CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.2984

Cost minimization in utility computing systems

Satish Penmatsa [1] and Anthony T. Chronopoulos [2,*,†]

[1] *Department of Math. & Computer Science, University of Maryland Eastern Shore, Princess Anne, MD 21853, USA*
[2] *Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA*

SUMMARY

Utility computing is a form of computer service whereby the company providing the service charges the users for using the system resources. In this paper, we present system-optimal and user-optimal price-based job allocation schemes for utility computing systems whose objective is to minimize the cost for the users. The system-optimal scheme provides an allocation of jobs to the computing resources that minimizes the overall cost for executing all the jobs in the system. The user-optimal scheme provides an allocation that minimizes the cost for individual users in the system for providing fairness. The system-optimal scheme is formulated as a constraint minimization problem, and the user-optimal scheme is formulated as a non-cooperative game. The prices charged by the computing resource owners for executing the users jobs are obtained using a pricing model based on a non-cooperative bargaining game theory framework. The performance of the studied job allocation schemes is evaluated using simulations with various system loads and configurations. Copyright © 2012 John Wiley & Sons, Ltd.

Received 6 October 2011; Revised 9 November 2012; Accepted 14 November 2012

KEY WORDS: job allocation; utility computing; cost minimization; fairness

1. INTRODUCTION

Utility computing is the packaging of computing resources, such as computation, communication, and storage, as a metered service. These utility computing systems help users with very large computation requirements by avoiding the cost and delays that would result from physically acquiring a large number of computers. Users pay some price (which is economic in nature) for the resources they use. Because these systems consist of heterogeneous computing resources, poor allocation of jobs to the resources in these systems may result in higher (economic) payments by the users for the execution of their jobs. Also, in a distributed computing system, some computers (computing resources) may be heavily loaded than the others. Because jobs allocated to overloaded computers may take a longer time to complete than the jobs of similar size allocated to lightly loaded computers, some users may have to pay a higher price than the others for the execution of their jobs when using utility computing systems, which charge prices based on the time taken to complete the jobs (or based on the wall-clock time). Hence, efficient job allocation schemes that minimize the cost (or payments) for executing the users jobs in utility computing systems are essential.

The resources of utility computing systems (e.g., utility systems based on computational grids) may belong to various owners. Because a computationally intensive job or an application usually requires the resources from more than one owner, the utility systems should be able to assign the jobs from various users to the different owned resources efficiently. To obtain the prices charged by the resource owners, we use a pricing model based on bargaining game theory proposed in [1].

*Correspondence to: Anthony T. Chronopoulos, Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA.
†E-mail: atc@cs.utsa.edu

The software agents of the user and the resource owner play an incomplete information, alternating-offer and non-cooperative bargaining game to obtain an agreed price-per-unit-resource.

1.1. Contribution and related work

In this paper, we present system-optimal and user-optimal price-based job allocation schemes for utility computing systems whose objective is to minimize the cost (or price) for the users. The price-based global-optimal (system-optimal) scheme (GOSP) provides an allocation of jobs to the computing resources that minimizes the overall cost for executing all the jobs in the system. The price-based user-optimal scheme (NASHP) provides an allocation that minimizes the cost for individual users in the system. The system-optimal scheme is formulated as a constraint minimization problem, and the user-optimal scheme is formulated as a non-cooperative game. The objective of NASHP is to provide *fairness* to all the users, that is, all the users should have to pay approximately the same price, independent of the computers allocated for the execution of their jobs (of approximately the same size). Fairness is very important in modern distributed systems such as the utility computing systems, grid computing systems, and cloud computing systems.

Preliminary work on GOSP and NASHP can be found in [2]. In this paper, more experimental results are presented evaluating the performance of the proposed schemes. The proportional job allocation scheme (PROP) [3] is also implemented for comparison purposes. New results presented are the following: variation of expected price with system utilization for various price vectors based on the allocation of NASHP; effect of system utilization on the expected prices provided by GOSP, NASHP, and PROP; effect of system utilization on the fairness indices obtained by GOSP, NASHP, and PROP; variation of expected price with speed skewness for various price vectors based on the allocation of NASHP; effect of speed skewness on the expected prices provided by GOSP, NASHP, and PROP; effect of speed skewness on the fairness indices obtained by GOSP, NASHP, and PROP; variation of expected price with system size for various price vectors based on the allocation of GOSP and NASHP; effect of system size on the expected prices provided by GOSP, NASHP, and PROP; effect of system size on the fairness indices obtained by GOSP, NASHP, and PROP. The definition and details of a non-cooperative game are provided on the basis of which NASHP scheme is derived. The BEST-REPLY' and NASHP job allocation algorithms are presented. Proofs for Theorems 3 and 4 (for computing the best reply of a user and for proving the correctness of 'BEST-REPLY' algorithm) are provided.

We assume all jobs are of the same size in terms of the computation time required to be executed by the slowest computer. In the case where there exist jobs of unequal sizes, we assume that they are divisible. We thus assume that they are divided into jobs of the same size before they are scheduled for execution.

Job allocation/load balancing in conventional distributed computer systems has been studied extensively ([4–13] and references therein). Most of these studies minimized the mean response time (average execution time) of jobs in the distributed system. Economic pricing was not taken into account. In grid and utility computing systems, because the computing resources may be managed by different owners who charge different prices, job allocation schemes, which minimize the expected price (or cost) for executing the users' jobs are also essential.

Resource management and scheduling schemes based on pricing for clusters or market-like grid systems based on different economic and system models were studied in ([14–19] and references therein). Price-based static job allocation for single/multi-class (single/multi-user) job distributed systems for providing a system-optimal solution has been studied in [1, 20–22]. Dynamic price-based job allocation schemes for single/multi-class job grid systems have been studied in [23, 24]. In [25], a price-based single-server many-computer scheduling scheme is proposed for a grid system model with a single server which accepts each user's jobs and assigns them to the computers. However, in the previously mentioned studies, fairness to the users is not considered. Using non-cooperative game theory, a price-based job allocation scheme for multi-class job grid systems has been studied in [26].

Load balancing in distributed systems based on economic game theory was studied in ([27–29] and references therein). However, these previous studies are for single-class job grid systems, and

DOI: 10.1002/cpe

pricing was not included in the models. Many economic models similar to [1] for resource allocation in distributed systems have been studied. For example, [30, 31] and references therein.

Data replication across multiple servers based on game theory was studied in [32]. Task scheduling on multi-core processors for simultaneous optimization of performance and energy based on game theory was studied in [33]. A replica placement technique based on cooperative game theory was studied in [34]. Energy-aware scheduling in computational grids based on genetic algorithms was studied in [35, 36]. Workload-aware reliability evaluation model for grid systems was studied in [37].

Game-theoretic studies for resource allocation in cloud computing systems and for cloud services also exist. In [38], fixed and market pricing for cloud services were studied. A game-theoretic approach was considered in [39] for resource and revenue sharing with coalition formation of cloud providers.

Several studies on resource allocation for specialized distributed systems (e.g., grid systems) based on auctions exist. For example, [40–42] and references therein. Cloud resource allocation by using a continuous double auction was studied in [43]. A resource allocation model for data grids with cost-performance ratio was proposed in [44]. Performance and cost optimization for multiple large-scale grid workflow applications was studied in [45].

1.2. Organization

The rest of the paper is organized as follows. In Section 2, we present the pricing model used for obtaining the prices charged by the resource owners. In Section 3, we present the system model considered. The system-optimal (global-optimal) scheme (GOSP) is presented in Section 4, and the user-optimal scheme (NASHP) is presented in Section 5. The performance of the proposed schemes is evaluated in Section 6. Conclusions are drawn in Section 7.

2. PRICING MODEL

The set of computing resources belonging to a utility computing system will be typically managed by a set of servers. In utility computing systems, where the computing resources belong to various owners (e.g., utility systems based on computational grids), an agreement should be reached between the software agents at the servers acting on behalf of the users and the software agents of the resource owners at the computing resources for executing the users jobs. For computationally intensive applications that require multiple resources, the software agent of a user might have to negotiate with multiple owner agents. The negotiation will be for an agreed price-per-unit-resource that has to be paid by the user for executing her/his jobs to the resource owner.

To obtain the prices charged by the resource owners, we use a pricing model based on bargaining game theory studied in [1]. The software agents of the user (at the server) and the resource owner (at the computer) play an incomplete information, alternating-offer, non-cooperative bargaining game to obtain an agreed price-per-unit-resource. For simplicity, in the following, we use the terms ‘users’ and ‘computers’ instead of ‘software agents of the users’ and ‘software agents of the owners/computers’, respectively.

The two players (users and computers) have no idea of each other’s reserved valuations [46], that is, the maximum offered price for the user (acting as the buyer of resource) and the minimum expected price for the computers (acting as the seller of the resource). The user has to play an independent game with each computer concerned to form the price-per-unit-resource vector, p^j for user j. So, in a system with m users and n computers at time t, we have $m \times n$ bargaining games as shown in Figure 1.

Both the players try to maximize their utility functions. The bargaining protocol is as follows: one of the players starts the game. If the user starts the game, she/he proposes an offer, which will be much less than her/his own reserved valuation. If the *offered price* $\geq$ *the computer’s standard price with highest expected surplus*, then the computer accepts the offer (*standard price*s are the

DOI: 10.1002/cpe

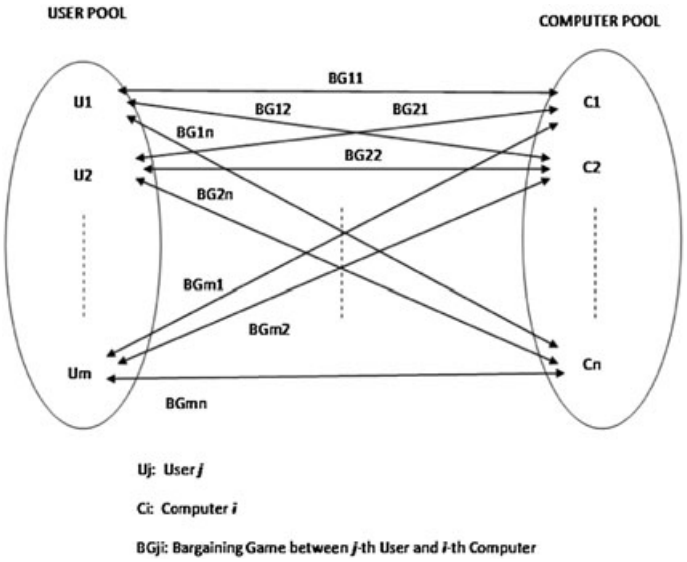
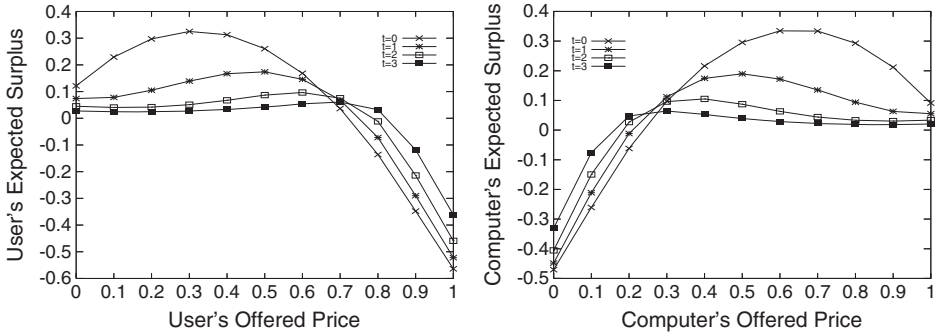


Figure 1. Bargaining game mapping between the users and computers.



a. Expected surplus of the User b. Expected surplus of the Computer (Resource Owner)

Figure 2. Expected surplus versus offered prices: (a) expected surplus of the user; (b) expected surplus of the computer (resource owner).

different offered prices by the players). Else, it makes a counter offer. If this *counter offer* $\leqslant$ *the users' standard price with the highest expected surplus*, she/he accepts. Else, the user counter offers again. This procedure continues until an agreement is reached. At each step, the expected surplus of each player is based on the probability of acceptance, breakdown or counter-offer of the other player. In general, they are given by: *Expected Utility* $=$ *E[Surplus]* $=$ *(reserved valuation of* x $-$ *standard price of* x*)* $\times$ *probability (standard price)* [47], where x stands for the user or the computer and *probability* (*standard price*) is the probability that the standard price will be accepted by the other player as predicted by itself, and the *standard price*s are the different offered prices used by the players to compute their expected surplus. Also, at each step, if an offer is rejected, then the players will update (i.e., reduce) the *probability* (*standard price*), which monotonically decreases as the alternatives come closer to their reserved valuations, where it is more likely to be accepted by the opponent [48].

We simulated this pricing model based on the assumptions given in [1]. Figure 2(a) and (b) shows the expected surplus (profit) earned by the user and the computer respectively against their various offered prices with time. As the time increases, the expected surplus gradually decreases, which makes both the players offer prices that are much closer to their reserved valuations, and that helps the game to converge.

DOI: 10.1002/cpe

3. SYSTEM MODEL

We consider a utility system model with m server agents (for m users) and n computers (nodes) as shown in Figure 3. The terminology and assumptions used are as follows:

- ϕ^j: job arrival rate of user j (the number of jobs of user j arriving to the system per unit time).
- Φ: total job arrival rate of the system. So, $\Phi = \sum_{j=1}^{m} \phi^j$.
- μ_i: service rate of computer i (the maximum number of jobs that can be processed at node i per unit time).
- p_i^j: price per unit resource on computer i for user j.
- p^j: price vector of user j. So, $p^j = \left[p_1^j, p_2^j, \ldots, p_n^j\right]^T$.

Each computer is modeled as an M/M/1 queuing system [49]. In these queuing systems, the inter-arrival times and the service (processing) times are exponentially distributed, and jobs arrive in a single queue (which is assumed to have infinite capacity) to a single computing resource with a First Come, First Served service discipline. For system stability, we assume that the total job arrival rate Φ is less than the aggregate service rate of the system (i.e., $\Phi < \sum_{i=1}^{n} \mu_i$). The server agent for user j keeps track of the price per unit resource p_i^j for user j at computer i (the bargaining game is played prior to the job allocation) and the service rate μ_i of computer i.

Let s_i^j be the fraction of workload (jobs) of user j that are sent to computer i $\left(\sum_{i=1}^{n} s_i^j = 1, \sum_{i=1}^{n} s_i^j \phi^j = \phi^j, 0 \leqslant s_i^j \leqslant 1, i = 1, \ldots, n\right)$. Thus, $\mathbf{s}^j = \left(s_1^j, s_2^j, \ldots, s_n^j\right)$ denotes the workload fractions of user j, and the vector $\mathbf{s} = (\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^m)$ denotes the load fractions of all the users. The job allocation problem is to find the load fractions (s_i^j) of each user j that are assigned to computer i such that the expected cost (price) for executing the users' jobs is minimized. The objective of GOSP is to find optimal s_i^j's that minimize the overall cost for executing all the jobs in the system, and the objective of NASHP is to find optimal s_i^j's that minimize the cost for individual users in the system for providing fairness.

On the basis of the assumption that each computer is modeled as an M/M/1 queuing system, the expected (mean or average) response time at computer i is given by

$$F_i(\mathbf{s}) = \frac{1}{\mu_i - \sum_{j=1}^{m} s_i^j \phi^j} \tag{1}$$

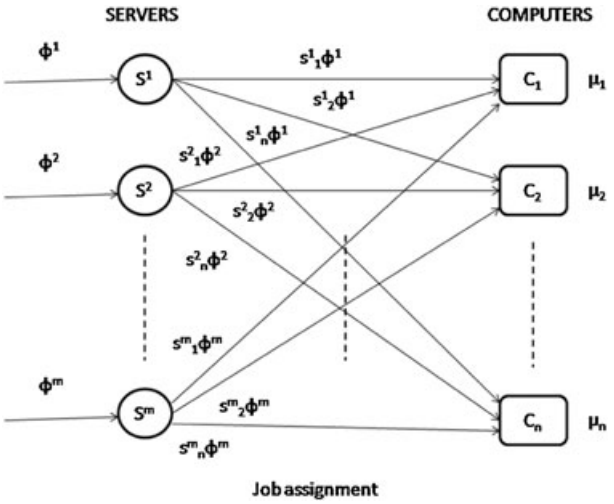


Figure 3. System model.

DOI: 10.1002/cpe

Thus, the expected response time of user j is given by

$$T^j(\mathbf{s}) = \sum_{i=1}^{n} s_i^j F_i(\mathbf{s}) = \sum_{i=1}^{n} \frac{s_i^j}{\mu_i - \sum_{k=1}^{m} s_i^k \phi^k} \tag{2}$$

Let k_i be a constant that maps the execution time to the amount of resources consumed at computer i and p_i^j be the agreed price per unit resource between user j and computer i as a result of the bargaining game.

Thus, the expected cost of user j can be expressed as

$$D^j(\mathbf{s}) = \sum_{i=1}^{n} k_i p_i^j s_i^j F_i(\mathbf{s}) = \sum_{i=1}^{n} \frac{k_i p_i^j s_i^j}{\mu_i - \sum_{k=1}^{m} s_i^k \phi^k} \tag{3}$$

and the overall expected cost of the system (i.e., of all the users) is given by

$$D(\mathbf{s}) = \frac{1}{\Phi} \sum_{j=1}^{m} \phi^j D^j(\mathbf{s}) \tag{4}$$

which is equivalent to

$$D(\mathbf{s}) = \frac{1}{\Phi} \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{k_i p_i^j \phi^j s_i^j}{\mu_i - \sum_{k=1}^{m} s_i^k \phi^k} \tag{5}$$

subject to the constraints

$$s_i^j \geqslant 0, \qquad i = 1, \ldots, n, \qquad j = 1, \ldots, m \tag{6}$$

$$\sum_{i=1}^{n} s_i^j = 1, \qquad j = 1, \ldots, m \tag{7}$$

$$\sum_{j=1}^{m} s_i^j \phi^j < \mu_i, \qquad i = 1, \ldots, n \tag{8}$$

4. PRICE-BASED GLOBAL-OPTIMAL SCHEME

In this section, we present the GOSP whose objective is to minimize the overall cost for executing all the jobs in the system.

The optimal load fractions ($\mathbf{s}$) for minimizing the overall expected cost of the system are obtained by solving the nonlinear optimization problem $D(\mathbf{s})$ (Equation (5)) based on the constraints in Equations (6)–(8). To obtain the optimal solution, the load fractions of each user are found by taking into account the load on each computer due to the other user allocations. We denote the *available processing (service) rate at computer i as seen by user j* by μ_i^j, where $\mu_i^j = \mu_i - \sum_{k=1, k\neq j}^{m} s_i^k \phi^k$.

Theorem 1

Assuming that computers are sorted in decreasing order of their available processing rates $\left(\mu_1^j \geqslant \mu_2^j \geqslant \ldots \geqslant \mu_n^j\right)$, the load fractions for user j are given by

$$s_i^j = \begin{cases} \frac{1}{\phi^j}\left(\mu_i^j - \sqrt{k_i p_i^j \mu_i}\, \frac{\sum_{k=1}^{c_j} \mu_k^j - \phi^j}{\sum_{k=1}^{c_j} \sqrt{k_k p_k^j \mu_k}}\right) & \text{if } 1 \leqslant i < c_j \\ 0 & \text{if } c_j \leqslant i \leqslant n \end{cases} \tag{9}$$

DOI: 10.1002/cpe

where c_j is the minimum index that satisfies the inequality

$$\frac{\mu_{c_j}^j}{\sqrt{k_{c_j} p_{c_j}^j \mu_{c_j}}} \leqslant \frac{\sum_{k=1}^{c_j} \mu_k^j - \phi^j}{\sum_{k=1}^{c_j} \sqrt{k_k p_k^j \mu_k}} \tag{10}$$

Proof
See *Proof of Theorem 1* in the Appendix. □

On the basis of the previously mentioned theorem, the following algorithm (BEST-FRACTIONS) is derived for determining user j's optimal load fractions.

BEST-FRACTIONS $\left(\mu_1^j, \ldots, \mu_n^j, \phi^j, p_1^j, \ldots, p_n^j, k_1, \ldots, k_n\right)$

Input: Available processing rates for user j: $\mu_1^j, \mu_2^j, \ldots \mu_n^j$
Total arrival rate of user j: ϕ^j
The price per unit resource vector of user j: $p_1^j, p_2^j, \ldots p_n^j$
The constants vector: $k_1, k_2, \ldots k_n$
Output: Load fractions for user j: $s_1^j, s_2^j, \ldots s_n^j$

1. Sort the computers in decreasing order of $\frac{\mu_1^j}{\sqrt{k_1 p_1^j \mu_1}} \geqslant \frac{\mu_2^j}{\sqrt{k_2 p_2^j \mu_2}} \geqslant \ldots \geqslant \frac{\mu_n^j}{\sqrt{k_n p_n^j \mu_n}}$.
2. $t \leftarrow \frac{\sum_{i=1}^n \mu_i^j - \phi^j}{\sum_{i=1}^n \sqrt{k_i p_i^j \mu_i}}$.
3. **while** $\left(t \geqslant \frac{\mu_n^j}{\sqrt{k_n p_n^j \mu_n}}\right)$ **do**
   $s_n^j \leftarrow 0$
   $n \leftarrow n - 1$
   $t \leftarrow \frac{\sum_{i=1}^n \mu_i^j - \phi^j}{\sum_{i=1}^n \sqrt{k_i p_i^j \mu_i}}$
4. **for** $i = 1, \ldots, n$ **do**
   $s_i^j \leftarrow \frac{1}{\phi^j}\left(\mu_i^j - t\sqrt{k_i p_i^j \mu_i}\right)$

The time complexity of BEST-FRACTIONS is $O(n \log n)$. The following theorem proves the correctness of the earlier mentioned algorithm.

Theorem 2
The load fractions $\left\{s_1^j, s_2^j, \ldots, s_n^j\right\}$ computed by the BEST-FRACTIONS algorithm solves the optimization problem $D(\mathbf{s})$ and are the optimal fractions for user j.

Proof
See *Proof of Theorem 2* in the Appendix. □

To compute the optimal load fractions of all the users, there should be some communication between them in order to obtain the load information from the other users and compute the μ_i^j's. On the basis of the BEST-FRACTIONS algorithm presented earlier, the following iterative algorithm is devised where each user periodically updates her/his load fractions taking into account the existing load fractions of the other users in a round-robin fashion.

We use the following notations:

j	- the user number;
l	- the iteration number;
$\mathbf{s}^j(l)$	- the load fractions of user j computed at iteration l;
$D^j(l)$	- user j's expected price at iteration l;

DOI: 10.1002/cpe

ϵ - a properly chosen acceptance tolerance;
Send$(j, (p, q, r))$ - send the message (p, q, r) to user j;
Recv$(j, (p, q, r))$ - receive the message (p, q, r) from user j;
(where p is a real number, and q, r are integer numbers).

GOSP Job Allocation Algorithm:

User j, $(j = 1, \ldots, m)$ executes:
1. Initialization:
$\mathbf{s}^j(0) \leftarrow \mathbf{0}$;
$\mathbf{D}^j(0) \leftarrow \mathbf{0}$;
$l \leftarrow 0$;
$norm \leftarrow 1$;
$sum \leftarrow 0$;
$tag \leftarrow$ CONTINUE;
$left = [(j-2) \bmod m] + 1$;
$right = [j \bmod m] + 1$;
2. **while** (1) **do**
if$(j = 1)$ {User 1}
if $(l \neq 0)$
Recv $(left, (norm, l, tag))$;
if $(norm < \epsilon)$
Send $(right, (norm, l, \text{STOP}))$;
exit;
$sum \leftarrow 0$;
$l \leftarrow l + 1$;
else {the other users}
Recv$(left, (sum, l, tag))$;
if $(tag = \text{STOP})$
if $(j \neq m)$
Send $(right, (sum, l, \text{STOP}))$;
exit;
for $i = 1, \ldots, n$ **do**
Obtain μ_i^j by inspecting the run queue of each computer
$\left(\mu_i^j \leftarrow \mu_i - \sum_{k=1, k \neq j}^{m} s_i^k \phi^k\right)$;
$\mathbf{s}^j(l) \leftarrow$ BEST-FRACTIONS$\left(\mu_1^j, \ldots, \mu_n^j, \phi^j, p_1^j, \ldots, p_n^j, k_1, \ldots, k_n\right)$;
Compute $D^j(l)$;
$sum \leftarrow sum + |D^j(l-1) - D^j(l)|$;
Send $(right, (sum, l, \text{CONTINUE}))$;
endwhile

The GOSP algorithm can be restarted periodically by the scheduling agents when the system parameters (or system load) change above some threshold. Once the accepted tolerance is reached, the users will continue to use the same load fractions, and the system operates at the optimal cost. GOSP minimizes the expected cost over all the jobs executed by the utility computing system.

5. PRICE-BASED USER-OPTIMAL SCHEME

In this section, we present the NASHP whose objective is to minimize the cost for individual users in the utility computing system for providing fairness to all the users.

The previously mentioned problem is formulated as a non-cooperative game among the users (under the assumption that the users are 'selfish'), where each user minimizes the total cost of her/his own jobs independently of the others. We use the game theory terminology introduced in [4].

DOI: 10.1002/cpe

The goal of user j ($j = 1, \ldots, m$) is to find a feasible job allocation strategy $\mathbf{s}^j$ such that $D^j(\mathbf{s})$ (Equation (3)) is minimized. The vector $\mathbf{s}^j = \left(s_1^j, s_2^j, \ldots, s_n^j\right)$ is called the job allocation *strategy* of user j, and the vector $\mathbf{s} = (\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^m)$ is called the *strategy profile* of the job allocation game. The strategy of user j depends on the job allocation strategies of the other users.

The assumptions for the existence of a feasible strategy profile are the constraints in Equations (6)–(8).

A *non-cooperative job allocation game* consists of a set of players, a set of strategies, and preferences over the set of strategy profiles:

(i) *Players*. The m users.
(ii) *Strategies*. Each user's set of feasible job allocation strategies.
(iii) *Preferences*. Each user's preferences are represented by her/his expected cost (D^j). Each user j prefers the strategy profile s to the strategy profile s', if and only if $D^j(s) < D^j(s')$.

A solution to the previously mentioned job allocation game can be obtained at the Nash equilibrium [50]. At the Nash equilibrium, a user j cannot further decrease her/his expected cost by choosing a different job allocation strategy when the other users' strategies are fixed. The equilibrium strategy profile can be found when each user's job allocation strategy is a *best reply* to the other users' strategies. The best reply of user j, which is the solution of the optimization problem $D^j(\mathbf{s})$ (Equation (3)) is given by the following theorem.

Theorem 3
Assuming that computers are sorted in decreasing order of their available processing rates $\left(\mu_1^j \geqslant \mu_2^j \geqslant \ldots \geqslant \mu_n^j\right)$, the solution $\mathbf{s}^j$ of the optimization problem $D^j(\mathbf{s})$ is given by

$$s_i^j = \begin{cases} \frac{1}{\phi^j}\left(\mu_i^j - \sqrt{k_i p_i^j \mu_i^j}\,\frac{\sum_{k=1}^{c_j}\mu_k^j - \phi^j}{\sum_{k=1}^{c_j}\sqrt{k_k p_k^j \mu_k^j}}\right) & \text{if } 1 \leqslant i < c_j \\ 0 & \text{if } c_j \leqslant i \leqslant n \end{cases} \tag{11}$$

where c_j is the minimum index that satisfies the inequality

$$\sqrt{\frac{\mu_{c_j}^j}{k_{c_j} p_{c_j}^j}} \leqslant \frac{\sum_{k=1}^{c_j}\mu_k^j - \phi^j}{\sum_{k=1}^{c_j}\sqrt{k_k p_k^j \mu_k^j}} \tag{12}$$

Proof
See *Proof of Theorem 3* in the Appendix. □

In finding the solution to $D^j(\mathbf{s})$, the strategies of all the other users are kept fixed, and so, the variables in $D^j(\mathbf{s})$ are the strategies of user j.

In the following, we present an algorithm ('BEST-REPLY') for determining user j's best reply strategy.

BEST-REPLY' $\left(\mu_1^j, \ldots, \mu_n^j, \phi^j, p_1^j, \ldots, p_n^j, k_1, \ldots, k_n\right)$
Input: Available processing rates for user j: $\mu_1^j, \mu_2^j, \ldots \mu_n^j$
Total arrival rate of user j: ϕ^j
The price per unit resource vector of user j: $p_1^j, p_2^j, \ldots p_n^j$
The constants vector: $k_1, k_2, \ldots k_n$
Output: Load fractions for user j: $s_1^j, s_2^j, \ldots s_n^j$

1. Sort the computers in decreasing order of $\sqrt{\frac{\mu_1^j}{k_1 p_1^j}} \geqslant \sqrt{\frac{\mu_2^j}{k_2 p_2^j}} \geqslant \ldots \geqslant \sqrt{\frac{\mu_n^j}{k_n p_n^j}}$.
2. $t \leftarrow \frac{\sum_{i=1}^{n}\mu_i^j - \phi^j}{\sum_{i=1}^{n}\sqrt{k_i p_i^j \mu_i^j}}$.

DOI: 10.1002/cpe

3. **while** $\left(t \geqslant \sqrt{\frac{\mu_n^j}{k_n p_n^j}}\right)$ **do**
$s_n^j \leftarrow 0$
$n \leftarrow n-1$
$t \leftarrow \frac{\sum_{i=1}^{n} \mu_i^j - \phi^j}{\sum_{i=1}^{n} \sqrt{k_i p_i^j \mu_i^j}}$
4. **for** $i = 1, \ldots, n$ **do**
$s_i^j \leftarrow \frac{1}{\phi^j}\left(\mu_i^j - t\sqrt{k_i p_i^j \mu_i^j}\right)$

The time complexity of BEST-REPLY' is $O(n \log n)$. The following theorem proves the correctness of the above algorithm.

Theorem 4
The job allocation strategies $\left\{s_1^j, s_2^j, \ldots, s_n^j\right\}$ computed by the BEST-REPLY' algorithm solves the optimization problem $D^j(\mathbf{s})$ and are the best reply strategies of user j.

Proof
See *Proof of Theorem 4* in the Appendix. □

To obtain the equilibrium allocation, we need an iterative algorithm where each user updates her/his strategies (by applying the BEST-REPLY' algorithm) periodically by fixing the other usersŠ strategies. In the following, we present the iterative algorithm (NASHP) for computing the Nash equilibrium for our non-cooperative job allocation game (similar to GOSP).

NASHP job allocation algorithm:
Each user j, $j = 1, \ldots, m$ performs the following steps in each iteration:

1. Receive the current strategies of all the other users from the left neighbor.
2. If the message is a termination message, then pass the termination message to the right neighbor and EXIT.
3. Update the strategies (s^j) by calling the BEST-REPLY'.
4. Calculate D^j (Equation (3)) and update the error norm.
5. Send the updated strategies and the error norm to the right neighbor.

6. EXPERIMENTAL RESULTS

We performed simulations with various system loads and configurations to evaluate the performance of the proposed schemes. The simulation software package used is Sim++ [51]. The performance metrics used are the *expected price* and the *fairness index* [52]. The *fairness index* (defined from the users' perspective) is defined as

$$I(\mathbf{C}) = \frac{\left[\sum_{j=1}^{m} C^j\right]^2}{m \sum_{j=1}^{m} {C^j}^2}. \tag{13}$$

The input $\mathbf{C}$ is the vector $\mathbf{C} = (C^1, C^2, \ldots, C^m)$ where C^j is the expected cost (or price) for executing user j's jobs. This index is a measure of the 'equality' of the users' total expected cost. If all the users have the same total expected cost, then $I = 1$ and the system is 100% fair to all users, and it is cost-balanced. If the differences on C^j increase, I decreases and the job allocation scheme favors only some users.

We also implemented the PROP [3] for comparison purposes. According to this scheme, each user allocates her/his jobs to computers in proportion to their processing rates as follows:

$$s_i^j \longleftarrow \frac{\mu_i}{\sum_{k=1}^{n} \mu_k} \tag{14}$$

DOI: 10.1002/cpe

6.1. *Effect of system utilization*

In this section, we study the effect of system utilization (system load) on the proposed job allocation schemes. To study the effect of system utilization, we simulated a heterogeneous system consisting of 32 computers with eight different processing rates. This system is shared by 20 users.

The system configuration is presented in Table I. The first row contains the relative service rates of each of the eight computer types. Here, the relative service rate for computer C_i is defined as the ratio of the service rate of C_i to the service rate of the slowest computer in the system. The second row contains the number of computers in the system corresponding to each computer type. The third row shows the service rate of each computer type in the system. The last row shows the values for k_i, the constant which maps the response time to the amount of resources consumed at computer i. They are assigned on the basis of the service rates of the computers as in [1]. A faster computer has a higher k_i, because it will expect more price to perform the same amount of work as other slower computers and thus the 'effective resources' consumed in that case is higher. The price vector p^j for user j is based on the alternating offer bargaining game described in Section 2.

For each experiment, the total job arrival rate of the system (Φ) is determined by the system utilization (or system load), and the aggregate service rate of the system. *system utilization* (ρ) is defined as the ratio of the total arrival rate to the aggregate service rate of the system:

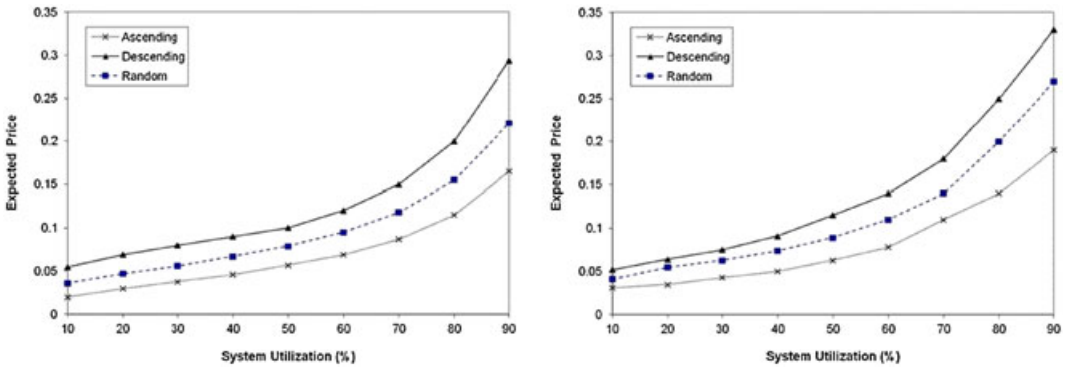
$$\rho = \frac{\Phi}{\sum_{i=1}^{n} \mu_i} \tag{15}$$

We choose fixed values for the system utilization and determined the total job arrival rate Φ. For example, if we consider $\rho = 10\%$ and an aggregate service rate of 1220 jobs/s, then $\Phi = 122$ jobs/s.

Figure 4 presents the expected price (or cost) (in some monetary units) of the system (cost for executing all the jobs in the system) for various system utilizations based on the allocations provided by GOSP and NASHP. The price increases with system utilization, because the higher the ρ, the more the load on the computers and hence the higher the price. Three curves are shown corresponding to random, strictly decreasing, and strictly increasing price vectors (the computers are

Table I. System configuration.

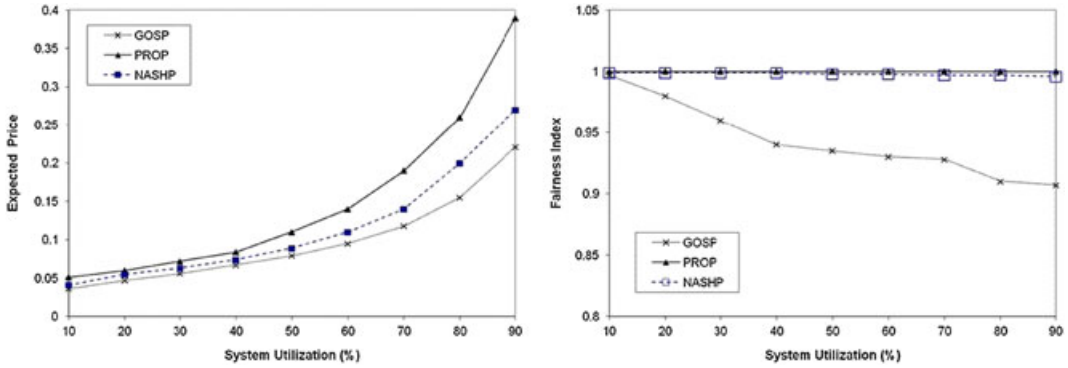
Relative μ_i	1	2	3	4	5	7	8	10
No. of computers	7	6	5	4	3	3	2	2
μ_i (jobs/s)	10	20	30	40	50	70	80	100
k_i	1	2	3	4	5	7	8	10



a. GOSP b. NASHP

Figure 4. Variation of expected price with system utilization for various price vectors. (a) GOSP; (b) NASHP.

DOI: 10.1002/cpe



a. Expected Price b. Fairness Index

Figure 5. Effect of system utilization: (a) expected price; (b) fairness index.

initially numbered in decreasing order of their service rates). The random price vector is the one obtained by the pricing strategy described in Section 2, and the corresponding curve lies between that for the ascending and the descending price vector cases. This is because, if the faster computers charge less (in the case of price vector in ascending order), then they will get the bulk of the work (load) resulting in lower total cost for executing the jobs. Similarly, if the faster devices charge more (in the case of price vector in descending order), then, they will get fewer jobs resulting in higher total cost for executing the jobs.

Figure 5(a) presents the expected cost of the system (cost for executing all the jobs in the system) for different values of system utilization (ranging from 10% to 90%) based on the allocations provided by GOSP, NASHP, and PROP using the random price vector. This corresponds to a total job arrival rate ranging from 122 to 1098 jobs/s. The expected price obtained by GOSP is the lowest because it provides a system-optimal solution. The expected price provided by NASHP is close to GOSP for low and medium system utilizations and is between GOSP and PROP for high system utilizations. For example, at 50% system utilization, the expected price obtained by NASHP is around 12% higher than GOSP and at 90% system utilization, the expected price obtained by NASHP is around 23% higher than GOSP. This is because NASHP tries to provide fairness and not a system-optimal solution. At 90% system utilization, the expected price obtained by PROP is around 75% higher than GOSP. PROP overloads the less powerful computers and so its expected cost is higher than the other schemes.

Figure 5(b) presents the fairness index for different values of system utilization based on the allocations provided by GOSP, NASHP, and PROP using the random price vector. It can be observed that the fairness index for PROP is always 1 (which can be easily checked from Equation (13)), and the fairness index for NASHP is very close to 1 for various system loads. The fairness index for GOSP drops from 0.99 (at low system load) to 0.9 (at high system load). So, GOSP whose objective is to reduce the overall cost of the system is less fair to the users than NASHP.

6.2. Effect of heterogeneity

In this section, we study the effect of heterogeneity on the proposed job allocation schemes. A simple way to characterize system heterogeneity is to use the processor speed. Furthermore, it is reasonable to assume that a computer with high speed processor will have matching resources (memory and I/O). One of the common measures of heterogeneity is the *speed skewness* [8] which is defined as the ratio of *maximum service rate* to *minimum service rate* of the computers in the system.

We study the performance of the job allocation schemes by varying the speed skewness. We simulated a heterogeneous system consisting of 32 computers (eight, slow; 16, medium-fast; and eight, fast) to study the effect of heterogeneity. The slow computers have a relative service rate of 1 and the relative service rate of the medium-fast and fast computers is varied from 1 (homogeneous system) to 10 and 100, respectively (highly heterogeneous system). For example, the highly heterogeneous

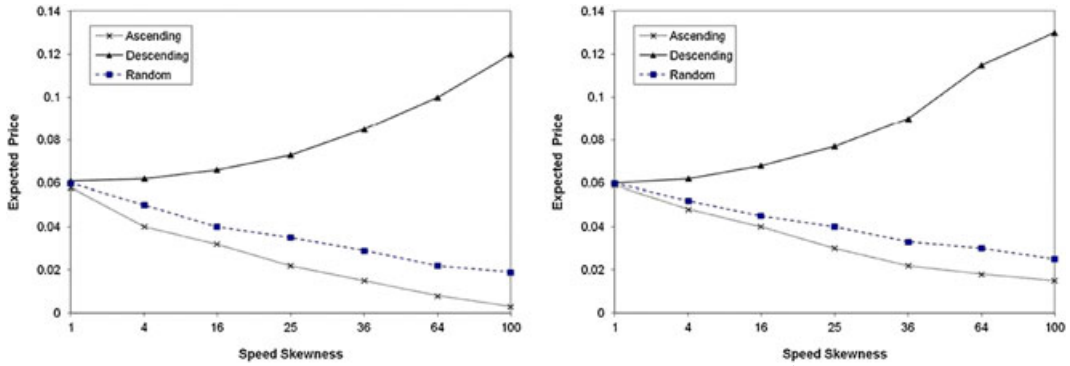
DOI: 10.1002/cpe

system has eight slow computers with $\mu = 5$ jobs/s, 16 medium-fast computers with $\mu = 50$ jobs/s, and eight fast computers with $\mu = 500$ jobs/s. The system utilization was kept constant at 60%. The total job arrival rates (Φ) are calculated using Equation (15).

Figure 6 presents the expected price for speed skewness ranging from 1 to 100 based on the allocations provided by GOSP and NASHP. Similar to Figure 4, three curves are shown corresponding to random, strictly decreasing, and strictly increasing price vectors (with the computers initially numbered in decreasing order of their service rates). The expected price for the random price vector lies between that for the ascending and descending price vectors for similar reasons as discussed in the previous section.

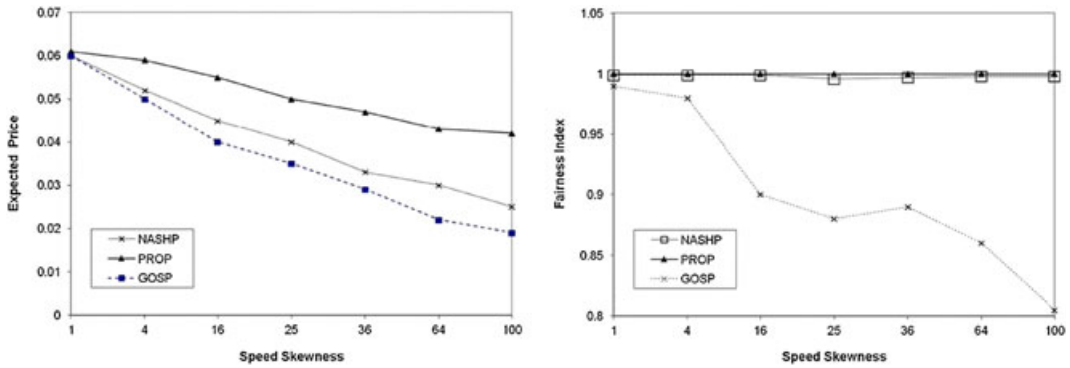
Figure 7(a) presents the expected cost of the system for speed skewness ranging from 1 to 100 based on the allocations provided by GOSP, NASHP, and PROP using the random price vector. It can be observed that for low skewness, the expected costs of GOSP and NASHP are very close. GOSP provides the lowest expected price for any speed skewness, because it provides a system-optimal solution. The expected cost provided by NASHP is about 12.5% (skewness = 16) to 50% (skewness = 64) higher than that of GOSP. At skewness = 64, the expected cost provided by PROP is about 95% higher than that of GOSP. The poor performance of PROP is because the less powerful computers are significantly overloaded.

Figure 7b presents the fairness index for speed skewness ranging from 1 to 100 based on the allocations provided by GOSP, NASHP, and PROP using the random price vector. It can be again observed that the fairness index for PROP is always 1 and the fairness index for NASHP is very close to 1 for various levels of skewness. The fairness index for GOSP drops from 0.99 (at skewness = 1) to 0.8 (at skewness = 100). This shows that the GOSP scheme produces an allocation which does not guarantee equal expected price for all the users in the system.



a. GOSP b. NASHP

Figure 6. Variation of expected price with speed skewness for various price vectors: (a) GOSP; (b) NASHP.



a. Expected Price b. Fairness Index

Figure 7. Effect of speed skewness: (a) expected price; (b) fairness index.

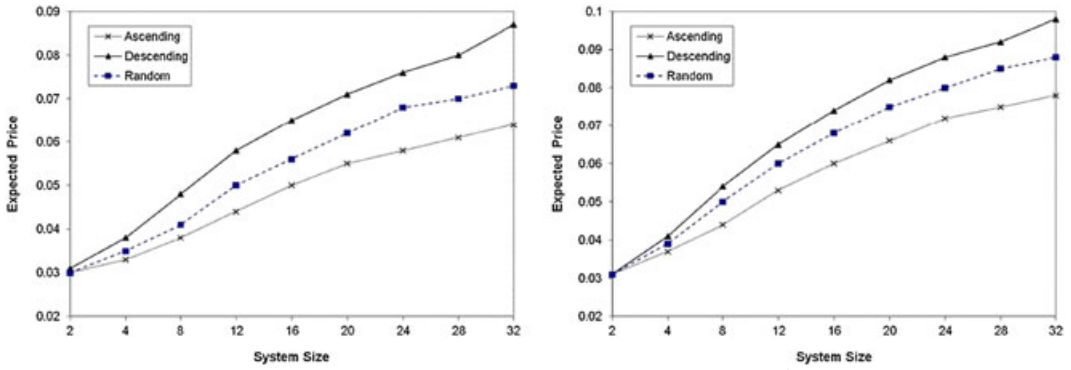
DOI: 10.1002/cpe

6.3. *Effect of system size*

In this section, we study the effect of system size on the proposed job allocation schemes. To study the effect of system size we simulated a heterogeneous system consisting of four types of computers: slow computers (relative service rate = 1, 2 and 5) and fast computers (relative service rate = 10). For a system size of 2, two fast computers are used. To increase the system size, the number of fast computers are kept constant and the number of slow computers are increased. The system utilization was kept constant at 60%, and the total arrival rates (Φ) are calculated using Equation (15).

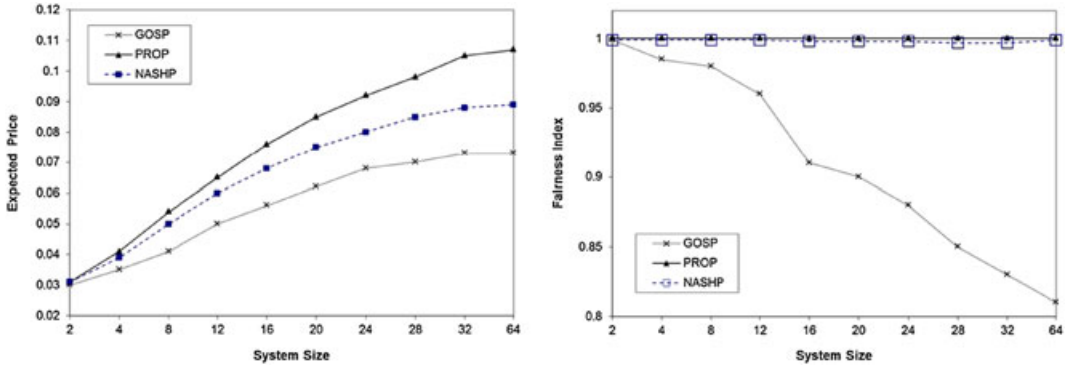
Figure 8 presents the expected price for system sizes ranging from 2 to 32 based on the allocations provided by GOSP and NASHP. Similar to Figure 4, three curves are shown corresponding to random, strictly decreasing, and strictly increasing price vectors (with the computers initially numbered in decreasing order of their service rates). The expected prices for the random price vectors lie between that for the ascending and descending price vectors for similar reasons as discussed in the previous section.

Figure 9(a) presents the expected cost of the system for system sizes ranging from 2 to 64 based on the allocations provided by GOSP, NASHP, and PROP using the random price vector. It can be observed that for small system sizes the expected cost provided by all the schemes are close. As the system size increases, the expected cost for all the schemes increases (because of the system configurations considered). The expected cost provided by GOSP is the lowest for any system size and the cost provided by NASHP lies between that of GOSP and PROP. For example, at a system size of 32, the expected price provided by NASHP is around 20% higher than GOSP and the expected price provided by PROP is around 45% higher then GOSP.



a. GOSP b. NASHP

Figure 8. Variation of expected price with system size for various price vectors: (a) GOSP; (b) NASHP.



a. Expected Price b. Fairness Index

Figure 9. Effect of system size: (a) expected price; (b) fairness index.

DOI: 10.1002/cpe

Figure 9(b) presents the fairness index for system sizes ranging from 2 to 64 based on the allocations provided by GOSP, NASHP, and PROP using the random price vector. It can be again observed that the fairness index for PROP is always 1 and the fairness index for NASHP is very close to 1 for various system sizes. The fairness index for GOSP drops from 0.99 (for system size = 2) to 0.81 (for system size = 64).

From Figures 9(a) and (b), it can also be observed that the expected price increases (sub-linearly) reasonably 'slowly' with system size, and the fairness index changes by a small percentage with scaling up the system size in the case of GOSP (-2.5%) and remains unchanged for NASHP.

Remark 6.1

In the GOSP and NASHP algorithms, one iteration of the algorithm is said to be complete when a full circle of messages is completed (in a virtual ring interconnection topology). In all the above experiments, the acceptance tolerance (ϵ) chosen is 10^{-5} for the algorithms to converge. In the case of GOSP, the convergence is achieved in at most three iterations for any system utilization from 10% to 90%. In the case of NASHP, the convergence is achieved in: five iterations for low system utilization (10–30%), about 18 iterations for medium system utilization (40–60%), and about 20–24 iterations for high system utilization (70–90%).

7. CONCLUSIONS

In this paper, we studied two job allocation schemes (GOSP and NASHP) based on pricing for utility computing systems. The objective of GOSP is to provide a system-optimal solution, and the objective of NASHP is to provide a user-optimal solution (fairness). The problem of providing a system-optimal solution is formulated as a constraint minimization problem, and the problem of providing a user-optimal solution is formulated as a non-cooperative game. Algorithms for computing the optimal load (job) fractions for the users are devised. Simulations were performed using a variety of system configurations to compare the schemes in a fair manner. Experimental results showed that GOSP will be suitable for systems/cases where a minimum cost to execute all the jobs in the system is desired. However, it is not fair to the individual users. Some users may have to pay higher price than the others (for executing jobs of approximately the same size). NASHP will be suitable for systems in which the fair treatment of the users is as important as other performance characteristics. In future work, we plan to use a more practical queuing model to model the computers in the system.

APPENDIX A

In this section we present the proofs of the theorems presented in the paper.

Proof of Theorem 1

We begin with the observation that the stability condition (Equation (8)) is always satisfied because of the fact that the total arrival rate (Φ) does not exceed the total service rate of the system. Thus, we consider $D(\mathbf{s})$ problem with only two restrictions, Equations (6) and (7).

We first show that $D(\mathbf{s})$ is a convex function in $\mathbf{s}$ and that the set of feasible solutions defined by the constraints (Equations (6) and (7)) is convex.

From Equation (5), it can be easily shown that $\frac{\partial D(\mathbf{s})}{\partial s_i^j} \geqslant 0$ and $\frac{\partial^2 D(\mathbf{s})}{\partial \left(s_i^j\right)^2} \geqslant 0$ for $i = 1, \ldots, n$. This means that the Hessian of $D(\mathbf{s})$ is positive, which implies that $D(\mathbf{s})$ is a convex function of the load fractions $\mathbf{s}$. The constraints are all linear and they define a convex polyhedron.

Thus, $D(\mathbf{s})$ involves minimizing a convex function over a convex feasible region, and the first order Kuhn–Tucker conditions [53] are necessary and sufficient for optimality.

DOI: 10.1002/cpe

Let $\alpha \geqslant 0$, $\eta_i^j \geqslant 0$, $i = 1,\ldots,n$, $j = 1,\ldots,m$ denote the Lagrange multipliers [53]. The Lagrangian is

$$L\left(s_1^1,\ldots,s_n^m,\alpha,\eta_1^1,\ldots,\eta_n^m\right) = \sum_{j=1}^{m}\sum_{i=1}^{n}\frac{k_i p_i^j \phi^j s_i^j}{\Phi\left(\mu_i - \sum_{k=1}^{m} s_i^k \phi^k\right)} - \alpha\left(\sum_{j=1}^{m}\sum_{i=1}^{n} s_i^j - m\right) - \sum_{j=1}^{m}\sum_{i=1}^{n}\eta_i^j s_i^j \tag{A.1}$$

The Kuhn–Tucker conditions imply that s_i^j, $j = 1,\ldots,m$, $i = 1,\ldots,n$ is the optimal solution to D(**s**) if and only if there exists $\alpha \geqslant 0$, $\eta_i^j \geqslant 0$, $j = 1,\ldots,m$, $i = 1,\ldots,n$ such that

$$\frac{\partial L}{\partial s_i^j} = 0 \tag{A.2}$$

$$\frac{\partial L}{\partial \alpha} = 0 \tag{A.3}$$

$$\eta_i^j s_i^j = 0, \eta_i^j \geqslant 0, s_i^j \geqslant 0, j = 1,..,m; i = 1,..,n \tag{A.4}$$

These conditions become

$$\frac{k_i p_i^j \phi^j \mu_i}{\Phi\left(\mu_i^j - s_i^j \phi^j\right)^2} - \alpha - \eta_i^j = 0, \quad j = 1,..,m; i = 1,..,n \tag{A.5}$$

$$\sum_{i=1}^{n} s_i^j = 1, \qquad j = 1,\ldots,m \tag{A.6}$$

$$\eta_i^j s_i^j = 0, \eta_i^j \geqslant 0, s_i^j \geqslant 0, j = 1,..,m; i = 1,..,n \tag{A.7}$$

These are equivalent to

$$\alpha = \frac{k_i p_i^j \phi^j \mu_i}{\Phi\left(\mu_i^j - s_i^j \phi^j\right)^2}, if\ s_i^j > 0; 1 \leqslant j \leqslant m; 1 \leqslant i \leqslant n \tag{A.8}$$

$$\alpha \leqslant \frac{k_i p_i^j \phi^j \mu_i}{\Phi\left(\mu_i^j - s_i^j \phi^j\right)^2}, if\ s_i^j = 0; 1 \leqslant j \leqslant m; 1 \leqslant i \leqslant n \tag{A.9}$$

$$\sum_{i=1}^{n} s_i^j = 1, \quad s_i^j \geqslant 0; \quad j = 1,\ldots,m, \quad i = 1,\ldots,n \tag{A.10}$$

Claim

Obviously, a computer with a higher average service rate should have a higher fraction of jobs assigned to it. Under the assumption on the ordering of computers $\left(\mu_1^j \geqslant \mu_2^j \geqslant \ldots \geqslant \mu_n^j\right)$, we have the following order on load fractions for each user: $s_1^j \geqslant s_2^j \geqslant \ldots \geqslant s_n^j$. This implies that there may exist situations in which the slow computers have no jobs assigned to them by the users. This means that there exists an index c_j $(1 \leqslant c_j \leqslant n)$ so that $s_i^j = 0$ for $i = c_j,\ldots,n$ for each user.

DOI: 10.1002/cpe

From Equation (A.8), and based on the previously mentioned claim, we can obtain by summation the following equation for each user:

$$\sum_{i=1}^{c_j-1}\sqrt{k_i p_i^j \phi^j \mu_i} = \sqrt{\alpha\Phi}\left(\sum_{i=1}^{c_j-1}\mu_i^j - \sum_{i=1}^{c_j-1} s_i^j \phi^j\right) \tag{A.11}$$

Using Equation (A.9), the previously mentioned equation becomes:

$$\sqrt{\alpha\Phi} = \frac{\sum_{i=1}^{c_j-1}\sqrt{k_i p_i^j \phi^j \mu_i}}{\sum_{i=1}^{c_j-1}\mu_i^j - \sum_{i=1}^{c_j-1} s_i^j \phi^j} \leqslant \frac{\sqrt{k_{c_j} p_{c_j}^j \phi^j \mu_{c_j}}}{\mu_{c_j}^j} \tag{A.12}$$

This is equivalent to

$$\mu_{c_j}^j \sum_{i=1}^{c_j}\sqrt{k_i p_i^j \mu_i} \leqslant \sqrt{k_{c_j} p_{c_j}^j \mu_{c_j}}\left(\sum_{i=1}^{c_j}\mu_i^j - \phi^j\right) \tag{A.13}$$

Thus, the index c_j is the minimum index that satisfies the previously mentioned equation and the result follows.

□

Proof of Theorem 2

The while loop in step 3 finds the minimum index c_j for which $\frac{\mu_{c_j}^j}{\sqrt{k_{c_j} p_{c_j}^j \mu_{c_j}}} \leqslant \frac{\sum_{k=1}^{c_j}\mu_k^j - \phi^j}{\sum_{k=1}^{c_j}\sqrt{k_k p_k^j \mu_k}}$. In the same loop, s_i^j are set to zero for $i = c_j, \ldots, n$. In step 4, s_i^j is set equal to $\frac{1}{\phi^j}\left(\mu_i^j - \sqrt{k_i p_i^j \mu_i}\frac{\sum_{k=1}^{c_j}\mu_k^j - \phi^j}{\sum_{k=1}^{c_j}\sqrt{k_k p_k^j \mu_k}}\right)$ for $i = 1, \ldots, c_j - 1$. These are in accordance with Theorem 1. Thus, the allocation $\{s_1^j, \ldots, s_n^j\}$ computed by the BEST-FRACTIONS algorithm is the optimal solution for each user. □

Proof of Theorem 3

Similar to the proof of Theorem 1, we observe that the stability condition (Equation (8)) is always satisfied because of the fact that the total arrival rate (Φ) does not exceed the total service rate of the system. Thus, we consider $D^j(\mathbf{s})$ (Equation (3)) problem with only two restrictions, Equations (6) and (7).

From Equation (3), it can be shown that $\frac{\partial D^j(\mathbf{s})}{\partial s_i^j} \geqslant 0$ and $\frac{\partial^2 D^j(\mathbf{s})}{\partial\left(s_i^j\right)^2} \geqslant 0$ for $i = 1, \ldots, n$. This means that $D^j(\mathbf{s})$ is a convex function of the load fractions $\mathbf{s}^j$, and the constraints are all linear, and they define a convex polyhedron.

Thus, $D^j(\mathbf{s})$ involves minimizing a convex function over a convex feasible region, and the first order Kuhn–Tucker conditions [53] are necessary and sufficient for optimality.

Let $\alpha \geqslant 0$, $\eta_i \geqslant 0$, $i = 1, \ldots, n$ denote the Lagrange multipliers [53]. The Lagrangian is

$$L\left(s_1^j, \ldots, s_n^j, \alpha, \eta_1, \ldots, \eta_n\right) = \sum_{i=1}^{n}\frac{k_i p_i^j s_i^j}{\left(\mu_i - \sum_{k=1}^{m} s_i^k \phi^k\right)} - \alpha\left(\sum_{i=1}^{n} s_i^j - 1\right) - \sum_{i=1}^{n}\eta_i s_i^j \tag{A.14}$$

The Kuhn–Tucker conditions imply that s_i^j, $i = 1, \ldots, n$ is the optimal solution to $D^j(\mathbf{s})$, if and only if there exists $\alpha \geqslant 0$, $\eta_i \geqslant 0$, $i = 1, \ldots, n$ such that

$$\frac{\partial L}{\partial s_i^j} = 0 \tag{A.15}$$

DOI: 10.1002/cpe

$$\frac{\partial L}{\partial \alpha} = 0 \tag{A.16}$$

$$\eta_i s_i^j = 0, \eta_i \geqslant 0, s_i^j \geqslant 0, i = 1, .., n \tag{A.17}$$

These conditions become

$$\frac{k_i p_i^j \mu_i^j}{\left(\mu_i^j - s_i^j \phi^j\right)^2} - \alpha - \eta_i = 0, \quad i = 1, .., n \tag{A.18}$$

$$\sum_{i=1}^{n} s_i^j = 1, \tag{A.19}$$

$$\eta_i s_i^j = 0, \eta_i \geqslant 0, s_i^j \geqslant 0, i = 1, .., n \tag{A.20}$$

These are equivalent to

$$\alpha = \frac{k_i p_i^j \mu_i^j}{\left(\mu_i^j - s_i^j \phi^j\right)^2}, if\ s_i^j > 0; 1 \leqslant i \leqslant n \tag{A.21}$$

$$\alpha \leqslant \frac{k_i p_i^j \mu_i^j}{(\mu_i^j - s_i^j \phi^j)^2}, if\ s_i^j = 0; 1 \leqslant i \leqslant n \tag{A.22}$$

$$\sum_{i=1}^{n} s_i^j = 1, \quad s_i^j \geqslant 0; \quad i = 1, \ldots, n \tag{A.23}$$

Claim

Because a computer with a higher average service rate should have a higher fraction of jobs assigned to it, under the assumption on the ordering of computers $\left(\mu_1^j \geqslant \mu_2^j \geqslant \ldots \geqslant \mu_n^j\right)$, we have the following order on load fractions for each user: $s_1^j \geqslant s_2^j \geqslant \ldots \geqslant s_n^j$. This implies that there may exist situations in which the slow computers have no jobs assigned to them by the user. This means that there exists an index c_j $(1 \leqslant c_j \leqslant n)$ so that $s_i^j = 0$ for $i = c_j, \ldots, n$ for each user.

From Equation (A.21), and on the basis of the previously mentioned claim, we can obtain by summation the following equation for each user:

$$\sum_{i=1}^{c_j - 1} \sqrt{k_i p_i^j \mu_i^j} = \sqrt{\alpha} \left(\sum_{i=1}^{c_j - 1} \mu_i^j - \sum_{i=1}^{c_j - 1} s_i^j \phi^j \right) \tag{A.24}$$

Using Equation (A.22), the previous equation becomes:

$$\sqrt{\alpha} = \frac{\sum_{i=1}^{c_j - 1} \sqrt{k_i p_i^j \mu_i^j}}{\sum_{i=1}^{c_j - 1} \mu_i^j - \sum_{i=1}^{c_j - 1} s_i^j \phi^j} \leqslant \sqrt{\frac{k_{c_j} p_{c_j}^j}{\mu_{c_j}^j}} \tag{A.25}$$

DOI: 10.1002/cpe

This is equivalent to

$$\sqrt{\mu_{c_j}^j} \sum_{i=1}^{c_j} \sqrt{k_i p_i^j \mu_i^j} \leqslant \sqrt{k_{c_j} p_{c_j}^j} \left(\sum_{i=1}^{c_j} \mu_i^j - \phi^j \right) \tag{A.26}$$

Thus, the index c_j is the minimum index that satisfies the previous equation and the result follows.

□

Proof of Theorem 4

The while loop in step 3 finds the minimum index c_j for which $\sqrt{\frac{\mu_{c_j}^j}{k_{c_j} p_{c_j}^j}} \leqslant \frac{\sum_{k=1}^{c_j} \mu_k^j - \phi^j}{\sum_{k=1}^{c_j} \sqrt{k_k p_k^j \mu_k^j}}$. In the same loop, s_i^j are set to zero for $i = c_j, \ldots, n$. In step 4, s_i^j is set equal to $\frac{1}{\phi^j} \left(\mu_i^j - \sqrt{k_i p_i^j \mu_i^j} \frac{\sum_{k=1}^{c_j} \mu_k^j - \phi^j}{\sum_{k=1}^{c_j} \sqrt{k_k p_k^j \mu_k^j}} \right)$ for $i = 1, \ldots, c_j - 1$. These are in accordance with Theorem 3. Thus, the allocation $\{s_1^j, \ldots, s_n^j\}$ computed by the BEST-REPLY' algorithm is the optimal solution for each user. □

ACKNOWLEDGEMENTS

We gratefully acknowledge the following: (i) the reviewers for their helpful and constructive suggestions, which considerably improved the quality of the manuscript; (ii) support by NSF grant (HRD-0932339) to the University of Texas at San Antonio; and (iii) time grants to access the facilities of FutureGrid at Indiana University Bloomington and also the facilities of TACC at University of Texas at Austin.

REFERENCES

1. Ghosh P, Roy N, Das SK, Basu K. A pricing strategy for job allocation in mobile grids using a non-cooperative bargaining theory framework. *Journal of Parallel and Distributed Computing* 2005; **65**(11):1366–1383.
2. Penmatsa S, Chronopoulos AT. Job allocation schemes in computational grids based on cost optimization. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Joint Workshop on HPGC and HIPS*, Denver, Colorado, USA, 2005; 180–187.
3. Chow YC, Kohler WH. Models for dynamic load balancing in a heterogeneous multiple processor system. *IEEE Transactions on Computers* 1979; **C-28**(5):354–361.
4. Grosu D, Chronopoulos AT. Noncooperative load balancing in distributed systems. *Journal of Parallel and Distributed Computing* 2005; **65**(9):1022–1034.
5. Kameda H, Li J, Kim C, Zhang Y. *Optimal Load Balancing in Distributed Computer Systems*. Springer Verlag: London, 1997.
6. Subrata R, Zomaya A, Landfeldt B. A cooperative game framework for QoS guided job allocation schemes in grids. *IEEE Transactions on Computers* 2008; **57**(10):1413–1422.
7. Shah R, Veeravalli B, Misra M. On the design of adaptive and decentralized load balancing algorithms with load estimation for computational grid environments. *IEEE Transactions on Parallel and Distributed Systems* 2007; **18**(12):1675–1686.
8. Tang X, Chanson ST. Optimizing static job scheduling in a network of heterogeneous computers. *Proceedings of the International Conference on Parallel Processing*, Toronto, Canada, August 2000; 373–382.
9. Georgiadis L, Nikolaou C, Thomasian A. A fair workload allocation policy for heterogeneous systems. *Journal of Parallel and Distributed Computing* 2004; **64**(4):507–519.
10. Kim J-K, Hensgen DA, Kidd T, Siegel HJ, John DS, Irvine C, Levin T, Porter NW, Prasanna VK, Freund RF. A flexible multi-dimensional QoS performance measure framework for distributed heterogeneous systems. *Cluster Computing, Special Issue on Cluster Computing in Science and Engineering* 2006; **9**(3):281–296.
11. Braun TD, Siegel HJ, Maciejewski AA, Hong Y. Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions. *Journal of Parallel and Distributed Computing* 2008; **68**(11):1504–1516.
12. Penmatsa S, Chronopoulos AT. Game-theoretic static load balancing for distributed systems. *Journal of Parallel and Distributed Computing* 2011; **71**:537–555.
13. Cherry D, Li M. Mediagrid: a distributed storage system for archiving broadcast media content. In *Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications*. IGI Global: Hershey, PA, USA, May 2009; 136–146.

DOI: 10.1002/cpe

14. Buyya R, Abramson D, Giddy J, Stockinger H. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience* 2002; **14**:1507–1542. DOI: 10.1002/cpe.690.
15. Yeo CS, Buyya R. Pricing for utility-driven resource management and allocation in clusters. *International Journal of High Performance Computing Applications* 2007; **21**(4):405–418.
16. Yeo CS, Buyya R. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software: Practice and Experience* 2006; **36**(13):1381–1419.
17. Qu Y, Lin C, Li Y, Shan Z. Performability evaluation of resource scheduling algorithms for computational grids. *5th International Conference on Grid and Cooperative Computing*, Hunan, China, October 2006; 319–326.
18. Zhang Y, Koelbel C, Kennedy K. Relative performance of scheduling algorithms in grid environments. *7th IEEE International Symposium on Cluster Computing and the Grid*, Brazil, May 2007; 521–528.
19. Xiao L, Zhu Y, Ni L, Xu Z. Incentive-based scheduling for market-like computational grids. *IEEE Transactions on Parallel and Distributed Systems* 2008; **19**(7):903–913.
20. Penmatsa S, Mantena VP. Static load balancing for cost minimization in distributed computing systems. *Proceedings of the 22nd International Conference on Parallel and Distributed Computing and Communication Systems (PDCCS 2009)*, Louisville, KY, September 24–26, 2009; 19–24.
21. Ghosh P, Basu K, Das SK. A game theory-based pricing strategy to support single/multiclass job allocation schemes for bandwidth-constrained distributed computing systems. *IEEE Transactions on Parallel and Distributed Systems* 2007; **18**(3):289–306.
22. Ghosh P, Das SK. Mobility-aware cost-efficient job scheduling for single-class grid jobs in a generic mobile grid architecture. *Future Generation Computer Systems* 2010; **26**(8):1356–1367.
23. Zheng Q, Tham C-K, Veeravalli B. Dynamic load balancing and pricing in grid computing with communication delay. *Journal of Grid Computing* 2008; **6**(3):239–253.
24. Penmatsa S, Chronopoulos AT. Comparison of price-based static and dynamic job allocation schemes for grid computing systems. *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications (IEEE NCA09)*, Cambridge, MA, July 9–11, 2009; 66–73.
25. Ghosh P, Roy N, Basu K, Das S. A game theory based pricing strategy for job allocation in mobile grids. *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium*, Santa Fe, New Mexico, USA, 2004; 26–30.
26. Penmatsa S, Chronopoulos AT. Price-based user-optimal job allocation scheme for grid systems. *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium*, Rhodes Island, Greece, April 25–29, 2006; 1–8.
27. Subrata R, Zomaya A, Landfeldt B. Game theoretic approach for load balancing in computational grids. *IEEE Transactions on Parallel and Distributed Systems* 2008; **19**(1):66–76.
28. Kwok YK, Hwang K, Song S. Selfish grids: game-theoretic modeling and NAS/PSA benchmark evaluation. *IEEE Transactions on Parallel and Distributed Systems* 2007; **18**(5):621–636.
29. Rzadca K, Trystram D, Wierzbicki A. Fair game-theoretic resource management in dedicated grids. *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*, Brazil, May 2007; 343–350.
30. Bai X, Marinescu DC, Boloni L, Siegel HJ, Daley RA, Wang I-J. A macroeconomic model for resource allocation in large-scale distributed systems. *Journal of Parallel and Distributed Computing* 2008; **68**(2):182–199.
31. Buyya R, Abramson D, Venugopal S. The grid economy. *IEEE Special Issue on Grid Computing* 2005; **93**(3):698–714.
32. Khan SU, Ahmad I. A pure Nash equilibrium-based game theoretical method for data replication across multiple servers. *IEEE Transactions on Knowledge and Data Engineering* 2009; **21**(4):537–553.
33. Ahmad I, Ranka S, Khan SU. Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy. *IEEE International Symposium on Parallel and Distributed Processing*, Miami, FL, April 14–18, 2008; 1–6.
34. Khan SU, Ahmad I. A cooperative game theoretical replica placement technique. *International Conference on Parallel and Distributed Systems*, Hsinchu, Taiwan, December 5–7, 2007; 1–8.
35. Kolodziej J, Khan SU, Xhafa F. Genetic algorithms for energy-aware scheduling in computational grids. *2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Barcelona, October 26–28, 2011; 17–24.
36. Guzek M, Pecero JE, Dorronsoro B, Bouvry P, Khan SU. A cellular genetic algorithm for scheduling applications and energy-aware communication optimizatios. *2010 International Conference on High Performance Computing and Simulation (HPCS)*, Caen, France, June 28 – July 2, 2010; 241–248.
37. Xiao P, Hu Z. Workload-aware reliability evaluation model in grid computing. *Journal of Computers* 2012; **7**(1):141–146.
38. Abhishek V, Kash IA, Key P. Fixed and market pricing for cloud services. *2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Orlando, FL, March 25–30, 2012; 157–162.
39. Niyato D, Vasilakos AV, Kun Z. Resource and revenue sharing with coalition formation of cloud providers: game theoretic approach. *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Newport Beach, CA, USA, 2011; 215–224.
40. Stokely M, Winget J, Keyes E, Grimes C, Yolken B. Using a market economy to provision compute resources across planet-wide clusters. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2009)*, Rome, Italy, 2009; 1–8.

DOI: 10.1002/cpe

41. Grosu D, Das A. Auctioning resources in grids: model and protocols. *Concurrency and Computation: Practice and Experience* 2006; **18**(15):1909–1927.
42. Danak A, Manor S. Efficient bidding in dynamic grid markets. *IEEE Transactions on Parallel and Distributed Systems* 2011; **22**(9):1483–1496.
43. Sun D, Chang G, Wang C, Xiong Y, Wang X. Efficient Nash equilibrium based cloud resource allocation by using a continuous double auction. *2010 International Conference on Computer Design and Applications (ICCDA)*, Qinhuangdao, P. R. China, June 25–27, 2010; V1–94–V1–99.
44. Zhao X, Xu L, Wang B. A resource allocation model with cost-performance ratio in data grid. *8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, Qingdao, China, 2007; 371–376.
45. Duan R, Prodan R, Fahringer T. Performance and cost optimization for multiple large-scale grid workflow applications. *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, Reno, Nevada, USA, 2007; 1–12.
46. Owen G. *Game Theory*. Academic Press: Waltham, Massachusetts, 1982.
47. Winoto P, McCalla G, Vassileva J. An extended alternating-offers bargaining protocol for automated negotiation in multi-agent systems. *Proceedings of the 10th International Conference on Cooperative Information Systems*, Irvine, CA, USA, 2002; 179–194.
48. Larson K, Sandholm T. An alternating offers bargaining model for computationally limited agents. *AAMAS'02*, Bologna, Italy, 2002; 135–142.
49. Kleinrock L. *Queueing Systems - Volume 1: Theory*. John Wiley and Sons: Hoboken, NJ, 1975.
50. Fudenberg D, Tirole J. *Game Theory*. The MIT Press: Cambridge, MA, 1994.
51. Cubert R, Fishwick P. Sim++ reference manual. *Cise*, University of Florida, July 1995.
52. Jain R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley-Interscience: Hoboken, NJ, 1991.
53. Reklaitis GV, Ravindran A, Ragsdell KM. *Engineering Optimization: Methods and Applications*. Wiley-Interscience: Hoboken, NJ, 1983.

DOI: 10.1002/cpe