# A Continuous Double Auction Method for Resource Allocation in Computational Grids

Hesam Izakian, Behrouz Tork Ladani, Kamran Zamanifar, Ajith Abraham and Václav Snášel

*Abstract*— **In this paper, we introduce a continuous double auction method for grid resource allocation in which resources are considered as provider agents and users as consumer agents. In each time step, each provider agent determines its requested value based on its workload and each consumer agent determines its bid value based on two constraints: the remaining time for bidding, and the remaining resources for bidding. We study this method in terms of economic efficiency and system performance. Experimental results show that the proposed method is better than Earliest Deadline First (EDF) method, which is a default strategy in many schedulers.**

## I. Introduction

COMPUTATIONAL grids are emerging as the promising next generation of computational platforms for executing large-scale resource intensive applications arising in science, engineering, and commerce [1, 19-26]. They support the creation of virtual organizations and enterprises that enable the sharing, exchange, selection, and aggregation of geographically distributed heterogeneous resources. Each user (resource consumer) can use grid resources by running a grid portal such as Globus [15] or Legion [16] on his/her machine and each resource owner can share his/her resources by running a grid portal too. Different users demand different requirements and various resources have different capabilities and availabilities and on the basis of their policies provide access to them. At any moment, different resource owners with different resources and services are added to or removed from the grid. On the other hand, different users with varying requirements can enter the grid. As a result, grid environment is highly dynamic, heterogeneous, and uncontrollable and is distributed across different administrative domains.

The conventional methods for grid resource management cannot be applied simply because they assume complete control over resources and requests. Since the grid resources are distributed in different geographically regions and belong to various administrative domains, using decentralized methods for grid resource management is a suitable solution. An appropriate grid resource management exploits the capability of resources efficiently and satisfies the user's reasonable requests. In recent years, usage of market based methods for grid resource management has received much attention in many studies.

A sustainable market-like computational grid has two characteristics: it must allow resource providers and resource consumers to make autonomous scheduling decisions, and both parties of providers and consumers must have sufficient incentives to stay and play in the market [12]. Two categories of market based models that are used for grid resource management are commodities market models and auction models. In commodity market model, providers specify their resource price and charge users according to the amount of resource they consume. In auction model, each provider and consumer acts independently and they agree privately on the selling price.

Auctions are used for products that have no standard values and the prices are affected by supply and demand at a specific time. Auctions require little global price information, are decentralized, and easy to implement in grid setting [2]. Based on interactions between consumers and providers, auctions can be classified into four basic types: the ascending auction (English auction), the descending auction (Dutch auction), the first-price and second-price sealed auction, and the double auction.

The double auction model has a high potential for grid computing [2]. In a double auction model, consumers submit bids and providers submit requests at any time during the trading period. If at any time there are bids and requests that match or are compatible with a price then a trade is executed immediately.

Three most popular double auctions are: Preston-McAfee Double Auction Protocol (PMDA) [3], Threshold Price Double Auction Protocol (TPDA) [4], and Continuous Double Auction Protocol (CDA). Kant and Grosu [5] showed that CDA protocol is better than both resource's and user's perspective providing high resource utilization in grid environments.

In this paper, we use a continuous double auction method for grid resource allocation. The results illustrate that the proposed method is effective in resource utilization and is an incentive from both resource consumers' and resource providers' perspective. The remainder of this paper is

Hesam Izakian, Islamic Azad University, Ramsar branch, Ramsar, Iran (email: hesam.izakian@gmail.com).

Behrouz Tork Ladani, Department of Computer Engineering, Faculty of Engineering, University of Isfahan, Isfahan, Iran (email: ladani@eng.ui.ac.ir).

Kamran Zamanifar, Department of Computer Engineering, Faculty of Engineering, University of Isfahan, Isfahan, Iran (email: zamanifar@eng.ui.ac.ir).

Ajith Abraham, Machine Intelligence Research Labs –MIR Labs (email: ajith.abraham@ieee.org).

Václav Snášel, Faculty of Electrical Engineering and Computer Science VSB-Technical University of Ostrava, Czech Republic (email: vaclav.snasel@vsb.cz).

organized in the following manner. In Section 2 we investigate the related works and in Section 3 we formulate the problem. In section 4, we introduce the proposed framework followed by experimental results in Section 5 and finally Section 6 concludes this work.

## II. Related Works

Economic based resource management systems have been investigated by several researchers in [2, 5, 6, 7, 8, 9, 10, 11, 12]. Buyya [2] used economic based concepts including commodity market, posted price modeling, contract net models, bargaining modeling, etc. for grid resource allocation. Huang et al. [6] proposed a resource advance reservation through agents participating in multiple sequential auctions. They used cognitive agents that can automatically adapt to the environment, exchange private information, and learn new experiences from their network neighborhoods. Attanasio et al. [7] developed an auction mechanism based on a progressive Lagrangean heuristic and showed that it was able to provide comparable efficiency to centralized heuristics. Reddy et al. [11] developed a sealed bid method for optimizing the time and budget in grid environments. Xiao et al. [12] present an incentive-based scheduling scheme which utilizes a peer-to-peer decentralized scheduling framework to maximize the success rate of job execution and to minimize fairness deviation among resources. Anthony et al. [13] developed a heuristic decision-making framework through which an autonomous agent can exploit to tackle the problem of bidding across multiple auctions with varying start and end times and with varying protocols including English, Dutch and Vickrey auctions. He et al. [14] introduced a bidding strategy for obtaining goods in multiple overlapping English auctions. They used fuzzy sets to express trade-offs between goods and exploited neuro-fuzzy techniques to predict the expected closing prices of the auctions and to adapt the agent's bidding strategy.

## III. Problem Formulation

The entities in our grid environment are users (resource consumers) and resource owners. Users have one or more independent computational-intensive jobs for execution and are willing to pay for it. Also resource owners have computational resources and are willing to rent them for profit. We use resource consumer agents that work on behalf of the users and resource provider agents that work on behalf of resource owners. The consumer agents and provider agents are two intelligent entities having their own specific objectives. They interact with each other in form of a double auction protocol for obtaining their objectives.

We assume that the grid consists of $m$ resources $R = \{R_1, R_2, ..., R_m\}$ each represented by a provider agent and a set of $k$ users $S = \{S_1, S_2, ..., S_k\}$ each represented by a consumer agent. Each consumer has one or more independent jobs and we assume that in time period $T$, consumers altogether have $n$ jobs $J = \{J_1, J_2, ..., J_n\}$. Each job $J_i$ can be submitted at various times $t_i$ in which $0 \le t_i \le T$. Job $J_i$ is characterized by a three-tuple $J_i = (l_i, b_i, d_i)$, in which $l_i$ is the length of the $i$th job and is specified by millions of instructions (MI), $b_i$ represents the budget allocated to $J_i$. The cost of execution job must not exceed its allocated budget. In other words, the maximum amount that $J_i$ can pay per MI is $\varphi_i = b_i / l_i$. Also $d_i$ determines job deadline by which the consumer desires the job to be finished. Each consumer agent aims at executing its jobs within its corresponding deadlines and minimizing the cost.

Each resource can be characterized by four-tuple $R_j = (c_j, st_j, r_j, mp_j)$, in which $c_j$ is the computational speed of resource $R_j$ which is expressed in terms of millions of instruction the resource can process in one second (MIPS). $st_j$ is the workload of $R_j$ and means that $st_j$ seconds are required for executing the jobs that already are accepted by it. In other words, it is the start time for execution of new accepted job. $r_j$ refers to the lowest price for providing a service by $R_j$ (also called reserve price) and is expressed in form of Grid units per MIPS (G$/MIPS). Also $mp_j$ is the maximum price for using $R_j$ and is expressed in form of G$/MIPS. $mp_j$ can be set by the resource owner or can be set by provider agents through collaborating with other providers. In this paper we assume that each resource can execute one job at a time and resources use First Come Fist Served (FCFS) method for executing the accepted jobs. Figure 1 illustrates the scheduling scheme in the proposed method.

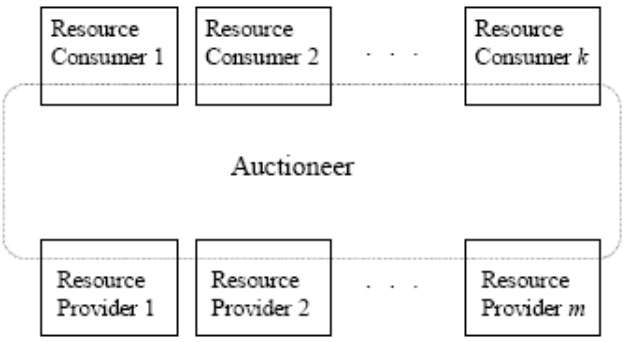


Fig. 1. Scheduling scheme in the proposed framework

## IV. The Continuous Double Auction Model

As mentioned in the previous Section, consumer agents aim at executing jobs within their corresponding deadlines and with the minimum cost. Also the allocated budget for each job determines the maximum cost that a user is willing to pay for executing it. On the other hand, the provider

agents aim at obtaining more profit. For this purpose, they try to sell their resources at higher prices and compete with each other for accepting more jobs. In continuous double auction method at each time unit, consumers and providers submit bids and requests to the auctioneer in the form of G$/MIPS. An auctioneer maintains a list of the current bids and requests and matches the two offers when the highest bid is higher than or equal to the lowest request. The trade occurs at the average of matching request and bid prices. Determining the bid and request value by consumers and providers can be done autonomously and based on their objectives. In this paper, we propose two decision making methods for determining bid values by consumers and request values by providers.

### A. *Determining Bid Values for Consumer Agents*

The consumer agent determines a bid value in each time unit based on two parameters: average remaining time for bidding and remaining resources for biding. This method is inspired by the work presented in [13]. The consumer agent based on these parameters and its corresponding job decides a bid value autonomously.

*1) Determining Bid Value Based on the Number of Remaining Resources for Bid*: in this method, in each time unit, the consumer agent determines a bid value based on the number of remaining resources that can bid for them. A job can bid for a resource if the resource can perform the job within its deadline and the reserve price of the resource (in form of G$/MIPS) is less than or equal to the maximum value the job can pay per MI. Formally $J_i$ can bid for $R_j$ if

$$d_i - st_j - \frac{l_i}{c_j} \geq 0 \quad \text{and,} \quad \varphi_i \geq r_j \tag{1}$$

in which $l_i / c_j$ is the execution time of $J_i$ on $R_j$. At the time of submitting the job, the number of remaining resources has the maximum amount (called $N_i^{\max}$) and with elapsing the time, the number of remained resources is decreased (because of accepting new jobs by resources). In each time unit, the bid value for $J_i$ based on remaining resources can be determined using (2).

$$bid^{i,t}_{\substack{remaining\_\\resources}} = \left[ \frac{r_{\min}}{\varphi_i} + \left(1 - \frac{r_{\min}}{\varphi_i}\right)\left(1 - \frac{N_i^t - 1}{N_i^{\max}}\right)^{\frac{1}{\alpha}} \right] \times \varphi_i \tag{2}$$

*where*, $0.01 \leq \alpha \leq 100$

In this equation $r_{\min}$ is the minimum reserve price among remaining resources and $N_i^t$ is the number of remaining resources at time $t$ for $J_i$. Using the polynomial function defined in (2), different shapes of curve can be obtained by varying the value of $\alpha$. When $\alpha < 1$, the consumer maintains a low bid value until the number of the remaining resources gets close to zero. On the other hand, when $\alpha > 1$ the consumer starts with a bid value close to $\varphi_i$, the maximum bid value per MI. Figure 2 depicts the different convexity degrees of the curves with $\alpha = .2, 1, 5$. In this method when the number of remaining resources is decreased, the bid value is increased.

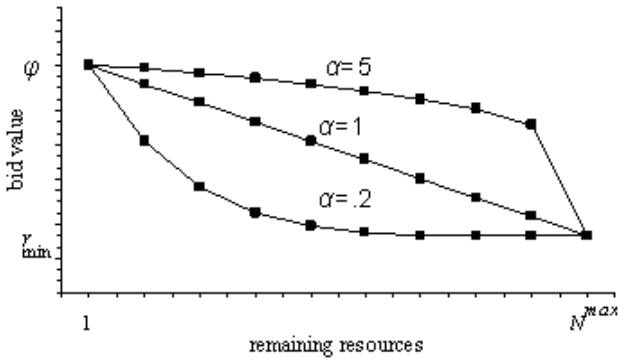


Fig. 2. Bid value for remaining resources

*2) Determining Bid Value Based on the Mean Remaining Time for Bid*: in this method in each time unit, each consumer agent determines a bid value based on the mean remaining time for bidding to the resources. Assume that $J_i$ is submitted at $t$ and $0 \leq t \leq T$. The remaining time that the consumer corresponding $J_i$ can bid to the resource $R_j$ can be determined using (3).

$$rt^t{}_{i,j} = \left( d_i - st_j - \frac{l_i}{c_j} \right) \times F \quad where,\ F = \begin{cases} 1 & if\ \varphi_i \geq r_j \\ 0 & else \end{cases} \tag{3}$$

Also $rt^t{}_{i,j} < 0$ means that resource $R_j$ can not perform $J_i$ within its deadline. The mean remaining times for biding can be obtained using (4).

$$rt^t{}_i = \frac{\sum_{j=1}^{m} \left( rt^t{}_{i,j} \times y_{i,j} \right)}{N_i^{\max}} \quad where,\ y_{i,j} = \begin{cases} 1 & if\ rt^t{}_{i,j} > 0 \\ 0 & else \end{cases} \tag{4}$$

At the time of submitting the job, the mean remaining time for biding has the maximum amount (called $rt_i^{\max}$). The bid values based on the mean remaining time for bid can be obtained using (5).

$$bid^{i,t}_{\substack{remaining\_\\time}} = \left[ \frac{r_{\min}}{\varphi_i} + \left(1 - \frac{r_{\min}}{\varphi_i}\right)\left(1 - \frac{rt_i^{\,t}}{rt_i^{\max}}\right)^{\frac{1}{\beta}} \right] \times \varphi_i \tag{5}$$

*where*, $0.01 \leq \beta \leq 100$

In (5) the parameter $\beta$ is similar to $\alpha$ in (2) and is used for controlling convexity degrees of the curve.

*3) Calculating the final bid value*: after determining the bid values for each of the constraints mentioned above, the consumer agent combines them for calculating the final bid amount. This is obtained using (6).

$$bid^{i,t} = \lambda \times bid^{i,t}_{\substack{remaining\_\\resources}} + (1 - \lambda) \times bid^{i,t}_{\substack{remaining\_\\time}} \tag{6}$$

*where*, $0 \leq \lambda \leq 1$

$\lambda$ in (6) is used to regulate the effectiveness of parameters used in this equation. $\lambda = 1$ means that only the remaining resources constraint is considered in final bid value and $\lambda = 0$ means that only the remaining time constraint is considered. Also $\lambda \in (0,1)$ means that both parameters are taken into account.

### B. Determining the Request Value for Provider Agents

The provider agent aims at obtaining more profit. For this purpose, it tries to sell its resource at a higher price and compete with other providers for accepting more jobs. In this paper, the provider agent uses a method similar to Dutch auction method. We assume that at the moment of joining the grid, the workload of resource is zero and the provider sets the price to reserve price for accepting a job. After accepting a job it updates its workload (start time for a new job) and sets its price to the maximum price, *mp*. Gradually, the workload of the resource is decreased and gets close to zero. By decreasing the workload, the provider decreases its resource price and in the case the workload is equal to zero, it sets the price to the reserve price. Alternatively by accepting each job, the provider agent sets its resource price to maximum price. The maximum price can be determined by the resource owner or the provider through collaborating with other providers. The provider determines its resource price using (7) and requests it from the auctioneer.

$$request_j^t = \left[ \frac{r_j}{mp_j} + \left(1 - \frac{r_j}{mp_j}\right)\left(\frac{st_j^t}{wl_j^t}\right)^{\frac{1}{\delta}} \right] \times mp_j \qquad (7)$$

In (7) $request_j^t$ is the request value of the *j*th provider at time *t*, $wl_j^t$ is the workload of resource $R_j$ after the last allocation, and $st_j^t$ is the current workload or start time of the new job if it is accepted at time *t*. Also $\delta$ is the same as $\alpha$ and $\beta$ in Equations 2 and 5 respectively and is used for controlling convexity degrees of the curve. Figure depicts the different convexity degrees of the curves with $\delta = .2, 1, 5$.

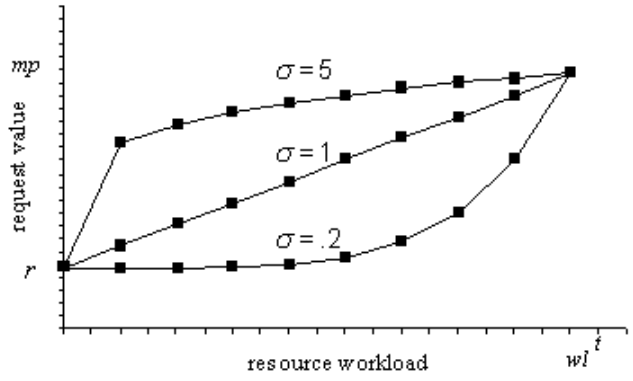


Fig. 3. Determining request value based on workload

### C. Auctioneer Role

In each time unit, consumer agents and provider agents determine their bid and request values and send them to the auctioneer. The auctioneer sorts the bid values in increasing order and request values in decreasing order. If the highest bid is more than or equal to the lowest request, then the trade occurs at the following price:

$$price = \frac{1}{2}(highest\ bid + lowest\ request) \qquad (8)$$

## V. Simulation and Experimental Results

In order to study the efficiency of the method presented in this paper, we developed a computational grid simulator using java agent development framework (JADE) [17]. JADE is a middleware aimed at developing multiagent systems and applications conforming to FIPA standards for intelligent agents. In our simulated system, consumers and providers modeled as two kinds of agents. In our experiments, we set $\alpha = \beta = 5, \delta = .1$ and $\lambda = .6$. Also we assume that there are three types of resources with the specifications presented in Table 1.

Table 1. Resource types and their specifications

| Resource type | Computational capacity (MIPS) | Reserve price (G$/MIPS) | Maximum price (G$/MIPS) |
|---|---|---|---|
| I | 10 | 2 | 6 |
| II | 20 | 2 | 6 |
| III | 30 | 2 | 6 |

Moreover, we assume that there are 500 jobs submitted to the grid one by one. Each job is submitted in random time intervals within the range [1, 20] after its previous job. The length of each job is considered as a random number within the range [1000, 10000] MI sampled from a uniform distribution. After determining the length of each job $J_i$, the deadline is set according to the following expression:

$$d_i = rand\left(\frac{l_i}{10}, 1000\right) + t_i \qquad (9)$$

in which $t_i$ is the time of submitting the job $J_i$. Also the budget allocated to each job $J_i$ is set according the following expression.

$$b_i = rand(2, 6) \times l_i \qquad (10)$$

After submitting each job, a consumer agent is activated and tries to allocate that job by bidding for resources in each time step and after allocation of the job, it is deleted.

### A. Experiment 1

For investigating the efficiency of the presented method, we compare it with the earliest deadline first (EDF) algorithm that is a centralized and the default policy for many scheduling systems. For implementing EDF algorithm, we assume that at the first time step of simulation we know the deadline and the time of submitting each job to the grid. The scheduler sorts the jobs based on their deadline in

descending order. Then in each step, the scheduler selects a job with the earliest deadline and allocates it to a resource which can perform it within its deadline. If there are no resources for performing the job within its deadline the scheduler deletes that job. Also if there are multiple resources able to perform the job within its deadline, the scheduler can use a number of resource selection policies for example, selecting the resource which can perform the job closest to its deadline (also called *EDF_CTD*), or the resource with the minimum completion time (also called *EDF_MCT*), or random selection (also called *EDF_RND*). The effect of these policies and our proposed method will be evaluated in this experiment.

We conduct four case studies for comparison. In the first case study, we assume there is one resource of each type shown in Table 1 (altogether three resources). In the second we assume there are two resources of each type (altogether six resources). Also in third and fourth case studies, we assume there are three and four resources of each type respectively. Results are obtained as an average of ten simulations.

Three criteria, *successful execution rate*, *utilization rate* and *fairness deviation* [12], are used for the comparison.

*1) Successful Execution Rate*: successful job execution means that a job is performed within its deadline. The successful execution rate can be obtained using Eq. (11).

$$\theta = \frac{\sum_{i=1}^{n} \rho_i}{n} \quad where, \quad \rho_i = \begin{cases} 1 & if\ T_C^i \le d_i \\ 0 & else \end{cases} \qquad (11)$$

Here $T_C^i$ denotes the completion time of job $J_i$. As shown in Figure 4, the proposed method outperforms EDF methods in case studies 2, 3, and 4.

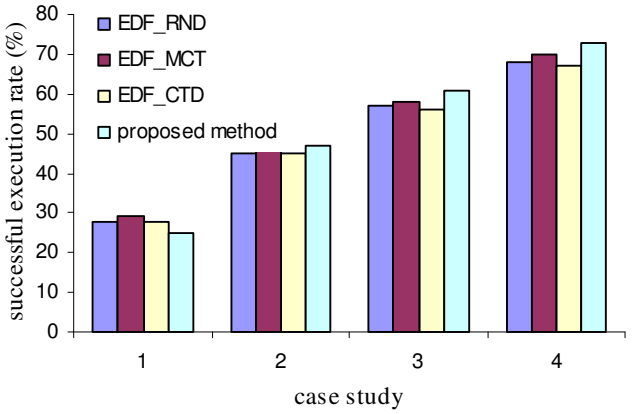


Fig. 4. Comparison of successful execution rate between our method and EDF methods

*2) Resource utilization rate*: the resource utilization rate defined as the percentage of time a resource is busy executing jobs and can be obtained using (12).

$$u_j = \frac{\sum_{i=1}^{n} (te_i - ts_i) \times \gamma_{ij}}{T} \quad where, \quad \gamma_{ij} = \begin{cases} 1 & if\ J_i\ allocated\ to\ R_j \\ 0 & else \end{cases} \qquad (12)$$

In (12) $te_i$ and $ts_i$ are the completion time and start time of job $J_i$ respectively. $T$ in this equation is the total simulation time. Figures 5 and 6 illustrate the comparison between the proposed method and EDF methods for case studies 2 and 4 respectively. As shown in these Figures, our method has more resource utilization rate for most resources.

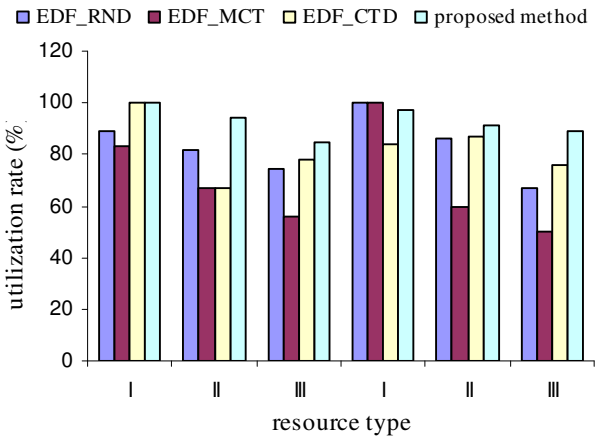


Fig. 5. Comparison of resource utilization between our method and EDF methods in case study 2

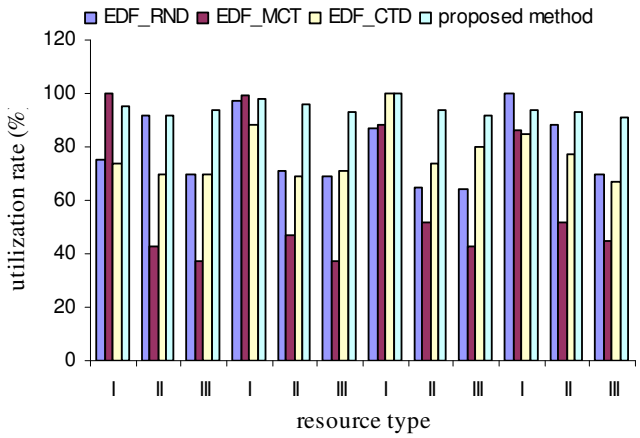


Fig. 6. Comparison of resource utilization between our method and EDF methods in case study 4

*3) Fairness deviation*: the fairness of the market means that each resource owner has an equal opportunity to offer its resource and it can obtain a fair profit according to its capability [12]. The fairness of the market is an intensive metric for resource owners to stay in grid and play role. A resource allocation scheme is fair if fairness deviation of resources is minimized. The fairness deviation of the grid system can be obtained using (13)

$$\sigma = std\_dev(\Delta_1,...,\Delta_m) \ \ where, \ \ \Delta_j = \frac{p_j \Big/ \sum_{l=1}^{m} p_l}{c_j \Big/ \sum_{l=1}^{m} c_l} \qquad (13)$$

In (13), $p_j$ denotes the profit of resource $R_j$. Figure 7 shows a comparison of fairness deviation between our method and other methods in the four case studies. As shown in Figure 7, in the proposed method, by increasing the number of resources, the fairness deviation is decreased.

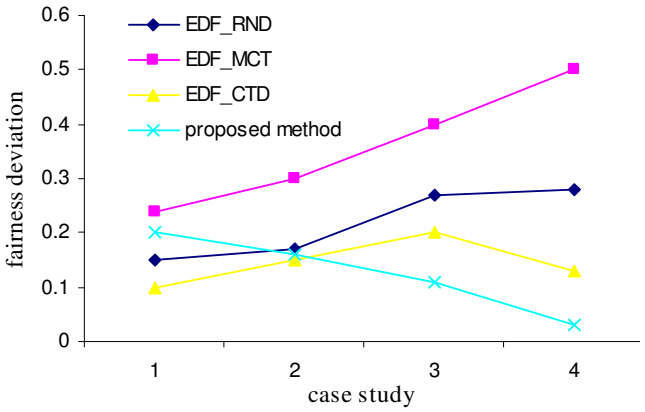


Fig. 7. Comparison fairness deviation between our method and EDF methods

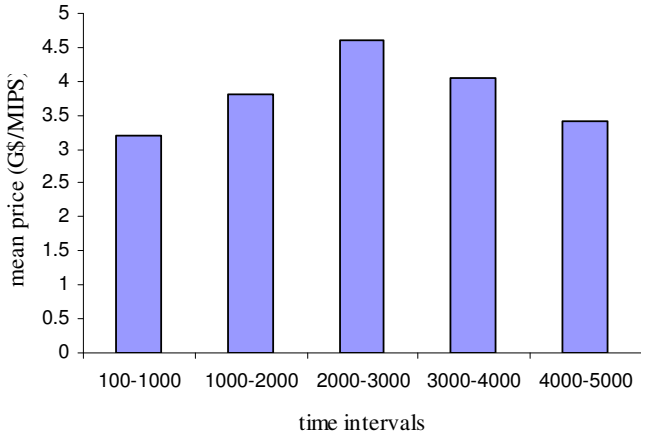


Fig. 8. Mean price in different demand within various time intervals

### B. *Experiment 2*

In this experiment, we show that the proposed framework supports the *supply and demand* model [18]. In a supply and demand model, prices change very often based on supply and demand changes. Basically, when the demand increases or supply decreases, prices are increased and when the supply increases or demand decreases, prices are decreased accordingly. We investigate the changes of prices with different number of demands in various time intervals. Here we conduct case study 4 in Experiment 1 with 50 demands within deadlines between [100, 1000] time intervals, 100 demands within deadlines between [1000, 2000] time intervals, 200 demands within deadlines between [2000, 3000] time intervals, 100 demands within deadlines between [3000, 4000] time intervals, and 50 demands within deadlines between [4000, 5000] time intervals. Figure 8 shows the mean of resource prices in different time intervals. As shown in Figure 8, when the demand increases, prices are increased and when the demand decreases prices are decreased.

## VI. CONCLUSIONS

A computational grid must allow resource owners and resource consumers to make autonomous scheduling decisions, and both parties must have sufficient incentives to stay and play in the grid. In this paper, we proposed a continuous double auction method for grid job scheduling. We developed two agent types: provider agents and consumer agents that are responsible for autonomous decision making on behalf of the resource owners and the resource consumers respectively. Each provider agent determines its bid value based on its workload and each consumer agent determines its bid value based on two constraints remaining time for bidding and remaining resources for bidding. Experimental results clearly illustrate that the proposed method is efficient and intensive for both resource owners and resource consumers.

## REFERENCES


[1] I. Foster, C. Kesselman (Eds.), "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers, SanFrancisco, USA, 1999.

[2] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing", The Journal of Concurrency and Computation, Volume 14, Issue 13-15, pp. 1507 – 1542, Wiley Press, USA, 2002.

[3] R. P. McAfee, "A dominant strategy double auction", Journal of Economic Theory, 56:434–450, 1992.

[4] M. Yokoo, Y. Sakurai, and S. Matsubara, "Robust double auction protocol against false-name bids", In Proc. of the 21st IEEE International Conference on Distributed Computing Systems, pages 137–145, April 2001.

[5] U. Kant, D. Grosu, "Auction-Based Resource Allocation Protocols in Grids", In 16th International Conference on Parallel and Distributed Computing and Systems, Nov. 2004, pp. 20–27.

[6] Z. Huang, Y. Qiu, "Resource trading using cognitive agents: A hybrid perspective and its simulation", Future Generation Computer Systems 23 (2007) 837–845.

[7] A. Attanasio, G. Ghiani, L. Grandinetti, and F. Guerriero, "Auction algorithms for decentralized parallel machine scheduling", Parallel Computing 32 (2006) 701–709.

[8] K. M. Chao, R. Anane, J. H. Chen, and R. Gatward, "Negotiating agents in a market-oriented grid", In Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 406–407, May 2002.

[9] P. Huang, H. Peng, Piyuan Lin, and X. Li, "Macroeconomics Based Grid Resource Allocation", Future Generation Computer Systems 24 (2008) 694–700.

[10] J. Gomoluch and M. Schroeder, "Market-based resource allocation for grid computing: A model and simulation", In Proc. of the 1st International Workshop on Middleware for Grid Computing, pages 211–218, June 2003.

[11] S. R. Reddy, A. Gupta, "Auction Based Resource Allocation in Grids", Springer-Verlag Berlin Heidelberg 2006.

[12] L. Xiao, Y. Zhu, L. M. Ni, and Z. Xu, "Incentive-Based Scheduling for Market-Like Computational Grids", IEEE Transactions on parallel and distributed computing, Vol. 19, No. 7, pp. 903-913, 2008.

[13] P. Anthony, N. R. Jennings, "Developing a Bidding Agent for Multiple Heterogeneous Auctions", ACM Transactions on Internet Technology, Vol. 2, No. 3, pp. 185 – 217, 2003.

[14] M. He, N.R. Jennings, A. Prugel-Bennett, "An adaptive bidding agent for multiple English auctions: A neuro-fuzzy approach", In: Proceedings of IEEE Conference on Fuzzy Systems, Hungary, 2004, pp. 1519–1524.

[15] I. Foster, C. Kesselman, "Globus: A meta-computing infrastructure toolkit", International Journal of Supercomputer Applications, 11(2):115-128, 1997.

[16] A.S. Grimshaw, W.A. Wulf, and the Legion Team, "The legion vision of a worldwide virtual computer", Communications of the ACM, 40(1), pp. 39 – 45, 1997.

[17] JADE. (2004). Java Agent Development Framework. http://jade.cselt.it.

[18] L. W. McKnight and J. Boroumand, "Pricing Internet Services: Approaches and Challenges", IEEE Computer, Vol. 33, No. 2, pp. 128-129, IEEE CS Press, USA, Feb. 2000.

[19] A. Abraham, R. Buyya and B. Nath, "Nature's Heuristics for Scheduling Jobs in Computational Grids", In Proceedings of 8th IEEE International Conference on Advanced Computing and Communications, (ADCOM2000), Sinha P.S. and Gupta R. (Editors), ISBN 0070435480, Tata McGraw-Hill Publishing Co. Ltd, New Delhi, India, pp. 45-52, 2000.

[20] A. Abraham, H.B. Liu, W. Zhang and T.G. Chang, "Job Scheduling on Computational Grids Using Fuzzy Particle Swarm Algorithm", 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, Springer Verlag, Germany, Lecture Notes in Artificial Intelligence, B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): Part II, Lecture Notes on Artificial Intelligence 4252, pp. 500-507, 2006.

[21] A. Abraham, H.B. Liu, C. Grosan and F. Xhafa, "Nature Inspired Metaheuristics for Grid Scheduling: Single and Multiobjective Optimization Approaches", Metaheuristics for Scheduling: Distributed Computing Environments, Studies in Computational Intelligence, Springer Verlag, Germany, ISBN: 978-3-540-69260-7, pp. 247-272, 2008.

[22] H.B. Liu, A. Abraham and F. Xhafa, "Peer-to-Peer Neighbor Selection Using Single and Multiobjective Population-based Metaheuristics", Metaheuristics for Scheduling: Distributed Computing Environments, Studies in Computational Intelligence, Springer Verlag, Germany, ISBN: 978-3-540-69260-7, pp. 323-340, 2008.

[23] F. Xhafa, E. Alba, B. Dorronsoro, B. Duran and A. Abraham, "Efficient Batch Job Scheduling in Grids Using Cellular Memetic Algorithms", Studies in Computational Intelligence, Springer Verlag, Germany, ISBN: 978-3-540-69260-7, pp. 273-299, 2008.

[24] F. Xhafa and A. Abraham, "Meta-heuristics for Grid Scheduling Problems", Metaheuristics for Scheduling: Distributed Computing Environments, Studies in Computational Intelligence, Springer Verlag, Germany, ISBN: 978-3-540-69260-7, pp. 1-37, 2008.

[25] F. Xhafa, B. Duran, A. Abraham and K. Dahal, Tuning Struggle Strategy in Genetic Algorithms for Scheduling in Computational Grids, Neural Network World, Volume 18, No. 3, pp. 209-225, 2008.

[26] F. Xhafa, J. Carretero, and A. Abraham, Genetic Algorithm Based Schedulers for Grid Computing Systems, International Journal of Innovative Computing, Information and Control, Volume 3, Issue 5, pp. 1053-1071, 2007.