Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 760458, 10 pages
http://dx.doi.org/10.1155/2014/760458

Research Article

A Time-Efficient Solution to the General Resource Placement Problem in Cloud

Wei Wei, Yuhong Zhang, and Yang Liu

College of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, China

Correspondence should be addressed to Wei Wei; nsyncw@126.com

Received 13 April 2014; Revised 7 August 2014; Accepted 21 August 2014; Published 29 September 2014

Academic Editor: Chengjin Zhang

Copyright © 2014 Wei Wei et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cloud based large-scale online services are faced with regionally distributed stochastic demands for various resources. With multiple regional cloud data centers, a crucial problem that needs to be settled is how to properly place resources to satisfy massive stochastic demands from many different regions. For the general stochastic demands oriented cross region resource placement problem, the time complexity of existing optimal algorithm is linear to total amount of resources and thus may be inefficient when dealing with a large number of resources. To end this, we propose an efficient algorithm, named discrete function based unbound resource placement (D-URP). Experiments show that in scenarios with general settings, D-URP can averagely achieve at least 97% revenue of optimal solution, with reducing time by three orders of magnitude. Moreover, due to the generality of problem setting, it can be extended to get efficient solution for a broad range of similar problems under various scenarios with different constraints. Therefore, D-URP can be used as an effective supplement to existing algorithm under time-tense scheduling scenarios with large number of resources.

1. Introduction

Large-scale online services (such as YouTube [1] and Facebook [2]) often face highly dynamic user demands from distributed physical locations. A proper online service can leverage cloud computing to lower cost and serve more users. Based on typical cloud platform like [3, 4], the target of resource scheduling in large-scale online services can be viewed as placing given resources at geographically dispersed data centers and serving as many demands as possible. For a specific request, one prefers to be served in local data center rather than in remote one. This is most obviously because the locality of data centers needs to be considered to guarantee QoS (quality of service). Moreover, under scenarios like online scheduling, a relative high response time is often a must, and it is essential to recompute placement quickly and efficiently to handle high dynamic demands with time. Therefore, how to minimize operation cost and to find optimal placement efficiently in such scheduling scenarios is an impending need to meet in cloud platform.

For instance, the cloud based content distribution network can be viewed as an example that falls under above paradigm, where a number of geographically dispersed cloud data centers are selected to be close to end users. Under this scenario, user's requests are preferred to be served by a local data center, and service provider would like to predistribute content replicas at each data center so as to meet as much content demand as possible in the most efficient way. Another example is cloud based media streaming, in which service provider faces the problem of where to place virtual machine instances, so as to satisfy time-varying bandwidth demands from users spreading across various geographical regions.

In above paradigm, online services are faced with stochastic demands which are represented by demand distribution for each resource type in each region. Since we do not suppose stochastic demands to follow a specific distribution, a variety of scenarios can be covered: (1) demands with various patterns and distributions, for example, realistic demand patterns, extracted from a large number of traces in production servers [5], which contains several different kinds of variability, autocorrelation, and periodicity. (2) Demands that vary across data centers, as in [6], and demands in different data centers may be correlated with each other. (3) Demands vary across resources; for example, the demands

change significantly across channels in media streaming services like YouTube [7].

Summarized from above scenarios, the general resource placement problem can be formulated as follows: given a global resource budget, how to satisfy as many requests as possible by placing resources optimally over regions. Since each request can be satisfied either by the data center in same region with high QoS (i.e., high revenue) or by the data center from a different region with low QoS (i.e., low revenue), algorithm's aim is to maximize expected revenue of resources.

By considering stochasticity of demands, algorithm can be more efficient than prior mean demand based ones. Consider the scenario of placing k resources in a system with two regions and one resource type. Demands from the first region have a deterministic demand of k requests, and the demands from the second region have bursty demands of nk $(n > 1)$ requests with probability $1/n$ and 0 request with probability $1 - 1/n$. The aim of algorithm is to maximize the expected number of satisfied requests (i.e., the revenue). Assuming that one resource can satisfy one request, then the algorithm using stochastic demand will place all k resources in the first region with a revenue of k. In comparison, the algorithm using demands mean will place $k/2$ resources in each region, yielding a revenue of $(n + 1)k/2n$, which approximates $k/2$ if $n \gg 1$; that is, considering stochasticity in demands may significantly improve utilization of resources.

However, stochastic demands will significantly increase computation complexity. It is challenging to efficiently solve a resource allocation problem that combines combinatorial aspects and arbitrary stochastic demand distribution in time-tense scheduling scenarios. To the best of our knowledge, only one SSP (successive shortest path) based optimal algorithm was proposed for a similar stochastic demand based problem [8], whose time complexity, however, is linear to total amount of resources and then may be inefficient when dealing with a large number of resources.

To reduce the computational complexity, one can efficiently optimize servicing revenue by utilizing the inner structure of the problem. We develop an efficient placement algorithm accordingly, suitable for time-tense scheduling in large-scale systems. Our paper mainly makes three contributions. (1) We prove that the problem can be transformed to two subproblems, and the solutions to these subproblems can be represented as functions of demands and resources. (2) We derive mathematical description of all necessary functions, to facilitate computation of optimal solution. (3) We develop an efficient discrete function based resource placement algorithm (called D-URP), which gets solution quickly by leveraging discrete functions to approximate continuous ones. Under the scenarios with general settings, we compare the effect and efficiency of D-URP with commonly used mean demand based algorithm and the optimal SSP based algorithm. Our analysis and preliminary experiments indicate that D-URP outperforms mean demand based algorithm under all scenarios and can averagely achieve at least 97% revenue of optimal SSP based algorithm, with time complexity reduced to nearly one thousandth of it. Therefore, it can be used as an effective supplement to existing algorithm in time-tense scheduling scenarios. And due to the generality of problem settings, it can be extended to get efficient solution for a broad range of similar problems under various scenarios with different constraints.

2. Related Work

One part of related works was cloud based resource scheduling for multiple services [9–11]. Reference [9] handled the problem of scheduling multiple resources under budget constraint in one data center. The viewing behaviors of users in a multichannel VoD (video on demand) application were analyzed to get a queueing network and then a dynamic resource provisioning algorithm is provided to reduce cloud utilization cost. Reference [10] investigated the response time and resource cost minimization problem under three typical scenarios: single-service, multiservice, and priority-service scenarios. Queueing model based algorithms were developed to optimally allocate cloud resources for each service in all scenarios. Reference [11] proposed QoS-aware resource elasticity (QRE) framework. Based on the assessment of the application behavior, QRE framework can help dynamic adjust cloud resources.

There was also a substantial research effort being made about similar problems of resource consolidation [12, 13]. Reference [12] investigated resource provisioning problem in shared Internet hosting platforms, which was similar to the scenarios in distributed cloud centers. The proposed algorithm overbooked cluster resources in a controlled fashion; thus, it could provide application isolation and performance guarantees at run time. Reference [13] treated virtual machine as "moldable" during consolidation and developed a genetic algorithm to minimize the overhead of transition between system states.

Other similar works were cloud based content delivery [14–18]. MetaCDN in [14] intelligently matches and places users' content onto many storage clouds and exploits the difference in storage cloud to maximize utility functions like quality of service. Reference [15] studied the cost optimization problem in content multihoming scenario, where a content publisher uses multiple CDN to aggregate the diversity of individual CDN providers on features, performance, and commitment. The proposed algorithm can reduce publishing cost by 40%. Reference [16] proposed a hierarchical framework to solve the deployment problem in cloud-oriented content delivery network. In the framework, inter- and intracloud communication resources were simultaneously considered and the problem was transformed to replica placement and graph partitioning problem. Reference [17] handled the problem of efficiently moving the data from different geographical locations into a cloud for further processing. The two proposed online algorithms can efficiently find the optimal routing paths and the right data centers for processing. Reference [18] studied the cost minimization problem of content migration in cloud of clouds. An online heuristic algorithm is proposed to optimize the cost of migration and adapt to the changes of user access pattern.

Some related works in P2P (peer-to-peer) systems also investigated similar problems [19, 20]. Reference [19] addressed the problem of content placement in "distributed

server networks" scenario of P2P systems, where the content placement strategy can help resolve similar problems in cloud related scenarios. Reference [20] investigated the relationship between the number of videos and peers, the storage capacity of each peer, and the resultant offloading of video server bandwidth. The conclusion can also help design cloud based VoD systems.

Most of existing works were carried out only based on mean demands of resource placement. However, a more realistic scenario is that resource demands from massive users are dynamic and stochastic. The main challenge posed by this paradigm is how to deal with an arbitrary multidimensional (high-dimensionality) stochastic demand. A method of resource placement for similar problem is proposed in [8]. The main differences between their problem setting and ours are that their problem setting is for data centers with dedicated servers (with resource constraints in each region), while our setting is for cloud data center (with only a global resource budget). The typical scenario of their problem is placing resources over traditional data centers with limited resources. To solve the problem, they transformed the resource placement problem into min-cost flow one, and the optimal solution can be obtained by a customized version of SSP (successive shortest path) algorithm. But under scenario with large number of resources, the SSP based algorithm is not efficient enough since its time complexity is linear to total amount of resources. Different from the SSP based idea, we propose an efficient algorithm by leveraging the special structure of our problem. The algorithm can achieve at least 97% revenue of optimal solution, and its time complexity is independent of the total amount of resources.

3. Problem Formulation

We consider a system consisting of J regions indexed by j $(1 \le j \le J)$. In each region, one can place multiple resources of different types indexed by i $(1 \le i \le I)$. We denote by L_i^j the amount of type-i resources placed in region j, the set of resources placed in these regions is called a *placement* and denoted by $P = \{L_i^j\}$, with $L_i = \sum_{j=1}^{J} L_i^j$ as the total amount of type-i resources. There is no limit on the amount of resources in each region, which is consistent with the model of cloud computing. For a type-i request from region j, if the request is assigned to a type-i resource, it is labelled as "*satisfied*." If a type-i request from region j is satisfied, it is assigned to either (1) type-i resources located in region j, that is, the request is satisfied locally, or (2) A type-i resource located in a different region, that is, the request is satisfied remotely.

Let S^{loc} and S^{rem} denote the number of requests satisfied locally and remotely; then the revenue under placement P is

$$R^P = w_{\text{loc}} S^{\text{loc}} + w_{\text{rem}} S^{\text{rem}}. \tag{1}$$

Denote S^{sat} as the total number of satisfied requests; there is $S^{\text{sat}} = S^{\text{loc}} + S^{\text{rem}}$, and then revenue can be rewritten as

$$R^P = w_{\text{loc}} S^{\text{loc}} + w_{\text{sat}} S^{\text{sat}}, \tag{2}$$

where w_{loc}, w_{sat}, and w_{rem} are revenue weights. For stochastic demand, let D_i^j be a random variable denoting the demand for type-i resource in region j and let D_i be a random variable denoting the demand for type-i resource in all regions. Their CDFs (cumulative distribution functions) are denoted by $\text{cdf}_{i,j}$ and cdf_i, respectively. Given demand realization d_i^j and d_i, the revenue can be represented as

$$R^P = w_{\text{sat}} \sum_{i=1}^{I} \min\left(L_i, d_i\right) + w_{\text{loc}} \sum_{i=1}^{I} \sum_{j=1}^{J} \min\left(L_i^j, d_i^j\right). \tag{3}$$

For any arbitrary placement P and stochastic demand D_i^j, the expected revenue is

$$\begin{aligned} E\left(R^P\right) = {} & w_{\text{sat}} \sum_{i=1}^{I} E\left(\min\left(L_i, D_i\right)\right) \\ & + w_{\text{loc}} \sum_{i=1}^{I} \sum_{j=1}^{J} E\left(\min\left(L_i^j, D_i^j\right)\right). \end{aligned} \tag{4}$$

Then given a global resource constraint U, the aim of algorithm is to find the optimal placement P that maximizes the expected revenue. Consider

$$\begin{aligned} \max \quad & \left(E\left(R^P\right)\right) \\ \text{s.t.} \quad & \sum_{i=1}^{I} \sum_{j=1}^{J} L_i^j = U. \end{aligned} \tag{5}$$

4. Algorithm

Generally, the problem formulated by (5) can be solved by traversing all possible placements $P \in \{P\}^U$ (Algorithm 1), where $O = \{L_i\}$ is *type-placement*, which only presents the total amount of resources of each type. And a type-placement O can be concretized to a set of placements $\{P\}^O$. We denote by $\{P\}^U$ and $\{O\}^U$ the set of placements and type-placements under global constraint U, respectively.

However, the traversing algorithm is of high complexity; for example, in a typical cloud based scenario, there is a very large data space to search. And in each iteration, the computation of $(E(R^P))$ is time-expensive, and,for instance, needs to use the $I \times J$ distributions with each one represented by N data units (in some specific applications, it is possible that $I \times J > 10^4$). To find efficient algorithm, we need to decompose the problem firstly, which is as illustrated below.

Firstly, the problem to be solved by lines (3)–(5) in Algorithm 1 can be described as Subproblem 1.

Subproblem 1. Given a type-placement $O' = \{L_{i'}\}$, find the max expected revenue of all possible placements $P \in \{P\}^{O'}$, which can be formulated as follows:

$$\begin{aligned} \max \quad & \left(E\left(R^P\right)\right) \\ \text{s.t.} \quad & \sum_{j=1}^{J} L_{i'}^j = L_{i'} \quad \forall i' \in [1, I]. \end{aligned} \tag{6}$$

input: total amount of resources U, the demand distributions.
output: the optimal placement $\widehat{P}$
(1) $\widehat{R} = -\infty$; $\widehat{P} = null$;
(2) **for** $O \in \{O\}^U$ **do**
(3) **for** $P \in \{P\}^O$ **do**
(4) **if** $(E(R^P)) > \widehat{R}$ **then** $\widehat{R} = (E(R^P))$, $\widehat{P} = P$;
(5) **end**
(6) **end**

Algorithm 1: The traversing search algorithm.

And the problem to be solved by lines (2)–(6) in Algorithm 1 can be described as Subproblem 2.

Subproblem 2. Given a global resource constraint U, find the type-placement O' whose solution to Subproblem 1 is the maximal one among all $O \in \{O\}^U$.

It can be easily observed that the solution of Subproblem 1 can contribute to solution of Subproblem 2, and Algorithm 1 is just a solution to Subproblem 2 using traversing search algorithm. To find efficient solution, we need to investigate the properties of these subproblems.

4.1. Efficient Solution to Subproblem 1. To investigate the properties of solution to Subproblem 1, we introduce some notations. Denote by *type-i revenue* of placement P (R_i^P) the revenue of type-i resource in the placement; then the *expected type-i revenue* for P (i.e., $E(R_i^P)$) is

$$E\left(R_i^P\right) = w_{\text{sat}} E\left(\min\left(L_i, D_i\right)\right) + w_{\text{loc}} \sum_{j=1}^{J} E\left(\min\left(L_i^j, D_i^j\right)\right). \tag{7}$$

For placement P, its expected revenue $E(R^P)$ can be rewritten as the summation of all its expected type-i revenues. Consider

$$E\left(R^P\right) = \sum_{i=1}^{I} E\left(R_i^P\right). \tag{8}$$

Given type-placement O', L_i is constant for each i. $E(R_i^P)$ is only functions of L_i^j with $\sum_{J=1} L_i^j = L_i$. Then for all $i \in [1, I]$, the maximal value of $E(R_i^P)$ can be achieved independently of each other, which is as follows:

$$\max\left(E\left(R^P\right)\right) = \sum_{i=1}^{I} \max\left(E\left(R_i^P\right)\right) \quad \forall P \in \{P\}^{O'}. \tag{9}$$

Therefore, we can get Subproblem 1.2.

Subproblem 1.2. Given a type-placement $O' = \{L_{i'}\}$ and i', find the max expected type-i' revenue of all possible placements $P \in \{P\}^{O'}$, which can be as follows:

$$\begin{aligned} &\max \quad \left(E\left(R_i^P\right)\right) \\ &\text{s.t.} \quad \sum_{j=1}^{J} L_i^j = L_i, \end{aligned} \tag{10}$$

where L_i is the amount of type-i resources in type-placement O'. It is obvious that Subproblem 1 can be decomposed into I Subproblem 1.2, which can also be solved by traversing search algorithm with high computation complexity. Fortunately, we are able to find properties of solutions after investigating the problem in continuous domain. These properties will be later utilized to devise efficient algorithms for the problem in discrete domain. To get the closed form of (10) in continuous domain, we first introduce Lemma 1.

Lemma 1. *For any continuous nonnegative random variable X with CDF $cdf_X(\cdot)$ and constant C, it satisfies $E(\min(C, X)) = \int_0^C (1 - cdf_X(v))dv$.*

The proof of lemmas, corollaries, and theorems in the paper can be found in Appendix. Then the closed form of $E(R_i^P)$ in continuous domain is

$$E\left(R_i^P\right) = w_{\text{sat}} \int_0^{L_i} \left(1 - cdf_i(n)\right) dn + w_{\text{loc}} \sum_{j=1}^{J} \int_0^{L_i^j} \left(1 - cdf_{i,j}(n)\right) dn. \tag{11}$$

To investigate the properties of solutions to Subproblem 1.2, the following theorem is introduced first.

Theorem 2. *For N functions $f_1(x_1), f_2(x_2), \ldots, f_N(x_N)$, with $x_n \in [0, \infty)$ and $f_n(x_n) \geq 0$ $(1 \leq n \leq N)$, the derivative functions $f_1'(x_1), f_2'(x_2), \ldots, f_N'(x_N)$ are continuous and monotonic decreasing functions, with $0 \leq f_n'(x_n) \leq Y (Y > 0)$. Then for constraint $\sum_{i=1}^{N} x_n = U$, the maximal value of $\sum_{i=1}^{N} f_i(x_i)$ is achieved when $f_1'(x_1) = f_2'(x_2) = \cdots = f_N'(x_N)$.*

Note that in Theorem 2 the optimal X may not be unique, since $f_n'(x_n)$ need not be strictly monotonically decreasing and there may be $f_n'(x_n') = f_n'(x_n'')$ with $x_n' \neq x_n''$; thus, we only

need to find one of the optimal solutions to achieve the maximal value of $f(X)$. Since L_i $(1 \le i \le I)$ in (5) *is constant for a given O, then according to Theorem 2, the solution to Subproblem 1.2 can be calculated as follows.*

Corollary 3 (the efficient solution to Subproblem 1.2). *Given type-placement O and given i* $(1 \le i \le I)$*, if a placement* $P' \in \{P\}^O$ *satisfies* $L_i^j = cdf_{i,j}^{-1}(1 - G_i^{-1}(L_i))\ (\forall j \in [1, J])$*, then* $\max(E(R_i^{P'}))$ *is achieved. Here* $G_i(v) = \sum_{j=1}^{J} cdf_{i,j}^{-1}(1 - v)$ $(0 \le v \le 1)$.

According to Corollary 3, by leveraging some functions, we can quickly calculate the solution to Subproblem 1.2; as a result, we can get the solution to Subproblem 1.

Additionally, for a given i, according to Corollary 3 and the definition of $E(R_i^P)$ *as in* (11), $\max(E(R_i^P))$ *is a function of* L_i *under given demand distribution, which is denoted by* $h_i(L_i)$*. Then the maximal expected revenue of solution to Subproblem 1 can be quickly calculated as follows:*

$$\max\left(E\left(R^P\right)\right) = \sum_{i=1}^{I} h_i\left(L_i\right) \quad \forall P \in \{P\}^O. \tag{12}$$

4.2. Efficient Solution to Subproblem 2. For a given global resource constraint U, Subproblem 2 can be rewritten as follows:

$$\begin{aligned} &\max \quad \left(\sum_{i=1}^{I} h_i\left(L_i\right)\right) \\ &\text{s.t.} \quad \sum_{i=1}^{I} L_i = U. \end{aligned} \tag{13}$$

According to Theorem 2, the solution to Subproblem 1 can be as characterized below.

Corollary 4. *Given* $U > 0$*, for a type-placement* $O' \in \{O\}^U$*, if it satisfies* $L_i = h_i'^{-1}(H^{-1}(U))$ *for any i, then* $\max(\sum_{i=1}^{I} h_i(L_i))$ *is achieved. Here* $H(s) = \sum_{i=1}^{I} h_i'^{-1}(s)$ *and* $h_i'(L_i) = w_{\text{sat}}(1 - cdf_i(L_i)) + w_{\text{loc}} G_i^{-1}(L_i)$.

In Corollary 4, $G_i^{-1}(\cdot)$*,* $H^{-1}(\cdot)$*, and* $h_i'^{-1}(\cdot)$ *are inverse function of* $G_i(\cdot)$*,* $H(\cdot)$*, and* $h_i'(\cdot)$*, respectively. By solving Subproblem 2, we can get the maximal expected revenue in original problem of* (5) *and the type-placement* O'*; then the corresponding optimal placement P can be located by solving Subproblem 1 for type-placement* O'.

The above continuous functions can be approximated by discrete ones and the optimal placement for global resource constraint U can be obtained as in Algorithm 2. The discrete functions are firstly calculated iteratively (lines (1)–(14)) and then the optimal solution is produced by calling their inverse functions (lines (15)–(18)). We denote the algorithm as discrete function based resource placement algorithm (D-URP).

In our implementation of D-URP, each of these discrete functions is recorded in hash map data structure, and its time complexity of querying is $O(1)$*. And for discrete function* f*, an absent* $f(x)$ *is approximated proportionally by*

$$f(x) = f\left(x_1\right) + \left(x - x_1\right) \frac{f\left(x_2\right) - f\left(x_1\right)}{x_2 - x_1}, \tag{14}$$

where x_1 *and* x_2 *are the nearest present values to x with* $x_1 < x < x_2$*. By binary search algorithm, the time complexity of finding* x_1 *and* x_2 *within K keys is* $O(\log(K))$*. As in [8], the demand distributions from input can be precalculated in* $O(1)$ *and are available in an external data base. As a result, the worse time complexity of codes in Algorithm 2 is* $O(I * J)$ *(lines (1)–(3)),* $O(IJK + IK \log K)$ *(lines (4)–(14)), and* $O(IJ \log K)$ *(lines (15)–(18)); one can easily calculate that the time complexity of D-URP is no more than* $O(IJK + IK \log K)$ *and less than* $O(IJK)$ *when* $\log K \le J$*, where K is the cycle count in Algorithm 2. Additionally, iterations in the algorithm can be easily parallelized to get further acceleration. If some iterations are executed in parallel (lines (1)–(3), (6)–(10), and (15)–(18), resp.), the time complexity is less than* $O(JK + IK \log K)$.

5. Experiments

In this section, we evaluate D-URP algorithm to study the following subjects: (1) comparing the effectiveness of D-URP with SSP based algorithm in [8], which is denoted by S-URP (SSP based unbounded resource placement) and (2) checking the relationship of time complexity and effectiveness of D-URP.

Similar to the experiment settings of prior works [8], we consider demands that follow Zipf distribution; that is, the probability of a single request to demand a type-i resource is

$$p_i = \frac{i^{-e}}{\sum_{n=1}^{I} n^{-e}}, \tag{15}$$

where $e > 0$ is a real number. Without loss of generality, we choose D_i^j to be Poisson distribution with a rate of $p_i \lambda / J$, where λ is the expected number of total requests and the expectation of demands from each region is equal.

First we compare the effect of proportional mean method, S-URP, and D-URP under different scenarios, where in method of proportional mean, the number of type-i resources in region j is proportional to the mean demand:

$$L_{i'}^{j'} = \frac{E\left(D_{i'}^{j'}\right)}{\sum_{i=1}^{I} \sum_{j=1}^{J} E\left(D_i^j\right)} U. \tag{16}$$

From Figures 1, 2, and 3, the expected revenues of three methods are compared under surplus scenario ($U = 1500 > \lambda$), balanced scenario ($U = 1000 = \lambda$), and deficit scenario ($U = 500 < \lambda$), respectively. The expected revenue is drawn against Zipf parameters. The number of regions $J = 4$ and the revenue parameters $w_{\text{loc}} = w_{\text{sat}} = 1$. The parameter settings are $I = 500$ and $\lambda = 1000$ and the cycle count of D-URP is $K = 100$. Similar to previous works [8], the Zipf parameter is set to be wide enough and ranges from 0.7 to 1.4.

It is observed from Figures 1, 2, and 3 that D-URP can outperform proportional mean based methods with different

input: total amount of resources U;
demand distribution $\mathrm{cdf}_i(\cdot)$, $\mathrm{cdf}_{i,j}(\cdot)$, $\mathrm{cdf}_i^{-1}(\cdot)$ and $\mathrm{cdf}_{i,j}^{-1}(\cdot)$; $X_{i,j} = \min(\mathrm{cdf}_{i,j}(x) = 1 | \forall x)$.
output: amount of resources of each type placed in each region L_i^j.

(1) **for** $i = 1$ **to** I **do**
(2) $G_i^{-1}(x) = h'_i(x) = 0$, for $x \geq \sum_{j=1}^{J} X_{i,j}$.
(3) **end**
(4) **for** $k = 0$ **to** K **do**
(5) $v = w_{\mathrm{loc}} * k/K$.
(6) **for** $i = 1$ **to** I **do**
(7) $L'_i = \sum_{j=1}^{J} \mathrm{cdf}_{i,j}^{-1}(1 - v)$.
(8) $G_i(v) = L'_i$.
(9) $h'_i(L'_i) = w_{\mathrm{sat}}(1 - \mathrm{cdf}_i(L'_i)) + w_{\mathrm{loc}} v$.
(10) **end**
(11) $s = (w_{\mathrm{sat}} + w_{\mathrm{loc}}) * k/K$.
(12) $H(s) = \sum_{i=1}^{I} h_i'^{-1}(s)$.
(13) **if** $H(s) \geq U$ **then** break;
(14) **end**
(15) **for** $i = 1$ **to** I **do**
(16) $L_i = h_i'^{-1}(H^{-1}(U))$.
(17) **for** $j = 1$ **to** J **do** $L_i^j = \mathrm{cdf}_{i,j}^{-1}(1 - G_i^{-1}(L_i))$;
(18) **end**

Algorithm 2: The steps of D-URP.

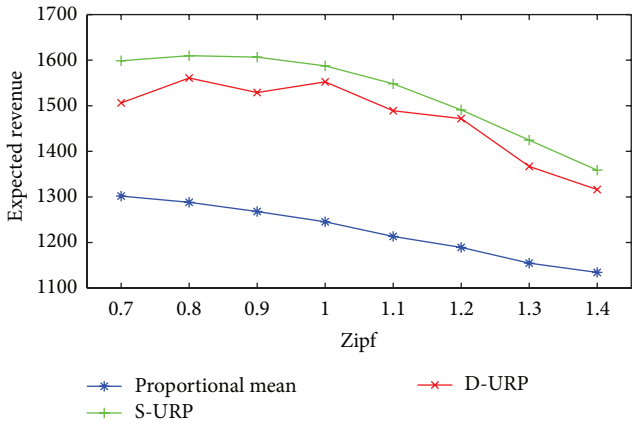


Figure 1: Expected revenue of mean, S-URP, and D-URP with different Zipf values under deficit scenarios ($U = 500$).

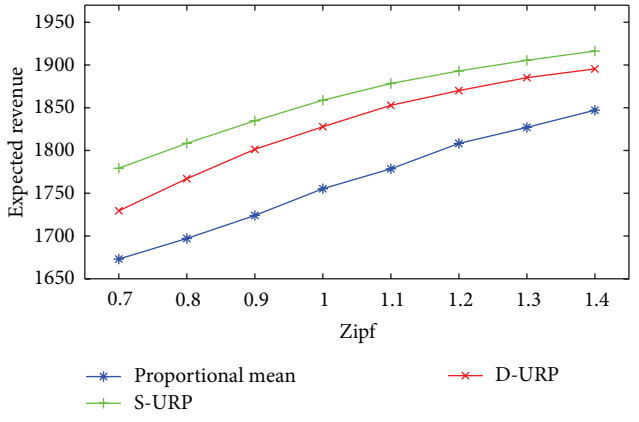


Figure 2: Expected revenue of mean, S-URP, and D-URP with different Zipf values under balanced scenarios ($U = 1000$).

Zipf values; for example, the expected revenue of D-URP (line marked with "×") is always larger than that of proportional mean based method (line marked with "∗"). And the effect of D-URP is close to that of S-URP; for example, the difference in expected revenue between D-URP and S-URP (distance between line marked with "×" and line marked with "+") is always within 100 in all three figures.

As shown in Figures 4, 5, and 6, the expected revenues of these three methods are compared under surplus ($U = 1500 > \lambda$), balanced ($U = 1000 = \lambda$), and deficit scenario ($U = 500 < \lambda$), respectively. The expected revenue is drawn against number of resource types (I). The number of regions $J = 4$ and the revenue parameters $w_{\mathrm{loc}} = w_{\mathrm{sat}} = 1$. The parameter settings are Zipf = 1.0 and $\lambda = 1000$ and the cycle count of D-URP is $K = 100$. The number of type I resources ranges from 100 to 1000 to cover the result in different orders of magnitude of I.

It is shown from Figures 4, 5, and 6 that D-URP always outperforms proportional mean based methods with different number of resource types, since the expected revenue of D-URP (marked with "×") is always larger than that of proportional mean based method (marked with "∗"). Because the difference in expected revenue between D-URP and S-URP (between line marked with "×" and "+") changes

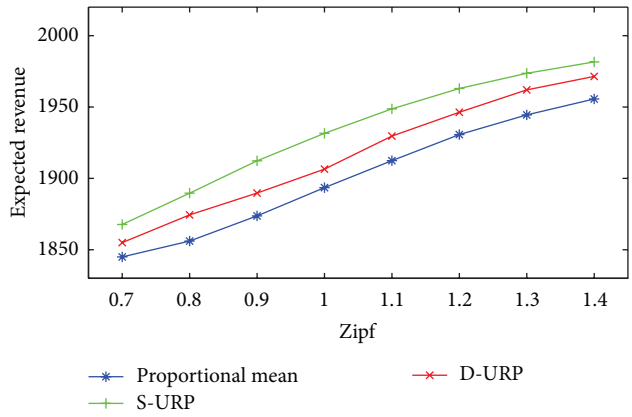


Figure 3: Expected revenue of mean, S-URP, and D-URP with different Zipf values under surplus scenarios ($U = 1500$).

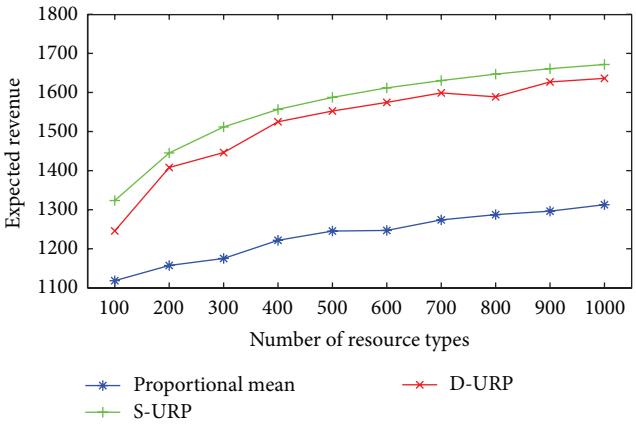


Figure 4: Expected revenue of mean, S-URP, and D-URP with different number of resource types under deficit scenarios ($U = 500$).

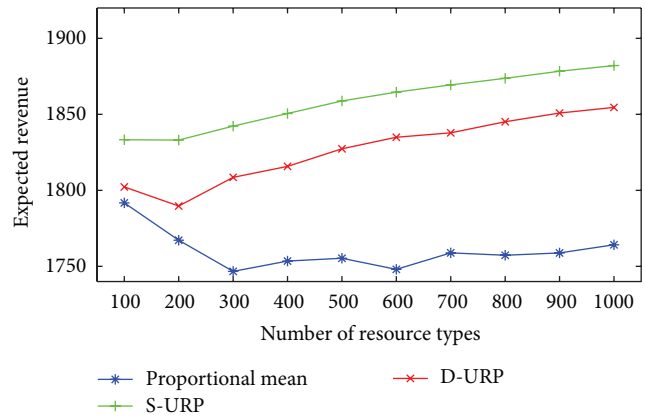


Figure 5: Expected revenue of mean, S-URP, and D-URP with different number of resource types under balanced scenarios ($U = 1000$).

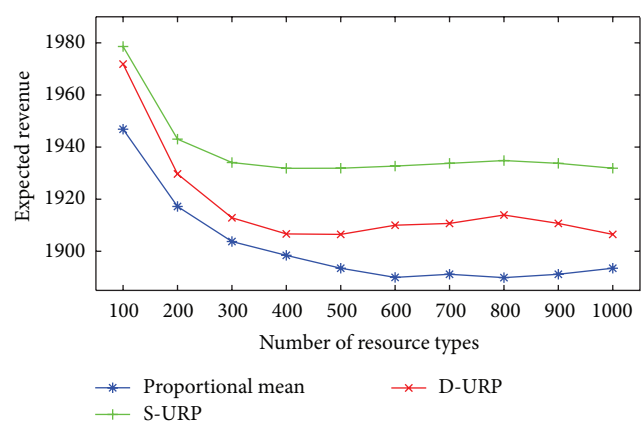


Figure 6: Expected revenue of mean, S-URP, and D-URP with different number of resource types under surplus scenarios ($U = 1500$).

little (always within 100) in all three figures, the effect of D-URP is close to that of S-URP with increasing number of resource types.

In brief, from Figures 1, 2, 3, 4, 5, and 6, we can see that D-URP can outperform proportional mean based methods, and the effect of D-URP is close to S-URP in all scenarios. However, it is observed that, along with the increasing of available resources (U increases from 500 to 1500), the effect of mean based method (line marked with "$*$") approaches that of optimal solution S-URP (line marked with "+"). The underlying reason is that the number of unsatisfied demands decreases with more resources, which leads to the decreasing of room for optimization. Since resources cannot be surplus enough in most of real scenarios, D-URP and S-URP can still be used to gain more revenue in common scenarios.

Since the result of S-URP is optimal, we then investigated the effect of D-URP using the revenue ratio of D-URP to S-URP. In Figures 7 and 8, the revenue ratio is drawn against Zipf parameters with $\lambda = 1000$. The number of regions $J = 4$ and the revenue parameters $w_{\text{loc}} = w_{\text{sat}} = 1$. In Figure 7, Zipf ranges from 0.7 to 1.4. We compared D-URP and S-URP using the revenue ratio of D-URP to S-URP, with $K = 100$ and $I = 100$, under surplus scenario ($U = 1500 > \lambda$), balanced scenario ($U = 1000 = \lambda$), and deficit scenario ($U = 500 < \lambda$). In Figure 8, we compare D-URP and S-URP with $K = 100$ and I ranging from 100 to 1000, under surplus, balanced, and deficit scenarios. We can see that the effect of D-URP does not change significantly along with the increasing of Zipf and number of resource types (I).

In Figure 9, we investigate the time complexity of D-URP using different expected numbers of total requests ($\lambda = 1000, 5000, 10000$), Zipf values (0.7 to 1.4), I (100 to 1000), and available resources U (0.5λ to 1.5λ with interval of 0.1λ). For each pair of λ and U, S-URP and D-URP are applied to get revenue ratio, with D-URP's cycle count K ranging from 10 to 200. The average revenue ratio (of all U) for each K under different λ is plotted in Figure 9. We can see that averagely D-URP can achieve more than 97% revenue of optimal solution and can achieve more than 99% when $\lambda = 10000$.

In brief, the following is shown in above experiments.

(1) D-URP is lightly affected by the popularity distribution of resources (Zipf) and number of resource types (I).

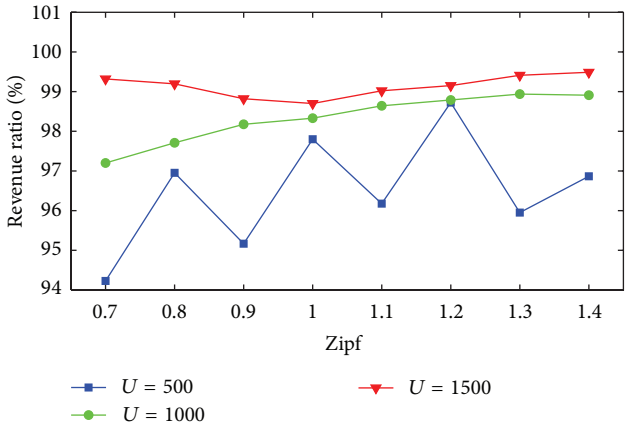


Figure 7: Revenue ratio of D-URP to S-URP with different Zipf values under deficit, balanced, and surplus scenarios.

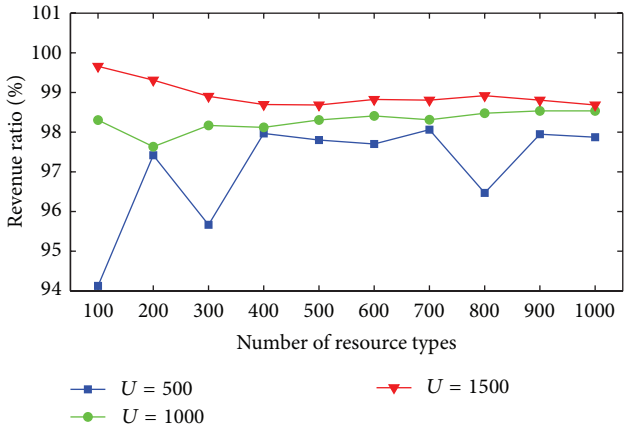


Figure 8: Revenue ratio of D-URP to S-URP with different number of resource types under deficit, balanced, and surplus scenarios.

(2) D-URP is affected by the scarcity of resources. It is observed that it is better in surplus scenario or with more total available resources. By inspecting the data, we find that since the optimal solution is rounded to integers, the revenue under smaller U is more affected by rounding.

(3) In D-URP, near optimal result can be calculated with much lower time complexity. The average effect of D-URP can still achieve at least 97% of optimal solution in all considered scenarios. Note the time complexity of S-URP and D-URP is $O(IJU)$ [8] and $O(IJK + IK \log K)$ (as stated in Section 3), respectively. When $\lambda = 10000$ and $K = 10$, the time complexity is reduced by nearly three orders of magnitude and the effect of D-URP can still achieve 99% of optimal result.

6. Conclusion

In this paper, we propose a fast algorithm D-URP for resource placement in cloud platform considering stochastic demands with multidimensional distribution. Our analysis and the preliminary experiments indicate that D-URP can achieve at most 99% revenue of optimal solution, and compared with existing algorithms, its time complexity is reduced by nearly three orders of magnitude in our scenarios with general settings. Therefore, D-URP can be an effective supplement to existing algorithms under time-tense scheduling scenarios.

Our future interesting works may include more precise approximation of continuous functions using discrete ones and efficient algorithms to variant of resource placement problem with more constraints.

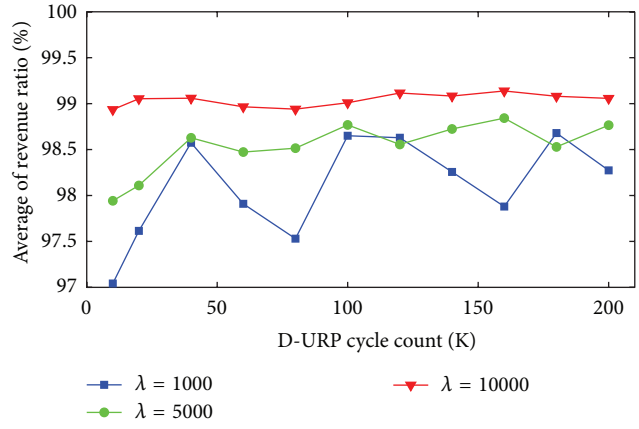


Figure 9: Average revenue ratio of D-URP to S-URP with different D-URP cycle counts under scenarios with different expected number of requests.

Appendix

Proof of Lemma 1. Note that for given constant C, it has

$$\Pr(\min(C, X) \le v) = \begin{cases} \Pr(X \le v) & v \le C \\ 1 & v > C. \end{cases} \quad (A.1)$$

Denote random variable $Y = \min(C, X)$ and by $\mathrm{cdf}_X(\cdot)$ and $\mathrm{cdf}_Y(\cdot)$; X and Y's cumulative distribution function (CDF) then (A.1) can be rewritten as

$$\mathrm{cdf}_Y(v) = \begin{cases} \mathrm{cdf}_X(v) & v \le C \\ 1 & v > C. \end{cases} \quad (A.2)$$

According to [21], for every real-valued nonnegative random variable Z with probability density function $\mathrm{cdf}_Z(\cdot)$, it has $E(Z) = \int_0^\infty (1 - \mathrm{cdf}_Z(v))dv$. Substituting Z by Y, we can get $E(\min(C, X)) = E(Y) = \int_0^\infty (1 - \mathrm{cdf}_Y(v))dv = \int_0^C (1 - \mathrm{cdf}_X(v))dv$. Then Lemma 1 is proved. □

Proof of Theorem 2. Denote $f(X) = \sum_{i=1}^N f_i(x_i)$. Suppose $f_a'(x_a) < f_b'(x_b)$ when maximal value of $f(X)$ is achieved. Since both $f_a'(x_a)$ and $f_b'(x_b)$ are monotonic decreasing function, then there must exist $x_a' > x_a$ and $x_b' < x_b$ with $x_a' + x_b' = x_a + x_b$ and satisfy $f_a'(x_a') = f_b'(x_b')$. Set $f_a'(x_a') = f_b'(x_b') = Z$; since $f_a(x_a') - f_a(x_a) = \int_{x_a}^{x_a'} f_a'(n) > Z(x_a' - x_a)$

and $f_b(x_b) - f_b(x_b') = \int_{x_b'}^{x_b} f_b'(n) < Z(x_b - x_b')$, then it has $f_a(x_a') - f_a(x_a) > f_b(x_b) - f_b(x_b')$; that is, $f_a(x_a') + f_b(x_b') > f_a(x_a) + f_b(x_b)$. It shows that for any X not satisfying $f_1'(x_1) = f_2'(x_2) = \cdots = f_N'(x_N)$, there exists another X' that satisfies the equation and $f(X') > f(X)$. Then Theorem 2 is proved. □

Proof of Corollary 3. We define $f_{i,j}(L_i^j)$ and $f_{i,j}'(L_i^j)$ as

$$\begin{aligned} f_{i,j}\left(L_i^j\right) &= \int_0^{L_i^j} \left(1 - \mathrm{cdf}_{i,j}(n)\right) dn, \\ f_{i,j}'\left(L_i^j\right) &= 1 - \mathrm{cdf}_{i,j}\left(L_i^j\right). \end{aligned} \tag{A.3}$$

For the following optimization problem

$$\begin{aligned} \max \quad & \left(\sum_{j=1}^{J} f_{i,j}\left(L_i^j\right)\right) \\ \text{s.t.} \quad & \sum_{j=1}^{J} L_i^j = L_i, \end{aligned} \tag{A.4}$$

where L_i is a constant for a given O and i, $L_i^j \in [0, \infty)$ and $f_{i,j}'(L_i^j) \in [0, 1]$ is monotonic continuous function. According to Theorem 2, when $f_{i,1}'(L_i^j) = f_{i,2}'(L_i^j) = \cdots = f_{i,J}'(L_i^j)$, maximal value is achieved. We set $f_{i,j}'(L_i^j) = v$ for any j and denote $G_i(v) = L_i$; the solution to problem in (A.4) is

$$\begin{aligned} L_i^j &= \mathrm{cdf}_{i,j}^{-1}(1 - v) = \mathrm{cdf}_{i,j}^{-1}(1 - v), \\ G_i(v) = L_i &= \sum_{j=1}^{J} L_i^j = \sum_{j=1}^{J} \mathrm{cdf}_{i,j}^{-1}(1 - v). \end{aligned} \tag{A.5}$$

We denote $G_i^{-1}(L_i) = v$; L_i^j can be written as follows:

$$L_i^j = \mathrm{cdf}_{i,j}^{-1}(1 - v) = \mathrm{cdf}_{i,j}^{-1}\left(1 - G_i^{-1}\left(L_i\right)\right), \tag{A.6}$$

where $G_i^{-1}(L_i) \in [0, 1]$ and $G_i^{-1}(0) = 1$. It is shown that there is a maximal value for any given L_i in (A.4); thus, the maximal value is a function of L_i and is denoted by $f_i^{\max}(L_i)$. $\max(E(R_i^{P'}))$ is also a function of L_i and is denoted by $h_i(L_i)$. Consider

$$\begin{aligned} \max\left(E\left(R_i^{P'}\right)\right) = h_i\left(L_i\right) = w_{\mathrm{sat}} \int_0^{L_i} \left(1 - \mathrm{cdf}_i(n)\right) dn \\ + w_{\mathrm{loc}} f_i^{\max}\left(L_i\right). \end{aligned} \tag{A.7}$$

Therefore, for given type-placement O and i, if $P' \in \{P\}^O$ and L_i^j satisfies (A.6), $\max(E(R_i^{P'}))$ is achieved since $f_i^{\max}(L_i)$ is achieved. Then Corollary 3 is proved. □

Proof of Corollary 4. Firstly $f_i'^{\max}(L_i)$ is expressed by

$$f_i'^{\max}\left(L_i\right) = \frac{f_i^{\max}\left(L_i + \Delta\right) - f_i^{\max}\left(L_i\right)}{\Delta}, \tag{A.8}$$

where Δ is a small constant. Assuming that $\{L_i^j\}$ $(1 \le j \le J)$ leads to $f_i^{\max}(L_i)$ and $\{L_i^j + \Delta_j\}$ $(1 \le j \le J)$ leads to $f_i^{\max}(L_i + \Delta)$, where $\Delta = \sum_{j=1}^{J} \Delta_j$, then $f_i'^{\max}(L_i) = \sum_{j=1}^{J}(f_{i,j}(L_i^j + \Delta_j) - f_{i,j}(L_i^j))/\Delta = \sum_{j=1}^{J} \Delta_j f_{i,j}'(L_i^j)/\Delta = f_{i,j}'(L_i^j) = v = G_i^{-1}(L_i)$.

Then $h_i'(L_i)$ of $h_i(L_i)$ in (A.7) can be rewritten as

$$h_i'\left(L_i\right) = w_{\mathrm{sat}}\left(1 - \mathrm{cdf}_i\left(L_i\right)\right) + w_{\mathrm{loc}} G_i^{-1}\left(L_i\right). \tag{A.9}$$

According to Corollary 3, maximization problem is written as

$$\begin{aligned} \max \quad & \left(\sum_{i=1}^{I} h_i\left(L_i\right)\right) \\ \text{s.t.} \quad & \sum_{i=1}^{I} L_i = U. \end{aligned} \tag{A.10}$$

Since U is a constant, $L_i \in [0, \infty)$ and $h_i'(L_i) \in [0, w_{\mathrm{sat}} + w_{\mathrm{loc}}]$ is monotonic continuous function. According to Theorem 2, when $h_1'(L_1) = h_2'(L_2) = \cdots = h_I'(L_I)$, $\max(\sum_{i=1}^{I} h_i(L_i))$ is achieved. We set $h_i'(L_i) = s$ for any i and denote $H(s) = U$; then it has

$$H(s) = U = \sum_{i=1}^{I} L_i = \sum_{i=1}^{I} h_i'^{-1}(s). \tag{A.11}$$

And L_i can be rewritten as a function of U as follows:

$$L_i = h_i'^{-1}(s) = h_i'^{-1}\left(H^{-1}(U)\right). \tag{A.12}$$

Therefore, for given $U > 0$, if a $O' \in \{O\}^U$ and its L_i satisfies formula above, $\max(\sum_{i=1}^{I} h_i(L_i))$ is achieved. Then Corollary 4 is proved. □

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper. The authors have no financial and personal relationships with other people or organizations that can inappropriately influence our work; there is no professional or other personal interest of any nature or kind in any product, service, and/or company that could be construed as influencing the position presented in or the review of this paper.

Acknowledgments

This paper is supported by the National Natural and Science Foundation of China (nos. 61003052 and 61103007), Natural Science Research of the Education Department of Henan Province (nos. 2010A520008, 13A413001, and 14A520018), Henan Provincial Key Scientific and Technological Plan (no. 102102210025), Program for New Century Excellent Talents of Ministry of Education of China (no. NCET-12-0692), Doctor Foundation of Henan University of Technology (nos. 2012BS011 and 2013BS003), and Plan of Nature Science Fundamental Research in Henan University of Technology.

References

[1] YouTube, http://www.youtube.com.

[2] http://www.facebook.com/.

[3] Amazon EC2 home page, http://aws.amazon.com/ec2/.

[4] Microsoft Azure home page, http://www.windowsazure.com.

[5] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119–128, May 2007.

[6] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proceeding of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 460–468, Orlando, Fla, USA, March 2012.

[7] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahnt, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, pp. 1–14, October 2007.

[8] Y. Rochman, H. Levy, and E. Brosh, "Resource placement and assignment in distributed network topologies," in *Proceeding of the 32nd IEEE Conference on Computer Communications (INFOCOM '13)*, pp. 1914–1922, Turin, Italy, April 2013.

[9] Y. Wu, C. Wu, B. Li, X. Qiu, and F. C. M. Lau, "CloudMedia: When cloud on demand meets video on demand," in *Proceedings of the 31st International Conference on Distributed Computing Systems*, pp. 268–277, July 2011.

[10] X. Nan, Y. He, and L. Guan, "Queueing model based res ource optimization for multimedia cloud," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 928–942, 2014.

[11] P. D. Kaur and I. Chana, "A resource elasticity framework for QoS-aware execution of cloud applications," *Future Generation Computer Systems*, vol. 37, no. 7, pp. 14–25, 2014.

[12] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in a shared internet hosting platform," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 239–254, 2002.

[13] L. He, D. Zou, Z. Zhang, C. Chen, H. Jin, and S. A. Jarvis, "Developing resource consolidation frameworks for moldable virtual machines in clouds," *Future Generation Computer Systems*, vol. 32, pp. 69–81, 2014.

[14] J. Broberg, R. Buyya, and Z. Tari, "MetaCDN: harnessing storage clouds for high performance content delivery," *Journal of Network and Computer Applications*, vol. 32, no. 5, pp. 1012–1022, 2009.

[15] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, "Optimizing cost and performance for content multihoming," in *Proceeding of the ACM SIGCOMM Conference Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '12)*, pp. 371–382, New York, NY, USA, August 2012.

[16] C. Papagianni, A. Leivadeas, and S. Papavassiliou, "A cloud-oriented content delivery network paradigm: modeling and assessment," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 287–300, 2013.

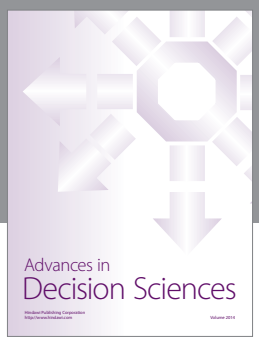
[17] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. Lau, "Moving big data to the cloud: an online cost-minimizing approach," *Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

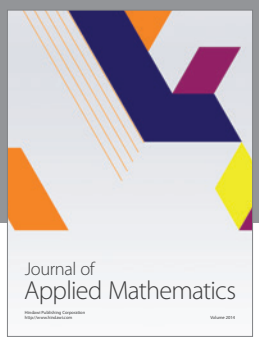
[18] L. Zeng and Y. Wang, "Optimization on content service with local search in cloud of clouds," *Journal of Network and Computer Applications*, vol. 40, no. 4, pp. 206–215, 2014.

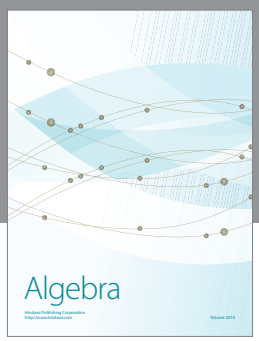
[19] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," in *Proceedings of the IEEE (INFOCOM '11)*, pp. 694–702, Shanghai, China, April 2011.

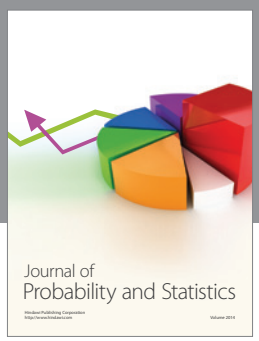
[20] Y. Zhou, T. Z. Fu, and D. M. Chiu, "Statistical modeling and analysis of p2p replication to support vod service," in *Proceedings of IEEE INFOCOM*, pp. 945–953, 2011.

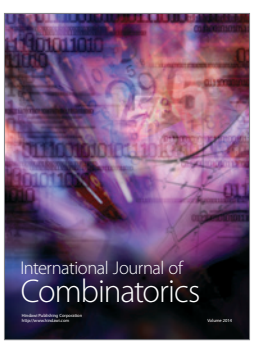
[21] S. M. Ross, *Introduction to Probability Models*, Academic press, 2006.

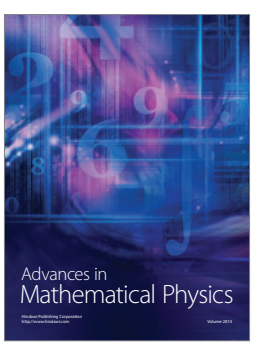


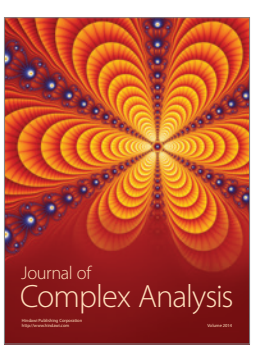


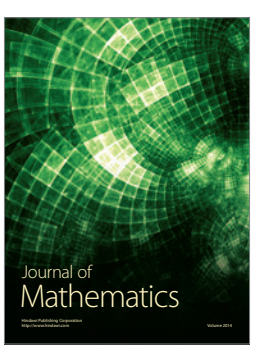


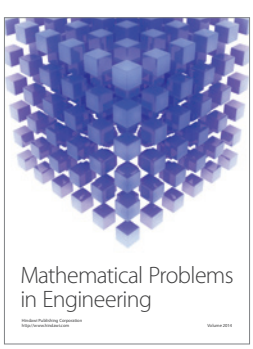


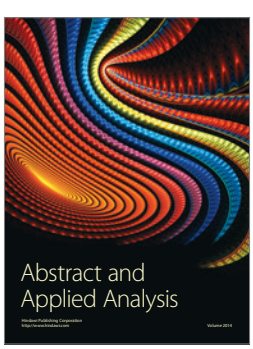


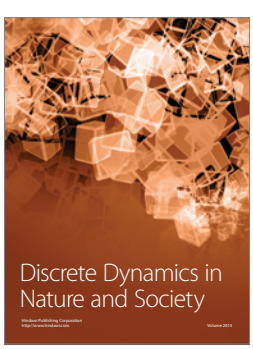


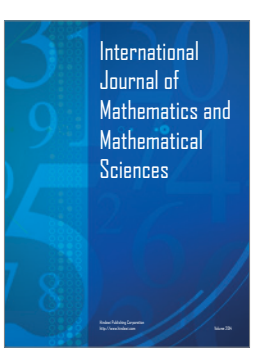












Hindawi

Submit your manuscripts at
http://www.hindawi.com