Dynamics and Control 18 (1994) 61–96. North-Holland

Characterizing effective trading strategies

Insights from a computerized double auction tournament*

John Rust
University of Wisconsin, Madison, WI 53706, USA

John H. Miller
Carnegie Mellon University, Pittsburgh, PA 15213, USA

Richard Palmer
Duke University, Durham, NC 27706, USA
Santa Fe Institute, Santa Fe, NM 87501, USA

Received January 1992, final version received September 1992.

This paper presents a comparative analysis of 30 computer trading programs that participated in a double auction tournament held at the Santa Fe Institute in 1990 and 1991. Our objective is to characterize the form of effective trading strategies in double auction markets. We find that a simple rule-of-thumb is a highly effective and robust performer over a wide range of trading environments, significantly outperforming more complex algorithms that use statistically-based predictions of future transaction prices, explicit optimizing principles, and sophisticated 'learning algorithms'.

Correspondence to: John Rust, Department of Economics, University of Wisconsin, Madison, WI 53706, USA.

*We are grateful for the generosity of the Santa Fe Institute and its Economics program for providing the facilities, salary, administrative, and computational support that made this research feasible. John Rust is grateful to the Alfred Sloan Foundation for providing tournament prize money, and National Science Foundation grants SES-8721199, which provided computer hardware, and SES-9010046, which provided funding for matching human experiments at the Experimental Science Laboratory at the University of Arizona. International Business Machines provided funding for tournament organizational expenses. John Miller would like to acknowledge an equipment grant from Sun Microsystems. We received many helpful comments from participants in seminars at the University of Oslo, Stockholm, and the London School of Economics.

1. Introduction

The double auction (DA) market is the primary trading institution for many types of commodities and financial instruments in organized markets around the world. A classic example of a DA market is the trading pit of the Chicago Board of Trade where commodity traders call out offers to buy (*bids*) or offers to sell (*asks*) attempting to negotiate the best possible deals. DA markets have remarkable efficiency properties which have been documented in numerous laboratory experiments using human subjects. By assigning human subjects *tokens* with fixed *redemption values* and *token costs*, well-defined supply and demand curves can be constructed. The intersection of these curves defines the price and quantity at which neoclassical economic theory predicts trading will occur, the *competitive equilibrium* (CE) solution. The complication is that in most experimental markets each trader only knows their own token values: no single trader has enough information to determine the market supply and demand curves in order to compute the CE. As a result, traders in DA markets face a complex sequential decision problem: how much should they bid or ask for their own tokens, how soon should they place a bid or ask, and under what circumstances should they accept an outstanding bid or ask of some other trader? Since other traders' token values are private information, each trader must try to make inferences about these values from the unfolding history of observed bids, asks, and acceptances. This information has considerable value, and we would ordinarily expect traders to delay making bids and asks so they have a chance to learn about market conditions. However, experimental DA markets run for a fixed amount of time, typically several minutes. This creates a difficult trade-off, for if traders spend too much time looking for a good deal, they may find themselves locked out of the market without trading anything.

Modern economic theory has attempted to explain observed trading behavior in DA markets as the rational equilibrium outcome of a well-defined game of incomplete information. The 'null hypothesis' is that observed trading behavior is a realization of a Bayesian–Nash equilibrium (BNE) of this game. This approach treats traders as having initial prior distributions about the unknown token values of their opponents, and assumes that traders have common knowledge of each other's prior beliefs. In a BNE, traders also have common knowledge of the trading strategies used by their opponents. In particular, each trader knows that his opponents use strategies that maximize their expected trading profits given any realized history of the game: a BNE is an N-tuple of trading strategies that are mutual best responses. These strategies will ordinarily be nontrivial functions of the entire realized history of the game since all traders continually update their beliefs about their opponents' token values according to Bayes Rule.[1]

[1] For example, in *Markov Equilibria* strategies depend on the trading history via each trader's posterior distribution of opponents' token values.

Unfortunately, due to the inherent complexity of continuous-time games of incomplete information, it is extremely difficult to compute or even characterize these equilibria. As a result, relatively little is known theoretically about the nature of equilibrium trading strategies.[2] Although experimental studies have provided considerable empirical evidence on trading *behavior*, they have failed to cast light on trading *strategies* which are essentially unobservable.

In order to try to shed some new light on the problem we sponsored a computerized DA tournament. Thirty entrants submitted computer programs playing the roles of buyers and sellers in a simplified, discrete-time DA market, vying for cash prizes totaling $10,000 distributed in proportion to the trading profits earned by their programs over the course of the tournament. In exchange, we obtained a computerized DA market that serves as a unique laboratory for studying trading strategies *and* their implied behavior.

The use of computer tournaments to get insights on complicated dynamic games is somewhat unorthodox but not without precedent. Axelrod (1984) sponsored a tournament to study the repeated prisoner's dilemma game. The results of that study were surprising: the winning program, *Tit for Tat*, turned out to be one of the simplest strategies, significantly outperforming much more sophisticated strategies employing statistical predictions, formal optimization principles, and punishment strategies suggested by game-theoretic analyses.[3] We obtain a similar result in our DA tournament: the winning program, submitted by economist Todd Kaplan of the University of Minnesota, turned out to be simple rule-of-thumb which was one of the shortest programs submitted to the tournament. The strategy proved to be a highly effective and robust performer over a wide range of trading environments, significantly outperforming much more complex strategies that used statistically-based predictions of future transaction prices, explicit optimizing principles, and sophisticated 'learning algorithms'. The winning strategy can be described quite simply: *wait in the background and let others do the negotiating, but when bid and ask get sufficiently close, jump in and 'steal the deal'.*

[2]To our knowledge, the only available results for the continuous DA are due to Wilson (1987), who provided a partial characterization of a set of strategies that satisfy necessary conditions for a BNE. In Wilson's equilibrium, 'traders are endogenously matched for transactions via a signalling process using delay as the primary signal' (p. 412). However, a recent analysis by Cason and Friedman (1992) shows that beyond the prediction of high *ex post* efficiency, almost all the other predictions of Wilson's equilibrium are inconsistent with the behavior of human traders in laboratory experiments.

[3]More recently, Selten, Mitzkewitz, and Uhlich (1990) conducted a tournament for computer programs in a twenty-period repeated Cournot duopoly game. They also found that the system tended to converge to efficient cooperative solutions using fairness criteria to coordinate on 'ideal' points along the Pareto-efficient frontier. The winning strategies achieved efficient cooperative solutions using a 'measure for measure' policy that might be thought of as a generalized form of *Tit for Tat*.

Section 2 provides a brief description of the tournament and our implementation of a computerized DA market. Section 3 provides a taxonomy of the trading programs submitted to the tournament. We classify programs along several dimensions, including their relative complexity, adaptivity, randomness, and use of explicit optimization techniques. Section 4 summarizes the results of the 1990 (cash) tournament and subsequent noncash 'scientific tournaments'.[4] The tournament results reveal the robustness and clear superiority of the simple 'wait in the background strategy' over a wide range of environments. Section 5 presents some conclusions and directions for further research.

2. Structure of the DA tournament

The trading programs used in this study were submitted in response to advertisements for a 'Double Auction Tournament' held at the Santa Fe Institute in March 1990. Cash prizes totalling $10,000 were offered to a maximum of 100 entrants in proportion to the trading profits earned by their programs over the course of the tournament. In addition to prize money and wide publicity, a substantial effort was devoted to make the programming and debugging of trading strategies as easy as possible. This included development of the *Santa Fe Token Exchange* (SFTE) which opens at the start of each hour for token trading over the worldwide Internet computer network.[5]

The computerized DA was implemented via a *message-passing protocol* that specifies the form of allowable messages that programs can send, such as bids, asks, and buy and sell orders. Entrants were provided with a simple 'skeleton' trading program (written in C, Fortran, and Pascal) which handled all the message-passing housekeeping, allowing them to focus on the logic of their strategies, rather than on programming details. A central *monitor program* coordinates the trading process by communicating with all of the trading programs, executing their buy and sell orders and relaying their bids and asks to the other traders. Trading programs (which could also be interfaces to human traders) communicate only with the monitor and not directly with each other as illustrated in fig. 1. It is important to note that the monitor program is only a clerk: it is not an 'auctioneer' and has no market-clearing authority.[6]

[4]Since the focus of this paper is on the 'micro' behavior of individual trading strategies, our summary of the tournament results is intentionally brief. See Rust, Miller, and Palmer (1992) for a more in-depth analysis of the aggregate market behavior implied by these strategies.

[5]Many entrants reported that the SFTE was useful for refining their trading programs in advance of the actual tournament. We also distributed 'free-ware' to allow entrants who did not have Internet access to set up their own local token exchanges.

[6]The monitor does enforce trading rules, and can impose upper and lower price limits. It also has the authority to censor illegal or late messages, although no cpu-time limits were imposed in the actual tournament.

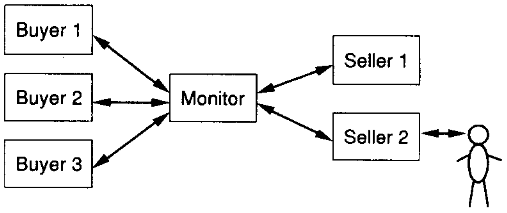

Fig. 1. Interplayer communication via the monitor.

The structure of our computerized DA market is very similar to the continuous-time experimental DA markets described in section 2. The major differences are 1) time is discretized into alternating *Bid/Ask* (BA) and *Buy/Sell* (BS) steps, and 2) transactions are cleared according to 'AURORA rules'. The DA market opens with a BA step in which all traders are allowed to simultaneously post bids and asks. After the monitor informs the traders of each others' bids and asks, the holders of the *current bid* (highest outstanding bid) and *current ask* (lowest outstanding ask) enter into a BS step.[7] During the BS step, either trader can accept the other trader's bid or ask. If an acceptance occurs, a transaction is executed.[8] A *trading period* is simply a set of *S* alternating BA and BS steps.

The discretization of time was adopted to simplify the programming of trading strategies and improve the synchronization of communications between traders and the monitor in a multiprocessing or network computing environment where delays may vary from trader to trader and moment to moment. In a continuous-time environment 'faster' traders have an inherent advantage. This speed advantage may arise due to communication delays (e.g., simultaneous messages sent from a trader in Japan and Chicago may arrive at different times at a central computer in New York) or due to processing delays (e.g., machines may be able to recognize and respond to certain conditions faster than humans). By discretizing time and setting sufficiently wide response-time limits, we can effectively guarantee that all traders have equal trading opportunities. In the limit, our implementation of a discrete-time DA market is not restrictive since a continuous-time trading environment can be arbitrarily well approximated by a discrete-time environment with very many short trading intervals.[9]

The AURORA rules were motivated by the proposed AURORA computerized trading system developed by the Chicago Board of Trade. AURORA

[7]If a current bid (ask) does not exist, then all buyers (sellers) enter into the BS step.

[8]If both parties accept each other's offers, the monitor randomly chooses between the current bid and ask to determine the transaction price.

[9]The same point applies to price units in our DA market, which were rounded to the nearest integer.

rules stipulate that only the holder of the current bid or current ask are allowed to trade. We adopted these rules as a substitute for *ad hoc* tie-breaking rules which are necessary in discrete-time trading environment when several traders are able to simultaneously accept an outstanding bid or ask. Presumably the Chicago Board of Trade had a very different motivation for considering the AURORA rules.[10] In our case, we experimented with alternative sets of trading rules and found that the AURORA rules produced a more interesting DA game, as well as one that was perceived as 'fairer' by participants. Strategically, our DA breaks into a series of two simultaneous move subgames: a multiplayer game to acquire the current bid or ask in a BA step, and a two-player game involving a binary accept/reject decision in a BS step. In early human experiments using random tie-breaking rather than AURORA rules, we found that traders often expressed frustration that trade execution seemed more a matter of luck than strategy after repeatedly losing random tie-breaks for the acceptance of an outstanding bid or ask. On the other hand, one might criticize the AURORA rules on the grounds that it makes it harder for traders to remain in the background by forcing them to 'show their hand' by posting a winning bid or ask before being allowed to trade. We have found, however, that these rules do not place a significant constraint on background traders: they can still stay quietly in the background for most of the trading period, jumping in the moment they detect an attractive bid or ask. Indeed, this is precisely the strategy followed by the winner of the tournament.

An individual DA *game* is divided into one or more *rounds*, and each round is further divided into one or more *periods*. A single period of the DA game consists of a fixed number of alternating BA and BS steps as described above. A single round of the DA game consists of a fixed number of periods. Token valuations are held constant for all trading periods within a given round, but are allowed to change between rounds. Similarly, the set of traders is held constant for all rounds within a given DA game. The reason for structuring DA games to have multiple rounds and periods within rounds is to control traders' abilities to learn about their opponents. DA games with many periods allow traders to learn the value of each others' *tokens*, while DA games with many rounds allow traders to learn about each others' *strategies*.

At the start of a DA game the monitor broadcasts *public information* to the traders, including the number of buyers and sellers and their identities, the number of rounds, periods, and time steps, the number of tokens each agent will have, and the joint distribution F from which the traders' token values are drawn. Next, the monitor sends each trader a packet of *private information*, namely, their realized token values. Since public information is provided by

[10]Since AURORA rules have the effect of making all transactions publicly observable, they may have been designed partly in response to trading abuses that were uncovered in the Chicago Exchanges in the late 1980's.

a simultaneous broadcast to all traders, it serves as a means of ensuring that traders have common knowledge about all relevant game parameters. The joint distribution F was communicated to traders using a four-digit *gametype* variable. Token values are represented by T_{jk}, where j indexes the trader and k indexes the token assigned to the trader. Tokens are randomly generated according to

$$T_{jk} = \begin{cases} A + B + C_k + D_{jk}, & \text{if } j \text{ is a } \textit{buyer} \\ A \phantom{{}+B} + C_k + D_{jk}, & \text{if } j \text{ is a } \textit{seller}. \end{cases} \tag{1}$$

where[11] $A \sim U[0, R_1]$, $B \sim U[0, R_2]$, $C_k \sim U[0, R_3]$, and $D_{jk} \sim U[0, R_4]$. Notice that when $R_1 = R_2 = R_3 = 0$, we have the standard independent private-values model where tokens are independently uniformly distributed on the interval $[0, R_4]$. A *gametype* equal to 0 indicates an environment where redemption values were generated by an unspecified process.

To insure that tournament earnings were not due to a series of lucky token draws, we developed a sampling scheme that guaranteed that all trading programs had equal surplus endowments with probability 1.[12] Once a random set of token values was drawn according to the sampling scheme given in (1), trading programs were randomly selected to play in a set of N games (where $N = 30$ is the total number of entrants) subject to the constraints that no program played a copy of itself in the same game and all programs played all positions (B1, B2, S1, S2, etc.) an equal number of times. After this set of N games was completed, the scheme was repeated with a new set of token values. It follows that the differences in the trading profits earned by the traders can be ascribed to differences in their trading ability, since each program received the same endowment of tokens and encountered roughly the same collection of opponents in a large number of replications of the DA game.

Tournament entrants were told that their programs would be placed in an unspecified number of alternative *environments*. Each environment is a complete specification of all relevant parameters of the DA game listed in table 1. Participants were informed of potential ranges for each of the parameters, but were not given any specific advance information about how the actual environments would be selected. The actual DA tournament consisted of playing a large

[11] Each of the four digits of the *gametype* variable correspond to $\{R_1, \ldots, R_4\}$ according to the base-3 coding, $R_i = 3^{k(i)} - 1$, where $k(i)$ is the ith digit of *gametype*.

[12] In the case of two trading programs that were only programmed to play one side of the market, a *Skeleton* stand-in trader was substituted in the games they refused to play. There are slight variations in actual token endowments caused by the fact that one program occasionally died midway through a trading period, resulting in forfeiture of its potential surplus in the remaining periods of the game.

Table 1
DA trading environments.

Parameter	Environment									
	BASE	BBBS	BSSS	EQL	LAD	PER	SHRT	SML	RAN	TOK
gametype	6453	6453	6453	0	0	6453	6453	6453	0007	6453
minprice	1	1	1	1	1	1	1	1	1	1
maxprice	2000	2000	2000	2000	2000	2000	2000	2000	3000	2000
nbuyers	4	6	2	4	4	4	4	2	4	4
nsellers	4	2	6	4	4	4	4	2	4	4
ntokens	4	4	4	4	4	4	4	4	4	1
nrounds	2	2	2	2	2	6	2	2	2	2
nperiods	3	3	3	3	3	1	3	3	3	3
ntimes	75	50	50	75	75	75	25	50	50	25
games	1624	1624	1624	1624	1624	1624	1624	3428	1624	1624
Games/player	56	56	56	56	56	56	56	112	56	56
Periods/player	336	336	336	336	336	336	336	336	336	336
Conversion ratio ($\times 10^{-4}$)	6.11	8.95	9.73	3.48	3.57	6.97	7.04	6.32	1.04	20.6

number of DA games in ten separate environments presented in table 1. Each of the ten environments were allocated \$1,000 prize money, and separate conversion factors were calculated to translate token profits into dollar earnings. The conversion factor $c(i)$ for environment i is the ratio $1000/TS(i)$, where $TS(i)$ is the total surplus available in environment i. Thus, if trading in the tournament was 100% efficient (i.e., total profits = total surplus), then payouts in the tournament would be exactly \$10,000. However, actual dollar payments in the tournament were \$8,937, corresponding to an 89% efficiency ratio. Overall, we ran a total of 18,114 games in the ten separate environments, comprising 30,312 individual periods of play.

One can see from table 1 that the tournament subjected programs to a wide range of trading conditions. The base case (BASE) was an environment similar to the one used in pre-tournament trials at the SFTE. Other environments include duopoly and duospony (BBBS and BSSS), an environment where all traders receive the same token values shifted by a common random constant (EQL), an independent private values environment where each trader's token is an *IID* draw from a uniform distribution (RAN), a single-period environment that prevented traders from learning from previous market outcomes (PER), a 'high-pressure' environment where the traders' time allotment was very short (SHRT), and an environment where each trader was only assigned a single token (TOK). Our intention was to force programs to compete under a broad range of conditions in order to provide a rigorous and comprehensive test of their effectiveness.

3. Summary of DA trading programs

Thirty programs were submitted to play in the first DA tournament in March, 1990, in which the $10,000 prize money was paid out. Table 2 presents a summary of thirty-four entries, the thirty DA entries listed by the name of the participant(s) who submitted the program together with four additional programs developed by the authors to serve as experimental controls in subsequent 'scientific tournaments'.[13] Of the thirty entries, fifteen were from economists, nine from computer scientists, three from mathematicians, and the remaining three were from an investment broker, a professor of marketing, and a joint entry from two cognitive scientists.[14] Three of the entries were outgrowths of research papers describing formal models of DA trading behavior, including the Ledyard–Olson entry [based on Easley and Ledyard (1992)] and Friedman–Kennet [based on Friedman (1991)]. Several of the entries emerged from working groups which co-developed sets of strategies, in some cases pre-testing them in 'local tournaments' using our double auction software. These groups include seven entries from the Economic Science Lab (ESL) at the University of Arizona, three from the University of Minnesota, and two each from the University of Colorado (Economics) and Carnegie Mellon University (Computer Science). All of the entrants programmed their strategies by filling out bid/ask and buy/sell subroutines of a skeleton trading program supplied by the organizers. Although versions of the skeleton program were available in C, Fortran, and Pascal, almost all of the entries (26 out of 30) were programmed in C. Only two were written in Fortran and two in Pascal.

The first step in our analysis was to classify the programs along several dimensions:

simple *vs.* complex,
adaptive *vs.* nonadaptive,
predictive *vs.* nonpredictive,
stochastic *vs.* nonstochastic,
optimizing *vs.* nonoptimizing.

[13]Obviously to avoid conflict of interest, the authors did not submit any of their own strategies to the original cash tournament held in March 1990. Among the non-SFI entries, in cases where trading programs were developed by teams of individuals and we were unable to determine the primary author, we substituted a program nickname supplied by the authors. We also received two anonymous entries.

[14]Four of strategies included in table 2 were programmed by the authors and were not included in the March 1990 tournament to avoid conflict of interest. These strategies include a 'skeleton program' provided as a simple example to all entrants, a program *ZI* which implements the Gode–Sunder (1992) 'zero intelligence' strategy, a 'truthteller' program which always places bids and asks equal to its true token valuations, and a 'pricetaker' program which attempts to implement a naive pricetaking strategy. The programs were designed to serve as experimental controls for testing various hypotheses about aggregate market behavior, and are not analyzed here. We refer the reader to Rust, Palmer, and Miller (1992) for analysis of the subsequent 'scientific tournaments' that included all programs listed in table 2.

Table 2

Taxonomy of DA trading programs.[a]

Author/Nickname	Institution	F	L	C	A	S	P	O	CPU
Anon-1	Anonymous	CS	C	·	2	·	1	·	86
Anon-2	Anonymous	CS	C	·	3	·	1	·	89
Jacobson	Carnegie Mellon	CS	C	·	3	X	1	·	86
Ringuette	Carnegie Mellon	CS	C	·	2	X	1	·	85
Golden Buffalo	Colorado	E	C	·	2	X	2	·	88
Silver Buffalo	Colorado	E	C	·	2	X	2	·	88
Lin	Portland State	E	C	·	2	X	1	·	89
Perry	Portland State	M	C	·	3	X	2	·	88
Anderson	Minnesota	E	C	·	2	·	1	·	88
Breton	Minnesota	E	C	·	3	X	1	·	88
Bromiley	Minnesota	MK	F	·	2	·	1	·	90
Kaplan	Minnesota	E	C	·	3	·	1	·	86
Pricetaker	SFI	EM	C	·	2	X	1	·	88
Skeleton	SFI	EM	C	·	2	X	1	·	84
Truthteller	SFI	EM	C	·	1	·	1	·	84
ZI	SFI	EM	P	·	1	X	1	·	82
Exp	Arizona ESL	E	C	·	2	·	2	·	86
Free	Arizona ESL	E	C	·	2	·	2	·	87
Gamer	Arizona ESL	E	C	·	1	·	1	·	84
Max	Arizona ESL	E	C	X	2	·	2	X	157
Max-R	Arizona ESL	E	C	X	2	·	2	X	261
Slide	Arizona ESL	E	C	·	2	X	2	X	96
Terminator	Arizona ESL	E	C	·	2	·	2	·	89
Bolcer	UC Irvine	CS	P	·	2	·	2	·	86
Burchard	Princeton IAS	M	C	X	5	X	2	X	99
Dallaway/Harvey	Sussex	CS	C	X	5	·	1	X	91
Kennet/Friedman	Tulane/UCSC	E	P	X	2	·	2	X	181
Kindred	Duke	CS	C	·	2	X	1	·	89
Ledyard/Olson	Cal Tech	E	C	X	3	X	2	·	187
Leinweber	MJT Advisors	B	C	·	2	X	1	·	87
Lee	British Columbia	CS	C	·	2	X	1	·	88
Staecker	Western Ontario	CS	C	X	3	·	2	X	88
Utgoff	Massachusetts	CS	C	·	2	X	2	·	86
Wendroff/Rose	Los Alamos NL	M	F	·	3	·	2	·	88

[a]F: Field – B = broker, CS = computer science, E = economics, M = math, physics, MK = marketing.
L: Programming Language – C, F = Fortran, P = Pascal.
C: Complex – X if program is 'complex' as defined in section 3.1.
A: Adaptive – Five-level ranking defined in section 3.2: 1 = least adaptive, 5 = most adaptive.
S: Stochastic – X if program makes use of random number generator.
P: Predictive – Three-level ranking defined in section 3.4: 1 = doesn't predict, 2 = predicts market variables.
O: Optimizing – X if program uses an explicit optimization principle.
CPU: Ratio of CPU time consumption to average for all programs.

Table 2 summarizes our classifications for each of the trading programs analyzed in this paper. Given that some of these features are somewhat difficult to define precisely, the remainder of this section provides a more detailed discussion that justifies our classifications in table 2. We believe that this taxonomy is the most useful way to comprehend the variety of strategies submitted to the tournament.

3.1. Complexity

The issue of whether effective trading programs are 'simple' or 'complex' is of substantial importance to this analysis. The DA tournament can be viewed as an *automata game* in which humans write computer programs to serve as their 'agents' to play the DA game. Theorists such as Abreu and Rubinstein (1988) have modelled players of automata games as having two conflicting objectives: they like the monetary rewards earned by their programs, but dislike complexity. In the DA tournament it is evident that the more complex programs took significantly more time to write and debug. In the absence of preferences for computer programming or complexity *per se*, we would expect entrants to write the simplest computer program capable of implementing any given strategy. Thus, the outcome of the automata game depends critically on the perceived relationship between complexity and payoff: Is it the case that extra effort expended in developing a more complex trading program has a payoff in terms of improved performance? The tournament results seem to indicate that the answer to this question is **no**: two of the simplest programs turned out to take first and second place in the tournament, beating the third-best program – a 'complex' program – by a significant margin.

It is important at this point to carefully distinguish the concepts of *trading strategy* and *trading program*. It is entirely possible that human traders may be using complex strategies in 'real time' trading, but may be unable to encode that strategy in a trading program. If humans have limited programming ability, then it is quite possible that writing more complex trading programs could have sharply diminishing or even negative returns. Thus, our results do not necessarily prove that effective trading strategies are simple.

Well-known results from the computer science literature show that it is very difficult to define an objective and unambiguous measure of program complexity.[15] There are many different measures of complexity, including space complexity (e.g., code length) and time complexity (e.g., cpu-runtime). The paradoxes that result from taking particular complexity measures too seriously [e.g.,

[15] The abstract theory of *computational complexity* is not directly applicable here, because it refers solely to the way computational requirements scale with problem size N, such as N^k or $\exp(N)$. But we have, in effect, fixed N.

Blum's (1967) 'Speedup Theorem'] are well-known. But despite the paradoxes, we think that it is extremely important to address the issue of complexity, because models that account for time constraints, computation costs, and 'bounded rationality' are likely to yield more realistic models of human trading behavior.[16]

In table 2 we classify a program as 'simple' if it is relatively short and uses a small number of rules-of-thumb, whereas we classify a program as 'complex' if it uses sophisticated mathematical operations and algorithms, or has lengthy and detailed program code which involves the extensive use of information variables or numerous multiply nested functions, structures, or procedure calls. The strategy of Kaplan is classified as simple since it is relatively short (363 lines), and encodes a small number of rules that require only a few easily computed aggregate market statistics.

An example of a complex program is the program of Mark Staecker, written as part of a senior honors thesis at the University of Western Ontario. Not only is the program among the longer trading programs received (1,378 lines), it is also more complex in its use of auxiliary functions and data structures which record numerous statistics on the trading behavior within the current round and period. Much of the complexity is due to the fact that the program attempts to predict the next high bid, low ask, and equilibrium price using general statistics on past behavior. Then using these three predictions, it decides if a transaction is likely to occur in the next BS step. If so, it tries to win the current bid or ask in the next BA step by placing an 'attractive' bid or ask.

Table 2 also presents a time-based measure of complexity – the average cpu-time consumed by the trading program. Almost all the programs had very similar cpu-consumption, mostly due to the overhead of bookkeeping and communicating with the monitor than from calculating decisions. There is a high correlation between the 'time' and 'space' measures of complexity; according to either criterion there were only seven submitted programs that we judged to be complex. Although the range of complexity was fairly wide, the majority of participants used short, heuristic programs based on a few simple decision rules.

Figs. 2 and 3 present block diagrams of the bidding strategies of the two top programs, Kaplan and Ringuette.[17] The rules for deciding when to place a bid

[16] For example, Abreu and Rubinstein (1988) modelled automata as Moore machines, and defined complexity as the number of states in the *Moore machine*. They showed that in a repeated two-player automata game, in any Nash equilibrium the players will submit programs of equal complexity; in particular, we would never see coexistence of simple and complex programs in a Nash equilibrium of their game.

[17] We omitted diagrams of the ask routines since they are symmetric to the bid routines. We also omitted diagrams of the buy/sell routines since they are both trivial: if the program holds the current bid and can make a profit by accepting the current ask, then accept, else reject.

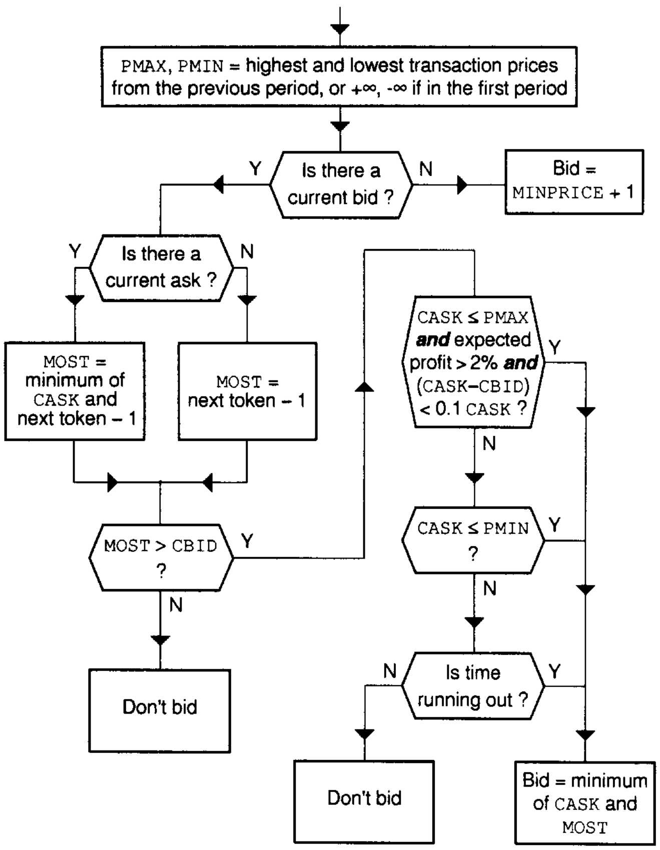
PMAX, PMIN = highest and lowest transaction prices from the previous period, or +∞, -∞ if in the first period
Is there a current bid ?
Y
N
Bid = MINPRICE + 1
Is there a current ask ?
Y
N
MOST = minimum of CASK and next token - 1
MOST = next token - 1
MOST > CBID ?
Y
N
Don't bid
CASK ≤ PMAX and expected profit > 2% and (CASK-CBID) < 0.1 CASK ?
Y
N
CASK ≤ PMIN ?
Y
N
Is time running out ?
N
Y
Don't bid
Bid = minimum of CASK and MOST

Fig. 2. Block diagram of Kaplan's bidding strategy.

or ask are very similar. Kaplan's buyer program places a bid equal to the previous current ask whenever the percentage gap between the current bid and ask is less than 10% (of course, no bid is placed if trade isn't profitable at that price). Ringuette's entry rule is to wait until the first time when the current bid exceeds the current offer less a 'profit margin'. The profit margin is set equal to one fifth of the variable *span* defined as the difference in price between the most expensive token and the least expensive token (plus 10 to account for the case when the player only has one token to trade). Instead of placing a bid equal to the previous current offer, Ringuette's strategy randomly overbids by an amount $\tilde{U} * span/20$, where $\tilde{U}$ is a uniform random deviate.

Fig. 4 presents the block diagram of *Skeleton's* bidding strategy (supplied to all entrants) which Ringuette's program calls as a subroutine (as can be seen in the right branch of fig. 3). Ringuette's version of the skeleton bidding routine is only

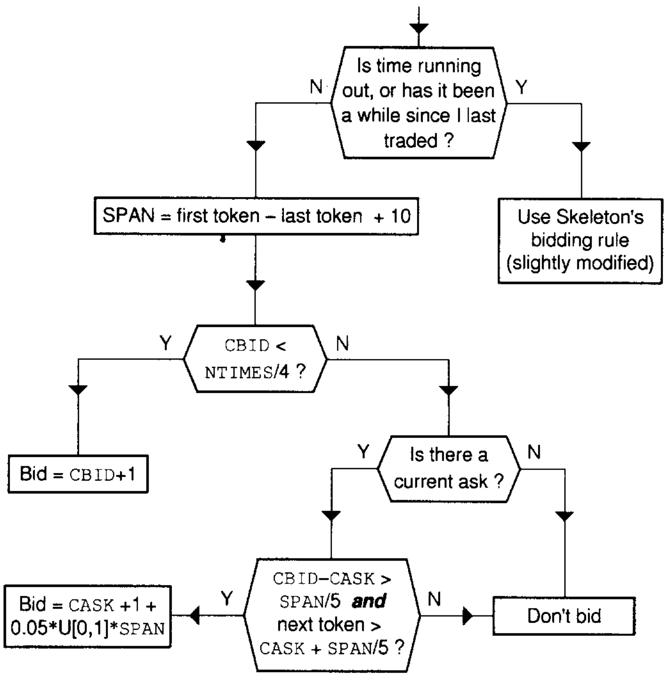
Is time running out, or has it been a while since I last traded ?
N
Y
SPAN = first token – last token + 10
Use Skeleton's bidding rule (slightly modified)
CBID < NTIMES/4 ?
Y
N
Bid = CBID+1
Is there a current ask ?
Y
N
CBID–CASK > SPAN/5 and next token > CASK + SPAN/5 ?
Y
N
Bid = CASK +1 + 0.05*U[0,1]*SPAN
Don't bid

Fig. 3. Block diagram of Ringuette's bidding strategy.

slightly modified from the sample code distributed to all DA participants: if there is no current bid, the program bids the smaller of the current offer or the least expensive token, less a profit margin equal to 30% of the *span* in value between the most and least expensive tokens.[18] If there is a current bid, the program bids a random weighted average of the current bid and the smaller of the current token value and current offer price. This bidding rule puts more weight on the current bid as the number of time periods decreases, starting out at a weight uniformly distributed between 30 and 40% and ending with a weight randomly distributed between 70 and 80%.[19]

The basic idea behind the Kaplan and Ringuette programs can be summarized in one line as: *wait in the background and let others do the negotiating, but when bid and ask get sufficiently close, jump in and 'steal the deal'*. We find it rather striking that the two top programs are so similar despite the fact that they

[18]This is slightly different than the original skeleton that subtracted a random percentage of *span*, where the random percentage was uniformly distributed between 25 and 35%.

[19]This differs from the original skeleton where the weight was always uniformly distributed between 25 and 35% regardless of how many time steps are left in the game.

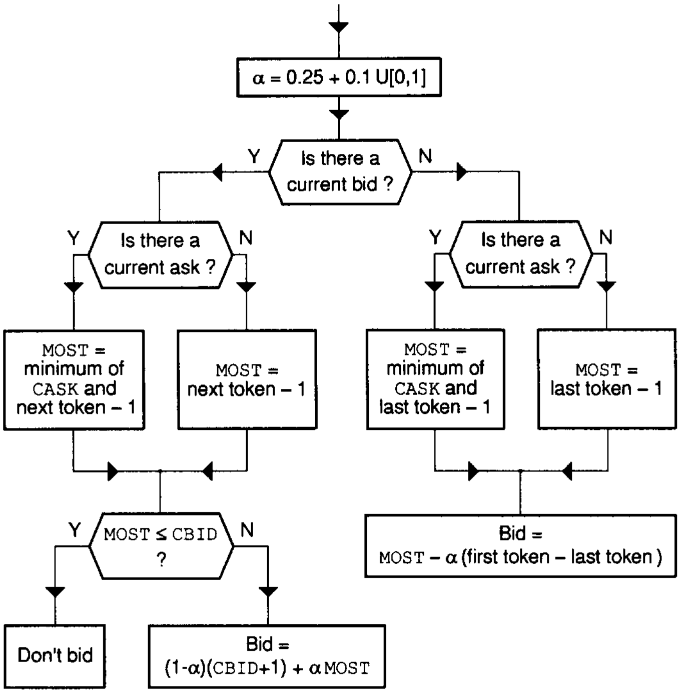

Fig. 4. Block diagram of *Skeleton's* bidding strategy.

were independently developed, which is one confirmation of the robustness of this approach. Although it is tempting to identify the Kaplan/Ringuette strategy as the analog of *Tit for Tat* in Axelrod's Prisoner's Dilemma tournament, Rust, Miller, and Palmer (1992) show that unlike *Tit for Tat*, there is a well-defined sense in which this strategy is not collectively stable.

3.2. Adaptivity

Another important strategic aspect is the level of adaptivity of the various programs. Is it the case that more adaptive programs perform better over a wide class of environments than less adaptive programs? Our results indicate that the answer to this question is also **no**: the adaptive programs were among the poorest performers in the tournament.

Table 2 provides a summary classification of the adaptivity levels of the programs submitted to the tournament. Most of the programs are relatively nonadaptive. In general the adaptive programs were more complex than the nonadaptive programs. In comparison to classifying programs as simple or complex, the adaptive versus nonadaptive classification is much more straightforward. However at the deepest level, there are some fundamental ambiguities here as well. We would certainly want to classify human traders as adaptive to the extent that they improve their trading performance after repeated plays of the game.[20] Thus, we might be tempted to define an adaptive strategy as one which is modified in light of previous trading experience. However, an arbitrary trading rule is just a mapping from the space of publicly observable trading histories into a decision space:

$$f(I_0, H_t) \Rightarrow D(H_t), \tag{2}$$

where I_0 represents the trader's initial private information at the beginning of the tournament, H_t represents the publicly available history at time t, and $D(\cdot)$ represents the set of feasible actions ($\{$ *accept/reject*$\}$ in a BS step or the interval [*minprice*, *maxprice*] in a BA step).[21] The point is that at the most general level, *any* trading rule can be represented in terms of (2). Viewed from this perspective, it is then less obvious how define 'adaptivity'. Since a strategy which is modified in light of previous trading experience can always be rewritten as a fixed function of histories as in (2), in what sense is such a rule adaptive? This suggests that we should classify the adaptivity of a strategy based on the length of history it depends on, but clearly there will be a certain arbitrariness in where we draw the line.

In the context of the DA game, it seems natural to classify a trading rule as nonadaptive if it only depends on the *immediate* history, whereas an adaptive trading rule would depend nontrivially on the *entire* history. However, because people have imperfect memories, it would be difficult to argue that human traders actually make use of the entire history. Thus, at some point we will face a difficult decision of classifying a trading rule that falls in the grey area of not depending on the entire history, but yet uses more than just the immediate history. Given the (*round*, *period*, *step*) structure of our DA game, it is natural to define five increasing levels of adaptivity:

1. Program only uses public information from the current *step*.
2. Program only uses public information from the current *period*.
3. Program only uses public information from the current *round*.

[20] Actually, the extent to which humans modify their trading behavior in light of past experience has never been formally studied (to our knowledge). In joint work with Vernon Smith and Shawn LaMaster at the University of Arizona, we are currently in the process of collecting long-time series data on human trading behavior to better understand the learning process.

[21] If the trader's strategy involves randomization, then f can be interpreted as a conditional probability distribution over $D(H_t)$. Note that the time index t should actually be interpreted as a vector index t = (*game*, *round*, *period*, *step*). This distinction will be important in our subsequent definition of adaptivity.

4. Program only uses public information from the current *game*.
5. Program uses public information from current and previous games.

Level 1 programs are clearly nonadaptive. An example of a level 1 program is *ZI* which makes bid, ask, and buy/sell decisions based only on its current token values without regard to any other information in the current period. Another example of a level 1 program is the *Gamer* program which uses a fixed rule that bids/asks 10% away from its token value and then accepts any profitable counteroffer if it holds the current bid or ask. This is obviously a nonadaptive program since it depends only on information from the current step of the game, and its 10% behavioral parameter does not change as a result of experience from game to game, or even within a game.

An example of an adaptive program is the Neural Net (NN) trader of Dallaway and Harvey, a simplified version of which is illustrated in fig. 5. Their program implements a recurrent neural network (Jordan network) with fourty inputs derived from market information variables feeding twenty 'hidden' units and two output units (one output provides a binary accept/reject decision in a buy/sell step and the other determines the level of the bid or ask in a BS step). The net is recurrent in the sense that the outputs of the hidden layer – lagged by one trading step – are also used as inputs to the hidden layer. Overall, the NN program involves 1,262 correction strength and bias parameters. For fixed connection strengths, the NN program is actually only level 2 adaptive since the net only uses input variables from the current period. However, Dallaway and Harvey employed a genetic algorithm (GA) to evolve the connection strength parameters based on the NN's performance over a series of DA games. Thus, the combined NN–GA algorithm is level 5 adaptive since evolution of the NN parameters implies that the program's decisions are an implicit function of the entire game history.[22]

Another example of an adaptive trading program is the 'cellular adaptive curve-fitter', submitted by mathematician Paul Burchard. The program uses an adaptive curve fitter based on 'cell division' to predict the high bid and low ask at the next step of the trading period. A cell is a recursive structure covering an endogenously determined range of data (bid and asks) from which it predicts future values using a pre-specified function of its observations such as minimum, maximum, or average. A cell 'divides' (spawns a child cell) when the data it contains are both too numerous and too ill-fitting, where the latter is determined

[22]The GA is implemented by Gray-coding the 1,262 connection strengths into a binary string, the 'DNA' file. Multiple copies of the NN program are allowed to play DA games, each associated with their own DNA file. After a fixed set of games, each of the DNA files are replaced by the offspring of two 'parent' DNA strings using crossover and mutation operators. Parents are selected with probability proportional to their 'fitness', where fitness is defined as the average trading efficiency over the games in the previous set.

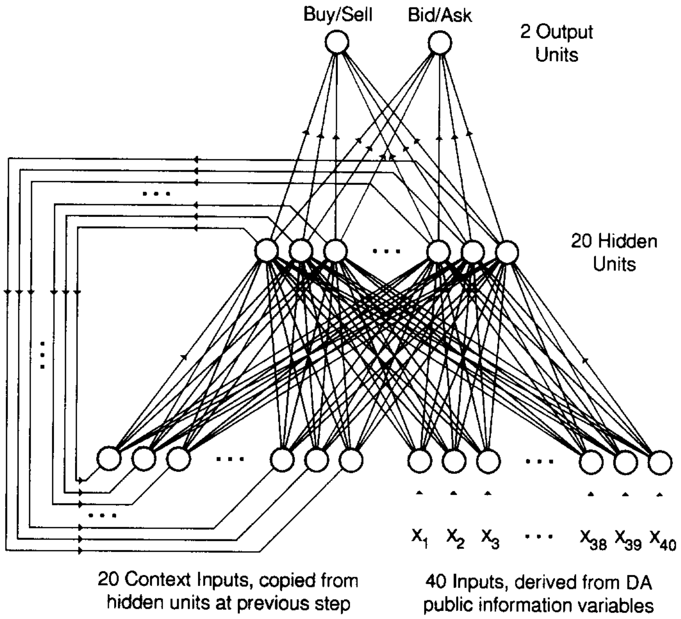
Buy/Sell
Bid/Ask
2 Output Units
20 Hidden Units
x_1 x_2 x_3 ... x_{38} x_{39} x_{40}
20 Context Inputs, copied from hidden units at previous step
40 Inputs, derived from DA public information variables

Fig. 5. Structure of the Dallaway–Harvey neural net.

by a pre-specified statistical metric such as the mean absolute deviation between predicted and actual bids and asks. We classify the program as adaptive because it has the capability to save fitted curves in a file and reload them for use in subsequent games. However, when this feature is turned off, the program is only level 2 adaptive since the predictions of the curve-fitter only use observations on bids and asks from the current trading period.

An example of a program with an intermediate level of adaptivity is the Bayesian player of Friedman and Kennet, which treats the DA as a game against nature (BGAN), updating the parameters of its Gaussian prior distribution as trading progresses. The program is level 2 adaptive due to the fact that the prior is updated using information only from the current period. However, since token values are fixed across periods, information from previous periods is clearly relevant for forming the posterior distribution. To the extent that the program is modified to use information from previous periods, the program would be classified as level 3 adaptive.

3.3. Use of information

One of the advantages of analyzing computer programs is that one can identify the information they use, and perhaps equally important, one can also identify information they ignore.[23] Of particular interest is the *gametype* variable that provides players with common knowledge about the distribution F from which token values are drawn. Only ten programs made use of the prior information provided by the *gametype* variable, and six of these entries were submitted from the Arizona Economic Science Lab. However, it appears that using this information did not confer a great advantage on the trading floor, since only three of these programs made it above the 50th percentile in trading performance.[24] Only three programs made use of prior information about the number of periods and rounds in the game, reflecting the fact that by and large none of the programs attempted to learn and exploit particular features of their opponents (which might be possible in games where the number of periods or rounds are very large). A total of twenty programs made use of prior information about the number of buyers and sellers they would be facing. Some programs, such as Staecker and Ledyard–Olson, used these variables to branch to 'monopoly subroutines' when they were the sole seller or buyer. Although one would have expected that knowledge of the number of buyers and sellers would be a key to good performance in the tournament, the top programs ignored this information while the programs that used it were among the worst performers in the tournament.[25]

Nearly all programs used the basic game parameters such as the number of time steps in each trading period and the total number of tokens: these variables were used together with information on the current elapsed time and the number of trades made so far to determine the amount of remaining time and tokens. Only three programs failed to use this information: *Truthteller*, *ZI*, and Wendroff–Rose. A few programs such as our *Skeleton* and the program by Leinweber kept track of elapsed time but didn't make use of the total number of time steps in the period: instead these programs used a variable which records the time step at which the last trade was made. Only two programs, Kaplan and Dallaway–Harvey, made use of a variable which records the time step at which they made their last transactions. The prevalent use of information about timing suggests that it is crucial for successful performance in the tournament: the worst programs were either too impatient and traded early in the period, or they kept poor track of time and failed to execute all potentially profitable deals by the end of the period.

As for the other public information variables, only the current bid and ask were used by all the programs. Together with traders' private information on their token values, these three public information variables represent a minimal

[23]It is a much more complicated matter to try to characterize which of the variables that the various programs use are in some sense 'most important' in their decision-making.

[24]The exceptions were the programs of Anon-1, Burchard, and Ledyard–Olson.

[25]The exceptions are the programs *Golden Buffalo* and *Silver Buffalo*, and the programs by Anderson, Burchard, Ledyard–Olson, Perry, and Staecker.

information set in the sense that all three are required in order to avoid trading at a loss when playing the role of buyer or seller. Relatively few programs kept track of the identities and individual histories of bids and ask of their opponents. This reflects the fact that none of the programs attempted to predict the individual strategic responses of their opponents. In general, the programs we judged to be complex were also the ones that made the most extensive use of the public information variables. Only the programs of Ledyard–Olson and Staecker made some use of the majority of the information available to them. Surprisingly, fourteen programs didn't even keep track of previous transaction prices. The fact that some programs, such as Ringuette, performed very successfully without using this information seems to confirm Friederik Hayek's (1945) conjecture about the 'remarkable economy of information need to take the right action in a competitive market'. Indeed, it is quite striking that the best programs required very limited information to do well. In summary, we find that in addition to traders' private information about their token valuations, the key public information variables used by the trading programs are:

Current elapsed time
Current bid
Current ask

3.4. *Use of statistical predictions, optimization, and randomization*

A third feature of trading programs that is closely related but distinct from complexity and adaptivity might be called *predictivity*. This measures the extent to which a trading program bases its decisions on predictions of future bids, asks, and transaction prices. Table 2 categorizes programs into one of three increasing levels of predictivity:

1. Program makes no explicit attempt to predict any future quantities.
2. Program explicitly predicts future values of certain aggregate market statistics.
3. Program explicitly predicts future behavior of its individual opponents.

We can see from table 2 that none of the programs attempt to forecast individual decisions of their opponents. Only a few programs fall in the intermediate category of attempting to make aggregate market predictions. Examples in this category are Burchard's adaptive cellular curve-fitter, which attempts to predict the current bid and ask at the next time step and whether a transaction will occur, and Staecker's program, which predicts next current bid, current ask, and competitive equilibrium price. Ledyard's program provides an example where interval rather than point prediction is used. This program makes a prediction of prices p_l and p_u forming the lower and upper support points where transaction prices are expected to occur in the next BS step. The program updates these predictions based on observations of bids, asks, and realized transactions prices. The majority of the programs did not attempt to predict any future quantities, basing their decision rules entirely on past trading outcomes.

Of course, there are a number of cases where the distinction gets rather cloudy. An example is Perry's 'statistical player'. This program makes implicit predictions of future transaction prices by computing a running average of transaction prices (and the associated standard deviations) from the current and previous periods. The statistical player begins by bidding about 2 standard deviations below its estimate of the average price at the beginning of the period, gradually increasing its bids as time in the current period runs out, so that by the end of the period it bids as much as 0.2 standard deviations over its estimate of the average price. Although this program does not make explicit predictions of the next market transaction price, it is clearly treating its current estimate of the average price as its prediction of the market equilibrium price. In general, programs that employ 'adaptive expectations' forecasting procedures (a term to be distinguished from the concept of 'adaptive behavior' defined above) will form their expectations as a function of lagged observations of various public information variables. In such cases it can be unclear whether trading rules that base their decisions on fixed functions of lagged public variables are nonpredictive, or are predictive but using an implicit forecasting rule based on adaptive expectations principles.

We define a *stochastic* strategy as one that relies at some point on an internal randomizing device (such as a uniform random number generator) in order to make a decision. An example of such a program is Breton's entry, which adds a random error term uniformly distributed on the interval $[-2, 2]$ to a target price defined for a buyer as a weighted average of the current bid and the current token redemption value. Overall about half of the entrants included some sort of stochastic device in their strategies. The primary use of randomization is to add noise to the bid or ask functions to prevent the program from being 'recognized' and systematically beaten by others. For example,when Kaplan's nonstochastic program places a bid, it does so at an amount equal to the last current ask. If other programs were able to recognize that they were playing against a Kaplan program, they could easily pre-empt it by bidding just one unit more than the last current ask. In contrast, Ringuette's program uses random overbidding of the last current ask to prevent this kind of pre-emption. The fact that Kaplan's program does better than Ringuette's suggests that its potential exploitability due to its failure to randomize was not an important factor; instead, Ringuette's program seems to have lost more profit by overbidding than it gained by pre-empting others for the right to hold the current bid.

Finally, we define an *optimizing* strategy as one that uses an explicit or implicit optimization principle to guide its behavior. An example is the program *Max*, which chooses a bid or ask to maximize its expected utility using the cumulative distribution function of bids and asks to calculate the probability that a seller/buyer will accept the program's bid/offer. An example of a nonoptimizing program is *ZI*, which randomly bids below its current token value at each BA step. As in the previous cases, the optimizing/nonoptimizing distinction may not always be clear. At a basic level all of the trading programs are presumably using an 'optimization principle', namely: maximize trading profits. However, this is not a sufficient basis to classify all programs as optimizing. Our distinction hinges on whether or not the program uses a more or less formal mathematical procedure to optimize a well-defined objective function as a fundamental part of its decision-making. Most of the programs that we have classified as nonoptimizing are based on simple *ad hoc* rules-of-thumb, or other heuristics that are clearly not attempting any sort of formal optimization. The programs that we classified as optimizing were designed to solve more or less explicit optimization problems, such as the Friedman–Kennet program that solves a Bayesian control problem, treating the responses of the other traders as 'nature'. The one exception was the NN program of Dallaway and Harvey. For fixed connection strengths, their program is just a fixed decision rule that does not use any explicit optimization principle. However, their GA algorithm that modifies the connection strengths does use an implicit optimization rule, namely, to evolve the DNA strings that encode the connection strengths to maximize the program's long-run trading efficiency.

In conclusion, based on the categorizations defined in this section and the tournament results to be presented in the next section, it appears that the best-performing program in the DA tournament can be characterized as follows:

Simple
Nonadaptive
Nonpredictive
Nonstochastic
Nonoptimizing

4. DA tournament results

Table 3 presents the rankings of dollar payoffs made to eligible trading programs in the March 1990 tournament, broken down by environment.[26] Average dollar payouts were \$309 per entry. The top program, Kaplan, earned a total of \$408, \$14 higher than the second-place program of Ringuette. The

[26]The NN program of Dallaway and Harvey was disqualified from the March 1990 tournament because it consistently incurred large losses. The authors submitted a revised version for the subsequent scientific tournament which performed much more satisfactorily.

gaps separating third, fourth, and fifth place were $7.45, $10.51, and $8.63, respectively. While these differences in earnings may not seem economically significant, the average standard deviation in dollar earnings shows that the differences are statistically very significant. Kaplan's earnings are over 2.5 standard deviations higher than Ringuette's second-place earnings, and the gaps separating first from second, third, and fourth places are 3.8, 5.6, and 7.1 standard deviations, respectively. The average standard deviation in profits of $5.75 was only slightly higher than the $5.40 standard deviation in surplus allocations calculated over each player's 3,360 individual periods of play in the ten environments. Note that our procedure for generating tournament games guaranteed that the token endowments of all traders are identical with probability 1. Given the large number of periods of play, an appeal to the law of large numbers allows us to be very confident that differences in traders' earnings reflect true differences in profitability rather than randomness due to player matchings and stochastic elements in the programs themselves.[27]

The bottom rows of table 3 present the Spearman rank correlations between the overall tournament payoffs and the payoffs in each of the ten environments. In addition, we computed Kendall's W-statistic which measures the degree of concordance in all the rankings. The fact that these statistics are so high shows that trader rankings are fairly robust to the choice of environmental parameters. Indeed, it is quite striking that Kaplan's program took first place in seven out of ten environments, coming in second place in environment EQL and third place in environment SHRT. The only place where Kaplan's program did not do well was the environment TOK where traders were endowed with only a single token. At the bottom end of the spectrum, the BGAN (Bayesian game against nature) program of Friedman and Kennet was consistently one of the worst performers in all ten environments. With earnings of $164.30, BGAN is over 10 standard deviations below the earnings of the next highest competitor.[28]

While we are very confident of our ability to distinguish the best and worst programs, we are much less confident about the relative rankings of the middle group of programs. After the large gap separating the fourth- and fifth-place entries (Anon-2 and Ledyard–Olson), the differences in payoffs of the next group

[27]Rust, Miller, and Palmer (1992) demonstrate that Kaplan and Ringuette remain the top-two-ranked programs in variations of the DA tournament which eliminate the worst-ranked traders, and in 'revolutionary tournaments' where the relative fractions of each type of trading program evolve endogenously based on their historical profitability.

[28]BGAN may have suffered as a result of problems converting the program from PC Turbo Pascal to Sun Pascal. Although the converted program compiles without error, its numerical integration routines generate under- and overflow errors at runtime, suggesting a possible incompatibility in its calls to certain functions. The low ranking of the program of Bolcer should be disregarded since the program was only programmed to play the role of seller. In terms of dollar payoffs per game played, Bolcer's program is about equivalent in performance to the programs *Max* and *Terminator* which placed 19th and 20th in overall earnings.

Table 3

Program rankings in the March 1990 double auction tournament.[a]

Trading program	Over-all	BASE	BBBS	BSSS	LAD	EQL	PER	RAN	SHRT	SML	TOK
Kaplan	1	1	1	1	2	1	1	1	3	1	14
Ringuette	2	3	6	3	1	3	3	6	13	3	2
Staecker	3	2	4	7	3	10	2	10	6	2	3
Anon-2	4	17	8	4	4	2	13	13	1	6	1
Ledyard–Olson	5	10	5	8	7	8	6	11	7	4	4
Perry	6	13	2	9	5	11	16	3	9	8	6
Breton	7	7	15	6	10	6	11	7	11	5	5
Anderson	8	15	3	5	14	7	21	9	2	10	8
Anon-1	9	4	7	11	8	14	4	12	5	9	16
Burchard	10	8	10	13	15	16	9	2	4	11	19
Terminator	11	6	21	2	9	18	8	14	18	14	9
Golden Buffalo	12	9	9	10	11	15	18	5	12	17	15
Lee	13	16	14	20	12	13	12	4	8	7	21
Leinweber	14	11	13	19	6	9	14	8	10	12	25
Silver Buffalo	15	14	19	18	16	4	5	15	14	21	10
Slide	16	5	12	17	13	20	7	17	15	13	18
Jacobson	17	12	20	14	19	5	20	16	16	16	7
Bromiley	18	18	18	12	21	19	10	18	17	20	17
Max	19	20	16	16	22	21	17	24	26	15	13
Max-R	20	22	11	21	20	22	22	21	25	19	11
Utgoff	21	19	22	22	18	12	15	23	21	18	23
Kindred	22	21	23	15	17	17	19	19	19	22	29
Free	23	23	17	26	27	24	25	20	20	24	28
Gamer	24	24	24	25	25	26	23	26	24	25	26
Wendroff	25	25	25	27	24	27	24	22	22	27	22
Lin	26	26	26	23	28	25	28	25	23	26	20
Exp	27	28	28	24	23	23	26	29	28	23	12
Kennet	28	27	29	28	26	29	27	27	27	28	24
Bolcer	29	29	27	29	29	28	29	28	29	29	27
Rank correlation	100	87	89	91	94	84	81	87	89	95	72

[a]Average rank correlation: 77,
Kendall's W-statistic: 79,
Marginal significance level: 3×10^{-34}.

of programs are within 1 standard deviation of each other. The next significant difference in payoffs is a $10 gap separating the 9th place entry of Anon-1 from the 10th place entry of Burchard. Even after 3,360 periods, it is clear that we would need many more observations to be confident of the relative rankings of programs between 5th and 9th place. In general, it is impossible to make any reliable performance distinctions if one is only able to observe traders over a small number of periods. The average dollar earnings of 9.4 cents per period of play is dominated by the per period standard deviation in profits of 10.0 cents. Most of the latter variation is attributable to the 9.2 cent standard deviation in surplus arising from traders' random token endowments. Thus, a computerized trading environment is virtually a necessity if one wants to reliably discriminate good traders from bad. It appears that it would be infeasible to make the same sorts of distinctions in markets with human traders given that it takes hundreds or even thousands of periods play before one can be sure that differences in relative performance are statistically significant.

4.1. Analysis of Kaplan and Ringuette

As noted in the previous section, the first- and second-place programs are remarkably similar, both employing the strategy of 'waiting in the background and let the others do the bidding'. Indeed, figs. 6 and 7 show that the behavior of the two programs is also remarkably similar. The figures present the distributions of four quantities: profit, surplus, efficiency, and trade ratio.[29] For comparison, the dotted lines present the corresponding distributions for all traders. Note that the distribution of surplus is (by construction) the same for all traders: the other distributions reflect differences in relative trading ability. The distributions for Kaplan and Ringuette are nearly identical: the main differences are that Ringuette's program sometimes trades at a loss, and succeeds in trading fewer of its profitable tokens, resulting in a mean trade ratio of 99% of its profitable tokens compared to 111% for Kaplan. The efficiency distributions reveal that Kaplan and Ringuette obtain significantly higher efficiency levels than the other traders, primarily by avoiding low profit margins.[30] Notice that the spike at 0% efficiency is approximately 6.5% for Kaplan and Ringuette as opposed to nearly 12% for the other traders. By comparing the spikes at 0 in the distributions of profits and surplus, we see that although Kaplan and Ringuette are given zero endowments about 13% of the time, they come away with zero profits only about 15% of the time as compared to well over 20% for the other traders.

Given the overall similarity in their programs, which factors account for the superior performance of Kaplan's program? We believe the difference can be ascribed to two factors: 1) Ringuette uses a more aggressive rule to trigger when to place a bid or ask, and 2) when Kaplan makes a bid, he does so at the previous

[29]Efficiency is defined as the ratio of profits to surplus, where the latter is computed under the assumption that all transactions occur at the midpoint of the CE price interval. The trade ratio is defined as the ratio of the number of actual number of trades to the number predicted under CE.

[30]The significant difference in mean efficiency between the two programs is sensitive to outliers that occur when CE profits are very small. The figures truncate the efficiency data at 10,000% and separately tabulate the number of $+\infty$ and $-\infty$ cases where CE profits are 0 and realized profits are positive and negative, respectively (cases where realized and CE profits are both 0 are assigned efficiencies and trade ratios of 100%). A better measure of overall trading efficiency is the ratio of total profits to total surplus, which is 119% and 110% for Kaplan and Ringuette, respectively.

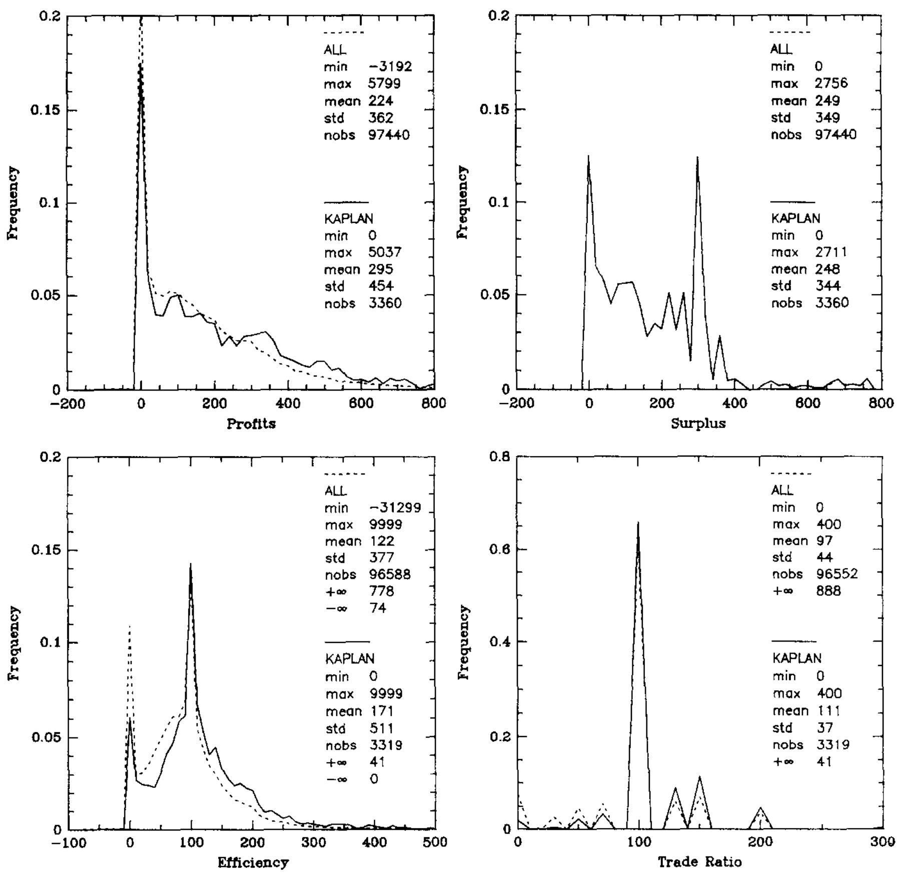

Fig. 6. Payoff distributions for Kaplan.

ask price, whereas Ringuette randomly overbids the previous ask.[31] It is not yet clear which of these features is most responsible for Ringuette's second-place showing. However, it does point out the general lesson that is applicable to all programs, namely, *mistakes in bidding are the principal source of poor trading performance.* Kaplan's more conservative bidding behavior results in a per period profit of 295 (in token units), that is 22 units higher than Ringuette's profit of 273. The only other major difference in the behavior of the two programs shows up in the single-token environment TOK, in which Ringuette's

[31] Kaplan's program is also slightly more adaptive than Ringuette's, using information on minimum and maximum transaction prices from the previous period to help it recognize 'good deals', whereas Ringuette's only uses information from the current period.

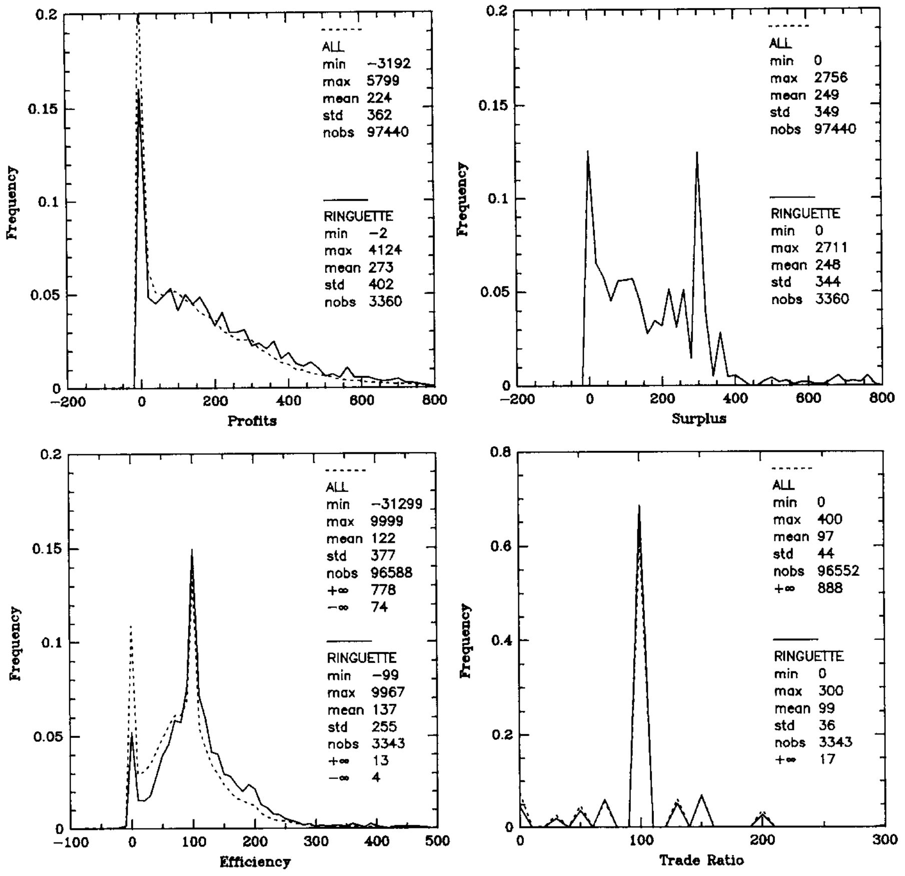

Fig. 7. Payoff distributions for Ringuette.

program takes 2nd place in comparison to a 14th place showing for Kaplan. Although Ringuette's program tends to trade its token less frequently than Kaplan, 84% versus 94%, when it does trade, it succeeds in trading it for a significantly higher price resulting in a trading efficiency ratio of 130% compared to 94% for Kaplan. We are not sure, however, of the precise features of Kaplan's program that cause it to do worse in the single-token environment.

Despite the fact that the increment in Kaplan's per period profits appears to be statistically insignificantly higher than Ringuette's when averaged over all environments, Kaplan's superiority is consistently revealed after a sufficiently large number of periods of play. This difference shows up even more clearly in the long-run or 'evolutionary' tournament described in Rust, Miller, and Palmer (1992).

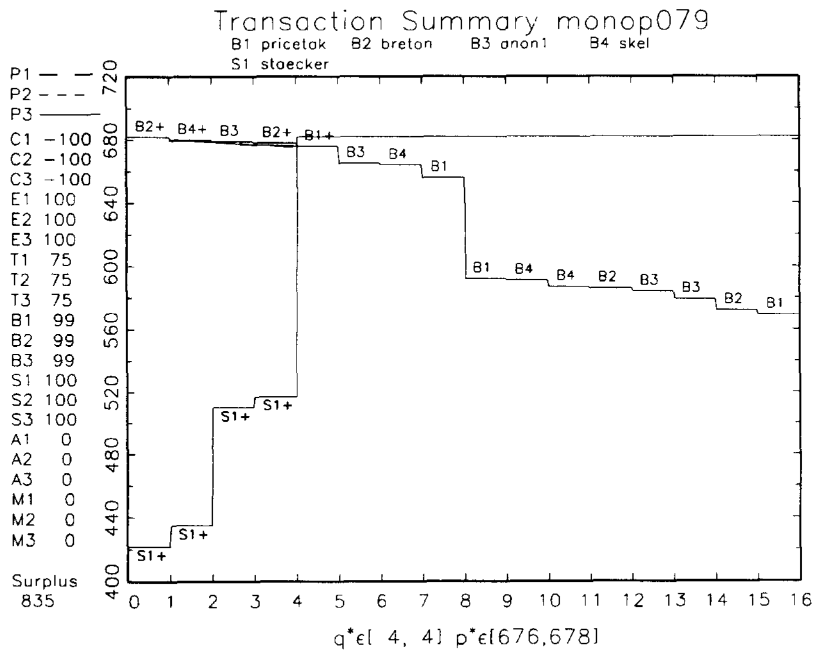

Fig. 8. Staecker as monopolist.

4.2. Analysis of Staecker

The third-place program of Staecker represents an interesting contrast to the first- and second-place programs of Kaplan and Ringuette: we classify Staecker's program as complex, optimizing, predictive, and adaptive. Actually, Staecker's program is level 3 adaptive according to the classification of section 4 since it only uses information from within the current round. Staecker's program was also one of the few programs to include a 'monopoly branch' that attempted to exploit monopoly power by 'walking down the demand curve' in cases where it is the only seller. Fig. 8 illustrates the effectiveness of this approach in a game in which Staecker plays the role of monopoly seller. The price trajectories for each succeeding period of play overlap each other, showing that Staecker's program was able to obtain the first-best outcome of 100% extraction of the available surplus. For comparison, fig. 9 shows the outcome of a game with a less successful monopolist, Leinweber. The values of T1, T2, and T3 on the left side of each figure indicate the step at which the last transaction occurred in each period. When Staecker is a monopolist, the last transaction occurs in the very last period, whereas in Leinweber's case, the last transaction occurs less than halfway through each trading period. This provides a stark illustration of the problems caused by excessive impatience. The values of B1,

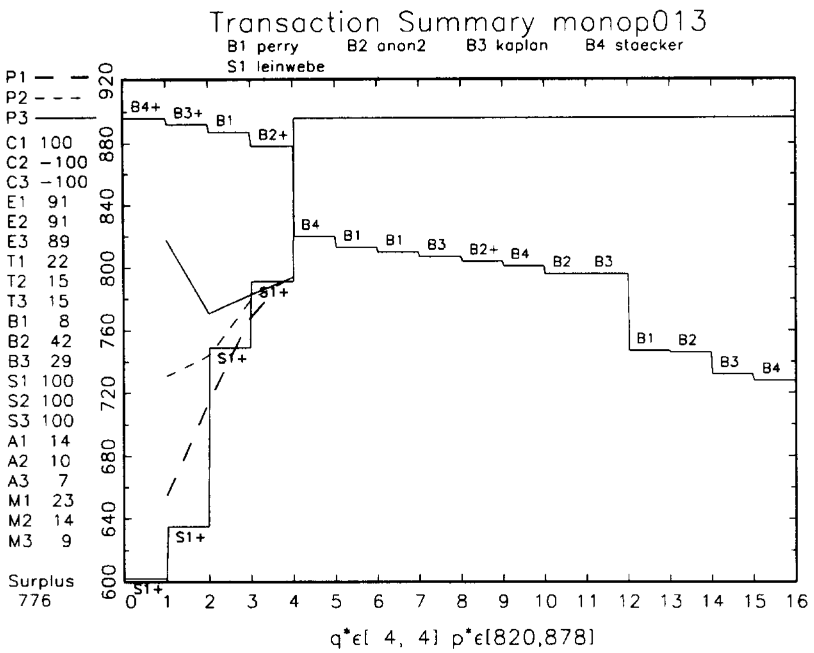

Fig. 9. Leinweber as monopolist.

B2, and B3 indicate the rank correlation coefficients of the order at which buyers transacted with the 'efficient order', i.e., the order of tokens along the demand curve. We can see that in Staecker's case, these correlations are 99%, indicating that he succeeded in 'walking down the demand curve' to extract virtually all surplus.[32] In Leinweber's case, these correlations are well under 50%. For example, in period 3 we can see that by failing to exercise his monopoly power, Leinweber created a situation where an extramarginal trade by B2 (Anon-2) was able to displace a much more potentially profitable trade by buyer B1 (note that the ' + ' signs denote which players made trades).

When there are other competitors on its side of the market, Staecker's program follows a passive bidding strategy until it predicts that a trade is likely to occur, in which case it attempts to 'steal the deal' by bidding one unit less than the winning offer it expects in the next BA step. This is basically similar to the strategy followed by Kaplan and Ringuette (K&R) except for three main differences: 1) K&R stay in the background by either not bidding or bidding the minimum allowable price, whereas Staecker adopts a slightly more activist

[32]Notice from the right hand of fig. 8 that three units of potential profits were lost due to the fact that B1 succeeded in bumping B3 in last BS step.

Table 4
Performance comparison of Kaplan and Staecker.

Case	ALL	BASE	BBBS	BSSS	LAD	EQL	PER	RAN	SHRT	SML	TOK
					Trade ratio						
Kaplan	111	107	111	109	136	114	115	103	104	113	94
Staecker	103	106	98	101	128	109	104	96	90	107	84
Ratio	108	101	113	108	106	105	111	107	116	106	112
					Efficiency						
Kaplan	119	120	121	114	122	119	125	108	121	120	95
Staecker	106	117	111	105	122	102	116	101	109	100	121
Ratio	112	103	109	108	100	111	108	107	111	120	79

bidding strategy by bidding the current ask less 5; 2) K&R's programs trigger a bid whenever the spread between current bid and ask from the *previous* BA step is sufficiently small, whereas Staecker's program triggers a bid whenever the spread between its prediction of current bid and ask on the *next* BA step is sufficiently small; and 3) K&R's programs attempt to 'steal the deal' by bidding an amount equal to the previous current ask (plus a random premium in the case of Ringuette), whereas Staecker's program bids an amount equal to one unit less than its prediction of current ask in the next BA step. Staecker's bidding strategy should involve less overbidding in comparison to K&R since the latter strategy is based entirely on the value of current ask from the preceding BA step, which will typically be displaced by a better ask in the succeeding BA step. However, if Staecker's program makes poor predictions of the next value for current ask, it could end up underbidding and losing the opportunity to trade. Indeed, comparing the mean trade ratios in table 4, we see that Staecker consistently trades fewer tokens than Kaplan. This seems to indicate that the main reason why Kaplan's program is a more efficient trader than Staecker's is that it is more successful in executing its trades.[33] Apparently the premium that Kaplan pays to obtain the current bid is offset by increased execution rates resulting in higher overall efficiency.

4.3. Analysis of Gamer and Max

The *Gamer* program, which ranked 24th in overall payoffs, shows that not all simple, nonoptimizing, nonadaptive, and nonpredictive programs succeeded in

[33]The main exception is the environment TOK, where Kaplan's program appears to do poorly despite the fact that it succeeds in trading its profitable tokens 94% of the time compared to 84% for Staecker. In this case Staecker's program behaves similar to Ringuette's in earning significantly higher profits when it does trade.

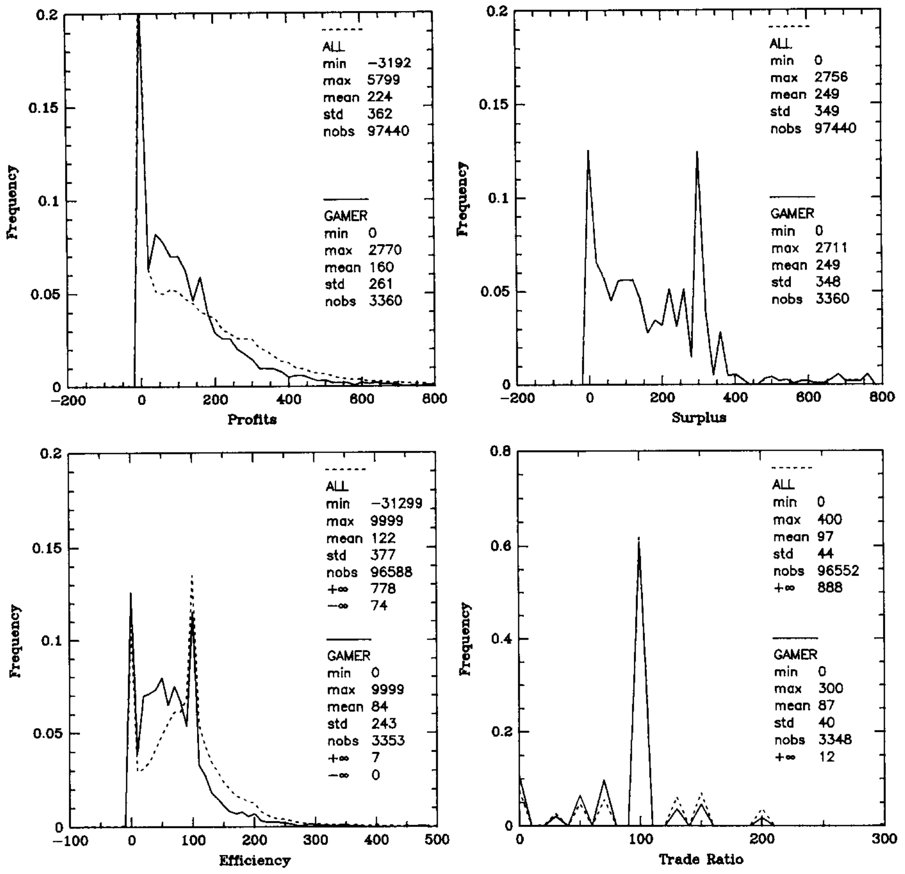

Fig. 10. Payoff distributions for *Gamer*.

doing well in the tournament. *Gamer* always bids an amount equal to 10% below its token value, and accepts any offer that yields a profit. This strategy is similar to the *ZI* and *Truthteller* strategies, and all three strategies performed about equally poorly. Fig. 10 summarizes the behavior of *Gamer* in the ten tournament environments. *Gamer's* average profits of 160 are only about 64% of its mean token endowment of 249. The efficiency distribution shows that *Gamer* earns zero profits in well over 20% of all trading periods, even though its surplus endowment was zero in only 13% of all trading periods. The efficiency distribution is shifted to the left, reflecting the fact that *Gamer* typically settled for significantly less than 100% of its potential profits in the times it did trade. However, the distribution of trade ratios also shows that *Gamer*

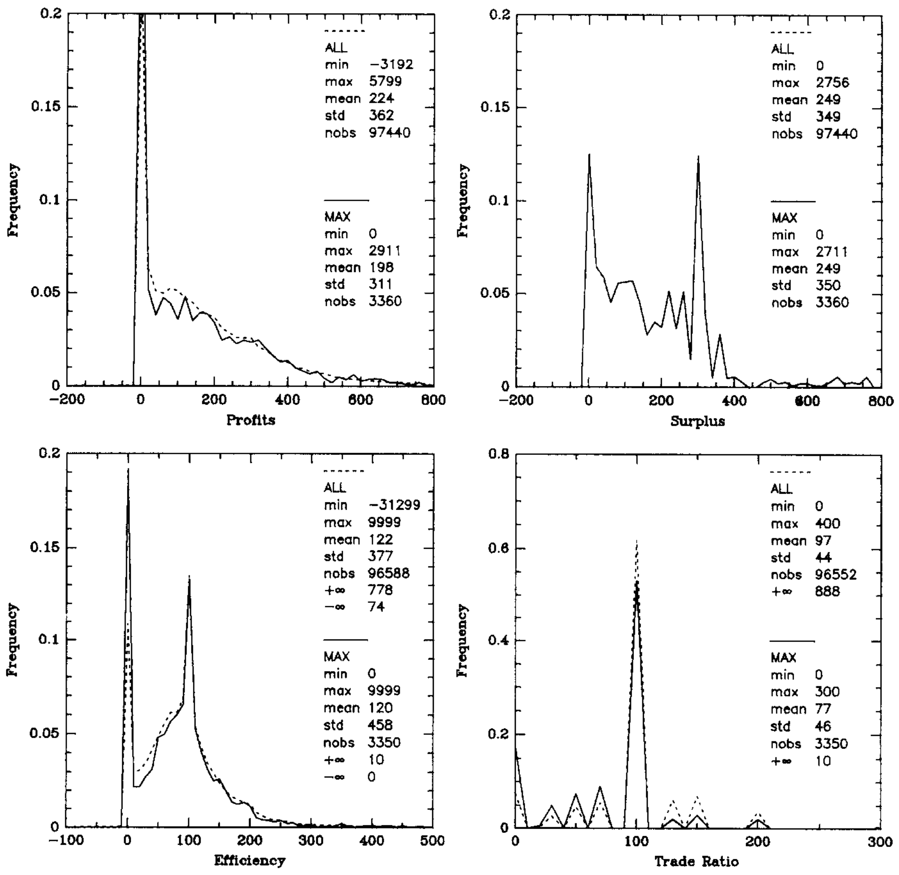

Fig. 11. Payoff distributions for *Max*.

systematically missed out on trading opportunities, trading only 86% of its profitable tokens compared to 97% for the market as a whole. The fact that it traded fewer tokens is probably due to the fact that its 10% underbidding parameter was hardcoded into the program, and not subject to change even if there were potentially profitable offers in the final steps of the period. This creates an inefficiency wedge that locked *Gamer* out of some potentially profitable deals.

The *Max* program provides an example of a complex, optimizing, predictive, and nonadaptive program that also performed poorly. *Max* uses the cumulative distribution of bids and asks, *G*, to calculate the probability that the seller will accept a bid. It is also one of the few programs to make use of the prior information about the token distribution *F* provided by the *gametype* variable.

Given its current estimate of the distribution of bids and offers G, it appears that the intention of *Max's* programmers was to choose an 'optimal bid', namely one that maximizes expected profits computed with respect to its current estimate of G. However, the implementation of this idea in the actual computer code is somewhat different. The optimal bid, B^*, is chosen as the solution to:

$$B^* = \min\left\{ b \mid b < T \text{ and } (T-b)\frac{G(b) \;\; - G(b-1)}{G(b)} > 1 \right\}, \tag{3}$$

where T denotes the program's highest-valued untraded token. Rather than maximizing expected profits, this code seems to be choosing the smallest bid such that conditional expected profits exceeds 1. We would expect such code to lead to significant underbidding. Indeed, the distribution of trade ratios in fig. 11 shows that *Max* trades only 75% of its tokens each period, and that *Max* gets locked out of a trading period without making any trades almost 20% of the time. The distribution of efficiencies indicates that, when it does succeed in trading, *Max* is able to get its fair share of the profits. However, the large spike at 0% efficiency indicates that the primary problem is systematic underbidding causing *Max* to get locked out of the period without having made any trades at all.

5. Conclusions

This paper illustrates the potential value of computers as a tool for studying trading strategies in DA markets. We have analyzed a collection of over thirty computer programs ranging in complexity from simple rules-of-thumb to sophisticated adaptive/learning procedures employing some of the latest ideas from the literature on artificial intelligence and cognitive science. In order to evaluate the programs, we conducted an extensive series of computer tournaments involving thousands of individual DA games, covering a wide range of trading environments and compositions of trading partners. A single program emerged as the clear winner in nearly all of the tournaments and trading environments considered. The winning program, submitted by economist Todd Kaplan of the University of Minnesota, was one of the simplest programs that we studied, and can be characterized as *nonadaptive, nonpredictive, nonstochastic,* and *nonoptimizing*. The basic idea behind the program is to *wait in the background to let others do the negotiating, but when bid and ask get sufficiently close, jump in and 'steal the deal'*. The program makes no use of prior information about the joint distribution of token values, and relies on only a few key variables such as its privately assigned token values, the current bid and ask, its number of remaining tokens, and the time remaining in the current period. The fact that one can design an effective trading program relying on only a few sufficient statistics seems to confirm Hayek's observation about 'the remarkable economy of knowledge that is required in order to take the right action in a competitive market'.

It appears that the success of Kaplan's strategy is due to the fact that if the current bid and ask are close, then it is likely to be the case that either 1) bid and ask are close to the equilibrium price interval, or 2) the current bid or ask are close due a mistake in which either the current bid or ask was made at an unfavorable price. Kaplan's program attempts to 'steal the deal' by placing a bid equal to the previous asking price, but only if it can make a profit at that price. As a result Kaplan's program tends to earn at least a normal profit if case 1) holds, and a super-normal profit if another trader has made a mistake. Since the decision of how much to bid is much more difficult than the binary buy/sell decision, it is not surprising that mistakes in bidding are a primary source of poor trading performance. By staying out of the bidding game, Kaplan's program is able to avoid making bidding mistakes on its own account while capitalizing on bidding mistakes of others.

Another reason for the relatively poor performance of the complex, adaptive, optimizing, and predictive strategies is the inherent difficulty of making accurate inferences in a noisy marketplace given only a limited number of observations on one's opponents. The randomness in traders' token endowments is the dominant source of uncertainty in any particular DA game. The additional variation in profits induced by mistakes or stochastic elements in the trading strategies is insignificant in comparison. As a result, one needs a very large number of observations on trading outcomes to be able to reliably distinguish good traders from bad. It follows that it is virtually impossible to try to recognize and exploit the individual idiosyncrasies of one's individual trading partners unless one is interacting with the same group over a very long horizon. The low signal/noise ratio of realized trading profits combined with the high dimensionality of the space of possible trading histories and trading environments implies that programs based on general learning principles (such as neural networks and genetic algorithms) require many thousands of DA training games before they are able to trade even semi-effectively.[34] Nearly all of the top-ranked programs were based on a fixed set of intuitive rules-of-thumb that encoded the programmer's prior knowledge of the trading process.

This finding suggests that our hopes of using markets populated by 'artificially intelligent agents' in order to evaluate alternative trading institutions may be too ambitious. It's an open question whether other approaches from the

[34]To quote from the entry by Dallaway and Harvey: 'Given that we are doing the equivalent of evolving monkeys that can type Hamlet, we think the monkeys have reached the stage where they recognize that they should not eat the typewriter. If we could have a 4 billion year time extension before handing in the entry, we are completely confident of winning.'

literature on artificial intelligence might be sufficiently powerful to discover effective trading strategies. Our impression, however, is that none of the currently available methods appear capable of the sorts of 'intuitive leaps' that humans seem to make in the process of conjecturing the form of good strategies based on limited trading experience. It is not clear whether our ability in this regard is due to some sort of inherited *a priori* knowledge about markets, or is simply due to the vast superiority of the hardware and software of the human brain.

On the other hand, Rust, Miller, and Palmer (1992) show that the set of top-ranked trading strategies yield a fairly 'realistic' DA market in the sense that their collective behavior matches the key 'stylized facts' observed in human DA experiments. Indeed, if we were to perform a 'Turing test' and ask a human to play a series of DA games and determine whether his opponents were computer programs or humans at remote terminals, we submit that most people would be unable to tell the difference. This finding may not seem surprising if entrants are merely encoding their 'market intuition' in their trading programs. However, given the complexity of the DA game and the sophistication of human intelligence, it is not obvious that human behavior in these markets can be reduced to a few simple decision rules.

In joint work with Vernon Smith, we are presently gathering data comparing the performance of human and computer traders. Clearly, if one is allowed to do experiments that change certain environmental parameters, then it is a relatively easy matter to distinguish human from computer traders. The key distinction is adaptivity. Most of the programs are 'hardwired' to expect a certain range of trading environments: if we start to move out of this range, we would expect to see a serious degradation in their performance relative to humans. However, we don't expect that humans will be able to significantly outperform computer traders in 'standard' trading environments.[35] Although we do not yet have results that we can report, our experience playing against the top-ranked programs leads us to conjecture that most human traders will be unable to consistently outperform simple Kaplan-type trading rules in anonymous markets consisting of short-run encounters with a stream of heterogeneous opponents, although some may eventually be able to learn to exploit certain idiosyncracies in repeated play against a fixed set of opponent programs. Anyone who has actually traded in one of these DA markets realizes that the flow of events is too fast to keep close track of individual opponents and do the detailed Bayesian updating suggested by game theory. Instead, one finds oneself relying on a few simple rules and focusing on a few key statistics not unlike some of the trading programs analyzed in this paper.

[35]This is certainly true in computerized chess, where programs such as 'DEEP THOUGHT' play at the grandmaster level.

References

Abreu, D. and A. Rubinstein, 1988, Finite automata play the repeated prisoner's dilemma, Econometrica 57, 345–352.

Andreoni, J. and J.H. Miller, 1990, Auctions with adaptive artificial agents, Santa Fe Institute working paper 90-01-004.

Axelrod, R., 1984, The evolution of cooperation (Basic Books, New York, NY).

Blum, M., 1967, A machine-independent theory of the complexity of the recursive functions, Journal of the ACM 14, 322–336.

Bollerslev, T. and I. Domowitz, 1992, Some effects of restricting the electronic order book in an automated trade execution system, in: D. Friedman and J. Rust, eds., The double auction market: Institutions, theories and evidence (Addison–Wesley, Redwood City, CA).

Cason, T.N. and D. Friedman, 1992, An empirical analysis of price formation in double auction markets, in: D. Friedman and J. Rust, eds., The double auction market: Institutions, theories and evidence (Addison–Wesley, Redwood City, CA).

Easley, D. and J. Ledyard, 1992, Theories of price formation and exchange in double oral auctions, in: D. Friedman and J. Rust, eds., The double auction market: Institutions, theories and evidence (Addison–Wesley, Redwood City, CA).

Friedman, D., 1991, A simple testable model of double auction markets, Journal of Economic Behavior and Organization, 47–50.

Gode, D.K. and S. Sunder, 1992, Allocative efficiency of markets with zero intelligence (ZI) traders: Market as a partial substitute for individual rationality, Journal of Political Economy, forthcoming.

Hayek, F., 1945, The use of knowledge in society, American Economic Review 35, 519–530.

Marimon, R., E. McGrattan, and T. Sargent, 1990, Money as medium of exchange in an economy with artificially intelligent agents, Computer Science in Economics and Management 5, 109–123.

Palmer, R.G., J. Rust, and J.H. Miller, 1990, Double auction tournament participant's manual, Santa Fe Institute publication.

Rust, J., J. Miller, and R. Palmer, 1992, Behavior of trading automata in a computerized double auction market; in: D. Friedman and J. Rust, eds., The double auction market: Institutions, theories and evidence (Addison–Wesley, Redwood City, CA).

Selten, R.,M. Mitzkewitz, and G. Uhlich, 1990, Duopoly strategies programmed by experienced players, University of Bonn manuscript.

Wilson, R.B., 1985, Incentive efficiency of double auctions, Econometrica 53, 1101–1116.

Wilson, R.B., 1987, On equilibria of bid–ask markets, in: G. Feiwell, ed., Arrow and the ascent of modern economic theory (Macmillan Press, London) 375–414.