Rational Bidding Using Reinforcement Learning
An Application in Automated Resource Allocation

Nikolay Borissov[1], Arun Anandasivam[1],
Niklas Wirström[2], and Dirk Neumann[3]

[1] University of Karlsruhe, Information Management and Systems,
Englerstr. 14, 76131 Karlsruhe
{borissov,anandasivam}@iism.uni-karlsruhe.de
[2] Swedish Institute of Computer Science, Box 1263, SE-164 29 Kista, Sweden
niwi@sics.se
[3] University of Freiburg, Platz der Alten Synagoge, 79085 Freiburg, Germany
dirk.neumann@vwl.uni-freiburg.de

Abstract. The application of autonomous agents by the provisioning and usage of computational resources is an attractive research field. Various methods and technologies in the area of artificial intelligence, statistics and economics are playing together to achieve i) autonomic resource provisioning and usage of computational resources, to invent ii) competitive bidding strategies for widely used market mechanisms and to iii) incentivize consumers and providers to use such market-based systems.

The contributions of the paper are threefold. First, we present a framework for supporting consumers and providers in technical and economic preference elicitation and the generation of bids. Secondly, we introduce a consumer-side reinforcement learning bidding strategy which enables rational behavior by the generation and selection of bids. Thirdly, we evaluate and compare this bidding strategy against a truth-telling bidding strategy for two kinds of market mechanisms – one centralized and one decentralized.

Keywords: Bid Generation, Reinforcement learning, Service Provisioning and Usage, Grid Computing.

1 Self-organized Resource Provisioning and Usage

Grid Computing is becoming more and more popular both as a research field, and in the industry. Prominent examples are Sun Microsystems' *Network.com*, Salesforce's *force.com*, *Amazon's Elastic Compute Cloud* (EC2) and its *Simple Storage Service* (S3) as well as *SimpleDB Service*. The companies often offer a fixed pay-per-use price for static resource configurations, which can lead to inefficient utilization, profit and usability, as it does not reflect the dynamics of the market supply and demand. Efficient provisioning and usage of computational resources as well as pricing in a thriving environment like Grid Computing is not manually manageable. Such processes should be automated with no or minimal

J. Altmann, D. Neumann, and T. Fahringer (Eds.): GECON 2008, LNCS 5206, pp. 73–88, 2008.

human interaction. Hence, market mechanism and strategic behavior play an important role for the design of the environment. Self-organization and automatic adaptation to the changing market conditions are key prerequisites for efficient allocation of computational resources [1].

To efficiently allocate consumers' jobs to providers' resources is a complex task where participants' decisions on resource provisioning and usage are executed online. Moreover, the wide heterogeneity of computational resources, complicates the process of finding an appropriate set of resources for given consumer's preferences. Since demand and supply of computational resources fluctuates in the course of time, information about current and future resource utilization and prices are often not known a-priori to the participants. In this case consumer and provider agents try to maximize their utilities by generating bids based on their valuations and historic experiences [2]. This enables strategic behavior both on provider and consumer side.

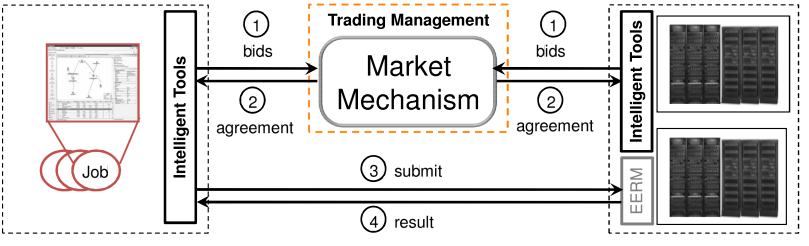

Fig. 1. Self-organized service offering and requesting

This paper is written in the context of the SORMA[1] project, with the focus on components and methodologies that constitutes the SORMA *Intelligent Tools*. Figure 1 depicts the approach taken by SORMA for automated provisioning and usage of computational resources. As illustrated, consumers and providers use the intelligent tools to generate and submit bids to the *Trading Management* component which executes a collection of different market mechanism. When an agreement has been made, the two parties are informed of this. The process of submitting and executing the job is exercised by the intelligent tools on the consumer side and the *Economically Enhanced Resource Management* (EERM) on the provider side.

This paper is structured as follows: Section 2 describes components and methodologies of the *Intelligent Tools* supporting the automated bidding process. Section 3 introduces the Truth-Telling strategy, and proposes a novel consumer-side reinforcement learning bidding strategy – the Q-Strategy. In Section 4 we evaluate this strategy against the Truth-Telling strategy using two different market mechanisms and show the convergence of the proposed Q-Strategy. Section 5 discusses related work and Section 6 concludes this paper.

[1] SORMA: Self-Organizing ICT Resource Management, `www.sorma-project.org`

2 Automated Bidding

Within this section we investigate the processes of automated bidding on provider and consumer sides and describe the tools supporting these. In SORMA, a set of *Intelligent Tools* are provided, which purpose is to assist consumers and providers by the description of their technical and economic preferences as well as by the automated generation of bids and offers.

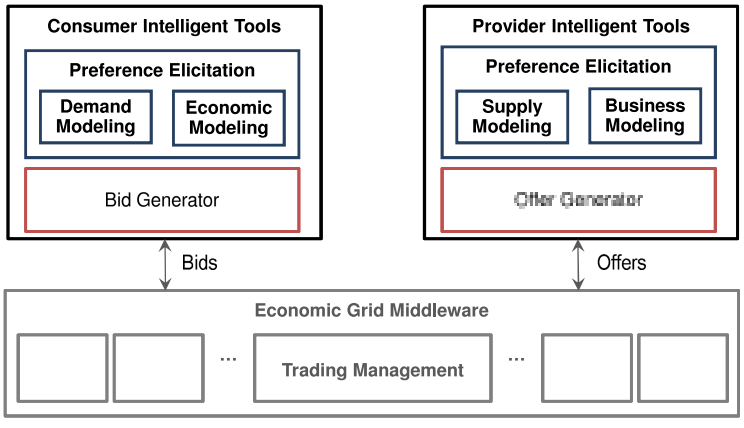

Fig. 2. Intelligent Tools for provider and consumer

Figure 2 shows the intelligent tools for the provider and consumer. In order to derive and describe their preferences, a consumer uses the *Demand Modeling* and *Economic Modeling* tools. The *Demand Modeling* component supports the consumer by the specification of the technical requirements on resources, such as CPU, Memory and Storage of her application. For this it offers a GUI for entering the technical requirements which are formatted in a predefined resource description language such as the *Job Submission and Description Language* [3]. *Economic modeling* allows consumers to describe their economic preferences that will determine their bidding strategies e.g. specifying the valuation of the required resources, validity of a bid and the preferred bidding strategy. The *Bid Generator* is the "intelligent" component that autonomously generates and places the consumer's bids on the market. For this purpose it considers the afore specified consumer preferences and the current state of the market, such as actual prices. The bid generator is implemented through agents using common and novel bidding strategies and learning algorithms. The bids are submitted to the *Trading Management* component, which implements mechanisms for technical and economic matching. Similar to demand modeling, *Supply Modeling* aids the provider to describe the technical specification of the offered resources. Within the *Business Modeling* component, providers have to describe the desired business models, which determine what bidding strategy to be used for the generation of their offers. For example one part of such a description is

the pricing policy that specifies if the consumer has to pay for booked timeslots or for the actual usage. As the example indicates, the models specified by means of this component depend on the implemented market mechanism. Analogously to the bid generator, the *Offer Generator* is the component implementing the bidding strategies for the generation of the offers. The offers are assembled from the technical resource descriptions and the business model of the respective provider. The offer generation component also submits offers to the *Trading Management*. The market mechanisms are implemented within the *Trading Management* component, which is a part of the SORMA *Economic Grid Middleware*. In the following sections we will focus on the components of the *Intelligent Tools* and present them in more detail.

2.1 Preference Elicitation

In order to request computational resources for its application or job, a consumer has to make some estimations regarding preferred technical resource description, QoS and a price. On the other side, a provider has to make a price estimation for its offered resources. These consumer and provider preferences for a specific application can be either static or dynamic. Static information is collected once, and can be manually provided by the consumer or provider each time a resource has to be acquired or offered. These static information may also be stored in databases enabling intelligent tools to use this information for predicting requirements of applications and services with similar properties. However, the requirements of a given application are often subject to change as technology evolves e.g. consumers' desired quality of streamed video might increase as their Internet bandwidth increases. It is thus desirable to dynamically adapt the resource requirements. In [4] the authors specify a model for *Job Valuation Estimation* using evolutionary techniques. The presented approach is based on the assumption that a consumer who wants to buy a set of resources does not generally know her exact valuation for them, but has only a rough estimate of her true valuation. Thus, she decides whether to accept the offer, or to continue her search and look for alternative offers.

Going through following iterative process steps, the bidding agent can approximate the results and successively refine its estimate:

1. *Initialization:* The market participant (consumer, provider) or bidding agent initially assumes a valuation of v_0^A for an application A based on its resource specification x^A. It is assumed that the current price for each resource is published by the market mechanism in a price vector $p_0 \in \Re_+^n$, and an initial weight $\Theta_0^A \in N$ set by the user: $v_0^A(\Theta_0^A, p_0, x^A) = \Theta_0^A p_o^T x^A$ For each run of the application j:
2. *Bidding:* the bidding agent bids on the market according to its estimate $v_j^A(\Theta_j^A, p_j, x^A)$ and selected bidding strategy.
3. *Refinement:* There are two possible outcomes of the bidding process:
   - *Successful:* The bidding agent obtains the necessary resources and reports this information to the participant. The participant then indicates

whether he is satisfied with the outcome or not, and the bidding agent refines its estimate.
- *Unsuccessful:* The bidding agent was not able to obtain the necessary resources. If the participant indicates that the price was indeed too high, the bidding agent does not update its estimate. If the user indicates that he would have preferred to pay that price rather than not getting the resources ($\Theta^A_{j+1} = 0$), the bidding agent updates its estimate of v^A with the current price.

The bidding agent will iteratively try to converge its estimate to the participant's true valuation and at the same time to assists her in identifying a bid rather than forcing the participant to directly reveal its valuation.

2.2 Demand Modeling

The *Demand Modeling* component supports consumers and providers on editing their estimated preferences – technical requirements on resources, such as CPU, Memory and Storage, QoS and price. The component implements a GUI for entering the consumer or provider preferences and generates a XML output in form of predefined resource description language such as *Job Submission and Description Language* [3].

The main parts of this component are:

- *User Interface* to allow the input of technical resource specifications on consumer and supplier side
- *Matchmaking library* implementing methods and algorithms for technical matching of resource requests to offers and
- *Service Description Language* that is able to express service specifications traded on the SORMA marketplace

The service description language contains a high-level specification of the service to be run on the requested resource. To allow different levels of abstraction and granularity, the resource needs to be technically specified in terms of its grounding hardware and the required software environment. Together with the technical specification comes the specification of several non-functional technical resource properties as for example the quality of service. Beyond that, further sections like economic parameters, job-specific parameters or possible inter-job dependencies complete the resource specification.

2.3 Bid and Offer Generator

In the SORMA Grid Market scenario each consumer and provider is configuring and using the intelligent tools i.e bidding agents in order to use or provide resources with the objective to maximize its own utility, and thus it acts rationally.

The bid and offer generator component is implemented within SORMA's *Bidding Agent Framework*, which core classes and relations are illustrated in Figure 3. The framework is defined and discussed in more detail in [5].

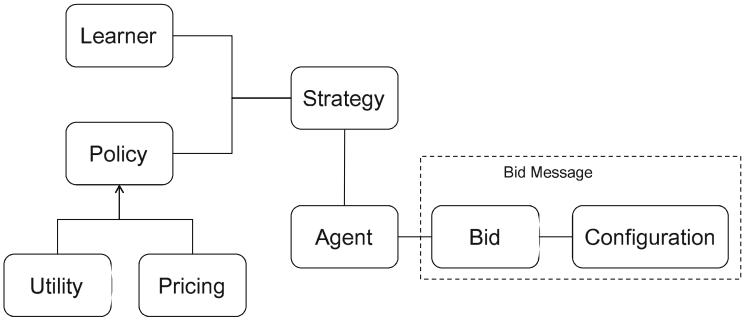

Fig. 3. Framework for Automated Bidding

According to the framework, a bidding *strategy* implements the bidding behavior of the bidding agent, e.g. how, when and what to bid. For this purpose it adopts learning algorithms to learn from earlier actions and predict future rewards by selecting a particular price for a given resource description.

Furthermore a strategy profile can be configured with policies, which are defined in a rule description language and executed within a rule engine. *Policies* in our case are *utility* and *pricing* functions, which are defined externally to the implementation and thus enable a flexible modification. Through the utility function, the participant specifies the overall objectives as a mathematical function that is to be maximized by its bidding agent. For a job j, the pricing policy enables a static specification of a valuation v_j or price calculation function, which is used by the bidding strategy to calculate the bid $\tilde{v}_j \leq v_j$, which is reported to the SORMA Grid Market.

For example, in the case of allocating computational resources for a job execution, a common utility function in the scheduling literature minimizes the weighted completion time $w * C$ of a job, where the weight w represents the importance of a job expressed in some unit. In the economic literature the unit is often a price or so called valuation. The authors of [6] propose a utility function $u_{i,j} = -w_j * C_{i,j} - \pi_{i,j}$, which forces the minimization of the weighted completion time and payments in their particular machine scheduling mechanism, where the weight w_j is the waiting costs of the job j, $C_{i,j}$ the reported completion time of machine i and $\pi_{i,j}$ the amount to be paid to be scheduled on that machine.

The core classes *Bid* and *Configuration* define together the format of the overall bid message [5]. The bid message contains in the consumer case, the consumer's technical and economic preferences and in the provider case, the provider's technical and economic description.

3 Bidding Strategy

To implement a strategic behavior on the consumer side, we implemented consumer agents using two rational bidding strategies, a *Truth-Telling* and a *Reinforcement Learning* bidding strategy. The reinforcement learning bidding

strategy is implemented through the Q-Learning algorithm with epsilon-greedy selection strategy (see Section 3.2).

3.1 Truth-Telling

In the model of [6], agents do not remember the outcomes of earlier market interactions but are somewhat "myopic" in the sense that they only consider the current situation. As shown for the model in [6], without knowledge about the future, at time $\tilde{r}_j$ it is a utility maximizing strategy s_j for agent $j \in J$ to report its true type t_j to the system and to choose the machine i which maximizes $\hat{u}_j(i|s_{-j}, \tilde{t}_j, t_j)$.

The *Truth-Telling* bidding strategy places a bid price, which equal to the provider's or consumer's valuation for a certain resource. Although it is a simple strategy, truth-telling is essential in case of incentive-compatible mechanisms, where this strategy guarantees to obtain the optimal pay-offs for consumers no matter what strategies are adopted by the others. In budget-balanced double-auction mechanisms, this strategy is not dominant [7].

3.2 Q-Strategy

Our aim is to develop rational agents with learning capabilities which may strategically misreport about their true valuation based on previous experiences. We refer to these strategies as "rational response strategies".

Algorithm 1. Q-Strategy: Bid Generation Rule

Require: $economicpreferencesofthejob$
if $\epsilon < Stochastic.random(0,1)$ **then**
  $//Explore:$
  $scale \in (0,1)$
  $price = Stochastic.random(scale * job.getValuation(), job.getValuation())$
else
  $//Exploit:$
  $state = State.getState(job)$
  $action = qLearner.bestAction(\text{state})$
  if $action! = nil$ **then**
    $price = action.getBidPrice()$
  else
    $price = job.getValuation()$
  end if
end if

The Q-Strategy consists of two algorithms (see Algorithm 1 and 2). The first algorithm describes the case where an agent generates a bid (or offer) for a given configuration of resources it wants to buy (or sell). The second algorithm applies to the case where an agent receives a number of offers for a given configuration of resources, and has to select one of them to buy.

Both algorithms are based on a reinforcement learning approach – Q-Learning [8] with a ϵ-greedy selection policy [9, 10]. Using this policy, the agent explores the environment with a probability of ϵ, by selecting an action randomly, and exploits its obtained knowledge with probability of $1 - \epsilon$, by selecting an action that has been beneficial in the past. We use the following notation:

- Each job j has a type $t_j = \{r_j, d_j, v_j\}$, where r_j represents the instance of time when the job was "created", d_j the requested duration and v_j its valuation.
- S is a finite discrete set of states, where each state s is defined by a tuple $\{d, v\}$, such that an agent is in state $s = \{d_j, v_j\}$ if it is to bid for a job with duration d_j and valuation v_j.
- A is a repertoire of possible actions, where, in the context of this paper each action a represents the assignment of a specific price to a bid.
- $Q(s, a)$ denotes the expected value of being in state s and taking action a.
- ρ is a mapping from stimuli observed, caused by an action, to the set of real numbers. Here, we use $\rho = -v_j C_j - \pi_j$, where C_j is the time-span between creation and completion of job j, and π_j is the price paid for it.

In other words, the objective is to learn the function $Q(s, a)$, so that, given any job with a specific duration and valuation, a price $\tilde{v} \leq v$ can be selected so that the utility is maximized. However, due to the sizes of the state and action spaces, and the fact that the environment in which the agent operates is continuously changing, only a rough estimate of the Q-function is feasible.

As stated earlier, learning is made through exploration of the environment. After finishing a job, the Q-function, is updated with the new information according to the *Q-Learning update rule*:

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha_t(s_t, a_t)[\rho_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

Here, s_t is the state defined by the duration and valuation of the job that the agent bids for at time t, a_t is the action selected at time t, ρ_t is the received utility of the job. The learning rate $\alpha_t \in [0, 1]$ determines how much weight we give to newly observed rewards. A high value of α_t results in that high importance is given to new information, while a low value leads to that the Q-function is updated using small steps. $\alpha_t = 0$ means no learning at all. The discounting factor γ defines how much expected future rewards affect current decisions. Low $\gamma \to 0$ imply higher attention to immediate rewards. With higher ($\gamma \to 1$) implies orientation on future rewards, where agents may be willing to trade short-term loss for long-term gain.

In Algorithm 1, during exploration, the bid is randomly generated within the interval $p^s \in [s * v_j, v_j]$ with $s \in (0, 1)$. During exploitation, the bid is retrieved from the Q-Table, the "best" bid that in the history achieved the highest average payoff.

In Algorithm 2, during exploration, the strategy selects the "best offer" (maximizing its utility) of a resource provider, for which there is no stored information in the Q-Table. During exploitation it selects the best offer, which maximizes its utility.

Algorithm 2. Q-Strategy: Offer Selection Rule

Require: $economicpreferencesofthejob; provideroffers$

if $\epsilon < Stochastic.random(0,1)$ **then**
  $//Explore:$
  $offer = bestOfferForProviderNotStoredInQTable(job, offers);$
else
  $//Exploit:$
  $offer = myopicBestResponse(state, offers);$
end if

4 Evaluation

Auction and strategy selection are closely connected in the sense that a given choice of strategy should affect the choice of auction, and vice versa. For example some bidding strategies perform well in a Continuous Double Auction (CDA), but not in a Dutch auction. This also implies that the choice of auction to participate in depends on the available strategies. Other factors to take into account in auction selection are the market rules, transaction costs, and the current and average prices in the different auctions.

In this section, we evaluate the Truth-Telling and Q-Learning strategies for two different types of market mechanisms for allocation of computational resources. The first market mechanism is a decentralized on-line machine scheduling mechanism, called Decentralized Local Greedy Mechanism (DLGM) [6]. The second one is a centralized continuous double auction (CDA) [11]. The market mechanisms are implemented within the *SORMA Trading Management* component. The simulation is run on a light version of this component.

As a measure we use the average utility per job received by the consumers. In the Truth-Telling scenario, this is the same as measuring the common wealth for this particular strategy, since all players have the same strategy. In the case of the Q-Learning strategy, however, this is not equivalent, since the behavior of the players diverge as the players observe different information.

In the case of the decentralized DLGM mechanism,, each time a job arrives on the consumer side, her bidding agent generates a bid in form of a job type $t_j = \{r_j, d_j, v_j\}$ (see Section 3) and report this to all providers. Based on this bid, each provider reports back a tentative completion time and tentative price for each of its machines. When sufficiently many provider offers have been collected, the consumer can decide, typically simplistically, which offer to choose. The providers in the DLGM market do not behave strategically and do not request compensation for the use of their resources. The payments are divided only among the consumer agents for compensating the displaced jobs.

In the centralized CDA, consumers and providers submit bids and offers consisting of only a price per time unit, and are matched based on this information. Providers act strategically, trying to achieve as much money as possible for their resources. To calculate the price of their bids, they use a ZIP (Zero Intelligence Plus) agent [12].

In the following section we describe the evaluation settings and simulation results.

4.1 Evaluation Setting

For each market – CDA and DLGM, and each strategy – Q-Learning and Truth-Telling, we simulated four different scenarios described by settings 1 through 4 in Table 1. In each scenario there are 50 consumers and 50 providers (each controlling a single machine). In each of the first three settings, the rate of which jobs comes in on the consumer side is determined by a Poisson process. To increase the competition in the market, we successively increased the mean λ of the Poisson process from λ=.1 (setting 1) to λ=.5 (setting 3). The amount of jobs for these settings is a direct consequence of these values. For these settings, the duration of each job is drawn from the normal distribution with a mean value of 5 hours and a variance s^2 of 3.

The fourth setting is based on the logs of a real workload at the LPC (Laboratoire de Physique Corpusculaire) cluster which is a part of the EGEE Grid environment, and located in Clermont-Ferrand, France [13]. The log contains 244,821 jobs that was sent to the nodes during a period of 10 months starting from August 2004 through May 2005. We have, however, only extracted jobs with a duration between one and 24 hours. The LPC log was chosen because it contains a large variety of jobs with different run-times, numbers of used CPUs, and varying submit and start times.

Table 1. Simulation settings

Setting	Arrival Rate	Duration	# Jobs	# Consumers	# Providers
1	$Poisson(0.1)$	$\max(1, N(5,3))$	751	50	50
2	$Poisson(0.3)$	$\max(1, N(5,3))$	1502	50	50
3	$Poisson(0.5)$	$\max(1, N(5,3))$	3004	50	50
4	As in LPC-Log	As in LPC-Log	105,578	50	50

4.2 Simulation Results

The results of the simulations are summarized in Table 2. Each line in the table represents the evaluation of one strategy for one setting. The first two columns represent the setting used (corresponding to those of Table 1) and the strategy evaluated. The next two columns represent the average utility μ per job achieved as well as the standard deviation σ of job budget and actual payment in the DLGM market, and the last two columns represent the results for the CDA in the same way.

The results show that the *Truth-Telling* strategy achieves the highest utility for all four settings.

The *Q-Strategy* reproduces a "rational behavior" by the generation of the bids. More specifically, we assume that rational agents have an incentive to

Table 2. Simulation results

Setting	Strategy	$DLGM\mu$	$DLGM_\sigma$	CDA_μ	CDA_σ
1	Truth-Telling	$-110,48$	$272,37$	$-7,92 * 10^4$	95,33
1	Q-Strategy	$-174,74$	$257,54$	$-10,33 * 10^4$	93, 56
2	Truth-Telling	$-212,66$	$285,16$	$-11,95 * 10^4$	94,13
2	Q-Strategy	$-392,42$	$265,96$	$-14,63 * 10^4$	93, 30
3	Truth-Telling	$-403,58$	$286,43$	$-7,89 * 10^4$	86,74
3	Q-Strategy	$-901,18$	$265,24$	$-23,22 * 10^4$	90, 77
4	Truth-Telling	-1104	$647,27$	$-9,91 * 10^4$	391,97
4	Q-Strategy	-1172	$580,68$	$-11,04 * 10^4$	313, 69

understate their true price in relation to their valuation. Due to the fact that they understate their true price the achieved utility is lower than by the *Truth-Telling* strategy. The simulation results showed that bidding truthfully in both mechanisms can only increase your utility. Understating the truthful valuation in lower bid results in a poorer "job priority" p_j/d_j by DLGM and this job can be displaced by other jobs which have higher priority. By the CDA mechanism, the price depends on the current demand and supply, bidding a lower price instead of the truth valuation increases the risk of no allocation by the mechanism.

Like in the DLGM market mechanism, the Truth-Telling strategy in the CDA market mechanism achieves higher average utility than the Q-Strategy. However, by the specified CDA market mechanism, the matching is based only on the price without considering the "priority" of a job as with DLGM, and thus achieves very low utility compared to DLGM. The origin of this can be searched in the CDA mechanism itself. First, each agent - provider and consumer - receives all the bids of the other agents as public information and based on this they adapt their bid/offer through the implemented bidding strategy. The CDA-provider agents are also acting strategically and adapting their offered price based on the received public information. Thus, the matching is based on the price resulting from the demand and supply and not on the "job priority" as with DLGM. Secondly, the CDA-provider machine agents do not maintain a priority queue of the submitted job bids and by an allocation the job is immediately submitted and executed on the provider machine. A provider submits an offer as soon as he becomes idle. Thus each time the agents are competing by adapting their job bids based on the used strategy.

Furthermore we investigated the price convergence of the Q-Strategy itself using the real workload data of setting 4 (105.578 jobs). Figure 4 shows six graphs, which represent the time development of the bid for particular classes of jobs, for six selected consumers and six selected jobs. The selection of the consumers and their jobs is based on statistical analysis of the output data, where we selected classes of jobs of different consumers, that have a statistically high number of generated bids. The minimum number of generated bids per job-class is 1, the maximal 49 and the average 12.

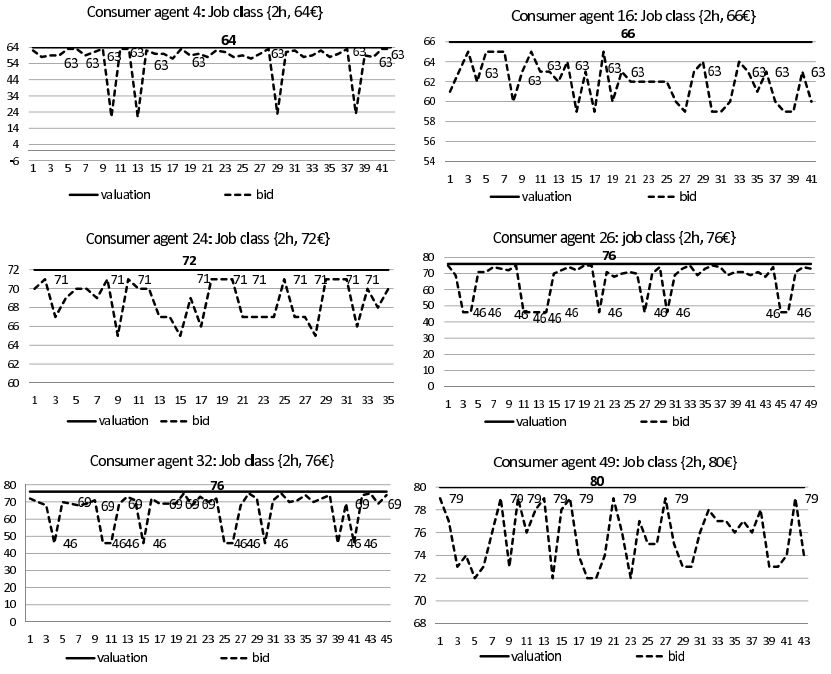

Fig. 4. Convergence of the bids using Q-Strategy

Each graph shows a valuation of a particular job class, the development of the bid over the time and the convergence trend of the bid. An interesting result is that for some cases of jobs with lower "job priority" – p_j/d_j, the bid does not converge to the truth-valuation and for jobs with higher priority, bids of the Q-Strategy converge to the truth-valuation of the specific job type.

5 Related Work

Preference elicitation deals with extraction of user's preferences for different combinations of resource configurations and prices. The aim of this methodology is to find the best choice of configuration, without explicitly presenting all possible choices. Two important approaches for job preference estimation are discussed in the literature – *Conjoint Analysis* and *Analytical Hierarchy Process* [4, 14, 15]. Conjoint analysis estimates the user's value for a certain attribute level by performing regression analysis on the user's feedback to the presented attribute profiles. In contrast to the conjoint analysis method which aims at determining the value of a certain attribute, the analytical hierarchy process tries to determine the relative importance of a certain attribute among a set of attributes [16]. The analytical hierarchy process suffers from the large number of pair-wise comparisons which the user has to perform to generate the matrix from

which the relative weights are computed. With n attributes, the user essentially has to do n-1 comparisons.

The field of autonomous bidding is explored by many researchers. [17] gives an overview of the various agents and their strategies used in the trading agent competition (TAC). The literature described trading agents in stock markets [18], supply chain management [19] and in various market mechanisms [20, 21]. Since agents are self-interested, they aim to implement a strategic behavior in order to maximize their utility. In this context the mechanism design and auction literature investigated various bidding strategies for market-based scheduling [7, 22, 23, 24].

Phelps elaborated co-evolution algorithms to learn the strategy space for autonomous bidding by the allocation of resources in market mechanisms. In his thesis he classified bidding strategies into non-adaptive strategies such as *Truth Telling*, *Equilibrium-Price* and *Zero Intelligence*, and adaptive strategies – *Zero-Intelligence Plus*, *Kaplan's Sniping Strategy*, *Gjerstad-Dickhaut* and *Reinforcement-learning*[9].

The non-adaptive Zero Intelligence ZI [25] agent ignores the state of the market when forming a bid or offer and generates and submits random values drawn from a uniform distribution, where Zero Intelligence Plus ZIP [12] agents maintain a profit margin, a ratio of the trader's profit to its valuation, that determines their bid or offer at any time during the trading process. Furthermore this profit margin adapts to the market conditions using a learning mechanism, so that the agent can submit bids or offers that remain competitive. GD-agents store history information about the submitted bids and use a belief function for the price estimation. FL-agents [21] use fuzzy logic to generate a bid or offer based on a base price, which is a median of previous prices. Risk-Based agents [20] perform prediction of expected utility loss resulting from missing out on a transaction. Kaplan agents [26] define strategic conditions ("juicy offer", "small spread" and "timeout") under which a bid is generated and submitted. [27] introduced a stochastic P-strategy which takes the dynamics and uncertainties of the auction process. A comparison between some bidding strategy is evaluated by [28, 29, 30]. Beside auction strategies, [24] discusses utility functions and strategic negotiations in Grid environments.

The AI literature introduces three main approaches for learning – supervised, unsupervised, and reward-based learning. These methods are distinguished by what kind of feedback the critic provides to the learner. In supervised learning, the critic provides the correct output. In unsupervised learning, no feedback is provided at all. In reward-based learning, the critic provides a quality assessment (the 'reward') of the learner's output. A wide summary of common learning algorithms and decision rules are presented by [11, 29, 31, 32, 33, 34, 35].

6 Conclusions and Outlook

In this paper we have described consumer and provider components supporting the automated bidding process. We introduced the Q-Strategy as novel consumer

bidding strategy, which implements a rational strategic behavior, and evaluated it against the *Truth-Telling* bidding strategy in two different market mechanism.

The Truth-Telling strategy "slightly" outperforms the Q-Strategy in both markets, but nevertheless it offers properties implementing a rational bidding behavior and learning capability. We show that it tends to converge to optimal action values. A common drawback of reinforcement learning algorithms is that they need some time to learn the environment and start to converge to an optimal action. To evaluate the properties of the Q-Strategy we need further research and simulations with different simulation settings and in various mechanisms. Future work will include its evaluation and analysis in further market mechanisms e.g. proportional-share [36, 37, 37] and pay-as-bid [38] as well as a comparison against state-of-the art bidding strategies like *ZIP*, *GD* and *Kaplan* agents.

Moreover, we looked at strategic behavior on consumer side, where truth telling is supposed to be an optimal bidding strategy in the sense of maximizing the consumer's utility. In the next step we are going to introduce strategic behavior on the provider side by extending the DLGM mechanism with payments for the resource usage. In this case the truth telling strategy could be not optimal.

References

1. Byde, A., Salle, M., Bartolini, C.: Market-based resource allocation for utility data centers. HP Lab, Bristol, Technical Report HPL-2003-188 (September 2003)
2. Smith, W., Foster, I., Taylor, V.: Predicting application run times using historical information. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1998, SPDP-WS 1998, and JSSPP 1998. LNCS, vol. 1459. Springer, Heidelberg (1998)
3. Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: Job Submission Description Language (JSDL) Specification, Version 1.0. Job Submission Description Language WG (JSDL-WG) (2005)
4. Stoesser, J., Neumann, D.: A model of preference elicitation for distributed market-based resource allocation. Working paper, University of Karlsruhe (TH) (2008)
5. Borissov, N., Blau, B., Neumann, D.: Semi-automated provisioning and usage of configurable services. In: 16th European Conference on Information Systems (ECIS 2008), Galway, Ireland (2008)
6. Heydenreich, B., Müller, R., Uetz, M.: Decentralization and Mechanism Design for Online Machine Scheduling. METEOR, Maastricht research school of Economics of TEchnology and ORganizations (2006)
7. Phelps, S.: Evolutionary mechanism design. Ph.D. Thesis (July 2007)
8. Watkins, C., Dayan, P.: Q-learning. Machine Learning 8(3), 279–292 (1992)
9. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning: A survey. Arxiv preprint cs.AI/9605103 (1996)
10. Whiteson, S., Stone, P.: On-line evolutionary computation for reinforcement learning in stochastic domains. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 1577–1584 (2006)
11. Tesauro, G., Das, R.: High-performance bidding agents for the continuous double auction. In: Proceedings of the 3rd ACM conference on Electronic Commerce, pp. 206–209 (2001)
12. Cliff, D.: Minimal-intelligence agents for bargaining behaviors in market-based environments. TechnicalReport, Hewlett Packard Labs (1997)

13. Medernach, E., des Cezeaux, C.: Workload analysis of a cluster in a grid environment. In: Feitelson, D.G., Frachtenberg, E., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2005. LNCS, vol. 3834. Springer, Heidelberg (2005)
14. Luce, R., Tukey, J.: Simultaneous conjoint measurement: A new type of fundamental measurement. Journal of Mathematical Psychology 1(1), 1–27 (1964)
15. Green, P., Rao, V.: Conjoint Measurement for Quantifying Judgmental Data. Journal of Marketing Research 8(3), 355–363 (1971)
16. Saaty, T.: Axiomatic foundation of the analytic hierarchy process. Management Science 32(7), 841–855 (1986)
17. Wellman, M., Greenwald, A., Stone, P.: Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition. MIT Press, Cambridge (2007)
18. Sherstov, A., Stone, P.: Three automated stock-trading agents: Acomparative study. In: Agent-mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems: AAMAS 2004 Workshop, AMEC 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers (2006)
19. Stone, P.: Multiagent learning is not the answer. it is the question. Artificial Intelligence (to appear, 2007)
20. Vytelingum, P., Dash, R., David, E., Jennings, N.: A risk-based bidding strategy for continuous double auctions. In: Proc. 16th European Conference on Artificial Intelligence, pp. 79–83 (2004)
21. He, M., Leung, H., Jennings, N.: A fuzzy-logic based bidding strategy for autonomous agents in continuous double auctions. IEEE Transactions on Knowledge and Data Engineering 15(6), 1345–1363 (2003)
22. Reeves, D., Wellman, M., MacKie-Mason, J., Osepayshvili, A.: Exploring bidding strategies for market-based scheduling. Decision Support Systems 39(1), 67–85 (2005)
23. Li, J., Yahyapour, R.: Learning-based negotiation strategies for grid scheduling. In: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006), vol. 00, pp. 576–583 (2006)
24. Li, J., Yahyapour, R.: A strategic negotiation model for grid scheduling. Journal International Transactions on Systems Science and Applications, 411–420 (2006)
25. Gode, D., Sunder, S.: Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. The Journal of Political Economy 101(1), 119–137 (1993)
26. Kaplan, S., Weisbach, M.: The success of acquisitions: Evidence from divestitures. The Journal of Finance 47(1), 107–138 (1992)
27. Park, S., Durfee, E., Birmingham, W.: An adaptive agent bidding strategy based on stochastic modeling. In: Proceedings of the third annual conference on Autonomous Agents, pp. 147–153 (1999)
28. Das, R., Hanson, J., Kephart, J., Tesauro, G.: Agent-human interactions in the continuous double auction. In: Proceedings of the International Joint Conference on Artificial Intelligence, vol. 26 (2001)
29. Sherstov, A., Stone, P.: Three automated stock-trading agents: A comparative study. In: Faratin, P., Rodriguez-Aguilar, J. (eds.) AMEC 2004. LNCS (LNAI), vol. 3435, pp. 173–187. Springer, Heidelberg (2006)
30. Kearns, M., Ortiz, L.: The penn-lehman automated trading project. Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications] 18(6), 22–31 (2003)
31. Stone, P.: Learning and multiagent reasoning for autonomous agents. In: The 20th International Joint Conference on Artificial Intelligence, pp. 13–30 (January 2007)

32. van den Herik, H.J., Hennes, D., Kaisers, M., Tuyls, K., Verbeeck, K.: Multi-agent learning dynamics: A survey. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 36–56. Springer, Heidelberg (2007)
33. Erev, I., Roth, A.: Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. The American Economic Review 88(4), 848–881 (1998)
34. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artificial Intelligence 171(7), 365–377 (2007)
35. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems 11(3), 387–434 (2005)
36. Lai, K., Rasmusson, L., Adar, E., Zhang, L., Huberman, B.: Tycoon: An implementation of a distributed, market-based resource allocation system. Multiagent and Grid Systems 1(3), 169–182 (2005)
37. Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S., Gehrke, J., Plaxton, C.: A proportional share resource allocation algorithm for real-time, time-shared systems. In: Proceedings of the 17th IEEE Real-Time Systems Symposium, pp. 288–299 (1996)
38. Sanghavi, S., Hajek, B.: Optimal allocation of a divisible good to strategic buyers. In: 43rd IEEE Conference on Decision and Control-CDC (2004)