# COMPUTER GAMES AND ECONOMICS EXPERIMENTS

Kay-Yut Chen and Ren Wu
*Hewlett-PackardLaboratories,*
*Hewlett-Packard Company,*
*1501 Page Mill Road, Palo Alto, CA 94304-1126*
*Email:kay-yut.chen@hp.com, ren.wu@hp.com*

Keywords: experimental economics, game theory, agent, software platform

Abstract: HP Labs has developed a software platform, called MUMS, for moderating economics games between human and/or robot participants. The primary feature of this platform is a flexible scripting language that allows a researcher to implement any economics games in a relative short time. This scripting language eliminates the need to program low-level functions such as networking, databases and interface components. The scripts are description of games including definitions of roles, timing rules, the game tree (in a stage format), input and output (with respect to a role, not client software). Definitions of variables and the use of common mathematical and logical operations are also allowed to provide maximum flexibility in handling the logic of games.

This platform has been used to implement a wide variety of business related games including variations of a retailer game with simulated consumers and complex business rules, a double sided call market and negotiation in a procurement scenario. These games are constructed to accurately simulate HP business environments. Carefully calibrated experiments, with human subjects whose incentives were controlled by monetary compensations, were conducted to test how different business strategies result in different market behavior. For example, the retailer game was used to test how the market reacts to changes of HP's contract terms such as return policies. Experiment results were used in major HP consumer businesses to make policy decisions.

## 1 INTRODUCTION

Laboratory experiments have become an increasing important tool in economics to study behavior, test policies and provide information to design better mechanisms. On one hand, seminal work such as those of Vernon Smith and Charles Plott has provided the link between economics theory and actual economics behavior in areas such as markets and auction processes. On the other hand, researchers have successfully test alternative policies in areas such as emission trading, natural-gas pipelines, transportation and allocation of precious NASA deep space resources. Hewlett-Packard Company (HP) has long recognized the potential of this methodology as a business decision-making tool. HP Labs, the research arm of HP, has started an experimental economics program since 1994. Its strategy is to develop experimental models that closely mirror specific businesses. Human subjects then participate in experiments based on these models and the results are used to isolate and evaluate the effects of policies.

The key component of this methodology is a software platform, named MUMS (multi-user multi stage) that allows researcher to implement and modify experimental models in a timely fashion with relatively ease. Each of these experimental models should be able to accommodate both human and robot participants.

There are two requirements guiding the design of this platform. First, it needs to support many types of games. This need stems from the fact that there are many different economics models and business processes that are of interests to HP Labs. It is more cost effective to invest upfront in a more sophisticated system which can be used to support a wider range of models. As shown in later parts of this paper, this strategy has paid off handsomely.

Secondly, the programming interface needs be simple. Researchers with little programming experience should be able to use it effectively.

The ease of programming is the determining factor governing how efficient researchers can implement their models and execute their experiments. Ease of programming is sometimes in conflict with flexibility. A very flexible system such as the C++ programming language would be a terrible choice for someone without the right computing experience. The challenge is to maintain the balance between ease of use and the flexibility of the system. We have decided on the approach of a *script-language* based system. This language will allow a user to define a game as a collection of high-level concepts: players, their inputs and outputs and sequential logic that govern the rules of the game. Basic computing functions such as elements of interface design, networking and database functions are taken out of the hands of the users.

The system is also designed to gather data efficiently. Multiple games with multiple groups of subjects can be executed simultaneously. This allows experimenters to conduct multiple sessions of same or different games at the same time.

Although the system was initially designed for human subjects, we have recognized the importance of robot players (software agents) both as a research area and as a decision support tool. Support for software agents was built into the system from a very early stage of development. This support is integrated into the script language. Behavior of robot players can be easily defined in this language as a plug-in to any game.

The MUMS system was used in several business applications and research projects. The primary area of business application area is channel management. HP conducts multi- billion dollars worth of consumer business through retail channels. There are many different types of retailers such as national retailers, regional retailers, mass merchants, discount clubs, mail order companies and so on. Each type of retailers has its own business objectives, rules and constraints. Policies are used to govern HP's relationship with its retailers. Examples are return policies that govern whether HP will take back products that are not sold, price protection policies that shield retailers from price fluctuations, and minimum advertised price policies (MAP) that limit what prices retailers can put on their advertisements. To design effective policies, HP must understand the implications of these policies on retailer behavior. We have created an experimental model based on this environment. A market with simulated consumers, modelled on the printer market, was created. Human subjects[1] were brought in to play the roles of retailers. At the end of each experiment, each subject was paid in US dollars according to how well they did in the experiment measured by a pre-determined business metrics (such as profit or return on investment). These laboratory experiments allow us to measure the differences in behavior as a function of HP policy changes. These results provided HP business with invaluable information to select between alternative policies as well as potential areas for modification.

Other experimental research areas, which uses the MUMS system, includes but not limited to information aggregation, reputation formation and dynamics, procurement risks analysis.

The paper is organized as follows. Section 2 contains a discussion of previous work. Section 3 is a description of the MUMS scripting language. We describe the software architecture and implementation in section 4. Section 5 describes one of the many applications and research projects, which uses the MUMS system. Section 6 concludes with some discussions about future research.

## 2 PREVIOUS WORK

The idea that to make a computer play games has always been fascinating to scientists. The effort of creating a computer game-playing program can be dated back to beginning of the computer age (Shannon, C. 1950). There are many research efforts to create strong computer playing programs on many classical games, with primary focus on chess. These research efforts have become a major area of artificial intelligence research. At present, the best computer program can play better than human world champions in some classical games, such as checkers (Schaeffer, J. 1997), chess (Campbell, M. et al, 2002), and Othello (Buro, M. 2002). For other games such as Go, computer program is still far behind compared to human experts (Muller, M. 2002).

In recent years, the research on computer games has been extended from classical games to some newer forms of games, in particular, commercial games. With increasing computer

[1] All of our subjects were recruited from the student body of Stanford University.

power, players of these games are no longer satisfied with only fancy graphics, but also demand strong, human-like behavior from the computer opponent. (Laird, J. et al. 2000)

In economics, laboratory experiments have become an increasing important research tool. In the early days before the arrival of mass computing, simple games such as the prisoners' dilemma and simple auctions were implemented with pen and paper. With the advent of personal computers and networking technologies, more sophisticated games such as smart markets, combinatorial auctions, and information markets are possible. Gradually, experimental economists' focus on computer technologies has gone beyond the issue of implementation of economics games. Scientists in both fields have started to exploit the synergies in economics and computer AI to ask the question of whether computers can be good economics agents either in place of or as support tools of human beings.

At the focal point of these fields, which are different in nature but similar in goal, is an obvious need of a generalized platform to support games and agents. The MUMS system is an attempt to create such a platform in which economists can experiment with human behavior and computer scientists can design artificial behavior.

While the key innovation of the MUMS system is its script language, the idea of script languages for particular games is not new. In chess for example, there are a few languages have been developed to simplify the knowledge acquiring process and to help creating better AI. (George, M. et al. 1990, Donninger, C. 1996). Script languages can also be found in commercial computer games, such as Age of Empire (to allow customization of computer behavior), and Neverwinter Night (to allow both customization of computer behavior and easy content addition).

These special script languages work very well in the game they were designed to run within. However, in experimental economics environments where many different types of games are needed, these specialized designs are no longer adequate. A much more general script language is needed. While the MUMS system is unique in its ability to support complicated games, there are several related experiment economics software platforms that may be of interests to the reader.

MUDA (Multiple Unit Double Auction) is a double auction based market system developed at Caltech. It allows subjects to trade in a network environment. This system is highly customizable in terms of the organizations of the markets. Nevertheless, it is designed solely for market-based games. Gencam (General Call Auction Mechanism) is another system developed at Caltech to implement "smarter" markets that allows for contingency bidding.

z-Tree (Zurich Toolbox for Readymade Economic Experiments ) is a more generic package that supports implementation of many types of experiments. Developed in University of Zurich and runs on Windows 95/98/NT platform. The main advantage offered by z-Tree is that it provides build-in support for some common experimental economics models. This enables the experimenter to program and customize experiments through the definitions of a sequence of tables. Its main limitations are its table-based framework and the lack of support for complicated game playing interfaces.

The very early stage of MUMS development was a collaboration research effort between HP Labs and Caltech. Some of the early core concepts were implemented into a DOS-based prototype. Later, the development has moved inside HP Labs. The script language was completely redesigned and a new software implementation was developed.

## 3 HP MUMS SCRIPT LANGUAGE

### 3.1 MUMS Language Overview

There are two main objectives when we designed the MUMS language. First, it needs to be flexible and general enough to express many different types of games. Furthermore, the language should not be limited to expressing small games (toy problem) commonly found in the research community. The ability to handle sufficient complexity is of paramount importance since we want to model sophisticated business scenarios found in HP's businesses. Secondly, the language needs to be easy enough for someone to become proficient without years of training. Thus, HP Labs economists should be able to implement experimental models without the need to be aware of any computing details such as the underlying distributed hardware topologies, low-level communications, and database management.

Although the MUMS language was designed to model realistic business and economics games, its power is not limited to economics. It is relatively easy to implement other multi-player games. As an example, one can design and implement an Internet enabled chess tournament using the MUMS language within a day. This kind of flexibility and efficiency should be of interest to the computer science community.

### 3.1.1 Building Blocks

MUMS is a general purpose language, and have the common features found in other languages, such as data types, multi dimension arrays, variables, functions, control statements, and so on. It has its root in the C programming language. The syntax is similar although the lower level functions such as pointers are completely eliminated.

The primary value of this language comes from support for expressing games. The language hides all the details of the environment, such as a distributed network where the game is played or the database where the data is logged. For example, a key element in the script is the "player" definitions. All input and output functions are called with reference to a player instead of a client machine. The language treats "players" universally in any definition of a game. However, during actual execution, a player can have several representations including a remote client, a local client or even a software agent.

"Stage" is another high-level concept. It is a block of executable script statements. This offers a way to divide the game into natural, smaller; self contain blocks that can be reused. Furthermore, it can also be used as a unit for timing. For example, one stage can be defined as synchronous and thus used to synchronize inputs from all players. Another can be defined as asynchronous and players can submit inputs any time. Very sophisticated timing can be created in a single game by mixing and matching different timing criteria for different stages.

The language provides build-in mechanism to capture inputs, send data and perform screen updates for any player in the game. Database support is also build-in. An experimenter can easily save and retrieve data from the database.

A typical game definition in the MUMS language will usually have following structures:

- definition of all players
- global variables
- stages with executable code
- within a stage, there may be definitions of data sent to players, screen updates, and inputs from players.
- any of these can be repeat many times.
- database statements that can be used to log user activities

Some of the features will be discussed in following sections.

### 3.1.2 Grid Object

The most common business interface is the Excel spreadsheet. Thus, the MUMS user interface is modelled after Excel.

Grid object is the high level abstract for the user's screen. There is one grid object for every user who participates in the game. The grid object supports multiple sheets of 2d arrays of cells. Each cell can be indexed by row, and column, and can contain a number, text, reference to another cell, or a formula. The full range of excel functions is supported. Essentially, the grid control is a book of multiple Excel spreadsheets.

There are many high level operations available to manipulate the grid object. One can assign a 2d string array to a grid in one statement. One can also retrieve data from a block of cells. We can also control whether a subject is allowed to modify any individual cells.

The grid object also enables limited client side programming such as client side input processing and validation (such as transforming an input or checking whether an input is valid or not) and client controlled data processing (for example: decision support tools for an experimental subject).

The full range of Excel formatting abilities is also supported.

### 3.1.3 Microsoft Excel Programming Interface

Grid object has enabled us to have a very rich user interface that is familiar to most human subjects such as business managers or students at local universities. However, creating a good interface design still requires a substantial programming effort. Even though the MUMS language offers excellent support, creating a complex grid object by the sole use of the script language is still a daunting task.

The MUMS system has a build-in interface to the Microsoft Excel spreadsheet application, which is the most widely used spreadsheet application in the world. The system can load an Excel file and create a duplicate grid object. All the cell content and format information are preserved. Once the spreadsheet is loaded, it can interact with rest of the system. Thus, the difficult task of interface design has become a much simpler job of designing Excel spreadsheets.

### 3.1.4 Robots

It is desirable to have automatic players[2] to be in the game. The MUMS language has automatic build-in support for robots. When a game is defined, no reference to robots is necessary. Robots are implemented via script plug-ins to the game. During execution, the experimenter can assign any robots, if available, as well as human controlled clients, to the role of any player. Without doing any additional work, a researcher can easily switch between an all-human experiment, a partially human experiment, and an all robot experiment.

Robots are excellent tools for debugging models during the design stage. They can also provide reference points of performances to benchmark human subjects.

### 3.1.5 External Program Interface

The MUMS language has been implemented as an interpretive language. Therefore, sometimes it is too slow when expensive computation, such as sophisticated statistical model, is needed. An external program interface was implemented which enables us to use other available high performance applications for this purpose. This also provides a way to plug-in robot players.

[2] In the rest of the paper, the use of the terms "automatic players", "software agents" and "robots" are interchangeable.

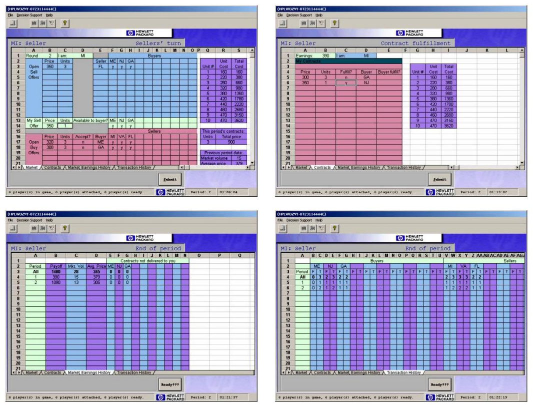
## 3.2 An Example

An example is the best illustration of the MUMS language. The following script implements a Cournot duopoly game with two firms and one homogenous product. Each firm simultaneously chooses a quantity to produce. The price is then determined as a function of the sum of the quantities both firms have chosen. This game obviously is very simple but will serve to illustrate several key features of the MUMS language.

```
// define two players
Player firm1, firm2;

// variables
Integer nplayer = 2;   // there are two players
Integer quantity[2], profit[2], cost[2], price;

// define input formats, these are used to get inputs from spreadsheets
Input GetQuantity(q) {
    q = grid(B2);      // read from the cell B2
}

Stage setup {          // executeable code is divided into stages
    cost[0] = 10;      // setting up parameters, these can  be done in a par file
    cost[1] = 20;
}

Stage input {
    for(i=0; i < nplayer; i = i+1) {     // loop through the players
        gridOn Player[i];              // turn on spreadsheet features
        gridFile("sample.xls") Player[i];// read in a spreadsheet

        GetQuantity:1(quantity[i]);      // read data from spreadsheet into var "quantity"
    }
    wait(2);                           // wait until everyone has entered their input
}
```

## 3.3 Sample of Screens

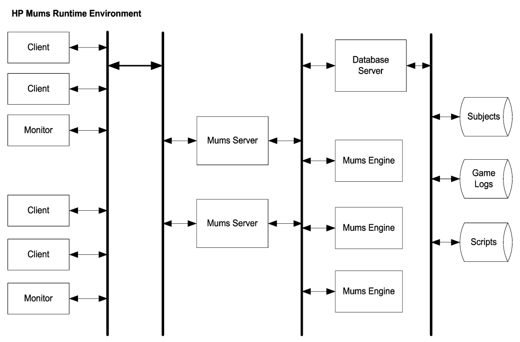
The MUMS system has been used for more than ten experimental projects. All the experiment is substantially more complex than the example described in the sample script. To give the reader some flavor of what is possible, some actual screens of a typical HP Lab experimental economics project are depicted below.



Each individual in this experiment will have access to four spreadsheets shown above.

## 3.4 System Overview

The MUMS system is fully distributed system that one used to conduct the experiment that was defined as scripts in MUMS language. There are five major components here. The engines are the cores of the system, it interpret the MUMS scripts and control the flow of the experiment. The servers are the command/communication center that coordinates all activities. The Clients are the programs that human subjects used to interface with the system. The monitors are for the experimenter to start, stop and monitor the progress of the experiment. The top-level architecture can be seen in figure below.



## 3.5 Implementation

The current implementation is a distributed system based on Microsoft technologies. Although not a requirement, the system is most efficient when located inside a controlled environment such as a dedicated computer lab. Many users can login as subjects and many games can be played simultaneously. A development version is also available in which all components can be running at same machine.

## 4 APPLICATIONS

## 4.1 Overview of Modelling Approach under MUMS

Several business applications and experimental economics research projects were implemented with the MUMS system. Standard experimental economics methodologies were employed in both cases. Human subjects were brought into the HP experimental economics lab and assigned roles appropriate to the model. We gave them accurate information about the game, and told them how their actual monetary compensation was determined by their performance over the course of the experiments. Experimental anonymity was preserved with respect to roles and payment. No deception was used.

However, the model design philosophy was quite different between some business applications and research projects. The versatility of MUMS is most evident when the system was successful in accommodating these very different modelling choices.

For example, in a research project that studies information aggregation mechanisms (Chen, Fine and Huberman 2001, 2002), the MUMS system was used to create a standard one-sided call market with multiple assets. This alone is nothing out of the ordinary since there is plenty of capable call market software created for experimental purposes. The same project also called for the implementation of a specially designed matching game that queries subjects about probability beliefs and calculate their payoffs based on a matching criteria.[3] In this game, each subject was given some probabilistic information about a random event that has ten possible outcomes. Each of them was then asked to report discrete probability distributions over the ten possible outcomes. Their payoffs were calculated based on the true outcome, what they have reported, and how well their reports were matching reports of other participants. The purpose of this experiment is to test whether special designed matching game could aggregate probabilistic information better than an

[3] Please see Chen, Fine and Huberman 2002 for a full description of the project. The purpose of this game is to solicit probability beliefs that contain private and public information.

information market. The MUMS system was also used to implement this very unique game.

The real power of MUMS, however, is best illustrated in complicated business applications. Since 1997, experimental methods have been used to create test-beds to test policy design in several HP businesses. Because it is expensive to test in the real market, and more importantly, it is infeasible to isolate and control a real market, the laboratory is an attractive alternative to study the impact of policy changes.

The design philosophy of these business test-beds sometimes runs counter to that of academic experiments. Instead of a preference of the simplest design that can encompass the modelling issues at hand, we decided to include as many features as possible to preserve the complexity of the field environment. This led to the obvious disadvantage of limited ability to isolate cause and effect. However, we would be able to identify problems in the policies in the most realistic circumstances. In fact, some of the exploits of the policies would not have been discovered if we did not take this approach. Furthermore, just the identification of these problems even without a firm theoretical understanding of why they happened was extremely useful to the business.

This approach utilized the full advantage of the flexibility of the MUMS platform. Furthermore, the ease-of-programming allowed us to implement the models and modify them in a very short time frame. Some of the modifications are done within hours. This capability is invaluable to maintain a project timeline that is consistent with the short decision cycle of businesses.

## 4.2 Minimum Advertised Price (MAP) Policy Evaluation

In a series of experiments (Charness and Chen 2002), behavior of retailers were studied with respect to the common industry practice of setting a minimum advertised price (MAP), which is a lower bound on the price a retailer can advertise for a particular product. Since thousands of products are involved, MAPs are usually not enforced by legal contracts. Instead, retailers face penalties such as discontinuance of shipments of certain products or withdrawal of market-development funds[4]. The focus of this application is to find the best punishment strategy amongst several alternatives.

MAPs are necessary because they reduce price competition between retailers. And if retailers perceive that price competition for HP products is too high, they may stock and promote other manufacturers' products resulting in a loss of market share for HP. Yet it is not clear which form of MAP is best and which enforcement policies are effective.

A market of seven heterogeneous retailers was modelled. Each participant played the role of a retailer. Consumer demand was computer simulated. These seven retailers interacted repeatedly in competing for consumer demand for products differentiated by price and manufacturers. Retailers made decisions about stocking, advertising and pricing. In each period, each retailer chose a price and competed for some percentage of the potential market. Most retailers could increase this percentage for individual product by advertising beyond a minimum exposure percentage. However, each retailer could have a different maximum exposure percentage. Advertising yielded diminishing marginal returns. To simplify the model, advertised prices were assumed to be the same as the selling prices. Thus retailers were required to enter one price per product in each period. If a retailer advertised and set a price below the MAP, a penalty would be invoked. Experiments with different punishment schemes were conducted.

Consumer demand was simulated using a random utility-based multi-level logit model (Dubin 1998; McFadden 1976). This model treats each product as a collection of attributes (such as price, brand, product features and so on). Each consumer assigns a numerical weight to each of the attributes and calculates a score based on a linear weighing function. The probability that the consumer purchases a product increases with this score. The number of consumers in the market was a known stochastic process to the retailers. They also received a signal each period and provide further information about the distribution of the number of consumers. The model also incorporated

[4] Market development funds are money retailers receive from manufacturers that are supposed to use for advertising. However, since the use of these funds are not strictly monitored nor enforced, it has little difference than cash rebates.

product obsolescence. Some products were phased out with retailers receiving notices five periods in advance.

Most retailers also had to make inventory decisions. The timing of deliveries to the retailers was a stochastic process known to the retailers. There is also a return policy (limit to 6% of a retailer's total shipment) to buffer against overstocking.

Another aspect of the heterogeneity of the retailers was their objective functions that reflect realistic business goals of the different categories of retailers. These measures include various combinations of gross profit, net income, revenue and a return on investment measure.

As one can see, this model is an attempt to emulate a very complex and even chaotic environment where no game theoretic analysis can fully address. Furthermore, during the course of the project, based on customer inputs as well as experimental results, many model assumptions were modified and additional features added. For example, we further restricted the number of products a retailer could advertise. A reputation effect based on history of prices was introduced. Many of these changes involved changes of the interfaces as well as the business logic. All changes were implemented through the script language in a very rapid fashion.

The first set of experiments was conducted in September 1999. Stanford students were used as subjects. Two possible ways of exploiting the structure of the policies were identified. The key insight is that retailers could exploit the timing of the penalties particular near the end of the life cycle of a product and avoid the penalties in full or in parts. Experimental evidence supports this by showing a clear increase in MAP violations towards the end of lives of products (Charness and Chen 2002).

A new design was created based on the insights we have discovered. A second set of experiments was conducted in February 2000. The new design was confirmed to have addressed the problem. We no longer observed an increase in MAP violations towards the end of product life cycles.

The new design was adopted by HP business in Oct 1999 even before the validating experiments (Feb 2002) can be completed. It is still the standard MAP penalty strategy today.

## 5 CONCLUSION

HP Labs has developed a multiplayer gaming platform primary for economics experiments. This platform supports a flexible scripting language that allows the researchers to program in the concepts of games. The core scripting language was implemented on decentralized, module-based software architecture. This system is scalable in three primary dimensions: complexity of the games, ease-of-programming and the size and number of games running at the same time.

Business applications were created taking advantage of the system's ability to handle complex games. Many of these applications also called for rapid modification of the games. The easy to program script language made it possible. Several of these experiment-based business applications have resulted in major policy decisions for HP's multi-billion dollar consumer business.

Currently, this system is used for several experimental economics project in areas including information aggregation, reputation mechanism designs, and procurement strategies.

Looking towards the future, there are two main research areas that HP Labs is pursuing in connection with the MUMS system. The first one is the natural extension of MUMS into a more expressive, easier to use and more powerful system. Enhancements includes:

- Automation of the whole experimental process including recruitment and management of subjects, access control into the system, scheduling and payment of subjects.
- Upgrades to the language
- Re-engineer the software with a JAVA implementation to facilitate cross-platform functionalities

So far, most of the experimental work was done with real human subjects. Robot players were mainly used as a debugging tool. The obvious research question is that can we design a computer player capable of participating in these games, just like other human? If so, how? Recently, we have completed one project in the procurement area in this direction. HP Labs Bristol has developed a software agent that will negotiate in the place of a human buyer in a procurement environment. We have developed a "procurement" game, in collaboration with HP procurement organization, to test the performance of this software agent against human opponents. Experimental results were helpful in identifying areas that this software agent needs to improve. There are internal HP Labs interests to use this game to test other software agents designed with different AI methodologies.

As the methodology of human economics experimentation has become more mature, new opportunities open up in the intersection between experimental economics and computer AI research. First, economics offer a wide range of interesting games that may not previously catch the attention of computer scientists. Secondly, analysis of experimental data of these games offer insights of how humans behave. This may turn into ideas for AI designs. Thirdly, as the software platform matures, the overhead of conducting joint research will diminish. For example, with the MUMS system, once a game is implemented, it will be costless to switch between human experimentation and AI experimentation.

## 6 REFERENCES

Buro, M. (2002). Improving heuristic mini-max search by supervised learning. Artificial Intelligence 134 (2002) 85-99.

Campbell, M. Hoane, A. J. and Hsu F.-h. (2002). Deep Blue. Artificial Intelligence 134 (2002) 57-83.

Gary Charness and Kay-Yut Chen, "Minimum Advertised Price Policy Rules and Retailer Behavior: An Experiment", to appear in Interfaces, special issue on Experimental Economics.

Kay-Yut Chen, Lesie Fine and Bernardo Huberman, "Eliminating Public Knowledge Biases in Small Group Predictions", working paper 2002

Kay-Yut Chen, Leslie Fine and Bernardo Huberman, "Forecasting Uncertain Events with Small Groups", In Proc. of the ACM Conference on E-commerce, Oct 2001.

Donninger, C. (1996). CHE: A graphical language for expressing chess knowledge. ICCA Journal, Vol.19, pp. 234-241.

Dubin, J. (1998), Studies in Consumer Demand – Econometric Methods Applied to Market Data, Kluwer Academic, Boston, MA, USA.

Fischbacher, U. (2000). Zurich Toolbox for Readymade Economic Experiments. URL http://www.iew.unizh.ch/ztree/index.php last time checked, 2002/06/05.

George, M. and Schaeffer, J. (1990). Chunking for Experience. ICCA Journal, Vol 13, pp 123-132.

Laird, J. Lent, M.v. (2000). Human-level AI's killer application: Interactive computer games. Proc. AAAI-2000, Austin, TX, 2000, pp. 1171-1178.

McFadden, D. (1976), "Quantal Choice Analysis: A Survey," Annals of Economic and Social Measurement, Vol. **5**, pp.363-390.

Muller, M. (2002). Computer Go. Artificial Intelligence 134 (2002) 145-179.

Schaeffer, J. (1997). One Jump Ahead: Challenging Human Supremacy in Checkers, Springer, Berlin, 1997.

Shannon, C. (1950). Programming a computer for playing chess, Philosophical Magazine 41 (1950) 256-275).