Learning the IPA Market with Individual and Social Rewards

Eduardo Rodrigues Gomes and Ryszard Kowalczyk
Swinburne University of Technology
Faculty of Information and Communication Technology
Hawthorn, 3122 Victoria, Australia
{egomes, rkowalczyk}@ict.swin.edu.au

Abstract

Market-based mechanisms offer a promising approach for distributed allocation of resources without centralized control. One of those mechanisms is the Iterative Price Adjustment (IPA). Under standard assumptions, the IPA uses demand functions that do not allow the agents to have preferences over some attributes of the allocation, e.g. different price or resource levels. One of the alternatives to address this limitation is to describe the agents' preferences using utility functions. In such a scenario, however, there is no unique mapping between the utility functions and a demand function. Gomes & Kowalczyk [10, 9] proposed the use of Reinforcement Learning to let the agents learn the demand functions given the utility functions. Their approach is based on the individual utilities of the agents at the end of the allocation. In this paper, we extend such a work by applying a new reward function, based on the social welfare of the allocation, and by considering more clients in the market. The learning process and the behavior of the agents using both reward functions are investigated through experiments and the results compared.

1. Introduction

The last years have seen the emergence of large distributed computational systems, such as the Grid, as a new approach to solve large-scale problems. One of the main challenges that these systems face is the efficient allocation of resources. Several factors contribute to complicate the problem. Computational and geographical distribution, dynamic architecture, lack of coherent global knowledge and lack of centralized control are just some of them [2, 7]. These features point to the direction that any successful allocation mechanism for these systems should be highly distributed and not rely on centralized control.

Various allocation mechanisms have been proposed to address the problem. Some of them are based on analogies between the system and real economies. Commonly called Market-based, those mechanisms offer a promising approach [3, 23, 24]. In this paper, we consider the Iterative Price Adjustment (IPA) method [6], a market-based mechanism that uses the concept of a *price* that is iteratively adjusted to find equilibrium between a set of demands and a limited supply of resources.

In the IPA, the interests of the agents in the resource allocation are described by means of demand functions. Under standard assumptions, those demand functions do not allow the agents to have preferences over some attributes of the allocation, for example the price. It makes difficult to influence and optimize the allocation quality in terms of the utility received by the agents. Gomes & Kowalczyk [10, 9] addressed the problem by letting the agents to describe their preferences over different attributes using utility functions. The agents then use Reinforcement Learning (RL) to obtain a demand function based on the utility functions. The authors evaluated the approach in two scenarios, both considered a single IPA market with one type of resource and two self-interested client agents. In the first scenario, the agents were trained against static clients (with pre-defined demand functions). Using a reward function based on their own utility functions, these agents presented good individual performance but comparatively lower social performance. The second scenario addressed the improvement of the social performance by training two agents jointly. These agents presented a better social performance, but still used a reward function based on the individual utilities. In this paper, we extend such a work by applying a new reward function, based on the social welfare, and by considering more clients in the market. The learning process and the behavior of the agents using both reward functions in the market are investigated and the results compared.

The paper is structured as follows. Next section presents the background information on the scenario, the IPA method and the RL model. Section 3 discusses the configuration and results of the learning experiments. Section 4 describes the performance of the learning agents in the

market. Section 5 discusses some related works. Finally, section 6 concludes the paper.

2. Background

We address the problem in which a limited amount of resources has to be allocated to a set of self-interested agents through the IPA method. To participate in the allocation process, the agents evaluate the current price of the resources, given by the market, and require the amount of resources that best satisfy their interests in the resource allocation. These interests, instead of being directly described by a demand function, like in the classical approach, are described by utility functions. In such a scenario, however, there is no unique mapping between the utility functions and the demands. Considering price as given and looking only into the utility functions, one could say that it is enough to request the demand that gives maximum utility itself and then the total utility of the agent will be maximized. This strategy, however, is not a good option because it will lead to a construction of a demand function constant in that demand level. Such a constant function may lead to a poor performance of the market, as it may increase the resources' price to an unacceptable level (given scarce supply), and may also generate a deadlock. In addition, considering price as given is not realistic because it assumes that the agent's behavior does not affect the market. The demand requests made by an agent affect the market price, which affects the demand of the others and, therefore, its own demand. This raises the possibility of gains for the agents who adopt a strategy adapted to the market situation. Furthermore, mapping the utility functions into a demand function adapted to the market "by hand" may turn into a very subjective and time-consuming task. The alternative proposed in [10, 9] is to learn a feasible demand function by Reinforcement Learning through interaction with the IPA market.

2.1. Iterative Price Adjustment

The IPA method decomposes the resource allocation optimization problem into smaller and easier sub-problems. Its behavior mimics the law of demand and supply. The process is a cycle that begins with a facilitator (the market) announcing the initial prices for the resources. Based on this price, agents interested in the resources decide on the amount of resources that maximize their private utilities (the sub-problems) and send these values to the facilitator. The facilitator adjusts the prices according to the total demand of the agents and announces the new prices. If the demand exceeds the supply, the price is raised, otherwise it is lowered. The process continues until an equilibrium price is reached, when we say that the market is cleared. In the equilibrium, the total demand equals the supply or the price of the excessive supply is equal to zero. Under some circumstances, the equilibrium price may not exist [11], but that problem is out of scope of this paper.

2.2. Reinforcement Learning

The RL model consists of a discrete set of environment states, a discrete set of agent actions and a set of scalar reinforcement signals. On each step, the agent receives a signal from the environment indicating its state and chooses an action. Once the action is performed, it changes the state of the environment, generating a reinforcement signal. The agent uses this reinforcement signal to evaluate the quality of the decision. The task of the agent is then to maximize (or minimize) some long-run measure of the reinforcement signal. The reward signal has a vital importance in the model. It is the way to communicate to the agent what to do without have to say how to do.

Q-learning [19] is probably the most used algorithm to apply the RL model. In this algorithm the agent maintains a table of *Q*(s,a) values and updates these values as it gather more experience in the environment. The *Q*-values are estimations of the *Q**(s,a) values, which are the sum of the immediate reward obtained by taking action *a* at state *s* and the total discounted expected future rewards obtained by following the optimal policy thereafter. By updating *Q*(s,a), the agent eventually makes it converge to the *Q**(s,a). The optimal policy is then followed by selecting the actions where the *Q**-values are maximum. The algorithm is to perform the following 4 steps indefinitely:

1. Observe the current state *s*, select an action *a* and execute it
2. Observe the new state *s'* and the reward *r(s,a)*
3. Update the Q-table according to:

$$Q(s,a) = Q(s,a) + \alpha(r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

4. Make the new state s' the current state s

In the algorithm, $0 < \alpha < 1$ is the learning rate and $0 < \gamma < 1$ is the discount rate. Considering a stationary environment, if each action is executed in each state an infinite number of times and α is decayed appropriately, the *Q*-values will converge with probability 1 to the optimal ones. The selection of the action in the first step is made based on an action selection mechanism, which allows to harmonize the trade-off between exploration and exploitation: we need to exploit actions known to be good but we also need to explore to avoid being trapped into sub-optimal behaviors. In this work, we use the ϵ-greedy method, which selects a random action with probability ϵ and the greedy, the one that is currently the best, with probability 1-ϵ.

3. Learning the Demand Functions with an Individual and a Social Reward Function

Gomes & Kowalczyk [10, 9] reported on the case of a single IPA Market with one type of resource, e.g. memory, and two client agents. The agents have preferences over price and amount of resources represented by two different utility functions and learn using a reward function based on these utility functions. We extend this case by defining a new reward function, based on the social welfare of the allocation, and by adding a third client to the market. In this section, we describe the learning experiments we have made to evaluate both reward functions in this new scenario.

3.1. Learning Setup

We tried as much as possible to use the same model used in [10, 9]. We start by describing the learning implementation. To apply the *Q*-learning algorithm we first have to define what are the states, actions and rewards. The only information the clients have available during the IPA negotiation process is the current price of the resources, so this information was mapped into the environment states. The agent's actions were mapped to the amounts of resource it can request.

The reward function proposed in [10, 9] is given by the function $U(p, m)$, which is a combination obtained from the product of $U_1(p)$, the utility function for price, and $U_2(m)$, the utility function for amount of resource. These functions are shown in Figure 1. The agent receives the positive reward given by $U(p, m)$ only when the market reaches the equilibrium and is cleared. In all the other situations the agents receives a reward equal to zero. We will refer to this reward function as individual reward function from this point on.

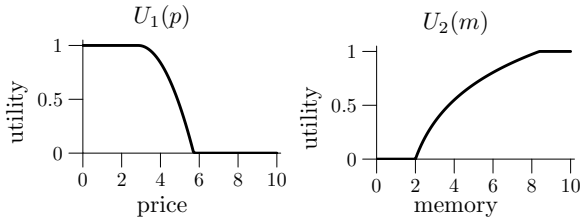

Figure 1. Agents' utility functions.

The new reward function is based on the social welfare function called Nash Product (NP) [3]. The NP is calculated as the product of the total individual utilities of the agents. It is regarded as a good compromise between the Utilitarian Social Welfare (USW), which is given by the sum of the utilities, and the Egalitarian Social Welfare (ESW), given by the utility of the agent which is worst off. The learning agents will receive a reward equal to the NP of the allocation for the states where the market is cleared, and zero for all other states. We will refer to this reward function as social reward function from this point on.

The learning experiments included the use of the same static agents defined in [10, 9]. The demand functions of these agents, shown in Figure 2, were defined "by hand" and based on subjective criteria. They also use the utility functions $U_1(p)$ and $U_2(m)$ to evaluate the quality of the allocation.

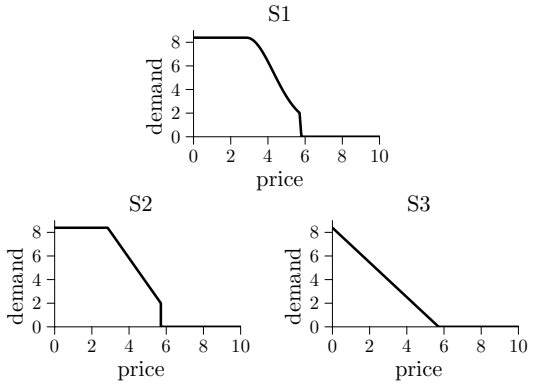

Figure 2. Static agents' demand functions.

Gomes & Kowalczyk [10, 9] used 10 units of resource in their experiments involving 2 clients. We increased this amount to 15 to allow for 3 clients. Neither the old nor the new amount permits for all the agents to have a complete satisfaction in the allocation, but they permit the analysis of the market and the learning under a condition of limited supply.

A series of preliminary experiments had been performed in order to identify a feasible configuration for the values of the parameters used in the learning algorithm. Based on these experiments we set $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.4$. The price of the resources is adjusted by the IPA market using a constant parameter α set to 0.05. The continuity of the states and actions is treated by the application of a rounding procedure. Both, states and actions, are rounded to 1 decimal place.

3.2. Learning Results

We ran learning experiments with both reward functions, iterating the number of learning agents for static agent. The experiments were run 10 times of 500 000 learning episodes each.

The evolution of the demand functions over the episodes has shown to be dependent on the number of learners in the market. In experiments with 1 learning and 2 static agents, it was very consistent, developing constantly towards a well-

defined shape (see Figure 3). There were some small instabilities, but an overall shape was always defined and visible. Experiments with 2 and 3 learning agents presented a less consistent evolution, showing higher instabilities.

These instabilities can have origin in several factors. The most probable is the implementation and application of *Q*-learning made in this work. The *Q*-learning algorithm is only proven to converge if the environment is stationary and if an appropriate decay rule for the learning rate parameter is applied [18, 19]. In the case with one learner, the environment is stationary and the probable cause is the application of a non-appropriate decay rule. With more than one learner, the environment becomes dynamic given the problem of co-adaptation [15, 20], and it is likely to be the cause of the higher instabilities. The co-adaptation occurs when one agent adapts its strategy to the others', and vice-versa, in a cycle that never ends. There are RL algorithms that are more appropriate than *Q*-learning for the multi-agent configuration [17]. They will be considered in our further investigations. Nevertheless, some works have applied *Q*-learning with success also in multi-agent environments [13, 8] and the results obtained in this research with this algorithm are satisfactory.

The learnt demand functions presented similar trends, with higher demands for lower prices, some middle demands in the central area and lower demands for higher prices. However, they differ in shape according to the number of learning agents in the market, the static agent and the reward function. Figure 3 shows examples of demand functions obtained at step 450 000 for both reward functions. Given the space requirements, we only show demand functions for the experiments with S2, but the pattern is the same for S1 and S3. In general, the application of the social reward moves the function slightly to the left. The effects of this in market are commented in section 4. The application of three learners lowers the upper part of the function, allowing a more smooth transition between the upper and the lower parts.

Even though the agents have not achieved exactly the same demand function, the functions obtained are consistent and presented a similar overall trend (see Figure 3).

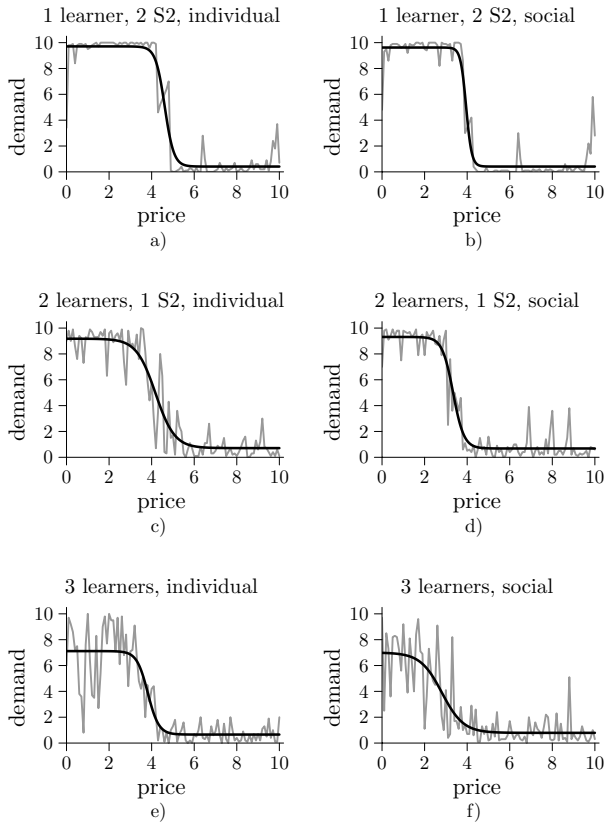

Figure 3. Examples of demand functions obtained by the learning agents, with trend lines.

4. Applying the Learnt Demand Functions

Instead of using the learnt demand functions directly in the ordinary IPA market, we use their trends. It is explained by two reasons [9]. First, to implement the learning algorithm we had to transform prices (states) and demands (actions) into discrete sets. However, in the IPA market, it is better to make use of the continuity of these components. Second, by using the trends we avoid the local instabilities present in the learnt demand functions.

The trends were obtained by a process of curve fitting using the software Mathematica [22]. As in [9], some of the learnt demand functions have shown to be non-monotonic. However, in order to avoid any local minima, it seems more appropriate to apply a monotonic model for the curve fitting. We used the Sigmoidal-Boltzman model, which is described by the equation:

$$y = a + \frac{b-a}{1+e^{-\frac{x-c}{d}}} \quad (1)$$

As in [10, 9] we used the demand functions obtained at step 450 000. Figure 3 shows examples of demand functions and respective trends for experiments with S2, using both reward functions.

4.1. Experiments Results

The resulting demand functions were applied in the ordinary IPA market in the same configurations they were learnt.

We first show the results from a social perspective. Figure 4 shows the comparison of the average NP of the market

when the agents learn using the individual and the social reward functions, by type of experiment. It also shows the NP of the market when it runs only static agents. The first point to note is that, in the case of the individual reward function, the average NP of the market running only static agents is in general better than the average NP of the market running 1 or 2 learning agents. This result is in some sense the same as found by [9], where the NP of the market running agents trained against static agents was, in general, worse than the NP of the market running only static agents. As in [9], the exception is the experiments with S2. It is also interesting to note that the NP of the market running three S3 agents is better even than the NP of the market running only learning agents.

The good social performance of three S3 agents in the market can be explained. The analysis of their demand functions and the market configuration used in the scenario point that the equilibrium price for these agents will be around 2, which is the price area where all S3 agents will request 5 units of resource. For S1 and S2, the equilibrium will be achieved in the price area between 4 and 6. As the price S3 pays for the resources is lower than the price S1 and S2 pay, its total utility is better. From this, we can say a good strategy would be to wait until the price is lower enough and then request the desired demand. However, this strategy only works when the agents have complete information, which it is not realistic. In the case of agents with same demand functions this strategy is adopted implicitly, this explain the case of S3. The implicit coordination may also be the cause of the good social performance achieved when only learning agents are used in the market, shown below.

Figure 4. Average NP of the market using the individual and the social reward functions.

Still in Figure 4, the average NP of the market using the individual reward function increases with the number of learning agents. The application of the social reward function follows the same pattern with the exception of the experiments with S3, where there is actually a decrease of the social welfare. This decrease is also generated in the effects of the good social performance achieved when the market runs more than one S3 agent. In the case of the individual reward function, such a decrease is not seen because the learning agents actually exploit S3, leaving to it few resources and, therefore, decreasing the social welfare with the increase of S3 agents.

The most important aspect shown in Figure 4 is the comparison between the average NP obtained by the market in the experiments involving the two reward functions. The new reward function, based on social welfare, was able to improve the NP of the market in all cases. For experiments with S1 and S2 involving 1 and 2 learning agents, this new reward function was also able to overcome the social welfare obtained by the market running only static agents. For experiments with S3, the new social welfare is almost the same as running only S3 agents.

Having discussed the results in the social level, it is worthwhile to discuss what happen at the individual level. Figure 5 shows the comparison of the average individual utilities of the clients over the experiments. For experiments with 1 learning and 2 static agents, the utilities of the learning agents decreased with the application of the social reward function. At the same time, the utilities of the two static agents increased. For experiments with 2 learning agents and 1 static, when one of the learning agents increased its utility, the other one decreased it, while the static agents always increased theirs. Experiments with 3 learning agents have shown small changes in the utilities. However, it is difficult to say that the changes are characteristics of the experiments, because the three agents act at the same time and without having information about the other agents' moves. These changes will be issue of further investigation, but we believe they are purely casualty.

Figure 6 shows the average individual utilities received by learning and static agents over the experiments with both reward functions. In general, the utilities received by the learning agents slightly decreased with the application of the social reward function. This decrease is followed by a sharply increase in the utilities received by the static agents. Analyzing the demand functions generated by both reward functions (Figure 3), we can note that the application of the social reward causes a small movement of the demand function to the left. This small movement is able to lower the equilibrium price of the market. A lower equilibrium price means more resources to the static agents, increasing their utility in two components (price and amount of resource). In addition, the same small movement increases the learner utility in the price component, compensating a little bit the losses made in the resource component. On the other hand, the experiments with 3 learning agents presented a small increase in the individual utilities. The demand functions obtained by these agents have the same features of the demand

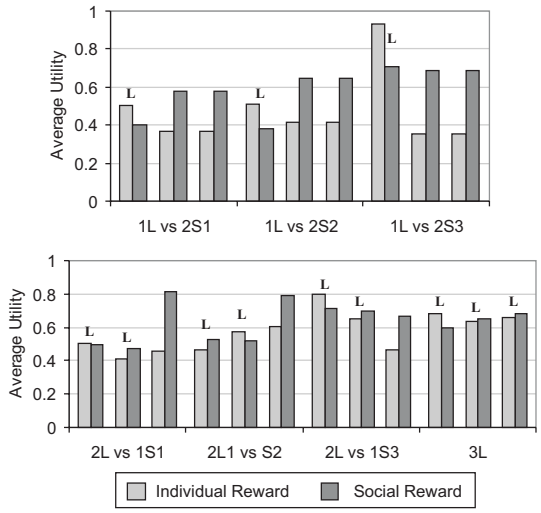

Figure 5. Average individual utilities received by the agents using the individual and the social reward functions.

function of S3 (discussed above): the equilibrium demand is found at a lower price, increasing the total utility in both price and amount of resources for all the agents.

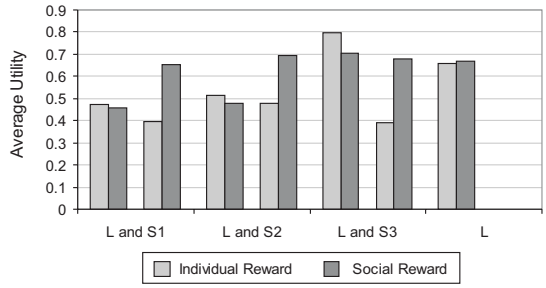

Figure 6. Average individual utilities received by learning and static agents over the experiments.

The experiments have shown that the application of the social reward function was able to give better social results than the application of the individual reward function. This is consistent with the expectations and intuition. In general, the social reward function slightly decreased the individual utilities of the learning agents and sharply increased the utilities of the static agents.

5. Related works

There are few works on machine learning techniques applied to resource allocation. Yeo & Buyya [25] surveyed 35 different market-based systems for resource allocation in large-scale distributed systems. From those, only CATNETS [16] makes use of it. Agents in CATNETS apply an evolutionary-like approach to learn negotiation strategies in a completely decentralized bargaining model. In Preist et al [14], an agent constructs a belief function to model probabilities of closing prices in different auction houses and uses it to coordinate its bids in the houses. In Galstyan et al. [8], agents learn which resource nodes to submit their jobs to, given that they are managed by local schedulers. Abdallah & Lesser [1] proposed a multi-agent reinforcement learning algorithm and applied it in distributed task allocation. Csáji & Monostori [5] modeled a resource allocation problem with precedence constraints as a Markov Decision Process and applied a distributed RL approach to solve it. Kephart & Tesauro [12] investigated the use of Q-Learning by two competitive "pricebots" to set prices of a commodity. As far as we are aware, no work has addressed the problem of learning a demand function based on a set of utility functions.

Regarding the application of utility-aware agents, Chunlin & Layuan [4] developed an algorithm based on utility functions for resource allocation for the Grid. However, that algorithm does not make reference about agents having preferences over the price of the resources, as we make in our work.

6. Conclusions

In this paper, we investigated the use of Reinforcement Learning in a market-based resource allocation mechanism called Iterative Price Adjustment. We extended the work of Gomes & Kowalczyk [10, 9], who applied a reward function based on the individual utilities of the agents to let them learn a feasible mapping between utility functions and a demand function in the IPA market. We proposed a new reward function, based on the social welfare of the allocation, and considered more clients than they have made. Both reward functions have been evaluated through experiments in the extended market. The learning process and the behavior of the agents were investigated and the results compared.

Under a social perspective, the proposed reward function, based on the social welfare, generated better results than the one proposed in Gomes & Kowalczyk [10, 9]. The increased social performance is followed by a small decrease in the individual utilities of the learning agents and a sharply increase in the individual utilities of the static agents. These changes are generated by a small dislocation to the left in the demand function of the learning agents. This dislocation is able to lower the equilibrium price, increasing the utility for price of both learning and static agents. The lower equilibrium price also decreases the amount of resources the learning agent is allocated and

increase the resources allocated to the static agents, increasing the social welfare of the allocation.

In general, the experiments have shown the feasibility of the approach and pointed out that further investigations in this direction are worthwhile. One of the limitations for its application in real systems is the amount of episodes needed to learn the demand functions. In an immediate step, we are going to investigate the quality of the demand functions obtained at different steps of the learning process in a tentative to speed up the process. A needed future work is the extension of the scenario to better reflect a real distributed system, such as the Grid. This extension is likely to involve the use of agents described by multiple utility functions and participating in more than one market at same time.

References

[1] S. Abdallah and V. Lesser. Learning the task allocation game. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, pages 850–857, New York, NY, USA, 2006. ACM Press.

[2] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. In M. Parashar and C. Lee, editors, *Proceedings of the IEEE*, volume 93 of *Special Issue on Grid Computing*, pages 698–714. IEEE Press, New Jersey, USA, Mar 2005.

[3] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. L. tre, N. Maudet, J. Padget, S. Phelps, J. A. R. guez Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[4] L. Chunlin and L. Layuan. Pricing and resource allocation in computational grid with utility functions. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, volume II, pages 175–180, Washington, DC, USA, Apr 2005. IEEE Computer Society.

[5] B. C. Csáji and L. Monostori. Adaptive algorithms in distributed resource allocation. In *Proceedings of the 6th international workshop on emergent synthesis (IWES 06)*, 2006.

[6] H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.

[7] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan-Kaufmann, Sao Mateo, USA, 1999.

[8] A. Galstyan, K. Czajkowski, and K. Lerman. Resource allocation in the grid using reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, volume 3, pages 1314–1315, Washington, DC, USA, 2004. IEEE Computer Society.

[9] E. R. Gomes and R. Kowalczyk. Learning in market-based resource allocation. In *Proceedings of the 6th IEEE International Conference on Computer and Information Science (ICIS 2007)*, page to appear. IEEE Computer Society, 2007.

[10] E. R. Gomes and R. Kowalczyk. Reinforcement learning with utility-aware agents for market-based resource allocation. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, 2007.

[11] P. Jennergren. A price schedules decomposition algorithm for linear programming problems. *Econometrica*, 41(5):965–980, Sep 1973.

[12] J. O. Kephart and G. Tesauro. Pseudo-convergent q-learning by competitive pricebots. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)*, pages 463–470, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[13] J. O. Kephart and G. J. Tesauro. Pseudo-convergent Q-learning by competitive pricebots. In *Proc. 17th International Conf. on Machine Learning*, pages 463–470. Morgan Kaufmann, San Francisco, CA, 2000.

[14] C. Preist, A. Byde, and C. Bartolini. Economic dynamics of agents in multiple auctions. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 545–551, New York, NY, USA, 2001. ACM Press.

[15] T. W. Sandholm and R. H. Crites. On multiagent Q–learning in a semi–competitive domain. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi–Agent Systems*, pages 191–205. Springer Verlag, Berlin, 1996.

[16] B. Schnizler, D. Neumann, D. Veit, M. Reinicke, W. Streitberger, T. Eymann, F. Freitag, I. Chao, and P. Chacin. Catnets - wp 1: Theoretical and computational basis, 2005.

[17] Y. Shoham, R. Powers, and T. Grenager. Multi-agent reinforcement learning: a critical survey., 2003.

[18] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

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

[20] M. Weinberg and J. S. Rosenschein. Best-response multiagent learning in non-stationary environments. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, volume 2, pages 506–513, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[21] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[22] S. Wolfram. *Mathematica: a system for doing mathematics by computer (2nd ed.)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1991.

[23] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. Analyzing market-based resource allocation strategies for the computational grid. *International Journal of High Performance Computing Applications*, 15(10):258–281, Aug 2001.

[24] T. Wu, N. Ye, and D. Zhang. Comparison of distributed methods for resource allocation. *International Journal of Production Research*, 43(3):515–536, 2005.

[25] C. S. Yeo and R. Buyya. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Softw. Pract. Exper.*, 36(13):1381–1419, 2006.