# A Decentralized Agent-Based Platform for Automated Trade and its Simulation

Erich Kutschinski[α], Daniel Polani[β] and Thomas Uthmann[α]

{eri,polani,uthmann@informatik.uni-mainz.de}

[α]Institut für Informatik
Johannes Gutenberg-Universität, 55099 Mainz, Germany
[β]Institute for Neuro- und Bioinformatics
Medical University Lübeck, 23569 Lübeck, Germany

**Abstract**

This paper discusses the design, development and application of the distributed multi-agent platform DMarks II. The platform consists of a flexible software library for the creation of applications based on multi-agent systems. As an example the DMarks II platform is applied to the distributed computation of a multi-agent simulation in a market scenario. The market model is presented in some detail. In the simulations we examine four different seller pricing strategies, two of which use a modified Q-learning algorithm to extract maximum profits. Special interest is put on asynchronous simultaneous multi-agent learning of pricing strategies in different competitive environments.

## 1 Introduction

For the last few years the Internet has profoundly affected the retail trade of standardized consumer goods, from books and CDs to intercontinental flights. For a single product consumers enjoy a wide choice of offers and can easily compare prices making use of the decreased effort of information retrieval and of software agent technology. But the interaction between the consumer (or his agent) and retailer is typically very limited: it mainly consists of obtaining price statements and eventually sending orders. For the future we envision a much more sophisticated trade on the Internet benefitting both, consumers and retailers: agents entering into actual negotiations with each other would be able to act on consumers' or retailers' behalf, locating specific products or variants, discussing terms of delivery or special conditions, and performing the transactions automatically, based on their owners' preferences. An extended model of this is presented in MarketSpace (Eriksson et al. [3]).

We believe this automated retail trade will be based on a decentralized dynamic system of autonomous software agents with asynchronous mechanisms for communication and transactions. In the present work, our first aim is to propose a design for such an agent system, discuss its

specific structural demands and develop a technical concept for its implementation. Based on the distributed market simulation system DMarks (Polani and Uthmann [10]) we develop an integrative project DMarks II consisting of:

1. A prototypical implementation of this general automated trading system, using Java technology;
2. An extension of this prototype to a distributed multi-agent simulator that allows the study of market scenarios.

Using the DMarks II simulator we then examine developments which could arise from widespread application of software like the DMarks II platform in real markets. Namely, we want to shed light on price developments and the power of seller strategies of different complexity in a mixed buyer population, with an increasing fraction of fully informed buyers. Here we make use of results achieved by Greenwald and Kephart [4, 5]. Different types of asynchronous Multi-agent Reinforcement Learning will be used to determine optimal seller strategies.

## 2 Design concepts of the DMarks II prototype

Due to the distribution of consumers and retailers over the internet our aim is to structure the agent platform as a highly decentralized multi-agent system with point-to-point interaction. Agents can enter and leave the system at any given time, so the composition of the system may constantly change. The major difficulty is how to provide the individual agents with knowledge of other active agents in a consistent and efficient way, especially on a large scale. As a centralized register is bound to be overloaded or flooded with obsolete information when the agent population grows too large, we propose a decentralized network of *forum agents* as the structural backbone of the agent system.

DMarks II agents are started by the users in their own runtime environment and join the agent space by entering a forum. The agents may be of different types derived from a common ancestor to perform different tasks. Apart from interaction within the agent framework, they can access external resources, such as databases or the World Wide Web to retrieve information (see figure 1).

The forum agents are decentralized components of the agent system. They bring together local groups of agents and thus subdivide the agent space. An agent that is member to a forum is visible to all other agents via that forum. In the decentralized and changing environment the forums provide up-to-date information of locally present agents, and their specific derived types or 'professions'. Forums can be linked together, such that an agent will be able to get to know agents outside its local forum group. The agent may also enter these other forums to make itself known to the other agents. The aim of the forum concept is to divide the agent space into groups of interest that can effectively be searched by an agent. When the DMarks II prototype is applied to a specific task, good consideration has to be taken as how to set up the forum network for efficient agent interaction.

Knowing the partner is a precondition for interaction between agents. Agents get to know about each other by actively searching the forum network, from a message received from an other agent, or by direct interaction with the agent's user, who may provide name and host of a forum

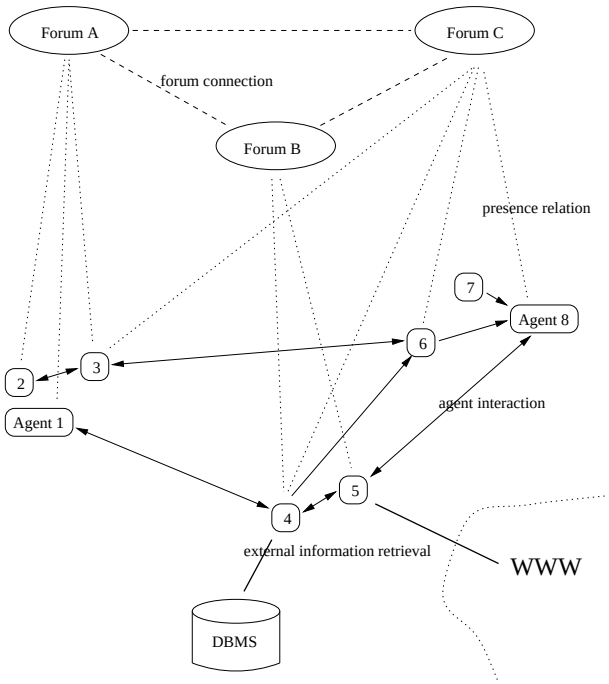


Figure 1: Structure of the DMarks II agent platform

or even of a single agent. Agents interact with each other through communication or transactions. Communication is restricted to point-to-point messages containing sender, addressee and an open type of content. The content types have to be provided when applying the DMarks II prototype to a specific purpose, and the agents need knowledge how to handle the different types. Transactions go beyond mere information exchange and require confirmation of the subject by both agents. They take the form of predefined two-sided contracts. As with the message content, these contracts and especially their effect on the agents' states need to be defined when applying the agent system. For a transaction, both partners need to sign their part of the contract. After the second agent has signed, the contracts are matched and confirmed, which results in a change of the agents' states. Through this mechanism no trusting in trading partners is required, but the DMarks II platform will ensure the correctness of the two-sided exchange. For example, a mutual exchange of information can be modelled as a transaction: Both agents promise to provide some information to the other through an exchange by signing their contracts. Only after both did successfully sign (i. e. not lie about information they do not have) will the information be delivered to the respective partner.

Here we need to emphasize that this decentralized transaction mechanism is modelled towards consistency and safety of transactions on the agents' level and does not explicitly address security issues on the Internet. It assumes a trustworthy design of the contract implementations by the application designers. The centralized transaction matching mechanism of the previous phase of study of a distributed market simulator, by which the current system was motivated, is described in [10].

## 3 Technical details of the prototype and the simulator

We chose Java as implementation language for the DMarks II prototype for various reasons: As Java is platform-independent the agent system can be run on heterogenous computer networks such as the Internet. Its powerful multi-threading features lend themselves to the design of multi-agent systems, in which agents runs independently of one another. Furthermore Java's remote method invocation (RMI) mechanism lets us create decentralized distributed applications in a fast and straightforward way, and is at the core of the current implementation of the DMarks II prototype: every agent is in itself a remote object with defined interfaces to its environment registered on the particular host machine it is running on.

The agent design follows a modular architecture with a communication manager, a transaction manager and the agent's control component which makes use of the two interaction managers to receive information and perform interactions. Additional modules, for example an extended reasoning or learning component, or an asynchroneous task processor, can easily be added to an agent to be used by the control component. Contact between agents is made only through their interaction managers. They provide means to deliver messages to an agent, and to sign and confirm contracts for transactions.

The forums are based on the agent architecture as well. Their control component allows other agents to enter or leave the forum, and to retrieve a list of the forums' agents. The forum agents' control component can be extended to provide additional functionality for any application of the agent system. Java RMI was found to run sufficiently fast when used in local networks. Even on wide area networks response time was fast enough to allow interactive usage of the software agents. Distributed simulation runs on wide area networks do not seem feasible.

The DMarks II prototype follows a rather generalized concept and can well be applied as a platform for automated trading. To show the potential of the approach and as a testbed we extend the prototypical agent platform to a multi-agent simulation system for market scenarios, the DMarks II simulator.

On the technical level, we need to introduce timesteps into the agent model to allow a periodical update of the agents' states. As the DMarks II prototype is based on asynchronous interaction, the simulator will operate asynchronously as well. We cannot use a simulation loop, because agents are running in a decentralized fashion. Instead we add an additional clock thread to each agent, to trigger all agents in the simulation in the same time intervals independently from each other. This means the simulation step interval needs to be estimated from known computational demands of the agents and their distribution on different machines *before* the simulation run, and cannot be adapted to the system load during runtime. This poses limits on simulation speed, but is the price to pay for a decentralized system without (by design) extensive synchronization mechanisms. Still, in our timing concept all agents have a guaranteed time slot, as long as the step interval is selected with care.

## 4 The DMarks II simulator's market model

Apart from the introduction of the discretized time concept, the modelling extensions for the DMarks II simulator are typical for any application of the DMarks II prototype to a given scenario. The necessary market model specifications have been motivated by the previous DMarks system

[10], Vriend [13] and Varian [11].

In our market scenario, units of a single good are exchanged against units of currency by two different agent types, *producers* and *consumers*. Producers generate units of this good and keep it in a store, until it can be sold to consumers. Production and storing incur costs, so to optimize their profit producers will have to find an adequate production capacity. Consumers have a certain demand for the good, which is modelled as an amount of money they are willing to pay for the good, their *bid level*. They will receive a fixed sum of money every simulation step to satisfy their demands. After buying a product, it will be consumed, and for satisfied demand a consumer reward will be paid on the consumer's reward account. The lower the price paid the higher the reward will be, so to maximize rewards the consumers will try to bring down the producers' prices.

In detail, producers keep an account $A$ which is initialized once with a small credit. They also keep track of a store $S$ for their goods, for which they are charged $c_s$ for every unit in store, for every timestep. Production incurs costs $c_p$ per unit produced and is performed by setting the production capacity $C$ to a certain level. At every simulation step, producers will first be charged $c_s \cdot S$ for the store, then $c_p \cdot C$ for the production, and then $S$ will be increased by $C$. If $A$ drops below 0 after any step, the producer is financially ruined and forced to leave the simulation.

Consumers also keep an account $A$ which is increased every step by a fixed income. After buying goods, they directly consume them, converting them into a consumer reward $r(q_0, q, p)$. It depends on the quantity $q_0$ already consumed during the current step before this consumption, the quantity just consumed $q$, and the price $p$ paid for the $q$ units. The rewards are accumulated on a consumer reward account $R$, and are determined according to the consumer's bid level $b(q_0 + q)$ of the quantity $q$ of the good (the amount per unit the consumer would be willing to spend for the next infinitesimal small quantity on top of $q_0 + q$). The reward is calculated as the difference between what the consumer would have been willing to pay for $q$ units on top of $q_0$, and the actual amount he paid: $r(q_0, q, p) = \int_{q_0}^{q_0+q} b(s)ds - pq$. Consumers therefore maximize their consumer reward by completely satisfying their demand $q$ for a given price $p$, where $b(q) = p$, after looking for the lowest $p$, taking into account their available funds $A$.

Message contents for this scenario have to include price enquiries, price statements, and orders of a certain quantity $q$ for a certain price $p$.

In this market scenario we need two different contracts for product exchange, a buy-contract for the consumer and a sell-contract for the producer side. Both contracts contain quantity $q$ and exchange rate $p$ of the transaction. Signing a buy-contract reduces the consumer's account by $p \cdot q$, its confirmation pays out the consumer reward $r(q_0, q, p)$. A sell-contract reduces a producer's store by $q$ upon signing, and increases its account by $p \cdot q$ upon confirmation.

This extends the DMarks II prototype to the DMarks II simulator, a distributed multi-agent simulation system based on a market scenario with interacting producer and consumer agents. The specifications of this model are by no means final; the design of the simulator allows easy extension by additional model features.

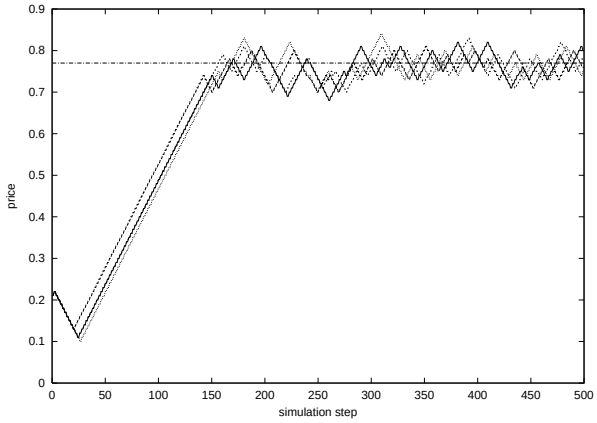


Figure 2: Price development of 4 sellers and 20 buyers at $\omega = 0.3$ and $T = 2C$ (with market clearing price in this setting at $p^* = 0.8$, horiz ontal line indicates average market price between simsteps 150 and 500)

# 5 Simulation results for different seller strategies

## 5.1 Introduction

We simulate a market scenario in which a fraction $\omega \in [0, 1]$ of the consumer population is fully informed about all sellers' prices, and the remainder knows the price of only one randomly selected seller every timestep. All consumers aim at optimizing their consumer rewards, and therefore select the seller with the lowest known price. After ordering, all known information of the buyers is invalidated and new price quotes need to be obtained from the sellers, as prices are bound to change frequently. Obviously, the fully-informed buyers will create a considerable amount of competition in the market between the sellers. The buyers' strategy in the following experiments is fairly simple and fixed to provide a stable environment for the examination of different seller strategies. The described short-term greedy strategy of the buyers is a major reduction of the complexity of the market scenario, and we plan to examine more complex behavioral strategies for the buyers in the future. The introduction of these extended strategies into the buyer model will be very straightforward, due to the modular architecture of the agent platform.

For the time being we want to focus on the seller side of the market: The seller strategies in the different experiments vary in knowledge, complexity and computational demand. We are especially interested in developments of price and in the quality of the different fixed and learned pricing strategies as the fraction of the fully informed buyers increases.

## 5.2 A market with constant supply

The first set of experiments was run with a fixed production capacity $C$ for each seller. This is meant to mirror a market with a highly inflexible production where sellers control their profits by mainly setting their prices, for example the oil producing industry. The general assumption for sellers is that a lower price will lead to increased sales. The sellers optimize their long term profits by keeping their store $S$ around a fixed target store level $T$, raising the price by a fixed amount when $S < T$, and lowering it when $S > T$. We found that with increasing price transparency $\omega$ in the consumer population competition between the sellers increased, leading to the sellers following

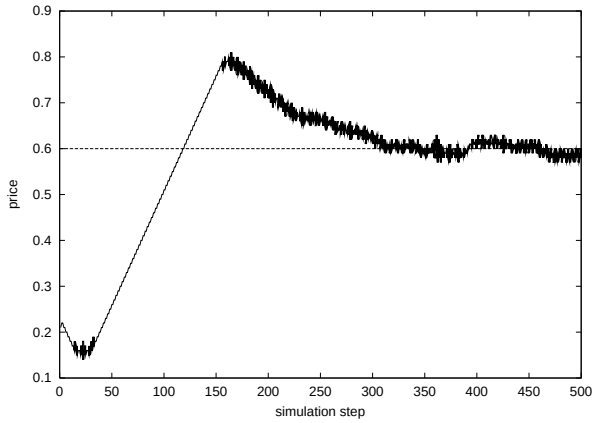


Figure 3: Price development of 4 sellers and 20 buyers at $\omega = 1$ and $T = 2C$ (with market clearing price in this setting at $p^* = 0.8$, horizontal line indicates average market price between simsteps 150 and 500)

each others' prices closely. In the extreme the sellers alternately take turns charging the lowest price of all competitors with increasingly regular phase shifts (see figure 2).

The general price level rose to and fluctuated around a value dependent on total supply, total demand, and $T$. While competition was not too strong ($\omega < 0.3$), the theoretical market clearing price was quickly and independently reached by the seller agents. With growing competition the price attained by the market was significantly lower than the equilibrium price. As a reason for this we isolated the concentration of market demand on the single cheapest seller, and its inability to serve this high demand due to its production limitations. This leads to a reduction of average total output as compared to the total consumer demand for any given price. Thus each seller will tend to set its price considerably *below* the equilibrium price to sell its whole production quantity (see figure 3).

The experiments imply that with a fixed production capacity, a store as a buffer for production, and a price adaption mechanism based on store levels does lead to collusive and coordinated price-setting behaviour of sellers, through competition induced by price-comparing buyers. A producer's level of storage seems to reflect its sale numbers well, even on longer terms. The greater the number of sellers though (proportionally increasing the number of buyers to keep the relation between total supply and total demand), the higher $T$ has to be set for the adaptation mechanism to work well and prevent the market price from crashing. The increasing of $T$ leads to higher amplitudes in the store fluctuations, and thus to higher total store costs, reducing total profit in a scenario with strong competition.

### 5.3 A market with highly adaptable supply

In the second set of experiments the production capacity was not fixed, but varied according to consumer demand (a "produce-on-demand" scenario). This was meant to reflect a product with a highly flexible production chain. Sellers optimize their profits by setting their prices only, with no need to take their store $S$ into account.

### Fixed pricing strategies

In this setting we used two seller strategies following Greenwald et al. [5], the Derivate Follower (DF) and Myopically Optimal (MO) pricing strategies. The DF-sellers use only the immediate reward from the last two price settings to adapt their future price to maximize expected profits. They keep increasing their price, as long as their achieved profit increases and switch the direction of the price change as soon as the achieved profit decreases. Doing so, they greedily try to climb the profit curve when selecting a new price. DF-sellers can reach a collusive price setting near the monopolistic price extracting maximum profit from the buyer population, as long as no other conflicting pricing strategy is introduced into the market. The MO-sellers have perfect knowledge of consumer demand and their competitors' price settings and use this information to select the price yielding the short-term maximum profit. In competitive scenarios with discrete fixed price levels this typically leads to undercutting behaviour and cyclical price wars ranging from the monopolistic price down to the production cost, as described extensively in Greenwald et al. [5]. The MO-pricing strategy is, due to its perfect knowledge and short-term rationality, by far superior when tested against the DF-strategy. The results achieved in our simulations were comparable to those of the preceding studies. It has to be emphasized that the observed price wars between two or more MO-Sellers are a direct result of the discrete price levels applied. In a model with continuous price selection no price wars will emerge, but the prices will drop to the Bertrand equilibrium price (for a description of the Bertrand equilibrium see Varian [12]).

### A learning strategy with little information

To bridge the gap between the reactive DF and the myopically rational MO-sellers, we here introduce a Price-Profit (PP) adaptation mechanism based on single-state Q-learning (for Q-learning see Watkins et al. [14, 15]). The PP-pricing strategy continually uses achieved profits to adapt profit expectations to a given price, and chooses offer prices through a Boltzmann stochastic selection mechanism based on these expectations. In particular, PP does not use information about the composition of the buyer population or the competitors' price settings (as does MO), but uses only achieved profits to adjust its prices (like DF), applying a robust machine-learning mechanism. Our expectation was that market demand and competitors' pricing strategies should, in the long run, implicitly be reflected in the general relation between prices and expected profits in the market. This would allow PP to adapt to any market situation without the need to know about its competitors' price strategies. PP did show behavior patterns seen before in DF and MO (price undercutting and collusive price setting) and, as expected, generally outperformed DF in the long run. Due to its slow adaptability to changing environments, it could not stand against the perfectly informed MO in any competitive setting.

An interesting question also was how two PPs would adapt their learned pricing strategies to each other, and whether the individual strategies would converge over time. To examine this we placed several PP-sellers in the market, each of them deciding to change its price with a fixed probability every simulation step. Thus an asynchronous multi-agent reinforcement learning framework was created. The analysis of these runs showed, that the learning process has to take place considerably slower (i. e. with a lower learning rate) than in the runs against fixed pricing strategies to assure convergence of the learning process. The simultaneous learning lead to the development of identical pricing strategies of all PP-sellers with a distinct price maximizing the

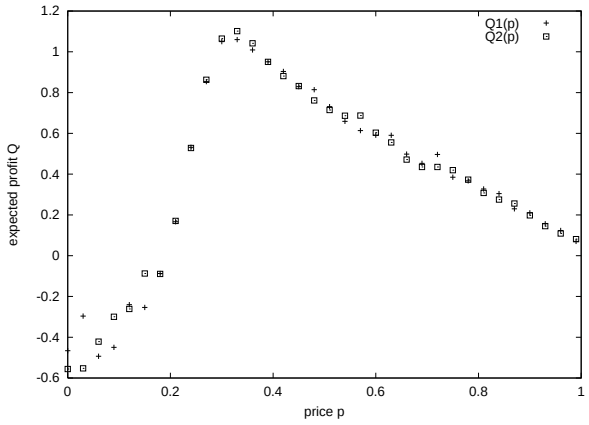


Figure 4: Learned profit functions $Q_1(p)$ and $Q_2(p)$ of 2 PP-sellers with 20 buyers at $\omega = 1$.

expected profit (see figure 4). This optimal price setting always lies between the monopolistic and the Bertrand equilibrium price, depending on the amount of competition in the market as determined by the buyer population.

### A more sophisticated learning strategy

Finally a learning pricing strategy was developed which also takes the price settings of the competitors into account. As above, it is based on Q-learning with a Boltzmann price selection mechanism, but the lowest competitor's price determines the state the Q-seller is in. This model of competitor's prices is sufficient here, because with the given buyer population, only the own and the lowest competitor's price determines the demand. We expected from the Q-sellers to respond more readily to the immediate market situation than the PP-sellers. The same asynchronous multi-agent reinforcement learning framework as with the PP-sellers was used to train and observe the agent's strategies in the competitive settings.

The first simulations runs were performed with no lookahead for the Q-learning mechanism. The Q-seller managed well to learn the profit function and optimal pricing strategy against any of the fixed strategies. Even against the PP-strategy and itself it developed this strategy without any need to slow down the learning process. As was expected when future rewards are discounted by the factor $\gamma = 0$, the evolved strategy was the myoptimal pricing strategy (see figure 5).

Due to the asynchronous training setup, the rate of convergence is considerably slower than in comparable experiments performed by Greenwald et al. [4]. Generally, alternating learning updates with full knowledge of the partner's value function are assumed in multi-agent reinforcement learning frameworks (see Littman [9] and Hu et al. [7]). In this work, asynchronous learning updates which do not make use of the partner's current value function led to similar results, which can be an indication that synchronicity or asynchronicity in the training process does not affect its results. Still, in the asynchronous setup more learning updates will be needed to remove the additional noise created through the stochastic rather than alternate learning updates.

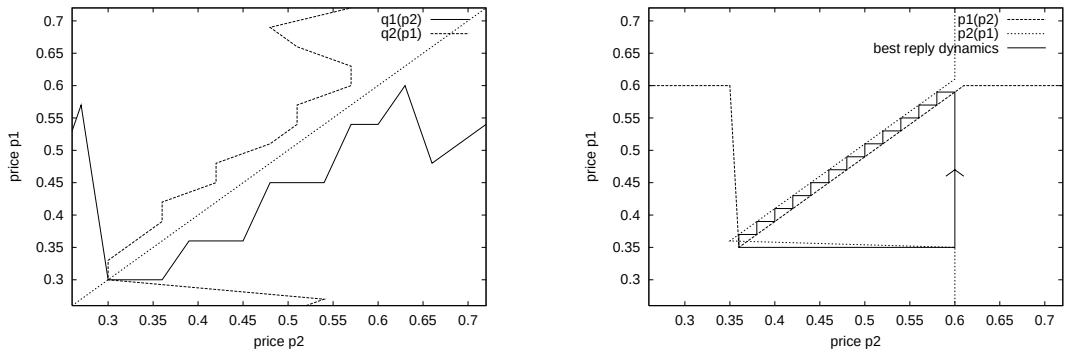


Figure 5: Pricing strategies relating competitor's price $p_2$ to own price $p_1$ (left: learned strategy; right: myoptimal strategy)

# 6 Conclusion and Outlook

On the technical level the developed agent framework performed quite well, even in the distributed computation of the simulation. The DMarks II platform and simulator do lend themselves to the development of asynchronous multi-agent systems and multi-agent simulations. Its implementation in the Java language allows it to be run on a varienty of OS-platforms and over the Internet. It can be extended easily and effectively by additional components or different simulation models due to its flexible architecture[1].

For future research, we plan to extend the behavior models of the agents: Apart from refinements of the behavior strategies of the sellers, the introduction of more complex buyer strategies should lead to interesting market developments. Furthermore, we want to include more than one product into the market model to look into dynamics arising from production chains, which would form a link between the constant and adaptable supply setting experiments.

On a larger scale, the examination of other typical market mechanisms could also be undertaken effectively using the above market model. The agent platform's design allows the emulation of virtually any agent behaviour by simply extending the agent's control component. Thus the effect of discount pricing on large product quantites or personalized bundle offers by the sellers, or group formation by the buyers (demand pooling) could be examined. Cartel formation of the seller agents is also a possible direction for application of the DMarks II simulator. The market model itself (i.e. the producer and consumer classes of the simulator) would need to be extended, possibly by a discounted investment cost, to explore market entering strategies of sellers, or regional effects, such as the impact of communication or transaction costs. Most of these extensions could be introduced in a straightforward way and examined using the distributed DMarks II simulator.

# References

[1] Craig Boutilier, Yaov Shoham, and Michael P. Wellman. Editorial: Economic principles of multi-agent systems. *Artificial Intelligence*, 94(1–2):1–4, 1997.

[1] The DMarks II agent platform is available for free download under the GNU General Public License from the author's web page at `http://nautilus.informatik.uni-mainz.de/~eri`.

[2] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagenten systems. In Sandip Sen, editor, *Collected papers from the AAAI-97 workshop on multiagent learning*. AAAI Press, 1997.

[3] Joakim Eriksson, Niclas Finne, and Sverker Janson. Sics market space: an agent-based market infrastructure. In *First International Workshop on Agent-Mediated Electronic Trading, AMET-98, Selected Papers*, Lecture Notes in Computer Science No. 1571. Springer, Berlin, 1999.

[4] Amy R. Greenwald and Jeffrey O. Kephart. Shopbots and pricebots. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, August 1999.

[5] Amy R. Greenwald, Jeffrey O. Kephart, and Gerald J. Tesauro. Strategic pricebot dynamics. In *Proceedings of the ACM Conference on Electronic Commerce (EC-99)*, Denver, November 1999.

[6] C. E. Hewitt. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47:79–106, 1991.

[7] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the International Conference on Machine Learning (ICML-98)*, 1998.

[8] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

[9] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*, pages 157–163, New Brunswick, 1994.

[10] Daniel Polani and Thomas Uthmann. Dmarks: Eine verteilte umgebung für agentenbasierte simulationen von marktszenarien. In G. Hohmann, editor, *Simulationstechnik, 13. Symposium in Weimar*, volume 3 of *Frontiers in Simulation*, pages 391–394, 1999.

[11] Hal R. Varian. A model of sales. *American Economic Review, Papers and Proceedings*, 70(4):651–659, September 1980.

[12] Hal R. Varian. *Intermediate Microeconomics*. Norton, New York, 4. edition, 1996.

[13] Nicolaas J. Vriend. A model of market-making. Working Paper No. 184, Department of Economics, Universitat Pompeu Fabra, Barcelona, Oktober 1996.

[14] Christopher C. J. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, 1989.

[15] Christopher C. J. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.