

Strategies for autonomous economic agents

Wolfgang Ketter

Dept of Computer Science and Engineering
University of Minnesota

Carlson School of Management - March 3rd, 2004

The MAGNET team: Amruddin Agovic, Alex Babanov, John Collins, Steven Damer,
Maria Gini, Ashutosh Jaiswal and Elena Kryzhnyaya.

Work supported in part by the National Science Foundation under awards

NSF/IIS-0084202 and NSF/EIA-9986042.

Overview

- Agents, Markets, and Auctions.
- Citysim
 - Heterogeneous market strategies, and controllable environment.
- MinneTAC
 - Strategies for managing the product life cycle.
- MAGNET
 - Combine auction winner determination with scheduling constraints.
- Conclusion

Agents, Markets, and Auctions

Our long term goal is to enable programs (“agents”) to do transactions on electronic markets on behalf of a user.

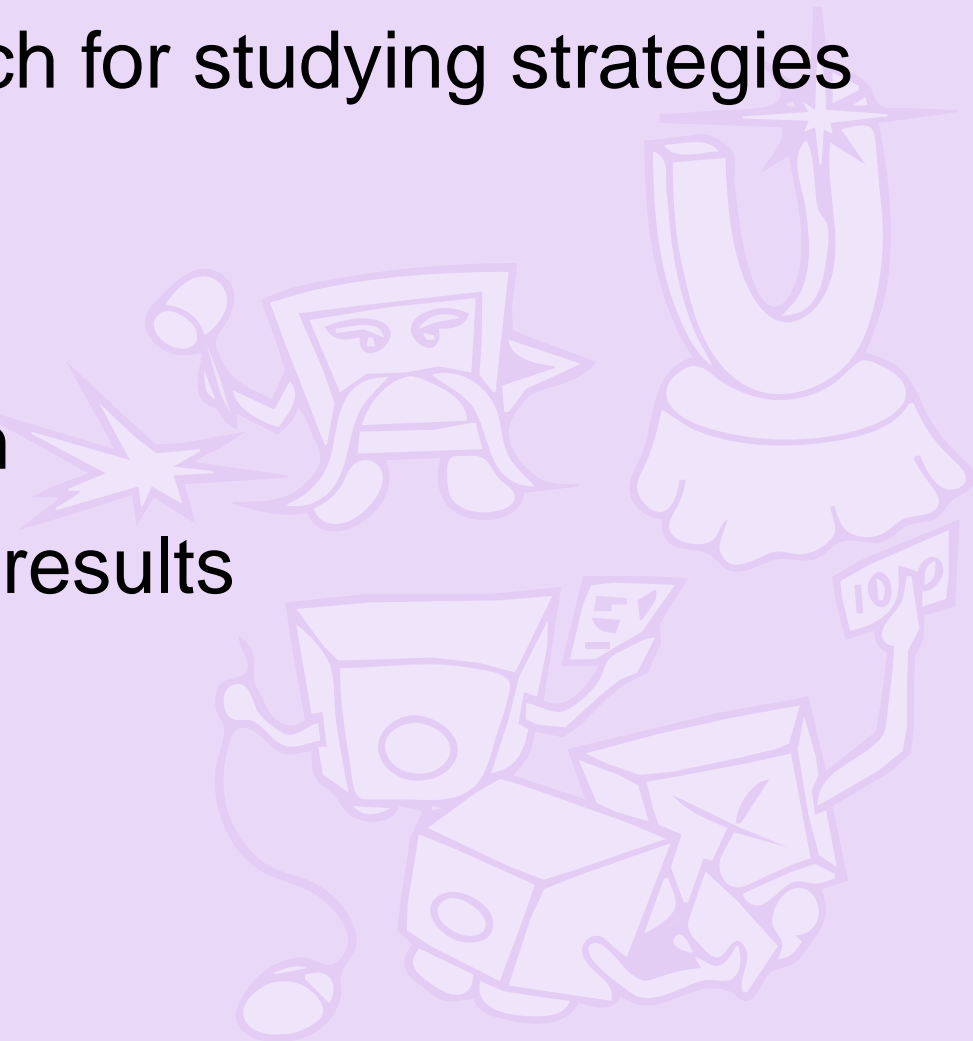
Why electronic markets and auctions?

- Electronic markets have the potential for reduced costs and increased access to world-wide markets.
- Auctions are a general and proven way to negotiate among rational entities.

Citysim

An evolutionary approach for studying strategies in markets.

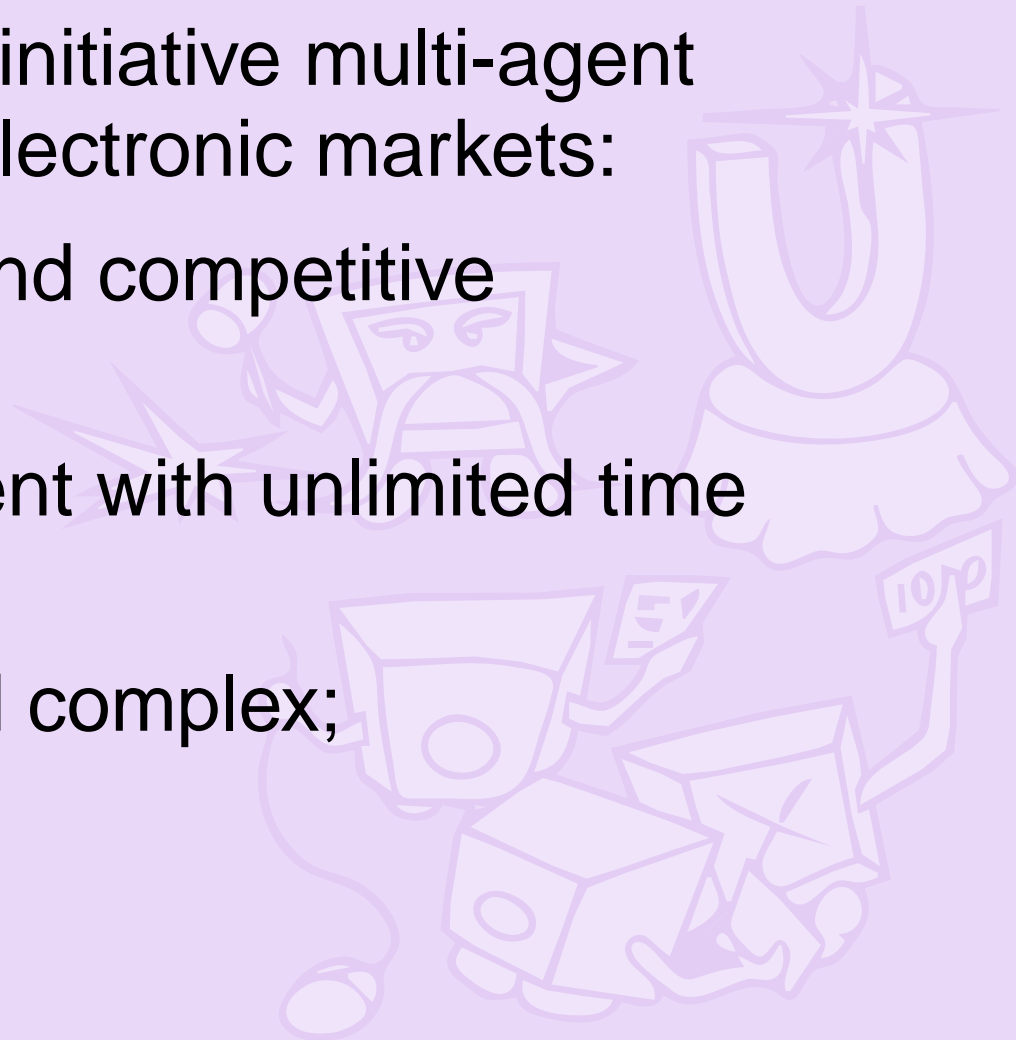
- The problem
- Suggested approach
- Example model and results
- Conclusion



Problem Domain

Automated and mixed-initiative multi-agent systems in emerging electronic markets:

- limited resources and competitive environment;
- dynamic environment with unlimited time frame;
- heterogeneous and complex;
- open environment.



Objective

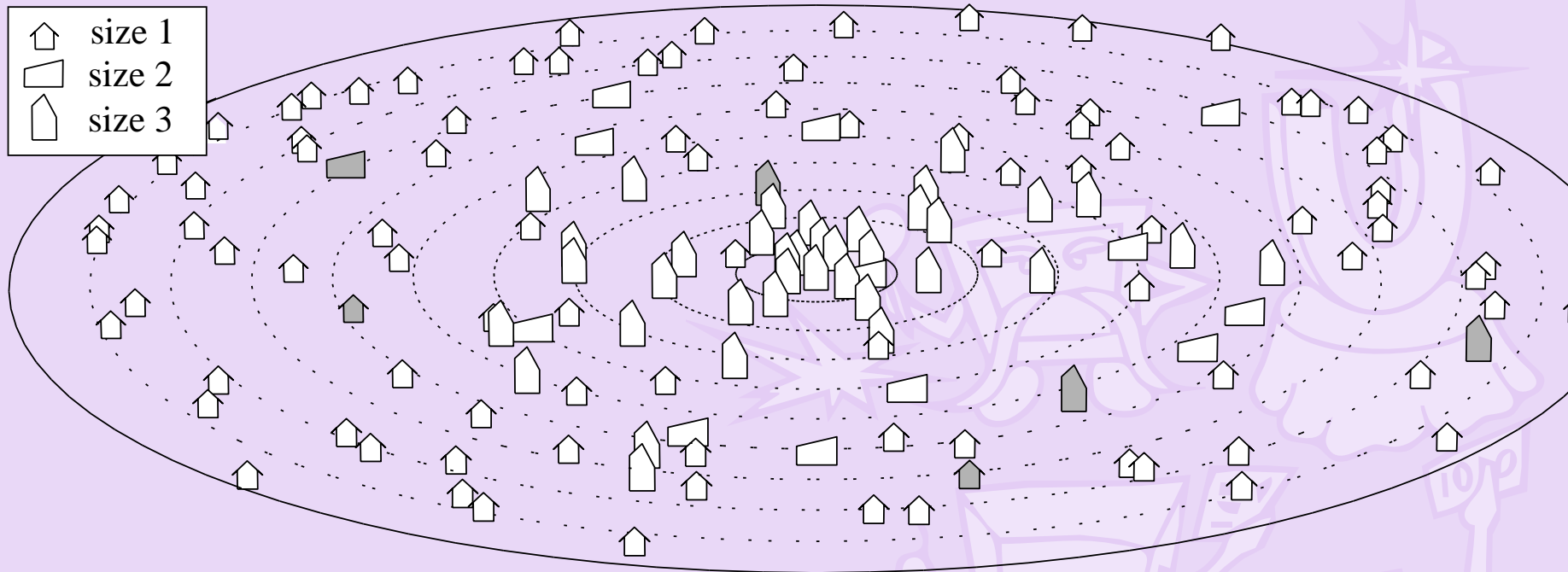
Study performance of agents' strategies in multiagent systems using an evolutionary framework.

- Find out which agent strategies are good for which market conditions.
- Perform comprehensive testing of electronic market implementations.

The Citysim Model

- Simulation of a society of suppliers of a service and their customers.
- The agents live and interact in a circular city.
- The simulation is based on a simple supply and demand model, where multiple service providers compete for customers, and where profitability is the criterion to stay in business.

Structure of the City



The city is divided into 10 zones. Customers are uniformly distributed in polar coordinates, i.e., the most people live near the city center and fewer toward the rim.

Customer Agent (1)

- Anonymous customers come to the market for a single transaction, with a fixed frequency λ^c :

$$t_{i+1}^c = t_i^c - \frac{1}{\lambda^c} \log U[0, 1]$$

where $U[x, y]$ is a random variable distributed uniformly on the interval $[x, y]$.

- The location of a new customer in polar coordinates is determined by the rules below:

$$r \sim U[0, R] \quad \text{and} \quad \alpha \sim U[0, 2\pi)$$

Customer Agent (2)

- Density of customers is inversely proportional to the distance from the city center.
- A customer minimizes its net cost:

$$\text{netcost} = \text{price} + \text{distance} \times c^{\text{mile}} + \text{delay} \times c^{\text{hour}}$$

- Customers do not change their properties during the simulation.

Supplier Agent:

- Suppliers are characterized by their pricing strategy, and the number of customers they can serve concurrently (size).

strategy = $\langle \text{location, price} \rangle$, type = $\langle \text{strategy, size} \rangle$

- A *strategy* decides where to locate a business and how much to charge for the service.
- A *type* is represented in the market by the corresponding *supplier generator*.
- Each supplier is audited at regular time periods and removed from the market if its profit becomes negative.

Reproduction of Strategies (1)

Auditor (Upper Learning Layer):

- Evaluates the performance of supplier agents' strategies based on suppliers' average profit over a specified period of simulation time.
- Agents that have a negative profit are removed from the market.
- The probability of creating a particular type of agent is proportional to the number of agents of its type that have survived in the market.
- There is a small probability (*noise*) that a new supplier is assigned a type at random.

Reproduction of Strategies (2)

Generator [1] (Lower Learning Layer):

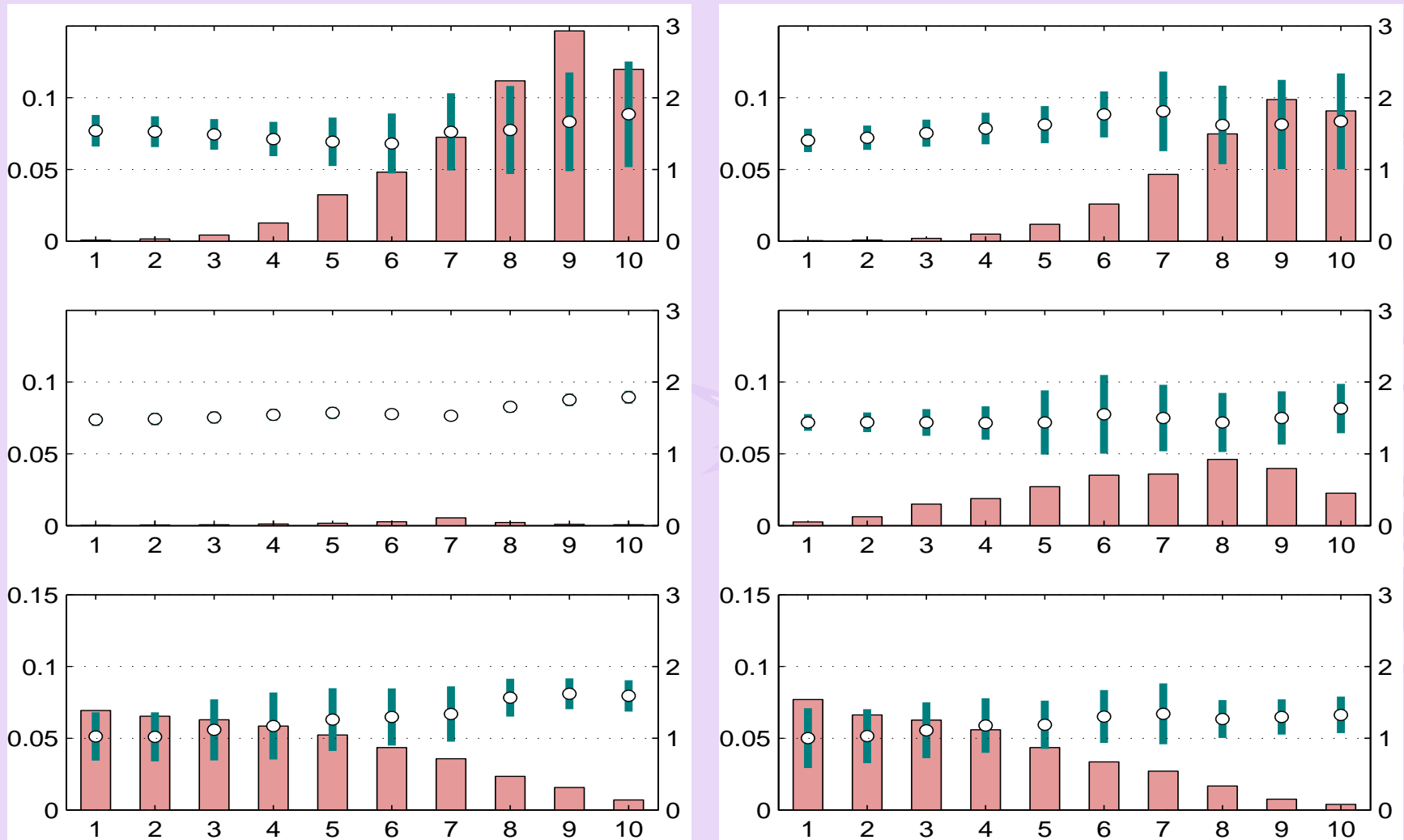
- Maintains a pool of information concerning the history and the current state of its type suppliers.
- Basic initial parameters are based on general demographic information.
- Calls a strategy-specific reproduction rule to select the rest of the parameters based on the initial parameters of all currently present agents of the same class.

Reproduction of Strategies (3)

Generator [2] (Lower Learning Layer):

- The supplier uses the location and price selected by its generator and never alters them.
- Only generators are capable of learning and adapting to the market situation.

Reproduction of Strategies (4)

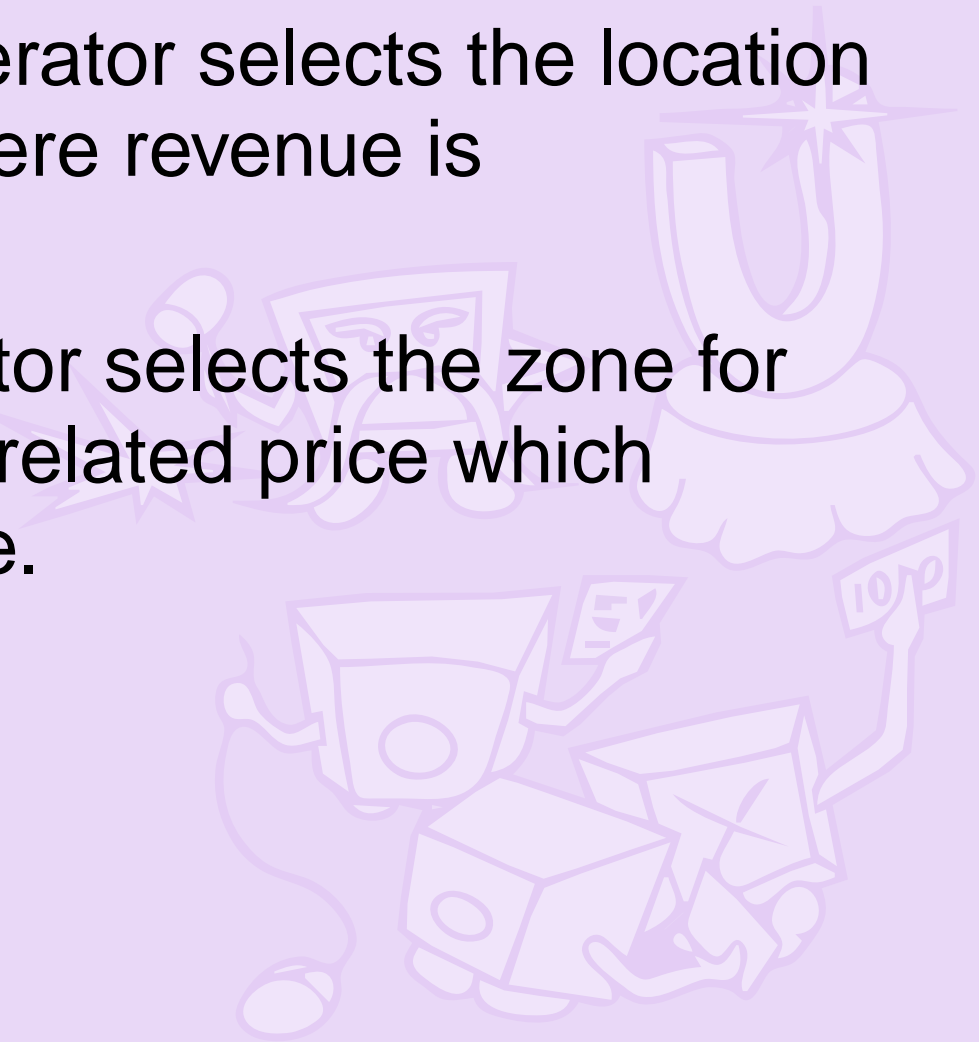


Example of gene pools for price suppliers.

Sample Simulation with Two Different Strategies:

Market Sampler The generator selects the location for each supplier where revenue is maximized.

Price Seeker The generator selects the zone for each supplier and a related price which depends on the zone.

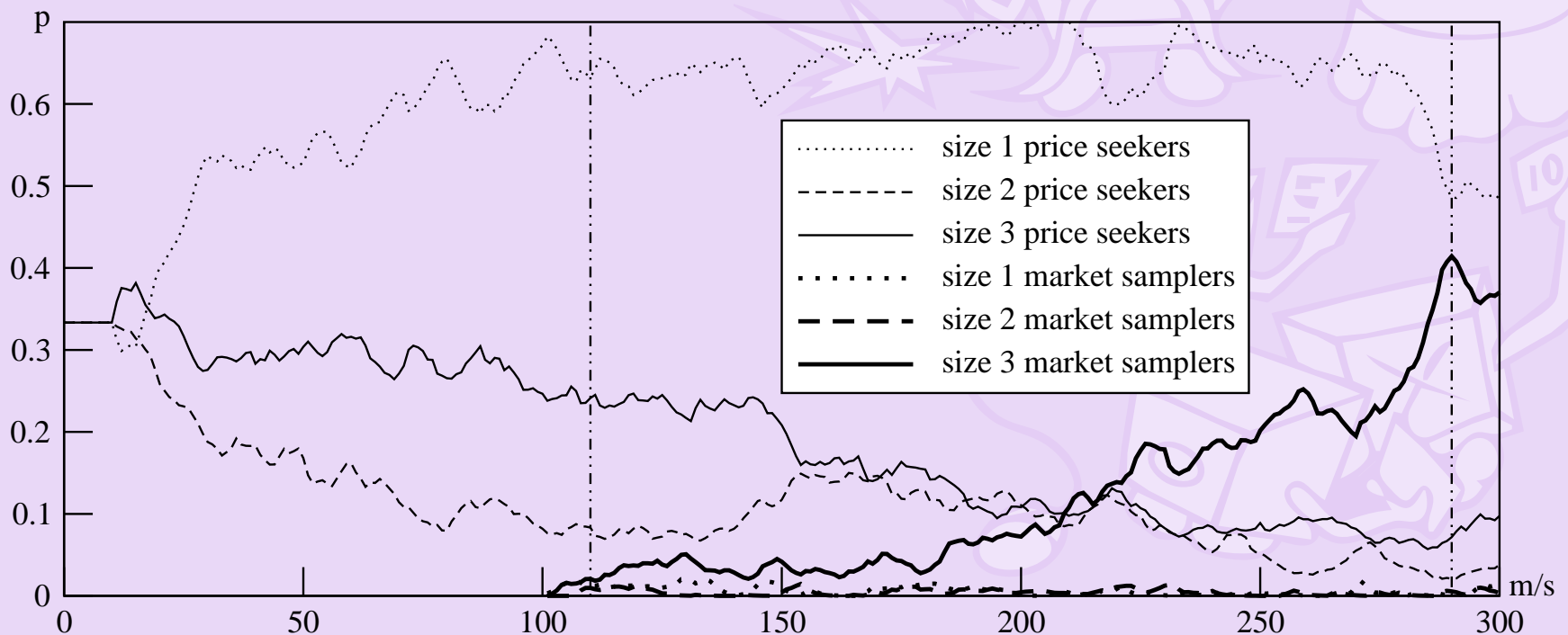


Why these Two Different Strategies?

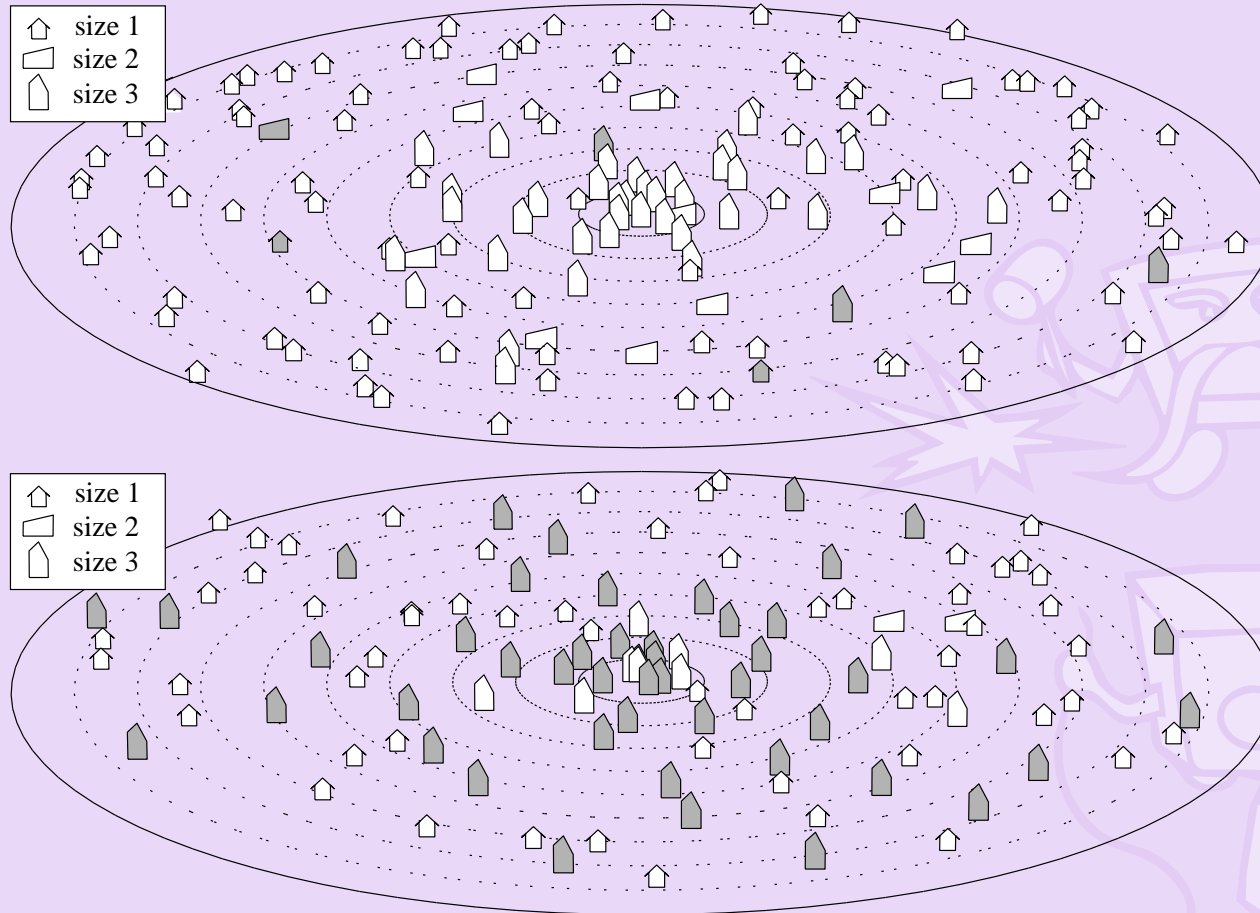
- The strategies exhibit sufficiently different behavior.
- The strategies were designed so that neither strategy has a strict advantage over the other.
- Because of that, the strategies can coexist and evolve in the market at the same time.

Simulation Time-Line:

Probabilities of a new supplier entry for different supplier types as a function of milestone numbers. Market sampler suppliers are introduced at milestone 100.



Results: Structure of the City

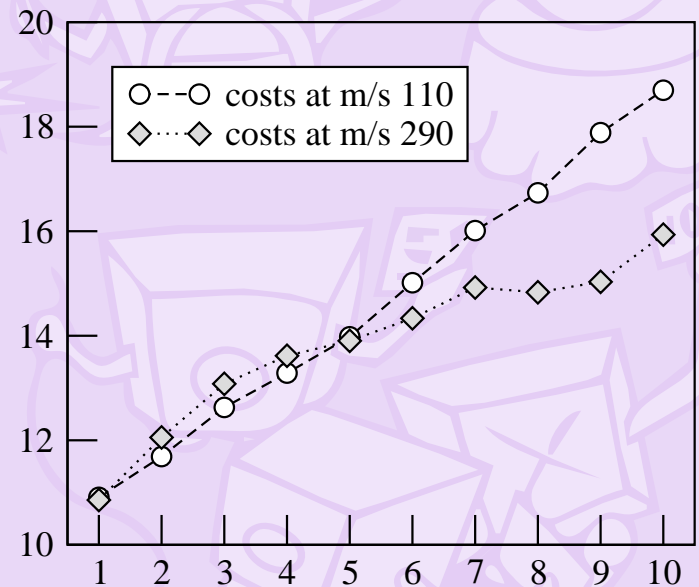
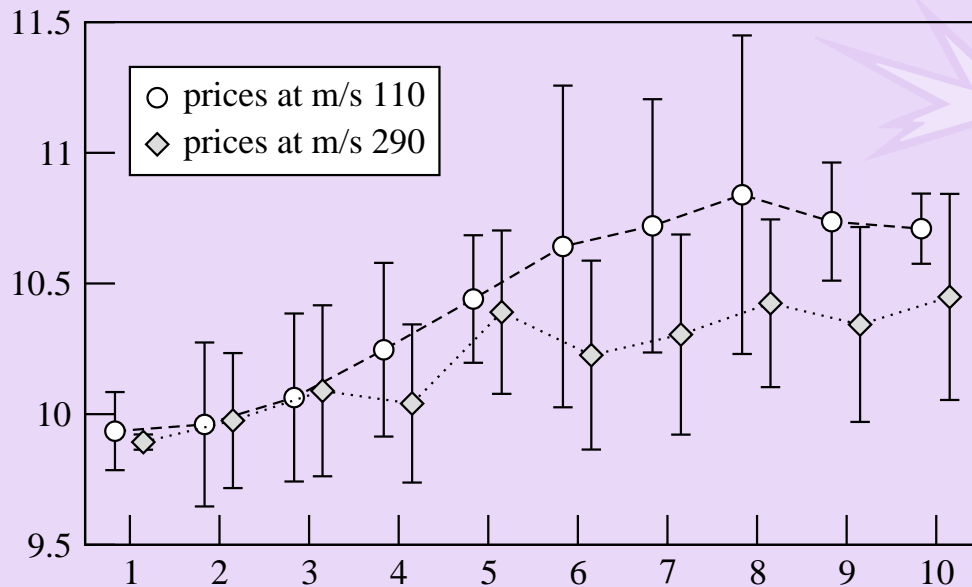


City at
milestone 110
(top) and 290
(bottom).

Price seeker
suppliers are
denoted by
white houses,
market
samplers are
gray.

Price Distribution

Average supplier prices with standard deviations (left) and 25 hour half-life decaying averages of customer costs (right) for 10 concentric city zones at milestones 110 and 290.

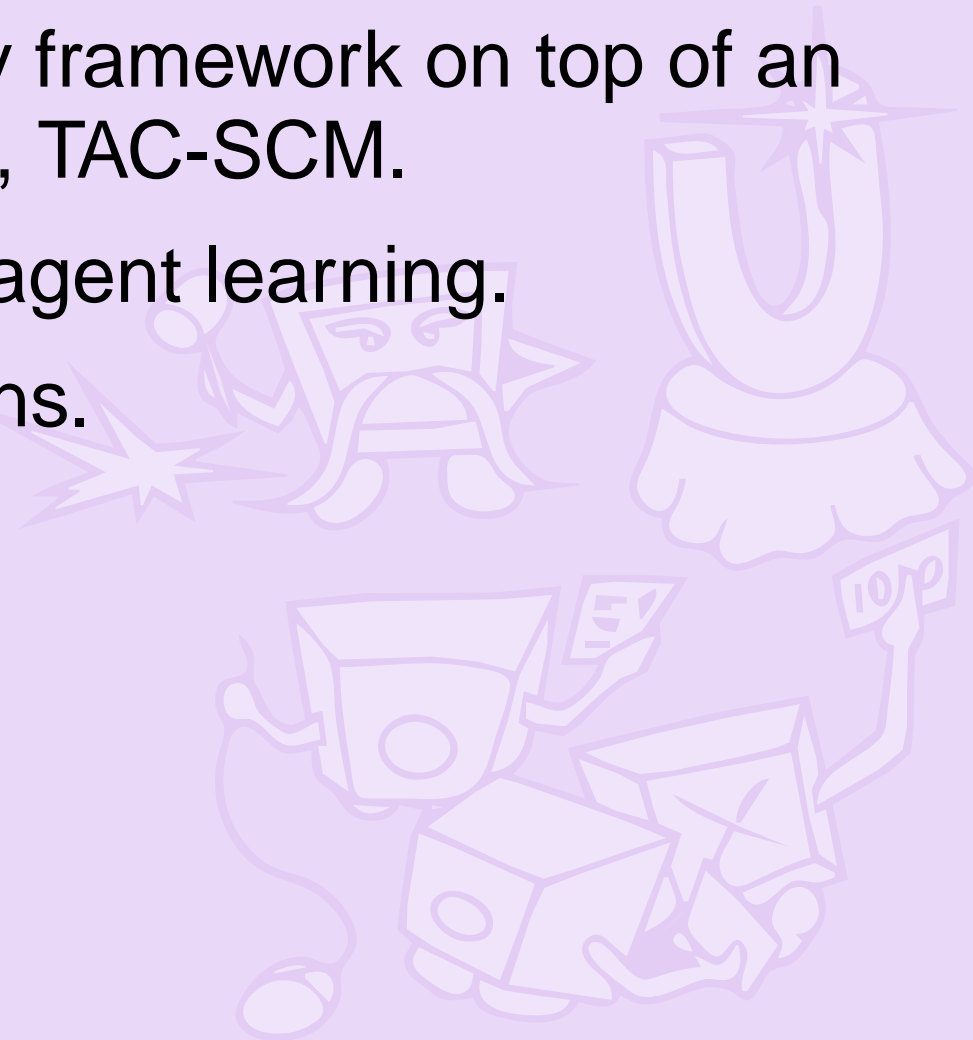


General Characteristics: Evolutionary Approach

- Controllable environment and data collection.
- A large range of problems can be studied.
- Testing over a long-time period is possible.
- The type and number of agents can change frequently.
- Reputation building is a vital part of any real system.
- Fully specified strategies.

Conclusions and Future Work

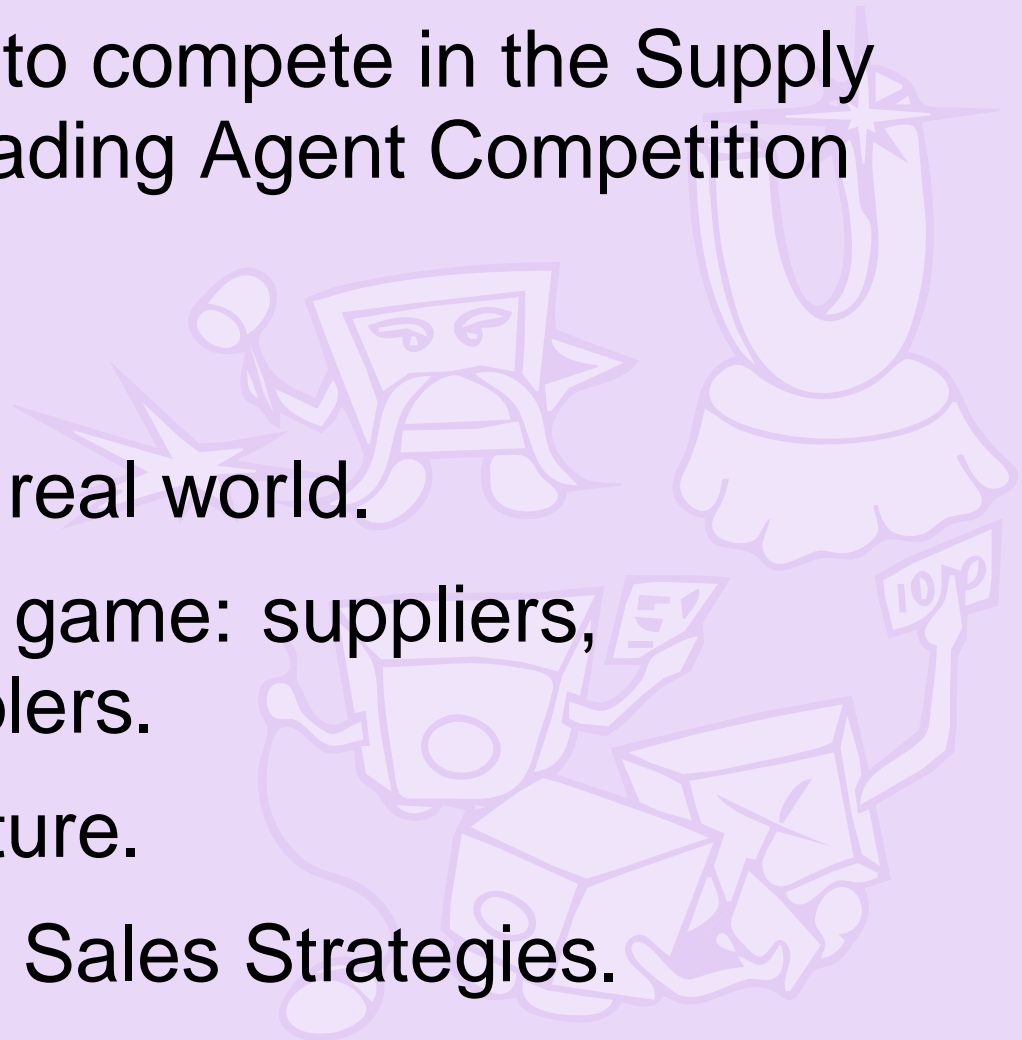
- Build an evolutionary framework on top of an existing system, e.g., TAC-SCM.
- Introduce individual agent learning.
- Theoretical extensions.



MinneTAC

An autonomous agent to compete in the Supply Chain Management Trading Agent Competition (TAC SCM).

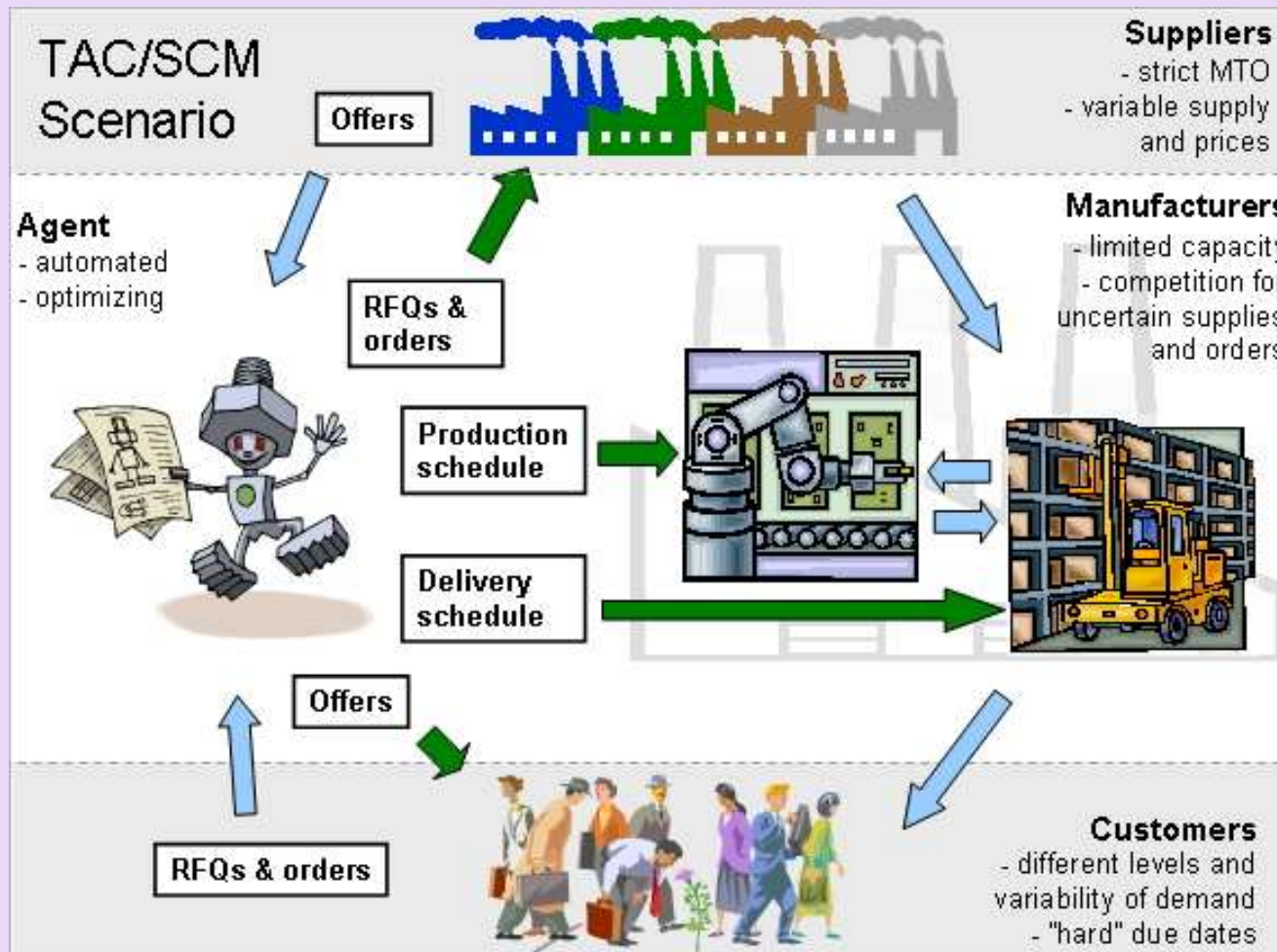
- Game overview.
- Implications for the real world.
- Components of the game: suppliers, customers, assemblers.
- MinneTAC architecture.
- Game analysis and Sales Strategies.
- Conclusions.



TAC SCM - Game Overview

- Six autonomous agents compete to maximize profits in a computer-assembly scenario.
- Agents compete for customer orders and for procurement of various components.
- The simulation takes place over 220 virtual days, each lasting fifteen seconds of real time.
- At the end (game/tournament), the agent with the most money in its bank account is the winner.

TAC SCM - Scenario (1)



Description and rules at www.sics.se/tac.

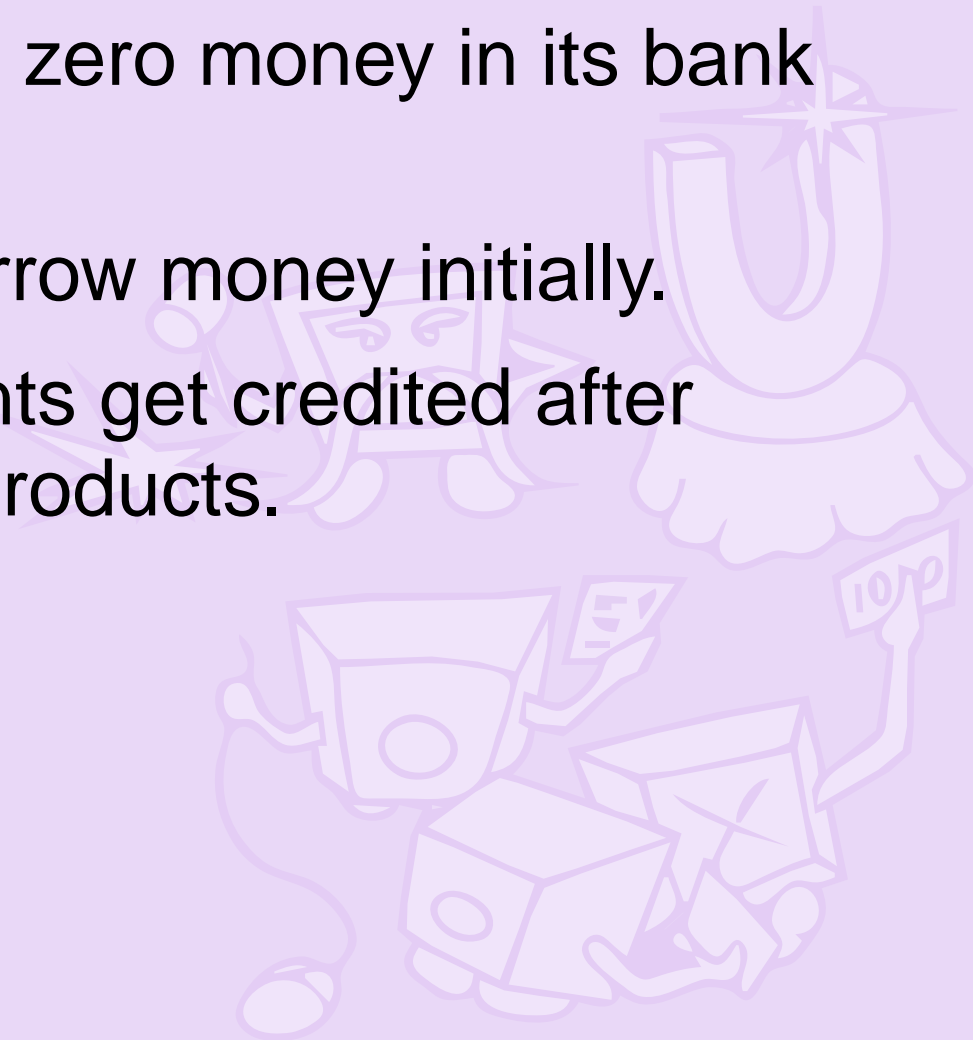
TAC SCM - Simulation of Business Environment

- Suppliers - produce raw materials
- Assembly agents
- Customers - buy assembled computers
- Raw materials market
- Finished goods market
- High degree of uncertainty
- Strategic
- Complex



TAC SCM - Simulation of the product life cycle

- Initially an agent has zero money in its bank account.
- The agent has to borrow money initially.
- Agents' bank accounts get credited after delivery of finished products.



Static vs. Dynamic Practices

- Today's supply chains are mostly static, relying on fixed, long-term trading partners. Not always optimal!
- Dynamic management allows for finding better matches between suppliers and customers as market conditions change.
- This is the goal of the Supply-Chain Management Trading Agent Competition (TAC SCM).

TAC SCM - Implications in the Real Business World

- Markets are changing quickly - Agents that have the ability to meet changing market demands in a timely and cost-effective manner will prosper.
- Decreased cost and performance improvement can be provided through new algorithms for procurement of components, production, and sales management.
- The algorithms can be used to provide information to a human decision maker.

TAC SCM - Components

- PCs are built from 4 component types: CPUs, motherboards, memory, and hard drives.
- CPUs and motherboards have two product families: Pintel and IMD.
- Given at the start of each game:
 - Component catalog: Information about the components (id, base price, supplier, name)
 - Bill of materials: Description of PC types (SKU, components, # assembly cycles)

TAC SCM - Suppliers (1)

- There are 8 suppliers in total, 2 for each component type.
- Each day an agent can send a maximum of 10 Request for Quotes (RFQs) to each supplier.
- This allows an agent to probe the supplier without swamping it with too many messages.

TAC SCM - Suppliers (2)

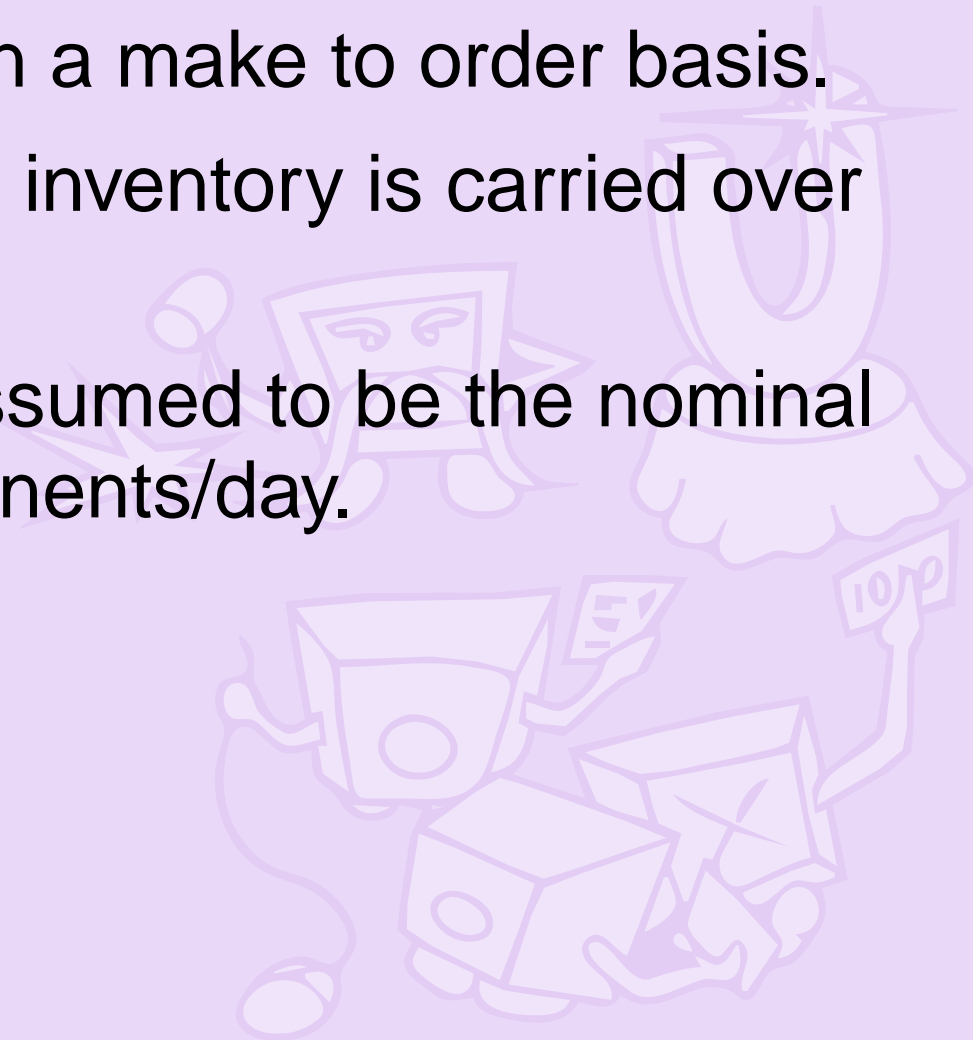
- The RFQs are bundled in an ordered list, going from highest to lowest priority.
- Each supplier collects all the RFQs from the agents and processes them at the end of the day.
- The supplier chooses an RFQ bundle and considers the next unprocessed RFQ in that bundle.

TAC SCM - Suppliers (3)

- RFQ response comes in the form of an offer:
<Offer-id, RFQ-id, Quantity, DueDate, Price>
- If the order cannot be filled in its entirety then two amended offers are sent: a partial offer and an earliest complete offer.
- The agent must then decide which offer, if any, to accept (it cannot accept both).

TAC SCM - Suppliers (4)

- Suppliers produce on a make to order basis.
- Excess capacity and inventory is carried over with no cost.
- Future capacity is assumed to be the nominal capacity, 500 components/day.



TAC SCM - Customers (1)

- Request PCs of different types to be delivered on a certain due date.
- The quantity of each order is chosen uniformly between [1,20].
- Agents must bid to satisfy the entire order (both in quantity and due date) for a bid to be acknowledged.

TAC SCM - Customers (2)

- Customer demand is expressed as $\#RFQ_s$.
- $\#RFQ_s = \text{Poisson}(\overline{\#RFQ} \text{ per day})$.
- $\overline{\#RFQ}$ per day varies using a trend updated with a random walk:
- $\overline{\#RFQ} = \min(320, \max(80, \overline{\#RFQ} \times trend))$
- $trend = \max(T_{min}, \min(T_{max}, trend + \text{random}(-0.01, 0.01)))$
- The start value of $\overline{\#RFQ}$ is chosen uniformly in the interval $[RFQ_{min}, RFQ_{max}]$
- The start value of $trend$ is 1.0 (reset at Opt.)

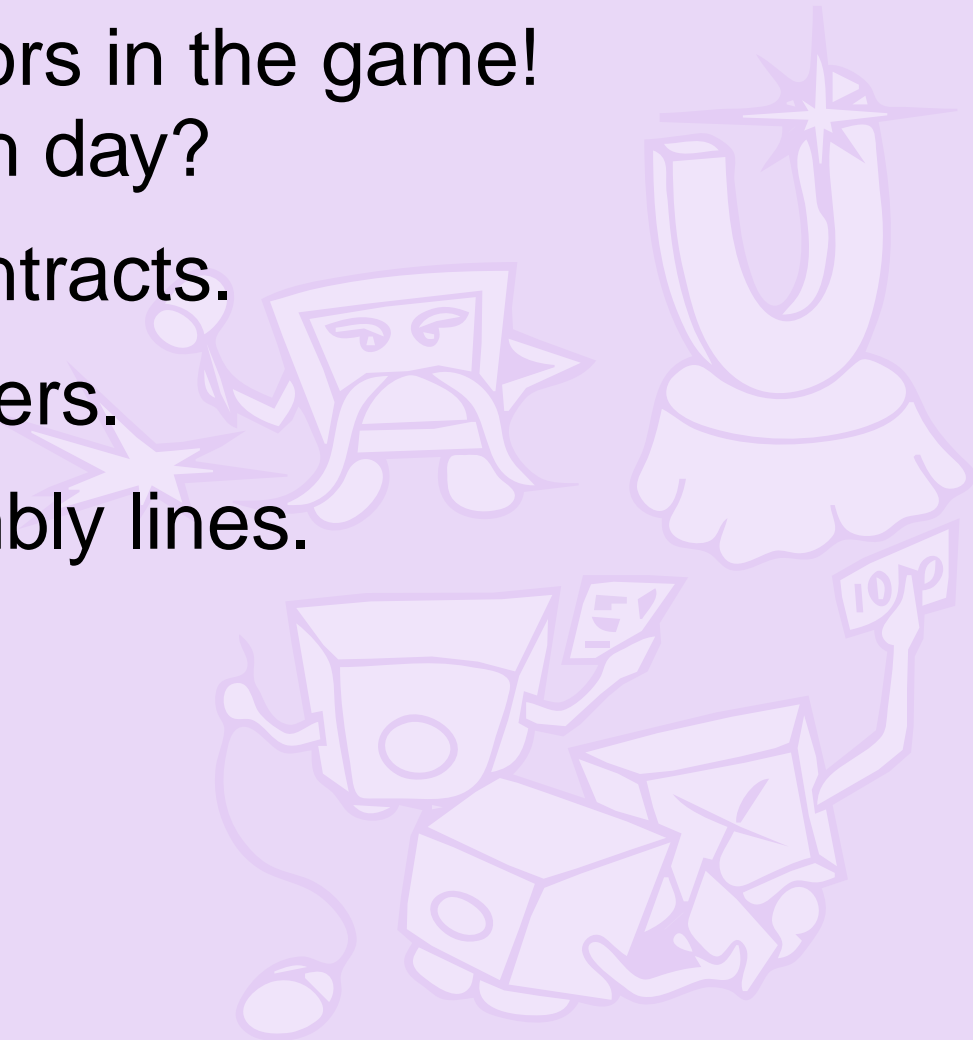
TAC SCM - Customers (3)

- All parameters of each RFQ are chosen uniformly between specified intervals.
- $(Qty_{min}, Qty_{max}) = (1, 20)$
- $(DueDate_{min}, DueDate_{max}) = (3, 12)$
- $(Penalty_{min}, Penalty_{max}) = 5\% - 15\%$ of customer reserve price.
- Customer reserve price = 75% - 125% of components base price.

TAC SCM - Assembly Agents (1)

These are the competitors in the game!
What do agents do each day?

- Negotiate supply contracts.
- Bid for customer orders.
- Manage daily assembly lines.



TAC SCM - Assembly Agents (2)

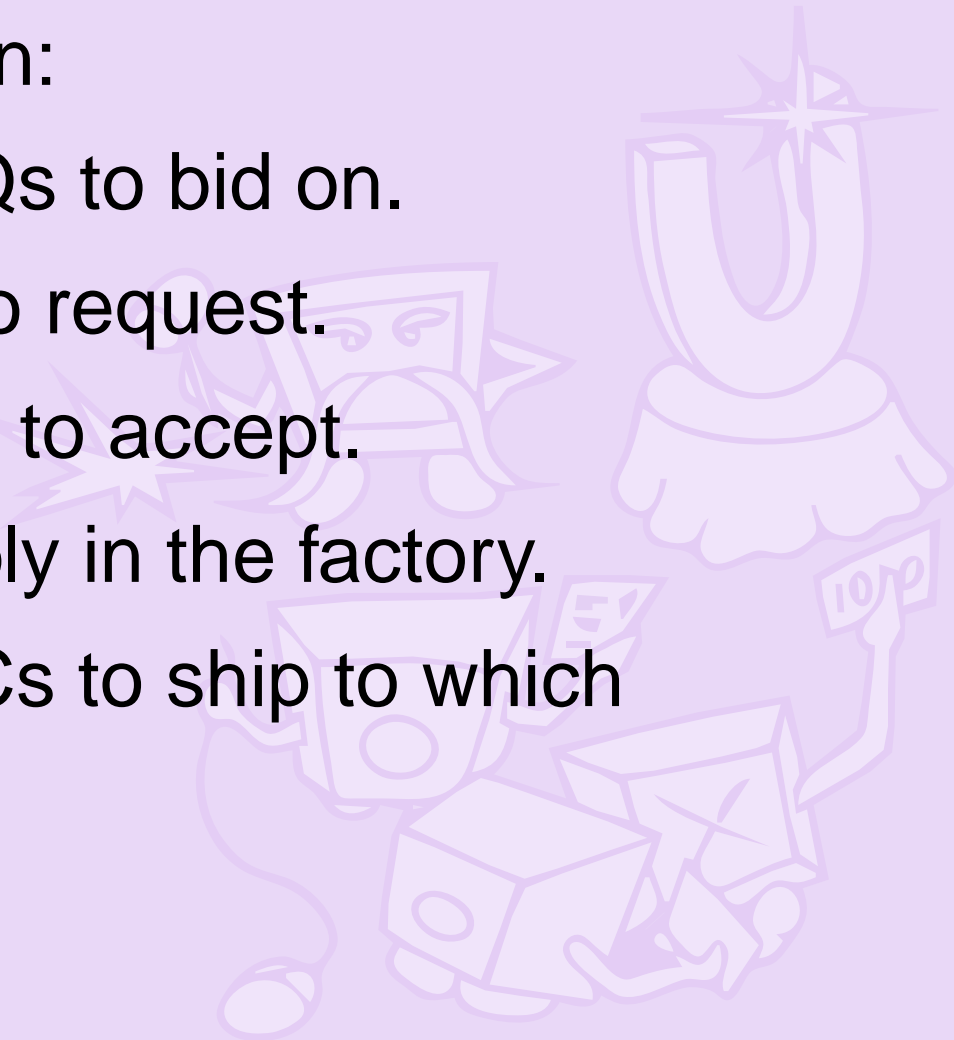
Each day the agents receive:

- RFQs and orders won from customers.
- Quotes and delivery of parts from component suppliers.
- Bank account statement.
- PC and component inventory report from the factory.

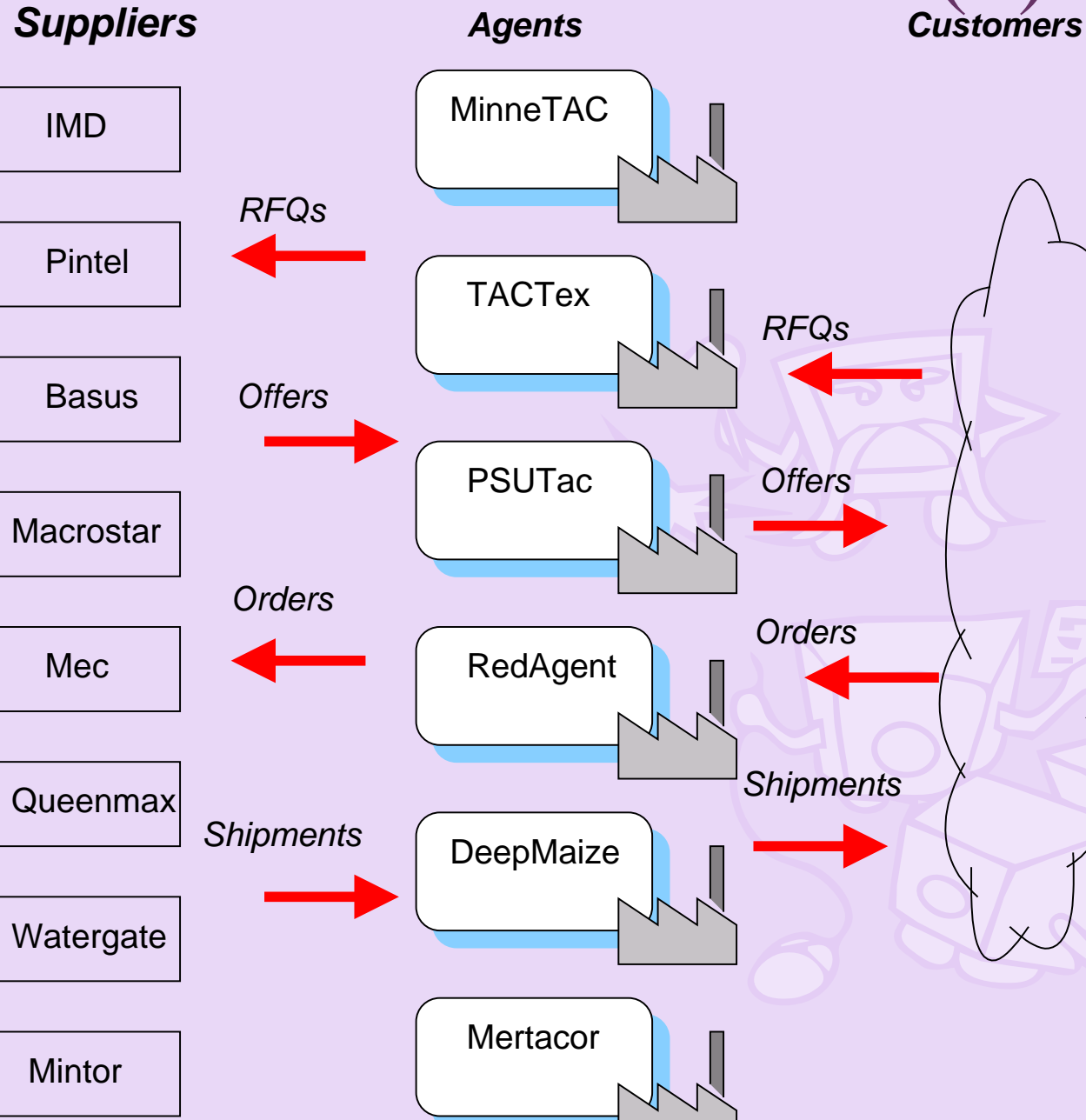
TAC SCM - Assembly Agents (3)

Each day the agents plan:

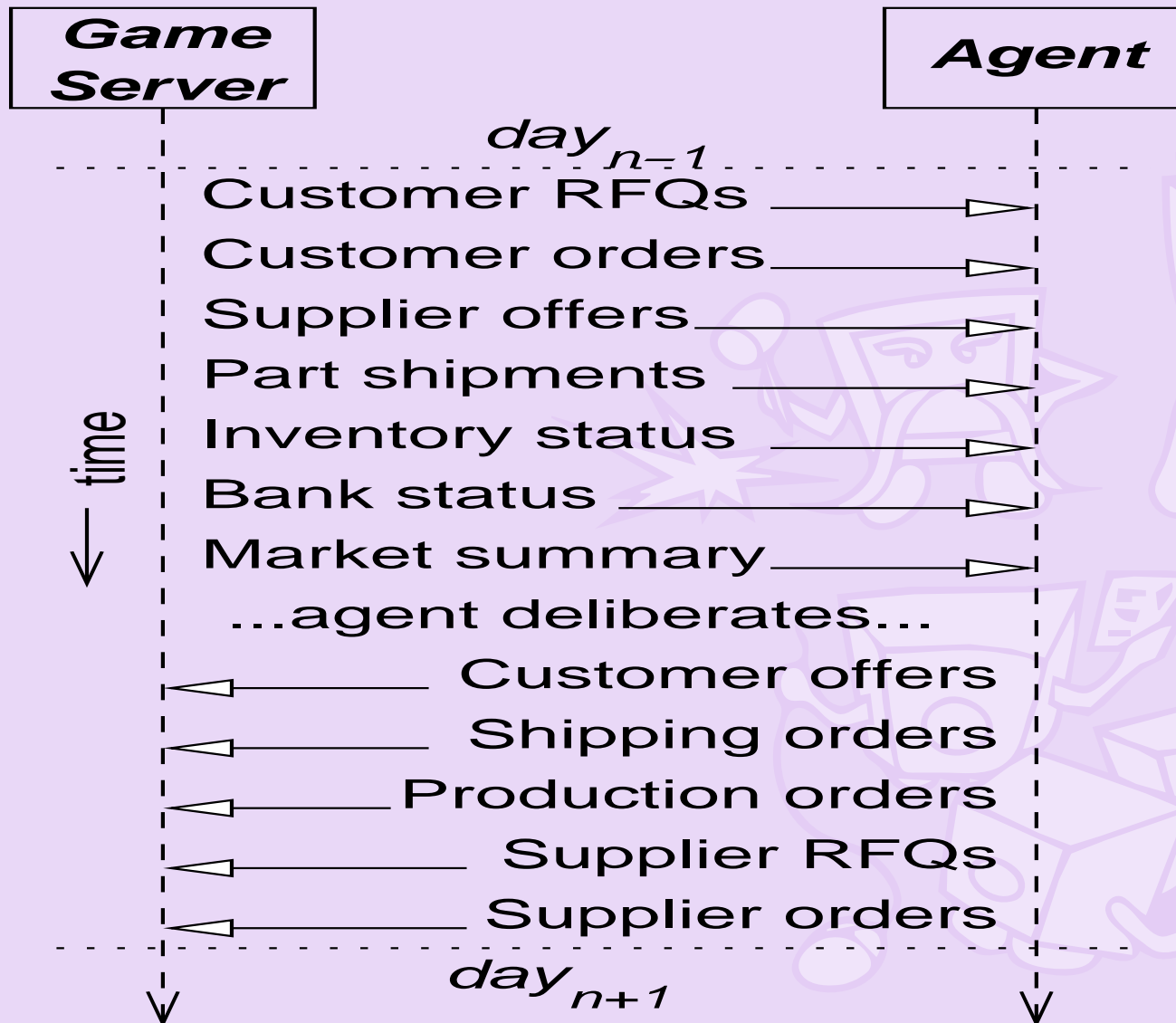
- Which customer RFQs to bid on.
- Which components to request.
- Which supplier offers to accept.
- What PCs to assemble in the factory.
- Which assembled PCs to ship to which customers.



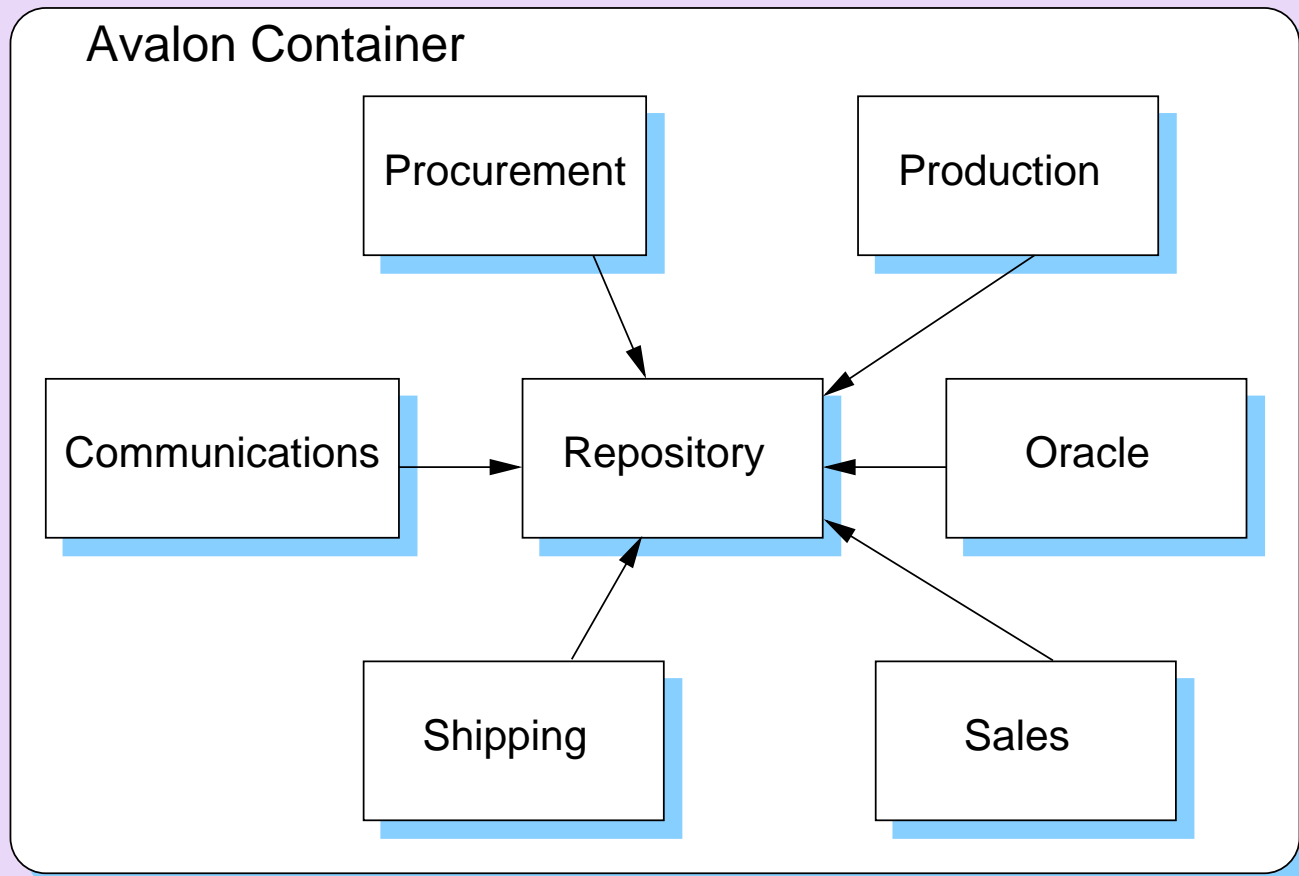
TAC SCM - Scenario (2)



TAC SCM - Communications



MinneTAC - Architecture



Arrows indicate API dependencies.

MinneTAC - Components (1)

Repository is the only component visible to all the other components. It serves as an internal database, maintains the state of the system, and notifies other components of changes in state.

Communications handles the communication with the game server, such as joining games, acquiring initial game parameter and the like.

Procurement procures parts. It may build and maintain target inventory levels, it may attempt to procure parts to meet customer orders, and the like.

MinneTAC - Components (2)

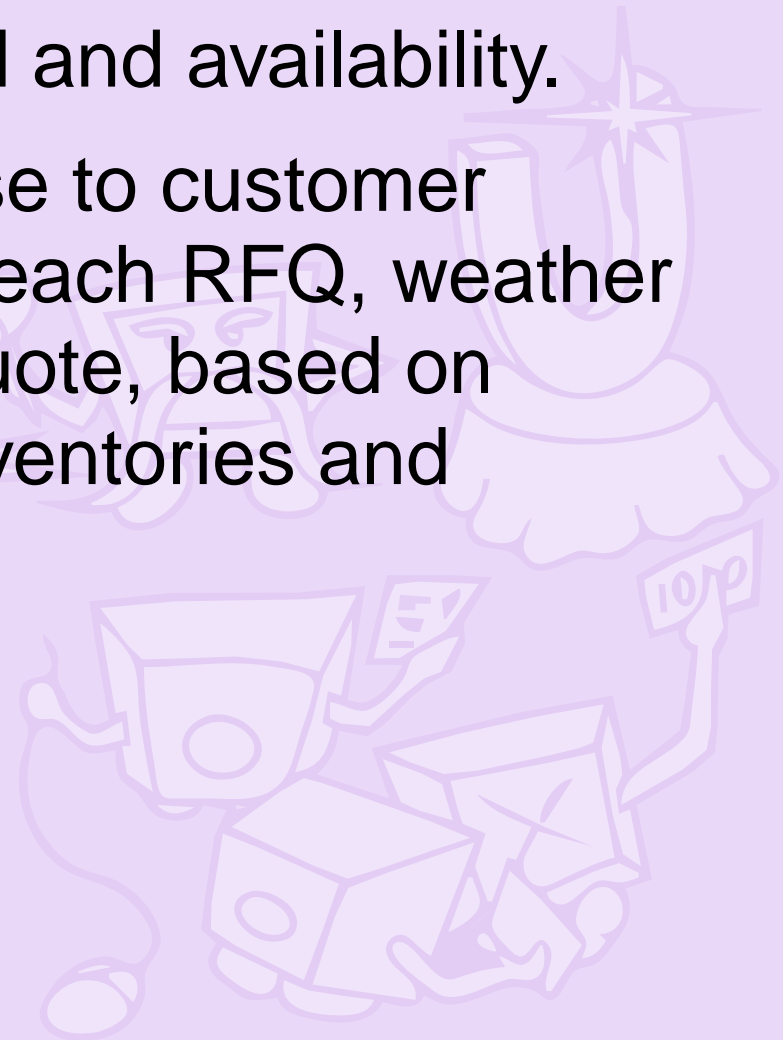
Production schedules the manufacturing facility. It may build and maintain target finished goods inventory levels, or it may build only to meet existing customer orders.

Shipping ships products to customers. In general, there is a benefit in shipping as late as possible, because this gives the agent an opportunity to minimize penalties for late deliveries.

MinneTAC - Components (3)

Oracle predicts future demand and availability.

Sales makes offers in response to customer RFQs. It must decide, for each RFQ, whether to bid and what price to quote, based on available and predicted inventories and current market conditions.



TAC SCM - A Priori Analysis

Bottleneck is the factor which limits PCs production on a day. Types of bottlenecks:

Demand bottleneck if the demand for PCs is less than the agents' production capacities and the amount of available supplies.

Production bottleneck if the limiting factor is the agents' production capacities.

Supply bottleneck if the limiting factor is the amount of supplies.

A supply bottleneck is most likely (from game analysis).

MinneTAC - Performance Analysis

- The a priori analysis led us to decide on a *supply-driven* strategy.
- TAC SCM 2003 - 50% discount of parts on first day
- Order many components up front: good in high-demand games, but bad in low-demand games.

W. Ketter, E. Kryzhnyaya, S. Damer, C. McMillen, A. Agovic, J. Collins and M. Gini. "MinneTAC Sales Strategies for Supply Chain TAC." submitted to Third Int'l Conf. on Autonomous Agents and Multi-Agent Systems, New York, 2004.

Sales Strategies - MaxEProfit (1)

Determines offer price to maximize expected profit margin from a potential customer order x :

$$E[ProfitMargin(x)] = ProfitMargin(x) \times p_{order}(x)$$

where $ProfitMargin = \frac{price - cost}{price}$ with the constraint that $price \geq targetAveragePrice$.

$ProfitMargin$ is calculated on the agent's moving average cost of the components. $targetAveragePrice$ is an internal parameter.

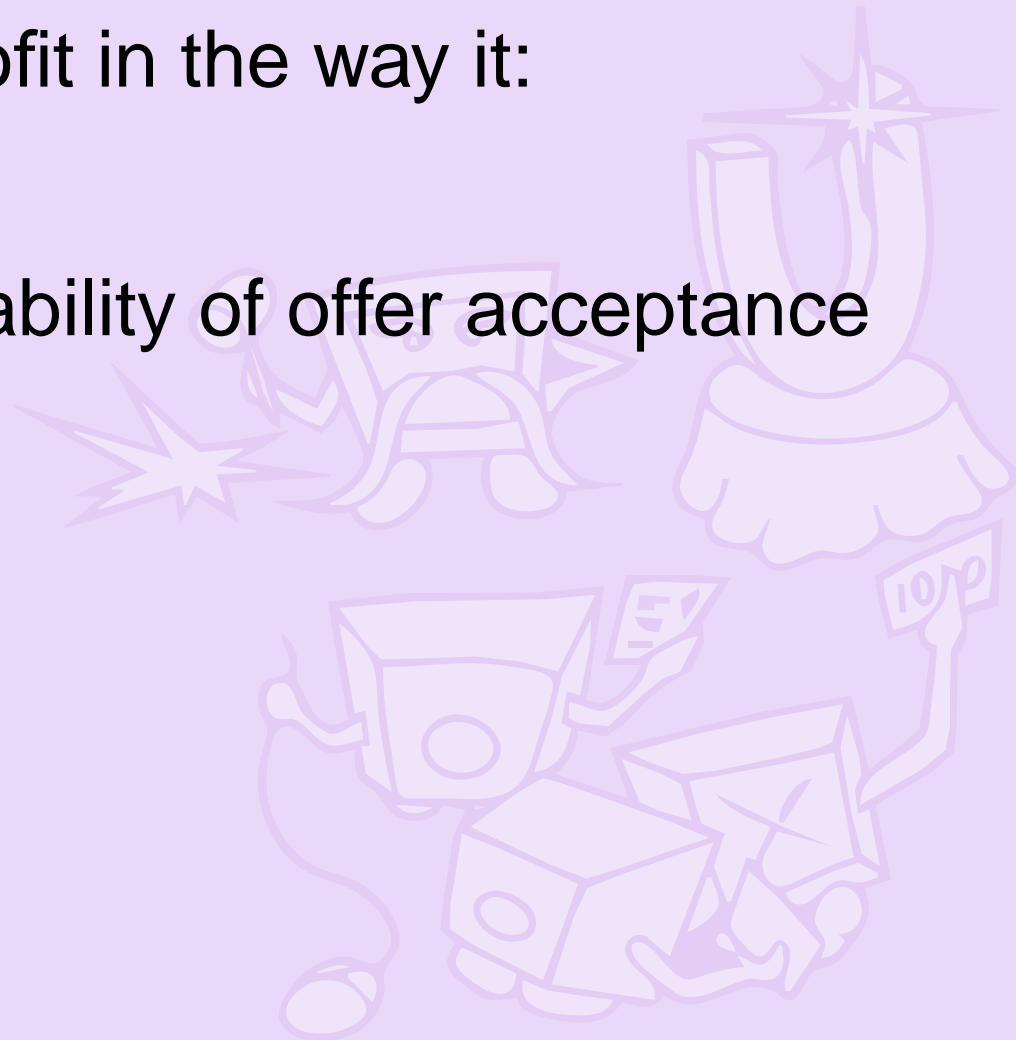
Sales Strategies - MaxEProfit (2)

- Offers are made only from uncommitted finished goods inventory and are sorted by Expected Profit Margin.
- The agent reserves a fraction of the PCs offered according to the estimated probability of receiving an order, p_{order} , i.e.
$$reserved_qty = RFQ_qty \times p_{order}(RFQ)$$
- Inventory is controlled by pulling stock.
- Procurement and production work to maintain target inventory levels until late in the game.

Sales Strategies - DemandDriven (1)

It differs from MaxEProfit in the way it:

1. sets offer prices
2. estimates the probability of offer acceptance



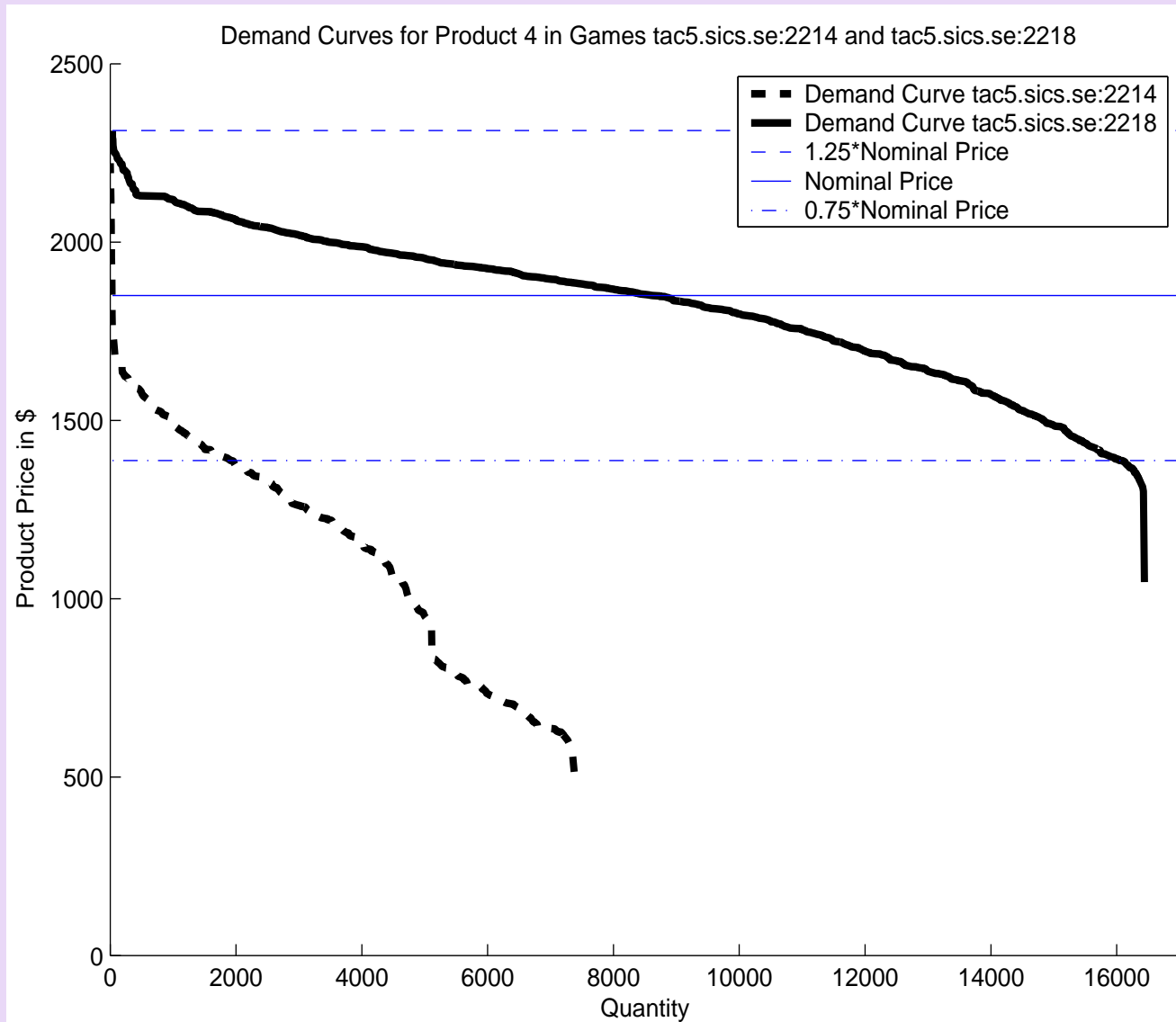
Sales Strategies - DemandDriven (2)

Determines the offer price based on a target probability of receiving a customer order. The *target_prob* is computed from the reverse cumulative density function (CDF) that models order probability. The goal is to make offers to sell out the inventory by the end of the game.

$$target_prob = \min\left(1, \frac{avail_FG + \# built \times days_left}{estimated_demand}\right)$$

where *avail_FG* is the number of finished goods available for a PC type, *# built* is the number of units of the product built each day.

MinneTAC - High-Demand vs Low-Demand Games (1)



MinneTAC - High-Demand vs Low-Demand Games (2)

Minimum, average and maximum score of each strategy in a set of games (In't Conf. on Electronic Commerce 2003). MaxEProfit works better in high-demand games, but is generally worse in low-demand games.

<i>Strategy</i>	<i>High</i>			<i>Low</i>		
	<i>Values (in \$M)</i>					
	<i>Min</i>	<i>Avg</i>	<i>Max</i>	<i>Min</i>	<i>Avg</i>	<i>Max</i>
MaxEProfit	-12.02	12.30	35.99	-66.90	-44.44	-7.36
DemandDriven	-23.65	8.70	30.89	-57.15	-34.49	30.89

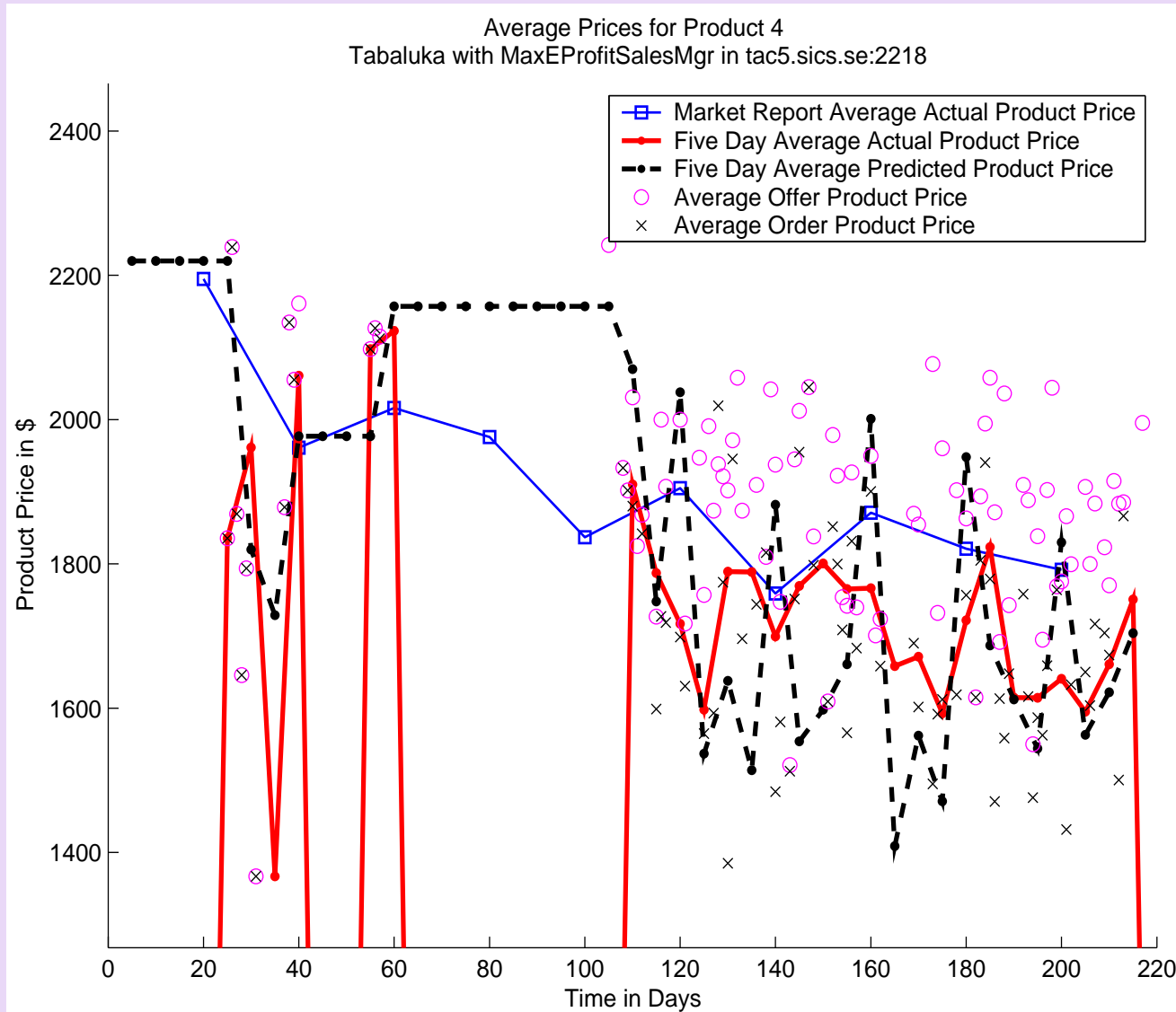
MinneTAC - High-Demand vs Low-Demand Games (3)

Game	Agents and their Result (in \$M)						Demand
	1	2	3	4	5	6	#RFQ/day
2214	team2	RedSox	MinneTAC	arnoch	RedAgent	Eini	99.44
	-10.43	-18.3	-31.06	-34.87	-38.08	-39.83	
2218	Tabaluka	RedAgent	arnoch	MinneTAC	team2	Eini	299.66
	31.23	30.69	23.24	20.8	8.86	7.89	

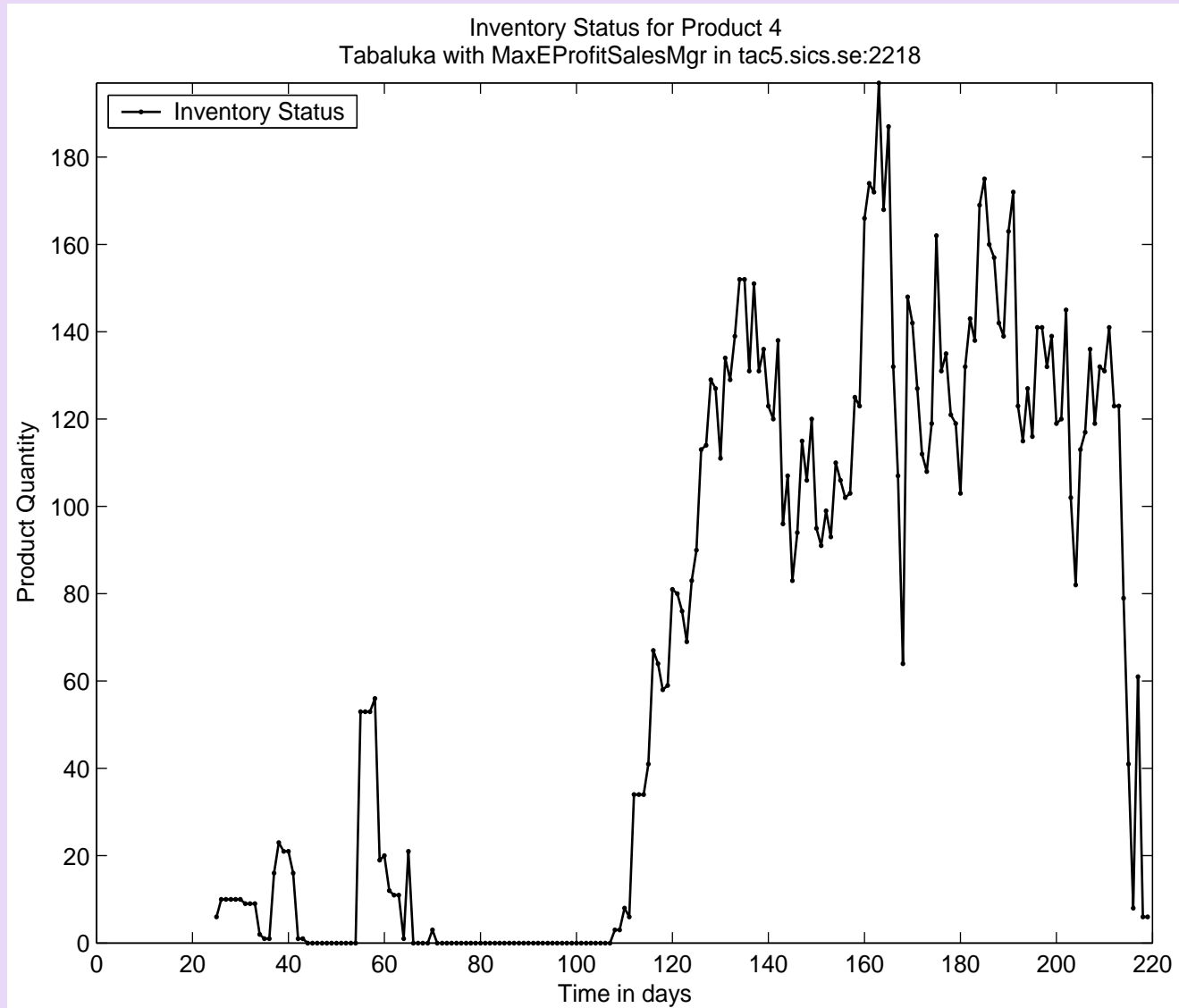
Table 1. Summary of two games. #RFQ/day is mean number of RFQs/day.

Customer RFQs are issued over 219 days in a game. Eini and MinneTAC use DemandDriven and Tabaluka uses MaxEProfit.

MaxEProfit - High-Demand (1)

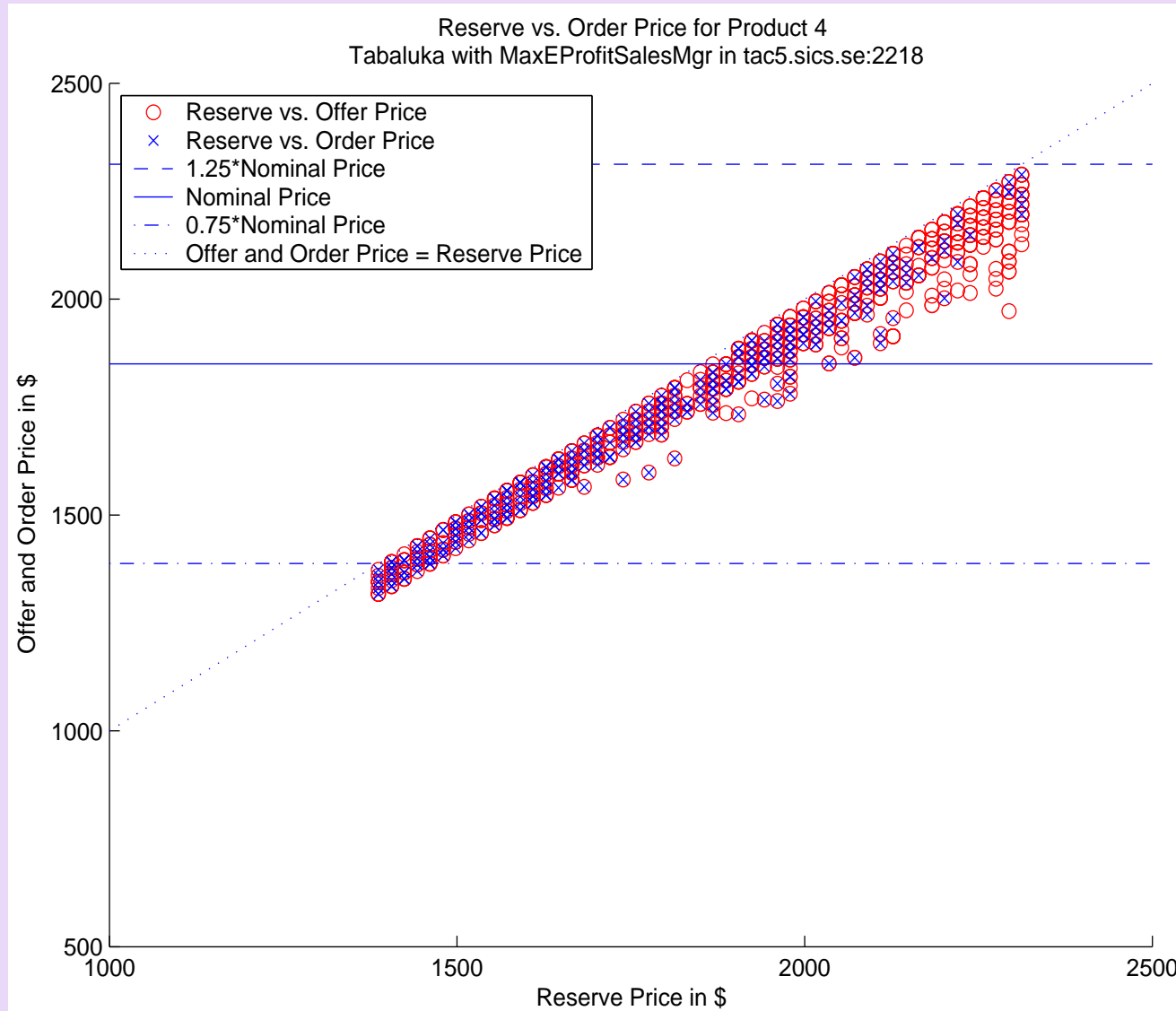


MaxEProfit - High-Demand (2)

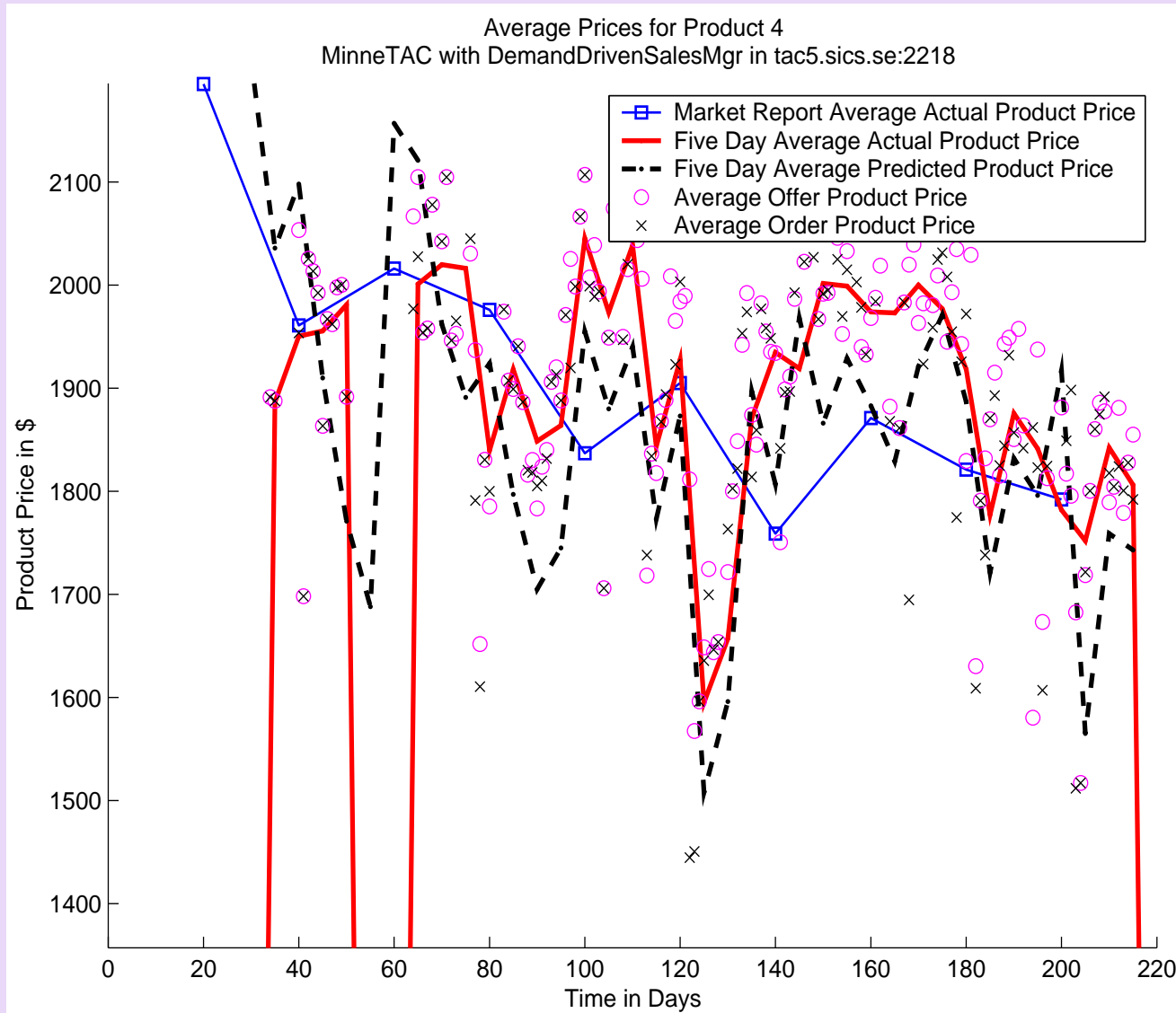


Game 2218 - inventory status: Tabaluka for product 4.

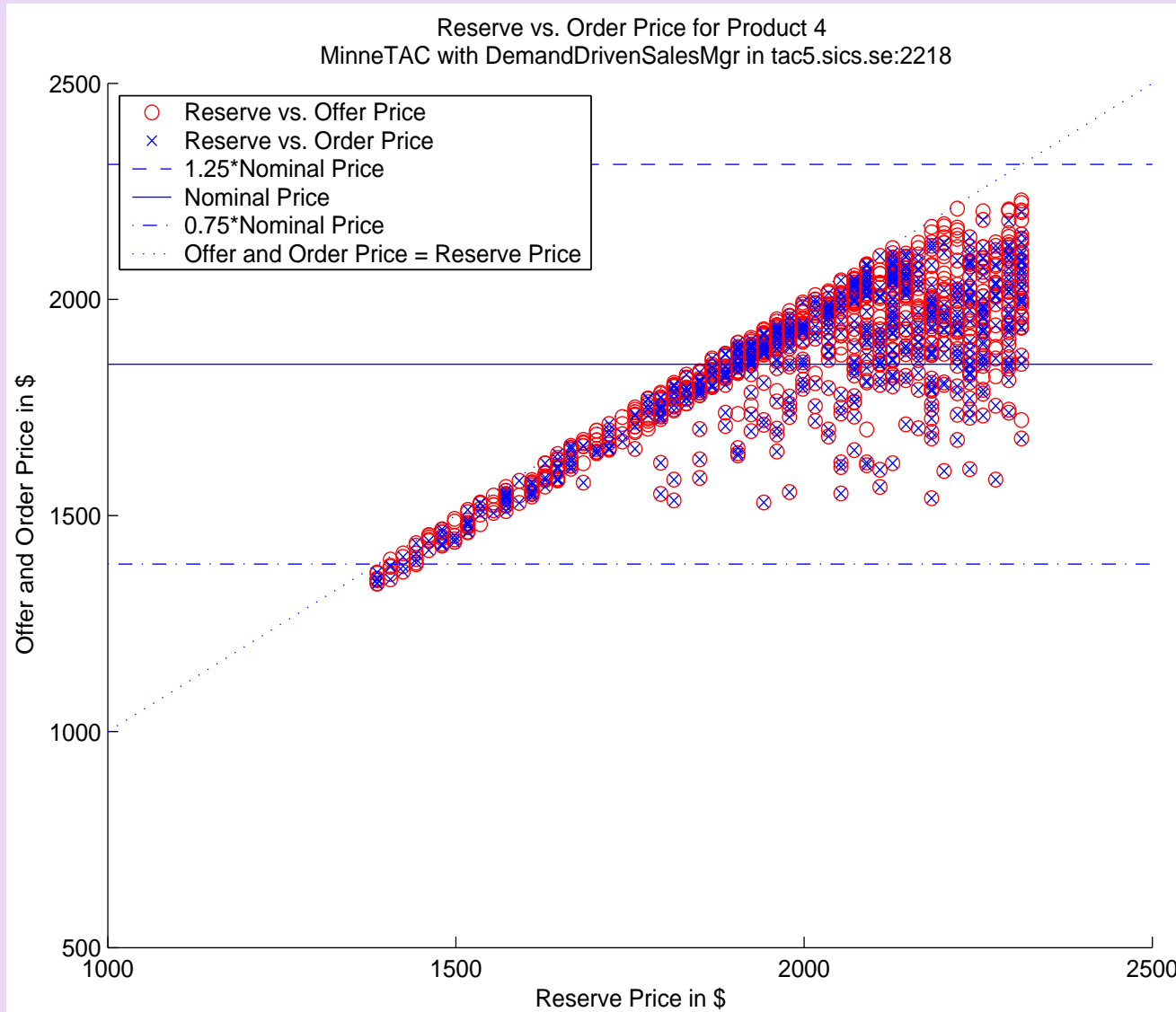
MaxEProfit - High-Demand (3)



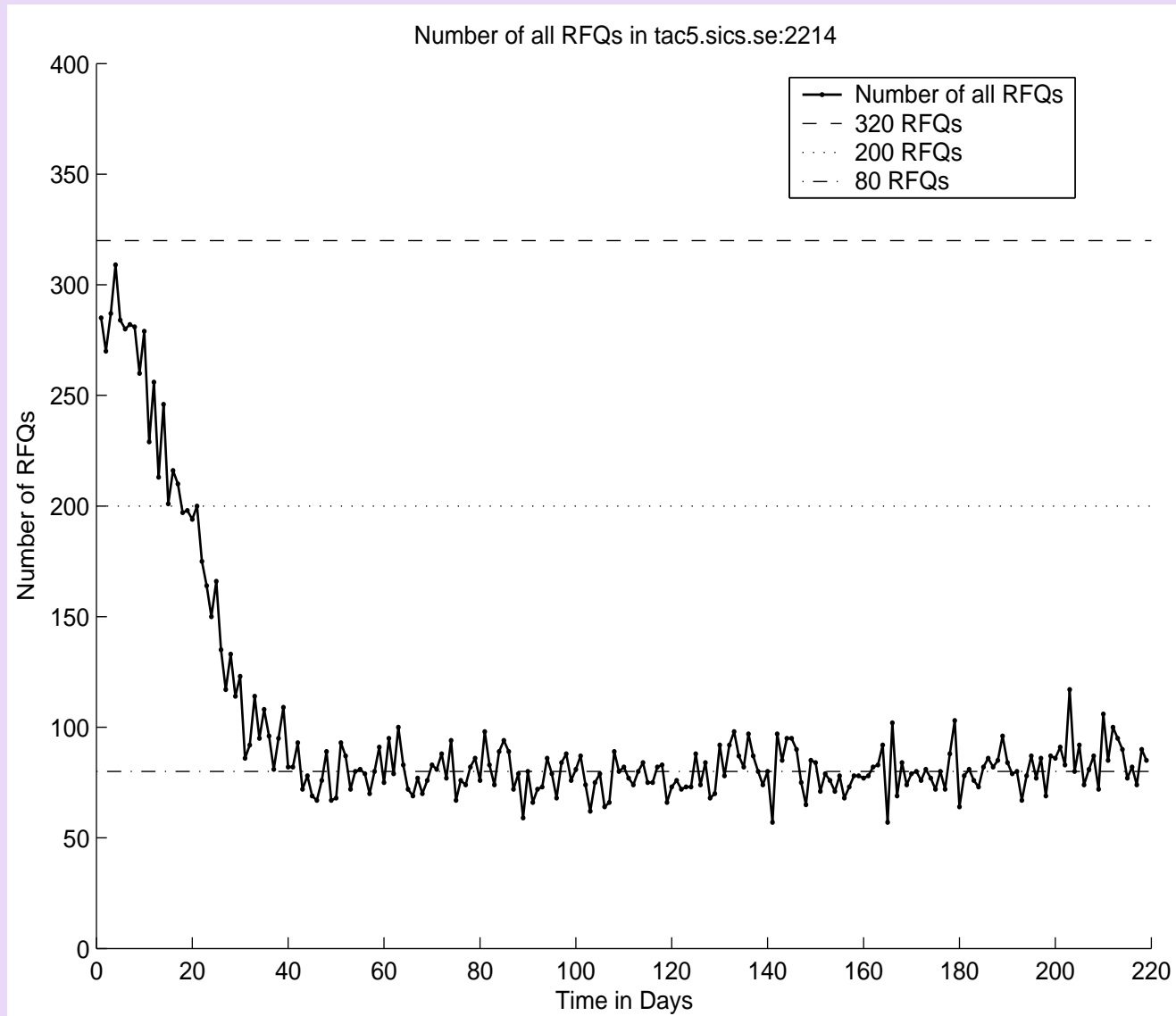
DemandDriven - High-Demand (1)



DemandDriven - High-Demand (2)

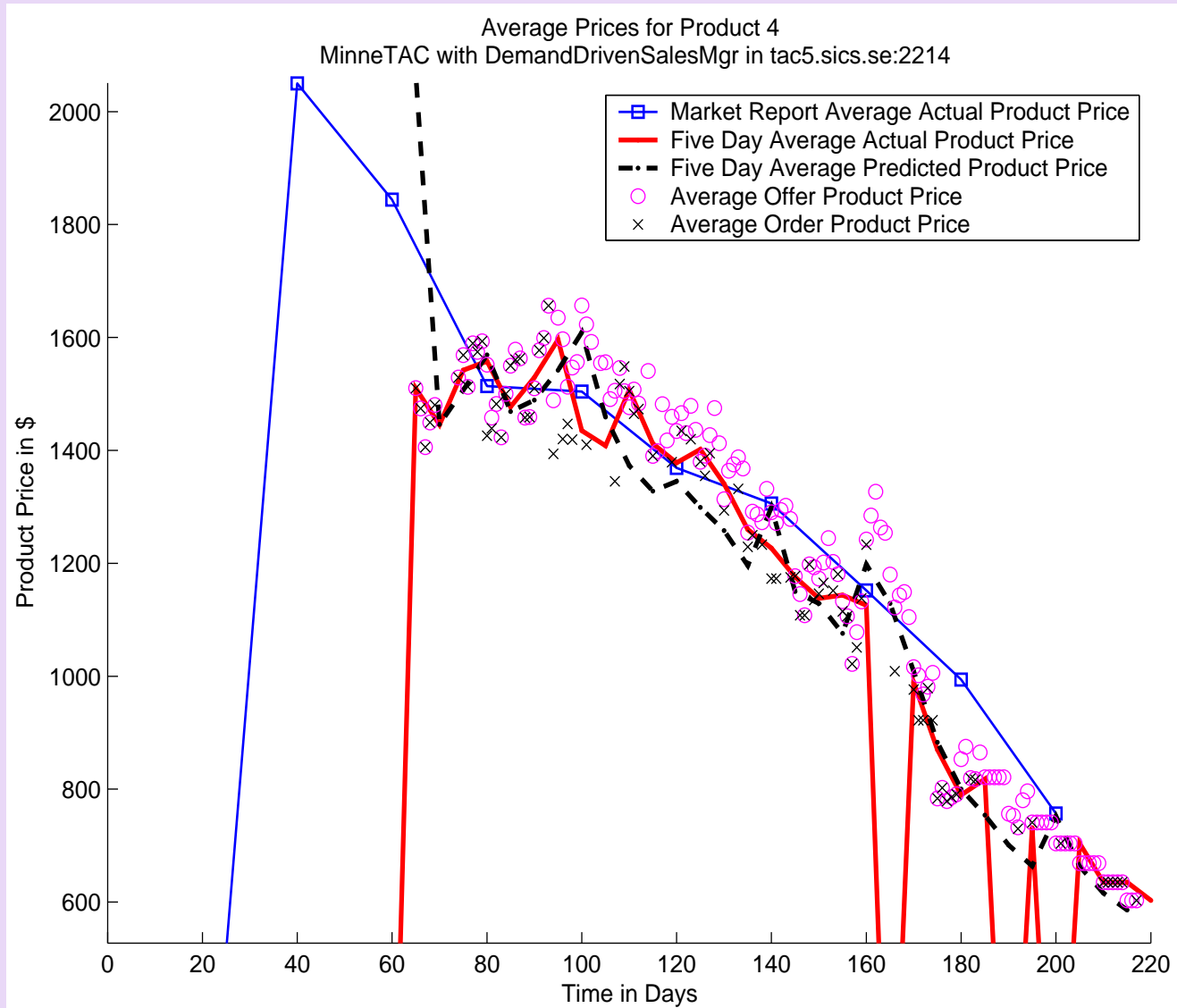


Low-Demand Games

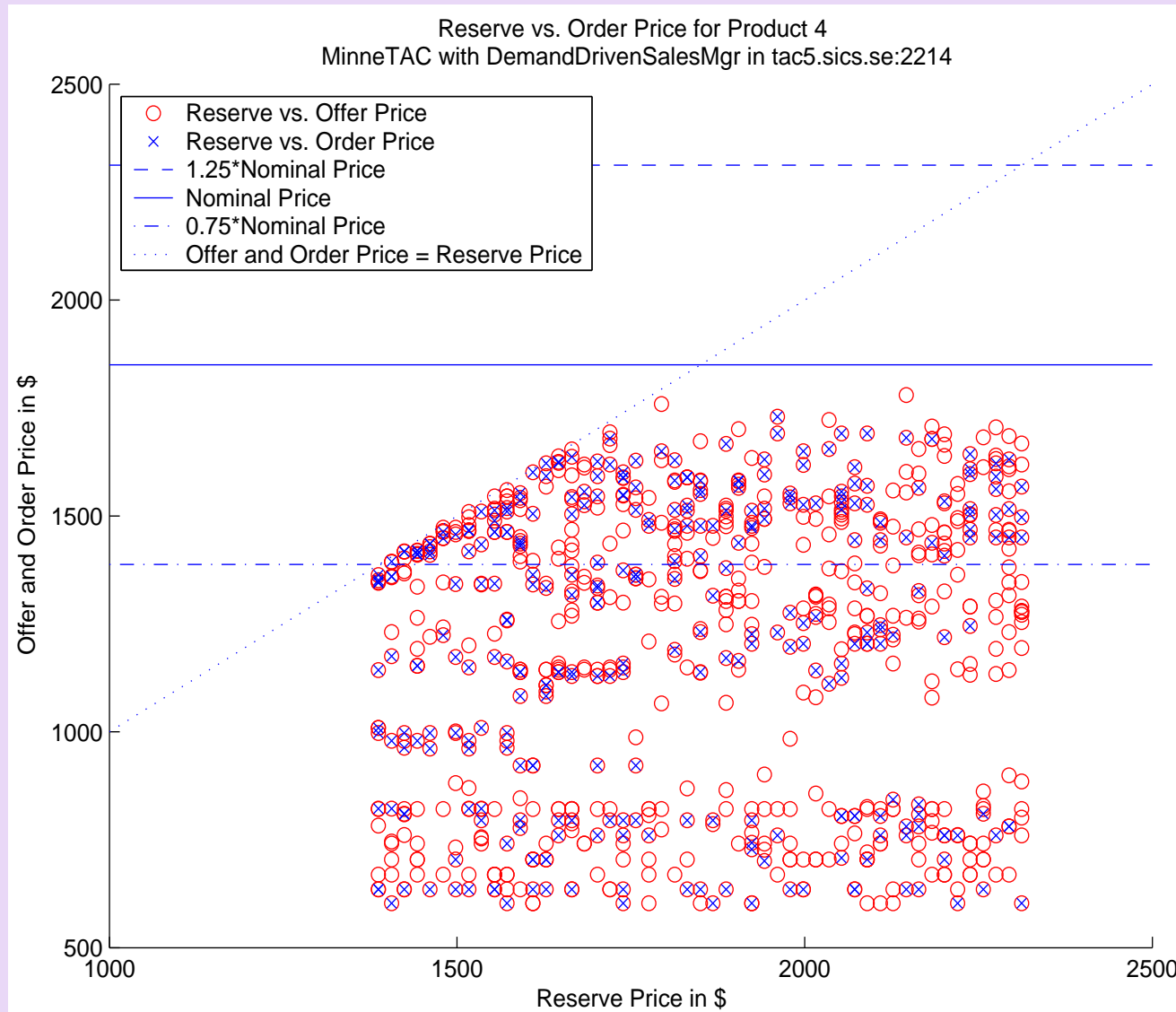


Game 2214 - Total number of RFQs.

DemandDriven - Low-Demand (1)



DemandDriven - Low-Demand (2)



MAGNET: What is Missing in Current Auctions

- Bids can specify only simple constraints between items, typically logical conjunctions or disjunctions.
- There is no notion of time or of scheduling constraints.

Our solution:

- to introduce the use of time in auctions, and
- to combine winner determination with scheduling constraint satisfaction.

MAGNET in a Nutshell

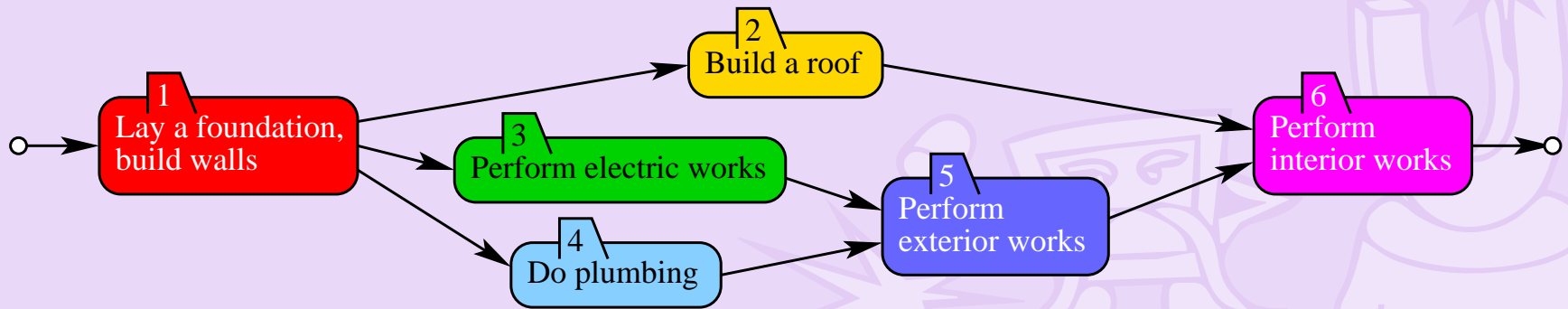
The Univ of Minnesota MAGNET (Multi-AGent NEgotiation Testbed) system supports:

- market-mediated multi-agent interactions;
- multiple agents with different roles (customers and suppliers);
- negotiation of contracts for tasks with temporal and precedence constraints;
- automated first-price sealed-bid reverse combinatorial auctions.

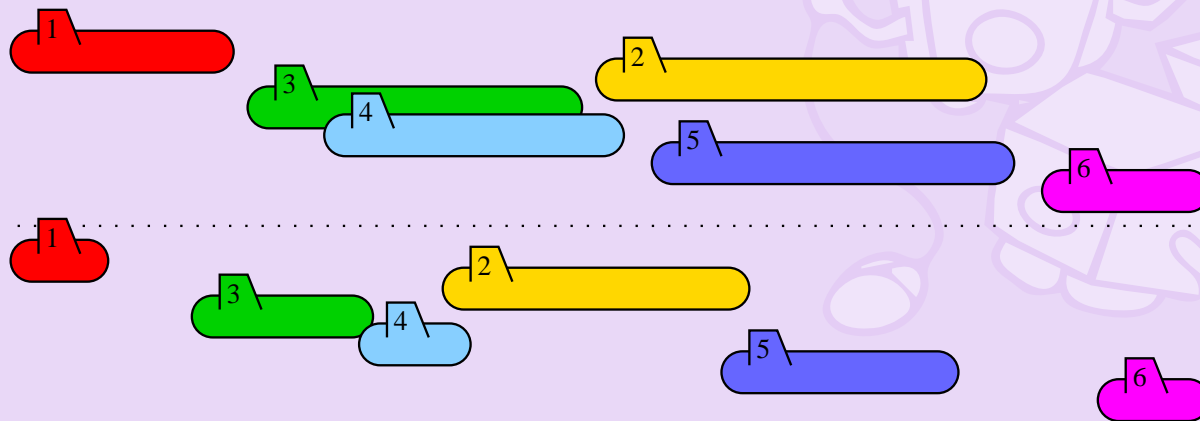
J. Collins, W. Ketter, and M. Gini. "A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints." Int'l Journal of Electronic Commerce, Vol 7, No 1, pp 35–57, 2002.

(1) Task Planning

A customer agent needs to complete a set of tasks with time and precedence constraints.

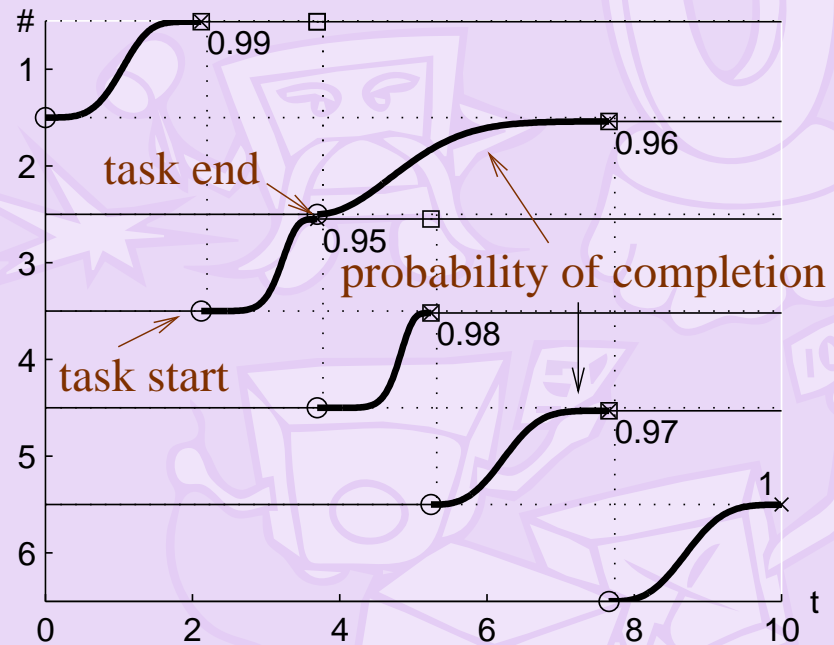
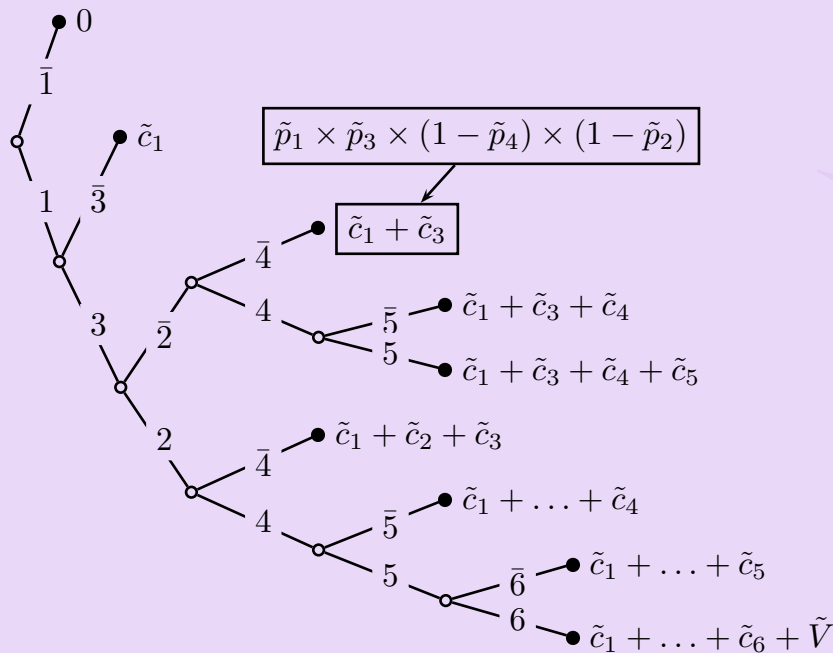


There are many ways of scheduling the same tasks.



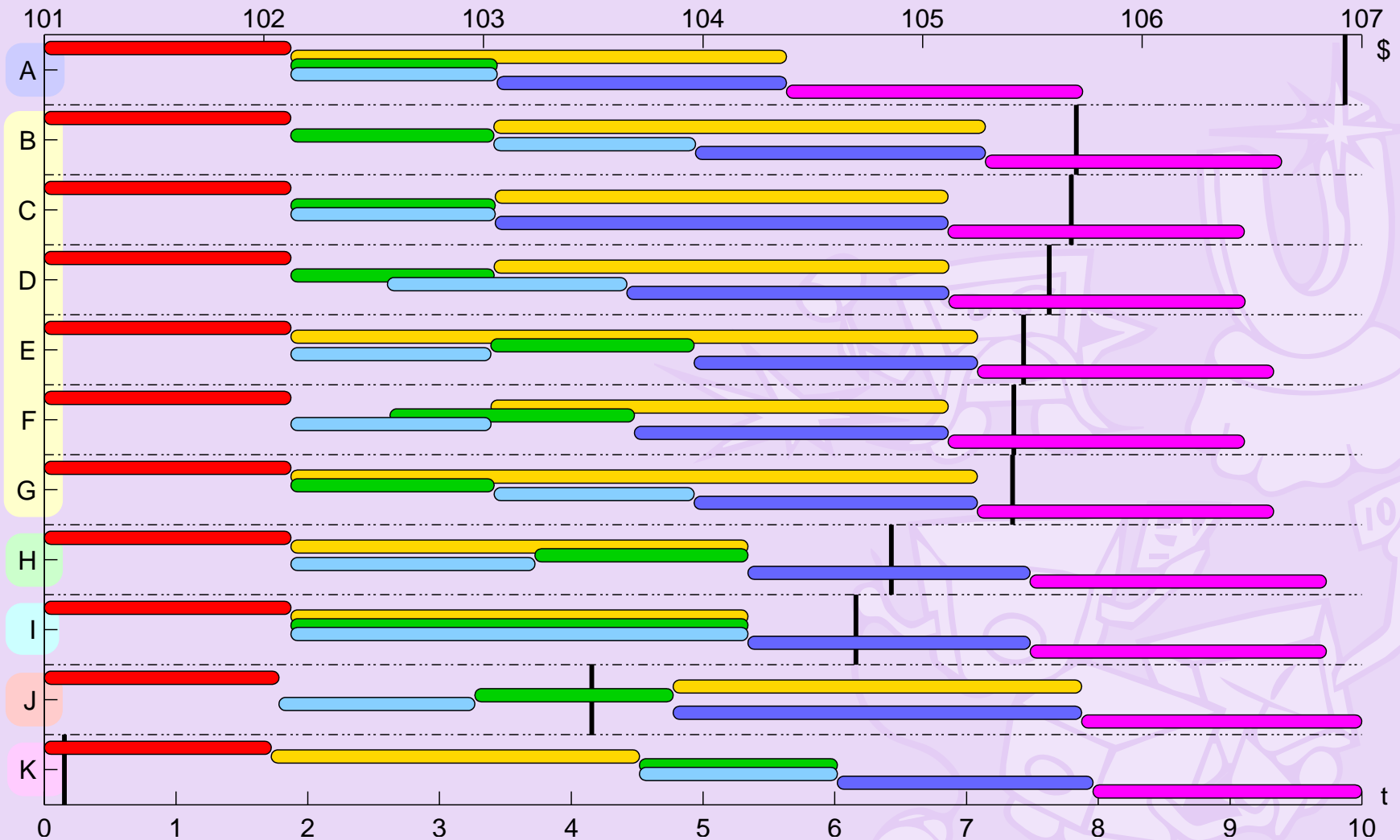
(2) Generation of RFQs

To generate a Request for Quotes (RFQ), the customer decides the ordering of tasks and durations that will maximize its Expected Utility.



Event tree corresponding to an optimal schedule. The outlined event corresponds to the failure of tasks 2 and 4 after successful completion of tasks 1 and 3.

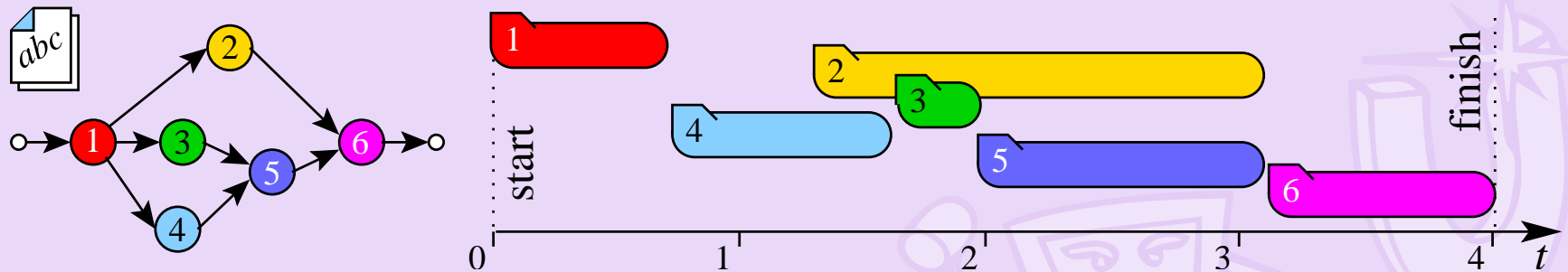
Schedules for RFQ Generation



Optimal schedules for a risk neutral customer by decreasing value.

(3) Bid Submission

Suppliers submit bids in response to a RFQ:



- Bids are for individual and/or combinations of tasks.
- Each bid includes price, duration, and time window within which the task(s) can start.

(4) Winner Determination

The customer selects the best feasible combination of bids and awards them.

The suppliers execute the tasks according to the awarded bids. The execution is monitored, and, if tasks are not completed as expected, rescheduling and/or rebidding takes place.

A. Babanov, J. Collins, and M. Gini. "Asking the right question: Risk and expectation in multiagent contracting." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol 17, No. 4, pp 173-186, 2003.

Formulation of Winner Determination

For m items and n bids

minimize $\sum_{i=1}^n c_i x_i$

subject to $x_i \in \{0, 1\}, i = \{1 \dots n\}$ i.e. Bid Selection

and $\sum_{i:s_j \in \mathcal{S}_i} x_i = 1, \text{ for each } s_j \in \mathcal{S}_r$ i.e. Coverage

and Local Feasibility

where c_i is the bid price for bid b_i , \mathcal{S}_i is the set of items in bid b_i . $i : s_j \in \mathcal{S}_i$ is the set of all i such that task s_j is in the set of tasks \mathcal{S}_i in bid b_i .

Algorithms for Winner Determination

We extended the following winner determination algorithms to deal with time windows:

- Integer Programming (IP): we preprocess precedence constraints to reduce number of constraints.
- A*: heuristically guided search strategy.
- Simulated Annealing (SA) : stochastic method.

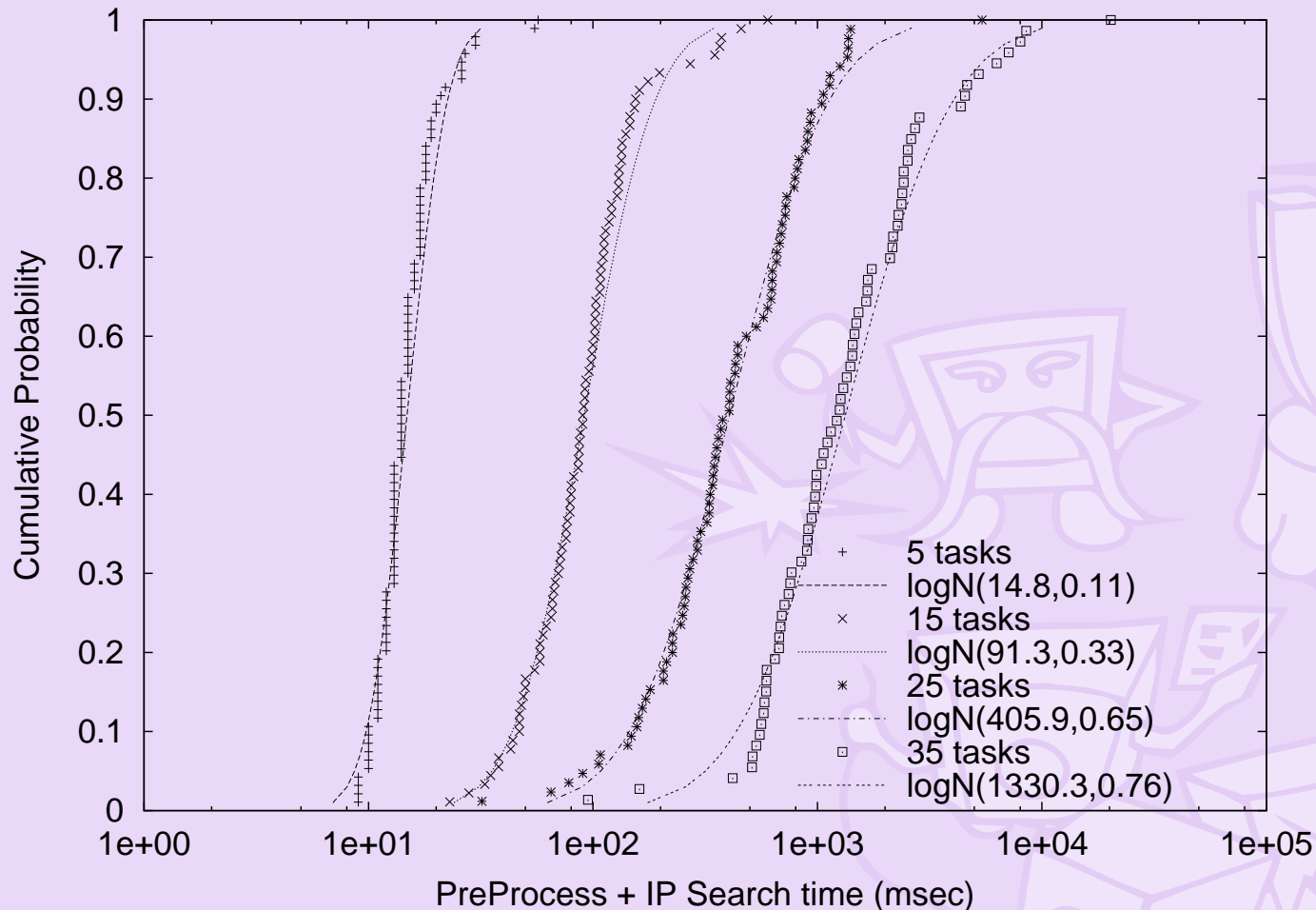
Performance Criteria

Estimate the **probability** that a solution can be found in a given amount of time, based on:

- number of tasks in task network;
- number of bids submitted;
- average **number of tasks specified in each bid**;
- average size of the precedence set of a task in the task network.

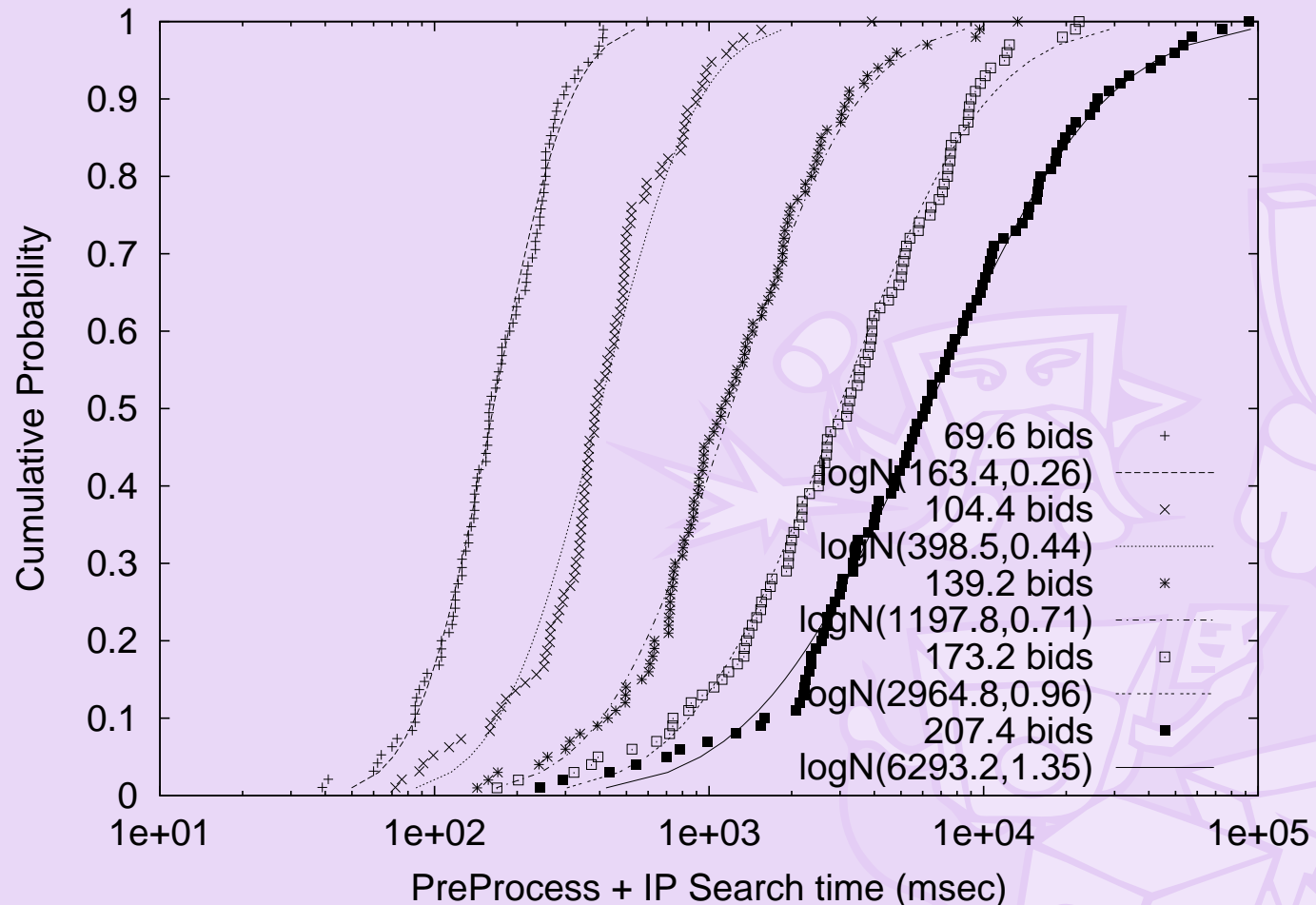
J. Collins. "Solving Combinatorial Auctions with Temporal Constraints in Economic Agents." PhD thesis, June 2002.

Estimating Solution Time (I)



Observed and inferred runtime distributions for Integer Programming given different numbers of tasks, and constant ratio of bids to tasks.

Estimating Solution Time (II)



Observed and inferred runtime distributions for Integer Programming given different numbers of bids, 20 tasks, and ≈ 7.3 tasks/bid.

Conclusions

- Broad study of electronic markets.
- Ability to automate the negotiation process for tasks with complex time constraints and interdependencies.
- Strategies for online supply chain management.

Contacts:

email: magnet@cs.umn.edu

URL: www.cs.umn.edu/magnet

Conclusions

- Broad study of electronic markets.
- Ability to automate the negotiation process.
- Strategies for online supply chain management

Contacts

email: magnet@cs.umn.edu

URL: www.cs.umn.edu/magnet