Future Generation Computer Systems 26 (2010) 228–235

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An auction method for resource allocation in computational grids

Hesam Izakian [a,b], Ajith Abraham [b,*], Behrouz Tork Ladani [a]

[a] Department of Computer Engineering, Faculty of Engineering, University of Isfahan, Isfahan, Iran

[b] Machine Intelligence Research Labs — MIR Labs, Scientific Network for Innovation and Research Excellence, P.O. Box 2259, Auburn, WA 98071, USA[1]

ARTICLE INFO

Article history:
Received 18 February 2009
Received in revised form
31 July 2009
Accepted 7 August 2009
Available online 19 August 2009

Keywords:
Computational grids
Resource allocation
Auction

ABSTRACT

A computational grid is composed of a set of resource consumers and resources providers. Usually these entities are independent and making decisions autonomously based on their policies and resource allocation in such systems is a challenging problem. In such systems using market-like techniques for this problem regulates the supply and demand for resources, provides an incentive for providers, and motivates the users to trade-off between deadline, budget, and the required level of quality of service. In this paper, we introduce a continuous double auction method (CDA) for grid resource allocation in which resources are considered as provider agents and users as consumer agents. In our proposed method these entities are allowed to participate in a grid independently and make decisions autonomously. We study this method in terms of economic efficiency and system performance. Experimental results illustrate that the proposed method is efficient in terms of successful execution rates, resource utilization rates and fair profit allocation.

1. 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]. They support the creation of virtual organizations and enterprizes that enable the sharing, exchange, selection, and aggregation of geographically distributed heterogeneous resources. Users and providers can access the grid networks by running grid portals such as Globus [2], Legion [3], etc on their machines. Also based on the grid resource management policies they can use grid resources or share their resources in a grid environment. 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, the grid environment is highly dynamic, heterogeneous, and uncontrollable and resources are distributed across different administrative domains.

Unlike scheduling problems in conventional distributed systems, this problem is much more complex as new features of grid systems such as its dynamic nature and the high degree of heterogeneity of jobs and resources must be tackled and resource allocation methods of traditional parallel and distributed computing cannot work efficiently. 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 is one of solutions which has received much attention. It enables the regulation of supply and demand for resources; provides an incentive for resource owners to participate in the Grid; and motivates the users to trade-off between deadline, budget, and the required level of quality of service [4].

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 [5]. Two categories of market based models that are used for grid resource management are commodities market models and auction models. In the commodity market model, providers specify their resource price and charge users according to the amount of resource they consume. In the 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,

* Corresponding author.
E-mail addresses: hesam.izakian@gmail.com (H. Izakian), ajith.abraham@ieee.org (A. Abraham), ladani@eng.ui.ac.ir (B.T. Ladani).

[1] http://www.mirlabs.org.

0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.08.010

and easy to implement in a grid setting [4]. 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 [4]. 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) [6], Threshold Price Double Auction Protocol (TPDA) [7], and Continuous Double Auction Protocol (CDA). Kant and Grosu [8] showed that the CDA protocol is better from both the resource's and the 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 the resource consumers' and the resource providers' perspective. The remainder of this paper is 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.

2. Related works

Economic based resource management systems have been investigated by several researchers in [4,8–14,5,15–19]. Buyya et al. [4] used economic based concepts including commodity market, posted price modeling, contract net models, bargaining modeling, etc. for grid resource allocation. Nimrod-G [16] is a computational economy driven resource broker that manages Grid resources. Nimrod-G supports several economic models such as commodities markets, spot markets, and contract net.

Grosu and Das [8] investigated the First-Price Auction, Vickrey Auction and Double Auction models for resource allocation in grids to find which one is more suitable for the grid environment from a users' and a resources' perspective. Authors found that when a mixture of risk-averse and risk-neutral users is considered, First-Price Auction favors resources while Vickrey Auction favors users. Also the Double Auction favors both users and resources. Also Reddy et al. [14] developed a sealed bid method for optimizing the time and budget in grid environments.

Gomoluch and Schroeder [13] investigated the performance of the double auction protocol for resource allocation. They compared it to the conventional round-robin approach and showed that the double auction protocol outperforms round-robin. Attanasio et al. [10] developed an auction mechanism based on a progressive Lagrangean heuristic and showed that it was able to provide comparable efficiency to centralized heuristics.

In [15] authors proposed a combinatorial auction-based resource allocation protocol in which a user bids a price value for each of the possible combinations of resources required for its task execution. The protocol involves an approximation algorithm for solving the combinatorial auction problem.

Huang et al. [9] 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. Also in [12] authors proposed an approach based on macroeconomics, which concerns the overall benefit of the whole Grid market and is well suited for service-oriented Grids. In this method, some realizations of resource allocation strategies driven by the macroeconomic principle are given. The proposed method increases the total profit of all of the grid service providers, and reduces the failure rate of Grid service requests and the inquiry time for resource consumers, while getting good load balancing of the whole Grid market.

In [18] authors present several multi-cost algorithms for the joint scheduling of the communication and computation resources that will be used by a grid task. Also they introduced a multi-cost scheme that performs immediate reservations and selects the computation resource to execute the task and determines the path to route the input data.

In [19], three meta-scheduling heuristics that minimize and manage the execution cost and makespan of user applications are proposed. Also a cost metric to manage the trade-off between execution cost and time is presented. Xiao et al. [5] presented 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. Wolski et al. [17] investigated G-commerce computational economies for controlling resource allocation in Computational Grids. They measured the efficiency of resource allocation based on commodities markets and auctions models.

Anthony et al. [20] 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. [21] introduced a bidding strategy for obtaining goods in multiple overlapping English auctions. Authors 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.

3. 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 the form of a double auction protocol for obtaining their objectives.

We assume that the grid consists of m resources $R = \{R_1, R_2, \ldots, R_m\}$ each represented by a provider agent and a set of k users $S = \{S_1, S_2, \ldots, S_k\}$, 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 $J = \{J_1, J_2, \ldots, J_n\}$. Each job J_i can be submitted at various times t_i in which $0 \leq t_i \leq T$. 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 ith 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 the 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 instructions that 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

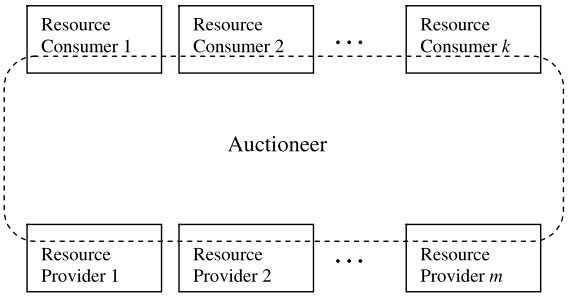

Fig. 1. Scheduling scheme in the proposed framework.

price for providing a service by R_j (also called the 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 the First Come First Served (FCFS) method for executing the accepted jobs. Fig. 1 illustrates the scheduling scheme in the proposed method.

4. 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 the 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.

4.1. 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 bidding. This method is inspired by the work presented in [20]. The consumer agent based on these parameters and its corresponding job decides a bid value autonomously.

4.1.1. Determining the bid value based on the number of remaining resources for a 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 time, the number of remaining resources decreases (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).

$$bid^{i,t}_{remaining\_resources} = \left[\frac{r_{\min}}{\varphi_i} + \left(1 - \frac{r_{\min}}{\varphi_i}\right)\left(1 - \frac{N_i^t - 1}{N_i^{\max}}\right)^{\frac{1}{\alpha}}\right] \times \varphi_i \tag{2}$$
where, $0.01 \leq \alpha \leq 100$.

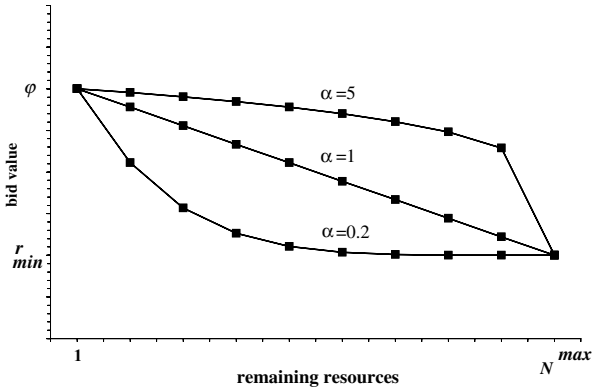

Fig. 2. Bid value for remaining resources.

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 α. 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 φ_i, the maximum bid value per MI. Fig. 2 depicts the different convexity degrees of the curves with $\alpha = .2, 1, 5$. In this method when the number of remaining resources decreases, the bid value will increase.

4.1.2. 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).

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

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

$$rt^t_i = \frac{\sum_{j=1}^{m}(rt^t_{i,j} \times y_{i,j})}{N_i^{\max}} \quad \text{where, } y_{i,j} = \begin{cases} 1 & \text{if } rt^t_{i,j} > 0 \\ 0 & \text{else.} \end{cases} \tag{4}$$

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

$$bid^{i,t}_{remaining\_time} = \left[\frac{r_{\min}}{\varphi_i} + \left(1 - \frac{r_{\min}}{\varphi_i}\right)\left(1 - \frac{rt_i^t}{rt_i^{\max}}\right)^{\frac{1}{\beta}}\right] \times \varphi_i \tag{5}$$
where, $0.01 \leq \beta \leq 100$

In (5) the parameter β is similar to α in (2) and is used for controlling convexity degrees of the curve.

4.1.3. 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).

$$bid^{i,t} = \lambda \times bid^{i,t}_{remaining\_resources} + (1-\lambda) \times bid^{i,t}_{remaining\_time} \quad (6)$$

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

λ 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 the 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.

4.2. 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 the resource is zero and the provider sets the price to the 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 decreases 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 the 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.

$$request^t_j = \left[\frac{r_j}{mp_j} + \left(1 - \frac{r_j}{mp_j}\right) \left(\frac{st^t_j}{wl^t_j}\right)^{\frac{1}{\sigma}} \right] \times mp_j. \quad (7)$$

In (7) $request^t_j$ is the request value of the jth provider at time t, wl^t_j is the workload of resource R_j after the last allocation, and st^t_j is the current workload or start time of the new job if it is accepted at time t. Also σ is the same as α and β in Eqs. (2) and (5) respectively and is used for controlling convexity degrees of the curve. Fig. 3 depicts the different convexity degrees of the curves with $\sigma = .2, 1, 5$.

4.3. 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:

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

5. Simulation and experimental results

In order to study the efficiency of the method presented in this paper, we developed a computational grid simulator using the java agent development framework (JADE) [22]. 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 are modeled as two kinds of agent. Also in this system, we ignore the network delay of communications and focus on the job execution and negotiations between consumer and provider agents.

In our experiments we use the system load concept i.e. the ratio of aggregated length of jobs submitted to the grid to the aggregated job length that the grid is capable of performing in the simulation period. The system load can be obtained using Eq. (9).

$$\phi = \frac{\sum_{i=0}^{n} l_i}{T \times \sum_{j=0}^{m} c_j}. \quad (9)$$

In this equation n is the number of jobs submitted to the grid and m is the number of resources. l_i denotes the length of job i (MI) and c_j is the computational capacity of resource j (MIPS). Also T is the period of simulation.

In our system, there are 50 consumers and 100 providers and the total jobs are 50,000. Consumers independently generate jobs from time to time and the interval between two job generations is exponentially distributed. The lengths of the jobs are considered as a random integer within the range [1000, 10000] sampled from a uniform distribution and the duration from the time instance of job generation to its deadline is uniformly distributed as well. Also the computational capacity of providers is normally distributed within the range [10, 100]. For a fair comparison, the reserve price and maximum price for all providers is considered as 2 G$ /MIPS and 6 G$ /MIPS respectively. The budget allocated to each job J_i is set according to Eq. (10). Also in our experiments, we set $\alpha = \beta = 0.2$, $\sigma = 5.0$, and $\lambda = 0.6$.

$$b_i = rand(3, 5) \times l_i. \quad (10)$$

5.1. Experiment 1

In this experiment we investigate the impact of α, β, σ, and λ on scheduling performance. We used the failure rates of jobs as metric which is the ratio of the number of jobs that miss their deadlines to the number of jobs that submitted to the grid. Table 1 shows the failure rates of jobs based on different values of α, β, σ. As can be seen in this table, our proposed method performs best where $\alpha = \beta = 0.2$, and $\sigma = 5.0$. When α, β are 0.2, the consumer maintains a low bid value until the number of the remaining resources and the remaining time for biding gets close to zero (see Fig. 2). On the other hand $\sigma = 5.0$ means that the provider requests a high value until the workload gets close to zero (see Fig. 3). Based on this strategy, consumers try to perform their jobs with a low cost and providers try to rent their resources with high values. Therefore only consumers which have high risk to miss their deadlines bid more values and consequently the failure rates of jobs decreases.

Table 2 shows the failure rates of jobs based on different values of λ. When $\lambda = 1$ only the remaining resources constraint is considered and when $\lambda = 0$, only the remaining time constraint is considered. In this table we can see that if both parameters are taken into account (for example $\lambda = 0.5$), the failure rates of jobs can be decreased.

5.2. Experiment 2

For investigating the efficiency of the presented method, we compare it with first come, first served (FCFS), shortest job first (SJF), earliest deadline first (EDF), and Incentive-Based (IB) [5] schemes. The first three are straightforward, easy to implement, and often taken as default policies in scheduling systems. For

Table 1
Failure rates of jobs based on different values of α, β, and σ.

System load	$\alpha = \beta = 0.2$			$\alpha = \beta = 1.0$			$\alpha = \beta = 5.0$		
	$\sigma = 0.2$	$\sigma = 1.0$	$\sigma = 5.0$	$\sigma = 0.2$	$\sigma = 1.0$	$\sigma = 5.0$	$\sigma = 0.2$	$\sigma = 1.0$	$\sigma = 5.0$
0.2	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%
0.4	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%
0.6	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%	00.00%
0.8	06.69%	04.86%	04.18%	07.01%	05.79%	05.60%	07.65%	07.23%	06.11%
1.0	16.21%	14.37%	13.89%	16.86%	14.79%	14.47%	17.96%	15.74%	15.33%

Table 2
Failure rates of jobs based on different values of λ.

System load	$\lambda = 0.0$	$\lambda = 0.5$	$\lambda = 1.0$
0.2	00.00%	00.00%	00.00%
0.4	00.00%	00.00%	00.00%
0.6	00.00%	00.00%	00.00%
0.8	08.23%	05.18%	06.05%
1.0	19.52%	14.74%	16.46%

implementing these three algorithms we assume that at the beginning of the simulation we know the deadline and the time of submitting each job to the grid. In the FCFS algorithm, the jobs are sorted by their arrival time, in the SJF, the jobs are sorted by their length, and in the EDF algorithm the jobs are sorted by their deadline. Then each job is dispatched to the resource that can perform it within its deadline. In these methods, in the case there is no resource that can perform the job within its deadline, the system removes that job. Also if there is more than one candidate resource for performing the job, then one of them will be selected randomly. The Incentive-Based scheduling (IB) scheme utilizes a peer to peer decentralized scheduling framework and can be used in market-like computational grids [5]. In this method for performing the job in the grid, the consumer submits a job announcement to the grid environment, and the job announcement is broadcast to all the providers. Upon receiving a job announcement, each provider estimates whether it is able to meet the deadline of the job. If yes, the provider sends a request that includes the price for executing the job to the consumer. After waiting for a certain time, the consumer processes all the requests and chooses the provider who charges the least, and sends the job to the selected provider. The provider who receives the job inserts it in the job queue based on a local scheduling algorithm and when the job is finished, sends the results to the consumer. We compare these scheduling methods with ours on the three evaluation criteria: *successful execution rate*, *resource utilization rate*, and *fairness deviation* [5].

5.2.1. Successful execution rate

Successful job execution means that a job is performed within its deadline. In order for users to perform their computational intensive tasks in grid and be willing to pay for that, they should make sure their tasks will be performed within the deadline. The more successful execution rates in the grid, the more users are motivated to perform their tasks through the grid. The successful execution rate can be obtained using Eq. (11).

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

where T_C^i denotes the completion time of job J_i. As shown in Fig. 4, the proposed method outperforms others over successful execution rate. The reason is that in our proposed method, the tasks with more remaining time and remaining resources for bidding typically bid a low amount. As time passes and the remaining time and remaining resources decrease, they will increase their bid value. Consequently, the tasks with a shorter remaining time and fewer remaining resources for bidding can win in the auction with a higher bid values and can perform before their deadlines. In fact, this is similar to the EDF method with the main difference being that in this method both the remaining time and remaining resources for bidding are considered.

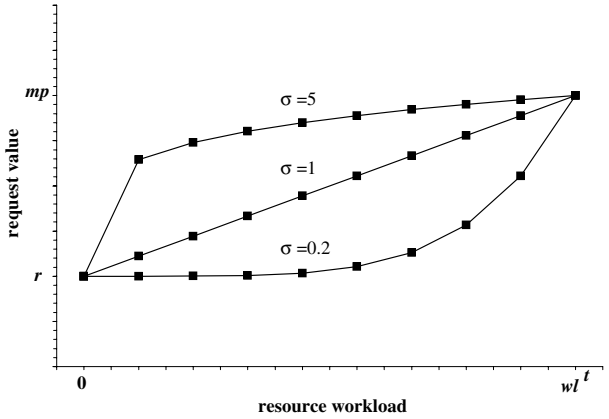

Fig. 3. Determining request value based on workload.

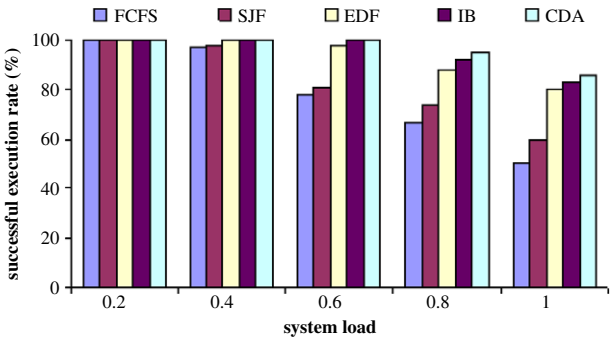

Fig. 4. Comparison of successful execution rate between our method and others.

5.2.2. Resource utilization rate and load balancing level

The resource utilization rate defined as the percentage of time a resource is busy executing jobs. In order for the grid resources to be efficiently used, the grid manager should increase their utilization rate to decrease their idle time. The resource utilization rate can be obtained using Eq. (12).

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

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. The average resource utilization rate u of all resources can be obtained

Table 3
Load balancing level.

System load	FCFS	SJF	EDF	IB	CDA
0.2	66.1%	65.6%	63.8%	97.8%	89.3%
0.4	74.5%	78.8%	74.1%	98.2%	94.4%
0.6	86.8%	87.4%	80.6%	99.2%	97.7%
0.8	91.8%	92.0%	85.0%	98.9%	98.5%
1.0	95.7%	95.0%	87.7%	98.6%	99.1%

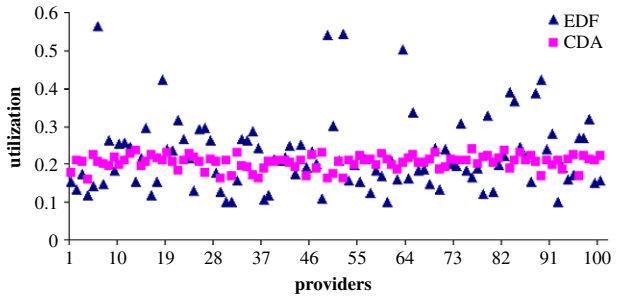

Fig. 5. Comparison of resource utilization between EDF and CDA with system load 0.2.

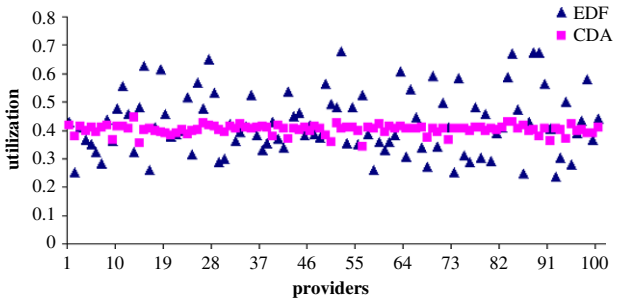

Fig. 6. Comparison of resource utilization between EDF and CDA with system load 0.4.

using (13), where m is the number of resources.

$$u = \frac{\sum_{j=1}^{m} u_j}{m}. \tag{13}$$

The load balancing level of the grid can be estimated using (14) [23].

$$LB = \left(1 - \frac{d}{u}\right) \times 100\% \quad \text{where, } d = \sqrt{\frac{\sum_{j=1}^{m}(u - u_j)^2}{m}} \tag{14}$$

where d is the mean square deviation of u_j. The most effective load balancing is achieved when d equals zero and LB equals 100%.

Table 3 shows the load balancing levels for compared methods. It is evidence that the IB and our proposed method have the greatest load balancing level. Also we can see that in the compared methods (except IB) an increase in system load leads to an increase in load balancing level.

Figs. 5–9 show the balanced utilization of our method and imbalanced utilization of EDF for system load 0.2, 0.4, 0.6, 0.8, and 1.0 respectively. In the presented method, with the decrease in the workload of each resource, its requested value will decrease and an increase in the workload will increase its requested value. Since the users attempt to perform their tasks with a lower price, typically the tasks are allocated to machines with lower workloads and hence the resources will not remain idle. In addition, this causes load balancing in the available resources.

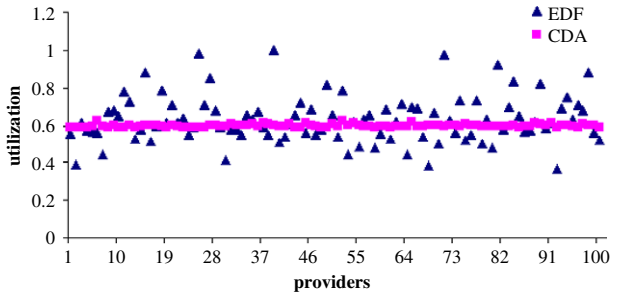

Fig. 7. Comparison of resource utilization between EDF and CDA with system load 0.6.

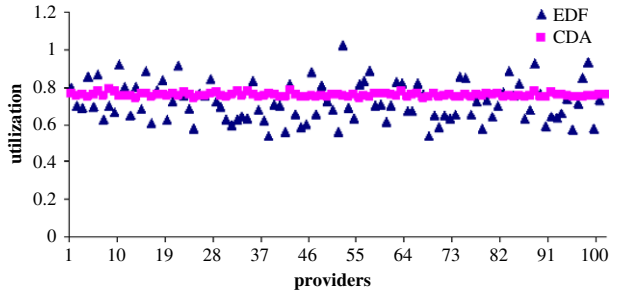

Fig. 8. Comparison of resource utilization between EDF and CDA with system load 0.8.

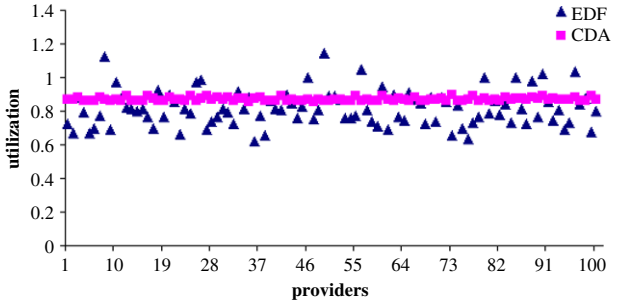

Fig. 9. Comparison of resource utilization between EDF and CDA with system load 1.0.

5.2.3. 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 [5]. The fairness of the market is an intensive metric for resource owners to stay in a grid and play a role. A resource allocation scheme is fair if the fairness deviation of resources is minimized. The fairness deviation of the grid system can be obtained using (15).

$$\sigma = std\_dev(\Delta_1, \ldots, \Delta_m) \quad \text{where, } \Delta_j = \frac{p_j \Big/ \sum_{l=1}^{m} p_l}{c_j \Big/ \sum_{l=1}^{m} c_l}. \tag{15}$$

In (15), p_j denotes the profit of resource R_j. Fig. 10 shows the comparison of the fairness deviation between our method and other methods under different system loads. As shown in Fig. 10, the fairness deviation of the IB method is less than others while our method obtains an admissible fairness deviation. In our proposed method, the higher the computational capacity of the resources, the shorter the execution time of the accepted tasks, and hence on the basis of the proposed method, the lower the request value.

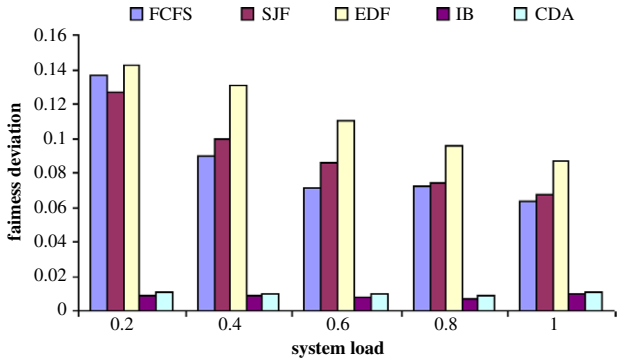

Fig. 10. Comparison on the fairness deviation.

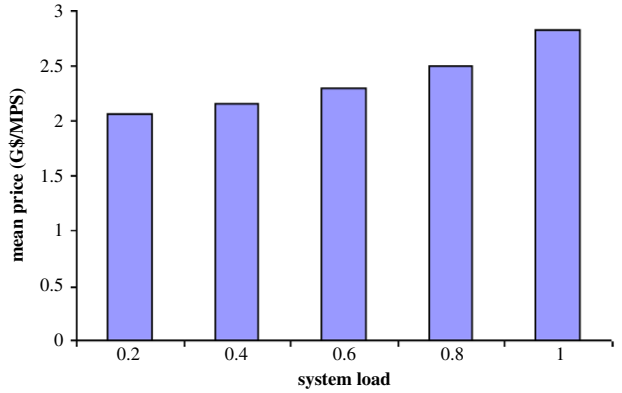

Fig. 11. Mean price in different system loads.

Since the users look for cheaper resources, more tasks will be allocated to resources with higher computational capacity. In other words, each resource accepts the tasks according to its computational capacity compared to the other available resources. On the other hand, to perform their tasks, the users will not spend their entire budget and will pay for resources according to the market price. This causes each resource to benefit according to its computational capacity and consequently results in a decrease in the fairness deviation.

In the IB scheme which was proposed in [5], each provider has to know some global information about other providers (for example the total capacity of providers) and the total length of allocated jobs. Also there is a predefined global market price where the providers have to set their price to it and they can increase or decrease their price a little. This leads to a very small fairness deviation, but obtaining this information in a real grid environment is difficult.

5.3. Experiment 3

In this experiment, we show that the proposed framework supports the *supply and demand* model [24]. 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 price with different system loads. Fig. 11 shows the mean of resource prices in different system loads. As shown in Fig. 11, when the system load (demand) increases, prices increase and when the system load decreases, prices decrease. This figure shows that the presented method supports the supply and demand model.

6. Conclusion

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 the remaining time for bidding and the 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] I. Foster, C. Kesselman, Globus: A meta-computing infrastructure toolkit, International Journal of Supercomputer Applications 11 (2) (1997) 115–128.
[3] A.S. Grimshaw, W.A. Wulf, Legion Team, The legion vision of a worldwide virtual computer, Communications of the ACM 40 (1) (1997) 39–45.
[4] R. Buyya, D. Abramson, J. Giddy, H. Stockinger, Economic models for resource management and scheduling in Grid computing, The Journal of Concurrency and Computation 14 (13–15) (2002) 1507–1542.
[5] L. Xiao, Y. Zhu, L.M. Ni, Z. Xu, Incentive-based scheduling for market-like computational grids, IEEE Transactions on Parallel and Distributed Computing 19 (7) (2008) 903–913.
[6] R.P. McAfee, A dominant strategy double auction, Journal of Economic Theory 56 (1992) 434–450.
[7] M. Yokoo, Y. Sakurai, S. Matsubara, Robust double auction protocol against false-name bids, in: Proc. of the 21st IEEE International Conference on Distributed Computing Systems, April 2001, pp. 137–145.
[8] 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.
[9] Z. Huang, Y. Qiu, Resource trading using cognitive agents: A hybrid perspective and its simulation, Future Generation Computer Systems 21 (2007) 837–845.
[10] A. Attanasio, G. Ghiani, L. Grandinetti, F. Guerriero, Auction algorithms for decentralized parallel machine scheduling, Parallel Computing 32 (2006) 701–709.
[11] K.M. Chao, R. Anane, J.H. Chen, R. Gatward, Negotiating agents in a market-oriented grid, in: Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, May 2002, pp. 406–407.
[12] P. Huang, H. Peng, Piyuan Lin, X. Li, Macroeconomics based grid resource allocation, Future Generation Computer Systems 21 (2008) 694–700.
[13] J. Gomoluch, 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, June 2003, pp. 211–218.
[14] S.R. Reddy, A. Gupta, Auction Based Resource Allocation in Grids, Springer-Verlag, Berlin Heidelberg, 2006.
[15] A. Das, D. Grosu, Combinatorial auction-based protocols for resource allocation in grids, in: The 19th IEEE International Parallel and Distributed Processing Symposium, IPDPS'05, 2005.
[16] D. Abramson, R. Buyya, J. Giddy, A computational economy for grid computing and its implementation in the Nimrod-G resource broker, Future Generation Computing Systems 18 (8) (2002) 1061–1074.
[17] R. Wolski, J.S. Plank, J. Brevik, T. Bryan, Analyzing market-based resource allocation strategies for the computational grid, The International Journal of High Performance Computing Applications 15 (3) (2001) 258–281.
[18] T. Stevens, M. De Leenheer, C. Develder, B. Dhoedt, K. Christodoulopoulos, P. Kokkinos, E. Varvarigos, Multi-cost job routing and scheduling in Grid networks, Future Generation Computer Systems 21 (2009) 912–925.
[19] S.K. Garg, R. Buyya, H.J. Siegel, Time and cost trade-off management for scheduling parallel applications on utility grids, Future Generation Computer Systems 21 (2009).
[20] P. Anthony, N.R. Jennings, Developing a bidding agent for multiple heterogeneous auctions, ACM Transactions on Internet Technology 2 (3) (2003) 185–217.
[21] 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.
[22] JADE, Java Agent Development Framework, 2004. http://jade.cselt.it.
[23] J Cao, D.P. Spooner, S.A. Jarvis, G.R. Nudd, Grid load balancing using intelligent agents, Future Generation Computer Systems 21 (2005) 135–149.

[24] L.W. McKnight, J. Boroumand, Pricing internet services: Approaches and challenges, IEEE Computer 33 (2) (2000) 128–129.

Hesam Izakian received a B.Sc. in Software Engineering from Birjand University (Birjand, Iran) in 2005 and an M.Sc. in Artificial Intelligence from University of Isfahan (Isfahan, Iran) in 2009. Currently he is a lecturer in Azad University, Ramsar branch (Ramsar, Iran). His research interests are Grid Computing, heterogeneous environments, computational intelligence, and evolutionary algorithms.

Ajith Abraham received the Ph.D. degree from Monash University, Melbourne, Australia in 2001 and a Master of Science degree from Nanyang Technological University, Singapore. His research and development experience includes over 19 years in the Industry and Academia. He works in a multi-disciplinary environment involving machine intelligence, network security, sensor networks, e-commerce, Web intelligence, Web services, computational grids, data mining and applied to various real world problems. He has given more than 30 plenary lectures and conference tutorials in these areas. He has published over 500+ publications (with colleagues from nearly 30 countries) and some of the works have also won best paper awards at International conferences and also received several citations. Some of the articles are available in the ScienceDirect Top 25 hottest articles. He co-chairs the IEEE Systems Man and Cybernetics Society Technical Committee on soft computing. Currently he is also coordinating the activities of the Machine Intelligence Research Labs (MIR Labs), International Scientific Network of Excellence, which has members from over 45 countries. He has a world wide academic experience with formal appointments in Monash University, Australia; Oklahoma State University, USA; Chung-Ang University, Seoul; Jinan University, China; Rovira i Virgili University, Spain; Dalian Maritime University, China; Yonsei University, Seoul and Open University of Catalonia, Spain, National Institute of Applied Sciences (INSA), France and Norwegian University of Science and Technology (NTNU), Norway. For about 2.5 years, he was working under the Institute of Information Technology Advancement (IITA) Professorship Program funded by the South Korean Government. He serves the editorial board of several reputed International journals and has also guest edited over 30 special issues on various topics. He is actively involved in the Hybrid Intelligent Systems (HIS); Intelligent Systems Design and Applications (ISDA); Information Assurance and Security (IAS); and Next Generation Web Services Practices (NWeSP) series of International conferences, besides other conferences. He is a Senior Member of IEEE, IEEE Systems Man and Cybernetics Society, IEEE Computer Society, IET (UK), IEAust (Australia) etc. More information at http://www.softcomputing.net.

Behrouz Tork Ladani received a B.Sc. in Software Engineering from University of Isfahan (Isfahan, Iran) in 1996, an M.Sc. in Software Engineering from Amir-Kabir University of Technology (Tehran, Iran) in 1998, and a Ph.D. in Computer Engineering from Tarbiat-Modarres University (Tehran, Iran) in 2005. He is currently an Assistant Professor in Department of Computer Engineering, University of Isfahan. His research interests include Formal Specification and Verification, Cryptographic Protocols, Multi Agent Security, and Web Services. He is a member of Iranian Society of Cryptology (ISC).