Dynamic Resource Provision for Cloud Broker with Multiple Reserved Instance Terms

Jiangtao Zhang[1,2], Shi Chen[1,3], Hejiao Huang[1,3], Xuan Wang[1,4(✉)], and Dingzhu Du[5]

[1] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, China
wangxuan@insun.hit.edu.cn
[2] Public Service Platform of Mobile Internet Application Security Industry, Shenzhen 518057, China
[3] Shenzhen Key Laboratory of Internet of Information Collaboration, Shenzhen 518055, China
[4] Shenzhen Applied Technology Engineering Laboratory for Internet Multimedia Application, Shenzhen 518055, China
[5] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080, USA

Abstract. Relying on the knowledge of the pricing benefit of long-term reserved resource and multiplexing gains, cloud broker strives to minimize its cost by utilizing infrastructure resources from public cloud service provider. Different reserved instance terms accompanied by different prices are provisioned by the provider. How to choose the appropriate ones from various terms to meet the dynamic user demands at the least cost is a great challenge. This paper addresses the challenge by two algorithms. Extensive real world traces driven evaluations show that the heuristic algorithm runs about twice as fast as the approximation one, while both algorithms can save almost the same resource cost up to 27 %.

Keywords: Cloud broker · Dynamic resource provision · Reserved instance terms · Cost minimization · IaaS

1 Introduction

Many public cloud service providers (CSPs) deliver infrastructure as a service (IaaS) to users. Each provider differentiates itself in terms of service area, virtual machine (VM) instance types (each instance with a fixed CPU, memory and storage etc.), prices of different sources and performance guaranteed. Especially, there are various pricing mechanisms for VM instances. The instance can be provisioned on-demand and billed at a cycle of hour (Amazon EC2 [3]), minute (Microsoft Azure [16]) or 15 min (Elasthosts [6]). Optionally, the instance can be reserved at any time in advance and then launched at any time as required [21]. The reserved instance is charged a total cost whenever it is active. For example, Amazon EC2 provides a choice of two reserved instance terms, 1 year and

G. Wang et al. (Eds.): ICA3PP 2015, Part I, LNCS 9528, pp. 339–356, 2015.
DOI: 10.1007/978-3-319-27119-4_24

3 years [3] and the terms are charged differently. VMware vCloud Air permits a more flexible subscription term from 1 month to 36 months [21]. Normally, the average price for reservation at each billing cycle is far lower than that of the on-demand one. Furthermore, the longer the term, the lower the price. For instance, the saving over on-demand price of Amazon m3.media instance can be up to 29 % and 60 %, for 1-year and 3-year terms, respectively [3].

Users can rent instances from the provider directly. But limited by their knowledge of the cloud IaaS market, it is a great pain for users to make a selection from various instances with various prices to cut their cost. Even sometimes, they have to bargain with more than one provider to meet the different services or service area requirements, thus incurring more overhead. Subjectively, these requirements drive users to seek the help of the third party and lead to the emergence of cloud broker in recent years [11].

A cloud broker can provide on-demand instance to users with lower price. On one hand, the broker can choose infrastructures among providers for their "expertise and experience" offering the service. On the other hand, based on multiplexing of relatively small requirements of individual users, the broker can make a bulk reservation in advance at a much lower reservation price. Then offer users on-demand instances at an intermediate price. Huge profit space exists and entices the broker to render brokerage services objectively.

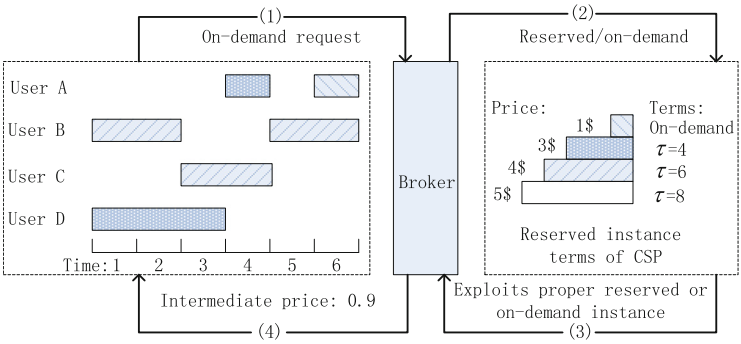


Fig. 1. Broker multiplexes the time-varying dynamic requests of four users. Then delivers on-demand services to users at a lower intermediate price by leveraging proper reserved and on-demand instances. The left user request bar is served by the instance bar with the same setting on the right CSP side.

Figure 1 depicts the service delivery scheme. Different users apply for on-demand instances from time 1 to 6 (step 1). The broker tries to rent instances from CSP to meet the requests at the least cost (step 2). CSP provides on-demand and reserved instances with different prices and terms τ. The price for on-demand instance is charged at each hour (billing cycle). For the reserved scenario, the price is the total cost of τ billing cycles. Based on the number and the lifetime of the aggregated request, the broker can exploit the pricing benefit

of long-term resource reservation and multiplexing gains (step 3). This enables the broker to offer users with a price of only 0.9$ which is lower than the on-demand prices of CSP (step 4). In reality, the broker can earn 1.9$, because he can charge 9.9$ and only need to pay 8$.

The challenge for the broker is how to exploit the price differences of on-demand and reserved resources from a cloud, so as to reduce his cost. Because every user dynamically submits his request, the aggregated request fluctuates with time. Policies for broker to rent instances from providers should be adapted accordingly. It is necessary to make the following decisions about renting instances, such as, on-demand launching or selecting an appropriate instance to reserve? How many instances and at what time to launch?

This paper aims to answer these questions. The main contributions are as follows:

1. We formulate the dynamic resource provision problem where the broker rents resources from a cloud service provider with multiple reserved instance terms.
2. A heuristic and an approximation algorithms are presented for the problem.
3. Extensive real world traces driven simulations demonstrate the effectiveness of both the algorithms. Significant resource cost saving can be achieved by using the algorithms.

The remainder of the paper is organized as follows. In Sect. 2 related work is reviewed. Section 3 formulates the problem. Section 4 and Sect. 5 present a heuristic and an approximation algorithms, respectively. They are evaluated in Sect. 6 and concluded in Sect. 7.

2 Related Work

Cloud Brokering has been proposed as a service and attracted plenty of attention in recent years [1,15]. Cost minimization and profit maximization are two main topics being explored. To minimize cost, the authors of [18] explore the possibility to optimally place VMs across multiple clouds. The capacity of each cloud and the load balance are considered. Eight heuristic algorithms which differ in diverse criteria for assigning priorities to VMs requests are presented in [17]. The authors try to maximize the profit of a broker who sublets on demand resources to customers. It is extended in [10] and a distributed simulated annealing algorithm is recommended. Different from the granularity of VMs, data center based graph clustering algorithm is presented to minimize the cost of the broker, including nodes (data centers), intra-cloud bandwidth and inter-cloud bandwidth [4]. Broker mechanism is also recommended to make cooperation decisions so as to maximize the cooperated CSPs [19].

In addition to cost and profit, quality of service is another main concern. Aiming to meet the requirements of users, a multi-objective decision strategy [2] sorts CSPs by scoring all kinds of constraints, especially on the technology heterogeneity, and then chooses the CSP with the maximum score. Reaction time minimizing and profit maximizing are explored in [13]. Based on Pareto optimum theory, the author formalizes the broker scheduling problem as a multi-objective

programming and solves it by a simplified multi-objective genetic algorithm. The placement of latency sensitive application in multiple CSPs environment is also studied [5]. The problem is formulated as a mixed-integer programming subject to the resource capacity, load balance and latency. Furthermore, two policies are given to address the faulty scenario of CSP. Based on different criteria, such as performance optimization, cost minimization and energy efficiency, the scheduling function of the broker is equipped with 0-1 integer programming based algorithm to select the optimal cloud to deploy a service. Modern portfolio theory is leveraged to choose an efficient broker policy so that the tradeoff between satisfying uncertain demand and risk of not delivering satisfied services is balanced [8]. A framework is proposed to select CSPs so that the quality of service is achieved by combining their trustworthiness and competence [9]. Trustworthiness is estimated by the historical record of quality or reputation. While competence is the claimed service level. However, the aforementioned works do not take into account the price difference between on-demand and reserved resources. The dynamic property of the request is also not captured.

Reserved resources provide a great opportunity for the broker to reduce his cost. A dynamic resource reservation strategy is recommended for broker to minimize cost [22,23]. A dynamic programming is used to characterize the optimal solution. The original combinatorial problem is decomposed into a number of subproblems by using a set of recursive bellman equations and each is solved more efficiently. Two approximation algorithms are proposed for off-line and online scenarios, respectively. Especially, the off-line algorithm is proved as 2-approximate. Broker federation is explored in [14]. But all of them deem that there is only one reserved instance term. It is inconsistent with the practice of the cloud computing industry. This work explores dynamic resource provision with multiple reserved instance terms and just fills the gap.

3 Formulation

3.1 User Demand and CSP Pricing

User demand. Suppose a broker has an estimation of aggregated request up to a rather long period T. For any $t \in [1,T]$, the aggregated request at t is d_t. It is reasonable for each user should have a plan for the future request. The aggregation can be estimated based on the users' plans [14,23], or based on the historical requests. Let $L = max_t(d_t)$ is the peak request. We divide the demand d_t into L levels. Define indicator d_t^l to represent whether there is a demand at level l at billing cycle t, i.e., $d_t^l = 1$ if $d_t \geq l$, and 0 otherwise. For example, a demand curve is depicted in Fig. 2. $d_3^4 = 0$ because there is no demand at billing cycle 3 at level 4. This billing cycle is called as a vacant billing cycle. $d_5^5 = 1$, because there is a demand at billing cycle 5 at level 5. Obviously, $\sum_{l=1}^{d_t} d_t^l = d_t$.

A demand curve is called convex if for any $t_0 \in [1,T]$, there is no other $t_1 \in [1,T], t_2 \in [1,T]$, such that $t_1 < t_0 < t_2$, $\sum_{l=1}^{L} d_{t_1}^l > \sum_{l=1}^{L} d_{t_0}^l$ and $\sum_{l=1}^{L} d_{t_2}^l > \sum_{l=1}^{L} d_{t_0}^l$. For example, in Fig. 2, the curve between time 1, 3 is convex.

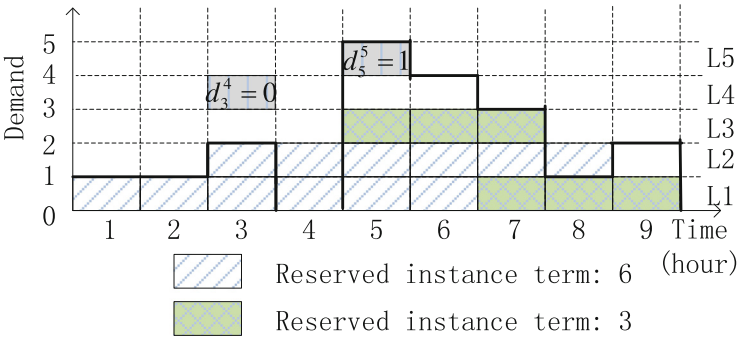


Fig. 2. Demand curve and level by level reservation with three reserved instance terms, where $\{\tau_0, \tau_1, \tau_2\, \tau_3\} = \{1, 3, 6, 10\}$, $\{c_0, c_1, c_2, c_3\} = \{1, 2.5, 3.8, 6\}$. $T = 9$. At level 1, τ_1 and τ_2 will be replaced by τ_3 at last because $c_1 + c_2 > c_3$.

CSP pricing. Suppose a CSP provides J increasing reserved instance terms $\tau_1, \tau_2, ..., \tau_J$ (an instance with term τ_j is also called instance τ_j or term τ_j when there is no ambiguity, $j = 1, ..., J$), and all terms are longer than one billing cycle. The corresponding costs are $c_1, c_2, ..., c_J$, respectively. Specifically, let c_0 denotes the on-demand price, hence τ_0 is 1 which means one billing cycle. Then we have

(1) $c_0 < c_1 < ... < c_J$. Namely, the longer the term, the bigger the cost. Otherwise, the short term with bigger cost can be replaced by the long term with smaller cost. It is unnecessary to set the shorter one.

(2) $c_0/\tau_0 > c_1/\tau_1 > ... > c_J/\tau_J$. Namely, the longer the term, the cheaper the average price at each billing cycle. Otherwise, suppose for $i, j, (i < j)$, $c_i/\tau_i < c_j/\tau_j$. Then users can reserve τ_i by $\lceil \tau_j/\tau_i \rceil$ times to achieve the same resource with less cost and more capital flexibility.

3.2 Problem Formulation

Let y_{ljt} is a boolean variable which indicates whether to allocate an instance τ_j at time t to serve the demand at this time at level l. It equals 1 if this demand is served by an instance τ_j rather than τ_0, and 0 otherwise. For example, at level 2 in Fig. 2, $y_{223} = 1$ because τ_2 is reserved at time 3. All other $y_{2jt} = 0$. This problem can be formulated from the point view of levels.

$$\min \sum_{l=1}^{L} \sum_{t=1}^{T} \sum_{j=0}^{J} y_{ljt} c_j \tag{0-1ILP}$$

$$\text{s.t.} \sum_{l=1}^{d_t} \Big(\sum_{j=1}^{J} \sum_{i=t-\tau_j+1}^{t} y_{lji} + y_{l0t} \Big) \geq d_t \qquad t = 1, 2, \cdots, T \tag{1}$$

$$y_{ljt} \in \{0, 1\}\, l = 0, 1, \cdots, L, j = 0, 1, \cdots, J, t = 1, 2, \cdots, T. \tag{2}$$

The objective is the total cost because whenever τ_j is reserved at any level, it should be charged c_j. In constraint (1), $\sum_{j=1}^{J}\sum_{i=t-\tau_j+1}^{t} y_{lji}$ is the number of instances which remain effective until time t at level l. y_{l0t} is the number of on-demand instances allocated at time t at level l. After calculating the sum of all levels, the left part of the constraint represents the number of all instances which can be used at time t. It should not be smaller than the demand d_t.

Programming 0-1ILP is a 0-1 integer programming and it belongs to one of the Karp's 21 NP-complete problems [12]. Hence it is also NP-hard.

Now, for each level l ($l = 1, 2, ..., L$), considering the following single level programming:

$$\min \sum_{t=1}^{T}\sum_{j=0}^{J} y_{ljt}c_j \qquad \text{(0-1ILPForLevel)}$$

$$\text{s.t.} \sum_{j=1}^{J}\sum_{i=t-\tau_j+1}^{t} y_{lji} + y_{l0t} \geq d_t^l \quad t = 1, 2, \cdots, T \qquad (3)$$

$$y_{ljt} \in \{0, 1\} \quad j = 0, 1, \cdots, J, t = 1, 2, \cdots, T. \qquad (4)$$

It is easy to check the feasible set of programming 0-1ILP and the intersection of the feasible set of all programming 0-1ILPForLevel are just the same. Let $f(y)$ denotes $\sum_{t=1}^{T}\sum_{j=0}^{J} y_{ljt}c_j$. Suppose y^* is the optimal solution for all programming 0-1ILPForLevel. That means, for any feasible solution of all 0-1ILPForLevel y, which is also feasible for 0-1ILP, we have $f(y) \geq f(y^*)$. So $\sum_{l=1}^{L} f(y) \geq \sum_{l=1}^{L} f(y^*)$. Thus

$$min \sum_{l=1}^{L} f(y) \geq min \sum_{l=1}^{L} f(y^*) = \sum_{l=1}^{L} f(y^*). \qquad (5)$$

The left part is just the optimal value of programming 0-1ILP. This motivates us to find the optimal solution of programming 0-1ILP level by level.

4 Heuristic Algorithm

4.1 Reservation Heuristic

With One Reserved Instance Term. We define $u^l = \sum_{t=1}^{T} d_t^l$, is the number of non-vacant billing cycles during period T at level l. In Fig. 2 $u^1 = 8$ because there is no demand at time 4. u^5 =1.

When there is only one reserved instance term τ_1 with cost c_1, the periodic reservation mechanism is recommended [23]. Considering the simplest scenario where $T \leq \tau_1$, because any reservation remains effective in T, the problem becomes trying to reserve instances as many as possible at time 1. At level 1, if $u^1 c_0 \geq c_1$, that means compared to reserved instance with lower cost c_1, more expense is necessary if to launch u^1 on-demand instances. So we should reserve an instance τ_1. Suppose $l-1$ instances have been reserved at bottom $l-1$ levels,

then the l-th instance should be reserved only if $u^l c_0 \geq c_1$, i.e., $u^l \geq c_1/c_0$. Note that u^l is non-increasing in l, we obtain a useful heuristic: reserve l instance at time 1 only if $u^l \geq c_l/c_0 > u^{l+1}$, if insufficient, launch other instances on demand. Because this heuristic finds the maximum l, for any level which is bigger than l, it is not economical any longer if adopting reservation policy.

In Fig. 3, considering the first scenario, where $T = 3 < \tau$, $d_3^3 = 0$ because there is no demand at billing cycle 3 at level 3, so even there is a reservation, it will not be used. $d_2^5 = 1$, because there is a demand at billing cycle 2 at level 5. $u^2 = 3$, $u^3 = 2$, $c_1/c_0 = 2.5$, $u^2 > c_1/c_0 > u^3$, so we reserve 2 instances at time 1.

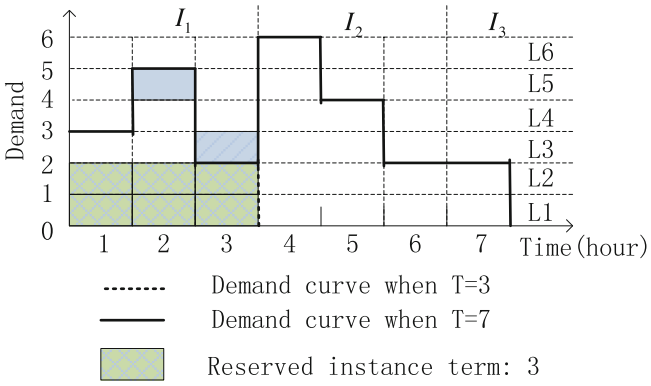


Fig. 3. Periodic reservation mechanism.

If $T > \tau_1$, T is divided into intervals and each with a length of τ_1. The upper heuristic is used in each intervals. In the upper figure, the demand curve where $T = 7$ is divided into 3 intervals.

With Multiple Reserved Instance Terms. When there are multiple reserved instance terms, it is impossible to find an appropriate term in advance to facilitate the periodic reservation. A new mechanism is necessary to address the problem. First, we give some notions to facilitate the mechanism presentation.

Length of each level. Note that the length of each level may be different. For level l, the length of this level is the number of billing cycles between the first demand time and the last demand time at level l, i.e.,

$$T^l = b_l - e_l + 1, \tag{6}$$

where $b_l = min\{t|d_t^l = 1\}, e_l = max\{t|d_t^l = 1\}$ are the first demand time and the last demand time, respectively. In Fig. 2, $[b_1, e_1] = [1, 9], T^1 = 9$, $[b_2, e_2] = [3, 9], T^2 = 7$.

Residue length of each level. For level l, let b' denotes the current non-vacant beginning time, then the residue length of this level is

$$T^{Rl} = e_l - b' + 1. \tag{7}$$

If beginning from b^l, $T^{Rl} = T^l$.

Note that given a level l, it is still economical to reserve a τ_j instance if

$$u_I^l c_0 \geq c_j, \tag{8}$$

where $u_I^l = \sum_{t \in I} d_t^l$, $I = [b, b + \tau_j]$. b is any given beginning time in the residue interval at this level and its initial value equals 1.

Feasible term. Given a level l with residue length T^{Rl}, a term τ_j $(j \geq 1)$ is called feasible if $\tau_j \leq T^{Rl}$ and inequation (8) holds.

Beginning time update. At each level, we try to find the longest feasible term from the beginning of the level, i.e., b_l. Let b' denotes the current beginning. Suppose the longest feasible term is τ_h $(h \geq 1)$ if it is found, or even τ_1 is still not feasible, then the beginning time is moved backward to find the next longest feasible term. The next beginning time can be determined as follows:

$$b = min\{t | d_t^l = 1, t \geq b' + \tau_h\}, \tag{9}$$

where $h = 1$ if there is no feasible term or τ_1 is the longest feasible term. Namely, the first demand time after the current interval. For example, in Fig. 2, at level 1, the first beginning time is 1 and the second beginning time is 7.

4.2 Level by Level Reservation Algorithm

We adopt a level by level reservation mechanism. Of course, every time the term with the lowest average price (i.e., the longest term) is preferred to others which also meet (8). If this longest term is shorter than the length of this level T^l, then the upper mechanism is repeated in the residue interval. It is detailed in Algorithm 1.

The first For loop (Lines 1–31) checks the demand level by level. At each level, If the residue length of the level is bigger than τ_1, the While loop (Lines 3–15) tries to reserve the longest term. The second For loop checks all the terms which are shorter than the residue length in a decreasing order (Lines 5–14). If a longer term is feasible, then we reserve such an instance and the related parameters are updated. Otherwise, we try the shorter instance, until we find one (Lines 7–10) or all the reservations are not economical (Lines 11–13). The beginning time is moved backward to begin a new interval try. For example, in Fig. 2, a τ_2 and then a τ_1 instance are reserved at level 1. The process is repeated until T^{Rl} is shorter than the shortest term τ_1. Then we decide whether to reserve τ_1 or serve the residue demands by on-demand instances (Lines 16–23). In Fig. 2, the demand at time 9 can only be served by an on-demand instance since there is no feasible term.

Algorithm 1. Level by level longest term preference decision (3LTPD)

Input: d_t: number of required instances at time t, $t = 1, 2, .., T$. $\tau_1, \tau_2, ..., \tau_J$: increasing reserved instance terms with increasing cost $c_1, c_2, ..., c_J$. c_0: on-demand price. L: number of levels

Output: A $T * J$ matrix $A = (A_{t,j})$: $A_{t,j}$ is the number of reserved τ_j instances at time t

1: **for** $l = 1, \cdots, L$ **do**
2: $A \leftarrow 0$, calculate length of this level T^l (6), $T^{Rl} \leftarrow T^l$, interval beginning time $b \leftarrow min\{t : d_t^l \neq 0\}$, $b' \leftarrow b$, this level cost: $lc \leftarrow 0$
3: **while** $T^{Rl} \geq \tau_1$ **do**
4: Find maximum $j \leftarrow \{j : \tau_j \leq T^{Rl}\}$
5: **for** $h = j, \cdots, 1$ **do**
6: $I \leftarrow [b, b + \tau_h]$, reservation indicator: $resvInd \leftarrow 0$, level reservation indicator: $ind \leftarrow 0$
7: **if** $u_I^l c_0 \geq c_h$ **then**
8: Reserve a τ_h instance at time b, $A_{b,h} \leftarrow A_{b,h} + 1$, update b (9) and T^{Rl} (7), $resvInd \leftarrow 1$, $ind \leftarrow ind + 1$, $lc \leftarrow lc + c_h$
9: break
10: **end if**
11: **if** $h = 1$ and $resvInd = 0$ **then**
12: Update b (9) and T^{Rl} (7), $lc \leftarrow lc + c_0$
13: **end if**
14: **end for**
15: **end while**
16: **if** $T^{Rl} > 0$ **then**
17: $I \leftarrow [b, e^l]$
18: **if** $u_I^l c_0 \geq c_1$ **then**
19: Reserve a τ_1 instance at time b, $A_{b,1} \leftarrow A_{b,1} + 1$, $lc \leftarrow lc + c_1$
20: **else**
21: $lc \leftarrow lc + u_{T^{Rl}}^l c_0$
22: **end if**
23: **end if**
24: **if** $lc > c_{j+1}$ **then**
25: Cancel all reservations at this level and $A \leftarrow 0$
26: Reserve a τ_{j+1} instance at time b', $A_{b',j+1} \leftarrow 1$
27: **end if**
28: **if** $ind = 0$ **then**
29: **return**
30: **end if**
31: **end for**

Suppose at level l, τ_j is the shortest term which is bigger than T^l. Then it is possible that the cost of τ_j is smaller than the current total cost at this level. Since it can meet the demand at this level, we will check it so as find a lower cost (Lines 24–27). As in Fig. 2, τ_1 and τ_2 at level 1 are replaced by τ_3 finally.

Because u_I^l is non-increasing of l, if no term is feasible in interval I at level l, then there is no feasible term in interval I at upper levels. It is necessary to

check so that the program terminates in time (Lines 28–30), or after all levels are traversed. Any demand which cannot be served by reserved instance should be served by the on-demand instance.

It is easy to show the time complexity of 3LTPD is $O(LTJ)$. If bisearch is used to seek the longest feasible term, then $O(LTlogJ)$ time is required.

The next theorem demonstrates that for the convex demand, this algorithm can find a 2-approximation solution.

Lemma 1. *For each level, algorithm 3LTPD finds a 2-approximation solution when the demand is convex.*

Proof. Suppose there are J reserved instances with increasing terms $\tau_1, \tau_2, ..., \tau_J$ and increasing cost $c_1, c_2, ..., c_J$, where $c_1/\tau_1 > c_2/\tau_2 > ... > c_J/\tau_J$. c_0 is the on-demand price. T^l is the length of this level. Let j is the biggest one for which $\tau_j \leq T^l$. C^{Al} is the cost of level l incurred by algorithm 3LTPD and opt^l is the optimal cost of level l. When the demand is convex, then the algorithm tries to fill the level by τ_j because it is the cheapest. Suppose o^l is the number of on-demand instances to be launched. We have $C^{Al} = \lfloor T^l/\tau_j \rfloor c_j + min\{o^l c_0, c_j\}$, $opt^l \geq \lfloor T^l/\tau_j \rfloor c_j$. So $\frac{C^{Al}}{opt^l} \leq \frac{min\{\lfloor T^l/\tau_j \rfloor c_j + min\{o^l c_0, c_j\}, c_{j+1}\}}{\lfloor T^l/\tau_j \rfloor c_j} \leq \frac{min\{(\lfloor T^l/\tau_j \rfloor + 1) * c_j, c_{j+1}\}}{\lfloor T^l/\tau_j \rfloor * c_j} \leq \frac{(\lfloor T^l/\tau_j \rfloor + 1) * c_j}{\lfloor T^l/\tau_j \rfloor * c_j} \leq \frac{2\lfloor T^l/\tau_j \rfloor * c_j}{\lfloor T^l/\tau_j \rfloor * c_j} = 2$. The fourth inequality is due to $T^l/\tau_j \geq 1$.

Theorem 1. *For the brokering problem, algorithm 3LTPD finds a 2-approximation solution when the demand is convex.*

Proof. Let C^A is the cost of 0-1ILP incurred by 3LTPD, opt is the optimal cost of 0-1ILP and opt^l is the optimal cost of level l. Based on inequality (5), for any feasible solution y for 0-1ILP, $\sum_{l=1}^{L} f(y) \geq \sum_{l=1}^{L} f(y^*)$. So $opt \geq \sum_{l=1}^{L} opt^l$. We have $\frac{C^A}{opt} = \frac{\sum_{l=1}^{L} C^{Al}}{opt} \leq \frac{2\sum_{l=1}^{L} opt^l}{\sum_{l=1}^{L} opt^l} = 2$where the inequality is due to Lemma 1.

5 Set Cover Based Approximation Algorithm

Given a level l, denote all non-vacant billing cycles at this level as set S^l whose element is the single non-vacant billing cycle. Taking all reserved and on-demand instances as a subset family $\tau = \{\tau_0, ..., \tau_J\}$, where each subset τ_j is attached with a cost c_j, then the cost minimization resource provision can be viewed as a cost minimization set cover problem. In this set cover problem, subset from τ can be repeatedly selected to cover S^l. Define the cost effectiveness for each τ_j at interval I as $e_j = c_j/u_I^l$ (I and u_I^l are as in inequation (8)). Then a simple algorithm based on set cover greedy algorithm (Chap. 1 of [20]) is adapted as follows. Each time it selects the feasible term (Sect. 4.1) with the lowest e_j from τ until S^l is covered. We call such term as the cheapest feasible term.

Suppose that the optimal cost of level l is opt^l. Obviously, if the optimal term is longer than T^l then cost found by Algorithm 2 is the same as opt^l. Thus Lemma 3 is established. So we only consider the case where the optimal terms are all smaller than T^l. Similar to the corresponding proof in Chap. 1 of [20], it

Algorithm 2. Set cover based algorithm (SCBA)

Input: A demand curve with L levels
Output: A resource provision solution
1: **for** $l = 1, \cdots, L$ **do**
2: Select the cheapest feasible term from $b = b_l$. Update b (9). Update residue length T^{Rl} (7). Repeated this process until S^l is covered. Calculate the current overall reservation cost C.
3: **if** Exists $\tau_j \geq T^l$ and c_j is smaller than C **then**
4: Cancel all reservations at this level and reserve a τ_j instance at time b^l
5: **end if**
6: **end for**

is easy to prove the following theorems. Number the billing cycles at level l in the sequence of covering as $1^l, ..., u^l$, we have

Lemma 2. *Given level l, for any $k \in 1^l, ..., u^l$, $e_k \leq opt^l/(u^l - k + 1)$ if the solution is not a term which is longer than T^l.*

Proof. Note that the optimal term must be feasible term, otherwise it can be replaced by any feasible ones and thus lead to a smaller cost.

Suppose opt^l is the optimal cost of level l. Because S^l can be covered by the optimal terms, during any iteration of Algorithm 2, the residue subset must can be covered by some terms with cost at most opt^l. Suppose the number of residue non-vacant billing cycles is r, then the average cost effective of the optimal terms is opt^l/r. So there must exists terms with cost effectiveness at most opt^l/r, during the iteration when k is covered, and the number of residue non-vacant billing cycles must be at most $u^l - k + 1$ elements, i.e. $r \geq u^l - k + 1$. Because in this iteration, the smallest cost effectiveness is selected, we have $e_k \leq opt^l/r \leq opt^l/(u^l - k + 1)$.

Lemma 3. *Algorithm SCBA gets a $H(u^l)-$ approximation for each level, where $H(u^l) = 1 + 1/2 + ... + 1/u^l$.*

Proof. If the solutions are some feasible terms, the cost incurred by Algorithm 2 is $\sum_{k=1}^{u^l} e_k$. Based on Lemma 2, we know the cost is at most $(1 + 1/2 + ... + 1/u^l)opt^l$. If the solution is a term which is longer than T^l, the cost is smaller than that of feasible terms found by the algorithm, and we get it.

Based on Lemma 3, similar to the proof of Theorem 1, it is easy to get

Theorem 2. *Algorithm SCBA finds a $H(m)-$ approximation for problem 0-1ILP, where $m = max_l H(u^l)$.*

Note that the only difference between 3LTPD and SCBA is the criterion to select the feasible term every time. The former selects the longest feasible term and the latter need to calculate the cost effectiveness for all feasible terms and then choose the smallest one. So SCBA will not run faster than 3LTPD as shown in experiments. For a general cost minimization set cover problem, it

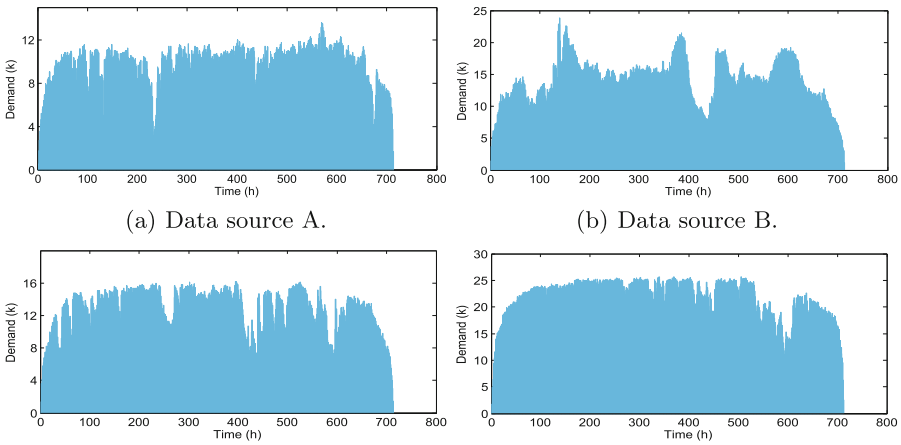
is proved that the greedy algorithm is tight. But maybe it is not true for the brokering problem where the set has a special structure. The next experiment demonstrates that 3LTPD can perform as well as SCBA.

6 Experimental Evaluation

6.1 Experiments Setup

Since public cloud workloads are often confidential, no real IaaS data trace is released so far. So we use Parallel Workloads Archive [7], a repository of job-level usage data from large scale parallel supercomputers, clusters and grids, to evaluate the performance of our algorithms. Four traces logs corresponding to four Intel Netbatch grid clusters, three on the west coast in the US and one in Israel, are used. Each log file contains one month's (i.e., November 2012) accounting records. The original logs are available as Intel-NetbatchX-2012-0 (where "X" is A, B, C, or D for the four different clusters). In experiments, we use A, B, C and D to represent the four data sources, respectively.

User demand and data preprocessing. Since the billing cycle is an hour, the demand at each billing cycle is rounded up, i.e., only when the lifetime of the request falling within the billing cycle is longer than 30 min, the request is counted as one demand.



(a) Data source A. (b) Data source B.

(c) Data source C. (d) Data source D.

Fig. 4. Demand curves of four data sources.

The four data sources have different demand volumes and fluctuations. The corresponding demand curves are depicted in Fig. 4. The peak hourly demands are more than 13k, 23k, 16k and 25k for the four data, respectively. Figure 5 demonstrates the mean values and standard deviations. Generally speaking, data

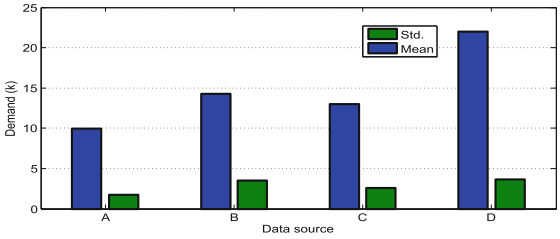


Fig. 5. Demand volume and fluctuation.

A has the smallest demand and D has the biggest demand. B and C have modest demand and B requests more than C. The fluctuation has almost the same tendency.

Parameter setting. The duration of the data trace is a month. Because there are no data at the beginning and end time of the month, we set T as the duration in which data are available, i.e., $T = 712\,\text{h}$.

Table 1. Two groups of resources

CSP 1	Term	Cost ($)	Discount	CSP 2	Term	Cost ($)	Discount
τ_0	1 h	0.059	N/A	τ_0	1 h	0.059	N/A
τ_1	1 day	1.340	5 %	τ_1	5 days	6.018	15 %
τ_2	2 days	2.549	10 %	τ_2	10 days	10.62	25 %
τ_3	1 week	7.930	20 %	τ_3	20 days	16.992	40 %
τ_4	2 weeks	13.877	30 %				

Suppose there are two groups of resources represented by CSP 1 and CSP 2, each with the following terms and prices as demonstrated in Table 1. The on-demand price is borrowed from that of Amazon E2C m3.media with Windows operation system. Limited by the duration of data source, the reserved instance term is shortened accordingly. The cost discount is within the range of business CSP (about 60 % discount for 3-year term for Amazon). For convenience of comparison of the saving cost, the same on-demand price is adopted for the two groups. Note that the terms of CSP 2 are relatively longer than that of CSP 1. This setting can reveal the factors affecting the resource cost[1].

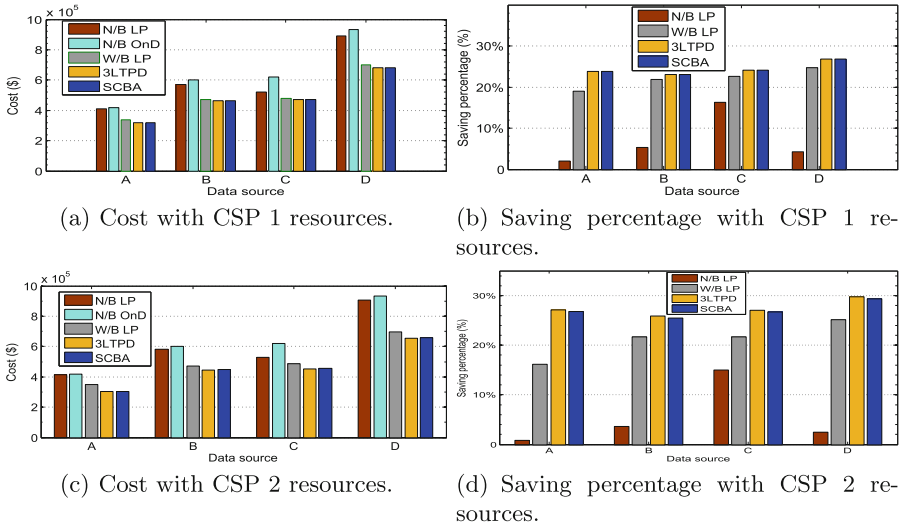
The simulation environment is set up in a Java platform. The platform is running on a PC (Lenovo Think Centre M4350t-N020Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz8G RAM).

[1] The result under different setting demonstrates a tendency similar to the following results under this setting and hence omitted.

6.2 Experimental Results

To evaluate the cost saving of the broker and the proposed algorithms, five different schemes, including the proposed two algorithms are compared. For the scenario without broker, (1) longest preference for each user's request (abbreviated as N/B LP): select the longest reserved instance term for a continuous demand every time and traversing vacant demand is prohibited. The feasible longest term is used because obviously that the shorter term will incur more cost; (2) all demand is served by on-demand instances (abbreviated as N/B OnD). In the scenario with a broker, (1) similar as N/B LP except that it operates on the aggregated demand curve (abbreviated as W/B LP); (2) 3LTPD; (3) SCBA.

Performance Evaluation. We exploit resources from CSP 1 or CSP 2 to serve the requests from data source A, B, C and D. The resource cost is depicted in Fig. 6. In Fig. 6(b) and (d), the cost of on-demand is used as the baseline to calculate the saving percentage.



(a) Cost with CSP 1 resources.

(b) Saving percentage with CSP 1 resources.

(c) Cost with CSP 2 resources.

(d) Saving percentage with CSP 2 resources.

Fig. 6. Cost efficiency when exploiting resources from different CSP.

Since the demand volumes of four data differ, the total costs are also different. From Fig. 6(a) and (c) we can find whichever group of resources is exploited, the total costs demonstrate the similar tendency as that of the demand volume in Fig. 5. Overall, more demand incurs more cost for all five schemes.

Even when there is no broker and the demands are not multiplexed, it is still efficient to use the reserved instances as more as possible. As illustrated in Fig. 6(b) and (d), the scheme that tries the longest term (N/B LP) saves more

cost. For CSP 1, it leads to an average saving up to 3.81 %. While 2.30 % for CSP 2. The reason lies in that the terms of CSP 1 are relatively shorter than that of CSP 2. When the demands are not multiplexed, the lifetime of single user's request is shorter, so the relatively longer terms of CSP 2 cannot be fully used. It reveals the effect of terms on resource cost.

However, after multiplexing by a broker, it demonstrates a contrary situation when the proposed 3LTPD is leveraged. On average, 27.23 % and 24.26 % cost are saved for CSP 2 and CSP 1, respectively. This is because that the aggregated request becomes longer and 3LTPD prefers the feasible longest term every time. Thus the relatively longer terms of CSP 2 can do better. Although W/B LP scheme also prefers the longest term, it does not use the heuristic. So, though it saves more than N/B LP, it is still not more efficient than 3LTPD. This also shows that 3LTPD can enable economical utilization of longer terms across vacant billing cycles.

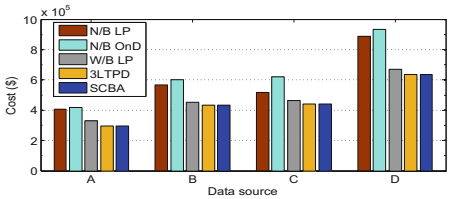
The fluctuation of demand has an effect on the cost as well. Because the more volatile the demand is, the more demand valleys exist, fluctuation hinders the utilization of longer terms. It is detailed in Table 2. Note that the cost saving has the same tendency as that of the standard deviation curve in Fig. 5 when the W/B LP scheme is used: the more fluctuant, the more cost is saved. But 3LTPD can mitigate the negative effect of fluctuation. In total, the gap between cost saving for different data is reduced, though most cost is saved for data D.

Table 2. Saving percentage with broker relevant to fluctuation

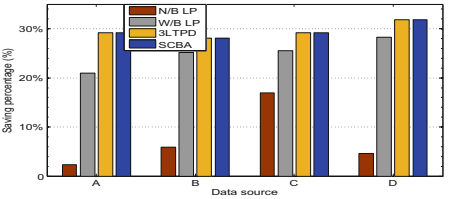
Data		W/B LP (%)	3LTPD (%)		W/B LP (%)	3LTPD (%)
A	CSP 1	19.02	23.84	CSP 2	16.20	27.21
B		21.97	23.07		21.87	25.63
C		20.94	23.84		19.45	26.50
D		22.81	26.29		24.73	29.59

Figure 7 further justifies the benefit of exploiting multiple terms for 3LTPD. Herein all the terms of CSP 1 and CSP 2 are viewed as available for the algorithm and hence more candidate terms can be chosen. Figure 7(a) and (b) plot the cost and saving percentage when all terms are used. Comparing with Fig. 6, it is shown that more terms lead to more cost saving. Especially, we compare the resource cost efficiency for 3LTPD in Fig. 7(c) and (s). When all terms of both CSPs are used, up to 46 thousand dollars (5 %) are saved compared with that when only the terms of CSP 1 are used, and 19 thousand dollars (2 %) are saved compared with CSP 2. This is due to that more terms lead to higher applicability to dynamic demands.

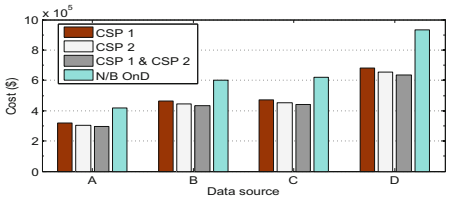
It is noteworthy that for all data sources, although we can only prove 3LDPP is 2-approximation for the convex demand curve, 3LDPP performs almost exactly the same as SCBA for all data sources.



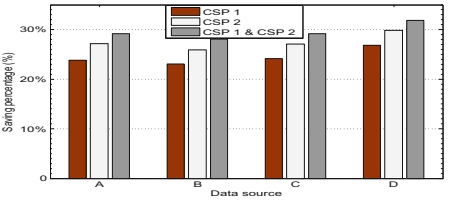
(a) Cost with more reserved instance terms.



(b) Saving percentage with more reserved instance terms.

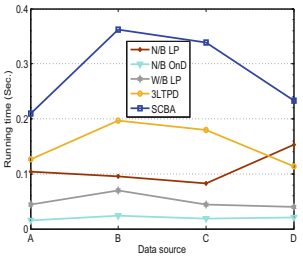


(c) Cost comparison for 3LTPD.

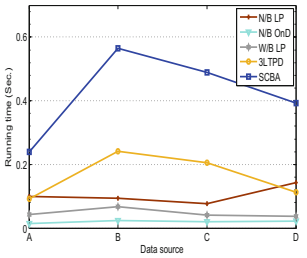


(d) Saving percentage comparison for 3LTPD.

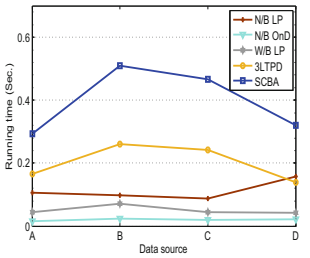
Fig. 7. Cost efficiency when more reserved instance terms are exploited.



(a) From CSP 1.



(b) From CSP 2.



(c) From terms of both CSPs.

Fig. 8. Time efficiency when exploiting resources from different CSP.

Running Time Efficiency Evaluation. Running time of the algorithms is compared in Fig. 8. The running time of algorithms which exploit terms from CSP 1, CSP 2 and both of them are depicted in Fig. 8(a), (b) and (c), respectively. All the three figures show a common pattern. The on-demand algorithm takes the least time. N/B LP takes more time because it seeks the longest term each time for each user. But after multiplexing the demand of each user by the broker, the longest preference scheme (W/B LP) runs faster than N/B LP. This lies in that the broker aggregates users' request and thus reduces the number of times to find the longest term. SCBA and 3LTPD run slower than the former three schemes. Since SCBA computes the cost effectiveness for each feasible term and then selects the smallest one, 3LTPD only selects the longest feasible term, 3LTPD runs almost twice as fast as SCBA. Recall that 3LTPD and SCBA

exhibit almost the same performance (Figs. 6 and 7), the superiority of 3LTPD is demonstrated.

7 Conclusion

Considering the multiple reserved instance terms, two algorithms are presented to facilitate broker to utilize infrastructure resources from public CSP at the least cost. One is heuristic and another is an approximation algorithm. Extensive traces driven evaluation demonstrates the effectiveness of both the algorithms. Our future work aims to exploit resources from multiple CSPs. In this scenario, though there are more candidate terms, there is also interoperability which impedes the multiplexing effect. How to balance the contradiction to achieve an efficient scheme is a great challenge.

Acknowledgments. This work was financially supported by National High-tech R&D Program of China (863 Program) with Grants No. 2015AA016008, National Natural Science Foundation of China with Grants No. 11371004, Shenzhen Strategic Emerging Industries Program with Grants No. JC201104210032A, No. ZDSY20120613125016389, No. JCYJ 20120613151201451 and No. JCYJ201303291532 15152 as well as Shenzhen Development and Reform Commission with Grants No. 2012720 and No. 2012900.

References

1. Amato, A., Di Martino, B., Venticinque, S.: Cloud brokering as a service. In: 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 9–16. IEEE (2013)
2. Amato, A., Venticinque, S.: Multi-objective decision support for brokering of cloud sla. In: 2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 1241–1246. IEEE (2013)
3. Amazon: Amazonvmpricing. http://aws.amazon.com/ec2/pricing/
4. Choi, T., Kim, Y., Yang, S.: Graph clustering based provisioning algorithm for optimal inter-cloud service brokering. In: 2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–6. IEEE (2013)
5. Diaz-Sanchez, F., Al Zahr, S., Gagnaire, M.: An exact placement approach for optimizing cost and recovery time under faulty multi-cloud environments. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), vol. 2, pp. 138–143. IEEE (2013)
6. Elastic: Elastichostsvmpricing. http://www.elastichosts.com/pricing-information/
7. Feitelson, D.G., Tsafrir, D., Krakov, D.: Experience with using the parallel workloads archive. J. Parallel Distrib. Comput. **74**(10), 2967–2982 (2014)
8. Gaivoronski, A.A., Strasunskas, D., Nesse, P.J., Svaet, S., Su, X.: Modeling and economic analysis of the cloud brokering platform under uncertainty: choosing a risk/profit trade-off. Serv. Sci. **5**(2), 137–162 (2013)
9. Ghosh, N., Ghosh, S.K., Das, S.K.: Selcsp: a framework to facilitate selection of cloud service providers. IEEE Trans. Cloud Comput. **3**(1), 66–79 (2014)

10. Iturriaga, S., Nesmachnow, S., Dorronsoro, B., Talbi, E.G., Bouvry, P.: A parallel hybrid evolutionary algorithm for the optimization of broker virtual machines subletting in cloud systems. In: 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 594–599. IEEE (2013)
11. Jamcracker: Csb solutions overview. http://www.jamcracker.com/solutions
12. Karp, R.M.: Reducibility Among Combinatorial Problems. Springer, New York (1972)
13. Kessaci, Y., Melab, N., Talbi, E.G.: A pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2496–2503. IEEE (2013)
14. Liu, K., Peng, J., Liu, W., Yao, P., Huang, Z.: Dynamic resource reservation via broker federation in cloud service: a fine-grained heuristic-based approach. In: 2014 IEEE Global Communications Conference (GLOBECOM), pp. 2338–2343, December 2014
15. Mechtri, M., Zeghlache, D., Zekri, E., Marshall, I.J.: Inter and intra cloud networking gateway as a service. In: 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet), pp. 156–163. IEEE (2013)
16. Microsoft: Microsoftvmpricing. http://azure.microsoft.com/zh-cn/pricing/details/virtual-machines/#Linux
17. Nesmachnow, S., Iturriaga, S., Dorronsoro, B., Talbi, E.G., Bouvry, P.: List scheduling heuristics for virtual machine mapping in cloud systems. In: VI High Performance Computing Latin America Symposium (2013)
18. Tordsson, J., Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M.: Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Gener. Comput. Syst. **28**(2), 358–367 (2012)
19. Truong-Huu, T., Tham, C.K.: A novel model for competition and cooperation among cloud providers. IEEE Trans. Cloud Comput. **2**(3), 251–265 (2014)
20. Vazirani, V.V.: Approximation Algorithms. Springer Science & Business Media, New York (2001)
21. VMVare: Vmvarepricing. http://vcloud.vmware.com/service-offering/pricing-calculator/subscription
22. Wang, W., Niu, D., Liang, B., Li, B.: Dynamic cloud resource reservation via iaas cloud brokerage. IEEE Trans. Parallel Distrib. Syst. **PP**(99), 1 (2014)
23. Wang, W., Niu, D., Li, B., Liang, B.: Dynamic cloud resource reservation via cloud brokerage. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS), pp. 400–409. IEEE (2013)