# A Combinatorial Auction Mechanism for Multiple Resource Procurement in Cloud Computing

G. Vinu Prasad, Abhinandan S. Prasad, *Member, IEEE*, and Shrisha Rao, *Senior Member, IEEE*

**Abstract**—In hybrid cloud computing, cloud users have the ability to procure resources from multiple cloud vendors, and furthermore also the option of selecting different combinations of resources. The problem of procuring a single resource from one of many cloud vendors can be modeled as a standard winner determination problem, and there are mechanisms for single item resource procurement given different QoS and pricing parameters. There however is no compatible approach that would allow cloud users to procure arbitrary bundles of resources from cloud vendors. We design the CLOUD-$\mathcal{CABOB}$ algorithm to solve the multiple resource procurement problem in hybrid clouds. Cloud users submit their requirements, and in turn vendors submit bids containing price, QoS and their offered sets of resources. The approach is scalable, which is necessary given that there are a large number of cloud vendors, with more continually appearing. We perform experiments for procurement cost and scalability efficacy on the CLOUD-$\mathcal{CABOB}$ algorithm using various standard distribution benchmarks like random, uniform, decay and CATS. Simulations using our approach with prices procured from several cloud vendors' datasets show its effectiveness at multiple resource procurement.

**Index Terms**—Cloud computing, combinatorial auction, cloud broker, dynamic pricing, linear programming, resource allocation

## 1 Introduction

CLOUD computing is a popular paradigm for offering services over the Internet [1], [2]. Currently, there are many companies like Amazon, salesforce.com, Microsoft, etc., that provide cloud services. These cloud vendors generally follow a fixed pricing strategy. Consider for example a user who wants to use a service in the form of a computing platform (PaaS) on a cloud. There are cloud vendors who provide versions of that platform at different prices, and with varying quality-of-service (QoS) parameters. The cloud user is charged based on the usage, but not based on the value derived from the service. This approach has its own limitations; for example, it cannot give the best services [3] considering the large numbers of cloud vendors who are available to provide services relating to a particular type of resource.

Most cloud vendors use the pay-as-you-go or fixed pricing model. Many vendors do not negotiate contracts, possibly for lack of understanding of the basis for and benefits of dynamic pricing. Any default agreement offered by the vendor may contractually benefit the vendor but not the user, resulting in a mismatch with user requirements [4]. Also, there often is no clear commitment on Service Level Agreements (SLAs) [4]. Dynamic pricing is the solution for these kind of problems [5]. Hence, procuring resources from the users' perspective is an important and interesting issue.

Some problems that are currently associated with fixed pricing are:

- Most often, the contracts in resource procurement favor cloud vendors. There might be instances where the requirements of both cloud vendors and cloud users are mismatched [4].
- SLAs are a very important aspect for enterprise customers, but it is very difficult to enforce SLAs given fixed pricing [4], [5].

Dynamic pricing [6], [7] overcomes these problems. The application of dynamic pricing in cloud computing is an interesting yet unexplored area [8].

Resource procurement is an important challenge in today's Internet, especially in large distributed systems like Grid, cloud, etc. Resource allocation is a very active area of research in Grid [9], [10], [11], [12].

Resource procurement can be accomplished using conventional or economic models. The conventional models [13], [14], [15] assume that resource providers are *non-strategic* (not seeking to maximize profit), whereas economic models assume that resource providers are *rational* and *intelligent*. In conventional methods, a user pays for the consumed service. In economic models, a user pays based on the value derived from the service. Hence economic models are more appropriate in the context of cloud computing.

The main strength of economic models is distributing incentives to the participants. But there are cases where the participants may not act truthfully. Hence, we assume that cloud vendors are selfish and rational. Also, the cloud broker performs reverse auctions [6] on behalf of the cloud user. [1]

- *G.V. Prasad is with the Viralmo Technologies Pvt. Ltd., #16/1, Meenakshi Nilaya, 2nd Cross, Tata Silk Farm, Basavanagudi, Bangalore 560 004, India. E-mail: vinu.prasad@iiitb.org.*
- *A.S. Prasad is with the Institut für Informatik, Georg-August Universität Göttingen, Goldschmidtstrasse 737077, Göttingen, Germany. E-mail: abhinandansp@ieee.org.*
- *S. Rao is with the International Institute of Information Technology-Bangalore, 26/C Electronics City, Hosur Road, Bangalore 560 100, India. E-mail: shrao@ieee.org.*

*Manuscript received 22 May 2015; revised 28 Feb. 2016; accepted 5 Mar. 2016. Date of publication 11 Mar. 2016; date of current version 5 Dec. 2018. Recommended for acceptance by B. Veeravalli.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TCC.2016.2541150*

1. See Parsons et al. [16] for a good introduction to auction theory from a computer science perspective.

With increased demand for cloud resources, especially for complex tasks requiring multiple resources, there has been an increased scope for disagreements between cloud service providers and cloud users. This has resulted in ineffective transactions between the two parties, which in turn results in suboptimal usage of cloud resources. We propose a resource procurement approach using combinatorial auctions and mechanism design, to address these issues.

Our work is an extension to Prasad and Rao [8], where a mechanism for resource allocation by the cloud broker among several cloud vendors bidding in the auction for a single resource is addressed. The work done by them is for procurement of a single resource at any given instance. Prasad and Rao assume that the cloud user requests a single resource at a time, and the vendor that wins the auction provides that resource. In practice, however, a cloud user may require a combination of resources which a single vendor may be unable to provide—this issue is not considered in their paper [8], which does not address the questions of how a user may require several resources together (rather than a single one at a time), and how a vendor may bid with a combination of resources rather than offering a single one.

To address these matters, we take into account multiple resources being considered for the auction by several cloud vendors at any instance of time. Multiple resource allocation is a combinatorial auction problem which has particular relevance in *hybrid cloud computing* which is little explored as of now, but is considered to be of importance in the future [17].

In real systems, neither the cloud vendor nor the cloud user anticipates the demand. Hence, there also is price uncertainty. Bichler et al. [18] state that uncertainty about the prices of goods, and little knowledge about market participants, are obstacles to dynamic pricing. Auctions are in particular helpful in this kind of situation [18], [19].

We propose a combinatorial-auction-based algorithm to implement dynamic pricing in hybrid clouds. Our algorithm CLOUD-$\mathcal{CABOB}$ deals with this, focusing on time complexity rather than incentive compatibility. CLOUD-$\mathcal{CABOB}$ allows dynamic pricing and does not reserve resources in advance, which overcomes drawbacks seen elsewhere [20]. Winner determination can be scaled down to one of the standard combinatorial auction problems, and one such scaling is deployed in our work.

In illustration, we perform combinatorial auctions with three resources: RAM, storage, and CPU power. Our results show that in sequential auctions, the cost per resource increases sequentially as the number of resources to be procured increases. In case of combinatorial auctions, as the number of resources to be auctioned increases, the combinatorial auction mechanism provides the best price for the requested set of resources.

Since the domain in which we suggest the use of our approach is cloud computing, the scalability of our approach is a pre-eminent issue. We have analyzed the time efficiency of our implementation on scaling our approach to large user requirements using random, uniform, decay and CATS distributions [21]. We have performed simulations with the parameters proposed by Sandholm et al. [22], and recorded the results. The primary advantage of our approach in cloud computing is that it is highly scalable and very fast in winner determination. The time taken in winner determination is of the order of $\mathcal{O}(|V| + |E|)$.

In a federated or hybrid cloud [17], the user has the choice to choose from different cloud vendors for the required resources, and cloud vendors are left to co-ordinate among themselves. Hence, the procurement module presented elsewhere [8] cannot be applied in this case. In our case, the cloud user has the option of procuring resources as a set of items from different cloud vendors. Hence, combinatorial auctions are appropriate for this context.

In combinatorial auctions, winner determination is a non-trivial task [23]. In real cloud systems, there are also expected to be a large number of cloud vendors. Hence, devising a scalable solution for performing combinatorial auctions in a cloud is non-trivial and interesting.

The set of bids is represented as tree nodes. Each tree node is then labeled as either *winning* or *losing*. The tree is searched using depth first search (DFS). Using heuristics, the contribution of unallocated items is calculated. This contribution along with the revenue generated from bids is used to decide whether to include a bid in the set of best solutions.

Before submitting the bids to the CLOUD-$\mathcal{CABOB}$ algorithm, we perform a *preprocessing* step to normalize the bids that are produced by the cloud vendors. By doing this, each bid has integer values associated with it for each resource being bid for.

In the initial step, the set of resources are divided such that no bid includes resources from more than one subset. The winner is determined separately in each subset to speed up the search. CLOUD-$\mathcal{CABOB}$ uses an upper threshold on the revenue that the unallocated resources can contribute. If the current solution is not better than the optimal solution, CLOUD-$\mathcal{CABOB}$ prunes the search path. We use a linear programming (LP) formulation for estimating the upper threshold. After estimating the upper threshold, we apply an integer relaxation where we can either accept the bid completely, or else reject it completely.

Our deployment enables the end user to automate the multiple resource selection process and scale the same for large resource requests. Our work would help a cloud broker decide the best set of cloud vendors who can service user requests. This aspect of intelligent resource allocation in a cloud was heretofore not explored in great detail, and ours is the first effort to accomplish the same. We consider cloud resource offerings from different cloud vendors, and tend to believe as likely a future scenario where standardization and interoperability between vendors are widespread, as suggested by Rochwerger et al. [17].

CloudSim [24] is a well-known simulation tool for cloud applications, but it does not support auction protocols [25]. Hence, we implemented the proposed approach using a standard cloud vendors dataset [26] based on user requests, and found that the winner determination for combinatorial auctions in cloud computing can be achieved by maximizing the profit to the cloud vendors while at the same time providing the best bid of requested resources to the end user. Our work also gives end users the luxury of just having to place their resource requests without being concerned with the mechanism of procuring them. The cloud broker performs auctions in the hybrid cloud environment, and

provides the requested resources at the best possible price and Quality of Service to the end user.

The rest of the paper is organized as follows. Section 2 briefly presents the background work done in this area. Section 3 describes the system model adopted in our work. Section 4 elaborates on the adopted mechanism to implement combinatorial auctions in cloud computing. Section 5 discusses the implementation, accuracy, and efficiency of the proposed model, we conclude with Section 6.

## 2 Related Work

### 2.1 Resource Allocation in Grid and Cloud

Resource allocation is an important challenge in today's Internet, especially in large distributed systems like Grid, cloud, etc. Resource allocation is a very active area of research in Grid [9], [10], [11], [12]. These resources are owned by the companies and are mostly distributed geographically. Resource allocation algorithms can be either centralized or decentralized. Centralized algorithms [13], [14], [15] require global knowledge and complete information in real time, which is rather unrealistic for most large distributed systems. The cost models of the centralized algorithms derive costs based on the usages of the resources.

Economic models for resource allocation use decentralized algorithms. These models derive cost based on the value a user derives from the service [12]. Most resource allocation algorithms based on economic models rely on single market mechanisms. Vilajosana et al. [27] develop a configurable auction server which gives the ability to configure markets dynamically. Buyya et al. [12] use economic models like commodity market, posted price, etc., for developing a grid resource broker for resource management.

Lin et al. [28] use dynamic auctions (based on the Vickrey auction) to perform resource allocation. Cloud users bid for resources and the highest bidder wins the auction. The winner pays the *second*-highest bid. Zaman and Grosu [29] design an auction-based mechanism to perform dynamic virtual machine provisioning and allocation based on the user demand during virtual machine provisioning.

### 2.2 Combinatorial Auctions

Parsons et al. [16] give a comprehensive introduction to auction theory, including various types of auctions, their characteristics, and their applications in computing.

Combinatorial auctions are a kind of auction where the auctioneer invites bids on combinations of items, rather than on single items as in conventional auctions. Vohra and Vries [30] survey the state of the art about combinatorial auction design and also discuss the integer programming required for designing combinatorial auctions. Combinatorial auctions are unfortunately an $\mathcal{NP}$-complete problem in the general case. Fujishima et al. [31] propose two methods to deal with the general case. In the first, a search space is structured such that depth first search avoids conflicting bids. This method reduces running time. The second method is a heuristic, market-based approach wherein a virtual agent places bids for each good in a virtual multi-round auction. Hoos and Boutilier [32] propose a stochastic local search algorithm to solve combinatorial auctions. Nisan [33] not only formalizes bidding languages but also compares their strengths. He also proves that the linear programming approach leads to optimal allocation if prices are attached to single items in the auction.

Sandholm [34] is possibly the first significant work on winner determination in combinatorial auctions. He presents a generic algorithm that allows combinatorial auctions to scale up to significantly larger numbers of items and bids than prior approaches to optimal winner determination [35], by capitalizing on the fact that the space of bids is sparsely populated in practice. This also presents the fact that basic combinatorial auctions only allow bidders to express complementarities of items.

Sandholm and Suri [36] present an algorithm that *branches on bids* to determine the winner in combinatorial auctions. This algorithm is also called the BOB algorithm. In the BOB algorithm, bids are represented as tree nodes and DFS is performed on this tree to determine the winner. Unfortunately, the BOB algorithm is not easy to implement in its basic form. So Sandholm et al. [22] suggest an improved version of BOB called the CABOB algorithm. CABOB is one of the fastest optimal algorithms for winner determination in combinatorial auctions, and is a basis of our work.

## 3 System Model

Our system model is based on Prasad and Rao [8]. A cloud user has resource requirements. A user wishes to perform a *reverse auction* for procuring resources (also called a *procurement auction*), and this is actually done by a cloud broker to reduce the load on the user. Cloud vendors are ready to offer the resources. The goal of the cloud user is to minimize the total cost of procuring resources, without compromising on the quality of service that is sought. In order to minimize the procurement cost, it is necessary for the cloud user to know the real cost of the cloud vendor.

The cloud users indicate their requirements to the cloud broker. The cloud broker passes the information on these requirements to all cloud vendors. The cloud vendors decide whether to participate in the auction based on this information, and submit bids if they are participating. We assume that cloud vendors are rational and intelligent, and that they might bid with false valuations to maximize their utility.

The cloud vendors are represented by $N = \{1, 2, \ldots, n\}$. In this procurement auction, each cloud vendor responds by bidding with price $p_i$ and promised $k$ QoS attributes $q_{i1}, q_{i2}, \ldots, q_{ik}$. We normalize the QoS attributes and let $q_i$ be the normalized QoS value of the $i$th cloud vendor.

The auctioneer has a set of resources, $M = \{1, 2, \ldots, m\}$, to be procured. Let $B_i$ be the bid of $i$th cloud vendor and $B_i = \langle S_i, p_i, q_i \rangle$ where $S_i \subseteq M$ is a set of resources provided by the cloud vendor.

One of the important assumptions made is that in the combinatorial auction, if the end user gets the combination of bids at a cheaper rate with the desired QoS when compared to obtaining the resources individually, he tends to go in for the bid even though he might not have got all of his requested resources in that particular bid.

Similarly, from the cloud vendor's perspective, if the cloud vendor can sell more resources by including more resources in the bid at a desired rate and desired QoS, it proceeds to bid

TABLE 1
AHP Score Table

| QoS Specification | Score |
|---|---|
| Main memory $<$ 1 GB | 1 |
| Main memory $>$ 1O GB | 4 |
| Latency $<$ 10 ms | 5 |
| Latency $>$ 20 ms | 3 |
| SLA $>$ 24 hrs | 2 |
| SLA $<$ 10 hrs | 4 |

with such resources and holds the remaining resources back for maximizing its profit, which is the cost of the bid coupled with the summation of the costs of all the remaining resources that it is left with. With these assumptions, the cloud broker proceeds to perform the bidding to determine the set of cloud vendors winning in the auction. In the real world, there are a very large number of cloud vendors offering different resources. Hence, possible numbers of total resource combinations are also huge. At the same time, it is very important to determine the winner in near real-time (as cloud users are unlikely to accept delays in procurement).

### 3.1 QoS Scaling

The QoS scaling aspect of the model is also from Prasad and Rao [8], which we summarize for completeness. Every cloud vendor provides different resources with different QoS levels. Hence, the QoS parameters are not alike for all cloud vendors. Also, in case of bidding for several resources, the QoS may differ for the set of resources compared against the same resources provided individually. QoS parameters can be either *positive* or *negative*. If smaller values indicate higher quality, then such QoS parameters are called negative, otherwise they are called positive. The comparison of different QoS parameters is a common problem in multiple criteria decision making.

Zeng et al. [37] use a well-known method called Simple Additive Weighting (SAW) to perform comparison of quality attributes of a web service. This, coupled with the work of Mohabey et al. [38], can be used to scale and normalize QoS to perform combinatorial procurement auctions of cloud services. We use the SAW method to perform QoS normalization.

The QoS parameters obtained from cloud vendors are often textual, like "99 percent uptime," etc. The SAW technique works only if the values of QoS parameters are integers. Hence, it is necessary to assign a suitable value for every QoS attribute. The Analytic Hierarchy Process (AHP) [39] is a well-known technique used in these kinds of complex situations.

Using AHP, scores are assigned to QoS parameters based on user criteria. Let $\gamma_{i,j}$ be the $k$th resource offered by cloud vendor $i$, and let $\delta_{\gamma_{(i,k)},j}$ be the QoS parameter $j$ offered by the cloud vendor $i$ for the resource $k$. This $\delta_{\gamma_{(i,k)},j}$ is determined using AHP.

We then construct a matrix $\Gamma = \{\gamma_{i,j}; 1 \leq i \leq n, i \leq j \leq k\}$. Each row of this matrix $\Gamma$ corresponds to the QoS parameters of cloud vendor $i$ for the subset of resources $k$. Let $\alpha_{i^{\max},k^{\max}}$ and $\alpha_{i^{\min},k^{\min}}$ be the maximum and minimum value of the QoS value of the cloud vendor $i$.

TABLE 2
QoS Parameters and Weights

| QoS Parameter | Weight |
|---|---|
| Bandwidth | 0.2 |
| Latency | 0.3 |
| SLA | 0.5 |

Let $B = \beta_{\gamma_{(i,k)},j}; 1 \leq i \leq n\}$ be the matrix and $\beta_{\gamma_{(i,k)},j}$ the normalized value of the QoS parameter $\delta_{\gamma_{(i,k)},j}$.

The SAW method involves two stages. They are:

1) Scaling: The QoS parameters can be either positive or negative. Hence, they are scaled differently.
The negative values are scaled using (1),

$$\beta_{i,j} = \begin{cases} \frac{\alpha_i^{\max} - \delta_{\gamma_{(i,k)},j}}{\alpha_i^{\max} - \alpha_i^{\min}} & \text{if } \alpha_i^{\max} \neq \alpha_i^{\min} \\ 1 & \text{otherwise.} \end{cases} \quad (1)$$

The positive values are scaled using (2)

$$\beta_{i,j} = \begin{cases} \frac{\delta_{\gamma_{(i,k)},j} - \alpha_i^{\min}}{\alpha_i^{\max} - \alpha_i^{\min}} & \text{if } \alpha_i^{\max} \neq \alpha_i^{\min} \\ 1 & \text{otherwise.} \end{cases} \quad (2)$$

2) Weighting: In this stage, the final score $q_i$ is computed with the score computed using (3),

$$score = \sum_{j=1}^{k} \beta_{i,j} \cdot w_k, \quad (3)$$

where $w_k \in [0,1]$ and $\sum_{j=1}^{k} w_j = 1$.
Let $s_f$ be the scaling factor and the final $q_i = score \cdot s_f$.

In this way, the QoS parameters are reduced to a single number. This summarization is necessary to enable comparison of QoS between cloud vendors. This number is used along with the cost in the procurement auction. We illustrate QoS scaling using a numerical example. In the example of Table 1, we consider a few common QoS parameters and assign scores arbitrarily for illustration.

Table 2 shows the weights, likewise chosen arbitrarily for illustration, associated with some QoS parameters.

Consider a user request for a subset of resources and that a cloud vendor responds with specifications for one of the subset of requested resources as: Latency 30 ms, SLA = 6 hours and 200 kbps bandwidth. Let $s_f = 10$. Therefore, the final score may be computed as: $v_{11} = 0.5, v_{12} = 1$ and $v_{13} = 1$. Then $score = (0.5 \cdot 0.3 + 1 \cdot 0.5 + 1 \cdot 0.2) \cdot 10 = 8.5$.

### 3.2 Linear Programming Formulation

The binary combinatorial auction winner determination problem is to label the bids as winning or losing, so as to maximize the auctioneer's revenue [7] under the constraint that each resource can be allocated to at most one bidder represented by (5). But just labeling them as winning or losing cannot suffice, since doing so is a well known $\mathcal{NP}$-complete problem [40]. In our case, the cloud user (or a broker or agent acting on its behalf) conducts the auction, and the goal is to minimize the procurement cost.

$$\min \sum_{j=1}^{n} p_j \cdot x_j \cdot q_j, \tag{4}$$

$$such\ that \sum_{j \mid i \in S_j} x_j \leq 1, \quad i \in \{1, \ldots, m\}\ and\ x_j \in \{0, 1\}. \tag{5}$$

Hence a slight variation of that, where bids could be accepted partially, can be considered. The problem would now become a linear program, which can be solved in polynomial time. Here we consider the LP formulation and its dual represented by (7) and (9), which is used in several places in our algorithm,

LP:

$$\min \sum_{j=1}^{n} p_j \cdot x_j \cdot q_j, \tag{6}$$

$$such\ that \sum_{j \mid i \in S_j} x_j \leq 1, \quad i \in \{1, \ldots, m\}\ and\ x_j \in \mathbb{R}. \tag{7}$$

Dual:

$$\max \sum_{i=1}^{m} y_i \cdot z_i, \tag{8}$$

$$such\ that \sum_{i \in S_j} y_i \geq p_j, \sum_{i \in S_j} z_i \geq q_j\ and\ (y_i, z_i) \in \mathbb{R}. \tag{9}$$

In the above LP formulation, $x_j$ is a function of $p_j$ (the price quoted by the vendor) and $q_j$ (the normalized QoS for the set of resources placed in the bid). In single resource allocation, $x_j$ is formulated based on the price and the QoS of the item to be procured. But, since here we are dealing with multiple resource procurement, we need to consider the price and the QoS factors for the whole set of items under consideration. This turns out to be a combinatorial auction problem, which can be formulated with the help of an LP where $x_j$ is a function of both $p_j$ and $q_j$, as shown in (7).

In combinatorial auctions, $x_j$ is either 0 or 1, indicating whether an item is included or not. Here, we intend to maximize the procurement of the requested resources with the best possible cost and QoS parameters. This becomes our objective function where the constraints for the LP formulation are given in (7). The dual of this formulation is the shadow price [41]. Hence, we need to calculate only one LP whose dual forms the upper and lower bounds of the CLOUD-$\mathcal{CABOB}$ algorithm. This increases the computational efficiency significantly, since we cannot deduce the upper bounds based on the lower bounds calculated using the LP itself.

In the dual formulation, the shadow price $y_i$ gives the price for each individual resource $i$ and the shadow price $z_i$ gives the QoS for the individual resource $i$. In the binary case, individual resources cannot generally be given prices, but each $y_i$ and $z_i$ value from the dual gives upper bounds on the price and QoS of resource $i$, which is used later in the algorithm.

All the relevant notation used is summarized in Table 3.

TABLE 3
Notation

| Symbol | Description |
|---|---|
| $n$ | Number of cloud vendors |
| $N$ | A set of cloud vendors, $\{1, 2, ..., n\}$ |
| $m$ | Number of resources |
| $M$ | A set of resources, $\{1, 2, \ldots, m\}$ |
| $B_i$ | Bid submitted by cloud vendor $i$ |
| $B$ | A set of bids, $\{B_1, B_2, \ldots, n\}$ |
| $S_i$ | Set of resources provided by cloud vendor $i$ |
| $p_i$ | Price quoted by cloud vendor $i$ |
| $q_i$ | Normalized QoS of the cloud vendor $i$ |
| $G$ | Bid Graph |
| $C$ | Set of bid graph components |
| $\epsilon$ | Number of bid graph components |
| $V$ | Number of vertices in bid graph $G$ |
| $E$ | Number of edges in bid graph $G$ |

## 4 CLOUD-$\mathcal{CABOB}$ ALGORITHM

There is no polynomial-time algorithm to solve winner determination for combinatorial auctions. Equation (7) is a well known *winner determination* problem and is $\mathcal{NP}$-complete. In one approach, approximation algorithms are used [36]. These approximation algorithms do not guarantee optimal solutions, but in special cases lead to good solutions.

Another approach is to restrict allowable bids [36]. Even though there are some restrictions under which we can solve in polynomial time, doing so leads to economic inefficiencies. So Sandholm and Suri [42] propose an algorithm to solve the unrestricted winner determination problem using search. This algorithm is popularly called the Branch on Bids (BOB) algorithm.

The set of bids are represented as tree nodes, which are labeled as either winning ($x_j = 1$) or losing ($x_j = 0$). The tree is searched using DFS. Using heuristics, the contribution of unallocated items is calculated. This contribution along with the revenue generated from bids is used to decide whether to include a bid in the best solution set. This is the main idea of the BOB algorithm.

In BOB, there is an one-to-one correspondence between tree leaves and feasible solutions, unlike branch-on-items algorithms where not every feasible solution is represented by any leaf. However, BOB was not implemented fully, though several attempts were made to implement it.

Our algorithm CLOUD-$\mathcal{CABOB}$ facilitates combinatorial auctions in cloud computing environments. It is based on CABOB [42], a depth-first branch-and-bound tree search that branches on bids which incorporates many of the techniques proposed in BOB and other algorithms.

Before submitting bids to the CLOUD-$\mathcal{CABOB}$ algorithm, we perform a *preprocessing* step to normalize the bids that are produced by the cloud vendors. Since each bid is a tuple, we submit a simple weighted sum of the cost and QoS parameters of each and every resource in the tuple. The weighted sum is defined by $I_i = q_i + (S_f \cdot c_i)$, where $I_i$ is a constant which is the weighted sum of cost and QoS of bid $i$, and $S_f$ is the scaling factor for the cost of the bid $i$. By doing this, each bid has integer values associated with it for each resource it is bidding for. Algorithm 1 gives the detailed pseudocode.

**Algorithm 1.** CLOUD-$\mathcal{CABOB}$ ($G$, $g$, $min$)

**Input:** Bid Graph $G$, revenue generated from winning bids $g$, minimum revenue $min$ per CLOUD-$\mathcal{CABOB}$
**Output:** Set of winning bids $F_{opt\_solved}$

1 **if** $|E| = \frac{n(n-1)}{2}$ **then**
2  $f_{opt} \leftarrow \max B$ ;
3  return $f_{opt}$ ;
4 **end**
5 **if** $|E| = 0$ **then**
6  Accept all the remaining bids;
7  update $f_{opt}$ and return $f_{opt}$;
8 **end**
9 `FindConnectedComponents`$G, C$ ;
10 $\alpha \leftarrow |C|$ ;
11 // $\epsilon$ is the number of components
12 **for** $i \leftarrow 1$ **to** $\epsilon$ **do**
13  calculate an upper threshold $(\mathrm{UT})_i$;
14 **end**
15 **if** $\sum_{i=1}^{\epsilon}(UT)_i \leq min$ **then**
16  return 0;
17 **end**
18 Apply *Integer Relaxation*;
19 **for** $i \leftarrow 1$ **to** $\epsilon$ **do**
20  calculate lower threshold $(\mathrm{LT})_i$;
21 **end**
22 $\triangle \leftarrow g + \sum_{i=1}^{\epsilon}(LT)_i - f_{opt}$;
23 **if** $\triangle > 0$ **then**
24  $f_{opt} \leftarrow f_{opt} + \triangle$; $min \leftarrow min + \triangle$ ;
25  **end**
26 **if** $n < 1$ **then**
27  Choose next bid $B_k$ to branch on ;
28  $f_{opt\_old} \leftarrow f_{opt}$; $f_{in} \leftarrow$ CLOUD-$\mathcal{CABOB}(G, g + p_k, min - p_k)$;
29  $min \leftarrow min + (f_{in} - f_{opt\_old})$;
30  $\forall B_j$ s.t. $B_j \neq B_k$ and $S_j \cap S_k \neq \emptyset$, $G \leftarrow G \cup B_k$;
31  $f_{opt\_old} \leftarrow f_{opt}$; $f_{out} \leftarrow$ CLOUD-$\mathcal{CABOB}(G, g, min)$;
32  $min \leftarrow min + (f_{in} - f_{opt\_old})$;
33  Return $max(f_{in}, fout)$;
34 **end**
35 $F_{opt\_solved} \leftarrow 0$; $H_{unsloved} \leftarrow \sum_{i=1}^{c}(UT)_i$;
36 $L_{unsloved} \leftarrow \sum_{i=1}^{c}(LT)_i$;
37 **for** each component $c_i \in C$ **do**
38  **if** $F_{opt\_solved} + H_{unsloved} \leq min$ **then**
39  return 0;
40  **end**
41  $t'_i \leftarrow F_{opt\_solved} + (L_{unsloved} - (LT)_i)$;
42  $f_{opt\_old} \leftarrow f_{opt}$;
43  $f_{opt\_i} \leftarrow$ CLOUD-$\mathcal{CABOB}(G_i, g + t'_i, min - t'_i)$;
44  $min \leftarrow min + (f_{opt\_old} - f_{opt})$;
45  $F_{opt\_solved} \leftarrow F_{opt\_solved} + f_{opt\_i}$;
 $H_{unsloved} \leftarrow H_{unsloved} - H_i$;
46  $H_{unsloved} \leftarrow H_{unsloved} - H_i$;
47 **end**
48 return $F_{opt\_solved}$

To begin with, the set of resources offered by a vendor is partitioned into pairwise-disjoint subsets. The winner is determined separately in each subset to hasten the search. At each search node, Algorithm 1 uses a data structure called the *bid graph*, denoted by $G$. The nodes of bid graph $G$ represent the bids of unallocated resources. Two nodes in $G$ share an edge whenever the corresponding bids share resources. Let $V$ be the set of vertices of $G$, and $E$ be the set of edges. At any point of time, $|V| \leq n$ and $|E| \leq \frac{n(n-1)}{2}$.

Let $f_{opt}$ be the value of the best solution found so far, as a global variable. We define $min$ as the minimum revenue the cloud vendor expects at the end of the auction, and $g$ as the revenue returned at a particular iteration on running the function CLOUD-$\mathcal{CABOB}$. We start searching by invoking CLOUD-$\mathcal{CABOB}(G, 0, 0)$.

Initially, the bid graph $G$ is empty and $f_{opt}$ is zero. We construct the bid graph $G$ incrementally by adding bids $B_i$. We call Algorithm 2 to find the components of the bid graph $G$.

**Algorithm 2.** FindConnectedComponents($G$, $C$)

**Input:** Bid Graph $G$
**Output:** Set of components $C = \{c_1, c_2, \ldots, c_n\}$

1 // `DFS annotates each vertex with discover and finishing time`
2 `DFS`($G$) ;
3 // In undirected graph $G$, $G^T = G$
4 // Consider vertices in decreasing finishing time
5 `DFS`($G$) ;
6 Vertices in each tree of the depth-first forest is a separate component ;

Algorithm 2 is a standard algorithm [43]. First we run DFS and annotate each vertex of $G$ with discover and finish times. Afterward, we compute the transpose graph and perform DFS according to the decreasing order of finishing times of the vertices. The vertices of the DFS forest are the separate components of the graph. In an undirected graph, the transpose is the very same graph itself. Hence, in line 2 of Algorithm 2, we perform DFS on the same graph twice. The time complexity of Algorithm 2 is $\Theta(|V| + |E|)$. We run Algorithm 2 on $G$, which results in $k$ independent graphs. In line 12, CLOUD-$\mathcal{CABOB}$ uses an upper threshold on the revenue the unallocated resources can generate. If the current solution is not better than the optimal solution $f_{opt}$, CLOUD-$\mathcal{CABOB}$ prunes the search path. We use an LP formulation for estimating the upper threshold.

The main aim of using the upper threshold is to speed up the search path pruning without affecting the optimality.

Before starting the LP, one could look at the condition in line 14 to determine the minimum revenue the LP has to produce so that the search branch would not be pruned. Once the LP solver finds a solution that exceeds the threshold, it could be stopped without pruning the search branch. If the LP solver does not find a solution that exceeds the threshold and runs to completion, the branch could be pruned. However, CLOUD-$\mathcal{CABOB}$ always runs the LP to completion, since it uses the solutions from the LP and the dual in several ways.

After estimating the upper threshold, we apply an integer relaxation where we can either accept the bid completely or reject it completely, as is shown in line 17. Partial acceptance is not possible, by the very nature of combinatorial auctions. A special case can be noticed where a single cloud vendor gives an exclusive offer of providing all the resources with a good cost tradeoff.

CLOUD-$\mathcal{CABOB}$ calculates a lower threshold on the revenue that the remaining resources can contribute, as shown in line 21. If the lower threshold is high, it can allow $f_{opt}$ to be updated, leading to more pruning and less search in the

subtree rooted at that node. Any lower thresholding technique could be used here. We use the following rounding technique. CLOUD-$\mathcal{CABOB}$ solves the remaining LP, which gives an *acceptance level* $x_j, 0 \leq x_j \leq 1$, for every remaining bid $B_j$. We insert all bids with $x_j \geq 0.5$ into the lower-threshold solution. We then try to insert the rest of the bids in decreasing order of $x_j$, skipping bids that share resources with bids already in the lower threshold.

Based on the value of the lower bound obtained, we calculate the value of the increment, which is nothing but the difference of the sum of current revenue obtained and the sum of the lower bounds and the current $f_{opt}$. If this is greater then zero, then we update the values of $f_{opt}$ and $min$ as shown in line 23. If the number of independent subgraphs is less than 1, we choose the next bid to branch upon, and update the values of $f_{opt}$ and $min$ accordingly. Finally, for each of the subgraphs that is being obtained, we recursively call CLOUD-$\mathcal{CABOB}$ to obtain the best auction results and declare a set of cloud vendors as the winners. This can be seen in lines 28 through 50.

After each iteration, we check whether the solution obtained covers most if not all of the requested resources from the cloud vendors. Then for each of the resources that is not being procured, we update the values of $min$ and $f_{opt}$, and recursively call CLOUD-$\mathcal{CABOB}$ as shown in lines 26 through 45. Finally, the set of winning cloud vendors is returned in line 47.

Our algorithm does not make copies of the LP table, but incrementally adds (or: deletes) rows from the LP table as bids are removed (or: re-inserted) into $G$ as the search proceeds down a path (or: backtracks). Hence, it has linear time complexity. The proof of CABOB's linear time complexity can be found in [42]. Therefore, the implementation of our algorithm runs in linear time.

## 5 Results

The current state-of-the-art in cloud resource pricing is that each cloud vendor follows a different price distribution structure. In this kind of scenario, winner determination and cost computation using Vickrey auctions are not optimal [7]. Hence, this approach cannot be followed in real-world cloud resource procurement auctions. We performed several experiments to show the cost effectiveness of combinatorial auctions over sequential auctions, by simulating a sample cloud requirement scenario. We also analyzed the time efficiency of our approach on scaling the user needs and applying various standard distributions benchmarks.

Currently, there is a need for a standard toolkit for evaluating combinatorial auctions in clouds. CloudSim is a popular simulation software for cloud applications, but supports neither auction protocols nor price generation [25]. Other popular cloud environments like Eucalyptus, etc., do not provide for price simulation. Hence, we implemented our simulation using Java based on the techniques presented in this work, without compromising on needed cloud properties.

In order to simulate cloud resource prices, we adopt the following strategy. Different cloud vendors use different prices depending on the resource. Cloudorado [26] is a popular web service statistics provider from several vendors of cloud resources that provide re-sizable computing capability in the cloud. We performed distribution fitting for these prices to find the probability distribution of the price. Cloud vendors' yearly contract price is seen to be in the interval of [57, 8000] and is lognormally distributed. There is no standard distribution that can be adopted for distribution of QoS. Hence the approach adopted by Prasad and Rao [8] is being used here as well.

TABLE 4
Resource Procurement Costs

| Resource | Price |
|---|---|
| 1 GB RAM | \$37 |
| 2 GB RAM | \$58 |
| 2 GB HDD Storage | \$28 |
| 5 GB HDD Storage | \$33 |
| 10 GB HDD Storage | \$35 |
| 20 GB HDD Storage | \$41 |
| 40 GB HDD Storage | \$50 |
| 1vCPU | \$106 |
| 2vCPU | \$138 |
| 4vCPU | \$174 |
| 1 GB RAM + 2 GB HDD Storage + 1vCPU | \$112 |
| 2 GB RAM + 5 GB HDD Storage + 2vCPU | \$124 |
| 10 GB HDD Storage + 4vCPU | \$200 |
| 20 GB HDD Storage + 40 GB HDD Storage | \$80 |

One important assumption to be made is that we mix the offerings from several cloud vendors. This is expected to become applicable in future where standardization and interoperability between cloud vendors are widespread. This kind of approach facilitates both the cloud vendors and the buyer who is auctioning for the resources. The vendor's total profit would then be a combination of costs of sold resources, along with the resources unsold in the auction. The buyer on the other hand gets the combination of several resources at a lower price compared to what the buyer would have paid on buying them individually.

### 5.1 Resource Procurement Costs

For simplicity of simulation, we performed combinatorial auctions with three resources: RAM, storage, and CPU power. With these three resources being requested by the user, the cost at which these are being provided is shown below, extracted at a particular instant of time.

Consider a sample simulation of multiple resource procurement under sequential and combinatorial auctions. In this simulation, the user specifications are as follows: RAM of capacities 1 and 2 GB [44], storage of capacities 2, 5, 10 and 20 GB, CPUs of type 1, 2 and 4 vCPU. For this request of the user, we performed both sequential auctions and combinatorial auctions. The cost details for the procurement of these resources are given in Table 4 below.

When we performed the simulations on the above-mentioned user requests by simulating the values for procurement costs versus the number of items in auction, we obtained two curves as shown in Fig. 1. In Fig. 1, we observe that in the curve for the sequential auctions, the cost per resource increases sequentially as the number of resources to be procured increases. This is primarily because, as the number of resources to be procured increases, it adds on sequentially based on the best price available for that resource. It

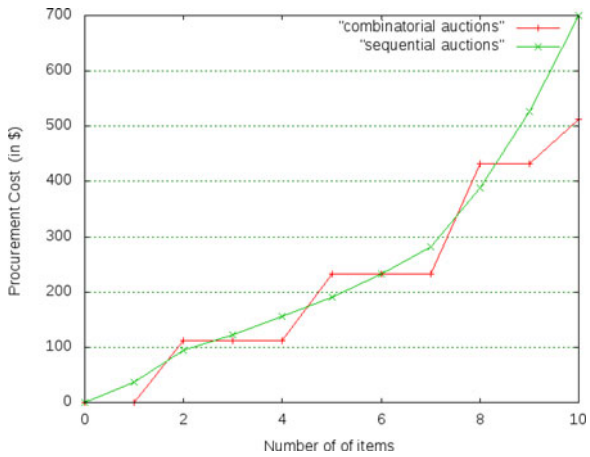


Fig. 1. Comparison of procurement costs of sequential and combinatorial auctions.

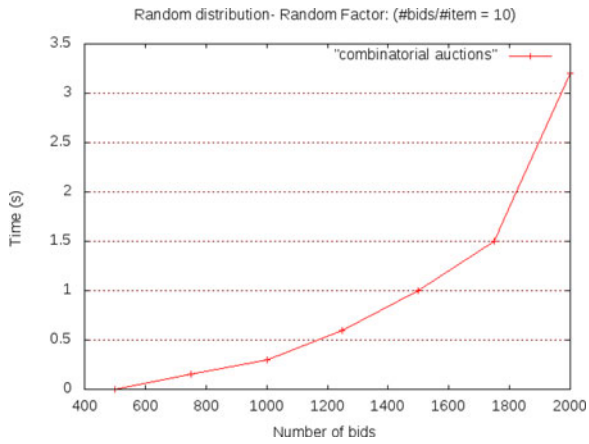


Fig. 2. Time estimation over random distribution.

turns out to be a single resource procurement problem [8], performed multiple times. Hence the total cost of procurement for this request turns out to be \$700. This proves to be less beneficial compared to resource procurement performed through a combinatorial auction mechanism, where multiple resources that the user requests are procured at a lesser price than the sum of their individual prices.

The curve for the combinatorial auctions shows the effectiveness of combinatorial auctions in cloud resource procurement. As we can see in Fig. 1, as the number of resources to be auctioned increases, the combinatorial auction mechanism provides the best price for the requested set of resources. This is aided by the fact that as the number of bids the cloud vendor wins increases, the greater is the number of resources that are sold compared to that of a sequential auction, where only one resource is considered at any instant of time. Hence, this is a trade-off where both the cloud vendor and the user requesting for the resource benefit. The total procurement cost using the combinatorial auction turns out to be \$516, which is more economical compared to the cost obtained by sequential auctions. (The detailed calculation of this is being omitted as tedious but uninformative, since it runs through recursive calls as explained in the algorithm.)

On comparing the resource procurement costs between sequential auctions and combinatorial auctions, we can observe that as the number of resources requested by the user increases, combinatorial auctions are far superior to sequential auctions, as shown in Fig. 1.

## 5.2 Time Efficiency

We have analyzed the time efficiency of our implementation on scaling our approach to large user requirements. We tested the scalability of CLOUD-$\mathcal{CABOB}$ on prominent combinatorial auction benchmark distributions—we considered the combination of distribution benchmarks given by Sandholm et al. [22], [34], and CATS [21], and recorded the following observations.

The following show the scalability of the CLOUD-$\mathcal{CABOB}$ approach in the field of cloud computing across various distributions. The experimental setup deployed by us was similar to the one in prior work [22].

### 5.2.1 Random Distribution

In this scenario, the requested resources for the user are picked randomly from several vendors. The user randomly chooses as many resources as can be supplied without replacement.

For the user specifications mentioned above, when we applied the suggested random distribution for sequential and combinatorial auctions, we observed that as the number of items requested increases, the number of bids required to satisfy a request increases, but the corresponding increase in the procurement cost is relatively low in the case of combinatorial auctions. This is mainly because the preprocessor of CLOUD-$\mathcal{CABOB}$ reduces a large number of bids.

We used the random distribution used in [34] in our simulated implementation where the ratio of number of bids to number of items is 10. This can be seen in Fig. 2 and Table 5, where as we increased the number of bids to as high as $2,000$ based on user requests, the procurement delay remained close to two seconds.

### 5.2.2 Uniform Distribution

In the uniform distribution, each cloud vendor offers the same number of items, where the items procured are mutually exclusive. As the number of items requested by the user increases, if we increase the number of items offered per vendor, the total procurement delay can be reduced significantly in combinatorial auctions.

This can be witnessed in Fig. 3 and Table 6, where there was an initial delay for the procurement of up to three items, but as we increased the number of items per bid, the

TABLE 5
Time Estimation of CLOUD-$\mathcal{CABOB}$ Using Random Distribution

| Number of Bids | Execution Time (in ms) |
|---|---|
| 750 | 0.15 |
| 1,000 | 0.3 |
| 1,250 | 0.6 |
| 1,500 | 1 |
| 1,750 | 1.5 |
| 2,000 | 3.2 |

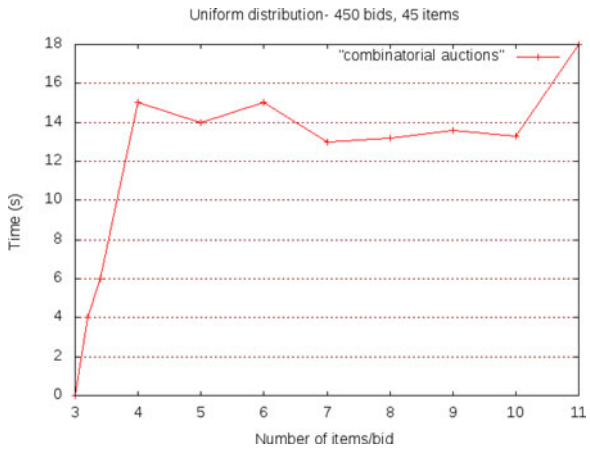


Fig. 3. Delay estimation over uniform distribution.

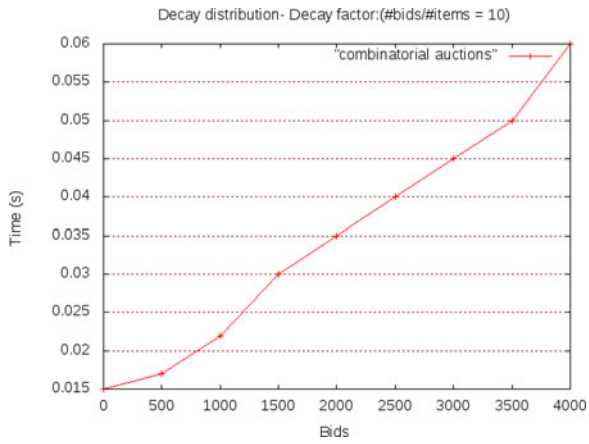


Fig. 4. Time estimation over decay distribution.

procurement delay remained nearly onstant. This suggests that the number of items per bid is mostly independent of time (except for the initial procurement delay) which makes our approach robust.

### 5.2.3 Decay Distribution

In case of decay distribution, a random item is selected from each vendor who is bidding in the auction. Then the user continues to select a new random item with a probability of decay constant, until an item is not already procured or all the requested resources are being serviced.

As a result, as we increase the number of bids to satisfy the user requests, the corresponding increase in procurement time is quite small. This can be observed in Fig. 4 and Table 7. The pruning of the bid graph at each instance of the CLOUD-$\mathcal{CABOB}$ algorithm cuts down on the number of bids, which in turn reduces the procurement costs.

### 5.2.4 CATS Distributions

CATS (Combinatorial Auction Test Suite) [34] is a standard combinatorial auction instance generator used mainly to test combinatorial auction algorithms. We tested the algorithm on all of the CATS which include paths, matching, regions, scheduling and arbitrary. For each one of them, we used the default parameters in the CATS instance generators, and varied the vendors offering the service and number of items offered. The time estimates derived are given in Table 8.

The CATS distributions were relatively easier to analyze, since most of them were solved by our LP formulation. As the number of items requested by the user increases, the number of vendors and the items in the bid also increase. Even in such scenarios, CLOUD-$\mathcal{CABOB}$ worked fast, mainly due to pruning of large numbers of bids.

The primary advantage of our approach in cloud computing is that it is highly scalable and very fast in winner determination. The time taken in winner determination is of the order of $\mathcal{O}(|V| + |E|)$ [22] where $V$ and $E$ corresponds to the vertices and edges of the generated bid graph $G$, which in turn corresponds to nothing but the time taken for the search path. This time is considered to be very fast among all winner determination algorithms in combinatorial auction problems. We can show that the time depends only on the search path; since we build the bid graph on an incremental basis, the time to build it is a constant. The step that might take lot of time are calculating the upper thresholds and lower thresholds, which involves LP solving, but present-day *interior point method solvers* [45] are of high speed and precision, and hence the time taken for this step is likely to be negligible in practice. Hence the overall time is restricted to the time in the search path, which makes our approach quite fast.

## 6 Conclusion

Currently, a cloud user pays for cloud resources or services on a pay-as-you-go basis; this type of pricing is called fixed pricing. Fixed pricing is very popular with telecommunication providers, but has no provision for price incentives for the users. Resource procurement is not only an important

TABLE 6
Delay Estimation of CLOUD-$\mathcal{CABOB}$ over Uniform Distribution

| Number of Items/Bid | Execution Time (in ms) |
|---|---|
| 4 | 15 |
| 5 | 14 |
| 6 | 15 |
| 7 | 13 |
| 8 | 13.2 |
| 9 | 13.6 |
| 10 | 13.3 |
| 11 | 18 |

TABLE 7
Time Estimation of CLOUD-$\mathcal{CABOB}$ Using Decay Distribution

| Number of Bids | Execution Time (in ms) |
|---|---|
| 500 | 17 |
| 1,000 | 22 |
| 1,500 | 30 |
| 2,000 | 35 |
| 2,500 | 40 |
| 3,000 | 45 |
| 3,500 | 52 |
| 4,000 | 60 |

TABLE 8
Time Estimation of CLOUD-$\mathcal{CABOB}$ Using CATS Distribution

| Number of Bids | Execution Time (in ms) |
|---|---|
| 4,000 | 10 |
| 5,000 | 16 |
| 7,000 | 20 |
| 10,000 | 24 |
| 12,000 | 28 |
| 15,000 | 33 |
| 17,000 | 38 |
| 20,000 | 40 |
| 22,000 | 97 |

problem in cloud computing, but is also a largely unexplored one. Currently, resource procurement is done manually. In order to automate procurement, an attempt was made by Prasad and Rao [8] to use mechanism design, but in reality, the number of resources requested by the user is more than one, and the user may procure different sets of resources from different cloud vendors.

Hence, we have proposed the CLOUD-$\mathcal{CABOB}$ algorithm, a domain-specific improvement of the CABOB algorithm, to permit fast winner determination in combinatorial auction mechanisms, and found a way to produce optimal resource procurement for the user requesting a set of resources. When tested with an actual sample data set from cloud computing, we found that resource procurement in combinatorial auctions in the proposed manner is far superior to sequential auctions. Also, combinatorial auctions in cloud computing can be scaled to large user requirements. We foresee a scenario where combinatorial auctions using this approach will be extensively used by a very large numbers of cloud users to procure sets of resources economically from the many cloud vendors who offer myriad sets of resources with different specifications that cannot be meaningfully compared and analyzed in any other way.

Our algorithm CLOUD-$\mathcal{CABOB}$ thus has advantages for both the service providers and the cloud users. As the number of resources requested increases, the challenges faced by service providers increase. This creates a need for service providers to come up with better procurement models that ensure quality of service while also improving utilization and profitability. This can be done at scale using our approach.

Cloud users will also benefit, as they will be able to procure their requested combinations of resources at an economical price, compared to procuring the same resources sequentially. Thus, our work has value for both the service provider and the cloud user with a win-win situation for both parties in a system using our approach.

## ACKNOWLEDGMENTS

The authors would like to thank Dr. P. Suresh for his valuable suggestions and comments. A brief extended abstract of this work was presented under the same title at the 12th International Conference on Intelligent Systems Design and Applications (ISDA 2012), Kochi, India, in November 2012.

## REFERENCES

[1] P. Mell and T. Grance. (2011, Sep.). *NIST Definition Cloud Comput*. National Institute of Standards and Technology (NIST), US Dept. of Commerce, NIST Special Publication, pp. 800–145. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] L. Badger, T. Grance, R. Patt-Corner. (2012, May). and J. Voas, *Cloud Comput. Synopsis Recommendations*. National Institute of Standards and Technology (NIST), US Dept. of Commerce, NIST Special Publication, pp. 800–146, http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf

[3] Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, 2010, pp. 91–98.

[4] F. Ridder and A. Bona. (2011, Feb.). four Risky Issues Contracting Cloud Serv.. Gartner Res. Rep. G00210385. [Online]. Available: http://bit.ly/S6L4Zx

[5] R. Weiss and A. Mehrotra, "Online dynamic pricing: Efficiency, equity and the future of e-commerce," *Virginia J. Law Technol.*, vol. 6, no. 2, pp. 1–11, 2001.

[6] G. Zhu, S. Sangwan, and L. Tingjie, "Mechanism design of online multi-attribute reverse auction," in *Proc. 42nd Hawaii Int. Conf. Syst. Sci.*, Jan. 2009, pp. 1–7.

[7] Y. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 91–98.

[8] A. S. Prasad and S. Rao, "A mechanism design approach to resource procurement in cloud computing," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 17–30, Jan. 2014, doi:10.1109/TC.2013.106.

[9] P. Ghosh, N. Roy, S. K. Das, and K. Basu, "A pricing strategy for job allocation in mobile grids using a non-cooperative bargaining theory framework," *J. Parallel Distrib. Comput.*, vol. 65, no. 11, pp. 1366–1383, 2005.

[10] S. Penmatsa, and A. Chronopoulos, "Price-based user-optimal job allocation scheme for grid systems," in *Proc. Int. Symp. Parallel Distrib. Process.*, Apr. 2006, p. 396.

[11] X. Xie, J. Huang, H. Jin, S. Wu, M. Koh, J. Song, and S. See, "Pricing strategies in grid market: Simulation and analysis," in *Proc. Int. Conf. Grid Cooperative Comput.*, Oct. 2008, pp. 532–538.

[12] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing," *Concurrency Comput.: Practice Exp.*, vol. 14, no. 13-15, pp. 1507–1542, 2002.

[13] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservations and co-allocation," in *Proc. Int. Workshop Quality Serv.*, 1999, pp. 27–36.

[14] H. Casanova and J. Dongarra, "Netsolve: A network server for solving computational science problems," *Int. J. Supercomput. Appl. High Perform. Comput.*, vol. 11, pp. 212–223, 1995.

[15] S. J. Chapin, D. Katramatos, J. F. Karpovich, and A. S. Grimshaw, "The legion resource management system," in *Proc. Job Scheduling Strategies Parallel Process.*, 1999, pp. 162–178.

[16] S. Parsons, J. A. Rodriguez-Aguilar, and M. Klein, "Auctions and bidding: A guide for computer scientists," *ACM Comput. Surveys*, vol. 43, no. 2, pp. 10:1–10:59, Jan. 2011, doi:10.1145/1883612.1883617.

[17] B. Rochwerger, J. Tordsson, C. Ragusa, D. Breitgand, S. Clayman, A. Epstein, D. Hadas, E. Levy, I. Loy, A. Maraschini, P. Massonet, H. M. noz, K. Nagin, G. Toffetti, and M. Villari, "RESERVOIR—when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, Mar. 2011.

[18] M. Bichler, J. Kalagnanam, K. Katircioglu, A. J. King, R. D. Lawrence, H. S. Lee, G. Y. Lin, and Y. Lu, "Applications of flexible pricing in business-to-business electronic commerce," *IBM Syst. J.*, vol. 41, no. 2, pp. 287–302, 2002.

[19] Y. Narahari, C. Raju, K. Ravikumar, and S. Shah, "Dynamic pricing models for electronic business," *Sadhana*, vol. 30, pp. 231–256, 2005, doi:10.1007/BF02706246.

[20] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 164–177, Apr.-Jun. 2012, doi:10.1109/TSC.2011.7.

[21] K. Leyton-Brown, M. Pearson, and Y. Shoham, "Towards a universal test suite for combinatorial auction algorithms," in *Proc. ACM Conf. Electron. Commerce*, 2000, pp. 66–76.

[22] T. Sandholm, S. Suri, A. Gilpin, and D. Levine, "CABOB: A fast optimal algorithm for winner determination in combinatorial auctions," *Manag. Sci.*, vol. 51, no. 3, pp. 374–390, 2005.
[23] S. d. Vries and R. Vohra. (2000). Combinatorial auctions: A survey," Northwestern University, Center for Mathematical Studies in Economics and Management Science, Discussion Papers [Online]. Available: http://EconPapers.repec.org/RePEc:nwu:cmsems:1296
[24] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Practice Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.
[25] G. Belalem, S. Bouamama, and L. Sekhri, "An effective economic management of resources in cloud computing," *J. Comput.*, vol. 6, pp. 404–411, 2011.
[26] (2015, Jan.). Cloudorado: Cloud computing price comparison engine [Online]. Available: http://www.cloudorado.com/
[27] X. Vilajosana, R. Krishnaswamy, and J. Marques, "Design of a configurable auction server for resource allocation in grid," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst.*, Mar. 2009, pp. 396–401.
[28] W.-Y. Lin, G.-Y. Lin, and H.-Y. Wei, "Dynamic auction mechanism for cloud resource allocation," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, 2010, pp. 591–592.
[29] S. Zaman and D. Grosu, "Combinatorial auction-based dynamic VM provisioning and allocation in clouds," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2011, pp. 107–114.
[30] S. de Vries and R. V. Vohra, "Combinatorial auctions: A survey," *INFORMS J. Comput.*, vol. 15, no. 3, pp. 284–309, Jul. 2003.
[31] Y. Fujishima, K. Leyton-brown, and Y. Shoham, "Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches," in *Proc. 16th IJCAI '99, Int. Joint Conf. AI.*, 1999, pp. 548–533.
[32] H. H. Hoos and C. Boutilier, "Solving combinatorial auctions using stochastic local search," in *Proc. 17th Nat. Conf. Artif. Intell.*, 2000, pp. 22–29.
[33] N. Nisan, "Bidding and allocation in combinatorial auctions," in *Proc. ACM Conf. Electron. Commerce*, 2000, pp. 1–12.
[34] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artif. Intell.*, vol. 135, no. 12, pp. 1–54, 2002.
[35] F. Chang, J. Ren, and R. Viswanathan, "Optimal resource allocation in clouds," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 418–425.
[36] T. Sandholm and S. Suri, "BOB: Improved winner determination in combinatorial auctions and generalizations," *Artif. Intell.*, vol. 145, pp. 33–58, 2003.
[37] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.
[38] M. Mohabey, Y. Narahari, S. Mallick, P. Suresh, and S. V. Subrahmanya, "A combinatorial procurement auction for QoS-aware web services composition," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, 2007, pp. 716–721.
[39] T. L. Saaty, "Analytic hierarchy process," in *Encyclopedia of Biostatistics*. Hoboken, NJ, USA: Wiley, 2005.
[40] M. H. Rothkopf, A. Pekec, and R. M. Harstad, "Computationally manageable combinational auctions," *Manage. Sci.*, vol. 44, no. 8, pp. 1131–1147, 1998.
[41] J. K. Ho, "Computing true shadow prices in linear programming," *Informatica, Lith. Acad. Sci.*, vol. 11, no. 4, pp. 421–434, 2000.
[42] T. Sandholm and S. Suri, "Improved algorithms for optimal winner determination in combinatorial auctions and generalizations," in *Proc. 17th Nat. Conf. Artif. Intell. 12th Conf. Innovative Appl. Artif. Intell.*, 2000, pp. 90–97.
[43] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[44] J. K. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, M. Rosenblum, S. M. Rumble, E. Stratmann, and R. Stutsman. (2009). The case for ramclouds: Scalable high-performance storage entirely in dram," *SIGOPS Operating Syst. Rev.* [Online]. *43*, pp. 92–105. Available: http://ilpubs.stanford.edu:8090/942/
[45] A. Andersson, M. Tenhunen, and F. Ygge, "Integer programming for combinatorial auction winner determination," in *Proc. 4th Int. Conf. MultiAgent Syst.*, 2000, pp. 39–46.

**G. Vinu Prasad** received the bachelor's of engineering degree from the University Visvesvaraya College of Engineering, Bangalore, where he received five gold medals as a university topper. Later, he received the master's of technology (MTech) from IIIT-Bangalore, a graduate school of information technology in Bangalore. He has previously worked with Texas Instruments, the Indian Institute of Science, and ABB Research Labs, all in Bangalore. He was a key member of the team which developed an evaluation tool for IEC-61211-compliant modules at ABB Research Labs. He presently heads the Development and Delivery vertical of Viralmo Technologies Pvt. Ltd., a company offering comprehensive cross-platform web, mobile, and cloud solutions.

**Abhinandan S. Prasad** received the MTech degree in information technology from IIIT-Bangalore, and after a spell working for Alcatel Lucent and Motorola in Bangalore, he is currently working toward the PhD degree at Georg-August Universität Göttingen, Germany. He is a Marie Curie fellowship holder and a CleanSky ITN Early Stage Researcher under the 7th Framework Programme of the European Union. His research interests include mechanism design and its application in cloud computing, network function virtualization, and computational linguistics applied to Sanskrit. He is a member of the IEEE Computer Society and the IEEE.

**Shrisha Rao** received the MS degree in logic and computation from Carnegie Mellon University and the PhD degree in computer science from the University of Iowa. He is a professor at IIIT-Bangalore. His primary research interests are in applications of distributed computing, specifically algorithms and approaches for resource management in complex systems such as used in cloud computing. He also has interests in energy efficiency, sustainable computing ("Green IT"), renewable energy and microgrids, applied mathematics, and intelligent transportation systems. He is a life member of the American Mathematical Society and the Computer Society of India, an ACM distinguished speaker, and a senior member of the IEEE.