Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 35 (2008) 3860–3884
computers & operations research
www.elsevier.com/locate/cor

Exact algorithms for the joint object placement and request routing problem in content distribution networks

Tolga Bektaş[a,b,*], Jean-François Cordeau[a], Erhan Erkut[c], Gilbert Laporte[a]

[a] *Center for Research on Transportation, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7*
[b] *School of Business, University of Alberta, Edmonton, Alberta, Canada T6G 2R6*
[c] *Faculty of Business Administration, Bilkent University, 06800 Bilkent, Ankara, Turkey*

Available online 15 Februray 2007

Abstract

This paper describes two exact algorithms for the joint problem of object placement and request routing in a *content distribution network* (CDN). A CDN is a technology used to efficiently distribute electronic content throughout an existing Internet Protocol network. The problem consists of replicating content on the proxy servers and routing the requests for the content to a suitable proxy server in a CDN such that the total cost of distribution is minimized. An upper bound on end-to-end object transfer time is also taken into account. The problem is formulated as a nonlinear integer programming formulation which is linearized in three different ways. Two algorithms, one based on Benders decomposition and the other based on Lagrangean relaxation and decomposition, are described for the solution of the problem. Computational experiments are conducted by comparing the proposed linearizations and the two algorithms on randomly generated Internet topologies.

Keywords: OR in telecommunications; Content distribution network; Linearization; Benders decomposition; Lagrangean relaxation and decomposition

1. Introduction

Recent advances in information and computer technology have considerably eased the access to electronic information. However, the amount of information readily available in such networks and the scarcity of resources pose new challenges to the efficient distribution of electronic content. This is especially true for the Internet, where there is a phenomenal growth of demand for any kind of electronic information, thus placing a high burden on the underlying infrastructure. As the size of the content delivered and the number of users have increased tremendously in recent years, clients have started to experience unacceptable delays. Other consequences of this high usage rate are increased loads on the servers, and network congestion. It has recently been observed by Saroiu et al. [1] that the average size per request of the delivered content has changed from about 2 KB to 4 MB in about three years, an increase of three orders of magnitude.

* Corresponding author. Center for Research on Transportation, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7 Canada. Tel.: +1 514 343 6111; fax: +1 514 343 7121.
E-mail addresses: tolga@crt.umontreal.ca (T. Bektaş), cordeau@crt.umontreal.ca (J.-F. Cordeau), erkut@bilkent.edu.tr (E. Erkut), gilbert@crt.umontreal.ca (G. Laporte).

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.02.005

The delays experienced by end users have important economic consequences, especially in electronic marketing. A widely observed standard is that a typical client will abandon a Web site failing to download in less than 8 s. According to a report by Zona Research conducted in 1999, "the amount of time taken for Web pages to load is one of the most critical factors in determining the success of a site and the satisfaction of its users" [2]. In the same report, the potential losses in 1999 due to unacceptably slow response times is estimated to be about $4.35 billion. In a similar research conducted two years later, Zona Research [3] reported the corresponding figure for 2001 to be over $25 billion.

Distributing electronic content effectively to public has become a major problem. Efforts to overcome delays and to alleviate the Internet traffic has given way to the new technology called *Content distribution* (or *delivery*) *networks* (CDNs). The goal of CDNs is to replicate the content from the origin server(s) to geographically distributed surrogate sites, referred to as *proxy servers*, from which the clients receive the requested content on behalf of the origin server. CDNs therefore aim at moving the content as close as possible to the clients. A CDN can significantly improve the performance of a network and reduce the total cost associated with distributing content, since the clients are no longer served by the origin server, but from a proxy server located nearby.

The performance of a Web site is measured in terms of throughput, latency, execution time, and transaction time. *Throughput* represents the number of service requests served in a given time interval of interest. The term *Latency* denotes the time between sending a request and receiving the response. *Transaction time* corresponds to the amount of time elapsed whilst the service is completing a transaction, whereas *execution time* refers to the amount of time required by a service to process a set of requests. It is desirable for a Web site to have high throughput rate and low latency, with low execution and fast transaction times.

A *content publisher*, who issues the content for the public, may choose to resort to a commercial CDN to have its content efficiently distributed to its users. It is often the case that there is some kind of a service level agreement (SLA) between the publisher and the *content distributor*, usually the hosting service, that includes some kind of a quality of service (QoS) requirement. The QoS requirement is in general associated with the quality, both functional as well as nonfunctional, aspect of a content distribution service. The QoS may include certain guarantees on performance, reliability, integrity, accessibility, availability, interoperability, and security of the service. Therefore, in performing the distribution operations, the CDN must also take into account the QoS. An upper bound on the delay associated with serving an object to a client is a good example of a QoS requirement.

There are already many companies offering hosting services for content distribution. Vakali and Pallis [4] point out that about 2500 companies were reported to be using CDNs as of December 2003. As an example, a popular hosting service, Akamai, has already deployed 15,000 servers over 1100 networks across 65 countries and hosts popular customers such as Apple, IBM, Reuters, Yahoo and Warner Music Group [5].

Ideally, one would like to have all the content present in the origin server to be replicated to all proxy servers installed throughout the CDN. This is referred to as *full replication* and is often used when the content consists of small-sized objects and the capacities of the proxy servers are sufficiently large. However, in most cases, the scarcity of capacity resources of proxy servers and the high cost of replication renders such an approach impractical. Moreover, this approach may also result in an overuse of the network resources, since one may unnecessarily replicate a part of the content that is rarely requested. This gives way to what is called *partial replication*, which consists of locating only specific subsets of the content on the proxy servers. Therefore, a CDN must decide on how to replicate the subsets of the content in an intelligent manner, so as to efficiently use the network resources. This problem is referred to as *object placement* (or *object replication*). Another challenging task for a CDN is, given a request from a client, to identify the best proxy server that should respond to this request. This is referred to as *request routing*. Both of these problems are interrelated, and thus should be considered together for a CDN to operate in an efficient manner.

The remainder of the paper is organized as follows. Section 2 provides a review of related work. In Section 3, we give a formal description of the design problem and the proposed nonlinear integer programming formulation. This section also includes a discussion on how the formulation is capable of handling additional constraints, such as those related to QoS specifications. In Section 4, we offer three different linearizations for the nonlinear formulation. Based on two of the linearizations, we describe two solution approaches in Sections 5 and 6, respectively. Results of computational experiments in comparing the proposed solution approaches are given in Section 7, followed by conclusions in Section 8.

2. Previous research

CDNs are a relatively new avenue for research and most papers published on this subject are due to the computer science community. In this section, we briefly review the studies that are most relevant to our problem, with an emphasis on operations research approaches.

One of the first topics studied in CDN research is the proxy server location problem, which consists of optimally locating a number of proxy servers in a network and assigning each client to an established proxy server so as to minimize the total cost of location and assignment. Li et al. [6] were among the first to study this problem on a tree-like Internet topology. They proposed a dynamic programming algorithm for the problem. Later, Qiu et al. [7] argued that the Internet hardly has a tree-like structure and offered several placement algorithms. These authors also stated that one can benefit from the well-known facility location or p-median problem for the solution of the problem. Woeginger [8] improved the dynamic programming algorithm of Li et al. [6] using an observation that the underlying topology has the Monge property.[1] Recently, Tang and Xu [9] studied the replica placement problem on a tree topology in which QoS requirements are considered.

The object placement problem is also well studied with respect to CDNs, if not extensively. Leff et al. [10] present the first hierarchical model for object placement is developed, where the authors present a polynomial time exact solution method (which was later generalized in [11]), along with some distributed heuristic approaches. Cidon et al. [12] propose a distributed algorithm to allocate electronic content over a network with a tree structure, in order to minimize the total storage and communication cost. Kangasharju et al. [13] and Yang and Fei [14] take into account the limited storage capacity in solving the object placement problem. The former authors formulate the problem as an integer program to minimize the average travel time of the objects, whereas the latter emphasize the distribution of objects in multimedia applications.

The object placement problem is in some ways similar to the *database location problem* in computer communications networks. This problem in general consists of placing copies of a database throughout a computer network considering the tradeoff between the cost of accessing the various copies of the database in the network and the cost of storing and updating the additional copies. Fisher and Hochbaum [15] presented a mixed integer model for this problem. To solve a variant of this problem, Pirkul [16] proposed a Lagrangean based solution algorithm along with a heuristic procedure. The reader may also refer to Gavish [17], Gavish and Suh [18], Chari [19] and Hakimi and Schmeichel [20] for additional studies on the database location problem.

There exist a number of papers in which several problems within a CDN are studied simultaneously. For example, Ryoo and Panwar [21] study the problem of distributing multimedia files in networks involving the determination of the communication link capacities, sizing the multimedia servers and distributing different types of content to each server. In another study, Xu et al. [22] investigate the problem of determining the optimal number and location of proxies along with the placement of replicas of a single object on the installed proxies, given a maximum number of potential proxies and a tree-like topology. Xuanping et al. [23] discuss the joint problem of proxy server placement and object placement in a CDN, subject to a budget constraint. The authors assume that each client is assigned to its closest proxy. In a similar context, Laoutaris et al. [24] consider the joint problem of optimal location of the objects together with the capacity dimensioning of the proxies. Another study by Laoutaris et al. [25] addresses the storage capacity allocation problem for CDNs, which takes into account the optimal location of the proxies, the capacity of each proxy and the objects that should be placed in each proxy. The assignment of clients is not considered as a decision problem, since they assume a given hierarchical topology where the assignment of clients is predetermined.

In a relatively recent study, Almeida et al. [26] considered the problem of jointly routing requests and placing proxy servers in streaming CDNs. These authors presented an optimization model for the problem and proposed a number of heuristics for its solution in an attempt to minimize the total server and network delivery cost. More recently, Nguyen et al. [27] considered the problem of provisioning the so-called overlay distribution networks, which includes proxy server placement, request routing and object replication. These authors proposed an integer linear programming formulation along with a heuristic solution algorithm based on Lagrangean relaxation. Their study also considers improving scalability of the model with object clustering.

---

[1] A matrix $C = (c_{ij})$ is said to have a Monge structure if $c_{ij} + c_{rs} \leqslant c_{is} + c_{rj}$ for all $1 \leqslant i < r \leqslant m$ and $1 \leqslant j < s \leqslant n$.

A recent paper by Bektaş et al. [28] considers a design problem arising in CDNs, and simultaneously solves three problems: (i) the number and the location of the proxy servers to be used in the CDN among a given set of potential sites (*proxy server placement*), (ii) the objects to be located in each proxy server (*object placement*), and (iii) the assignment of each client to a proxy server (*request routing*). The design problem considered in Bektaş et al. [28] is solved at the strategic level. In this paper, we look at this problem from an operational point of view in that we only consider the object placement and request routing subproblems. The reason for this is that many commercial content providers have a number of proxy servers already established. The CDN then has to decide how to replicate the objects and how to assign the client requests to appropriate servers. This is a difficult task, bearing in mind the dependency of the two problems to one another and the need to solve them jointly. With respect to Bektaş et al. [28], we look at the problem in a slightly more detailed perspective and at a more in-depth level. Our design takes into account an operational level constraint, namely a QoS requirement that must be guaranteed by the CDN. In particular, we include in our design proposal a constraint that guarantees an end-to-end delay in object transfer. To solve the problem, we propose two algorithms based on Benders decomposition and Lagrangean relaxation. With respect to the Benders decomposition implementation of Bektaş et al. [28], the algorithms proposed in this paper constitute nontrivial implementations which better exploit the structural property of the problem.

3. Problem definition

We begin by formally introducing some terminology that will be used throughout this paper. The term *content* refers to any kind of publicly available information on the World Wide Web, such as Web pages, multimedia files and text documents. We will use the term *object* to refer to a specific item of the content. The term *content provider* refers to a unit, which holds the content for the access of others. We will call *clients* the network users who issue requests for content.

We assume a given complete network $G = (V, E)$, where V is the set of nodes and $E = (\{i, j\} : i, j \in V)$ is the set of links. The node set V is further partitioned into three nonempty subsets I, J and $S = \{0\}$, where I is the set of clients, J is the set of nodes where proxy servers are installed, and S is a singleton containing the origin server. The set of clients may be composed of Internet service providers (ISPs), corporate firms, universities, etc. We assume without loss of generality that no client can directly access the origin server (e.g., for security reasons). With each link $(i, j) \in E$ is associated a nonnegative unit transfer cost denoted by c_{ij}. The cost may be an indicator of, say, unit bandwidth cost, number of hops, etc. We denote by d_{ij} the delay representing the amount of time required to retrieve data between nodes i and j. While appropriate queueing models may be used to measure the delay in the network, we assume here that the average delay for each link is known, as in Nguyen et al. [27] and Wauters et al. [29], and can be calculated as the sum of propagation and transmission delays [30]. We also note that there are other factors contributing to this calculation, such as delays caused by routers or switches located between the two nodes, and the size of the object transferred. Since we assume that an existing network is in place, we ignore the option of network expansion as opposed to network design problems which usually consider addition of new links to the existing network.

Each client is assumed to be served by exactly (or at least) one proxy server. In any case, we assume that the client retrieves the requested object from a single server. This consideration is based on a well-stated result given by Kangasharju et al. [31], who have demonstrated through simulation that retrieving an object as a whole from a single proxy results in a better performance than retrieving different parts of the object from different proxies. We assume that the capacity of the potential server at site j is s_j. If large objects are to be distributed, then the capacity can be defined in terms of physical storage, which will be the bottleneck in such a situation. If not, it can be defined as the total bandwidth a server may support. We define K as the set of objects located in the origin server and assume that the size of each object $k \in K$ is b_k. Also we consider that the probability of client $i \in I$ requesting object $k \in K$ over a given time interval is denoted by λ_{ik}. The time span of the probability depends on the frequency of the problem being solved. A summary of the notation used in defining the problem is given in Table 1.

The problem considered in this study is to minimize the total cost of distributing the content under the following two constraint classes:

1. the assignment of each client to a single proxy server;
2. the objects to be located in each proxy server.

Table 1
Summary of the notation used for the CDN model

Sets	
I	set of clients
J	set of proxy servers
K	set of objects
S	set containing the origin server
Parameters	
s_j	capacity of the proxy server on node $j \in V$
b_k	size of object $k \in K$
λ_{ik}	probability of client $i \in I$ requesting object $k \in K$
c_{ij}	unit cost of transferring an object over a logical link $\{i, j\} \in E$
d_{ij}	unit delay caused by transferring an object over logical link $\{i, j\} \in E$

It is important to state that we consider this problem from the perspective of the CDN provider. This can be explained by the fact that it is the CDN which decides how the object placement should be performed and how the requests should be served. The CDN would like to minimize its total cost. Several other schemes may be employed in a CDN, such as dynamically selecting the proxy that offers the lowest response time. We ignore such a scheme as we do not consider real-time decisions and our design consists of making all decisions *a priori*. Although this may seem like a very static approach for such an application, replication for content distribution based on steady state demand rates have been shown to have significant benefits [32].

In such a setting, two strategies are possible in distributing content. The first is to assign each client to a single proxy server. Given a client request for an object, the request is served from the associated proxy if the object is already stored there. If not, then the client is able to access the object from the origin server via the path from the corresponding proxy server to the origin server, but at the expense of an additional transfer cost (see [14,30,33,34] for a similar architecture and cost structure). This may be regarded as *noncooperative caching*, in which the proxy servers do not cooperate in serving content. In other words, once a content is not found in a proxy server, the other proxies are not contacted, but rather the request is forwarded directly to the origin server. This is a viable strategy when the overheads associated with messaging and processing for cooperation is high [30]. The second strategy relaxes this assumption and allows a client to fetch a requested object from any of the proxies that hold it. In this setting, we now assume that a client can be connected to more than one proxy server to fetch the requested content, i.e., the client may retrieve a requested object from any one of the proxies that hold it. Objects can only be retrieved from the origin server if they are not stored at any proxy server. This strategy can be seen as a variant of *cooperative caching*, where the proxies of the CDN cooperate in serving a client with the requested object.

One may prefer one strategy to another in distributing electronic content and this would depend on the type of content distributed and on the structure of the CDN itself. In this study, we are concerned with the first setting where each client is connected to a single proxy server. Our motivation stems from the fact that if the number of servers is high, it may be very costly to search for the proxy servers that are able to satisfy the client's request. We should also point out that the second strategy may be modeled as a variant of the *Capacitated Facility Location Problem* (CFLP), for which there are a number of formulations and solution methods available in the existing literature (see, e.g., [35]).

3.1. The proposed formulation

To formulate the problem under consideration, we define the following binary decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if client } i \in I \text{ is assigned to proxy server } j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

$$z_{jk} = \begin{cases} 1 & \text{if proxy server } j \in J \text{ holds object } k \in K, \\ 0 & \text{otherwise.} \end{cases}$$

The formulation is as follows:

$$(\mathscr{P}) \quad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}c_{ij}x_{ij}z_{jk} + b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij}(1-z_{jk})) \tag{3.1}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I, \tag{3.2}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J, \tag{3.3}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I, \quad j \in J, \tag{3.4}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J, \quad k \in K. \tag{3.5}$$

The first part of the objective function (3.1) denotes the total cost of transferring the content for the case where client i receives content k that is located in proxy j (reflected by the cost $b_k\lambda_{ik}c_{ij}$ summed over all the proxies, clients and objects). In the case when the requested object is not located in the proxy server, an additional cost is incurred to further request the object from the origin server. This is reflected in the second part of the summation by c_{j0}. Constraints (3.2) imply that a client can only be assigned to a single proxy server. Constraints (3.3) state that the total capacity required by the objects held in each proxy server is constrained by the available capacity. We additionally note that the integrality requirements (3.4) can be dropped from this formulation and replaced by $x_{ij} \geqslant 0$.

Although the problem may directly be formulated as an integer linear program, we prefer to first formulate the problem with a quadratic objective since it provides a unified way of deriving the three linearizations presented below. This formulation is capable of accommodating other constraints related to bandwidth limitations or QoS constraints. This is explored further in the following section.

3.2. Inclusion of QoS-type constraints

The above formulation can be extended to take into account several other constraints that may arise in distribution of content, such as bandwidth limitations, QoS constraints, or any other constraint that takes into account specific details of a particular application. Here, we consider an important aspect that is considered by many commercial CDNs, namely a QoS constraint that imposes a delay limit on end-to-end object transfer delays. Denote by $t(b_k, d_{ij})$ the amount of delay caused by transferring object k over link (i, j) (We will explain in detail in Section 7 how this can be calculated.) Now, consider a client i connected to a proxy j and making a request for object k. The associated delay is $t(b_k, d_{ij})$ if the requested object is located in the proxy server j. If not, then an additional delay of d_{j0} arises as a result of retrieving the object from the origin server (we denote this by $t'(b_k, d_{ij})$). Consequently, given a client and an object pair, the delay experienced by the client can be calculated as $t(b_k, d_{ij})x_{ij}z_{jk} + t'(b_k, d_{ij})x_{ij}(1 - z_{jk})$. This expression is generally constrained by an upper bound, say Δ_d that can be defined separately for each client, for each object, or both, depending on the agreed SLA between the content publisher and the commercial CDN. The following is a QoS constraint on the delay requirements that may be introduced in the formulation presented above:

$$\sum_{j\in J} t(b_k, d_{ij})x_{ij}z_{jk} + \sum_{j\in J} t'(b_k, d_{ij})x_{ij}(1 - z_{jk}) \leqslant \Delta_d. \tag{3.6}$$

The proposed model can directly accommodate (3.6) and similar constraints. Indeed, one can *a priori* identify the edges violating the QoS constraints within a preprocessing scheme and remove these from the formulation before solving the problem as follows. Define, for each pair (i, j) the sets $\mathscr{R}_{ij} = \{k \in K | t(b_k, d_{ij}) > \Delta_d\}$, $\mathscr{Q}_{ij} = \{k \in K | t(b_k, d_{ij}) \leqslant \Delta_d$ and $t'(b_k, d_{ij}) > \Delta_d\}$, and $\mathscr{G} = \{(i, j) | i \in I, j \in J$ and $\mathscr{R}_{ij} \neq \emptyset\}$. Observe that for every pair $(i, j) \in \mathscr{G}$, one can remove x_{ij} from the formulation since this implies that there exists at least one $k^* \in \mathscr{R}_{ij}$ for which client i cannot be served by proxy j without exceeding the delay limit. Moreover, for each

$i \in I, j \in J, k \in \mathcal{Q}_{ij}$, constraint (3.6) can equivalently be written as $x_{ij} \leqslant z_{jk}$. We will include these constraints in formulation $\mathcal{P}$. A similar constraint applies to situations where the bandwidth resource of each link is limited and imposes a capacity restriction on the amount of content that may be transmitted through that link.

It can be seen that the proposed formulation is nonlinear in the objective function, which makes it hard to solve. We establish the complexity of the problem in the following proposition.

Proposition 1. *Problem $\mathcal{P}$ is $\mathcal{NP}$-hard.*

Proof. We prove the proposition by restriction. Consider the special case of the problem with $J = \{j\}$ and $t'(b_k, d_{ik}) \leqslant \Delta_{ik}$, for all $i \in I$ and $k \in K$. Since there is only one proxy server, $x_{ij} = 1$ for all $i \in I$. In addition, the QoS constraints are easily seen to be redundant since the delay for every client–object pair is less than the upper bound. Hence we can remove constraints (3.2) and (3.6) from $\mathcal{P}$. We can also simplify the objective function as follows:

$$\sum_{i\in I}\sum_{k\in K}(b_k\lambda_{ik}c_{ij}z_{jk} + b_k\lambda_{ik}(c_{ij} + c_{j0})(1 - z_{jk}))$$

which further reduces to

$$\sum_{i\in I}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij} + c_{j0}) - b_k\lambda_{ik}c_{j0}z_{jk}). \tag{3.7}$$

Dropping the constant term $(b_k\lambda_{ik}(c_{ij} + c_{j0}))$ from (3.7), we can rewrite formulation $\mathcal{P}$ as follows (we also drop the index j for notational convenience):

$$(\mathcal{P}_r) \quad \text{Maximize} \quad \sum_{k\in K}\widehat{c}_k z_k \tag{3.8}$$

subject to

$$\sum_{k\in K} b_k z_k \leqslant s, \tag{3.9}$$

$$z_k \in \{0, 1\} \quad \forall k \in K, \tag{3.10}$$

where $\widehat{c}_k = \sum_{i\in I} b_k\lambda_{ik}c_{j0}$. Note that $\mathcal{P}_r$ is the classical knapsack problem, which is known to be $\mathcal{NP}$-hard [36]. □

In the following section, we propose several linearization procedures.

4. Linearization procedures

In this section, we make use of three linearization procedures to transform the proposed nonlinear programming formulation into an equivalent integer linear programming formulation. Each linearization is described under the corresponding heading.

4.1. Linearization I

The proposed formulation is nonlinear due to the quadratic term that appears in the objective function as a result of the multiplication of x_{ij} and z_{jk} variables. A well known and standard linearization of the product of two binary terms can be performed by introducing a new binary variable and three sets of constraints (see [37,38] for details). However, due to the special structure of the objective function (3.1), we propose a linearization that only makes use of one continuous variable and two sets of constraints. This linearization is also used in a more general setting by Bektaş et al. [28]. The linearization is given in the following proposition.

Proposition 2. *Using the transformation* $\varphi_{ijk} = z_{jk}x_{ij}$, *the following constraints linearize formulation* $\mathscr{P}$:

$$\varphi_{ijk} - x_{ij} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in K, \tag{4.1}$$
$$\varphi_{ijk} - z_{jk} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in K, \tag{4.2}$$

where $0 \leqslant \varphi_{ijk} \leqslant 1$.

Proof. First, rewrite the objective function as $\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij} - b_k\lambda_{ik}c_{j0}\varphi_{ijk})$, where $\varphi_{ijk} = z_{jk}x_{ij}$. The proof relies on the observation that the coefficient of φ_{ijk} in the objective function is $-b_k\lambda_{ik}c_{j0}$, which is always negative. By definition, φ_{ijk} should be 1 if and only if $z_{jk} = 1$ and $x_{ij} = 1$, and 0 otherwise. Now assume that $z_{jk} = 1$ and $x_{ij} = 1$ for a specific (i, j, k) triplet. Then, according to constraints (4.1) and (4.2), φ_{ijk} is only constrained by the upper bound 1, and the minimizing objective function implies $\varphi_{ijk} = 1$. In all other cases (i.e., $x_{ij} = 1$, $z_{jk} = 0$; or $x_{ij} = 0$, $z_{jk} = 1$; $x_{ij} = 0$, $z_{jk} = 0$) constraints (4.1) and (4.2) jointly imply $\varphi_{ijk} = 0$. □

Note that the linearizing variable φ_{ijk} is actually an indicator of whether client i is connected to the proxy server j *and* the proxy server holds the requested object k or not. Under the proposed linearization, we can now construct the integer linear programming formulation of the problem as follows:

$\mathscr{L}_1(\mathscr{P})$ Minimize
$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij} - b_k\lambda_{ik}c_{j0}\varphi_{ijk}) \tag{4.3}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I, \tag{4.4}$$
$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J, \tag{4.5}$$
$$\varphi_{ijk} - x_{ij} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in K, \tag{4.6}$$
$$\varphi_{ijk} - z_{jk} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in K, \tag{4.7}$$
$$x_{ij} - z_{jk} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in \mathscr{Q}_{ij}, \tag{4.8}$$
$$x_{ij} \geqslant 0 \quad \forall i \in I,\ j \in J, \tag{4.9}$$
$$z_{jk} \in \{0, 1\} \quad \forall j \in J,\ k \in K, \tag{4.10}$$
$$\varphi_{ijk} \in [0, 1] \quad \forall i \in I,\ j \in J,\ k \in K. \tag{4.11}$$

4.2. Linearization II

The second linearization is based on that proposed by Glover [39] and uses the following observation. The objective function (3.1) can be rewritten as

$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij} - \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k\lambda_{ik}c_{j0}x_{ij}z_{jk}. \tag{4.12}$$

We now introduce a new continuous variable $y_{ij} = x_{ij}\sum_{k\in K} b_k\lambda_{ik}c_{j0}z_{jk}$ for each pair (i, j) in the second summation of the objective function (4.12). Variable y_{ij} has the property that, for a specific pair (i, j), it takes the value 0 when $x_{ij} = 0$, whereas it is equal to the term $\sum_{k\in K} b_k\lambda_{ik}c_{j0}z_{jk}$ when $x_{ij} = 1$. This can be formulated by the following constraints:

$$y_{ij} \leqslant M x_{ij} \quad \forall i \in I,\ j \in J,$$
$$y_{ij} \leqslant \sum_{k\in K} b_k\lambda_{ik}c_{j0}z_{jk} \quad \forall i \in I,\ j \in J,$$

where M is a sufficiently large constant (which may be chosen as $M = \sum_{k\in K} b_k \lambda_{ik} c_{j0}$). Using these constraints, we provide below the second linearization of problem $\mathscr{P}$ (henceforth referred to as $\mathscr{L}_2(\mathscr{P})$):

$$\mathscr{L}_2(\mathscr{P}) \quad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k \lambda_{ik}(c_{ij}+c_{j0})x_{ij} - \sum_{i\in I}\sum_{j\in J} y_{ij} \tag{4.13}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I, \tag{4.14}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J, \tag{4.15}$$

$$y_{ij} - M x_{ij} \leqslant 0 \quad \forall i \in I,\ j \in J, \tag{4.16}$$

$$y_{ij} - \sum_{k\in K} b_k \lambda_{ik} c_{j0} z_{jk} \leqslant 0 \quad \forall i \in I,\ j \in J, \tag{4.17}$$

$$x_{ij} - z_{jk} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in \mathscr{Q}_{ij}, \tag{4.18}$$

$$x_{ij} \geqslant 0 \quad \forall i \in I,\ j \in J, \tag{4.19}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J,\ k \in K, \tag{4.20}$$

$$y_{ij} \geqslant 0 \quad \forall i \in I,\ j \in J. \tag{4.21}$$

Although this linearization will not be used as a basis for the algorithms developed in this paper, it remains useful for comparison purposes.

4.3. Linearization III

The last linearization proposed for formulation $\mathscr{P}$ is similar to $\mathscr{L}_2(\mathscr{P})$, but differs with respect to the linearizing variable. This time the linearizing variable v_{jk} is chosen such that $v_{jk} = z_{jk}\sum_{i\in I} b_k \lambda_{ik} c_{j0} x_{ij}$, for each pair (j,k) in the second summation of the objective function (4.12). The following constraints can then be used to linearize the quadratic term:

$$v_{jk} \leqslant M z_{jk} \quad \forall j \in J,\ k \in K,$$

$$v_{jk} \leqslant \sum_{i\in I} b_k \lambda_{ik} c_{j0} x_{ij} \quad \forall j \in J,\ k \in K.$$

Based on these constraints, the third linearization problem $\mathscr{P}$ is given as follows:

$$\mathscr{L}_3(\mathscr{P}) \quad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k \lambda_{ik}(c_{ij}+c_{j0})x_{ij} - \sum_{j\in J}\sum_{k\in K} v_{jk} \tag{4.22}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I, \tag{4.23}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J, \tag{4.24}$$

$$v_{jk} - M z_{jk} \leqslant 0 \quad \forall j \in J,\ k \in K, \tag{4.25}$$

$$v_{jk} - \sum_{i\in I} b_k \lambda_{ik} c_{j0} x_{ij} \leqslant 0 \quad \forall j \in J,\ k \in K, \tag{4.26}$$

$$x_{ij} - z_{jk} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in \mathscr{Q}_{ij}, \tag{4.27}$$

$$x_{ij} \geqslant 0 \quad \forall i \in I,\ j \in J, \tag{4.28}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J,\ k \in K, \tag{4.29}$$

$$v_{jk} \geqslant 0 \quad \forall j \in J,\ k \in K. \tag{4.30}$$

We compare the three linearized formulations in terms of the number of constraints (n_c), along with the number of variables (n_v) and binary variables (n_b) in Table 2.

Table 2
Comparison of $\mathscr{L}_1(\mathscr{P})$, $\mathscr{L}_2(\mathscr{P})$ and $\mathscr{L}_3(\mathscr{P})$ in terms of size

	$\mathscr{L}_1(\mathscr{P})$	$\mathscr{L}_2(\mathscr{P})$	$\mathscr{L}_3(\mathscr{P})$
n_c	$\|I\|+\|J\|+3\|I\|\|J\|\|K\|$	$\|I\|+\|J\|+2\|I\|\|J\|+\|I\|\|J\|\|K\|$	$\|I\|+\|J\|+2\|J\|\|K\|+\|I\|\|J\|\|K\|$
n_v	$\|I\|\|J\|+\|J\|\|K\|+\|I\|\|J\|\|K\|$	$2\|I\|\|J\|+\|J\|\|K\|$	$\|I\|\|J\|+2\|J\|\|K\|$
n_b	$\|J\|\|K\|$	$\|J\|\|K\|$	$\|J\|\|K\|$

As can be seen from Table 2, all formulations have an equal number of integer variables but differ with respect to the total number of constraints and variables. It can be stated that, in general, $\mathscr{L}_1(\mathscr{P})$ is larger than the other two. Nevertheless, it will be difficult to solve directly any of the formulations presented here to optimality using integer programming software for large size instances. Amongst many types of solution approaches developed for $\mathscr{NP}$-Hard problems, a decomposition based approach seems attractive for the problem and the formulations at hand, which allows one to partition a formulation into smaller and easier-to-solve subproblems.

This is the approach taken in this paper. In what follows, we will offer two different solution algorithms, both based on decomposition. More specifically, we will describe a Benders decomposition procedure that is based on $\mathscr{L}_1(\mathscr{P})$ and a Lagrangean relaxation and decomposition procedure based on $\mathscr{L}_3(\mathscr{P})$. These are further explained in Sections 5 and 6, respectively.

5. A Benders decomposition algorithm for the solution of $\mathscr{L}_1(\mathscr{P})$

In this section, we present an algorithm for the solution of $\mathscr{L}_1(\mathscr{P})$. Our preliminary experimentation has shown that this formulation has the strongest linear programming bound. It is therefore chosen as a candidate for the algorithm that will be based on Benders decomposition [40]. As shown by Costa [41], the choice of this methodology is natural for this class of problems. In developing the algorithm, we exploit the structure of $\mathscr{L}_1(\mathscr{P})$ as much as possible.

Let $\mathscr{Z}$ be the set of binary vectors satisfying constraints (4.5) and (4.10). Given a vector $z^* \in \mathscr{Z}$. $\mathscr{L}_1(\mathscr{P})$ reduces to

$$\mathscr{SP}(z^*) \quad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij} - b_k\lambda_{ik}c_{j0}\varphi_{ijk}) \tag{5.1}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I, \tag{5.2}$$

$$\varphi_{ijk} - x_{ij} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in K, \tag{5.3}$$

$$\varphi_{ijk} \leqslant z^*_{jk} \quad \forall i \in I,\ j \in J,\ k \in K, \tag{5.4}$$

$$x_{ij} \leqslant z^*_{jk} \quad \forall i \in I,\ j \in J,\ k \in \mathscr{Q}_{ij}, \tag{5.5}$$

$$x_{ij} \geqslant 0 \quad \forall i \in I,\ j \in J. \tag{5.6}$$

Observe that the constraints $\varphi_{ijk} \in [0, 1]$ are dropped in $\mathscr{SP}(z^*)$ since these are implied by the remaining constraints. Moreover, for $i \in I$, $j \in J, k \in \mathscr{Q}_{ij}$, it is easily seen that constraints (5.3) and (5.5) imply (5.4), and therefore the latter are dropped for such triplets. A closer look at $\mathscr{SP}(z^*)$ shows that it can be decomposed into $|I|$ subproblems $\mathscr{SP}_i(z^*)$, one for each $i \in I$:

$$\mathscr{SP}_i(z^*) \quad \text{Minimize} \quad \sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij} - b_k\lambda_{ik}c_{j0}\varphi_{ijk}) \tag{5.7}$$

subject to

$$\sum_{j\in J} x_{ij} = 1,$$

$$\varphi_{ijk} - x_{ij} \leqslant 0 \quad \forall j \in J,\ k \in K,$$

$$\varphi_{ijk} \leqslant z^*_{jk} \quad \forall j \in J,\ k \notin \mathscr{Q}_{ij},$$

$$x_{ij} \leqslant z^*_{jk} \quad \forall j \in J,\ k \in \mathscr{Q}_{ij},$$

$$x_{ij} \geqslant 0 \quad \forall j \in J.$$

At this point, we define for each $i \in I$, $\mathscr{F}_i = \{j \in J | \mathscr{Q}_{ij} = \mathscr{R}_{ij} = \emptyset\}$ and $\mathscr{H}_i = \{j \in J | z^*_{jk} = 1, \forall k \in \mathscr{Q}_{ij}\}$. We now state a proposition regarding the solution of $\mathscr{SP}_i(z^*)$.

Proposition 3. *$\mathscr{SP}_i(z^*)$ is feasible and bounded if either $\mathscr{F}_i \neq \emptyset$, $\mathscr{H}_i \neq \emptyset$, or both.*

Proof. Since $\mathscr{SP}_i(z^*)$ is an assignment problem, it has a feasible solution if there exists at least one pair (i, j) for which x_{ij} can be equal to 1. If $\mathscr{F}_i \neq \emptyset$, then one can pick any $j \in \mathscr{F}_i$ that will provide a feasible solution to the problem. If, on the other hand, $\mathscr{F}_i = \emptyset$, then one has to have at least one pair (i, j) for which x_{ij} can be equal to 1. This would require $z^*_{jk} = 1$ for all $k \in \mathscr{Q}_{ij}$, hence $j \in \mathscr{H}_i$. The problem is bounded since all variables are upper bounded by 1. □

In the next proposition, we will show that when $\mathscr{SP}_i(z^*)$ is feasible, it is solvable by inspection.

Proposition 4. *$\mathscr{SP}_i(z^*)$ is solvable by inspection with an optimal solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\varphi}}) = \{\tilde{x}_{ij} \in \mathbb{B} | i \in I, j \in J\}$, $\{\tilde{\varphi}_{ijk} \geqslant 0 | i \in I, j \in J, k \in K\}$ satisfying*

$$\tilde{x}_{ip} = 1 \quad \textit{where } p \in \underset{j \in \mathscr{F}_i \cup \mathscr{H}_i}{\operatorname{argmin}} \left(\sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}) - \sum_{k \in K : z^*_{jk} = 1} b_k \lambda_{ik} c_{j0} \right), \tag{5.8}$$

$$\tilde{x}_{ij} = 0 \quad \textit{for } j \neq p, \tag{5.9}$$

and

$$\tilde{\varphi}_{ijk} = \begin{cases} 1 & \textit{if } z^*_{jk} = 1 \textit{ and } \tilde{x}_{ij} = 1, \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. Observe that since i can only be assigned to $j \in \mathscr{F}_i \cup \mathscr{H}_i$, the optimal assignment will be that with the minimum assignment cost. As there are no capacity constraints, given an $\tilde{x}_{ij} = 1$ for a specific pair $(i \in I, j \in J)$, we may set $\tilde{\varphi}_{ijk} = 1$ for all $j \in J, k \in K$ with $z^*_{jk} = 1$ and $\tilde{\varphi}_{ijk} = 0$ otherwise. □

Let $\boldsymbol{\alpha} = \{\alpha_i \in \mathbb{R} | i \in I\}$, $\boldsymbol{\theta} = \{\theta_{ijk} \geqslant 0 | i \in I, j \in J, k \in K\}$, $\boldsymbol{\omega} = \{\omega_{ijk} \geqslant 0 | i \in I, j \in J, k \notin \mathscr{Q}_{ij}\}$, $\boldsymbol{\zeta} = \{\zeta_{ijk} \geqslant 0 | i \in I, j \in J, k \in \mathscr{Q}_{ij}\}$ be the sets of dual variables associated with constraints (5.2), (5.3), (5.4), and (5.5), respectively. Then, the dual of the linear programming relaxation of $\mathscr{SP}_i(z^*)$ can be stated as follows:

$$\mathscr{DSP}_i(z^*) \quad \text{Maximize} \quad \alpha_i - \sum_{j \in J} \sum_{k \notin \mathscr{Q}_{ij}} z^*_{jk} \omega_{ijk} - \sum_{j \in J} \sum_{k \in \mathscr{Q}_{ij}} z^*_{jk} \zeta_{ijk} \tag{5.10}$$

subject to

$$\alpha_i + \sum_{k \in K} \theta_{ijk} - \sum_{k \in \mathscr{Q}_{ij}} \zeta_{ijk} \leqslant \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}) \quad \forall j \in J, \tag{5.11}$$

$$-\theta_{ijk} \leqslant -b_k \lambda_{ik} c_{j0} \quad \forall j \in J,\ k \in \mathscr{Q}_{ij}, \tag{5.12}$$

$$-\theta_{ijk} - \omega_{ijk} \leqslant -b_k \lambda_{ik} c_{j0} \quad \forall j \in J,\ k \notin \mathscr{Q}_{ij}, \tag{5.13}$$

$$\theta_{ijk}, \omega_{ijk}, \zeta_{ijk} \geqslant 0 \quad \forall j \in J,\ k \in K. \tag{5.14}$$

Let $\mathscr{D}_i$ be the feasible region of $\mathscr{DSP}_i(z^*)$ and $\mathscr{P}_i^{\mathscr{D}}$ be the set of extreme points of $\mathscr{D}_i$. As a result of Proposition 3, $\mathscr{DSP}_i(z^*)$ is bounded and feasible if either $\mathscr{F}_i \neq \emptyset$, $\mathscr{H}_i \neq \emptyset$, or both.

We now elaborate on the solution of $\mathscr{DSP}_i(z^*)$. First note that, $v(\mathscr{SP}_i(z^*)) = v(\mathscr{DSP}_i(z^*))$. The following proposition shows that an optimal solution to the dual subproblem can be found by inspection.

Proposition 5. *$\mathcal{DSP}_i(z^*)$ is solvable by inspection with an optimal solution $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\zeta}})$ satisfying*

$$\tilde{\alpha}_i = \min_{j \in \mathcal{F}_i \cup \mathcal{H}_i} \left\{ \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}) - \sum_{k \in K : z^*_{jk} = 1} b_k \lambda_{ik} c_{j0} \right\},$$

$$\tilde{\theta}_{ijk} = \begin{cases} b_k \lambda_{ik} c_{j0} & \text{if } k \in \mathcal{Q}_{ij}, \\ b_k \lambda_{ik} c_{j0} & \text{if } k \notin \mathcal{Q}_{ij} \text{ and } z^*_{jk} = 1, \\ 0 & \text{otherwise}, \end{cases}$$

$$\tilde{\omega}_{ijk} = \begin{cases} b_k \lambda_{ik} c_{j0} & \text{if } z^*_{jk} = 0, \\ 0 & \text{otherwise}, \end{cases}$$

$$\sum_{k \in K : z^*_{jk} = 0} \tilde{\zeta}_{ijk} = \tilde{\alpha}_i + \sum_{k \in K} \tilde{\theta}_{ijk} - \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}) \quad \forall j \in J.$$

Proof. It can be easily shown that $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\zeta}}) \in \mathcal{D}_i$. Now, observe that constraints (5.11) imply

$$\alpha_i \leqslant \min_{j \in J} \left\{ \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}) - \sum_{k \in K} \theta_{ijk} + \sum_{k \in \mathcal{Q}_{ij}} \zeta_{ijk} \right\}.$$

We consider two cases:

- If $j \notin \mathcal{F}_i$, then there exists at least one $k \in \mathcal{Q}_{ij}$. In this case, $\theta_{ijk} = b_k \lambda_{ik} c_{j0}$ by constraint (5.12). If $z^*_{jk} = 0$ for any $k \in \mathcal{Q}_{ij}$, then one can observe that constraint (5.11) will never be binding since one chooses all $\tilde{\zeta}_{ijk}$ with $z^*_{jk} = 0$ so as to satisfy $\sum_{k \in K : z^*_{jk} = 0} \tilde{\zeta}_{ijk} \geqslant \tilde{\alpha}_i + \sum_{k \in K} \tilde{\theta}_{ijk} - \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0})$. Note that since $z^*_{jk} = 0$ implies $\tilde{x}_{ij} = 0$, this solution satisfies the complementary slackness condition $\tilde{\zeta}_{ijk}(z^*_{jk} - \tilde{x}_{ij}) = 0$. On the other hand, if $z^*_{jk} = 1$ for all $k \in \mathcal{Q}_{ij}$, then it is clear that $j \in \mathcal{H}_i$. In this case, $\tilde{\zeta}_{ijk} = 0$ by the objective function (5.10). This implies that α_i is only constrained by constraints (5.11) defined over $j \in \mathcal{F}_i \cup \mathcal{H}_i$, i.e.,

$$\alpha_i \leqslant \min_{j \in \mathcal{F}_i \cup \mathcal{H}_i} \left\{ \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}) - \sum_{k \in K} \theta_{ijk} + \sum_{k \in \mathcal{Q}_{ij}} \zeta_{ijk} \right\}.$$

  Note also that since $\tilde{\zeta}_{ijk} = 0$, this solution satisfies the complementary slackness condition $\tilde{\zeta}_{ijk}(z^*_{jk} - \tilde{x}_{ij}) = 0$.
- If $j \in \mathcal{F}_i$, then $\mathcal{Q}_{ij} = \emptyset$. If $z^*_{jk} = 0$, then the objective function together with constraints (5.11) and (5.13) imply $\tilde{\omega}_{ijk} = b_k \lambda_{ik} c_{j0}$ and $\tilde{\theta}_{ijk} = 0$. Since $z^*_{jk} = 0$ implies $\tilde{\varphi}_{ijk} = 0$, the complementary slackness condition $\tilde{\omega}_{ijk}(z^*_{jk} - \tilde{\varphi}_{ijk}) = 0$ is also satisfied. If, on the other hand, $z^*_{jk} = 1$, then $\tilde{\omega}_{ijk} = 0$ and $\tilde{\theta}_{ijk} = b_k \lambda_{ik} c_{j0}$. Given $z^*_{jk} = 1$ for all $k \notin \mathcal{Q}_{ij}$, since we can set $\tilde{\varphi}_{ijk} = 1$ (see Proposition 4), the complementary slackness condition $\tilde{\omega}_{ijk}(z^*_{jk} - \tilde{\varphi}_{ijk}) = 0$ is also satisfied for this case.

Finally, given an optimal primal solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\varphi}})$ obtained by Proposition 4 and a dual solution $(\tilde{\alpha}, \tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\zeta}})$ given above, one can see that the values primal and dual objective functions are equal and given by the expression

$$\min_{j \in \mathcal{F}_i \cup \mathcal{H}_i} \left\{ \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}) - \sum_{k \in K : z^*_{jk} = 1} b_k \lambda_{ik} c_{j0} \right\},$$

proving that $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\zeta}})$ is also optimal. □

In the case that $\mathcal{SP}_i(z^*)$ is infeasible, $\mathcal{DSP}_i(z^*)$ will be unbounded. Let $\mathcal{W}_i^{\mathcal{D}}$ be the set of extreme rays of $\mathcal{DSP}_i(z^*)$. In this case, every extreme ray $(\boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\zeta}) \in \mathcal{W}_i^{\mathcal{D}}$ will induce a feasibility cut of the form

$$\alpha_i - \sum_{j\in J}\sum_{k\in K} z_{jk}(\omega_{ijk} + \zeta_{ijk}) \leqslant 0. \tag{5.15}$$

Letting $\xi_i = \alpha_i - \sum_{j\in J}\sum_{k\in K} z_{jk}(\omega_{ijk} + \zeta_{ijk})$, one obtains the Benders reformulation of $\mathcal{L}_1(\mathcal{S})$, which is henceforth referred to as the master problem $(\mathcal{MP})$:

$$(\mathcal{MP}) \quad \text{Minimize} \quad \sum_{i\in I} \xi_i \tag{5.16}$$

subject to

$$\xi_i + \sum_{j\in J}\sum_{k\in K} z_{jk}(\omega_{ijk} + \zeta_{ijk}) \geqslant \alpha_i \quad (\boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\zeta}) \in \mathcal{P}_i^{\mathcal{D}}, \tag{5.17}$$

$$\alpha_i - \sum_{j\in J}\sum_{k\in K} z_{jk}(\omega_{ijk} + \zeta_{ijk}) \leqslant 0 \quad (\boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\zeta}) \in \mathcal{W}_i^{\mathcal{D}}, \tag{5.18}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J,$$

$$z_{jk} \in \{0, 1\} \quad \forall j \in J, \ \ k \in K.$$

In $\mathcal{MP}$, constraints (5.17) are usually referred to as *Benders optimality cuts* while constraints (5.18) are called *Benders feasibility cuts*. The $\mathcal{MP}$ includes a very large number of Benders cuts, one for each extreme point in the set $\mathcal{P}^{\mathcal{D}} = \bigcup_{i\in I} \mathcal{P}_i^{\mathcal{D}}$ and for each extreme ray in the set $\mathcal{W}^{\mathcal{D}} = \bigcup_{i\in I} \mathcal{W}_i^{\mathcal{D}}$. However, the problem may be solved by initially relaxing all the cuts and generating them dynamically only when they are violated by the solution to the master problem. Let k denote the iteration number and $\mathcal{P}^{\mathcal{D}_k}$ and $\mathcal{W}^{\mathcal{D}_k}$ denote the restricted sets of extreme points and extreme rays of $\mathcal{D}$ at iteration k. We call the resulting program the *relaxed master problem* and denote it by $\mathcal{MP}^k$. One can then dynamically generate the remaining cuts by solving $\mathcal{SP}(z^*)$ with a given set of $z^* \in \mathcal{Z}$. Such an approach is usually called a delayed constraint generation algorithm, for which we provide an outline below:

A delayed constraint generation algorithm.

1. Let $\mathcal{LB} = -\infty$, $\mathcal{UB} = \infty$, $k = 0$.
2. Solve the relaxed master problem $\mathcal{MP}^k$ to obtain a solution z^* and $\mathcal{LB} = v(\mathcal{MP}^k)$.
3. Solve the subproblem $\mathcal{SP}(z^*)$.
   (a) If $\mathcal{SP}(z^*)$ is feasible, let (x^*, φ^*) and $(\alpha^*, \theta^*, \omega^*, \zeta^*)$ be the primal and dual optimal solutions to $\mathcal{SP}(z^*)$ and $\mathcal{DSP}(z^*)$, respectively.
      (i) If $v(\mathcal{SP}(z^*)) = v(\mathcal{MP}^k)$, then stop. This implies that (x^*, φ^*) is also optimal to the original problem.
      (ii) If $v(\mathcal{SP}(z^*)) > \mathcal{LB}$, then set $\mathcal{UB} = \min\{v(\mathcal{SP}(z^*)), \mathcal{UB}\}$, and $\mathcal{P}^{\mathcal{D}_{k+1}} = \mathcal{P}^{\mathcal{D}_k} \cup \{(\boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\zeta})\}$ to generate an optimality cut.
   (b) If $\mathcal{SP}(z^*)$ is infeasible, then $\mathcal{W}^{\mathcal{D}_{k+1}} = \mathcal{W}^{\mathcal{D}_k} \cup \{(\boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\zeta})\}$ to generate a feasibility cut.
4. Set $k \leftarrow k + 1$ and return to Step 2.

In any iteration of the Benders algorithm, the optimal solution value of the $\mathcal{MP}$ is a lower bound on the optimal solution value of the main problem. The optimal solution value of $\mathcal{SP}(z^*)$, on the other hand, is an upper bound which is not necessarily decreasing at each iteration. This is why the upper bound is chosen as $\mathcal{UB} = \min\{v(\mathcal{SP}(z^*)), \mathcal{UB}\}$ in Step (3) of the algorithm. In addition, we note that to avoid the master problem from being unbounded in the first few iterations of the algorithm, we generate a number of cuts from feasible solutions which are initially added to the $\mathcal{MP}$.

6. A Lagrangean relaxation scheme for the solution of $\mathcal{L}_3(\mathcal{P})$

In this section, we will demonstrate that a certain Lagrangean relaxation for formulation $\mathcal{L}_3(\mathcal{P})$ yields a special structure that enables us to decompose the formulation into efficiently solvable substructures.

To this extent, we first replace the QoS constraints (4.27) with the following equivalent set of constraints:

$$x_{ij} - Mz_{jk} \leqslant 0 \quad \forall i \in I,\ j \in J,\ k \in \mathcal{Q}_{ij}. \tag{6.1}$$

As will be seen shortly, this modification alters the objective function of the relaxed problem to prevent numerical problems that may arise with the use of the big M factor. We now propose to relax constraints (4.25) and (6.1) in a Lagrangean fashion by associating Lagrange multipliers γ_{jk} and η_{ijk} to constraints (4.25) and (6.1), respectively. The relaxation process results in the following problem, denoted by $\mathcal{LR}(\gamma, \eta)$:

$$\text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij}+c_{j0})+\eta_{ijk})x_{ij}+\sum_{j\in J}\sum_{k\in K}(\gamma_{jk}-1)v_{jk}-M\sum_{j\in J}\sum_{k\in K}\left(\gamma_{jk}+\sum_{i\in I}\eta_{ijk}\right)z_{jk}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I,$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J,$$

$$v_{jk} - \sum_{i\in I} b_k\lambda_{ik}c_{j0}x_{ij} \leqslant 0 \quad \forall j \in J,\ k \in K,$$

$$x_{ij} \geqslant 0 \quad \forall i \in I,\ j \in J,$$

$$z_{jk} \in \{0, 1\} \quad \forall j \in J,\ k \in K.$$

The relaxed problem $\mathcal{LR}(\gamma, \eta)$ decomposes into two subproblems, $\mathcal{SP}_1$ and $\mathcal{SP}_2$, where

$$(\mathcal{SP}_1) \quad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij}+c_{j0})+\eta_{ijk})x_{ij}+\sum_{j\in J}\sum_{k\in K}(\gamma_{jk}-1)v_{jk}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I,$$

$$v_{jk} - \sum_{i\in I} b_k\lambda_{ik}c_{j0}x_{ij} \leqslant 0 \quad \forall j \in J,\ k \in K,$$

$$x_{ij} \geqslant 0 \quad \forall i \in I,\ j \in J,$$

$$v_{jk} \geqslant 0 \quad \forall j \in J,\ k \in K$$

and

$$(\mathcal{SP}_2) \quad \text{Maximize} \quad M\sum_{j\in J}\sum_{k\in K}\left(\gamma_{jk}+\sum_{i\in I}\eta_{ijk}\right)z_{jk}$$

subject to

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J,$$

$$z_{jk} \in \{0, 1\} \quad \forall j \in J,\ k \in K.$$

$\mathcal{SP}_1$ is an assignment problem which can be solved by inspection as shown in the following proposition:

Proposition 6. *The optimal solution* $(\boldsymbol{x}^*, \boldsymbol{v}^*)$ *to* $\mathcal{SP}_1$ *can be calculated as*

$$v^*_{jk} = \begin{cases} 0 & \text{if } \gamma_{jk} - 1 \geqslant 0, \\ \sum\limits_{i\in I} b_k\lambda_{ik}c_{j0}x_{ij} & \text{otherwise} \end{cases}$$

for every $j \in J, k \in K$ *and* $x^*_{ip} = 1$, *where* $p \in \operatorname{argmin}_{j\in J}\{\sum_{k\in K}(b_k\lambda_{ik}(c_{ij} + c_{j0}) + \eta_{ijk})\}$ *for every* $i \in I$.

Proof. The proof is based on the observation that variables v_{jk} only appear in the second set of constraints in $\mathcal{SP}_1$. Therefore, if the objective function coefficient of v_{jk} is nonnegative, then $v^*_{jk}=0$. If not, it is equal to $\sum_{i\in I} b_k\lambda_{ik}c_{j0}x_{ij}$. Hence,

$$(\gamma_{jk}-1)v_{jk}=\min\{0,\gamma_{jk}-1\}\sum_{i\in I} b_k\lambda_{ik}c_{j0}x_{ij}$$

will hold in an optimal solution of $\mathcal{SP}_1$. The optimal values of the x variables can be obtained through the trivial subproblem that remains when variables v_{jk} are removed. □

As for subproblem $\mathcal{SP}_2$, it is easy to show that it further decomposes into $|J|$ unidimensional knapsack problems, one for each proxy $j\in J$. It is known that each knapsack problem can be solved by dynamic programming in $\mathcal{O}(|K|s_j)$ for $j\in J$. Therefore, for a given set of Lagrange multipliers γ_i, subproblem $\mathcal{SP}_2$ can be solved in $\mathcal{O}(|J||K|s_j)$ time.

It is clear that the solution to $\mathcal{LR}(\gamma,\eta)$ for a set of multipliers is a lower bound for $\mathcal{L}_3(\mathcal{P})$. Then, one is interested in finding the best possible lower bound for this problem, i.e., the solution to $\max_{\gamma,\eta\geqslant 0}\{LR(\gamma,\eta)\}$. This problem is a piecewise linear concave optimization problem which can be solved by means of subgradient optimization [42]. We provide below an outline of the algorithm:

Subgradient optimization algorithm.

1. Start with an initial vector of multipliers γ^1,η^1. Set the incumbent lower bound as $\mathcal{LB}=-\infty$ and the incumbent upper bound as $\mathcal{UB}=\infty$. Set $t=1$.
2. Repeat the following until $gap=(\mathcal{UB}-\mathcal{LB})/\mathcal{UB}<\varepsilon$ or the maximum number of iterations has been reached.
   (a) Solve $\mathcal{LR}(\gamma^t,\eta^t)$. If $v(\mathcal{LR}(\gamma^t,\eta^t))>\mathcal{LB}$, set $\mathcal{LB}=v(\mathcal{LR}(\gamma^t,\eta^t))$.
   (b) Update multipliers as follows:

$$\gamma^{t+1}=\max\{0,\gamma^t+s_1^t\cdot g_1^t\},$$
$$\eta^{t+1}=\max\{0,\gamma^t+s_2^t\cdot g_2^t\},$$

   where g_1^t and g_2^t are the subgradient vectors at iteration t and s_1^t and s_2^t are the steplengths. The (j,k)th component of g_1^t is defined as

$$(g_1^t)_{jk}=v_{jk}-Mz_{jk}.$$

   Similarly, the (i,j,k)th component of g_2^t is defined as

$$(g_2^t)_{ijk}=x_{ij}-Mz_{jk}.$$

   The steplengths s_i^t $(i=1,2)$ are calculated as follows:

$$s_1^t=\lambda\frac{\mathcal{UB}-v(\mathcal{LR}(\gamma^t,\eta^t))}{\|g_1^t\|^2}$$

   and

$$s_2^t=\lambda\frac{\mathcal{UB}-v(\mathcal{LR}(\gamma^t,\eta^t))}{\|g_2^t\|^2}.$$

   (c) Set $t\leftarrow t+1$.

Next, we discuss how one can calculate $\mathcal{UB}$, which is required in Step 2(b) of this algorithm.

6.1. Computation of upper bounds

In the subgradient algorithm, one needs to calculate at each iteration a valid upper bound ($\mathcal{UB}$) in order to be able to calculate the steplength given in (6.2). For this purpose, we will use formulation $\mathcal{P}$. Note that, for a given set of

$z^* \in \mathscr{Z}$, $\mathscr{P}$ becomes

$$\text{Minimize} \quad \sum_{i \in I} \sum_{j \in \mathscr{F}_i \cup \mathscr{H}_i} \hat{c}_{ij} x_{ij}$$

subject to

$$\sum_{j \in \mathscr{F}_i \cup \mathscr{H}_i} x_{ij} = 1 \quad \forall i \in I,$$

$$x_{ij} \geqslant 0 \quad \forall i \in I, \ j \in \mathscr{F}_i \cup \mathscr{H}_i,$$

where $\hat{c}_{ij} = \sum_{k \in K} b_k \lambda_{ik} (c_{ij} + c_{j0}(1 - z^*_{jk}))$. It is easy to see that this is an assignment problem, which can be solved in polynomial time. More specifically, for each $i \in I$, the optimal solution lies in $x_{ip} = 1$, where $p \in \operatorname{argmin}_{j \in \mathscr{F}_i \cup \mathscr{H}_i} \{\hat{c}_{ij}\}$. Since the solution of $\mathscr{SP}_2$ outputs a feasible z^* vector, one can identify the corresponding set of x_{ij}'s, and therefore the corresponding feasible solution, in $\mathscr{O}(|I||J|)$ time. When the set $\mathscr{F}_i \cup \mathscr{H}_i$ is empty, then $\mathscr{UB}$ is not updated in the algorithm and the value of the previous iteration is used.

Note that, although we demonstrate here one way of implementing Lagrangean relaxation, other relaxations are also possible. For an example, one might consider relaxing constraints (4.23), (4.25) and (4.26). Note, however, that this relaxation would require the optimization of three sets of multipliers as opposed to only two.

7. Computational results

We now provide the results of computational experiments to demonstrate the efficiency of the proposed solution procedures. The computational experiments were performed using randomly generated data, where some real-life aspects of a content distribution environment were incorporated in the data generation process as much as possible. To this extent, the topologies used for the experiments were generated using an Internet topology generator GT-ITM [43] that mimics the characteristics of the real Internet topology. The unit transfer cost between nodes i and j was interpreted as the number of hops (logical links) between these nodes. While any other parameter, such as the unit bandwidth cost, can be used as the cost metric, these may be hard to obtain in practice. In contrast, the hop count between two nodes is relatively easy to obtain and is stated to be "a decent indicator of the path's proximity, reliability, and stability" [7]. Moreover, it is generally assumed in the literature that the amount of latency (or delay) between two nodes is proportional to the number of hops between these two nodes [44]. Further, Huang and Abdelzaher [45] report that the "average network latency of downloading a file is roughly proportional to its size when the file size is between 1 and 100 KB", where the reported range contains a significant portion of Web objects [1]. We have therefore generated the size of each object to be a continuous uniform random variable between 1 and 100. The capacity of each proxy server was defined as 40% of the total size of all objects. To determine the latency parameters, we have defined object groups in terms of their sizes and associated a nonnegative value for both unit delay and the latency bound, which can be seen in Table 3. For simplicity, we assume that the unit delay u_l^k is the same for a given content class and given in milliseconds (ms). The delay caused by transferring an object k over link (i, j) is then calculated as $t(b_k, d_{ij}) = u_l^k d_{ij}$, where $d_{ij} = \psi c_{ij}$ and ψ is a constant. Similarly, the latency bound Δ_d is the same for all objects in a specific content group and is also given in milliseconds. We assume that the minimum delay that a client experiences to reach the nearest

Table 3
Latency parameters for the objects

Group	size (KB)	u_l^k (ms)	Δ_d (ms)
1	$1 \leqslant b_k \leqslant 15$	20	100
2	$16 \leqslant b_k \leqslant 31$	30	200
3	$32 \leqslant b_k \leqslant 63$	40	400
4	$64 \leqslant b_k \leqslant 100$	50	600

server in retrieving an object is 20 ms (see the measurement results given in [46] on two major commercial CDNs) and increases with object size.

It it well known (see [47]) that the demand distribution of Web requests follows a Zipf-like distribution [48]. We have therefore used this distribution when generating the object request patterns. Let a set of objects be ranked in order of their popularity where object i in this order is the ith most popular object. The Zipf-like distribution assumes that the probability of a request for an object is inversely proportional to $i^{-\alpha}$. More specifically, given an arrival for a request, the conditional probability that the request is for object i is given as

$$P_K(i) = \Omega i^{-\alpha},$$

where

$$\Omega = \left(\sum_{j=1}^{K} j^{-\alpha} \right)^{-1}$$

is a normalization constant. When $\alpha = 1$, we have the true Zipf-distribution. In Breslau et al. [47], it is shown that α varies from 0 to 1 for different access patterns and is usually between 0.64 and 0.83 for Web objects. This value was recorded to be $\alpha = 0.733$ for multimedia files in Yang and Fei [14]. We have used this specific value in our implementation.

All the experiments were performed on a 3 GHz Pentium 4 workstation running Linux. The decomposition procedures were coded in C. The callable library of CPLEX 9.01 was used to solve all linear and integer subproblems. The parameters of CPLEX were in their default setting. Specific implementation details for each procedure are further explained below.

7.1. Implementation details of the Benders decomposition procedure

Although Benders decomposition partitions the original problem into two subproblems that are relatively easier to solve as compared to the original problem, it is known that $\mathscr{MP}$ is still a bottleneck of this solution procedure. This can be explained by the fact that the difficulty of solving the $\mathscr{MP}$ in our case increases with the number of proxy servers ($|J|$) and the number of objects ($|K|$), since the number of binary variables in $\mathscr{MP}$ is $|J||K|$. In addition, the number of constraints of the $\mathscr{MP}$ increases by an amount of $|I|$ at every iteration, since a single optimality cut is added to the $\mathscr{MP}$ for every $i \in I$. It is clear that the solution of the $\mathscr{MP}$ will be arduous especially in the later iterations of the algorithm.

It is worth pointing out that in the following experiments, the primal subproblem turned out to be feasible in every iteration of the Benders decomposition procedure (hence eliminating the need to introduce feasibility cuts). This is an expected situation, since in practice it is likely that one can find several proxy servers to which a client can be assigned without violating the QoS restrictions.

We denote by $\mathscr{BD}_1$ the standard implementation of the Benders decomposition algorithm as stated in Section 5. Below, we discuss some techniques used to improve the convergence and/or efficiency of the algorithm.

7.1.1. Generating Pareto-optimal cuts

The proof of Proposition 5 suggests that the dual subproblems may have multiple optimal solutions. Each of these solutions will yield a valid optimality cut of the form (5.17), although some may lead stronger cuts than others. Let $(\boldsymbol{\alpha}^1, \boldsymbol{\theta}^1, \boldsymbol{\omega}^1, \boldsymbol{\zeta}^1)$ and $(\boldsymbol{\alpha}^2, \boldsymbol{\theta}^2, \boldsymbol{\omega}^2, \boldsymbol{\zeta}^2)$ be two points in $\mathscr{P}_i^{\mathscr{D}}$. The Benders cut generated from the former is said to dominate that of the latter if and only if

$$\sum_{i\in I} \alpha_i^1 - \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} z_{jk}(\omega_{ijk}^1 + \zeta_{ijk}^1) \geqslant \sum_{i\in I} \alpha_i^2 - \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} z_{jk}(\omega_{ijk}^2 + \zeta_{ijk}^2) \tag{7.1}$$

for all $z \in \mathscr{Z}$ with strict inequality for at least one point. If there exists a cut that is not dominated by any other cut, it is said to be *Pareto-optimal* [49]. These types of cuts have been used by Van Roy [50] and Wentges [51] for the

capacitated facility location problem. Let $\mathscr{Z}^{\mathrm{LP}} = \{z \in [0,1]^{|J||K|} : z \text{ satisfies (4.5)}\}$ and $ri(\mathscr{Z}^{\mathrm{LP}})$ denote the relative interior of $\mathscr{Z}^{\mathrm{LP}}$. In order to determine an optimal solution to the $\mathscr{DSP}_i(z^*)$ that yields a Pareto-optimal cut, the following problem must be solved for each $i \in I$, where $z^0 \in ri(\mathscr{Z}^{\mathrm{LP}})$:

$$\text{Maximize} \quad \alpha_i - \sum_{j\in J}\sum_{k\notin \mathscr{Q}_{ij}} z^0_{jk}\omega_{ijk} - \sum_{j\in J}\sum_{k\in \mathscr{Q}_{ij}} z^0_{jk}\zeta_{ijk} \tag{7.2}$$

subject to

$$\alpha_i - \sum_{j\in J}\sum_{k\notin \mathscr{Q}_{ij}} z^*_{jk}\omega_{ijk} - \sum_{j\in J}\sum_{k\in \mathscr{Q}_{ij}} z^*_{jk}\zeta_{ijk} = v(\mathscr{DSP}_i(z^*)), \tag{7.3}$$

$$\alpha_i + \sum_{k\in K}\theta_{ijk} - \sum_{k\in \mathscr{Q}_{ij}}\zeta_{ijk} \leqslant \sum_{k\in K} b_k\lambda_{ik}(c_{ij} + c_{j0}) \quad \forall j \in J,$$

$$-\theta_{ijk} \leqslant -b_k\lambda_{ik}c_{j0} \quad \forall j \in J,\ k \in \mathscr{Q}_{ij},$$

$$-\theta_{ijk} - \omega_{ijk} \leqslant -b_k\lambda_{ik}c_{j0} \quad \forall j \in J,\ k \notin \mathscr{Q}_{ij},$$

$$\theta_{ijk}, \omega_{ijk}, \zeta_{ijk} \geqslant 0 \quad \forall j \in J,\ k \in K.$$

In this problem, constraints (7.3) assure that an extreme point will be chosen from the set of optimal solutions to the original dual subproblem as the Pareto-optimal cut. Finding a point from the relative interior of the convex hull of integer solutions can be $\mathscr{NP}$-hard. In our experiments, we choose a point z^0 satisfying $0 < z^0 < 1$. Note that while such a point does not guarantee the identification of a cut which is undominated over the convex hull of integer solutions, it will nonetheless provide a Pareto-optimal cut over a set which includes this convex hull.

To generate the best possible cut, one has to spend extra computational effort to solve the auxiliary problem stated above. In contrast, such cuts may considerably help in improving the convergence of the algorithm (see, e.g. [52]). We refer to the Benders decomposition algorithm using the Pareto-optimal cut generating scheme stated above as $\mathscr{BD}_2$.

7.1.2. Master problem relaxation

One alternative way to accelerate the decomposition procedure, as suggested by McDaniel and Devine [53], is to partition the procedure into two stages. In the first stage, the linear programming relaxation of the $\mathscr{MP}$ is solved, until either (i) a prespecified amount of gap is obtained or (ii) a certain number of iterations is reached. In the second stage, integrality constraints are added back to $\mathscr{MP}$ and the algorithm is restarted to solve the integer programming problem to optimality. Note that in this modification, the lower bound provided by the LP relaxation of the $\mathscr{MP}$ in the first stage will still be a valid bound for the original problem. We refer to the Benders decomposition algorithm using this modification as $\mathscr{BD}_3$. In implementing this variation, the $\mathscr{MP}$ is solved as a linear program until $gap = (\mathscr{UB} - \mathscr{LB})/\mathscr{UB}$ falls below 1%, after which the integrality constraints are imposed on the z variables and the $\mathscr{MP}$ is solved henceforth as a mixed integer linear program.

7.1.3. Cut aggregation

As previously stated, the solution of the $\mathscr{MP}$ may be very tedious as the number of iterations (and therefore the number of cuts added) increases heavily with the number of iterations. To overcome this drawback, one may choose to add at every iteration a single cut that is stated as follows:

$$\sum_{i\in I}\xi_i + \sum_{j\in J}\sum_{k\in K} z_{jk}\left(\sum_{i\in I}(\omega_{ijk} + \zeta_{ijk})\right) \geqslant \sum_{i\in I}\alpha_i, \tag{7.4}$$

which is an aggregation of $|I|$ optimality cuts (5.17). In this scheme, one may also generate Pareto optimal cuts by solving only a single linear program that is stated as follows:

$$\text{Maximize} \quad \sum_{i\in I}\alpha_i - \sum_{i\in I}\sum_{j\in J}\sum_{k\notin\mathcal{Q}_{ij}} z^0_{jk}\omega_{ijk} - \sum_{i\in I}\sum_{j\in J}\sum_{k\in\mathcal{Q}_{ij}} z^0_{jk}\zeta_{ijk} \tag{7.5}$$

subject to

$$\sum_{i\in I}\alpha_i - \sum_{i\in I}\sum_{j\in J}\sum_{k\notin\mathcal{Q}_{ij}} z^*_{jk}\omega_{ijk} - \sum_{i\in I}\sum_{j\in J}\sum_{k\in\mathcal{Q}_{ij}} z^*_{jk}\zeta_{ijk} = v(\mathcal{DSP}(z^*)), \tag{7.6}$$

$$\alpha_i + \sum_{k\in K}\theta_{ijk} - \sum_{k\in\mathcal{Q}_{ij}}\zeta_{ijk} \leqslant \sum_{k\in K} b_k\lambda_{ik}(c_{ij}+c_{j0}) \quad \forall i\in I,\ j\in J,$$

$$-\theta_{ijk} \leqslant -b_k\lambda_{ik}c_{j0} \quad \forall i\in I,\ j\in J,\ k\in\mathcal{Q}_{ij},$$

$$-\theta_{ijk} - \omega_{ijk} \leqslant -b_k\lambda_{ik}c_{j0} \quad \forall i\in I,\ j\in J,\ k\notin\mathcal{Q}_{ij},$$

$$\theta_{ijk}, \omega_{ijk}, \zeta_{ijk} \geqslant 0 \quad \forall i\in I,\ j\in J,\ k\in K,$$

where $v(\mathcal{DSP}(z^*)) = \sum_{i\in I} v(\mathcal{DSP}_i(z^*))$. We refer to the Benders decomposition algorithm using a cut aggregation strategy as $\mathcal{BD}_4$.

7.1.4. Approximate solutions of the master problem

Our preliminary experimentation suggested that it is indeed very hard to solve the $\mathcal{MP}$ to optimality even after a few iterations, especially when $|I|$ is large. We have therefore resorted to a strategy in which the $\mathcal{MP}$ is not solved to optimality, but solved in an approximate manner so as to obtain a solution that is within a certain vicinity of the optimal solution. More specifically, we stop the branch-and-bound search after finding a feasible integer solution within $\kappa\%$ of the current best lower bound. The gap κ is initially set to a fairly high value and is decreased in a progressive manner through the iterations. The value of κ is chosen as a decreasing function of the iteration number. In our implementation, we have used the function $\kappa = 100/\sqrt{(k+1)}$, with $k = 1, 2, \ldots$ being the iteration number.

In implementing this variation, CPLEX has been forced to emphasize on obtaining the best possible bound for the approximate solution of the $\mathcal{MP}$ through its callable library function `CPXsetdblparam(env, CPX_PARAM_MIPEMPHASIS, 3)`.

7.1.5. Cut elimination

In order to reduce the number of cuts in the $\mathcal{MP}$, we have also employed a cut elimination strategy in which the cuts are appended only if they are violated by the current solution.

We would also like to mention that several cut dropping schemes were tested in implementing the Benders decomposition algorithm. However, these schemes did not prove to be useful as the cuts that are dropped were seen to be regenerated at later iterations. Of notable importance, we have noticed that there is always a tradeoff between the number of cuts dropped and the improvement in the gap. More specifically, as the number of dropped cuts is increased, the improvement over the gap decreases through the iterations. Following this observation, no cut dropping schemes were used in implementing the algorithm.

7.2. Implementation details of the Lagrangean relaxation and decomposition procedure

In implementing the Lagrangean relaxation and decomposition procedure, each knapsack problem was solved using the algorithm of Pisinger [54], which is publicly available at the address `http://www.diku.dk/~pisinger/minknap.c`. Note that this code requires as an input integer coefficients, whereas the coefficients of $\mathcal{SP}_2$ are fractional. This is easily handled by using a suitable scaling on the original coefficients and rescaling the optimal objective function value.

The parameters of the subgradient optimization algorithm are chosen as follows: λ is initially set to 2.0 and is halved if there is no improvement in the best known upper bound for 50 successive iterations. The multipliers are initialized to zero at the beginning of the algorithm.

Table 4
Comparison of the procedures $\mathscr{BD}_1$ and $\mathscr{BD}_2$ in terms of the solution values, gap and time

$\|J\|$	$\|I\|$	$\|K\|$	$v(\mathscr{BD}_1)$	$g(\mathscr{BD}_1)$	$t(\mathscr{BD}_1)$	$v(\mathscr{BD}_2)$	$g(\mathscr{BD}_2)$	$t(\mathscr{BD}_2)$
3	10	10	2736.43	21.98	67.02	2747.23	10.90	33.80
3	10	20	3136.62	38.28	1361.16	3019.18	11.34	54.06
3	10	30	3275.61	44.89	3512.91	3126.72	16.54	126.13
3	10	40	3237.16	46.30	2689.96	3057.19	11.58	96.74
3	10	50	3441.29	53.88	317.71	3135.77	14.57	140.79

Table 5
Effect of solving the $\mathscr{MP}$ approximately on the solution values, number of iterations and gap

$\|J\|$	$\|I\|$	$\|K\|$	$v(\mathscr{BD}_o)$	$i(\mathscr{BD}_o)$	$g(\mathscr{BD}_o)$	$c(\mathscr{BD}_o)$	$v(\mathscr{BD}_a)$	$i(\mathscr{BD}_a)$	$g(\mathscr{BD}_a)$	$c(\mathscr{BD}_a)$
3	50	10	15 268.15	16	11.36	800	15 266.14	46	14.08	2300
3	50	20	15 854.67	7	26.31	350	15 101.62	40	18.18	2000
3	50	30	16 227.30	5	26.29	250	15 872.94	39	21.28	1950
3	50	40	16 002.31	4	29.48	200	15 986.33	39	23.12	1950
3	50	50	18 345.50	3	32.40	150	17 027.17	35	25.40	1750

7.3. Results of the computational experiments

The proposed procedures were tested on randomly generated problems. This section includes the results of the computational experiments.

7.3.1. Effect of Pareto-optimal cuts

In order to see the effect of Pareto-optimal cuts, we compare $\mathscr{BD}_1$ and $\mathscr{BD}_2$ on a rather small sized set of instances. Since our aim is to see the effect of such cuts in the earlier steps of the algorithm, we have set a small common limit of 50 iterations for both versions of the algorithm.

The results of this experiment are given in Table 4, where the best solution value, the final gap and the time recorded as a result of running both variants at the end of the iteration limit are denoted, respectively, by $v(\cdot)$, $g(\cdot)$ and $t(\cdot)$. The gap is calculated as $g(\cdot) = 100((\mathscr{UB} - \mathscr{LB})/\mathscr{UB})$.

The results of Table 4 show that, even with very small size instances, there is a significant reduction in the final gap with the use of Pareto-optimal cuts. Interestingly, the use of Pareto-optimal cuts also helps in improving the convergence of the algorithm for these instances by considerably reducing the time needed to perform the 50 iterations. Based on these results, we have excluded $\mathscr{BD}_1$ from future comparisons.

7.3.2. Effect of solving the $\mathscr{MP}$ approximately

In this section, we provide some computational results to see the effect of solving the $\mathscr{MP}$ approximately as opposed to solving it to optimality. In this case, one may expect the algorithm to be able to perform more iterations. However, one may also expect to see a degradation of performance of the algorithm in terms of convergence. In order to clarify these issues, we report the results of our experiments in Table 5. In this table, we present the results of using two variations of $\mathscr{BD}_2$; one where $\mathscr{MP}$ is solved to optimality (denoted by $\mathscr{BD}_o$) and one where $\mathscr{MP}$ is solved approximately as explained in Section 7.1.4 (denoted by $\mathscr{BD}_a$). In order to see how much the suggested scheme will improve the decomposition algorithm, we compare the two variants under a common time limit of 600 s (10 min) and report the value of the feasible solution, the number of iterations performed and the final gap obtained by both variants. A notation similar to that of the previous table has been used in this table, with the additional indicators $i(\cdot)$ and $c(\cdot)$ denoting the total number of iterations performed by the algorithms and the total number of cuts in the $\mathscr{MP}$ at the end of the time limit.

As shown by the results given in Table 5, solving the $\mathscr{MP}$ approximately has a positive effect on the number of iterations that can be performed in a given time limit. Moreover, not only does one obtain a better gap as a result of

Table 6
Effect of the cut elimination strategy on the solution values, solution time and gap

$\|J\|$	$\|I\|$	$\|K\|$	$v(\mathcal{BD})$	$g(\mathcal{BD})$	$t(\mathcal{BD})$	$c(\mathcal{BD})$	$v(\mathcal{CBD})$	$g(\mathcal{CBD})$	$t(\mathcal{CBD})$	$c(\mathcal{CBD})$
3	50	10	15 154.79	12.88	725.45	2500	15 263.94	16.27	176.16	1583
3	50	20	15 101.62	16.81	3561.41	2500	15 147.60	19.53	430.44	1676
3	50	30	15 872.94	19.80	4650.41	2500	15 853.67	21.41	839.37	1707
3	50	40	15 872.85	21.97	1460.81	2500	15 577.95	21.13	594.17	1704
3	50	50	16 425.05	21.62	1735.06	2500	16 397.06	23.27	1400.25	1724
3	100	10	20 998.14	12.90	1887.65	5000	20 989.97	15.47	651.04	2992
3	100	20	24 918.55	15.68	7423.33	5000	24 528.78	16.22	1430.74	3179
3	100	30	24 838.63	18.04	12 476.59	5000	24 501.25	18.85	1641.99	3258
3	100	40	28 589.61	20.80	15 464.05	5000	27 610.12	19.89	1939.02	3274
3	100	50	29 087.86	20.25	9238.00	5000	29 843.76	23.04	3438.12	3475

being able to add more cuts to the $\mathcal{MP}$, but such a strategy also results in the algorithm producing better solutions. We have therefore employed this strategy in the remaining computational experiments.

7.3.3. Effect of cut elimination

As mentioned in Section 7.1.5, a cut elimination strategy was adopted to facilitate the solvability of the $\mathcal{MP}$. As this procedure may remove some necessary cuts from the $\mathcal{MP}$, one may wonder what the tradeoff is between removing some of the optimality cuts and the reduction achieved in the overall solution time as a result of this process. To shed some light on this issue, we have conducted some experiments where we compare $\mathcal{BD}_2$ with and without the cut elimination scheme (denoted henceforth by $\mathcal{CBD}$ and $\mathcal{BD}$, respectively). We additionally note that the $\mathcal{MP}$s in these experiments were solved approximately, as discussed in the previous section. The results are given in Table 6. For comparison purposes, we have imposed a common limit of 50 iterations for both algorithms and recorded the solution value, the required solution time, the gap and the total number of cuts in the $\mathcal{MP}$ at the end of the iteration limit (denoted by $v(\cdot)$, $t(\cdot)$, $g(\cdot)$ and $c(\cdot)$ in Table 6).

The results given in Table 6 demonstrate that one may indeed benefit from a cut elimination scheme as it results, for some instances, in better feasible solutions with significant time savings. Furthermore, looking at the gaps obtained by both variants, one can observe that not much is lost. In fact, for some instances, the scheme with cut elimination resulted in a lower optimality gap. Based on these results, we have employed the cut elimination scheme in all the variants of the Benders decomposition algorithm in the following computational experiments.

7.3.4. Comparison of the variants of Benders decomposition algorithm

In this section, we provide computational results on comparing the three variants of the Benders decomposition algorithm, namely $\mathcal{BD}_2$, $\mathcal{BD}_3$ and $\mathcal{BD}_4$. We note that both the cut elimination and approximate solving of $\mathcal{MP}$ strategies were implemented in all three variants. A common time limit of 3600 s (1 h) was imposed on all the variants of the algorithm and the results are reported in Table 7. Similar notation as in the previous tables were also used in this table.

The results given in Table 7 have several interesting features. First, we note that $\mathcal{BD}_3$ performs better in terms of the optimality gap, whereas $\mathcal{BD}_4$ performs better in terms of the value of the feasible solution. The latter is due to the fact that $\mathcal{BD}_4$ is able to perform more iterations under the given time limit, since a single cut is appended to the $\mathcal{MP}$ at each iteration, rendering it relatively easier to solve. This is not the case for the other two variants, where one may notice that the difficulty of solving the $\mathcal{MP}$ limits $\mathcal{BD}_2$ and $\mathcal{BD}_3$ to perform a significantly lower number of iterations as compared to $\mathcal{BD}_4$.

7.3.5. Overall comparisons

The algorithms proposed in this paper were also compared with the simplex based branch-and-cut method of CPLEX 9.01. For comparison purposes, a limit of 3600 s (1 h) was imposed on all the algorithms. For this experiment, the algorithms were tested on larger size problems than those reported in the previous tables.

Table 7
Comparison of variants of the Benders decomposition algorithm

$\|J\|$	$\|I\|$	$\|K\|$	$v(\mathscr{BD}_2)$	$i(\mathscr{BD}_2)$	$g(\mathscr{BD}_2)$	$v(\mathscr{BD}_3)$	$i(\mathscr{BD}_3)$	$g(\mathscr{BD}_3)$	$v(\mathscr{BD}_4)$	$i(\mathscr{BD}_4)$	$g(\mathscr{BD}_4)$
5	50	10	5286.82	96	4.16	5342.84	50	3.19	5379.55	257	16.13
5	50	20	8132.11	55	5.58	8266.99	31	8.26	8157.20	187	17.82
5	50	30	9176.25	55	9.62	9304.92	36	8.71	9207.12	292	20.04
5	50	40	9595.57	66	10.24	9826.51	44	10.55	9711.59	418	21.90
5	50	50	10 164.27	65	11.30	10 116.48	48	8.65	10 171.13	411	22.07
10	50	10	5500.45	27	18.97	5469.18	29	10.21	5474.09	98	34.16
10	50	20	6985.38	26	24.85	7030.05	38	14.63	7032.14	120	32.95
10	50	30	7112.00	33	24.67	7167.82	54	16.96	7034.04	209	30.05
10	50	40	7453.84	41	27.54	7567.16	72	17.24	7256.76	283	31.56
10	50	50	8089.81	39	28.76	7797.38	98	18.68	7829.31	238	32.96
5	100	10	17 105.41	55	2.86	17 156.66	26	7.23	17 057.11	244	11.98
5	100	20	20 768.97	46	2.29	21 033.70	24	11.90	20 715.85	197	3.05
5	100	30	19 820.31	45	9.14	20 083.40	34	13.00	19 668.68	270	22.81
5	100	40	21 083.87	44	11.69	22 437.04	36	13.45	21 021.90	250	14.93
5	100	50	23 026.01	46	13.83	23 331.67	39	13.71	22 884.12	405	23.94
10	100	10	14 645.10	22	30.31	14 169.32	18	17.42	14 003.84	79	35.67
10	100	20	15 470.81	27	28.66	14 865.03	28	19.78	14 664.53	94	40.19
10	100	30	17 001.94	30	29.26	15 794.65	37	20.98	15 551.80	173	38.37
10	100	40	17 464.20	30	32.22	17 289.31	50	20.86	16 964.83	190	38.28
10	100	50	17 473.94	11	43.67	17 297.37	60	20.49	17 226.43	180	39.08

The results of this experiment are reported in Table 8. The first three columns of this table are self-explanatory. The remaining columns provide, for CPLEX, the Benders decomposition algorithm (denoted by $\mathscr{BD}$) and the Lagrangean relaxation and decomposition algorithm (denoted by $\mathscr{LRD}$), the corresponding solution value that was obtained within the time limit, and the final gap between the upper and lower bounds (using the same notation as in the previous tables). For $\mathscr{BD}$ and $\mathscr{LRD}$, we also report the number of iterations under columns $i(\mathscr{BD})$ and $i(\mathscr{LRD})$, respectively.

For convenience, the best solution values and the best gaps are written in boldface characters in Table 8. The results provided in this table demonstrate that CPLEX was unable to find even a feasible solution within the given time limit for all of the instances with $\mathscr{L}_1(\mathscr{P})$. As far as the other two formulations are concerned, we observe that $\mathscr{L}_2(\mathscr{P})$ performs better than $\mathscr{L}_3(\mathscr{P})$, both in terms of the solution value and the corresponding gap, especially as the size of the instances increase.

The results given for $\mathscr{BD}$ are those obtained by using the variant $\mathscr{BD}_2$. We note that, although the other variants performed better than $\mathscr{BD}_2$ in the previous experiments for medium size problems, this was not the case for large size problems. In fact, $\mathscr{BD}_3$ was not at all successfull since the algorithm was not able to complete the first stage (i.e., solving the LP relaxation of the $\mathscr{MP}$) of the procedure within the given time limit. As for $\mathscr{BD}_4$, this variant performed poorly in terms of both the value of the feasible solution and the optimality gap. The reason for this can be explained by the fact that the aggregated cuts do not help as much as the disaggregated ones.

Looking at the figures given in Table 8, we observe that both the $\mathscr{BD}$ and $\mathscr{LRD}$ are able to find better solutions than all the three formulations solved by CPLEX 9.01 in the same amount of time, especially as the size of the instances grows. Note, however, that $\mathscr{BD}$ performs very poorly in terms of the optimality gap. The main reason for this can be explained by the difficulty of solving the $\mathscr{MP}$, which slows down the algorithm heavily as the number of iterations (and thereof the number of cuts) increases.

In contrast, we observe that $\mathscr{LRD}$ is able to do better than both CPLEX and the $\mathscr{BD}$ for large size problems. This is due to the fact that, in this algorithm, the main problem is decomposed into many small sized subproblems that are easily solvable. This enables the algorithm to perform significantly more iterations within the given time limit and therefore to produce solutions with better values.

Table 8
Comparison of all the formulations and algorithms on large instances in terms of solution values and gaps

$\|J\|$	$\|I\|$	$\|K\|$	$\mathscr{L}(\mathscr{P})$						$\mathscr{BD}$			$\mathscr{LRD}$		
			$v(\mathscr{L}_1(\mathscr{P}))$	$g(\mathscr{L}_1(\mathscr{P}))$	$v(\mathscr{L}_2(\mathscr{P}))$	$g(\mathscr{L}_2(\mathscr{P}))$	$v(\mathscr{L}_3(\mathscr{P}))$	$g(\mathscr{L}_3(\mathscr{P}))$	$v(\mathscr{BD})$	$i(\mathscr{BD})$	$g(\mathscr{BD})$	$v(\mathscr{LRD})$	$i(\mathscr{LRD})$	$g(\mathscr{LRD})$
20	200	60	N/A	N/A	**27 207.29**	38.68	27 880.14	41.26	29 664.20	18	59.82	30 739.34	6592	46.45
30	200	60	N/A	N/A	**26 988.93**	37.30	27 305.54	38.07	28 650.37	19	57.48	29 260.74	4420	42.21
40	200	60	N/A	N/A	29 427.27	40.68	**28 918.91**	39.64	29 879.46	17	59.87	30 182.08	3269	42.17
50	200	60	N/A	N/A	30 649.46	43.33	35 899.41	51.62	**28 618.52**	16	59.67	29 171.71	2582	40.46
60	200	60	N/A	N/A	29 222.75	41.05	33 985.52	49.31	**27 768.56**	17	60.58	28 483.11	2153	39.52
20	200	80	N/A	N/A	**31 413.67**	38.18	34 302.08	44.25	34 672.48	15	62.20	35 881.12	4981	46.70
30	200	80	N/A	N/A	**27 963.23**	36.50	30 659.31	42.10	30 250.65	15	65.65	30 997.00	3303	42.90
40	200	80	N/A	N/A	34 133.66	44.93	38 394.30	51.05	**32 247.84**	14	69.56	32 776.00	2476	42.65
50	200	80	N/A	N/A	33 494.56	40.79	41 940.25	52.71	**32 506.89**	16	69.23	33 131.00	1980	40.14
60	200	80	N/A	N/A	34 804.52	44.07	43 917.59	55.68	**31 664.30**	16	69.65	32 454.12	1646	40.02
20	200	100	N/A	N/A	33 808.70	39.28	36 428.73	44.31	36 503.85	17	80.20	**32 618.58**	4019	38.21
30	200	100	N/A	N/A	40 870.57	43.47	54 039.86	57.31	38 983.29	12	84.24	**32 620.78**	2728	38.30
40	200	100	N/A	N/A	35 472.44	45.10	46 975.74	58.55	33 231.23	5	89.57	**32 646.99**	2043	40.35
50	200	100	N/A	N/A	38 278.50	43.67	59 072.87	63.50	35 376.68	7	89.45	**33 303.75**	1615	35.25
60	200	100	N/A	N/A	34 597.12	36.44	48 751.34	54.90	33 223.39	6	90.27	**33 056.24**	1367	36.42

8. Conclusions and further issues

In this paper, three integer linear programming formulations and two exact algorithms for the joint problem of object placement and request routing in a content distribution network (CDN) were described. The proposed models capture the fundamental characteristics of the problem, such as the scarcity of capacity constraints and the necessity to support a certain level of quality of service (QoS). Computational experiments provided in the paper demonstrate that, through the algorithms offered in this paper, one can often obtain better solutions than those produced by a state-of-the-art integer programming solver. As a final remark, we conclude by suggesting the development of heuristic algorithms as a further research topic for this problem, since realistic instances are often rather large.

Acknowledgments

This research has benefitted from a Strategic Project Grant of the Natural Sciences and Engineering Research Council of Canada (NSERC rgpin2481). We gratefully acknowledge this support. The first author wishes to thank Dr. Iradj Ouveysi for introducing him to the topic of content distribution networks. Thanks are also due to the anonymous referees, for their comments and corrections. We are in particular indebted to one of the referees for suggesting Proposition 6 and providing comments that have helped improve the Lagrangean relaxation algorithm.

References

[1] Saroiu S, Gummadi KP, Dunn RJ, Gribble SD, Levy HM. An analysis of Internet content delivery systems. In: Proceedings of fifth symposium on operating systems design and implementation (OSDI), Boston, December 2002.
[2] The economic impacts of unacceptable web-site download speeds. Technical report, Zona Research, April 1999. Available at ⟨http://also.co.uk/e-hosting.html⟩ [accessed 24.11.2005].
[3] The Need for Speed II. Zona market bulletin, Zona Research, April 2001. Available at ⟨http://www.websiteoptimization.com/speed/1/⟩ [accessed 24.11.2005].
[4] Vakali A, Pallis G. Content delivery networks: status and trends. IEEE Internet Computing 2003;7:68–74.
[5] Akamai. ⟨http://www.akamai.com⟩ [accessed 24.11.2005].
[6] Li B, Golin MJ, Italiano GF, Deng X, Sohraby K. On the optimal placement of web proxies in the Internet. In: Proceedings of IEEE INFOCOM'99, vol. 3, New York, 1999, p. 1282–90.
[7] Qiu L, Padmanabhan VN, Voelker GM. On the placement of web server replicas. In: Proceedings of IEEE INFOCOM'01, vol. 3, 2001, p. 1587–96.
[8] Woeginger GJ. Monge strikes again: optimal placement of web proxies in the internet. Operations Research Letters 2000;27:93–6.
[9] Tang X, Xu J. QoS-aware replica placement for content distribution. IEEE Transactions on Parallel and Distributed Systems 2005;16:921–32.
[10] Leff A, Wolf JL, Yu PS. Replication algorithms in a remote caching architecture. IEEE Transactions on Parallel Distributed Systems 1993;4: 1185–204.
[11] Korupolu MR, Plaxton CG, Rajaraman R. Algorithms for hierarchical cooperative caching. Journal of Algorithms 2001;38:260–302.
[12] Cidon I, Kutten S, Soffer R. Optimal allocation of electronic content. Computer Networks 2002;40:205–18.
[13] Kangasharju J, Roberts J, Ross KW. Object replication strategies in content distribution networks. Computer Communications 2002;25: 376–83.
[14] Yang M, Fei Z. A model for replica placement in content distribution networks for multimedia applications. In: Proceedings of IEEE international conference on communications (ICC '03), vol. 1, 2003, p. 557–61.
[15] Fisher ML, Hochbaum DS. Database location in computer networks. Journal of the Association for Computing Machinery 1980;27:718–35.
[16] Pirkul H. An integer programming model for the allocation of databases in a distributed computer system. European Journal of Operational Research 1986;26:401–11.
[17] Gavish B. Optimization models for configuring distributed computer systems. IEEE Transactions on Computers 1987;C-36:773–93.
[18] Gavish B, Suh MW. Configuration of fully replicated distributed database system over wide area networks. Annals of Operations Research 1992;36:167–92.
[19] Chari K. Resource allocation and capacity assignment in distributed systems. Computers & Operations Research 1996;23:1025–41.
[20] Hakimi SL, Schmeichel EF. Locating replicas of a database on a network. Networks 1997;30:31–6.
[21] Ryoo J, Panwar SS. File distribution in networks with multimedia storage servers. Networks 2001;38:140–9.
[22] Xu J, Li B, Lee DL. Placement problems for transparent data replication proxy services. IEEE Journal on Selected Areas in Communications 2002;20:1383–98.
[23] Xuanping Z, Weidong W, Xiaopeng T, Yonghu Z. Data replication at web proxies in content distribution network. Lecture notes in computer science, vol. 2642. Xian: Springer; 2003. p. 560–9.
[24] Laoutaris N, Zissimopoulos V, Stavrakakis I. Joint object placement and node dimensioning for Internet content distribution. Information Processing Letters 2004;89:273–9.
[25] Laoutaris N, Zissimopoulos V, Stavrakakis I. On the optimization of storage capacity allocation for content distribution. Computer Networks 2005;47:409–28.

[26] Almeida JM, Eager DL, Vernon MK, Wright SJ. Minimizing delivery cost in scalable streaming content distribution systems. IEEE Transactions on Multimedia 2004;6:356–65.
[27] Nguyen TV, Safaei F, Boustead P, Chou CT. Provisioning overlay distribution networks. Computer Networks 2005;49:103–18.
[28] Bektaş T, Oğuz O, Ouveysi I. Designing cost-effective content distribution networks. Computers & Operations Research 2007;34:2436–49.
[29] Wauters T, Coppens J, De Turck F, Dhoedt B, Demeester P. Replica placement in ring based content delivery networks. Computer Communications 2006;29:3313–26.
[30] Erçetin Ö, Tassiulas L. Market-based resource allocation for content delivery in the Internet. IEEE Transactions on Computers 2003;52: 1573–85.
[31] Kangasharju J, Ross KW, Roberts J. Performance evaluation of redirection schemes in content distribution networks. Computer Communications 2001;24:207–14.
[32] Venkataramani A, Yalagandula P, Kokku R, Sharif S, Dahlin M. The potential costs and benefits of long-term prefetching for content distribution. Computer Communications 2002;25:367–75.
[33] Krishnan P, Raz D, Shavitt Y. The cache location problem. IEEE/ACM Transactions on Networking 2000;8:568–82.
[34] Datta A, Dutta K, Thomas H, VanderMeer D. World wide wait: a study of Internet scalability and cache-based approaches to alleviate it. Management Science 2003;49:1425–44.
[35] Klose A, Drexl A. Facility location models for distribution system design. European Journal of Operational Research 2005;162:4–29.
[36] Martello S, Toth P. Knapsack problems: algorithms and computer implementations. Chichester: Wiley; 1990.
[37] Padberg MW. The Boolean quadric polytope: some characteristics, facets and relatives. Mathematical Programming 1989;45:139–72.
[38] Plastria F. Formulating logical implications in combinatorial optimisation. European Journal of Operational Research 2002;140:338–53.
[39] Glover F. Improved linear integer programming formulations of nonlinear integer programs. Management Science 1975;22:455–60.
[40] Benders JF. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 1962;4:238–52.
[41] Costa AM. A survey on benders decomposition applied to fixed-charge network design problems. Computers & Operations Research 2005;32:1429–50.
[42] Held M, Wolfe P, Crowder HP. Validation of subgradient optimization. Mathematical Programming 1974;6:62–88.
[43] GT-ITM. Georgia Tech Internetwork Topology Models. ⟨http://www.cc.gatech.edu/projects/gtitm⟩ [accessed 24.11.2005].
[44] Radoslavov P, Govindan R, Estrin D. Topology informed Internet replica placement. Computer Communications 2002;25:384–92.
[45] Huang C, Abdelzaher T. Bounded-latency content distribution: feasibility and evaluation. IEEE Transactions on Computers 2005;54:1422–37.
[46] Johnson KL, Carr MS, Day JF, Kaashoek MF. The measured performance of content distribution networks. Computer Communications 2001;24:202–6.
[47] Breslau L, Cao P, Fan L, Phillips G, Shenker S. Web caching and Zipf-like distributions: evidence and implications. In: Proceedings of IEEE INFOCOM'99, vol. 1, New York, 1999, p. 126–34.
[48] Zipf GK. Human behavior and the principle of least-effort. Cambridge, MA: Addison-Wesley; 1949.
[49] Magnanti TL, Wong RT. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. Operations Research 1981;29:464–84.
[50] Van Roy TJ. A cross decomposition algorithm for capacitated facility location. Operations Research 1986;34:145–63.
[51] Wentges P. Accelerating Benders' decomposition for the capacitated facility location problem. Mathematical Methods of Operations Research 1996;44:267–90.
[52] Cordeau J-F, Soumis F, Desrosiers J. Simultaneous assignment of locomotives and cars to passenger trains. Operations Research 2001;49: 531–48.
[53] McDaniel D, Devine M. A modified Benders' partitioning algorithm for mixed integer programming. Management Science 1977;24:312–9.
[54] Pisinger D. A minimal algorithm for the 0-1 knapsack problem. Operations Research 1997;45:758–67.