A Cost Greedy Price Adjustment based Job scheduling and Load Balancing in Grids

K Jairam Naik[3] , Asst Professor, Dept. of CSE, Vasavi College of Engineering, Hyderabad, India.

Dr A Jagan, , Professor, Dept. of CSE, Padmasri Dr B V Raju Institute of Technology, Hyderabad, India., jairam.524@gmail.com
Dr N Satyanarayana, Professor, Dept. of CSE, Nagole Institute of Technology & Science, Hyderabad, India., jagan.amgoth@bvrit.ac.in

Abstract - Balanced job scheduling in computational grids should take the motives for both grid users and resource providers into account. However, in computational grids most of existing studies on balanced job scheduling only address the motive for one party i.e. either the resource providers or the users. Motive for both parties are considered by very few studies on balanced job scheduling in computational grids. The accurate cost of the resource is one of the most attractive motives for users, which was not addressed. In this paper, we propose a balanced job scheduling algorithm which can optimizes the motive for both parties in computational grid. Benefits addressed by the proposed multi-objective optimization approach includes: (i) balanced scheduling increases successful execution rate of jobs. (ii) Minimizing the combined cost - motives for grid users. (iii) Reduce the fairness deviation of profits - motive for resource providers.

We proposed a heuristic based balanced job scheduling algorithm called as A Cost Greedy Price Adjusted Job scheduling and Load Balancing in Grids (BCGPA) to optimize the motives for both parties. The proposed approach could offer sufficient motives for both the parties, to stay and play in the computational grid by balancing jobs among the resources based on deadline and cost. Simulation result shows that: BCGPA algorithm is effective, Could lead to higher successful execution rate, smaller combined cost and lower fairness deviation compared with some popular algorithms in most cases.

Keywords: Computational Grid, Balanced Scheduling, Fairness Deviation, Success Rate, Candidate resources.

3 Author's Address: K Jairam Naik , Asst Professor, Dept. of CSE, Vasavi College of Engineering, Hyderabad, India.
Dr A Jagan, , Professor, Dept. of CSE, Padmasri Dr B V Raju Institute of Technology, Hyderabad, India., *jairam.524@gmail.com*
Dr N Satyanarayana, Professor, Dept. of CSE, Nagole Institute of Technology & Science, Hyderabad, India., *jagan.amgoth@bvrit.ac.in*
"Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than IJCIR must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee."
© International Journal of Computing and ICT Research 2008.
International Journal of Computing and ICT Research, ISSN 1818-1139 (Print), ISSN 1996-1065 (Online), Vol.10, Issue 1, pp.19-31, June 2016.

IJCIR Reference Format:

K Jairam Naik, A Jagan and N Satyanarayana. A Cost Greedy Price Adjustment based Job scheduling and Load Balancing in Grids, Vol. 10, Issue.1pp 19-31. http://www.ijcir.org/volume 10-issue 1/article 3.pdf.

1. Introduction

Recently, grid computing has been moving towards a pay-as-you-go model, in which resource providers expect an economic compensation for the computational services offered to the users. From the perspective of grid users, there are two major concerns in job scheduling: Exact cost incurred and Successful execution rate of jobs (SERoJ) [L. Xiao et al. 2008, H. Izakian et al 2010]. These two concerns are important since, grid users generally hope to successfully complete their submitted jobs in expected time and cost. On contrary, if jobs frequently miss their deadlines or the cost incurred is high, users tend to lose interest in the grid system and may finally leave the system. Therefore, increasing SERoJ is a motive for grid users, and so is reducing cost [I. Foster et al 2002, S.K. Garg et al 2011], which shall be considered in job scheduling.
On the other hand, from the perspective of grid resource providers, a major concern in job scheduling is *profit fairness*. Generally the providers hope to have equal opportunities to offer their resources and gain fair profits according to their properties. Therefore, increasing profit fairness, or equivalently reducing the *fairness deviation of profits*, is desirable from the perspective of resource providers [C. Xu et al 1997].

Balanced job scheduling in computational grids should consider the incentives for both grid users and resource providers. However most of the existing studies of balanced job scheduling in computing grid only address the motive for one party i.e. either the resource providers or the users. The accurate cost of the resource is one of the most attractive motives for users, which was not addressed. In this chapter, we propose a multi-objective Heuristic algorithm (BCGPA) for job scheduling in computational grid which addresses the major motives for both i.e.

1. Maximizing the SERoJ and minimizing the combined cost (motives for grid users),
2. Minimizing the FDoP (motive for resource providers).

The rest of this paper is organized as follows. Section 2 gives a review of related work on job scheduling in grid environment. Section 3 presents the system model we used. We put forward the optimization objectives for users and resource providers and propose a heuristic scheduling algorithm called BCGPA algorithm in Section 4. Simulations, results and analyses are given in Section 5 and concluded this paper in Section 6.

2. Related work

Many literatures have studied the job scheduling problem in grid environment. In general,

existing grid scheduling algorithms can be classified into two types based on *scheduling time*, i.e. Immediate mode scheduling and b) Batch mode scheduling [F. Xhafa et al. 2010] In immediate mode, a job is scheduled as soon as it arrives at the scheduler. While in batch mode, jobs are collected into a group and mapped to resources at the End of a fixed scheduling interval. The immediate mode is suitable for the situation of low arrival rate, while batch scheduling can take better advantage of job and resource information [M. Maheswaran et al. 1999]. Batch mode scheduling is commonly adopted in literatures which investigate job scheduling problem in grid environment [S. K Garg et al. 2010, S. K Garg et al. 2011 & J. Yu et al 2005].
In [K.Z. Gkoutioudi et al 2012] classified the existing heuristic based task allocation methods into several groups, such as economic heuristic, population-based heuristic, and so on. The algorithms that mainly focus on minimizing *makespan*, which is the finishing time of the latest job, without considering the budget requirement of users. In [Heyang Xu et al 2015], the authors formulate a nonlinear programming model to maximize the aggregate utilities of all grid users and propose an optimization based resource pricing algorithm. The grid user utility is defined as a function of resource units allocated to the grid user. In [Heyang Xu et al 2015], CGPA sets high priority to the jobs with few candidate resources and maps a job to the candidate with lowest cost. It adopts price-adjusting algorithm to adjust the price of candidate resources to avoid the fairness deviation becoming worse. But was not more efficient to improve SERoJ and the tendency to fail the user job is more.

In [S. K garg et al. 2011], the authors study how to decrease the aggregate cost of all jobs under some QoS requirements. They propose a constrained linear programming model and a linear programming genetic algorithm (LPGA) to minimize the combined spending of all users. Although all these approaches improve the performance, they ignore the motives for resource providers, for example, the FDoP which is considered in this paper. Huang et al. [J. Yu et al 2005] propose a series of motive-based algorithms to maximize the successful execution rate of jobs and to minimize fairness deviation among resource providers. Also, their research is based on immediate mode scheduling which is different from our work. They extend the study [L. Xiao et al. 2008], by further taking resource utilization rate and load balancing level into consideration. We tried to optimize an important motive, *combined cost*, which is most attractive to grid users however, all the above studies ignore considering job execution cost. If the cost of executing jobs is too high, grid users will lose interest in the grid system.

3. System model

Grid users are the active entities in computational grid among resource providers and grid scheduler. Grid users submit jobs to the grid scheduler with certain QoS requirements. Resource providers can offer their resources to execute jobs submitted by grid users via the scheduler. The grid scheduler is responsible for mapping the jobs submitted by grid users to the provider's resources. Grid users and resource providers interact through the grid scheduler. Here, some important QoS factors which users most concern about are budget, deadline, and the number of required successive CPU's [H. Izakian et al 2010, S. K Garg et al. 2011]. Each job consists of several tasks and each task requires one processor. Tasks of the same job should be executed concurrently on the same resource [S. K Garg et al. 2010]. The grid scheduler collects jobs submitted by grid users, gathers resource information (such as available CPUs, successive rate of the CPU, the price of using one CPU per unit time and computational speed of its CPUs) from

resource providers and sends each job to a resource to be solved in a batch mode at the end of the scheduling interval. Then, resource providers execute jobs and charge for executing them. Generally, the scenario of job scheduling is as follows: each user in the computational grid may submit one or more jobs, each job can be composed of several tasks; the grid scheduler is responsible for mapping each job to a resource to be solved.

4. Problem formation

Suppose that the computational grid consists of m ($m \geq 1$) resources; $R = (R_0, R_1 \ldots R_{m-1})$. Let R_j (a_j, p_j, s_j,, e_j) represents the resource information that meta-scheduler gathers from resource provider j ($j \in \{0, 1, \ldots, m-1\}$), where a_j is the number of available CPUs of resource R_j, p_j is price (equals to the cost of using a single CPU per second) of a CPU, s_j is the CPU speed, in million instructions per second (MIPS) and e_j is the resource success rate. Jobs that meta-scheduler collects during the scheduling interval T, is denote by $J = (J_0, J_1, \ldots, J_{n-1})$ where ($n \geq 1$). J_i (t_i, b_i, d_i, l_i, L_{ik}) represents information of the job i ($i \in \{0, 1, \ldots, n-1\}$). Where, t_i is the submitted time of job J_i ($0 \leq t_i \leq T$) and bi is the budget constraint, which means that the cost of executing job J_i must not exceed b_i, di represents the deadline by which the user desires the job to be completed. The number of tasks that job J_i contains is l_i ($l_i \geq 1$) and L_{ik} is the length of task k ($k \in \{0, 1, \ldots, l_i - 1\}$) in job J_i in terms of millions of instructions (MI).

4.1. Objectives for providers and users

A fundamental optimization objective for resource providers is the fairness of obtained profits. It means that a resource provider could obtain the same share of profit as the capacity that it invests to the system. In computational grid, each resource provider should have equal opportunity to offer its resource and gain a fair profit according to its capacity. Fairness of obtained profits is attractive to both providers with low capacity and those with high capacity. Therefore, we take *minimizing FDoP* (denoted by σ, shown in Eq. (3)) as the objective for resource providers, which is adopted in [L. Xiao et al. 2008] as well.

Let we first state several relevant definitions prior to describing Eq. (3). Each resource R_j bills for the jobs which are successfully executed by them. We use $profit_j$ to denotes the obtained profit of resource provider j, which is given by

$$Profit_j = \sum_{i=0}^{n-1} x_{ij} * \left(\sum_{k=0}^{l_i-1} \left(\frac{L_{ik}}{s_j} * p_i(1 + e_j) \right) \right) \quad (1)$$

Where, $x_{ij} = 1$, when job J_i is allocated to resource R_j; otherwise, $x_{ij} = 0$.

and $e_i = \frac{N_s}{N_s + N_f}$

Where,

Ns and N_f represents the number of jobs successes, fails the execution respectively in given deadline by the resource.

The obtained fairness of profit of resource provider j is denoted by μ_j, which is defined as the profit obtained of resource provider j divided by its total available CPU capacity, as given in Eq.

(2)

$$\mu_j = \frac{Prifit_j}{a_j * s_j} * (1 + e_j) \quad (2)$$

The FDoP for all resource providers is the standard deviation of their fairness of obtaining profit. Therefore, the optimization objective of resource providers is given by Eq. (3). Min σ = *std-dev* ($\mu_0, \mu_1, \ldots, \mu_{m-1}$)

$$= \sqrt{\frac{1}{m}\sum_{j=0}^{m-1}(\bar{\mu} - \mu_j)^2}, \quad for\ all\ i \quad (3)$$

Many objectives could be defined for grid users, but what attracts them most is that their jobs could be successfully executed at low cost and in deadline. If the cost of executing jobs is too high or jobs frequently miss their deadlines, users will lose interest in the grid system. Consequently, we endeavor to *minimize the combined cost C* (shown in Eq. (4)), which is the sum of cost of all users, and *maximize the SERoJ* (denoted by θ, shown in Eq. (5)) under the constraints of budget and deadline.

$$Min\ C = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} x_{ij} * \left(\sum_{k=0}^{l_i-1}\left(\frac{L_i}{S_j*(1+f_j)} * p_i\right)\right) \quad (4)$$

$$Max\ \theta = \frac{\sum_{i=0}^{n-1}\Psi_i}{n} \quad (5)$$

Subject to

(i) $x_{ij} \in \{0, 1\}, \forall_i \in \{0, 1, \ldots, n\text{-}1\}, j \in \{0, 1, \ldots, m\text{-}1\}$.

(ii) $\sum_{j=1}^{m} x_{ij} \le 1, \forall_i \in \{0, 1, \ldots, n\text{-}1\}$.

(iii) $\sum_{j=1}^{m} x_{ij} . l_i \le a_j, \forall_j \in \{0, 1, \ldots, m-1\}$.

(iv) $\forall_i \in \{0,1,\ldots,n\text{-}1\}$, if $x_{ij} = 1$, then $(T\text{-}t_i) + Max_{0\le k\le li-1}\left\{\frac{L_{ik}}{S_j*(1+e_j)}\right\} \le d_i$.

(v) $\forall_i \in \{0,1,\ldots,n\text{-}1\}$,if $x_{ij} = 1$,then $\sum_{k=0}^{l_i-1}\left(\frac{L_{ik}}{S_j} * p_j \le b_i\right)$.

(vi) $\forall_i \in \{0,1,\ldots,n\text{-}1\}$,for $x_{ij} = 1$, $e_i = N_s / (N_s + N_f)$ $e_i = N_s / (N_s + N_f)$; If job i is completed before its deadline, then $N_s = N_s + 1$; otherwise N_s value remains unchanged.

If job i is completed before its deadline, then $\Psi_i = 1$; otherwise, $\Psi_i = 0$. The first constraint defines the feasible range of variable x_{ij}. The (ii) Constraint ensures that a job should be assigned to no more than one resource. Constraint (iii) specifies that the total allocated CPUs on a resource should not exceed its available number. Actually, this constraint is a set of *m* constraints because for each resource there is a constraint of total allocated CPUs on it. In constraint (iv), $(T - t_i)$ means the waiting time of J_i; L_{ik} is the length of the k^{th} task of J_i; $\frac{L_{ik}}{S_j*(1+e_j)}$ is the execution time of the k^{th} task of J_i; $Max_{0\le k\le li-1}\left\{\frac{L_{ik}}{S_j*(1+e_j)}\right\}$ is execution time of J_i, Which equals to the maximum execution time among all its tasks. Therefore (iv) constraint means that, if J_i is mapped to resource R_j, then the sum of its waiting time and execution time must be less than its

deadline. Constraint (v) indicates that, if J_i is mapped to the resource R_j, then the cost of executing the job must be less than its budget. Constraint (vi) indicates, if greater the N_f value, increases the cost. Resource with higher cost has more chances to miss the deadline. Such resources are preferred less by the user and recommends price balancing.

In this paper, we deal with the job scheduling in computational grids as a multi-objective optimization problem, i.e., *minimizing the FDoP* (Eq. (3)), *minimizing the combined cost* (Eq. (4)) and *maximizing the SERoJ* (Eq.(5)). It can be noted that, when substituting Eq.(1)and (2) into Eq.(3), the first optimization objective, Eq. (3), are not linear concerning x_{ij}. Therefore, the proposed optimization problem is a nonlinear one. Moreover, from the first two constraints we can see that this problem is a combinatorial optimization problem, which has been proved to be an NP-hard problem. Due to the NP-hardness of the grid job scheduling, the approximation algorithms that suffice to find a near optimal solution are more promising. Therefore, we propose a heuristic algorithm which is presented in the next section.

4.2. Proposed BCGPA algorithm

Lots of algorithms have been developed for scheduling jobs in a computational grid. The majority of them aim to minimize the job completion time [S.K. Garg et al 2010], optimize load balance [Y. Lee, S et al 2011], or improve resource utilization [H. Izakian et al 2010]. Heuristic algorithms have shown to be useful approaches for solving varieties of hard-to-solve combinatorial and multi-objective optimization problems. Therefore, in this paper, we propose a heuristic algorithm named as *balanced cost-greedy price-adjusting* (BCGPA) *algorithm*, as shown in Algorithm 1.

Definition 1. *Candidate Resource: For job J_i, if a resource R_j can complete J_i before its deadline and the cost is less than its budget, then R_j is a candidate resource of J_i. If there is more than one candidate resource for a job, the resource with the lowest cost will be selected to handle the job.*

Algorithm 6.1: Balanced Cost-greedy Price-adjusting (BCGPA) algorithm

Input: *set of jobs (submission time, budget, deadline constraints, number of tasks, length of each task) and set of resources (num of available CPUs, CPU capacity, price of CPU, Resource success Rate).*
Output: *Mapping of jobs to resources.*

```
1   while(there is unmapped job)
2     foreach unmapped job Ji do
3       foreach resource Ri do
4         if Ri can satisfy constraints of Ji then
5           add Ri to candidate resource set Si of job Ji ;
6         end if
7       end foreach
8     if Si = Ω then
9       add job Ji to unsuccessful mapping set U;
10      change Ji 's state to failed allocation;
```

```
11      else if |Si | = 1 then
12       map job Ji to the candidate resource in Si ;
13       change the number of available CPUs of resource Ri ;
14       change i's state to successful allocation
15       end if
16     end if
17    end foreach find the candidate resource set Si (|Si| > 1)
       which contains minimal elements:
18  find the resource Ri (from Si ) which minimizes the cost of Ji;
19  map job Ji to resource Ri ;
20  change the number of available CPUs of resource Ri ;
21  change Ji 's state to successful allocation;
22  AdjustResourcePrice (Si );
23  end while
```

BCGPA algorithm is an iterative greedy approach in which, initially all jobs are set in *unmapped* state. Then, BCGPA executes the following process(lines 2–24, Algorithm 1) iteratively: First, for each unmapped

Algorithm 2: AdjustResourcePrice (Si)
Input: *candidate resource set Si ; α, β which are coefficients of increasing and decreasing price, respectively.*

job J_i, BCGPA orderly checks m resources R_0, R_1, . . . , R_{m-1} whose subscripts are randomly generated to find the candidate resources of J_i and these candidate resources compose J_i's candidate resource set S_i (lines 3-7, Algo. 1). In this process, if certain S_i is an empty set, then job J_i cannot be successfully mapped and the state of J_i is changed to *failed allocation* (lines 8–10, Algorithm 1); if certain S_i contains only one resource R_j, then BCGPA maps job J_i to resource R_j and the state of J_i is changed to *successful allocation* (lines 11–16, Algo 1).

Second, jobs for whose candidate resource set contains more than one candidate resource, BCGPA selects the job J_i whose candidate resource set contains minimal number of candidate resources and maps J_i to the candidate resource R_j in S_i which can minimize J_i's cost (lines 19–23, Algorithm 1). Then, BCGPA invokes the function of *AdjustResourcePrice* (S_i), shown in Algorithm 2, to adjust the price of candidate resources in S_i (line 24, Algorithm 1). Continue the steps mentioned above until there is no job in *unmapped* state.

Output: *calculated new price.*

```
1    foreach resource Rj in Si do
2        if Rj is the selected resource then
3            Rj .price = α × Rj .price;
4        Else
5            Rj .price = β × Rj .price;
6        end if
7    end foreach
```

As shown in Algorithm 2, for each resource R_j in S_i, if R_j is selected to execute job J_i, then the price of R_j will increase by α, which is a decimal slightly greater than 1, to avoid R_j always being selected in the following choice (lines 2–3, Algorithm 2). Otherwise, the price of R_j will decrease by β, which is a decimal slightly less than 1, to avoid R_j never being selected in the following choice (lines 4–5, Algorithm 2).

5. Simulation configurations

5.1 Simulation configurations

Assume all the users, providers and jobs are independent with characteristics shown in Table 6.1 and Table 6.2.

The initial price of all resources varies between 4G$ and 5G$ with average value of 4.5 G$. The budget allocated to each job set J_i is according to Eq. (6), it is the value of multiplying the number of jobs J_i by 500 and varied by 20%. The value 50 is obtained by multiplying the average estimated execution time (10 s) of a task by the maximal resource price (5 G$). The X is an integer variable which is uniformly distributed within the range [0, 100]. Y is an integer variable with the value of 1 or 0. Deadline requirement allocated to each job J_i is set the sum of estimated average run time and interval with 10% variation.

$$b_i = 50.l_i + X \,.\, pow\,(-1, y). \qquad (6)$$

In experiment 1, we investigate the impact of two parameters, α and β, which are used in our method. In experiment 2, we compare the proposed algorithm with other four representative algorithms to investigate the efficiency of the proposed algorithm. In experiment 3, we investigate the efficiency of the proposed algorithm under real workload traces.

Table 6.1: Grid resource characteristics

Number of Resources	1 - 4
Number of Machines	1 - 25
Number of PEs	100
PE ratings	10 or 50 MIPS
Bandwidth	1000 - 5000 B/S

5.2. Experime

Table 6.2: Scheduling parameters and their values

No. of providers	1 - 4
No. of users	200 –1500
No. of user job	200 - 500
Job length	0 – 50,000 MI
Resource price	4G$ - 5G$
I/O file size	100+(10–40%) MB

nt 1

This experiment investigates the impact of α and β on scheduling performance of our method. α and β are used in price adjusting algorithm, as shown in Algorithm 2, and they are price increasing and decreasing coefficients, respectively. We design price adjusting algorithm to improve the fairness deviation of all resource providers. In a real market, the price of certain commodity does not fluctuate largely after a deal. So, an important principle in our price adjusting algorithm is that a big change of price is inadvisable. Therefore, the pair of α and β, which change resource's price not so much, is what we want. We use average price of all resources as the metric to determine a suitable pair of values of α and β. Experiment shows the results obtained based on different pair of α and β after scheduling 100 jobs. When α = β = 1, the price of all resources has no change though the scheduling interval and the average resource is 4.47. From Experiment, it can be seen that the average price of all resources rapidly drops with β slightly decreasing. When α = 1.15 and β = 0.995, the average price is 4.19, which is the closest to 4.47 out of the results under four pairs of α and β. Thus, in experiment 2, α and β are set to 1.15 and 0.995, respectively.

5.3. Experiment 2

We compare the efficiency of Balanced cost-greedy price-adjusting (BCGPA) algorithm with other four algorithms adopted in related researches to investigate it i.e., min-min cost time trade-off (MinCTT) algorithm], modified min cost (MMC) algorithm, linear programming based genetic algorithm (LPGA) and CGPA, with varied system load ranging from 0.41 to 0.99. The system load is calculated by Eq. (7).

$$System_{load} = \frac{\sum_{i=0}^{n-1}\sum_{k=0}^{l_i-1} L_{ik}}{interval*\sum_{j=0}^{m-1} a_j*s_j*(1+e_j)} \quad (7)$$

Submission time and then dispatches each job to a candidate resource. If a resource can satisfy a job's QoS requirements, the resource is a candidate resource for the job. If a job has no candidate resource, the job fails to be scheduled and if there are many candidate resources, the one with the lowest cost will be selected. MinCTT algorithm maps each job to the resource with minimizing the cost metric, which is defined as the trade-off between response time and execution cost. MMC algorithm handles the jobs in two ways: for jobs with one candidate resource, their mapping is frozen and the jobs with more than one candidate resource will be mapped to the most economical resource. LPGA use genetic algorithm to generate a solution near to optimal solution, by seeding the approximate solutions from MMC. In the following parts, we compare these algorithms with our approach on three considered factors: SERoJ, combined cost and FDoP.

5.3.1. Successful execution rate of jobs (SERoJ)

A successful job execution means that a job is completed before its deadline. With higher successful execution rate, the grids provide more interest the users have in performing their jobs. The SERoJ is calculated according to Eq. (5). As shown in experiment and Fig. 1, the SERoJ obtained by all algorithms decreases with the increase of system load. LPGA has the highest successful execution rate, and the result of our proposed algorithm, BCGPA, is very close to that

of LPGA and outperforms the other three methods. That is because BCGPA preferably maps the job with tight QoS constraints (lines 11–21, Algorithm 1). Thus, all jobs will be more likely to be mapped to a resource to be completed before its deadline.

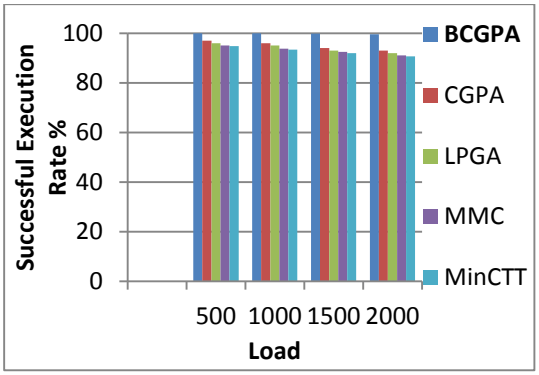

Fig. 1. Successful execution rate of jobs with different system loads.

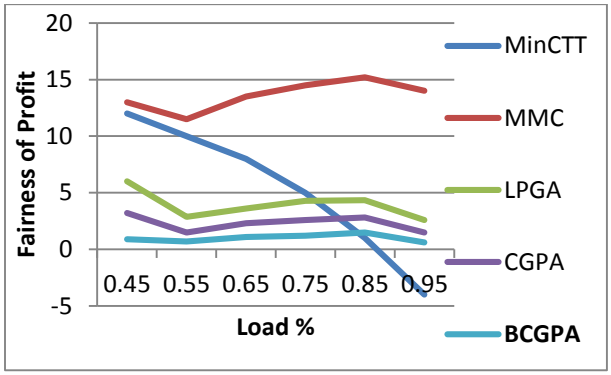

Fig. 2. Fairness deviation obtained by varied Algorithms with varied load.

5.3.2. Combined cost

According to Eq. (4) Combined cost is calculated, which is the total cost of all successfully mapped jobs. The comparison results of the proposed BCGPA algorithm with other four algorithms are shown. BCGPA algorithm obtains lower combined cost than that obtained by other algorithms in all varied system load situations because BCGPA tries to map a job to the candidate resource with the lowest cost (lines 20–21, Algorithm 1). Although FCFS can obtain relatively low combined cost, its job failure rate is higher than other methods. We use *cost saving ratio* (*CSR*) to elaborate how much combined cost that users can save by using the proposed BCGPA algorithm. The *cost saving ratio* is defined as the difference of combined cost between selected algorithm and the proposed BCGPA algorithm divided by the combined cost obtained by BCGPA algorithm.

For example, the cost saving ratio of MMC algorithm is calculated by Eq. (8). A positive value of certain CSR indicates that BCGPA algorithm obtains lower combined cost than the corresponding algorithm; otherwise, BCGPA algorithm brings higher combined cost than the corresponding algorithm. For example, when system load is 0.45, the value CSR MMC is about 13%, which means that the proposed BCGPA algorithm can save combined cost by 13% compared with that obtained by MMC algorithm. From Fig. 2, it can be seen that the cost saving ratio obtained by the other four algorithms is bigger than zero in most cases except the MinCTT algorithm with system load 0.95. That is because when system load is 0.95, many jobs fail to be mapped to a resource and these jobs produce no execution cost in MinCTT algorithm.

$$CSR_{CGPA} = \frac{\text{combined cost obtained by CGPA} - \text{combined cost obtained by BCGPA}}{\text{combined cost obtained by BGPA}} \qquad (8)$$

5.3.3. Fairness deviation of profits

The fairness of grid can be expressed in the way that all resource providers have equal opportunities to offer their resources and can obtain fair profits according to their resource capacities [L. Xiao et al. 2008]. FDoP (defined by Eq. (3)) indicates the dispersion of all resource providers' profits. The smaller the value of fairness deviation is, the fairer the providers' profits are. Fig. 3 show the results obtained by different algorithms under different system loads. It can be seen that our approach obtains lower fairness deviation than that of other four methods. The reason is that to adjust the price of all candidate resources for each job mapping the proposed CGPA algorithm uses price adjusting algorithm. If a candidate resource obtains a job, then the CGPA algorithm slightly increases its price to prevent it from always being selected in following mapping (lines 2–3, Algorithm 2); otherwise, the CGPA algorithm slightly decreases its price to avoid never being selected in following mapping (lines 4–5, Algorithm 2).
Thus, the proposed CGPA algorithm can get better fairness deviation. Also, from Fig. 3, we can find that the fairness deviation obtained by all approaches declines with the increasing of system load. The reason lies in that, with the increasing number of jobs, more and more resources get jobs near to their processing capacities, which leads to the decrease of fairness deviation.

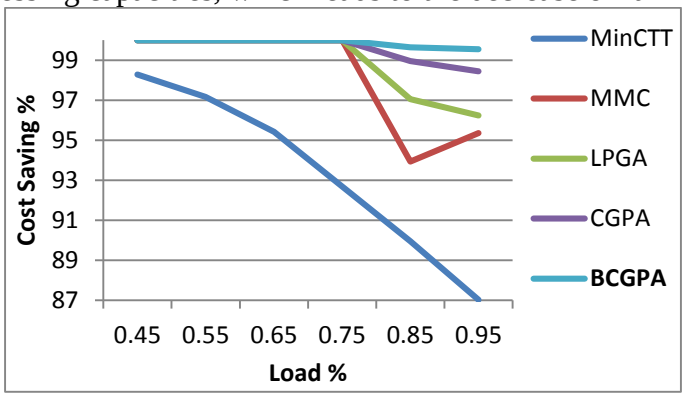

Fig. 2. Cost saving rate obtained by the four algorithms.

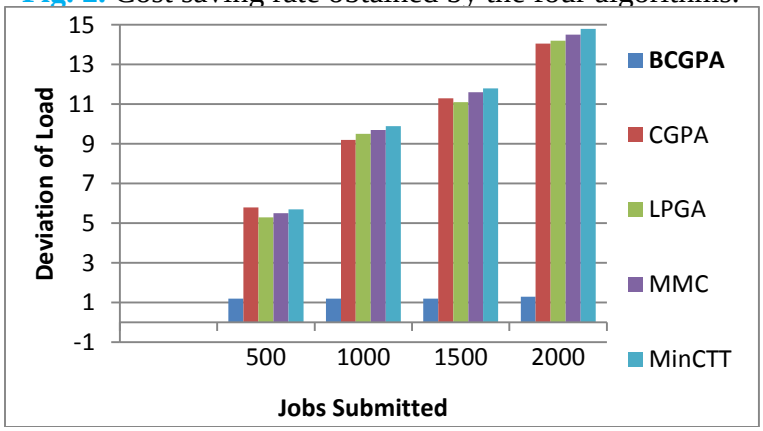

Fig. 4. Fairness deviation of Load**.**

6. Conclusion

The two important entities in computational grid are Grid users and resource providers and they have different objectives of interest and make autonomous scheduling decisions. This makes the problem of job scheduling more complex than ever before in commodity market-like grid.

In this paper, job scheduling is formulated in computational grid as a multi-objective optimization problem. It is called as a well-known NP-hard problem due to its combination property. So a heuristic, *balanced cost-greedy price-adjusting* (BCGPA) *algorithm* is proposed. In each mapping, BCGPA algorithm sets high priority to the jobs with few candidate resources and maps a job to the candidate with the lowest cost and then adopts a price-adjusting algorithm to adjust the price of candidate resources to avoid the fairness deviation be-coming worse. The simulation results clearly illustrate that our approach is efficient and could lead to a higher successful execution rate, lower combined cost and better fairness of providers' profits than other compared algorithms in most cases.

References

C. Xu, F.C. Lau, Load Balancing in Parallel Computers: Theory and Practice, Kluwer, Boston, MA, 1997.

F. Xhafa, Computational models and heuristic methods for grid scheduling problems, FGCS. 26 (4) (2010) 608–621.

Heyang Xu, Bo Yang, An incentive-based heuristic job scheduling algorithm for utility grids, Future Generation Computer Systems 49 (2015) 1–7.

H. Izakian, An auction method for resource allocation in computational grids, FGCS. 26 (2) (2010) 228–235.

I. Foster, C. Kesselman, J.M. Nick, S. Tuecke, Grid services for distributed sys. integration, Computer 35 (6) (2002) 37–46.

J. Yu, R. Buyya, C.K. Tham, Cost-based scheduling of scientific workflow applications on utility grids, in: Proceedings of the First Int Conf on e-Science and Grid Computing, 2005, pp. 140–147.

K.Z. Gkoutioudi, H.D. Karatza, Multi-criteria job scheduling in grid using an accelerated genetic algorithm, J. Grid Comput. 10 (2012) 311–323.

L. Xiao, incentive-based scheduling for market-like computational grids, IEEE Trans. PDS. 19 (7) (2008) 903–913.

M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R.F. Freund, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, J. PDC. 59 (2) (1999) 107–131.

S.K. Garg, P. Konugurthi, R. Buyya, A linear prog-driven genetic algo. for meta-scheduling on utility grids, Int. J. PEDS. 26 (6) (2011) 493–517.

S.K. Garg, R. Buyya, H.J. Siegel, Time and cost trade-off management for scheduling parallel app. on utility grids, FGCS. 26 (8) (2010) 1344–1355.

Y. Lee, S. Leu, R. Chang, Improving job scheduling algorithms in a grid environment, Future Gener. Comput. Syst. 27 (8) (2011) 991–998.