A Bargaining based Scheduling for Resources Advanced Reservation Using Simulated Annealing into Grid System

Seyedeh Yasaman Rashida[1], Hamidreza Navidi[2]
[1]Department of Computer Engineering, Shirgah Branch, Islamic Azad University
Shirgah, Mazandaran, Iran

[2]Department of Mathematics, Shahed University
Tehran, Iran

Abstract
The concept of grid computing is getting popular day to day with the emergence of the Internet as a ubiquitous media and the wide spread availability of powerful computers and networks as low-cost commodity components. In these environments requests are served from external users along with local users. Since there are a limited number of resources to be used in the grid system, in spite of vast requests, resources management and scheduling is a complex undertaking. The resource consumers adopt the strategy of solving their problems at low cost with in a required time frame and also the resource providers adopt the strategy of obtaining best possible return on their investment while trying to maximize their resource utilization by offering a competitive service access cost in order to attract consumers. In this paper, we propose a bargaining based scheduling for resource advanced reservation using Simulated Annealing such that consumers can choose providers that best meet their requirements with low price. To achieve the goals, we use a maximum conflict algorithm that we presented in 2010. The simulation results indicate that the scheduling lead to maximize number of reserved requests in their deadline and both consumers and providers obtain maximum profits.
Keywords: *Simulated Annealing, Scheduling, Grid Computing, Bargaining.*

1. Introduction

Grid computing has emerged as a new paradigm of distributed computing technology since mid1990s [22, 3]. It focuses on large-scale resource sharing and coordinated problem solving. Providing non-trivial quality of service (QoS) is one of the primary goals of the Grid approaches. Many applications largely depend on obtaining results within particular QoS requirements.

For achieving the goals, one mechanism can be Advance Reservation. When the allocation of computing resources is made using advance reservation mechanisms, the resources are allocated a long time before they are actually used [3, 4]. On the other hand, Advance reservation is the process of a consumer booking a resource from a provider for a job at some future date. The resources could be a cluster, network, room, and visualization system. These reservations may not all be at the same time as a user (e.g. scientist) may want to book some resources sequentially [22].

Grid scheduling is the process of scheduling jobs over grid resources. Improving overall system performance with a lower turnaround time and low cost is an important objective of grid scheduling. In attempting to solve these problems and increase total number of reserved jobs, we propose an extended allocating jobs algorithm that finds resources using Simulated Annealing.

In attempting to reason about interactions between users, the artificial intelligence community has recently developed an interest in game theory, a tool from economics [13]. Game theory aims to help us understand situations in which decision-makers interact. This paper applies a bargaining based scheduling to increase profit of consumers and providers.

The rest of this paper is organized as following. In section 2, we discuss related work. Section 3 describes the proposed scheduling Model. In section 4, we compare our proposed algorithm that uses simulated Annealing, with the algorithm which uses Hill climbing. Section 5 gives the concluding remarks.

2. Related work

Grid Computing is a platform for coordinated resource sharing and problem solving on a global scale among virtual organizations. Grid uses Grid Services to access and use a set of Grid resources [2].

Scheduling has been one of the key challenges and widely studied subjects in enabling computational grid systems in the last decade. Many conventional scheduling strategies, either centralized or distributed, are presented that are inefficient and required complicated, which resulted in performance loss. In following, we explain some of them and then propose our different strategy to solve some scheduling issues.
The paper [8] presents a novel load balancing approach in a heterogeneous distributed environment. The scheduler takes into account the threshold value, based on the ratio of service rates, along with the queue length to determine whether it is beneficial to migrate a given local task to another node in the system or not. Markov process model is used to describe the behavior of the heterogeneous distributed system under the proposed

policies. Kumar also proposes a Load balancing algorithm for fair scheduling, and compares it to other scheduling schemes such as the Earliest Deadline First, Simple Fair Task order, Adjusted Fair Task Order and Max Min Fair Scheduling for a computational grid. It addresses the fairness issues by using mean waiting time. It scheduled the task by using fair completion time and rescheduled by using mean waiting time of each task to obtain load balance. This algorithm scheme tries to provide optimal solution so that it reduces the execution time and expected price for the execution of all the jobs in the grid system is minimized [20]. Shahu Chatrapati et al. [21] propose Competitive Equilibrium Scheme (CES) that simultaneously minimizes mean response time of all jobs, and the response time of each job individually.

Recently we have witnessed a number of game and related economic theory applications in various research fields including but not limited to demand-side management, and communications [9], brokering [15], power management [7], workload balancing [3, 19] and incentive mechanism design [9]. In grid computing, game theory is extremely helpful in modeling behaviors of benefit-driven users. Typical game theoretical methods in grid mechanism design define objective functions in term of utility, and converge to system equilibrium state on the basis of revenue maximization. For example, in computational grid, scheduling and job execution strategy in non-cooperative game are investigated in [4] and [11] respectively, both demonstrate that Nash equilibrium is not the best out-come. The economic and game theoretical research also spurs development of market-oriented grid systems. For example, Buyya et al. have proposed Nimrod/G [6], a resource broker which coordinates resource allocations in grids by integrating multiple economic models. Luis Rodero-Merino and *et al* in [16] introduce and analyze an economic mechanism to set resource prices and resolve when to scale resources depending on the consumers' demand. They explain that no consumer hinders the execution of other consumers' tasks by getting too many resources, in order that the system has a strong emphasis on fairness. The results of simulation show how the proposed system can successfully adapt the amount of allocated resources to the demand, while at the same time ensuring that resources are fairly shared among consumers. The paper [17] depicts and evaluates broker selection strategies for job reservation and bidding. It analyzes two different types of existing algorithms simple and categorized aggregation algorithm. The first algorithm which aggregates the resource information acts as input for the categorized aggregation algorithm to assign rank for the resources. Meta broker allocates the job based on the rank. They proposes advanced job reservation algorithm for resource allocation. Using this advanced resource algorithm can reserve the resource for job allocation, even though no resources are free to run the job. They also propose bidding technique when more than one consumers approach same resources. The results show that the proposed system reduces the execution time and generates better revenue for Meta broker. The paper [23] also presents the use of commodity economy model for resource management and application scheduling in both computational and data grids.

Current literature of auction-based scheduling mainly focuses on single item auction. Grosu *et a*l. [14] have investigated popular auction methods and proposed double auction protocols for resource allocation. On the other hand, combinatorial auctions [22], although have been researched intensively in economic study for years, did not receive sufficient attentions in computer science until recently. A number of heuristic methods [10], [16], [17] aiming to solve the winner determination problem (WDP) have been proposed. However, these methods focused on approximation of WDP. To the best of knowledge, Foster and Kesselman are the first to apply the simultaneous ascending auction method [22] and systematically model it to suit realistic grid environments. Inspired by Wolski's G-commerce [15] and Ghosh's bargaining methods [1] in mobile grids, the proposed BarSAA algorithm is novel in that we combine the supply-demand adjustment of commodity markets in G-commerce and bargaining process in Ghosh's methods with auction theory. The allocation process is dynamically adaptive and achieves Vickrey-Clarke-Groves (VCG) outcome for both auctioneers and bidders. The most relevant research is proposed by Garg, *et al*. [18], in which a continuous double auction is employed by the meta-scheduler for resource mapping in global grids.

In this paper we propose a novel algorithm for resource advanced reservation using simulated annealing which try to ensure the end-to-end QoS and improvement of the efficiency of grid resources. This algorithm increases the total number of reserved jobs. Using simulated annealing instead of Hill climbing causes resources are found faster and also match with QoS of requests. The proposed algorithm also uses bargaining method to utility profit of users and resource providers. In follow, we explain our model in detail and then compare it with a deadline algorithm that presented in [12].

3. The Proposed Model

The model under consideration views the grid as a dynamic federation of resources contributed by various organizations. Each cluster constitutes a private management domain. It provides a set of grid services assumed to be exposed in a fashion that reflects the basic outlines of the OGSI recommendations [14]. Resources may join or leave the grid at any time without any disruption to the grid operation. The effect of this dynamic membership is limited to the configuration of neighboring clusters. Each cluster includes a set of users, providers and resources. Providers host the offered services or resources (see Fig. 1).

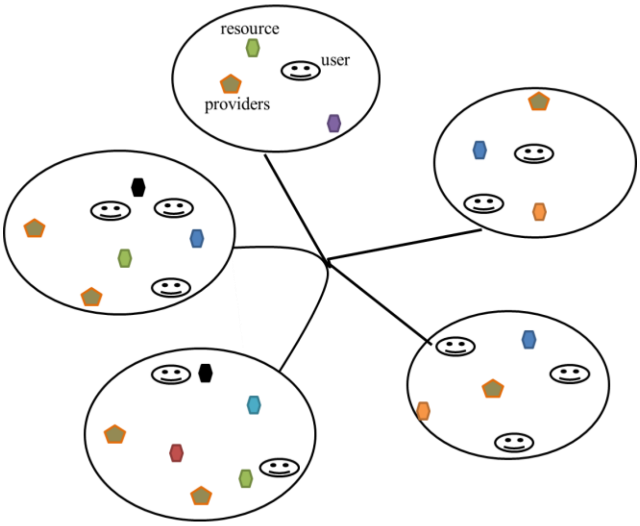

Fig 1. Representation of clusters with users and resources into grid system

Each user has own service requests to be scheduled that can submit his requests at any time. Users send their requests to scheduler which looks for matched resources base on a bargaining-based algorithm. How users and providers communicate together is shown in Fig. 2.

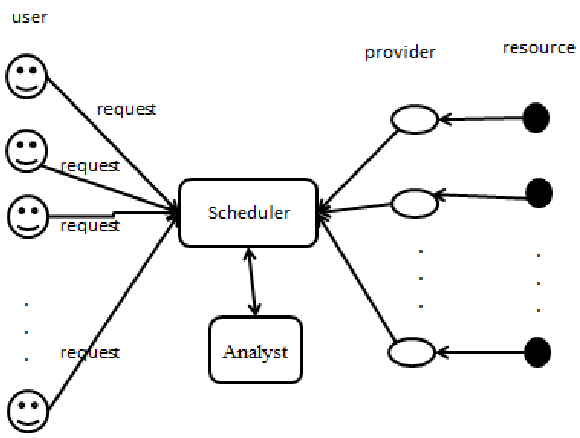

Fig. 2. The proposed model for reserving requests

In this model, there is a scheduler that is responsible to search, select and allocate the resource to requests. Since the model goals are increasing system utility and profit of users and resource providers, the scheduler uses analysis part called Analyst to analyze requests and controls overall status of grid. As shown in Fig 2, at first, users submit their service requests to the scheduler. The scheduler looks for the request's need by a bargaining based algorithm that we explain in next section.

3.1 Request Definition

We present a framework which use to design and develop an advanced reservation algorithm. A Grid resource would receive requests from different applications for execution of different tasks. We define each submitted advance reservation request by uniform seven dimension tuple: $<R, T_{start}, T_{stop}, T_{service}, Type, Q, Cost>$. Each request can require the set of resource to execute that are defined with R vector, $R = \{r_1, r_2, \ldots, r_m\}$. It is noticeable that each request has a deadline that it specifies the request should finish during this time otherwise the request will be removed and should resubmit later [12]. Job deadline is made on two parts, service time, $T_{service}$, and laxity, L, as shown in eq. 1.

$$DeadLine = L + T_{service} \tag{1}$$

Let T_{start} respectively represent the time at which the task associated an advance reservation request is available for execution, T_{stop} be the time that finish request deadline. Let $T_{service}$ be the time a task takes to finish executing on resources. Laxity, L is the time a job holds resources with no using, as shown in Fig. 3.

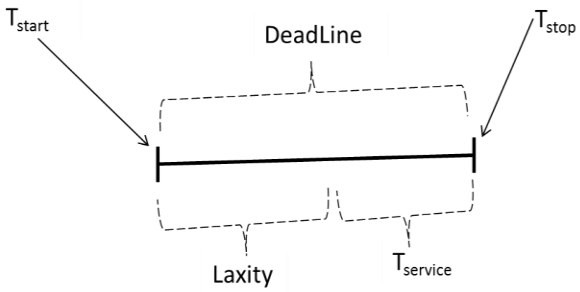

Fig. 3. Representation of elements of deadline

Let *Type* be the kind of resource and Q vector is quality of service of requested resources:

$$Q = \{Q_1, Q_2, \ldots, Q_m\}$$
$$Q_i = \{Q_i^1, Q_i^2, Q_i^3\}$$

Let Q_i be three mainly features of resource i as a user perspective. *Cost* be cost that user can cope with [12].

3.2 The Proposed Algorithm

The paper [12] presented a deadline based algorithm which used Hill climbing search to find requested resource for users. In this paper, we improve the algorithm and use simulated Annealing to search and match resources with requests. Then we show that our new algorithm is worked better than the old one. Additionally, our proposed algorithm uses bargaining theory to obtain best profit for both, users and resource providers. In follow, we explain the algorithm in detail (see Fig. 4).

Users submit their requests to scheduler for advance reservation resources at any time. When scheduler

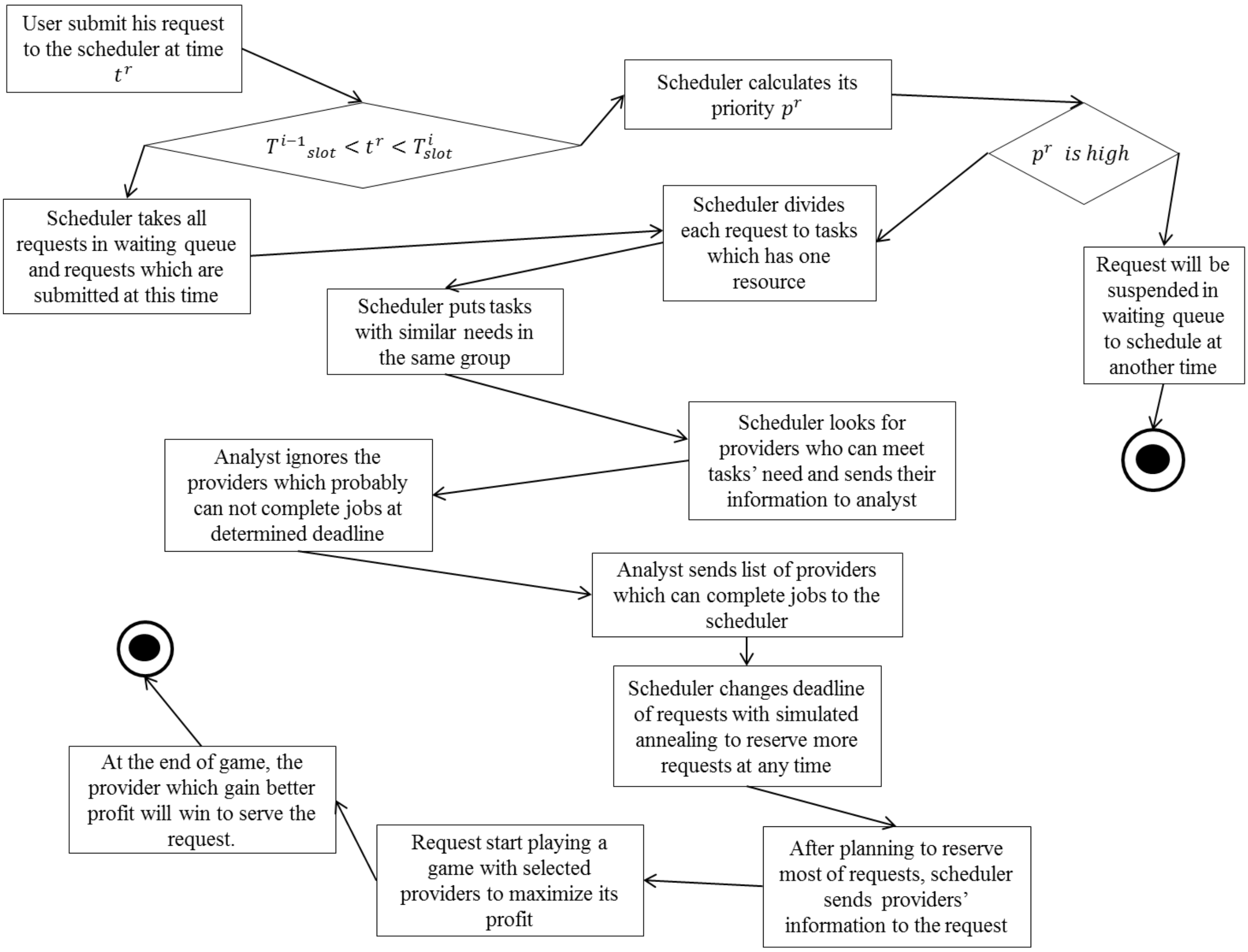

Fig. 3. The scenario of proposed scheduling

receives requests at time t^i that $T^{i-1}_{slot} < t^i < T^i_{slot}$, it surveys their priority p^r by eq. 2. Priority of request is computed by two parameters, deadline D^r and type of request T^r. Let T^r_{stop} be T_{stop} of request r and L^r be laxity of request r. And also T^i_{slot} is i-th time slot. Time slot is estimated by scheduler on the basis of traffic of network, system workload and number of submitted jobs. T_{slot} is not fixed and can be changed in time. α and β are important ratio of deadline and type of request.

$$p^r = \alpha T^r + \beta \frac{D^r}{T^r_{stop} - L^r - t^i} \quad (2)$$

$$0 < \alpha, \beta \leq 1 \quad (3)$$

After calculating the priority of requests, the scheduler surveys requests with high priority immediately after being submitted and the other requests with low priority will be embedded in waiting queue to be surveyed at start of next time slot, T^i_{slot} . In the case of a single request with no traffics at t^i in grid, the rejection probability can be written as

$$p^r_{reject} = 1 - \frac{1}{N^{t^i}} = 1 - 1/\sum_{j=1}^{n} N^i_{T^j_{slot}} \quad (4)$$

Let N^{t^i} be number of requests by time t^i and $N^i_{T^j_{slot}}$ be number of requests by time t^i which target a reservation for the time slot T^j_{slot}. n is request arrivals window size in integral multiples of one slot. Note that,

$$N^{t^i} = \sum_{j=1}^{n} N^{t^i}_{T^j_{slot}} \quad (5)$$

$N^{T^i{}_{slot}}$ is number of requests which target a reservation during time slot T^i_{slot} . However, if the advance reservation distribution is assumed to be a uniform distribution, such that $a_i = \frac{1}{K}$, (K is last time slot that can be reserved in advance) for each i, then the expression for rejection probability can be simplified, and

IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 1, November 2012
ISSN (Online): 1694-0814
www.IJCSI.org

$$p_{\text{reject}}^{T_{slot}^{i}} = 1 - {}^{1}\!/_{\sum_{i=1}^{n} \gamma/k} = 1 - {}^{1}\!/_{(\gamma n/k)} = 1 - \frac{K}{\gamma n} \qquad (6)$$

It is noticeable that γ is average number of reservation request arrivals per slot, such that $E\left(N_{T_{slot}^{i}}\right) = \gamma$, where E is the expectation operator and $N_{T_{slot}^{i}}$ is number of reservation request arrivals in the slot *i*. So, reservation probability $p_{\text{reserve}}^{T_{slot}}$ will be

$$p_{\text{reserve}}^{T^{i}_{slot}} = 1 - p_{\text{reject}}^{T^{i}_{slot}} = 1 - \left(1 - \frac{k}{\gamma n}\right) = \frac{k}{\gamma n} \qquad (7)$$

For example, as see in Fig. 5, request *R1* are submitted between time slot T_{slot}^{1} and suppose that it does not have high priority. Since expressed later, *R1* should be embedded in waiting queue. All requests which are in queue will be surveyed as a group at T_{slot}^{1} (see Fig. 6).

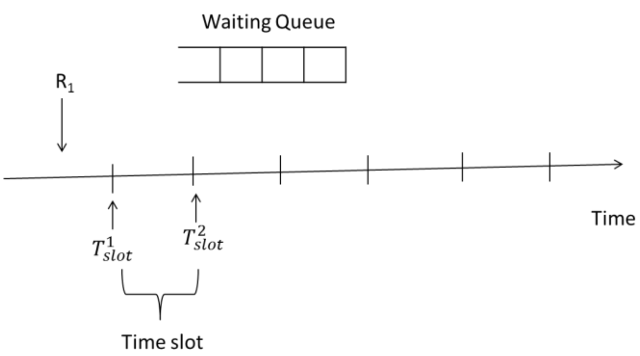

Fig. 5. Representation of a request with low priority before T_{slot}^{1}

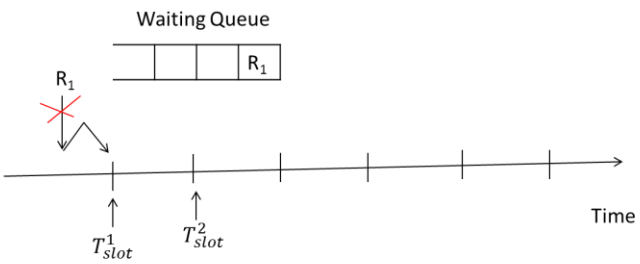

Fig. 6. Embedded low priority request into waiting queue for scheduling on T_{slot}^{1}

However, after investigating priority of requests, scheduler divides requests R in several tasks ts^{r} .

$$\mathrm{R} = \{\mathrm{ts}_{1}^{\mathrm{r}}, \mathrm{ts}_{2}^{\mathrm{r}}, \dots, \mathrm{ts}_{\mathrm{l}}^{\mathrm{r}}\} \qquad (8)$$

Each task includes one resource. Since request can demand several resources, we divide request to tasks that each task is related to one resource.

After the scheduler divides request to tasks, it applies simulated annealing algorithm for finding the providers that can make tasks' need. Simulated Annealing [9, 11,17] is a generalization of a Monte Carlo method for statistically finding the global optimum for multivariate functions. The concept originates from the way in which crystalline structures are brought to more ordered states by an annealing process of repeated heating and slowly cooling the structures. In Simulated Annealing, a system is initialized at a temperature T with some configuration whose energy is evaluated to be E. A new configuration is constructed by applying a random change, and the change in energy ΔE is computed. The new configuration is unconditionally accepted if it lowers the energy of the system. If the energy of the system is increased by the change, the new configuration is accepted with some random probability. In the original Metropolis scheme [9], the probability is given by the Boltzmann factor $e^{-\Delta E/T}$ [16]. This process is repeated sufficient times at the current temperature to sample the search space, and then the temperature is decreased. The process is repeated at the successively lower temperatures until a frozen state is achieved. This procedure allows the system to move to lower energy states, while still jumping out of local minima (especially at higher temperatures) due to the probabilistic acceptance of some upward moves. Simulated Annealing has been used in Operations Research to successfully solve a large number of optimization problems [19, 11] such as the Traveling Salesman problem and various scheduling problems [10]. Here, it is applied to the problem of request scheduling in a Grid environment.

However, after scheduler finds the providers using simulated annealing, as shown in Fig. 2, an analyst which interacts with scheduler surveys status of requests and analyzes whether request can be completed during its deadline, based on their deadline, network traffic, network bandwidth, type of matched resources (being local or global) and communication delay, etc. It is possible that scheduler finds a matched resource for a task but because of network traffic or the other reasons which be mentioned later the provider cannot complete task at specific time. So, non-complete requests will be ignored. And also the providers which cannot provide the requests' need during defined deadline will be ignored. After surveying, users and selected providers start playing a game to obtain the resources with high profit. The goal of game is maximizing user, provider and system profit, as shown in eq. 9, 10, 11.

Each user pays cost C_{i}^{r} for *i*-th resource from *r*-th request. Users trend to minimize their cost that pay for completing their tasks. And providers trend to maximize resource price. Grid scheduler's objective is to assign qualities and allocate resources to task agents, such that the system utility U^{s} is maximized. We now formulate the problem of grid scheduling optimization in computational grid as the following:

-User:

$$C^r = \sum_{i=0}^{l} (C_i^r + BW^r{}_i + delay_i{}^r)$$

Goal: $\min\left(U^r(ts_{0<i\le i}^r) = C^r - {}_p^rC\right)$ (9)

$$\textstyle\sum_{i=0}^{l} C_i^r \ll cost$$

In eq. 9, $BW^r{}_i$ is the bandwidth assigned to task *i* related to request *r* and also $delay_i{}^r$ denotes the network delay time for task *i* request *r*. C_i^r is cost of task *i* from request *r*. ${}_p^rC$ is the adaptive price that user and provider will accept the cost in the game. Let *l* be the total number of resource of request *r*. $\sum_{i=0}^{l} C_i^r$ should be less than the budget, *cost*, that user can pay for executing his job.

-Provider:

$$price^p = \sum_{i=0}^{n_p} \left(t_i^r \times price_i^r - p_f^r \times f_p{}^r - \frac{value_{in}{}^r}{y}\right)$$

$$0 \le p_f^r \le 1$$

Goal: $\max\left({}_p^rC - price^p\right)$ (10)

In eq. 10, we use $price^p$ for the total value of allocated resources to requests from provider *p*. Each provider has n_p number of resources and $price_i^r$ is related to price of resource *i* from request *r*. The duration when resource *i* is allocated to request *r* is defined with t_i^r. p_f^r is fault probability of resource *r* and $f_p{}^r$ is cost that provider *p* should pay for fault of its resource to request *r*. We suppose that each resource has initial value, $value_{in}{}^r$, that providers should pay the value for catching and allocating it to users. Each resource will work well in duration y, after this time the resource will be out of work.

-System:

$$U^s = \sum_{i=0}^{n} \omega_i^r U_i^r - \sum_{i=0}^{n} f_i^s$$

Goal: Max (U^S)
(11)

In eq. 11, ω_i^r is the priority weight assigned to task *i* from request *r* by the grid. In grid, there are *n* resources that might drop down, so we apply f_i^s for fault *i*-th resource in grid system, *s*. Grid scheduler try to find a possible task assignment that maximizes U^s subject to users' QoS constraints.

As express in follow, it is necessary a game model to obtain our goals. We will explain the game model in next part in detail. By the way, after playing game, user chooses the provider who can increase its profit. Then it informs the selected resource to the scheduler. Since it is possible that some users choose similar resources at the same time, the scheduler uses a deadline aware algorithm [12] to run maximum number of requests in time. For clarifying how the algorithm works, we explain it with an example. We suppose that there are four requests to schedule. Request R1 with deadline 3 days from 3th feb to 6th feb, and 2 days for $T_{service}$, R2 with deadline 3 days from 2th feb to 5th feb with a day for $T_{service}$, R3 with 10 days deadline from 4th feb to 14th feb with 6 days for $T_{service}$, and R4 with 4 days deadline from 4th feb to 8th feb with 2 days $T_{service}$ (seen in Fig. 7).

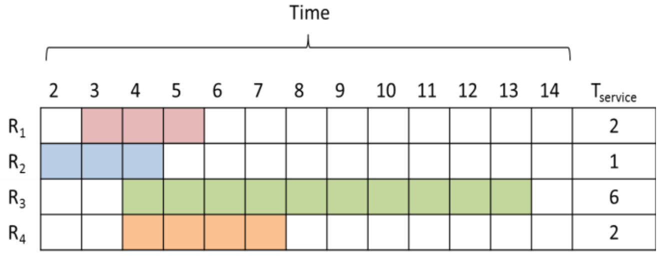

Fig. 7. Status of requests before using the proposed algorithm

As shown in Fig. 7, the four requests cannot reserve their resource expect one of requests. But if the scheduler changes their deadline by decreasing laxity of request, it is possible all of them or most of them schedule. We simulated the four requests with the algorithm and resulted that requests can reserve at the time that be shown in Fig. 8. The result shows that all of these requests can reserve their resources.

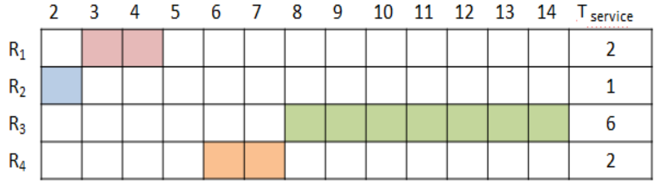

Fig. 8. Status of requests after scheduling with the proposed algorithm

Using the proposed algorithm leads to increase the number of reserved requests and consequently, increase requests' satisfaction and system utility. Until now, we explained how requests will reserve their resources before T^i_{slot}. It mentions that how requests in waiting queue will be surveyed is the same as requests with high priority before T^i_{slot}, except that in this time scheduler should survey submitted requests at T^i_{slot} and requests in waiting queue with together.

3.3 The Game Model

Consider N players (requests and providers) who bargain over a pie of size 1. Time is discrete. The game starts at $t = 0$ and ends at a predetermined deadline d^g. Each player i acts at a large but finite time grid $g^i = \{d^i_1, d^i_2, \dots, d^i_{L_i}\}$ where $d^i_k \in [0, d^g]$ for all k and $d^i_l < d^i_m$ if $l < m$.

Players play sequentially, so $g^i \cap g^j = \emptyset$ for any $i \neq j$. When player i acts at $t \in g^i$, he states some demand $a_i(t) \in [0,1]$. At every point in time all previous actions are common knowledge. For any point in time $t \in [0, d^g]$, denote the time of player i's next move by

$$\mathrm{next_i}(t) = \min\{t' \epsilon g^i | t' \geq t\},$$

and the time of player i's last move by

$$\mathrm{prev_i}(t) = \max\{t' \epsilon g^i | t' < t\}.$$

Let also $\mathrm{next_i}(t) = \min\{t' \cup_i g^i | t' > t\}$, be the time of the next move after t. The first move by player i, taken at $d^i_1 = next_i(0)$, is cost less. However, if he later (at $t > d^i_1$) changes his action, he has to pay a switching cost. If he concedes by changing his demand downwards, he pays a concession cost $C_i(t)$. If he demands more by changing his demand upwards, he pays demand costs $dm_i(t)$. We place no restriction on demand costs, except that $dm_i(t) > 0$ for any t. The assumption that demand costs, $dm_i(t)$, are strictly positive is only made for convenience. Assuming weak inequality, $dm_i(t) \geq 0$, does not change the equilibrium outcome and payoffs, but slightly complicates the analysis. We impose the following assumptions on the concession cost function: $C_i(t)$is strictly increasing in t with $C_i(0) = 0$ and $C_i(d^g) > 1$. These assumptions capture the idea that conceding is very cheap early in the process, but prohibitively expensive just before the deadline.

Finally, we specify payoffs. Denote player i's actions by $\bar{a}_i = (a_i(t))_{t \in g^i}$, all actions of all players by $\bar{a}_i = (\bar{a}_i)_{i \in N}$, and the final actions by all players by $a^* = (a_i(t^i_{L_i}))_{i \in N}$. Player i's payoffs are

$$\mathrm{u_i}(\bar{a}) = \pi_i(a^*) - \sum_{\{t \epsilon g^i - \{a^i_1\} : < a_i(t) a_i(\mathrm{prev_i}(t))\}} \mathrm{c_i}(t) - \sum_{\{t \epsilon g^i - \{d^i_1\} : a_i(t) > a_i(\mathrm{prev_i}(t))\}} \mathrm{d_i}(t)$$

(12)

where $\pi_i(a^*)$ is the usual demand game payoff

$$\pi_i(a^*) = \begin{cases} a^*_i & \text{if } \sum a^*_j \leq 1 \\ 0 & \text{if } \sum a^*_j > 1 \end{cases} \quad (13)$$

evaluated at the players' final demands.

The solution concept that we use is sub-game perfect equilibrium. While much of the analysis is carried out for arbitrary grids, our main interest lies in fine, nearly continuous grids. Thus, we define the *fineness* of a player's grid as $\varphi(g^i) = \max\{d^i_1, d^i_2 - d^i_1, d^i_3 - d^i_2, \dots, d^g - d^i_{L_i}\}$ and denote the game grid by $g = \{g^i\}^N_{i=1}$ and its fineness by $\varphi(g) = \max_i(\varphi(g^i)\}$. Our main result is a limiting result, when $\varphi(g)$ goes to zero [1].

4. Comparing the proposed algorithm with the deadline aware algorithm which uses Hill climbing

In this experiment, the Simulated Annealing Scheduler is compared to the Hill climbing Scheduler [12] to see which one leads to better estimated schedules when given the same information. The Simulated Annealing scheduler is outlined in Section III, this simple heuristic is used to speed up the search process and avoid unnecessary searches. The scheduling is done using consistent information in similar testbed.

The schedules that were generated were not actually run in this experiment since consistent machine information was required to test the schedulers, this information would have been stale by the time the runs were performed. The basic workload consists of 800 requests and it is modified into four different workloads. The best schedules and their predicted execution times are shown in Fig 9. This testbed was not able to handle 1000 requests or larger, so the largest problem that was scheduling 900 requests. The performance metrics, which are used for evaluation, are Executing time, Utilization and percentage of reserved requests.

The annealing scheduler is usually able to find a schedule having a better estimated execution time than the Hill climbing scheduler.

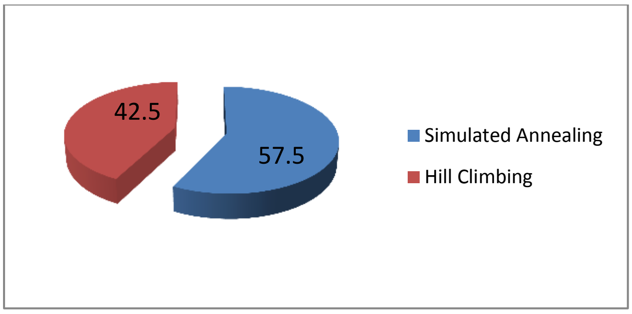

Fig 9. Estimated executing time in simulated Annealing in contrast with Hill Climbing

However, this estimated schedule depends on how accurately the Performance Model reflects reality.
As be seen in Fig. 10, we test percentage of reservation of two defined algorithm in different interval requests rate and it can be considerable that our new algorithm can reserve more requests than the old one.

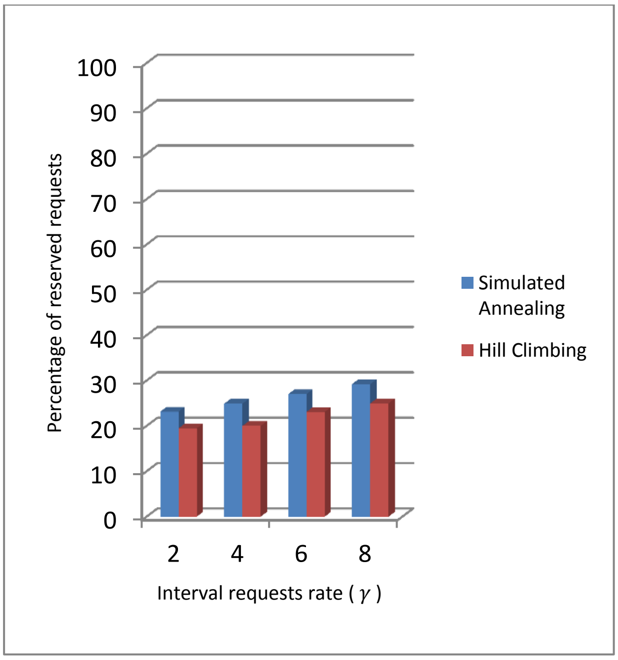

Fig 10. Percentage of reserved requests from applying two described algorithm in different interval requests rate.

As express before, the basic workload consists of 800 requests, and it is modified into four different workloads. Fig. 11 shows percentage of utilization on different interval rate. As been seen, our proposed algorithm improves system utilization in contrast the deadline aware algorithm.

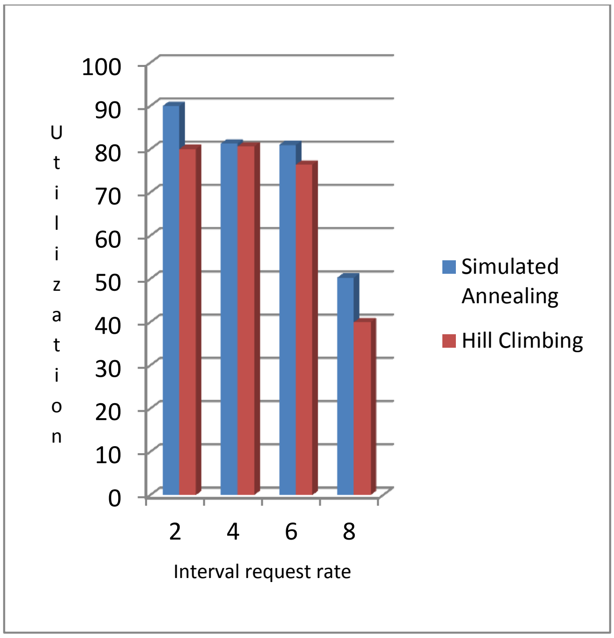

Fig. 11. Utilization rate from applying two described algorithm in different interval requests rate

As shown in results of simulation, the new proposed algorithm improve system utility, user and provider profit. This causes that both system and users meet their needs.

5. Conclusion

The Simulated Annealing scheduler generates schedules that have a better estimated execution time than those returned by the hill climbing scheduler. This is because the Simulated Annealing scheduler can avoid some of the local minimal that are not anticipated in the ordering imposed in the Hill climbing search. When the generated schedules are actually executed, the measured execution time for the Annealing Scheduler is approximately the same or just a little better than the hill climbing scheduler. Also, the measured execution time is sufficiently different from the estimated execution time that we need to reexamine the Performance Model being used. Also, using bargaining theory causes that users and resource providers contract with together on cost of resource, such that they meet their satisfactions.

References

[1]. G. Caruanaa, L. Einavb, D. Quint, “Multilateral bargaining with concession costs, Journal of Economic Theory”, Elsevior, Vol.132, No.1, Jan 2007, pp. 147-166.
[2]. B.B. Morales, “Survey: Grid Computing and Semantic Web”, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 5, May 2010.
[3]. I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the grid: enabling scalable virtual organization”, International Journal of High Performance

Computing Applications, Vol.15, No.3, 2001, pp. 200-222.

[4]. K. Rzadca, D. Trystram, and A. Wierzbicki, “Fair game-theoretic resource management in dedicated grids”, In CCGRID, 2007, pp. 343–350.

[5]. Y.K. Kwok, S. Song, and K. Hwang, “Selfish grid computing: game-theoretic modeling and NAS performance results”, In CCGRID, 2005, pp. 1143–1150.

[6]. R. Buyya, D. Abramson and J. Giddy, “Nimrod/G: An architecture of a resource management and scheduling system in a global computational grid”, In proc 4th International Conference on High Performance Computing in the Asia-Pacific Region, 2000, Vol. 1, pp. 283–289.

[7]. R. Buyya, D. Abramson, S. Venugopal, “The Grid Economy”, In proc the IEEE, March 2005, Vol. 93, NO. 3, pp. 698-714.

[8]. S. Bansal, B. Kothari, C. Hota, "Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm", International Journal of Computer Science Issues, Vol. 8, 2011, pp. 472-477.

[9]. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, “Equations of state calculations by fast computing machines”, J. Chem. Phys., Vol.21, 1953, pp.1087–1091.

[10]. Tindell, Burns, and Wellings, “Allocating hard real-time tasks: An NP-hard problem made easy”, RTSYSTS:Real-Time Systems, Vol.4, 1992.

[11]. www.wikipedia.com

[12]. s.y.rashida, M. Teshnelab, A.M. Rahmani, “An efficient negotiation based algorithm for resources advanced reservation using hill climbing in grid computing system”, In proc PDGC, 2010.

[13]. B. Blum, “A continuation method for Nash equilibria in structured games”, Journal of Artificial Intelligence Research, Vol.25, 2006, pp. 457- 502.

[14]. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, “Open Grid Services Infrastructure (OGSI)”, Global Grid Forum, 2003.

[15]. D. Ramyachitra, S. Poongodi, “Grid Meta broker selection strategies for job reservation and bidding”, International Journal of Grid Computing & Applications (IJGCA), Vol.2, No.3, Sep 2011.

[16]. R. Joshua, S. Raj, V. Vasudevan, “Beyond simulated annealing in grid scheduling”, International Journal on Computer Science and Engineering (IJCSE), Vol.3, No.3, March 2011, pp. 1312-1318.

[17]. I. H. Osman and C.N. Potts, “Simulated annealing for permutation flow-shop scheduling”, Omega 17, 1989, pp. 551-557.

[18]. S.K. Garg, S. Venugopal and R. Buyya, “A Meta-scheduler with auction based resource allocation for global grids”, unpublished.

[19]. S. Kirkpatrick, “Optimization by simulated annealing: quantitative studies”, Journal of Statistical Physics, Vol.34, No.5-6, 1984, pp.975–986.

[20]. U. K. Kumar, "A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling", International Journal of Computer Science Issues, Vol. 8, No. 1, 2011, pp. 123-129.

[21]. K. S. Chatrapati, J. U. Rekha, A.V. Babu, "Competitive Equilibrium Approach for Load Balancing a data Grid", International Journal of Computer Science Issues, Vol. 8, 2011, pp. 427-437.

[22]. I. Foster, C. Kesselman, “Globus: A Meta computing infrastructure Toolkit”, Int'l Journal of Supercomputer Applications, Vol. 11, No.2, 1997, pp.115-129.

[23]. G. Murugesan1, C. Chellappan, “An Economic-based Resource Management and Scheduling for Grid Computing Applications”, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 2, No 5, March 2010.

S. Y. Rashida received his B.S. in computer engineering from Sari Azad University, Mazandaran, in 2006, the M.S. in computer engineering from Arak Azad University, Markazi, in 2010. Now she is studying at Ph.D degree in computer engineering from IAU University, Tehran, in 2005. She is a teacher at Shirgah University in Mazandaran. Her research interests include grid computing and Fuzzy logic. She has some papers in these fields.

H. Navidi was born in 1965 and received his B.S. and M.S. degree in Pure Mathematics from Tehran University, Iran (respectively in 1989 and 1992) and the Ph.D degree in applied Mathematics from MSU in Operation Research and Game Theory in 2003 (Moscow, Russian). His current research interests include Game theory and it’s applications in Computer Science and IE, the optimization problems in LP and NLP. He is associate professor at the applied Mathematics and the Computer Science department of Shahed University (Tehran, Iran).