Learning in Market-based Resource Allocation

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. the price of the resources. To address this limitation, we study the case where the agents preferences are described by utility functions. In such a scenario, however, there is no unique mapping between the utility functions and a demand function. If made by hand, this task can be very subjective and time consuming. Thus, we propose and investigate the use of Reinforcement Learning to let the agents learn the best demand functions given their utility functions. The approach is evaluated in two scenarios.

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, 8]. 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, 14, 15]. In this paper we consider the Iterative Price Adjustment (IPA) method, 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. This method was first applied by Everett [6] in 1963 and is probably the first method to use the concept of price for resource allocation. Since then, a series of different pricing mechanisms have been proposed [4, 5, 7, 10, 13].

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 specify a relationship between price and demand, and as such 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. One of the alternatives to address this problem is to let the agents to describe their interests using utility functions instead of demand functions. In such a scenario, however, the mapping between the utility functions and a demand function is not unique. If made by hand, the definition of this mapping is very subjective and may turn into a very complex and timeconsuming task, especially if one considers agents with multiple utility functions. In addition, the resulting agents might not perform well in the real system. Thus, in this paper we propose and investigate the use of Reinforcement Learning (RL) [11] to let the agents learn the best demand functions given their utility functions.

The paper is structured as follows. Next section presents the background information on the IPA method and the RL model. Sections 3 and 4 evaluate the approach in two different scenarios. Section 5 discusses some related works. And, 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 selfinterested 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 described by a demand function like in the classical approach, are described by utility functions. As there is no unique mapping between these utility functions and the demand function, the agents learn a feasible mapping through interactions with the market using RL.

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 [10], 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, what generates 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 [12] 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 indefinitely perform the following 4 steps:

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. It allows us to harmonize the trade-off between exploration and exploitation: we need to exploit actions we know 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 IPA Market

In this section we consider the case of a single IPA Market with one type of resource, e.g. memory, and two client agents. Both agents have preferences over price and amount of resources represented by two different utility functions. But only one of them learns how to map its utility functions to a demand function. The demand function of the other is static and predefined.

The learning task of the learning agent in this case is to find a demand function that can maximize its total utility. To analyze this learning task, let *R* be the total amount of resources in the market, p_t be the price of the resources at time *t*, $d_i(p)$ be the demand function of agent *i* and $u_i(p,d)$ be the utility function of agent *i*. Assume that the sum of the clients' demands for maximum utility is greater than the total amount of resources, what implies that the price will not reach zero at any time during the negotiation, so $\sum d_i(p) > R$ when $\sum u_i(p,d)/n = 1$. Also assume that no demand request can be negative. Any set of demands that is able to equilibrate the system at any time of the negotiation process is then given by $\sum d_i(p_t) = R$. Considering our learning scenario, let $d_1(p)$ be a demand function for the learning agent and $d_2(p)$ be the demand function of the static agent. As $d_2(p)$ is fixed, the demands the learning agent can request and that are able to equilibrate the system for any price *p* are given by $d'_1(p) = R - d_2(p)$. The optimal

utility the learning agent can get from the allocation process is therefore given by the point $Q(p,d) \in d'_1(p)$ that maximizes $u_1(p,d)$. An optimal demand function $d_1*(p)$ for the learning agent will then be any demand function that is able to direct the IPA negotiation process to the point $Q(p,d)$. Therefore, the learning task is to find a demand function $d_1 * (p)$. This task would be easily solved if the learning agent knew $d_2(p)$. However it is not the case. Next subsection presents how we used *Q*-learning to solve it.

3.1. Learning experiments

The first task to apply the *Q*-learning algorithm is to define what are the states, the actions and the 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 agents actions were mapped to the amounts of resource it can require. And, the rewards mapped to a function of the utilities the agent receives at the end of the allocation procedure. The learning task herein is episodic and the final state is reached when the market is cleared. The agent receives a positive reward only when it reaches the final state, making it act towards to the market clearance. In all other states, the agent receives a reward equal to zero. The positive rewards are 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. The multiplication is used to stress the fact that both criteria are important in real applications. The utility functions $U_1(p)$ and $U_2(m)$ used by the agents are presented in Figure 1.

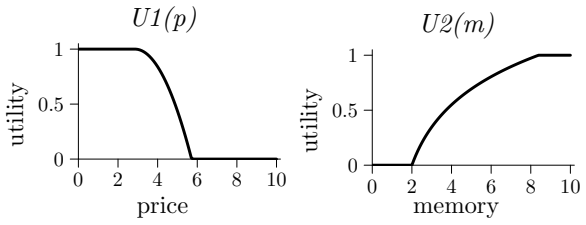

Figure 1. Agents' utility functions.

The learning experiments incorporated the use of three static agents, whose demand functions were defined "'by hand"' based on subjective criteria. They are S1, S2 and S3 and their demand functions were presented in Figure 2. Both, learner and static agents, use the same utility functions to evaluate the quality of the allocation.

The market had a supply of 10 units of memory in all experiments. This amount does not permit for all the agents to have a complete satisfaction but allows us to analyze the behavior of the market and the learning mechanism under a condition of limited supply.

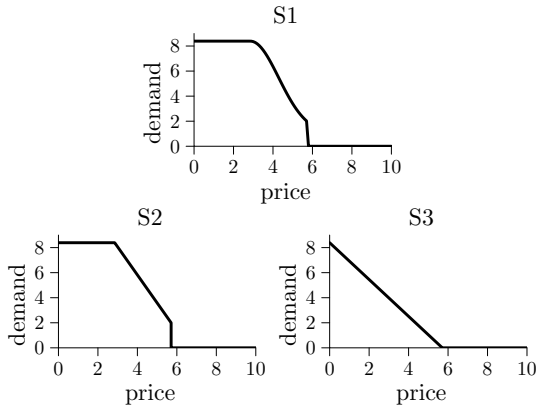

Figure 2. Static Agents demand functions.

We performed a series of preliminary experiments 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 0.1. 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.

We ran 10 learning experiments for each static agent. The experiments were run for a total of 1x10^6 episodes. The evolution of the demand functions over the episodes presented a similar behavior. In the first 50 000 episodes, they are still not so consistent, what can be explained by the fact that the agent has probably not visited all the actions in all the states enough times yet. From this point on, they start to evolve towards a well defined trend. This behavior can be observed in Figure 3, which shows 4 points during the learning process of one of the agents. The evolution is also not completely stable. Although the shape is always there and is quite stable between the same values for demand and price, the individual values presented some variability from one episode to another.

The trend of the learnt demand functions presented high demands for lower prices, some middle demands in the center area and low demands for high prices. This overall trend was expected. The only unexpected point is that in all experiments there was a surprisingly change in the function direction in the central area (see Figure 3 and Figure 4 for some examples). This behavior will be issue of further studies, but is likely to be generated by the fact that, after a sudden drop, the agent tries to direct the negotiation process to a more favorable point.

Even though the agents have not achieved exactly the same demand function, the functions obtained are consistent and presented a very similar overall trend.

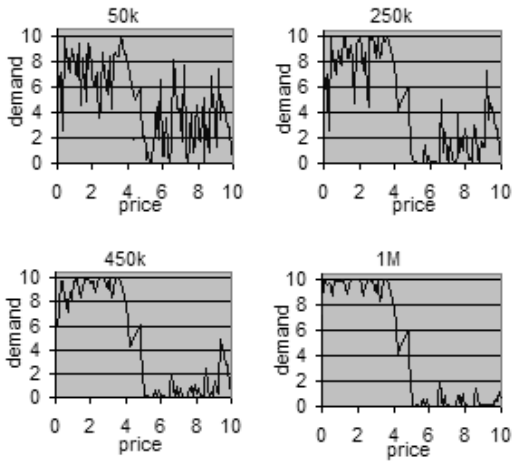

Figure 3. Evolution of the demand function of a learning agent.

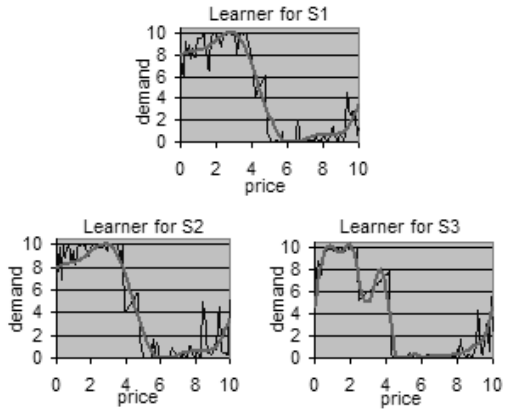

Figure 4. Examples of learning agents' demand functions with trends.

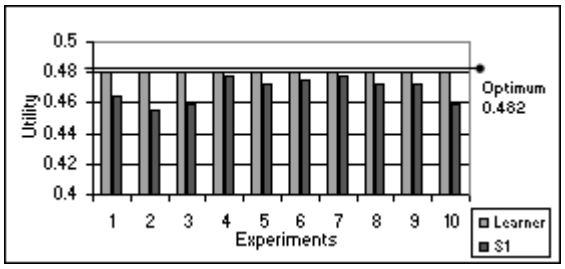

Figure 5. Utilities over experiments with S1.

3.2. Using the learnt demand functions

Instead of using the learnt demand functions directly in the ordinary IPA market, we use their trends. This use is explained by two reasons. First, to implement the learning algorithm we had to transform prices (states) and demands (actions) into discrete sets and in the IPA market it may be better to have these components as continuous ones. And second, we believe that by using the trends we avoid the local instabilities present in the learnt demand function.

The trends were obtained by a process of curve fitting where the best fit curve was selected for each learning agents. We used the demand functions obtained at step $450\text{x}10^3$. The rationale for this is that at this step most of the agents have already developed a demand function with a very consistent shape. Figure 4 shows an example of the trend obtained for a learner of each static agent.

The resulting demand functions were applied in the ordinary IPA market. Figure 5 presents the utilities received by the 10 learning agents over the experiments with S1. It is clearly seen that the utility received by them in all the experiments is very close to the optimum possible. It is also seen a consistency in the learners utilities over the experiments. The same results are observed for the experiments with S2 and S3 shown in Figure 6 and Figure 7, respectively.

It is important to comment that these experiments were not about the learning agents beating the static ones, but about the learning agents getting the most utility they can get based on the utility functions and on the static agents demand functions. This objective was quite well achieved.

4. Improving the social welfare

The previous section presented experiments where the learning agents were only concerned with maximizing their own utilities. However, it may be desirable in some systems to have a mechanism that can also maximize a measure of the Collective Utility, or Social Welfare [3]. For example, it may be desirable to have a fair allocation system, in which all the agents are as happy as the others at the end. This situation is very likely to happen in any system, distributed or not.

In this section we present some experiments in which the objective is to improve the social welfare. We use the same market configuration used in the previous section, a single IPA Market with one type of resource and two client agents. The difference is that both clients are now learners. They are self-interested, but trained jointly and with the same reward functions to encourage social welfare improvement.

4.1. Learning experiments

We ran 4 learning experiments whith $1\text{x}10^6$ episodes each. The learning parameters were set to $\alpha = 0.1$, $\gamma = 0.4$

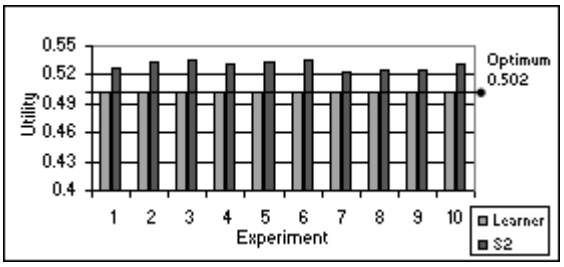

Figure 6. Utilities over experiments with S2.

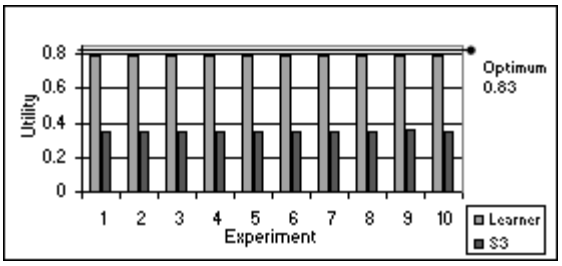

Figure 7. Utilities over experiments with S3.

and $\epsilon = 0.4$, and the market constant set to $\alpha = 0.1$. From these experiments we obtained 8 agents.

The learning results were very similar to the ones btained in the previous experiments. The evolution of the demand functions was also similar over the agents, being not completely consistent in the first 50 000 learning episodes and evolving towards a well defined trend from that point on (Figure 8). There is also the instability in the evolution in which concerns the local points, but this time a little bit bigger. This instability can have its origin in several factors, from the application of *Q*-learning by two simultaneous learners, going through the not appropriate decay rule for the learning rate, to the design choices for the learning implementation. The variability that is seen in the demand function presented in Figure 8 for prices higher than 10 is explained by the fact that in these experiments we allowed the prices to rise indefinitely and prices in that region are not so frequent making the agent not to visit those states often enough to actually learn them.

An interesting point is that the demand functions obtained here did not present a high demand for lower prices as it might be expected from observing the utility functions and is actually seen in the previous experiments (see Figure 3). This behavior is most likely generated from the learning format we used, which incentives the social welfare improvement rather than the individual, even though the agents are self-interested. Another interesting point is the non existence of unexpected changes in the demand function direction in the central area.

As in the previous experiments the agents have not

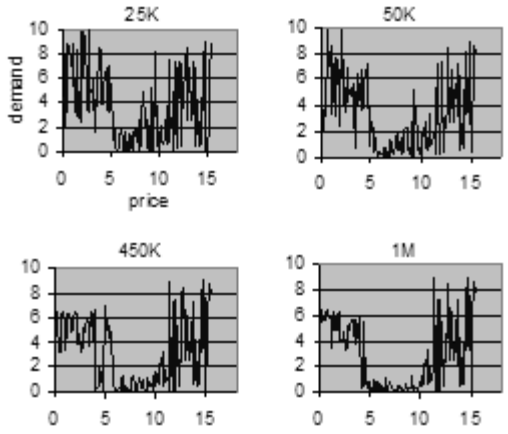

Figure 8. Evolution of the demand function of one of the learning agents.

achieved exactly the same demand function, but the obtained functions are consistent presenting a very similar overall trend.

4.2. Using the learnt demand functions

We used the trends of the learnt demand functions for the same reasons we used them in the previous experiments. We randomly selected 4 of our 8 agents to apply a curve-fitting method. As in the previous experiments we used the demand functions obtained at step $450\text{x}10^3$. The demand functions with respective trends are presented in Figure 9.

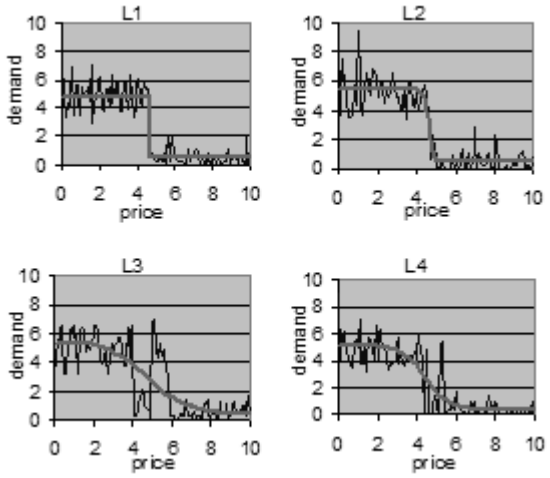

Figure 9. Learning agents demand functions with trends.

The experiments with the resulting demand functions

in the IPA market included runs against themselves and against the three static agents. We also run static versus static.

Figure 10 shows the individual utilities for experiments of type learner vs. learner and static vs. static. In general, the utilities of the learning agents are better than the static ones. There is also some similarity among the utilities received by the learning agents. The situation is little different for the static agents, where one of them, S3, achieved a very poor individual performance.

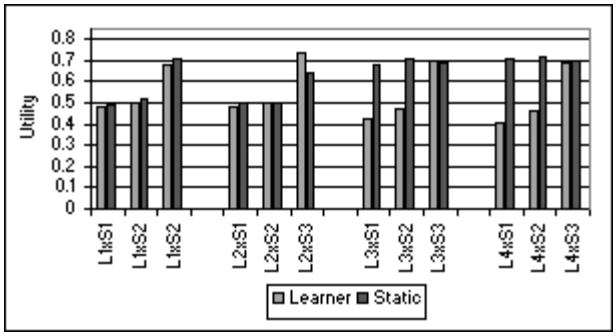

Figure 10. Individual utilities in learner vs. learner and static vs. static experiments.

Experiments of type learner vs. static (Figure 11) revealed that the four learning agents presented similar performances against the same static agents. This was expected as they learn using the same set of utility functions. These experiments also revealed that the performances of the learning agents were good. The utilities received by them are not so far from the optimum they can get. The learners received an average utility of approximately 0.45 for S1, 0.48 for S2 and 0.7 for S3, the optimums are 0.482, 0.502, 0.83 for S1, S2 and S3 respectively.

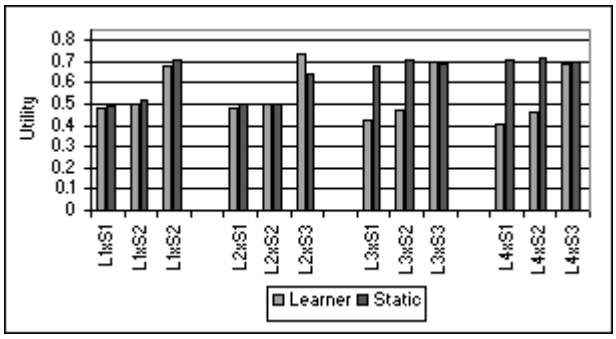

Figure 11. Individual utilities in learner vs. static experiments.

There are several different functions to calculate the Social Welfare (SW) of a system. [3] presents a substantial list of functions and indicates the applications where they may be useful. In this paper we use three of the most significant ones: the Utilitarian Social Welfare (USW), defined as the sum of the individual utilities; the Egalitarian Social Welfare (ESW), given by the utility of the agent which is worst off and regarded as a good measure of the fairness of a system; and the Nash Product (NP), obtained from the product of the individual utilities and viewed as a good compromise between the USW and the ESW.

Figure 12 shows the average SW obtained using these three SW functions for the current scenario in experiments of type learner vs. learner, learner vs. static and static vs. static. It also shows the results for experiments of type learner vs. static from the previous scenario. Figure 12 a) presents the average USW. Figure 12 b) shows the average ESW. And finally, Figure 12 c) presents the average NP.

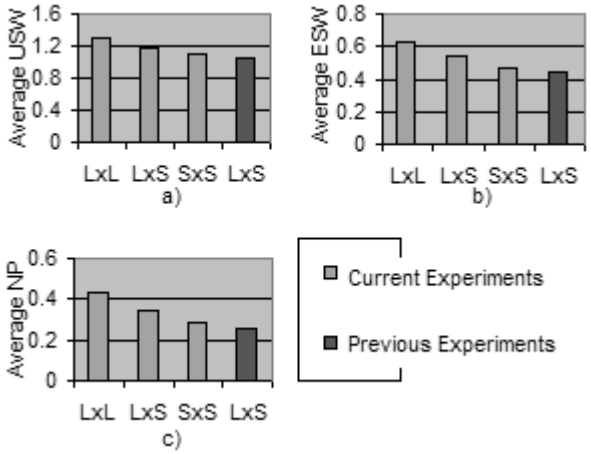

Figure 12. a) Average USW; b) Average ESW; c) Average NP.

The three social welfare functions show that the markets performance is improved using the learning agents from the current experiments. You can note that the social performance is always best when using only them. The use of one of the current learners and one of the static agents is also able to improve the SW. Not surprisingly, the learning agents from the previous experiments presented a poor social performance.

A final analysis about the individual utilities received by the learning agents from the current experiments and from the previous experiments revealed a small decrease in the values. The comparison is presented in Figure 13.

5. Related works

As far as we are aware no work has addressed the problem of learning a demand function based on a set of utility

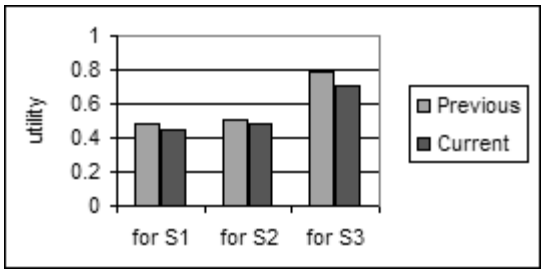

Figure 13. Average individual utilities of learning agents over previous and current experiments.

functions. However, the use of reinforcement learning in resource allocation is not new. Galstyan et al. [9] used reinforcement learning in a scenario where a large number of users submit their jobs to resources that are scheduled by a local scheduler in a grid environment. Abdallah and Lesser [1] proposed a multiagent reinforcement learning algorithm and applied this algorithm in distributed task allocation.

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

6. Conclusions

In this paper we proposed and investigated the use of Reinforcement Learning in a market-based resource allocation mechanism called Iterative Price Adjustment. Under standard assumptions, this mechanism uses demand functions that do not allow the agents to have preferences over some attributes of the allocation, e.g. the price of the resources. To address this limitation, we studied the case where the agents preferences in the resource allocation are described by utility functions and they learn the demand functions given the utility functions.

The approach has been evaluated in two scenarios. Both scenarios considered a single IPA market with one type of resource and two self-interested client agents. In the first scenario one of the agents was a learner and the other had a pre-defined static demand function. The experiments in this scenario have shown that through the application of RL the learning agents were able to learn very good demand functions, presenting good individual performances and receiving utilities very close to the optimum. However, they presented a not so good social performance.

In the second scenario both agents were learners. They were trained jointly and using the same reward functions to encourage them to improve social welfare. Once again the agents were able to learn good demand functions. They performed quite well under both the individual and the social perspective.

Compared with the first scenario, the learning agents of the second scenario achieved a much better social solution, being able to increase the markets social welfare. This increase was followed by a small decrease in the individual performances. Nevertheless, the impact of the increase in the social welfare was bigger than the impact of the decrease in the individual performance, what make us to believe that the approach used in the second scenario is a good one for learning the IPA Market.

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. An immediate step is to investigate how to minimize such limitation. The results of the second scenario may help in this task. They have shown that agents can learn very good demand functions if trained against a mirror. This learning schema is reasonable to be taken in a phase before the real market as it does not use knowledge about other agents. Once in the market, they would adapt the learnt demand functions to the markets specificities, reducing therefore the number of learning episodes in the market. However, it may be necessary to have some knowledge about the market, such as the amount of resources available. Another help may come from the application of curve fitting methods. Once we have obtained the points that best describe the optimal demand function, we would use curve fitting to approximate it. A needed future work is the extension of the scenarios 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*

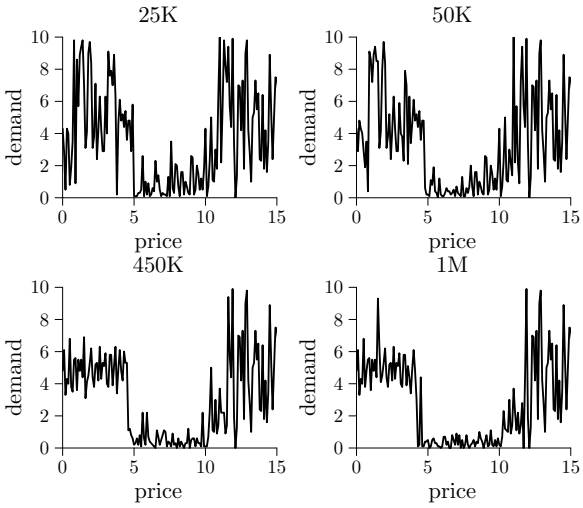

Figure 14. Static Agents demand functions.

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] A. S. Elmaghraby, A. Kumar, M. M. Kantardzic, and M. G. Mostafa. A scalable pricing model for bandwidth allocation. *Electronic Commerce Research*, 5(2):203–227, 2005.

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

[7] M. Feldman, K. Lai, and L. Zhang. A price-anticipating resource allocation mechanism for distributed shared clusters. In *Proceedings of the 6th ACM conference on Electronic commerce (EC'05)*, pages 127–136, New York, NY, USA, 2005. ACM Press.

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

[9] 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.

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

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

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

[13] 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.

[14] 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.

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