Resource Placement and Assignment in Distributed Network Topologies

Yuval Rochman
Tel Aviv University
Tel-Aviv, Israel
yuvalroc@post.tau.ac.il

Hanoch Levy
Tel Aviv University
Tel-Aviv, Israel
hanoch@cs.tau.ac.il

Eli Brosh
Vidyo
New Jersey, USA
eli@vidyo.com

***Abstract*—We consider the problem of how to place and efficiently utilize resources in network environments. The setting consists of a regionally organized system which must satisfy regionally varying demands for various resources. The operator aims at placing resources in the regions as to minimize the cost of providing the demands. Examples of systems falling under this paradigm are 1) A peer supported Video on Demand service where the problem is how to place various video movies, and 2) A cloud-based system consisting of regional server-farms, where the problem is where to place various contents or end-user services. The main challenge posed by this paradigm is the need to deal with an arbitrary multi-dimensional (high-dimensionality) stochastic demand. We show that, despite this complexity, one can optimize the system operation while accounting for the full demand distribution. We provide algorithms for conducting this optimization and show that their complexity is pretty small, implying they can handle very large systems. The algorithms can be used for: 1) Exact system optimization, 2) deriving lower bounds for heuristic based analysis, and 3) Sensitivity analysis. The importance of the model is demonstrated by showing that an alternative analysis which is based on the demand means only, may, in certain cases, achieve performance that is drastically worse than the optimal one.**

I. Introduction

The Internet has witnessed a surge in the usage of content distribution services, such as video streaming applications. These services feature large volumes of content and demands that are highly variable, and replicate content across multiple locations for better performance and availability. Such systems face the general problem of how to place and efficiently utilize (content) resources in a geographically distributed environment, usually organized in a regional structure, and demands for the resources which vary across the regions. In such settings, it is typically cheaper to provide a specific demand by a resource located in the same region, as opposed to providing it by a remote one. Thus, a challenge in engineering such systems is, first, how to place the resources in the various regions, and second, how to assign (provide) them as to optimize the operations of the system.

An example of an application that falls under this paradigm is a peer-assisted Video-on-Demand (VoD). In this service, a provider aims to serve video content to users which are spread across various geographical regions and are each equipped with a set-top box. Serving all requests from a central server can lead to bottlenecks, limiting the system's scalability. It can therefor make use of the users' boxes to store video content and serve it, using their upload channel, to other users. Serving videos across regions is more expensive than within a region. The provider faces the problem of where to place copies of the various movies so as to minimize the expected cost of servicing video requests. Another related example is that of a cloud-based service provider with geo-diverse users. To be close to the users, it maintains a number of server farms, each in the relative proximity of a target population. It is of course preferable to service a request from a a local farm. Ideally, the providers would like to pre-load the right content replicas at individual servers so as to satisfy the content demand in the most efficient way.

To address this problem, we consider a generic resource placement and assignment model for distributed network topologies. The system considered consists of multiple regions (areas) and resource types. The total number of resources that can be placed in each region can be restricted. The provider faces a stochastic demand in the form of a multi dimensional demand distribution, consisting of the demand distribution for each service and in each region. These distributions are neither necessarily identical nor necessarily independent of each other.

The provider's objective is to place the resources optimally over the regions – based on the demand distribution, and once placed, to assign them efficiently to the requests – based on the demand realization. Supplying each request can be done from the same region (low cost) or from a different region (medium cost); not supplying the demand incurs high cost, e.g., due to fall back to a central server. The objective of the combined placement-assignment algorithm is to minimize the expected cost of supplying the requests.

The generality of the distribution described above allows one to address a variety of scenarios: 1) Demands that vary across resources. For example, in VoD the demand can change drastically across movies; while some are very popular the others may be quite esoteric. In fact, studies suggest a large diversity of usage patterns exhibited by existing VoD services such as NetFlix, IPTV, and YouTube [1], [2]. 2) Demands that vary across regions or that may be correlated with each other. 3) Demands with various variabilities. High variability demands may be caused, for example, by flash crowds or by some network events (e.g. a campaign for a movie).

Our replication model, which is based on the demand full distribution, may deviate from prior approaches focused on the mean of the distribution. The use of a simple scheme like proportional mean replication may be quite inefficient (Proportional mean replication has been shown to be optimal under some conditions, see e.g. [3]) as demonstrated in the following simple example. Consider a single region system which can store up to n resources, and two types of resource to be placed. The first resource-type has a deterministic demand of n requests and the second one has a stochastic demand of nk^2 requests w.p $1/k$, and 0 requests w.p $1 - 1/k$. The objective of efficient placement is to maximize the expected number of requests granted (the system's revenue). The optimal algorithm will place n resources of the first type, yielding a revenue of n. In contrast, a proportional mean strategy which tunes the number of replicas to the average number of requests, will place $n/(k + 1)$ resources of the first type and $\frac{nk}{k+1}$ of the second type yielding a drastically smaller revenue of $2n/(k+1)$. Note that as k approaches infinity, the performance ratio between the policies approaches ∞. A further applicative example is discussed in Section VII.

Solving a resource allocation problem that combines combinatorial aspects and arbitrary stochastic demand distribution is challenging. In our prior work [4] we showed that a simpler variant of the problem, in which demands and capacity constraints are identical across regions, can be solved by an optimal greedy algorithm. The methodology was generalized in [5] to account for a global capacity restriction. However, these analysis techniques are not powerful enough to address a general asymmetric setting.

We demonstrate that, despite this complexity, one can optimize servicing costs while accounting for the full demand distribution by transforming the placement problem into min-cost flow problem. The min-cost problem is traditionally solved by the SSP algorithm. We further develop an optimized version of the SSP algorithm, tailored to our problem, provide low-complexity optimal algorithms suitable for handling large-scale systems. Our algorithms can serve as a lower bound for heuristic approaches, and can be used for sensitivity analysis. We use numerical analysis to validate our results and to demonstrate that alternative mean-based analysis, may, in certain cases, achieve performance that is drastically worse than the optimal one

The rest of this paper is organized as follows. In Section III we describe the model and the problem. Section IV deals with the matching problem and Sections V, VI-D with the placement problem. Section V describes our transformation of the problem into a max-flow problem and Section VI-D describes our optimized variant of the SSP algorithm which is a key to achieving a low complexity solution. Section VII uses numerical analysis to evaluate the system performance. Most of the article proofs are presented in a technical report [6].

II. Related Work

The problem of finding the right replication of resources in a distributed network is often addressed using facility location theory [7]. This area has received significant attention with respect to analytic analysis and algorithmic solutions. Our version of the problem differs than traditional facility location problems in that it incorporates stochastic demand and capacity constrains at the service facilities.

There is an extensive literature on efficient replication of content resources in the context of content distribution networks and web caches (see e.g. [8], [9]). These services are typically distributed across multiple locations for better performance and reliability. In this domain, the main focus is on optimizing the success ratios at servers given some access pattern (typically, a flow) of requests. These works pay little attention to capacity limits at the servers, an essential aspect of the model (and applications) we consider.

Our motivating application scenario of reducing video servicing costs through optimized replication has been considered in the context of P2P systems. [3] was perhaps the first to study network model similar to our with an exponentially expanding topology for file sharing systems. Their main result is that proportional replication, i.e., one based on the mean demand, minimizes the average number of links traversed in a download process. Similar result was derived in [10] for a mesh network setting. These works assume that number of files and peers is large enough that all possible requests are always be served. In contrast, our model allows for restricting the number of resources (files), resulting in max-percentile based replication, namely, one based on the tail distribution of the demand. Other works in this domain often focus on different goals. For example, [11] optimized file availability when peers are infrequently online, while [12] aimed to minimize the number of access failures under random assignment policy.

Other relatively close works to ours are [13], [14], [15] that focused on P2P VoD replication systems. [13] proposes an optimal replication algorithm, called RLB, based on the assumption that the number of movies is much smaller than the number of peers. [14] argues for proportional replication by applying asymptotic analysis to loss network performance model. These works focus on small-scale flat networks, and do not consider hierarchical network topologies used in practice by providers. [15] proposes placement framework for large-scale VoD service based on mixed integer program. While their model accounts for arbitrary demand pattern and network structure it assumes deterministic demand, whereas we consider stochastic one. Recently, [16] characterized the service efficiency of distributed content platforms as function of servers' storage size by using an asymptotic performance model for online matching algorithms. In this context, our work maps to a single content server storage model, which enables us to solve the combined optimization problem of matching and placement using exact analysis

This works solves a general setting that allows for asymmetric demands and different resource capacity restrictions across regions. Our previous work [4] focused on a simpler variant of the problem with symmetric demands and symmetric capacity restrictions, and proposed an optimal greedy algorithm based on max-percentile approach with near-linear complexity. While the solution technique can be generalized

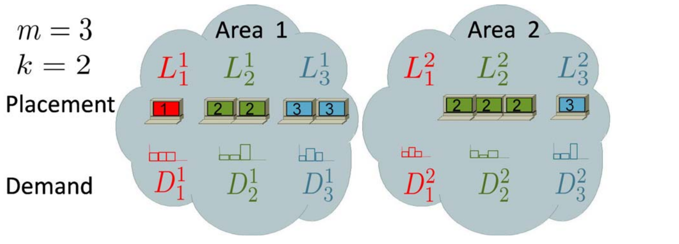


Fig. 1. An example of the system topology. Note that the storage is $s^1 = 5$ and $s^2 = 4$ and the placement is $L_1^1 = 1$, $L_1^2 = 0$, $L_2^1 = 2$ etc.

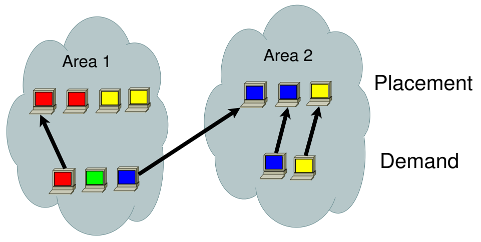


Fig. 2. Matching Example

to account for an aggregate system capacity restriction, as done in [5], it cannot be applied to asymmetric settings. To address the general problem, we build on the min-cost flow problem solved by the SSP algorithm (see e.g. [17], [18]). We leverage the special structure of our problem to establish a further optimization of SSP and obtain a low complexity solution.

III. The model and the problem

The system consists of k *areas* numbered by $1, 2, \dots, k$. In every area one can place multiple resources of different *types* numbered $1, 2, \dots, m$. The set of resources placed in these areas is called a *placement*. We assume that area $j = 1, 2, \dots, k$ is associated with a storage value s_j representing a bound on the number of resources that can be placed in area j. Placement L is called *feasible* if the number of resources in area j is not larger than s^j, i.e $L^j := \sum_{i=1}^{m} L_i^j \leq s^j$.

We consider a stochastic demand reflecting the demand at peak hours. Let D_i^j be a random variable denoting the number of requests for type-i resource in area j. We do not make any assumption on the distribution of D_i^j, namely it can be of an **arbitrary distribution**. Further, we **do not assume independence** between the demands, namely $D_{i_1}^{j_1}$ and $D_{i_2}^{j_2}$ are not necessarily mutually independent. In contrast the work we done in [4], we **do not** assume that the areas (regions) are symmetric. We assumed that the demand c.m.f values $\Pr(D_i^j \geq e)$ are calculated in $O(1)$. We assumed that the statistics are calculated by an external data base. An example of the topology of system is depicted in Fig. 1, where every computer represents a resource.

Consider a request made in area j and a placement L. If the request is assigned to a resource in L, then the request is called *satisfied*. If the request is satisfied, it is assigns to either: 1) A resource of L located in area j (and therefore the request is granted *locally*). 2) A resource of L located in a different area (in this case, the request is granted *remotely*). We denote the costs of those cases by C_{loc} (local cost), and C_{rem} (remote cost). The costs obeys $C_{loc} \leq C_{rem}$ since granting a remote demand is not cheaper than granting it locally. If a request is not assigned to any resource in L, then it is called an *unsatisfied* request. The cost of an unsatisfied is denoted by C_{unsat}, naturally obeying $C_{loc}, C_{rem} \leq C_{unsat}$.

Remark 3.1: In some applications, the unsatisfied requests are served by an external resource center, as in the peer-assisted video on demand (VoD) system.

We there for have:

$$C_{loc} \leq C_{rem} \leq C_{unsat}. \tag{1}$$

The objective of the system is to minimize the cost of servicing the requests. Let g_{loc}, g_{rem} and g_{unsat} denote the number of requests granted from a local area, requests granted from a remote area and unsatisfied requests, respectively. Note that $g_{loc} + g_{rem} + g_{ext}$ is the total number of requests, D. The system request service cost is given by

$$C = C_{loc} \cdot g_{loc} + C_{rem} \cdot g_{rem} + C_{unsat} \cdot g_{unsat}, \tag{2}$$

and the objective is to minimize this service cost.

An example of the matching between resources and requests is in Fig. 2. In this example, we have $k = 2$ areas, each with $s^1 = 4$, $s^2 = 3$ storage. The number of requests granted (served) locally, remotely and unsatisfied, are 1, 1 and 3, respectively. Therefore the cost in this case is $3 \cdot C_{loc} + C_{rem} + C_{unsat}$.

To this end, one should note that the system operation divides into two stages. First, at off-line mode, we place the resources into the areas. The resource placement is based on the knowledge of the demand distribution. Second, once the resources are placed, the system is faced with an actual demand, which is a *realization* of the demand distribution, at which time the system needs to decide how to assign the resources to the various demands. We call the former the *placement problem* and the latter the *assignment problem*. Note that the assignment problem can be solved in isolation; nonetheless the solution of the placement problem depends on that of the assignment problem. Formally the problems can be stated as follows:

1) The *assignment (matching) problem*: Given a placement, $L = \{L_i^j\}$, a demand realization, denoted by n_i^j, $i = 1, \dots, m$, $j = 1, \dots, k$, and the service cost parameters $C_{loc}, C_{rem}, C_{unsat}$, assign (match) the resources to the demands as to minimize the service cost C.
2) The *placement problem*: Given the demand distributions $\{D_i^j\}$, $i = 1, \dots, m$, $j = 1, \dots, k$, the service cost parameters $C_{loc}, C_{rem}, C_{unsat}$, area storages $s^1, s^2, \dots s^k$ and a matching algorithm solving the assignment problem, determine the optimal placement resources $L = \{L_i^j\}$, $i = 1, \dots, m$, $j = 1, \dots, k$, that minimizes the expected cost $E[C]$ among all feasible placements obeying $L^j = \sum_{i=1}^{m} L_i^j \leq s^j$.

Our objective in this work is to solve the placement problem; this, in turn, will be assisted by a solution of the assignment problem.

A. Transforming the Cost Function to a Revenue Function

For the analysis of the assignment and placement problem it will be convenient to transform the cost value problem to a revenue value problem. The way we define the transformation is critical, and defining it differently may complicate the analysis. The transformation was established in [4] as follows:

Claim 3.2: The following holds:

1) A matching algorithm M solves the assignment problem iff M maximizes the following function:

$$R = (C_{unsat} - C_{rem})g_{sat} + (C_{rem} - C_{loc})g_{loc}$$

where g_{sat}, g_{loc} represent the number of requests satisfied by the placement (which equals $g_{loc} + g_{rem}$) and the number of requests granted by matching same area (local), respectively.

2) A placement L solves the placement problem iff the placement maximizes $E(R)$.

For convenience we set $R_{sat} \doteq C_{unsat} - C_{rem} \geq 0$ and $R_{loc} \doteq C_{rem} - C_{loc} \geq 0$. We will denote the *revenue objective function* to be:

$$R = R_{sat} \cdot g_{sat} + R_{loc} \cdot g_{loc}. \quad (3)$$

By Claim 3.2 we have that R is a proper objective function for both the assignment problem and the placement problem.

IV. The assignment problem and the optimal matching algorithm

In [4] we solved the assignment problem using the *assignment algorithm*. The results of [4] are that the assignment algorithm maximizes the revenue and when applied an realization d_i^j, $i = 1, \ldots, m$, $j = 1, \ldots, k$, it yields:

$$R = R_{sat}\sum_{i=1}^{m} \min(L_i, d_i) + R_{loc}\sum_{i=1}^{m}\sum_{j=1}^{k} \min(L_i^j, d_i^j). \quad (4)$$

The results further state that for any arbitrary placement L and stochastic demand $\{D_i^j\}$ it will maximize the expected revenue yielding:

$$E(R^L) = \\ R_{sat}\sum_{i=1}^{m} E(\min(L_i, D_i)) + R_{loc}\sum_{i=1}^{m}\sum_{j=1}^{k} E(\min(L_i^j, D_i^j)). \quad (5)$$

The solution of the placement problem is to maximize the function in (5), when the free variables of the function are the placement resources, L_i^j, under the feasibility constrain, i.e the number of resources in area j is not larger than s^j ($\sum_{i=1}^{m} L_i^j = L^j \leq s^j$).

V. A Solution for the Placement problem

The placement problem as defined above seems to be, on its face value, of prohibitively high complexity and therefore challenging. The reason is that one must deal with a very large input data (m distributions, where each could be represented with $O(s)$ data elements, where in some applications s and m can reach values up to 10^4), combined with the combinatorial complexity of the assignment problem and the placement problem. Fortunately, we are able to establish powerful properties of arbitrary distribution functions, and therefore of our objective function, to be presented in Eq. 6 and Theorem 5.1 next. These properties will be later utilized to devise efficient algorithms for the placement problem.

A. Transformation of the cost function

In order to solve the placement problem, we will use Eq. 5 and transform from a revenue maximization problem to a cost minimization problem. We define the *alternative cost* of placement L in the following formula:

$$E(C^L) = R_{sat}\sum_{i=1}^{m}\sum_{n=1}^{L_i}(1 - \Pr(D_i \geq n)) + \\ R_{loc}\sum_{j=1}^{k}\sum_{i=1}^{m}\sum_{n=1}^{L_i^j}(1 - \Pr(D_i^j \geq n)) \quad (6)$$

Note that there are three main differences between the alternative cost and the system request service cost defined in Eq. 2: 1) As the number of total resources in a placement increases (i.e $\sum_{i=1}^{m}\sum_{j=1}^{k} L_i^j$)), the number of satisfied requests increases, and therefore the system service cost decreases. This is in contrast to Eq. 6, which increases as the number of total resources in a placement increases. 2) The system service cost contains three different types of cost parameter, while the alternative cost contains only two types. 3) The placement with the alternative minimal cost is the zero placement ($L_i^j = 0$), while the optimal placement, which is definitely not the zero placement, must minimize Eq. 2. We can solve the placement problem using the alternative cost as stated in the following theorem:

Theorem 5.1: If L minimizes $E(C^L)$ among placements such that for all area $1 \leq j \leq k$ we have $L^j = \sum_{i=1}^{m} L_i^j = s^j$ (s^j is the bound on the resource storage in area j), then L is an optimal placement.

We prove the theorem by the fact that every discrete non-negative random variable X and every constant C satisfies $E(\min(X, C)) = \sum_{k=1}^{C} \Pr(X \geq k)$, and proving that the optimal placement must satisfies $L^j = \sum_{i=1}^{m} L_i^j = s^j$. The full proof of the theorem can be found in the technical report [6].

B. A reduction to the min-cost flow problem

In order to find an optimal solution to the placement problem, we present the *min-cost flow problem*, which is a generalization of the notable max flow problem. In the problem, one considers a directed graph $G = (V, E)$ where every edge $e \in E$ has integer *capacity* $c(e)$ and a real-value non-negative *weight* $w(e)$ (alternatively, called *cost*). The graph must contain two different nodes: a source node x and a sink node y. An *x-y-flow* $f : E \rightarrow R^+$ is defined on the graph edges $(v, v') \in E$ in the same way as defined in the max-flow problem and must satisfy: 1) *Capacity constraint:* for each edge e, we have $0 \leq f(e) \leq c(e)$. 2) *Conservation of flows:* for every vertex $v \in V \setminus \{x, y\}$ we have $\sum_{(v',v)\in E} f(v', v) = \sum_{(v,v')\in E} f(v, v')$. In addition to the standard definitions , we define the *flow in node* $v \neq x, y$ as the income flow (and by conservation of flows, the outcome flow) to (from) node v. We denote it by $f^{in}(v)$, which equals $f^{in}(v) = \sum_{(v,v')\in E} f(v, v')(= \sum_{(v',v)\in E} f(v', v) = f^{out}(v))$. The *flow value* of f, as defined in the max-flow problem, is $|f| = \sum_{(x,v)\in E} f(x, v) = \sum_{(v,y)\in E} f(v, y)$. The *weight (or cost) of flow* f is $w(f) = \sum_{e\in E} f(e)w(e)$.

The *minimum-cost flow* problem with required flow $|f| = k$ is to find a flow f_{opt} of value k that has minimal weight among all flows of value k. This means that for every flow f' such that $|f'| = |f_{opt}| = k$ we have $w(f_{opt}) \leq w(f')$.

In Section VI-A we present the Successive Shortest Path algorithm for solving a min-cost flow problem on a general graph $G = (V, E)$. This algorithm was studied in [18]. Its time complexity is $O(|f||E||V|)$ where $|f|$ is the required flow value.

In the next subsection, we will reduce the placement problem into a min-cost flow problem. To this end, we define a directed graph $G = (V_1 \bigcup V_2 \bigcup \ldots V_k, E)$ to be a *k-layer graph* if the following holds: 1) The vertex sets $V_1, V_2, \ldots V_k$ are pairwise disjoint. 2) The vertices in V_i, which are called *layer-i* vertices, can be connected only to vertices in succussive layers (i.e if $(u, v) \in E$ then there is $1 \leq i \leq k$ such that $u \in V_i$ and $v \in V_{i+1}$). Note that a 2-layer graph definition is the equivalent definition of a bipartite graph.

C. Transformation to min-cost flow problem

We use Theorem 5.1 to reduce the placement problem into a min-cost flow problem in a 7-layer graph G^7. That means, finding the min-cost flow f_{opt} in G^7 will imply the optimal placement L that solves the placement problem.

The reduced graph $G^7 = (V'_1 \bigcup V'_2 \ldots \bigcup V'_7, E')$ (see Fig. 3) contains the source node x in the first layer ($V_1 = \{x\}$) and contains the sink y in layer-7 ($V_7 = \{y\}$). We set the required flow value to equals the total storage value over all areas (i.e $|f| = s = \sum_{i=1}^{k} s^k$, where s^j is the bound on the number of resources that can be placed in area j). Layer-2 contains *area nodes* denoted as $a^1, a^2, \ldots, a^k$. The min-cost flow f_{opt} in node a^j (i.e $f^{in}_{opt}(a^j)$) will represent the total resources placed allocated to area j (i.e $f^{in}_{opt}(a^j) = L^j$). The edges between the source x and area node a^j have capacity s^j, reflection a bound on the number of resources to be placed in area j. The cost of these edges are all 0. Layer-3 will contain pairs of *(area, type) nodes* (a^j, t^i) such that the flow in these nodes represents the storage in area j of resource type i (i.e $f^{in}_{opt}(a^j, t^i) = L^j_i$). We connect between layer-2 area nodes (a^j) and layer-3 (area, type) nodes (a^j, t^i) for all resource types $1 \leq i \leq m$ and all areas $1 \leq j \leq 1$. The capacity of these edges is infinity (the flow through the nodes are unlimited), and the cost is 0.

In order to represent the local part of the cost $E(C^L)$, the layer-4 nodes are the triples (a^j, t^i, r) called *(area, type, # resources) nodes* for areas $1 \leq j \leq k$, resource types $1 \leq i \leq t$ and an integer $1 \leq r \leq s$. Positive flow in node (a^j, t^i, r) represents that the number of type-i resources from area j is larger than or equal to r (in other words, if $L^j_i \geq r$, then $f^{in}_{opt}(a^j, t^i, r) = 1$, otherwise $f^{in}_{opt}(a^j, t^i, r) = 0$). To allow the flow values of 0 or 1, the edge connected between node (a^j, t^i) in layer-3 and node (a^j, t^i, r) in layer-4 will have capacity 1. Note that $L^j_i \geq r$ iff the number $R_{loc}(1 - \Pr(D^j_i \geq r))$ appears in the summational of Eq. 6. Thus, we define cost between nodes (a^j, t^i) and (a^j, t^i, r) to be $R_{loc}(1 - \Pr(D^j_i \geq r))$.

Layer-5 nodes are called *resource type nodes* or simply *type nodes*. The value $f^{in}_{opt}(t^i)$ represents the number of type-i resources from all areas (i.e $f^{in}_{opt}(i) = L_i$). For all $1 \leq j \leq k, 1 \leq r \leq s$, we connect layer-4 nodes (a^j, t^i, r) to resource type node t^i. The capacity of these edges is infinity (could equivalently set to 1, since the flow in (a^j, t^i, r) is 0 or 1), and the cost of these edges is 0. Layer-6 nodes represent pairs (t^i, r) of *(type, # resources) nodes*. Positive flow in node (t^i, r) represents if number of type-i resources is larger than(or equal to) r. Similarly to the edges between layer-3 and layer-4, the edges between t^i and (t^i, r) have capacity 1 and cost $R_{sat}(1 - \Pr(D_i \geq r))$. Finally, we connect all (resource type, # resources) nodes to the sink node y in layer-7. The cost of these edges is 0 and the capacity is infinity.

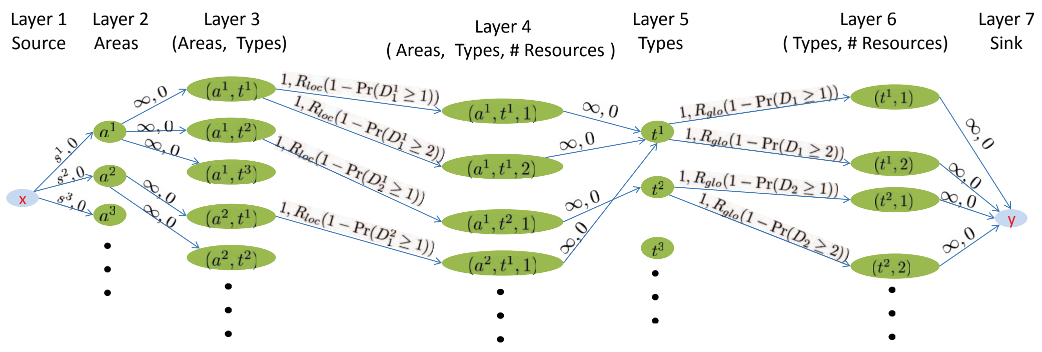
The reduction correctness relies on the following lemma:

Lemma 5.2: Let f be a flow in G^7, such that the flow in (area, type) node (a^j, t^i) is $f^{in}(a^j, t^i) = f^j_i$ and in resource type node t^i it is $f^{in}(t^i) = f_i$. Let us denote placement L such that $L^j_i = f^j_i$. Then there exists a unique flow $\hat{f}$ that obeys: 1) $\hat{f}^{in}(a^j, t^i) = f^j_i, \hat{f}^{in}(t^i) = f_i$ for all area j and resource type i. 2) $\hat{f}$ goes through layer-4 nodes $(j, i, 1), (j, i, 2), \ldots (j, i, f^j_i)$ and through layer-6 nodes $(i, 1), (i, 2), \ldots (i, f^j_i)$. Moreover, the cost of $\hat{f}$, which equals to $E(C^L)$, is not larger than the cost f.

The important point of Lemma 5.2 is that the optimal flow goes through first f^j_i cost edges emanating (a^j, t^i) node, thus reflecting consistent cost of placing f^j_i resources of type i in area j. The proof of this lemma can be found in our technical report [6]. Thus, by Theorem 5.1 we can conclude:

Corollary 5.3: If f_{opt} is a min cost flow of G^7, then the placement $L^j_i = f^{in}_{opt}(i, j)$ solves the placement problem.

Note that the number of (area, type, # resources) nodes is $O(smk)$. Thus, the number of vertices and the number of edges in G^7 is $O(smk)$. Running the Successive Shortest Path on $G^7 = (V', E')$ takes $O(|f||V'||E'|)$, where $|f| = O(s)$ is the required flow value. Expressing the time complexity of SSP in terms of total storage, resource types and areas (denoted as s, k, m) is $O(s^3m^2k^2)$. This time complexity is quite high, and therefore in the next section we will further develop this algorithm to yield a more efficient one, the Bipartite Graph (BG) algorithm.



Layer 1 Source
Layer 2 Areas
Layer 3 (Areas, Types)
Layer 4 (Areas, Types, # Resources)
Layer 5 Types
Layer 6 (Types, # Resources)
Layer 7 Sink
x
$s^1, 0$
$s^2, 0$
$s^3, 0$
a^1
a^2
a^3
$\infty, 0$
(a^1, t^1)
(a^1, t^2)
(a^1, t^3)
(a^2, t^1)
(a^2, t^2)
$1, R_{loc}(1 - \Pr(D_1^1 \geq 1))$
$1, R_{loc}(1 - \Pr(D_1^1 \geq 2))$
$1, R_{loc}(1 - \Pr(D_2^1 \geq 1))$
$1, R_{loc}(1 - \Pr(D_1^2 \geq 1))$
$(a^1, t^1, 1)$
$(a^1, t^1, 2)$
$(a^1, t^2, 1)$
$(a^2, t^1, 1)$
t^1
t^2
t^3
$1, R_{glo}(1 - \Pr(D_1 \geq 1))$
$1, R_{glo}(1 - \Pr(D_1 \geq 2))$
$1, R_{glo}(1 - \Pr(D_2 \geq 1))$
$1, R_{glo}(1 - \Pr(D_2 \geq 2))$
$(t^1, 1)$
$(t^1, 2)$
$(t^2, 1)$
$(t^2, 2)$
y

Fig. 3. The 7-layer graph

VI. Bipartite Graph (BG) Algorithm

In order to present the *Bipartite Graph* (in short *BG*) algorithm, we first have to describe the Successive Shortest Path (SSP) algorithm in the next subsection. In Subsection VI-B we describe the *Bipartite-like graph* G_f^B which the BG algorithm relies on. In Subsection VI-C we study how to construct the Bipartite-like graph G_f^B effectively, and finally we give the implementation of the Bipartite Graph (BG) algorithm in subsection VI-D.

A. The Successive Shortest Path (SSP) algorithm

The SSP algorithm is a well-known algorithm that solves the min-cost flow problem for general graphs. Given a flow f on a general graph $G = (V, E)$, the Successive Shortest Path (SSP) similarly to most of the max-flow algorithms, uses the *residual graph* $G_f = (V, E_f)$. On the residual graph edges we define weight w_f and capacity c_f. We construct the residual graph G_f from graph G and from flow f by the following steps: 1) We add to G_f edges from G, such that edge $(v, v') \in E$ will have weight $w_f(v, v') = w(v, v')$ and capacity of $c_f(v, v') = c(v, v') - f(v, v')$. 2) We add the reverse edges of G. That means, if $(v, v') \in E$, then we add edge (v', v) to G_f with weight $w_f(v', v) = -w(v, v')$ and capacity of $c_f(v', v) = f(v, v')$. Note that for every edge e in G_f we have $c(e) \geq 0$. 3) For every edge e with capacity $c(e) = 0$ we update its weight to be $w_f(e) = \infty$.

It is vital to be familiar with shortest path algorithms, since they are used by the SSP algorithm. One notable algorithm to calculate the shortest paths from one source v to all other vertices, is the *Bellman-Ford* algorithm. The running time of Bellman-Ford on graph $G = (V, E)$ is $O(|E| \cdot |V|)$. More information on Bellman-Ford can be found in [19].

Finally, we present the SSP algorithm on a general graph G with required flow k. In the initial step, the algorithm assigns the zero flow $f := 0$ (i.e $f(e) = 0$ for all $e \in E$) with flow value $|f| = 0$ and constructs the flow residual graph G_f. The algorithm works iteratively, and in the i^{th} iteration it calculates a minimum-cost flow of flow value larger than or equal to i. The algorithm execute the following steps in every iteration: 1) Check if the flow value $|f|$ equals to the required flow k. If so, then f is the optimal flow and the algorithm terminates. 2) Calculate the shortest paths from source x on G_f with respect to weight function w_f. We will retrieve the shortest path p between x and y, which is called the *augmenting path*. If the shortest path's weight is infinity (i.e $w_f(p) = \infty$)- then the maximal flow value of the graph G is strictly less than k, and the algorithm returns an error. 3) We augment $\delta = \min(k - |f|, \min\{c(e)|e \in E\}) > 0$ units of flow through p. That means if $(v, v') = e \in p$ is in the original graph (i.e $e \in E$) then we update $f(e) \leftarrow f(e) + \delta$, and if the reverse edge in G (i.e $(v', v) \in E$) then we update $f(v', v) \leftarrow f(v', v) - \delta$. 4) We create a new residual graph G_f, according to the updated flow f.

In [18], [17] the following theorem is proven:

Theorem 6.1: The flow f is a min-cost flow at the end of every iteration. Moreover, SSP returns a min-cost flow of required flow $|f| = k$.

Remark 6.2: An alternative algorithm for finding shortest path is *Dijkstra's algorithm*. Although Dijkstra's algorithm is faster than the Bellman-Ford algorithm, it requires the edge weights to be non-negative. Since the residual graph contains negative edges, we use Bellman-Ford algorithm. In [17] they present a method to optimize the SSP algorithm using the node potentials technique and to ensure that the edge weights will be non-negative. In further research we will optimize the algorithms presented in that article using this technique.

By running the SSP algorithm on the 7-layer graph G^7 (presented in Subsection V-C) we will get an optimal solution for the placement problem, according to Corollary 5.3. As presented in Subsection V-C, the time complexity of SSP on G^7 is $O(s^3 m^2 k^2)$. The Bipartite Graph algorithm presented in this section relies on SSP algorithm, and runs faster ($O(s^2 mk(m + k))$).

B. The Bipartite-like Graph

For presenting the Bipartite Graph algorithm, we will research the structure of shortest paths between two nodes in the 7-level residual graph G_f^7.

Let f be a flow that the SSP algorithm calculates in its j^{th} iteration. Let v_i represent a node in layer-i in G_f^7, for $1 \leq i \leq 7$. A directed path (v_2, v_3, v_4, v_5) in G_f^7 is called an *area-to-type path* since it is between an area node and a resource type node. Directed paths (v_1, v_2), (v_5, v_4, v_3, v_2),

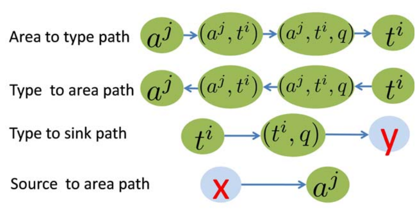


Fig. 4. Definitions of special paths in $G7_f$.

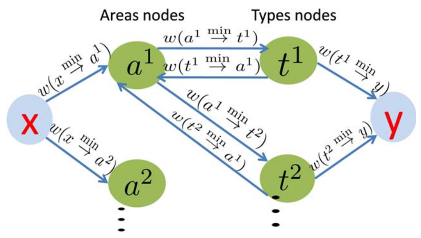

Areas nodes
Types nodes
$w(a^1 \overset{\min}{\rightarrow} t^1)$
$w(t^1 \overset{\min}{\rightarrow} a^1)$
$w(x \overset{\min}{\rightarrow} a^1)$
$w(t^1 \overset{\min}{\rightarrow} y)$
$w(a^1 \overset{\min}{\rightarrow} t^2)$
$w(t^2 \overset{\min}{\rightarrow} a^1)$
$w(x \overset{\min}{\rightarrow} a^2)$
$w(t^2 \overset{\min}{\rightarrow} y)$
x
a^1
a^2
t^1
t^2
y

Fig. 5. The bipartite graph G^B_f. We assume that $f^2 = f^2_2 = 0$, and $f^1, f^1_1, f^1_2 \neq 0$.

and (v_5, v_6, v_7) are called *source-to-area path*, *type-to-area path*, and *type-to-sink path* respectively, for the same reasons. Those paths, called *special paths* are presented in Fig. 4. Note that there are type-to-area paths in the residual graph G^7_f which are composed of reverse edges of G^7.

The number of area-to-type paths in G^7_f between an area node a and a resource type node t is $O(s)$. As we will see in the following analysis, G^7_f possesses a special property. The structure of G^7 implies that on any iteration of SSP only of these $O(s)$ paths needs to be considered. Focusing on this path will allow us shrinking the residual graph considerably to yield the mush smaller bipartite graph. To do so, we say that an area-to-type path p between area node a and resource type node t is *minimal path* if it is a path with minimal weight among all the area-to-type paths between a and t. We denote the path by $a \overset{\min}{\rightarrow} t$ and its weight by $w_f(a \overset{\min}{\rightarrow} t)$. We denote similar definitions for a minimal path between resource type node t and area node a, between a resource type t and the sink y, and between the source s and an area node a by $t \overset{\min}{\rightarrow} a$, $t \overset{\min}{\rightarrow} y$ and $x \overset{\min}{\rightarrow} a$, respectively. For the analysis done in this section, we may concatenate between two minimal paths; for example paths $a \overset{\min}{\rightarrow} t \overset{\min}{\rightarrow} y$ represent the concatenation of paths $a \overset{\min}{\rightarrow} t$ and $t \overset{\min}{\rightarrow} y$.

Remark 6.3: In case there are more than one minimal paths between two nodes v_1, v_2, one of them selected arbitrary to be we $v_1 \overset{\min}{\rightarrow} v_2$.

In the next lemma, we see the importance of minimal paths to characterize the shortest path in G^7_f, allowing us to eliminate non-minimal paths when computing a shortest path:

Lemma 6.4: Let f be a flow that the SSP algorithm calculates in its j^{th} iteration over the residual graph G^7_f. Then a shortest path between x and y, denoted by p_{opt}, has a decomposition formula as follows:

$$p_{opt} = x \overset{\min}{\rightarrow} a^{j_1} \overset{\min}{\rightarrow} t^{i_1} \overset{\min}{\rightarrow} a^{j_2} \ldots \overset{\min}{\rightarrow} a^{j_e} \overset{\min}{\rightarrow} t^{i_e} \overset{\min}{\rightarrow} y, \tag{7}$$

where x,y are the sink and source of G^7_f, and a^{j_l}, t^{i_l} are area and resource type nodes, for all $1 \leq l \leq e$.
The reader may verify the correctness of the lemma resulted from the structure of G^7 and G^7_f. The proof is in the technical report [6].

The lemma helps understanding the *bipartite-like graph*, denoted as G^B_f. The bipartite graph is a 4-layer graph represents the minimal paths in G^7_f. The graph is composed from the following layers: The first layer consists of the source node x, the second layer consists of area nodes a^j, the third layer consists of resource type nodes t^i, and the last layer consists of the sink node y. The edge weights are determine by the minimal paths weights in G^7_f. The graph, depicted in Fig. 5, resembles a bipartite graph, excluding the source and sink nodes.

To this end, using Lemma 6.4 we conclude that finding the shortest path in bipartite-like graph G^B_f is equivalent to finding a shortest path in G^7_f, as stated in the next corollary:

Corollary 6.5: Let f be a flow that SSP calculates in its j^{th} iteration over the residual graph G^7_f. Then $p_{opt} = x \overset{\min}{\rightarrow} a^{j_1} \overset{\min}{\rightarrow} t^{i_1} \overset{\min}{\rightarrow} a^{j_2} \ldots \overset{\min}{\rightarrow} a^{j_e} \overset{\min}{\rightarrow} t^{i_e} \overset{\min}{\rightarrow} y$ is the shortest path in G^7_f iff $\hat{p}_{opt} = (x, a^{j_1}, t^{i_1}, a^{j_2} \ldots, t^{i_e}, y)$ is the shortest path in G^B_f

In the next subsection, we will see how to calculate the Bipartite-like graph edges G^B_f (i.e the minimal path weight $w_f(v_1, v_2)$) in constant time ($O(1)$).

C. Calculating efficiently the Bipartite-like graph edges

In addition for shrinking the residual 7-layer graph G^7_f to a bipartite graph G^B_f, we will need to reduce the time complexity for constructing G^B_f. In the SSP algorithm we construct the residual graph G^7_f in $O(smk)$ time. We will show to construct the Bipartite-like graph G^B_f only in $O(mk)$ time. Lemmas proofs are provided in the technical report [6].

Let f be a flow that SSP calculates in its i^{th} iteration in G^7_f. Assume f^j_i represent the flow through (area, type) node (a^j, t^i), f_i the flow through resource type node t^i and f^j through area node a^j. We will prove that minimal paths in G^7_f (and therefore the edge weights in G^B_f) are non-trivially depended only on f^j_i, f_i and f^j ($O(mk)$ variables) and not on the flow in the other edges\vertices. Moreover, an edge weight $w_f(v_1 \overset{\min}{\rightarrow} v_2)$ is computed from those values in $O(1)$, as stated next:

Lemma 6.6: The minimal paths wights in G^7_f are:

1) $w_f(x \overset{\min}{\rightarrow} a^j) = 0$ if $f^j < s^j$ and otherwise ∞.
2) $w_f(a^j \overset{\min}{\rightarrow} t^i) = R_{loc}(1 - \Pr(D^j_i \geq f^j_i + 1))$.
3) $w_f(t^j \overset{\min}{\rightarrow} a^i) = -R_{loc}(1 - \Pr(D^j_i \geq f^j_i))$ if $f^j_i > 0$ and otherwise ∞.
4) $w_f(t^i \overset{\min}{\rightarrow} y) = R_{loc}(1 - \Pr(D_i \geq f_i))$.

Note that the weight of a minimal path $w_f(a^j \overset{\min}{\rightarrow} t^i)$ to that of $w_f(t^i \overset{\min}{\rightarrow} a^j)$.

Finally, since in every iteration of SSP we update the flow values, we can characterize the modification that SSP does to the flow values f_i^j, f_i and f^j, as stated follows:

Lemma 6.7: If the shortest path found in Step 2 of SSP is $p_{opt} = x \overset{\min}{\rightarrow} a^{j_1} \overset{\min}{\rightarrow} t^{i_1} \overset{\min}{\rightarrow} a^{j_2} \ldots \overset{\min}{\rightarrow} a^{j_e} \overset{\min}{\rightarrow} t^{i_e} \overset{\min}{\rightarrow} y$ then the flow values f_i^j, f_i and f^j are updated by SSP as follows: 1) $f_{i_l}^{j_l} \leftarrow f_{i_l}^{j_l} + 1$ for $1 \le l \le e$. 2) $f_{i_l}^{j_{l+1}} \leftarrow f_{i_l}^{j_{l-1}} - 1$ for $1 \le l \le e-1$. 3) $f_{i_e} \leftarrow f_{i_e} + 1$. 4) $f^{j_1} \leftarrow f^{j_1} + 1$.

From Lemmas 6.6, 6.7 and Corollary 6.5 we can construct the BG algorithm described in the next subsection.

D. The Bipartite Graph algorithm

We first initiate the zero flow $f_i^j = 0$, $f_i = 0$ and $f^j = 0$ for all areas $1 \le j \le k$ and resource types $1 \le i \le m$. In every iteration, the algorithm will do the following steps: 1) Finds a shortest $x - y$ path p_{opt} in G_f^B using Bellman-Ford. 2) Update f_i^j, f_i and f^j values according to the shortest path, p. This update uses the same updates SSP does on f_i^j, f_i and f^j values, described in Lemma 6.7. 3) Calculate the weight of the graph edges of G_f^B according to the new flow f. Since G_f^B satisfies the equivalent path property (see Corollary 6.5) then the weight of an edge in G_f^B is calculated as the weight of the shortest path in G_f^7 (see Lemma 6.6).

The resulted flow, f_i^j, is the flow value of node (a^j, t^i) which SSP finds. By Theorem 6.1, SSP retrieves the optimal min-cost f_{opt} in G^7. Also by Lemma 5.2, the placement $L_i^j = f_{opt}^{in}(i,j)$ solves the placement problem. Thus, we get that the placement $L_i^j = f_i^j$ is optimal.

The number of iterations the BG algorithm preforms is $O(s)$. Step 1 in every iteration takes $O(|V||E|) = O((m+k)mk)$, while Steps 2 and 3 take at most $O(V+E)$. Thus, the complexity of the algorithm is $O(smk(m+k))$, which is faster than applying SSP to G^7, whose complexity is $O(s^3m^2k^2)$.

VII. Numerical Examples

In this section, we evaluate the Bipartite Graph algorithm performance in order to study the following subjects: 1) The performance of the optimal placement under various scenarios. 2) The performance of a common used placement called *Proportional Mean* placement compared to the optimal placement.

We consider a demand that follows a Zipf distribution(which is consistent with prior analytical works [3], [14] and VoD empirical results [20], [1]). This means that there exists a real number $e > 0$ such that the probability for a single request to demand a type-i resource is $p_i = \frac{1}{i^e \cdot H}$ for all resource types $1 \le i \le m$, where $H = \sum_{j=1}^{m} \frac{1}{j^e}$. We will assume that the demand in area j is proportional to the area's storage, i.e the probability for a request to originate from area j is $q^j = \frac{s^j}{s}$. To this end, we choose the demand for type-i resources originating from area j, D_i^j, to be a Poisson distribution with a rate of $p_i \cdot q^j \cdot \lambda$, where λ is a constant number representing the expected number of requests.

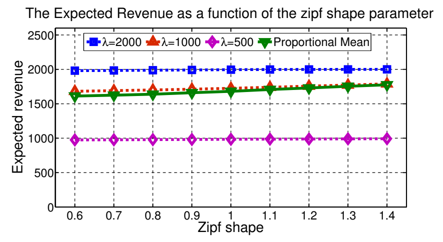


Fig. 6. The optimal performance under deficit, balanced and surplus modes.

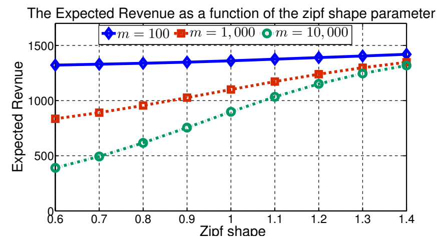


Fig. 7. A performance comparison between optimal placement revenues when the number of resource types are $m = 100$, $m = 1,000$ and $m = 10,000$.

In Fig. 6 we choose the number of resource types to be $m = 100$, with revenue parameters of $R_{sat} = R_{loc} = 1$. Since it is unclear how future demands behave (prior references mentioned above used a Zipf parameter of values 0.56 - 1.5, depending on the study) we consider a wide range of Zipf distributions and vary the Zipf parameter from 0.6 to 1.4. The number of areas is $k = 3$, containing storage of $s^1 = 500$, $s^2 = 300$ and $s^3 = 200$, and thus the storage of the system is $s = 1000$. We plot the expected revenue of the optimal placement under surplus scenario ($\lambda = 500$), under balanced scenario ($\lambda = 1000$), and deficit scenario ($\lambda = 2000$). We compute the expected revenue using Eq. 5. We also plot the performance of Proportional Mean placement in the balanced scenario.

An important result from Fig. 6 is the optimal placement in the different configurations. Its revenue in surplus mode is close to 1000, since almost even request is granted locally contributing $R_{loc} + R_{sat} = 2$. In the deficit scenario, the revenue is almost 2000, since about 1000 requests are granted and all locally . In the balanced scenario, some of the requests are granted locally, and others are granted remotely.

In Fig. 7 we use the balanced settings, but increase the variety of resource types to $m = 1,000$ and $m = 10,000$. We can see three major results: 1) As the number of resource types increases, the optimal placement revenue decreases. This is implied from the increasing number of esoteric resources (resources with rank larger than the storage value s) which cannot be satisfied. 2) As the Zipf parameter increases, the optimal placement's revenue increases. This can be explained by the fact since the number of esoteric resources decreases as the Zipf parameter increases.

The Proportional Mean placement used in several in articles ([3], [14], [10]), places its replicas $\{L_i^j\}$ proportional to the mean value of the demand distribution, D_i^j. We compare

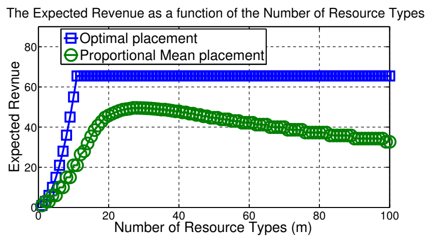


Fig. 8. Performance comparison of the proportional mean placement and the optimal placement in back-up services applications .

the expected revenue of Proportional Mean placement to the revenue of the optimal placement.

In Fig. 6 we see that in the balanced mode Proportional Mean placement performance is close to the optimal placement. In contrast, in scenario taken in Fig. 8 we see Proportional Mean performance decreases when the number of resource types goes to infinity, while the optimal placement revenue stabilizes. In that scenario we assume a single area system ($k = 1$) and the storage of area 1 is $s^1 = 500$. The revenue constants are $R_{sat} = 0$, $R_{loc} = 1$, and therefore the expected revenue represents the number of requests granted in area 1. The demand for type-i resource in area 1 is 0 with probability of $1 - 1/i$ ($\Pr(D_i^1 = 0) = 1 - 1/i$) and i^2 with probability $1/i$ ($\Pr(D_i^1 = i^2) = 1/i$).

The demand distributions $\{D_i\}$ reflects a scenario of an application providing back-up services to m companies. If a failure occurs in one of the companies, the application will provide back up to the users in the company. We can assume that as the company size gets larger, the company is more maintained and invulnerable to fails, reflected in its failure probability proportional to its size, namely $1/i$.

The figure demonstrates that when the variance of some of the demands is high proportional mean does not perform well.

VIII. Concluding Remarks

In this paper we formulated the problems of the resource placement and assignment in a network environment. We presented an exact solution leading to finding the optimal assignment as well as the optimal placement. The optimality was derived under the assumption of asymmetric stochastic demands with arbitrary distributions. Algorithms for solving the problems were proposed, and shown to be very efficient.

The analysis provided in this work can be further extended to deal with k-level hierarchies. This is done in an ongoing work. Also in ongoing work we study enhancements to the Bipartite Graph Algorithm that further reduce its complexity. We also study a variation of the problem where the number of resources in any region are not limited.

Acknowledgment

This research was supported in part by the Ministry of Science and Technology, Israel.

References

[1] V. Valancius, N. Laoutaris, L. Massoulie, C. Diot, and P. Rodriguez, “Greening the Internet with Nano Data Centers,” in *ACM CoNext*, Rome, Italy, Dec 2009.

[2] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, “I Tube, You Tube, Everybody Tubes: Analyzing the World’s Largest User Generated Content Video System,” in *ACM Internet Measurement Conference*, New York, NY, USA, October 2007.

[3] S. Tewari and L. Kleinrock, “Proportional replication in peer-to-peer networks,” in *IEEE INFOCOM*, Barcelona, Spain, April 2006.

[4] Y. Rochman, H. Levy, and E. Brosh, “Max percentile replication for optimal performance in multi-regional p2p vod systems,” in *Proceeding of the 9th International Conference on Quantitative Evaluation of SysTems (QEST) 2012*, London, UK, September 2012, to appear. [Online]. Available: http://www.cs.tau.ac.il/∼yuvalroc/publications/qest2012.pdf

[5] ——, “Efficient replication in multi-regional peer supported vod systems,” in *Workshop on MAthematical performance Modeling and Analysis (MAMA) 2012*, London, UK, June 2012.

[6] ——, “Resource placement and assignment in distributed network topologies- technical report,” 2013. [Online]. Available: http://www.cs.tau.ac.il/∼yuvalroc/publications/infocom2013tech.pdf

[7] Z. Drezner and H. W. Hamacher, *Facility Location: Applications and Theory*. Springer, 2002.

[8] F. L. Presti, C. Petrioli, and C. Vicari, “Distributed dynamic replica placement and request redirection in content delivery networks,” in *MASCOTS*, 2007, pp. 366–373.

[9] P. Krishnan, D. Raz, and Y. Shavitt, “The cache location problem,” *IEEE/ACM Transaction on Networking*, vol. 8, no. 5, pp. 568–582, October 2000.

[10] S. Tewari and L. Kleinrock, “On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks,” in *IFIP/TC6 Networking*, Waterloo, Ontario, Canada, May 2005.

[11] J. Kangasharju, K. W. Ross, and D. A. Turner, “Optimizing file availability in peer-to-peer content distribution,” in *IEEE INFOCOM*, Anchorage, Alaska , USA, May 2007.

[12] E. Cohen and S. Shenker, “Replication strategies in unstructured peer-to-peer networks,” in *ACM SIGCOMM*, 2002, pp. 177–190.

[13] Y. P. Zhou, T. Z. J. Fu, and D. M. Chiu, “Statistical modeling and analysis of p2p replication to support vod service,” in *IEEE INFOCOM*, Orlando, FL , USA, July 2011.

[14] B. Tan and L. Massoulié, “Optimal content placement for peer-to-peer video-on-demand systems,” in *IEEE INFOCOM*, Orlando, FL , USA, July 2011.

[15] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, “Optimal content placement for a large-scale vod system,” in *ACM CoNEXT*, Philadelphia, USA, Dec 2010.

[16] M. Leconte, M. Lelarge, and L. Massoulié, “Bipartite graph structures for efficient balancing of heterogeneous loads,” in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS, 2012, pp. 41–52.

[17] J. Edmonds and R. M. Karp, “Theoretical improvements in algorithmic efficiency for network flow problems,” *ACM 19*, pp. 248–264, 1972.

[18] R. G. Busacker and P. J. Gowen, “A procedure for determining a family of minimal cost network flow patterns,” Operational Research Office, Johns Hopkins University, Baltimore, MD, ORO Technical Report 15, September 1961.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[20] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, “Youtube traffic characterization: a view from the edge,” in *ACM Internet Measurement Conference*, ser. IMC ’07. New York, NY, USA: ACM, 2007, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298310