Inf Technol Manag (2012) 13:1–15
DOI 10.1007/s10799-011-0107-4

Designing content distribution networks for optimal cost and performance

Jason K. Deane · Terry R. Rakes · Anurag Agarwal

Published online: 4 August 2011

Abstract One strategy for alleviating excess latency (delay) in the Internet is the caching of web content at multiple locations. This reduces the number of hops necessary to reach the desired content. This strategy is used for web content such as html pages, images, streaming video, and Internet radio. The network of servers which store this content, and the collections of objects stored on each server, is called a content distribution network (CDN). In order to optimally design a CDN, given a network topology with available server storage capacity at various points in the network, one must decide which object collections to place on each server in order to achieve performance or cost objectives. The placements must be within the storage limits of the servers and must reflect the request patterns for each collection of objects to be cached. Researchers have suggested formulations for the CDN problem which address performance by minimizing latency (the average number of hops is a commonly accepted measure of latency) from client to content, or formulations that focus on minimizing cost of storage and/or bandwidth. In this research, we develop a model which allows for the simultaneous treatment of performance and cost, present examples to illustrate the application of the model and perform a detailed designed experiment to gain insights into cost/hops tradeoff for a variety of network parameters.

Keywords Content distribution networks · Multi-objective programming · Integer programming · Fuzzy sets

J. K. Deane · T. R. Rakes
Department of Business Information Technology,
Pamplin College of Business, Virginia Tech, Blacksburg,
VA 24061, USA
e-mail: jason.deane@vt.edu

T. R. Rakes
e-mail: trakes@vt.edu

A. Agarwal (✉)
Department of Information Systems and Decision Sciences,
College of Business Administration, University of South Florida,
Sarasota, FL 34243, USA
e-mail: agarwala@sar.usf.edu

1 Introduction

The simultaneous growth in the number of Internet users and the size of the typical content delivered is leading to increased latency, and often unacceptable response times, referred to by some network professionals as the "World Wide Wait" [13]. The effects of this poor quality of service range from mild annoyance on the part of Internet users to substantial lost revenues for companies engaged in on-line commerce. Given the bursty nature of user access patterns and the inability of the Internet to scale, new solutions may be necessary to help alleviate this congestion. One solution for this problem is *caching*, where content can be distributed or replicated, and stored at, multiple locations around the Internet [13, 14, 22, 23]. This puts the content closer to the requestor, minimizing the number of hops, and therefore the latency, when content is requested. A hop is the trip a data packet takes from one router, proxy server, or intermediate point to another in the network. Thus, a path followed from a point of object request to the copy of the content will be made up of multiple hops, depending on the physical network path to the content. Latency, or network transit time, is directly related to hops because each packet may experience queuing delay at each intermediate point as the packet is prepared for transmission, as well as the delay required for routing determination. Also, each time another hop is required, more transmission distance is involved, which increases latency. While hops is not a perfect predictor for latency, Kangasharju et al. [14] showed that for a large study of network delivery times, overall latency was closely related to the number of hops. Thus, hops are used as a common proxy for latency [13, 14].

The concept of caching has long been employed in IP address resolution, reducing the time to convert a symbolic domain name to an IP address [12]. The level of caching can vary from a single object such as one symbolic name/IP address pair to an entire domain hierarchy. In the same way that caching has been applied to domain names, the concept can be used to store files, html pages, fragments of pages, image files, streaming video, Internet radio, or any other content object desired by users [22]. The application of this caching concept has lead to the emergence of a structure known as content distribution networks (CDNs) [13, 14, 22, 23]. Assuming that every web object exists in some original location known as an origin server, the task in constructing a CDN is to determine if an object should be replicated, and if so, where it should be located throughout the network [14]. This simultaneous determination of replicated content and location is done subject to goals such as minimizing the average or total number of hops necessary to retrieve the content given a particular pattern of users' requests for content, minimizing content storage cost and the bandwidth cost for content transfer, or some other similar performance measure.

In regard to the business model for a CDN, at the lowest level of the hierarchy we have the *end-client*, who desires some type of service over the Internet. This end-client is willing to pay a *content provider* for the service, and this payment aggregated over a large group of end-clients represents the total income for the content provider. For example, an end-client may desire to download an mp3 file over the Internet, and is willing to pay an Internet music provider for that file. However, in order for the music provider to make his large base of end-clients happy with the service, he or she must have ample copies of the large database of potential mp3 files distributed throughout the network so that downloads will be fast (in other words, few hops will be necessary to reach a copy of the music file).

It would be rare for a content provider to have their own network of storage servers, so the content provider turns to a CDN provider whose business purpose is to cache and distribute copies of web objects on behalf of the content providers. Their income is derived from the fees which content providers pay in order to have their content cached and delivered, and these fees are based on where the content is stored, the size of the content, and the monthly bandwidth used to deliver the content. The content will typically be large files or databases, as in the case of Internet music. While any end-client request may result in a transfer of only a few megabytes of data (a few mp3 files), the service provider must store the entire listing of potential songs, which could be tens or hundreds of gigabytes. The same is true of on-line video, archived news files, or any other type of content from which end-clients may choose a sub-set to download. We refer to these large groups of similar objects as an object collection.

It is important to note that object collections do not have to be homogenous, nor do all of the contained objects need to have a similar popularity level. According to Pallis and Vakali [22], a typical practice is to group web content based on either correlation (object similarity) or access frequency, and then replicate objects in units of content clusters. In our analysis, we will only be concerned with the total request volume routed to a collection in aggregate, and with the average size of a download request from the collection, which together define the bandwidth needed for object transfer. Thus, most of the request volume could be for a few of the files in the collection, or the files could be equally popular. Whether the average size of a download request from a collection is dominated by a few large but frequently requested items within the collection, or by numerous similarly sized items, is irrelevant. Likewise, the collection could be a mixture of music, video, news files, or any other downloadable items. The content provider has complete discretion in how to do these groupings. From the standpoint of our analysis, the grouping logic is transparent and the driving forces are the request volume and the average download size.

In regard to the caching plan, this is a decision problem which involves both the content provider and the CDN provider. Typically, both parties will cooperate to develop a service-level agreement (SLA) which outlines the expected performance and costs associated with their business arrangement. The content provider wants an SLA which assures that end-clients are not kept waiting too long for downloads, and yet ensures that the storage and delivery fees which they must pay are low enough that they can charge reasonable fees to the end-client and still make a profit. While the CDN provider seeks to maximize revenue, they also want the content provider to be happy with the SLA and its implications, as this is a very competitive business where clients can easily switch to other CDN providers. To arrive at the best plan, the content provider will provide information about their collections and the pattern of object requests. The CDN provider will merge this information with the information about their network structure and pricing policies and then solve the decision problem which determines optimal caching locations given different combinations of preferences for cost versus performance. Thus, it is the CDN provider that generates solutions or plans, because only the CDN provider has all of the necessary information to determine these solutions. There will likely be back and forth negotiations or discussions between both parties until a stable set of solutions is reached, at which point the CDN provider will present to the content provider a set of possible caching plans along with the details about resulting cost and performance. It is then up to the content provider to decide which plan best suits their needs in terms of the balance between cost and performance.

There are numerous variations of the caching problem, involving what, where, and when to cache [13]. Because of the real costs (storage, access, etc.) associated with caching, not all content should be cached. The challenge of determining what to cache has been addressed, but is still an open research area. As for where to store, content may be cached at the client browser, at a proxy server somewhere between the client and the website, or at the host website server itself. And once content has been cached, there are important questions as to when to place items in a cache and when to remove them [13]. While numerous models have been developed for addressing where to cache based on either cost or latency, none to the best of our knowledge have attempted to model CDN design which strikes a balance among multiple important conflicting objectives such as latency and cost. In this research, we are concerned with optimal strategies for distributing content objects among various proxy sites in order to address these multiple objectives, assuming that the decisions on which objects to cache and when to cache them have been made.

This paper is organized as follows. In the next section, we review some existing formulations for the CDN location problem. In Section 3, we present a model for extending this problem to a multiple objective format which can address both cost and latency criteria. Section 4 illustrates the application of fuzzy multi-objective programming as a means to solve the formulation. Section 5 presents an example to illustrate the application of the model in a decision process. In Section 6, we give an example to show how pricing decisions might be evaluated using the model. Section 7 presents results of an experiment over a broader range of costs so that we may offer more general insights, and in Section 8 we close with some overall conclusions.

2 Previous research

Over the last 10 years, several studies have presented models for solving different variations of the CDN location problem. These have involved how to locate proxy servers, where to cache content, how to size caches, or optimal routing from clients to proxies. For example, Kangasharju et al. [14] developed a model for replicating objects in CDN servers in order to minimize the average travel time (hops) to the desired content. They showed that the model is NP-Complete, and then suggested four basic heuristics for determining possible placements (others, such as Qiu [25], have also suggested heuristics). Yang and Fei [32] studied a similar object placement problem with limited storage capacity. Cidon et al. [9] developed a distributed content algorithm for minimizing storage and communication cost. Datta et al. [13] also developed a model for minimizing the total cost (number of hops) to retrieve cached objects. They described their model as a simplistic starting point with the purpose of stimulating further research in the area. Kumar [15] looked at the performance of a caching policy which considers objects held by neighbor proxies in an effort to reduce redundancies. Kumar and Norris [16] proposed a new proxy-level web caching mechanism that exploits historical request patterns and deviations from normal usage patterns in order to determine what to cache.

Many of the papers in CDN design have attempted to study different aspects of the design problem simultaneously. Bektas et al. [3, 4] provided a good review of recent efforts in CDN design. Some of the papers they reviewed include [2, 17, 18, 21, 26, 29, 30]. We summarize their review as follows. Ryoo and Panwar [26] modeled the problem of distributing different types of content to different servers while determining necessary communication link capacities and server size. Xu et al. [29] studied the problem of determining the optimal number and location of proxy servers along with placement of object replicas on the servers. Xuanping et al. [30] examined proxy server placement and object replication subject to a budget constraint where each client is assigned to its closest proxy. Laoutaris et al. [17] studied optimal location of the objects along with the capacity dimensioning of the proxies. Laoutaris et al. [18] studied the storage capacity allocation problem which determines the optimal proxy location, the capacity of each proxy and the objects that should be cached in each proxy given fixed client assignments. Nguyen et al. [21] assumed that the total capacity of proxies is greater than the total demand and allowed client requests to be fractionally served by different proxies. They developed an integer linear programming formulation with a solution approach based on Lagrangean relaxation. Almeida et al. [2] developed an optimization model and several heuristics for the problem of jointly routing requests and placing proxies such that total server and network delivery costs are minimized.

Some of the most comprehensive models for CDN object placement have been those of Bektas et al. [3, 4]. Bektas et al. [4] developed a model for jointly locating proxy servers on a set of potential nodes, replicating content on the nodes, and routing the request for content to a suitable proxy server so that the total cost of distribution is minimized. They proposed a linearization scheme for the non-linear model and examined solution possibilities using both an optimization approach and a heuristic. In Bektas et al. [3], they assumed a fixed number of proxy servers and developed a model to replicate content and determine the request routing subject to a QoS constraint. They proposed two solution techniques based on Benders decomposition and Lagrangean relaxation.

While both of these are comprehensive treatments of object placement with respect to cost, neither attempt to include the minimization of latency, which has been shown to be another very important criteria. By concentrating only on cost, they were able to focus on the *link layer* which is the logical layer that sits on top of the *physical layer* or actual network. While the link layer can be viewed as a fully-connected mesh network which provides direct links (one-hop paths) that connect every pair of nodes in the network, the physical network may require many hops between nodes to fetch an object from a remote location. Therefore, if latency is an important criteria, we must consider the physical network topology in our treatment. In the next section, we discuss issues related to addressing both cost and latency in a CDN and present a model which allows for these multiple criteria.

3 The multiobjective CDN model for cost and latency

Latency is a major concern for providers of Internet content. Users have a very low tolerance for delay, often leading to abandoned web site connection attempts and a resulting loss in sales. Number of hops is a common surrogate for latency, as both routing and queuing delay is introduced each time we begin another leg in a web journey.

While latency is undoubtedly important in determining where to cache web objects, cost is also extremely important. Currently, CDN prices are high, but are decreasing. The average cost per gigabyte of streaming video transferred in 2004 was $1.75, and the average price to deliver a gigabyte of Internet radio was $1 [22]. These are the costs charged back by the CDN providers to the owners of the web content (publishers of the webpage). Part of this cost is likely due to the newness of the technology and the relatively small number of available providers. For example, Akamai Technologies owns approximately 80% of the overall CDN market, with more than 12,000 servers over 1,000 networks in 62 countries [22]. While costs are expected to decrease over time in parallel with decreasing bandwidth costs and increased competition, the increasing demand for CDN services will continue to make cost an important consideration. In 2004, more than 3,000 companies were using CDN services, spending more than $20 million per month, with annual growth rates for streaming video and Internet radio predicted to approach 40% [22].

We present a formulation which directly treats both latency and cost. As in Bektas et al. [4], we assume that proxy servers are fixed. As in Kangasharju et al. [14], we assume that each node in the network is an Internet Autonomous System (AS) or proxy server which represents a server location with finite storage capacity. Thus, the original source location for a web object (origin server) would be a possible AS, as would any proxy servers which have been added for distributed versions of the content. If origin servers are present, we can include them in the formulation by hard-coding the appropriate location/object variables. If the AS is acting as the originating point for a client request, we denote it as node i, where $i = 1, 2,\ldots I$, and the location of a server which fields a request is denoted as node j, $j = 1, 2,\ldots J$, where I is the number of originating points and J is the number of cache locations. Let k represent a collection of objects where $k = 1, 2,\ldots K$ (for example, $k = 1$ might be a collection of mp3s, $k = 2$ a collection of videos, etc.). In terms of cost, we follow the lead of numerous other researchers and include both the cost of object storage and the cost of object transfer. Bektas et al. [4] treated storage cost as a fixed *instantiation cost*. We calculate storage cost based on object collection size by using a cost per gigabyte stored.

Object transfer cost is essentially a bandwidth usage cost which is charged based on the volume (gigabytes) of data transferred. We assume cooperative caching where any proxy can contact another proxy in order to fulfill an object request such that our transfer cost is based on usage of "infrastructure bandwidth" or "in-CDN bandwidth" between proxies. Edge bandwidth is a sunk cost since all requests from clients must initially be transmitted to the client's assigned proxy and are therefore un-avoidable. Thus, we assume no bandwidth cost if the request is served locally. Also, we assume as did Bektas et al. [4], with no loss of generality, that the bandwidth price is charged for transmission at the link layer such that it is independent of the number of hops. This is a common pricing mechanism. However, if this is not true, all that is required is that bandwidth usage cost must be multiplied by distance to the object collection in the objective function so that hops are reflected in the pricing.

In regard to bandwidth pricing, it is common for CDN providers to offer quantity discounts (also referred to as volume-based pricing), especially the major CDN providers who focus on global delivery. The data in Table 1 illustrates the current trend in the market for the 1st quarter of 2009 for video delivery. Based on the total volume a customer has served on its behalf by the CDN provider, prices in early 2009 could vary from as high as $.52 per gigabyte at a low volume to $.025 per gigabyte at a high volume. While 500 TB per month of volume would be expected from a very large CDN customer, there are "more

Table 1 Typical infrastructure bandwidth charges in 2009

Volume (terabytes per month)	High quote (cost per gigabyte)	Low quote (cost per gigabyte)
50	\$0.52	\$0.45
100	\$0.40	\$0.25
250	\$0.20	\$0.10
500	\$0.10	\$0.025

Source: http://www.blog.streamingmedia.com/.a/6a00d834518e1c69e201156fc68141970c-popup [accessed 10-19-09]

than a handful" of these customers in the market [27]. With the constant decline in bandwidth prices, we could expect that in the future even better prices, at lower trigger levels, than those shown in Table 1 will be available. With these sorts of pricing variations available, a model which strives to minimize cost should certainly include volume-based pricing and the effect it could have on the hops/cost tradeoffs.

Before developing a model to accommodate multiple criteria, it is important to ascertain the best mode of criteria representation. It is possible to model this problem as a cost minimization problem with performance bounds as constraints. It is also possible to model it with a performance maximization objective subject to a budget constraint. However, we do not believe that either of these approaches is as powerful as a true multiobjective formulation. The purpose of treating both cost and performance as objectives is to avoid the necessity to create bounds and to allow the decision maker total flexibility in arriving at the best compromise between cost and performance. Once a bound is selected, the solution domain is restricted and cannot consider tradeoffs below (above) that bound. For example, if it were possible to increase performance (decrease latency) considerably with a negligible increase in cost, that option would not be open to us with a bounded formulation assuming the performance constraint is binding. In other words, within the bounded range tradeoffs between cost and performance would be determined solely on the marginal substitution rates dictated by the mathematical structure of the problem as opposed to the decision maker's preferences, and outside the bounded range no tradeoffs of any kind are possible. While sensitivity analysis around the bound can tell us the nature of marginal tradeoffs between the bounded value and the objective, once again these are based on the marginal substitution rates between the variables and does nothing to incorporate the decision maker's preference structure for the tradeoff. The multiobjective format allows us to specify the nature of the tradeoff preferences and to compare the solutions which would be best for each of those preference structures. The fact that the multiobjective method only examines non-dominated solutions means we lose nothing in comparison to the single objective bounded approach, but gain much in terms of ability to accommodate decision maker preferences in our solution process. Because it is non-dominated, the efficient frontier comprised by the multiobjective corner points will contain the solutions which the single objective model would provide, plus numerous other alternatives. Referring to a set of real-world public policy problems, Cohon [11] states "The imposition of a single-objective approach on such problems is overly restrictive and unrealistic. Multiobjective analysis allows several noncommensurable effects to be treated without artificially combining them. This is clearly a significant improvement in analytical capability." He also states that "It is generally true, however, that multiobjective approaches will indicate to decision makers a range of choice larger than the one 'optimal' project identified by single-objective methods. This larger range of choice is due to the articulation of value judgments regarding the objectives by decision makers in the project selection phase rather than during analysis. This aspect of multiobjective programming and planning is perceived as an advantage. A general rule for decision making which is assumed here is that more information (carefully presented) is better than less information. … Informed, rational decision making requires knowledge of the full range of possibilities. This can be provided by multiobjective analysis." Consequently, we suggest a multiobjective model which incorporates cost objectives (including volume-based pricing) and latency objectives as well as the other features discussed previously.

We use the following additional notation:

Parameters

r_{ik} = request volume originating at node i for any item in collection k (number of requests per month)
d_{ij} = distance from i to j (hops)
b_k = size of object collection k (gigabytes)
o_k = average size of a download request from collection k (gigabytes)
s_j = storage capacity at j (gigabytes)
c_j = cost to store cached content at j (dollars per gigabyte stored per month)
l = index of volume price levels based on bandwidth usage, $l = 1, 2,\ldots, L$, where L is the number of available price levels
q_l = the transferred bandwidth in gigabytes necessary to receive price level l
f_l = the price per gigabyte transferred at price level l
M = a very large positive constant

Variables

X_{ijk} = 1 if the demand originating at node i for an object in collection k is routed to a cache at node j, 0 otherwise
Y_{jk} = 1 if collection k is cached at j, 0 otherwise
V_l = 1 if the volume of bytes transferred $>q_l$ and $\leq q_{l+1}$, 0 otherwise

This yields the following multiobjective model:

$$\text{P1}: \quad \text{Min}\, Z_1 = \sum_i \sum_j \sum_k r_{ik} d_{ij} X_{ijk} \tag{1}$$

$$\text{Min}\, Z_2 = \sum_j \sum_k c_j b_k Y_{jk} + \sum_l \sum_i \sum_j \sum_k f_l V_l r_{ik} o_k X_{ijk}$$

$$\text{where } i \neq j \tag{2}$$

s.t.

$$\sum_i \sum_j \sum_k r_{ik} o_k X_{ijk} - q_l V_l \geq 0 \quad \forall\, l \text{ where } i \neq j \tag{3}$$

$$\sum_l V_l = 1 \tag{4}$$

$$\sum_k b_k Y_{jk} \leq s_j \quad \forall j \tag{5}$$

$$r_{ik} \left(\sum_j X_{ijk} - 1 \right) \geq 0 \quad \forall\, i, k \tag{6}$$

$$\sum_i X_{ijk} \leq M Y_{jk} \quad \forall j, k \tag{7}$$

$$\sum_i X_{ijk} + M(1 - Y_{jk}) \geq 1 \quad \forall j, k \tag{8}$$

The first objective function Z_1 minimizes the total number of hops per month in the network by multiplying the request volume r_{ik} which originates at each proxy by the distance d_{ij} to the placements suggested by X_{ijk} for each object collection k. The second objective function Z_2 minimizes total cost. The first term is the total cost of object storage across all j servers (it is worth noting again here that b_k is the size of collection k which is a collection of many objects and may be quite large, while o_k is the size of an average download request from collection k). The second term is the delivery (transmission) cost. Summing over all k object collections, the product of the request volume r_{ik} times the typical object size o_k represents the number of gigabytes that will have to be transferred via any object/route placement X_{ijk}. If $i = j$, then the delivery is local and there are no bandwidth charges. The second term in Z_2 also reflects volume pricing by multiplying the volume price f_l times the indicator variable V_l which indicates where we fall in the volume schedule. We sum over all possible volume levels (only one V_l will be non-zero) and multiply by the total transmitted volume.

Equations 3 and 4 ensure that only one volume pricing level indicator variable V_l is non-zero and that the price being charged for transmission is correct based on the total bandwidth consumed. The next constraint (Eq. 5) ensures that the total size of all object collections to be stored at server j does not exceed capacity at that server. Equation 6 ensures that all requests for objects are honored by making at least one of the flow paths from the request origin to the object collection equal to 1. When a node has no requests for objects from within a particular object collection, then no outgoing path to a copy of the collection should be activated. That is, when $r_{ik} = 0$, then the constraint becomes redundant as both the left and right sides become 0.

The last constraint sets (Eqs. 7, 8) are linking equations which stipulate that a copy of the object collection must exist at any node which is the receiver of a request for an object from within that collection. According to Eq. 7, if $\sum_i X_{ijk} > 0$ then Y_{jk} must be 1 and if $\sum_i X_{ijk} = 0$ then Y_{jk} is unrestricted. In Eq. 8, if $\sum_i X_{ijk} = 0$ then Y_{jk} must be 0 and if $\sum_i X_{ijk} > 0$ then Y_{jk} is unrestricted. Thus, the combination of Eqs. 7 and 8 ensure that if any object request flow comes into a node there must be a copy of the object collection at that node, and if no flow comes in, there will not be a copy.

In the second term of Z_2, the product of V_l and X_{ijk} creates a quadratic expression. To simplify the computational aspects of the model, we substitute $\lambda_{ijkl} = V_l X_{ijk} \quad \forall i, j, k, l$ in Z_2 and utilize the linearization suggested by Plastria [24] to obtain:

$$\lambda_{ijkl} - V_l \leq 0 \quad \forall i, j, k, l \tag{9}$$

$$\lambda_{ijkl} - X_{ijk} \leq 0 \quad \forall i, j, k, l \tag{10}$$

$$V_l + X_{ijk} - \lambda_{ijkl} \leq 1 \quad \forall i, j, k, l \tag{11}$$

$$\lambda_{ijkl} \in \{0, 1\} \quad \forall i, j, k, l \tag{12}$$

With this linearization, objective function Z_2 reduces to:

$$\text{Min}\, Z_2 = \sum_j \sum_k c_j b_k Y_{jk} + \sum_l \sum_i \sum_j \sum_k f_l r_{ik} o_k \lambda_{ijkl}$$

$$\text{where } i \neq j \tag{13}$$

We can now construct the linear version of the multiobjective CDN problem with volume-based pricing as:

$$\text{P2}: \quad \text{Min}\, Z_1 = \sum_i \sum_j \sum_k r_{ik} d_{ij} X_{ijk} \tag{14}$$

$$\text{Min}\, Z_2 = \sum_j \sum_k c_j b_k Y_{jk} + \sum_l \sum_i \sum_j \sum_k f_l r_{ik} o_k \lambda_{ijkl} \tag{15}$$

s.t.

Equations $(3) - (12)$

Obviously, Eqs. 14 and 15 are Eqs. 1 and 13, but have been renumbered for completeness.

4 Solving the multiobjective CDN model

Clearly, the two objective functions in P2 are at odds. In order to reduce the number of hops necessary to reach content as expressed in Z_1, content needs to be cached at many locations. This may also reduce content delivery charges which are based on consumed bandwidth. However, caching at many locations increases the total storage and administrative costs in Z_2.

Because of the incommensurate units of the two objectives (hops vs. dollars) it is not possible to combine them into a single objective in the traditional sense. Past approaches for developing compromise solutions have ranged from goal programming with an imposed preemptive priority structure to constructing an "efficient frontier" which serves as a surface of trade-offs that may be examined by the decision maker. In order to facilitate analysis of the model, we will use another established procedure: fuzzy compromise programming.

Fuzzy compromise programming was first suggested by Zimmerman [33–36] and has since been applied by numerous authors, including Bit [5–7], Chang et al. [8], Coffin and Taylor [10], Adb El-Wahed and Lee [1], Li and Lai [20], and Topaloglu and Selim [28]. By obtaining a marginal evaluation of each objective Z_g (a mapping which tells us to what degree the decision $x \in X$ makes the objective Z_g close to its aspiration level L_g) for $g = 1,\ldots G$ where G is the total number of objective functions, we can aggregate these marginal evaluations to find a compromise solution at which the global evaluation of the synthetic membership degree of optimum for all objectives is maximum [20]. We follow the basic outline of Li and Lai [20] in applying the fuzzy compromise principle. The following steps are adapted from their approach.

For each minimization objective Z_g, a value U_g can be viewed as the highest acceptable level of achievement, and L_g can be viewed as the aspired level of achievement. Hopefully, these values could be supplied by a domain expert or decision maker such that they reflect the preferences of the decision maker. Alternatively, these values can be determined by solving each of the $2g$ single-objective problems to obtain

$$\begin{aligned} U_g &= \max Z_g(x) \quad x \in X,\ g = 1,\ldots,G \\ L_g &= \min Z_g(x) \end{aligned} \tag{16}$$

Once U_g and L_g have been determined, the marginal evaluation for each objective can be found by:

$$\phi_g(x) = \begin{cases} 1 & \text{if} \quad Z_g(x) \le L_g \\ \frac{Z_g(x)-U_g}{L_g-U_g} & \text{if} \quad L_g \le Z_g(x) \le U_g \\ 0 & \text{if} \quad Z_g(x) \ge U_g \end{cases} \tag{17}$$

where $\varphi_g(x)$ is the fuzzy membership value of x on the interval [0,1] for objective function g.

Given the marginal evaluations $\varphi_g(x)$ for a given decision x, the final task is to determine a global subjective evaluation $\mu(x)$ with respect to all of the objectives. The optimal compromise solution is the decision at which the global subjective evaluation value is the maximum. The preferences of the decision-maker are reflected in the weights applied to each marginal evaluation in arriving at $\mu(x)$. Specifically, for a set of preference weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_g)$ which are normalized to sum to one, we are interested in an aggregation operator $\boldsymbol{\Phi}_{\mathbf{w}}:[0,1]^G \to [0,1]$ such that:

$$\mu(x) = \Phi_w(\varphi_1(x), \varphi_2(x), \ldots, \varphi_g(x)) \tag{18}$$

Several aggregation operators have been suggested in the literature. Li and Lai [20] discussed a family of aggregation operators called the weighted root-mean power operators, and discussed five variations within this family. The two most widely used operators from within this family are the weighted arithmetic mean and the conjunctive mean. The conjunctive mean operator, sometimes referred to as the "min" operator because it seeks to choose the maximum from among the minimum of the marginal evaluations of the objective functions, has been widely applied, but has disadvantages. Li and Lai [20], in their summary of aggregation operators, point out that the "min" operator is non-compensatory [31]. That is, the solution from "min" focuses on the worst situation and cannot be compensated by other members that are very good. Also, Lee and Li [19] showed that the "min" operator does not guarantee non-dominated solutions. For these reasons, we will utilize the weighted arithmetic mean operator whereby the decision-maker can see the results obtained from different preference weights and choose the solution which is most preferred.

Applying the fuzzy compromise programming approach to our multi-objective CDN model P2, for $x \in X$ we obtain the following:

$$\text{P3}: \quad \text{Max}\, \mu(x) = w_1\varphi_1(x) + w_2\varphi_2(x) \tag{19}$$

s.t.

Equations (3–12)

$U_1 = \max Z_1(x), L_1 = \min Z_1(x)$ where $Z_1(x)$ is equation (14)

$U_2 = \max Z_2(x), L_2 = \min Z_2(x)$ where $Z_2(x)$ is equation (15)

$$\phi_1(x) = \begin{cases} 1 & \text{if} \quad Z_1(x) \le L_1 \\ \frac{Z_1(x)-U_1}{L_1-U_1} & \text{if} \quad L_1 \le Z_1(x) \le U_1 \\ 0 & \text{if} \quad Z_1(x) \ge U_1 \end{cases}$$

$$\phi_2(x) = \begin{cases} 1 & \text{if} \quad Z_2(x) \le L_2 \\ \frac{Z_2(x)-U_2}{L_2-U_2} & \text{if} \quad L_2 \le Z_2(x) \le U_2 \\ 0 & \text{if} \quad Z_2(x) \ge U_2 \end{cases}$$

At this point, we should reiterate that solving P3 is a decision problem for the CDN provider, and choosing from among the eventual candidate solutions is a task for the content provider. Because the complete set of cache locations, and therefore the possible instances of decision variables X_{ijk} and Y_{jk}, are known only to the CDN provider, this model can only be solved by the CDN provider. In the next section, we give an example of the process by which the CDN provider can develop and present the set of best solutions to the content provider.

5 A numerical example

In order to illustrate application of the model, we present a case example. We assume ten proxy servers which can receive and service end-client requests or transfer the request to another server which has a copy. This is substantially larger than the three to five proxies used in illustrative examples by Bektas et al. [4] and Pallis and Vakali [22]. Since thousands of end-clients could be attached to each proxy, such a network could represent a sizable subset of a large CDN network, or could represent the entire network for a smaller CDN provider. Because of the large number of connected end-clients, the aggregate object request volume at each proxy can be very large. We are not concerned with the nature of the connectivity of the end-clients but with the resulting request volume at each proxy and how to serve those requests by providing cached copies of content.

We further assume five large object collections to be cached, and three levels of volume-based bandwidth pricing. Table 2 details the parameters for the example. While this case example is hypothetical, all of the parameters in Table 2 are based on typical values from the literature. For example, to arrive at the storage cost of $3 to $8 per GB per month, we consider what the CDN would charge the content provider, which is based on the cost of physical storage media, cost of maintenance and operations personnel, etc., and an appropriate profit margin. While we often think of storage as a cheap commodity because of the desktop as a frame of reference, enterprise-class storage devices with their power supplies, cooling fans, housing racks, and built-in redundancy are obviously much more expensive. Because enterprise-level mass storage devices with multi-terabyte capacities can run into the hundreds of thousands of dollars, we typically see an industry average of $25 to $75 per purchased gigabyte of redundant arrays of independent disks (RAID). Assuming a useful life of 2 years, the cost to purchase enough memory to provide a gigabyte of storage for a 30 day period using these industry averages would be between $1.05 and $3.15. Using these costs for procuring the media and adding a reasonable amount for overhead and profit, storage costs per gigabyte per month can easily reach $3 to $8. For many of the other parameters which have multiple values, such as the request volumes for collections which originate at many different locations (r_{ik}), we have used uniform distributions to generate the multiple values. Because the values of these parameters could be any value over some typical range, we use the uniform distribution which is a model of random (equally likely) outcomes.

Table 2 Parameters for the case example

Parameter	Value
# of nodes i and j	10
# of caches k	5
# of volume price levels l	3
r_{ik}	U(1,000, 4,000) per month*
d_{ij}	U(3, 8)*
b_k	U(1,000, 3,000) GB*
o_k	U(.1, 1) GB*
s_j	U(2,000, 5,000) GB*
c_j	U($3,$8) per GB per month*
q_l	(0, 50 TB, 100 TB) per month
f_l	($.40, $.20, $.05) per GB

* U(a, b) indicates that the parameters are uniformly distributed over the range a to b

Using the values from Table 2 to parameterize the model, we arrive at a model instance with 4,668 variables and 2,057 constraints. For a model of this size, translation into a solvable format can be a formidable task. To make this task easier, we developed a code generator written in Visual Basic which accepts the parameters as input and produces the model code in CPLEX format. Solution times in CPLEX for the full set of following problems ranged from 1.09 to 447.64 seconds on a typical desktop machine.

In order to apply fuzzy multi-objective analysis, we first determine the upper and lower bounds for the two objectives by solving the model to determine the maximum and minimum values possible for each objective. We obtain bounds on hops (Z_1) of $U_1 = 598{,}640$ and $L_1 = 218{,}920$. Likewise, we obtain bounds on cost (Z_2) of $U_2 = \$200{,}964.80$ and $L_2 = \$40{,}851.60$. Using these bounds and the parameters from Table 2, we obtain the solutions in Table 3 for various combinations of the weights (w_1, w_2). In addition to columns for the weights and values of the two objectives Z_1 and Z_2, Table 3 also has columns which indicate which volume-based pricing level is in effect for each solution (V_1, V_2, or V_3) and the values of the fuzzy membership functions represented by each solution (φ_1 and φ_2). From Table 3, it is easy to see the nature of the tradeoffs as the decision maker's emphasis switches between hops and cost. For weights of (0,1) where cost is emphasized and hops are given no weight, the lower bound cost is achieved along with the upper bound value for hops by using fewer cached copies of the collections resulting in lower storage costs, and allowing the forwarded request bandwidth volume to rise into the second pricing tier (above 50 TB) where we get more favorable bandwidth rates. When the weights are (1,0), we see the

Table 3 Results for the case example when $Q_2 = 50$ TB

(w_1, w_2)	Z_1 (hops in 1,000s)	Z_2 (in '000s)	V_1	V_2	V_3	φ_1	φ_2
(0, 1)	598.64	40.85	0	1	0	0	1
(.1, .9)	579.44	41.08	0	1	0	.0506	.9986
(.2, .8)	496.18	48.58	0	1	0	.2698	.9517
(.3, .7)	453.88	55.66	0	1	0	.3812	.9075
(.4, .6)	426.36	62.21	0	1	0	.4537	.8666
(.5, .5)	382.98	76.22	0	1	0	.5679	.7791
(.6, .4)	291.74	124.49	1	0	0	.8082	.4776
(.7, .3)	239.22	165.80	1	0	0	.9465	.2196
(.8, .2)	221.08	192.27	1	0	0	.9943	.0543
(.9, .1)	221.08	192.27	1	0	0	.9493	.0543
(1, 0)	218.92	200.96	1	0	0	1	0

Table 4 Results for the case example when $Q_2 = 70$ TB

(w_1, w_2)	Z_1 (hops in 1,000s)	Z_2 (in '000s)	V_1	V_2	V_3	φ_1	φ_2
(0, 1)	886.22	46.70	0	0	1	0	1
(.1, .9)	673.78	47.31	0	1	0	.3184	.9960
(.2, .8)	647.38	48.50	0	1	0	.3579	.9884
(.3, .7)	496.18	59.10	1	0	0	.5845	.9197
(.4, .6)	480.70	61.22	1	0	0	.6077	.9059
(.5, .5)	437.04	68.94	1	0	0	.6731	.8559
(.6, .4)	363.86	89.75	1	0	0	.7828	.7209
(.7, .3)	312.58	111.41	1	0	0	.8596	.5803
(.8, .2)	248.16	157.15	1	0	0	.9562	.2840
(.9, .1)	221.08	192.27	1	0	0	.9968	.0563
(1, 0)	218.92	200.97	1	0	0	1	0

lowest possible hops but at the largest possible cost. In this case, more copies are cached to reduce the necessary hops, resulting in more storage cost and a bandwidth usage in the most expensive range (below 50 TB) for those requests which must be forwarded. For all of the compromise solutions in between, we see solutions which achieve more of a balance for the membership functions. For example, at (.5,.5) it is possible to choose a solution which represents a relative achievement of .5679 on a scale of 0–1 for our goal of making hops as low as possible, and a relative achievement of .7791 for our goal of making cost as low as possible. Of course, it is up to the decision maker to decide among the comprise solutions as to which represents the best combined achievement from their perspective.

To further examine the nature of the tradeoffs in the presence of volume-based pricing, we run a second experiment using a higher value for the volume discount point Q_2 of 70 TB. Table 4 gives the results for this scenario.

With a higher threshold for the volume pricing discount, the (0,1) solution pushes the request volume to the highest level (above 100 TB) in order to achieve the lowest cost. This high request volume, and consequently low level of cached copies, pushes the number of hops to over 886,000. As we increase w_1 and lower w_2 gradually putting more emphasis on hops, the request volume decreases into the second tier of volume pricing, and then at (.3, .7) decreases into the first or lowest tier of volume pricing. As in the first scenario, the compromise solutions offer the decision maker wide latitude in deciding which solution best meets their preferences for hops versus cost.

As a final illustration, we graph the results of Tables 3 and 4 to show the tradeoffs. Figure 1 is a graph of hops for the volume discount levels of 50 and 70 TB plotted against w_1 (we could just as easily have graphed against w_2, and the graphs would be reversed starting at the lowest level of hops and then increasing). This plot illustrates the considerable effect that volume pricing thresholds have when hops are given a weight of 0 (a difference of almost 300,000 hops). When hops are given a weight of one, volume pricing thresholds have no effect as our efforts shift to minimizing hops regardless of the pricing level and cost. In Fig. 2, we graph cost for the volume discount levels of 50 and 70 TB again plotted against w_1. Here we see that as the weight on hops (w_1) increases up to about .7 or .8 causing cost to rise, we are able to achieve a lower cost for the 70 TB solution. Once again, this is because cost includes not only the cost of the bandwidth charges but also the storage cost, and by pushing request volume up over 70 TB in order to get the pricing discount we are able to cache fewer copies. For this case example, the 50 TB bandwidth volume discount level results in a tighter band of values for Z_1 and Z_2 in comparison to the 70 TB discount level. Stated differently, with a 50 TB discount policy we can achieve lower hops at virtually every level of preference weights but with a higher cost in relation to the 70 TB discount policy.

Obviously, these results are indicative of the parameters used in this illustrative case. While the chosen parameters are based on realistic values from the CDN literature, we do not claim that this set of decision outcomes would be either typical or appropriate to any specific company. Rather, our goal is to illustrate the decision process and the types of decision problems which are supported by this process. While other parameterizations might show enhanced (or diminished) effects of some aspects in relation to this example, our analysis clearly establishes the nature of the cost/performance tradeoff and the effects of volume-based pricing on these tradeoffs. Furthermore, our analysis illustrates the power of the multi-criteria approach and the depth of information related to tradeoffs provided by this approach.

6 The impact of pricing decisions

In this section, we further examine the effect of pricing decisions by the CDN provider on the choice of a best plan. In the previous example, we assumed $q_l = (0, 50, 100)$ terabytes per month and $f_l = (\$.40, \$.20, \$.05)$ per gigabyte, then examined the effect of an increase in the required bandwidth level to 70 terabytes to achieve the first-level quantity discount. While this increase in

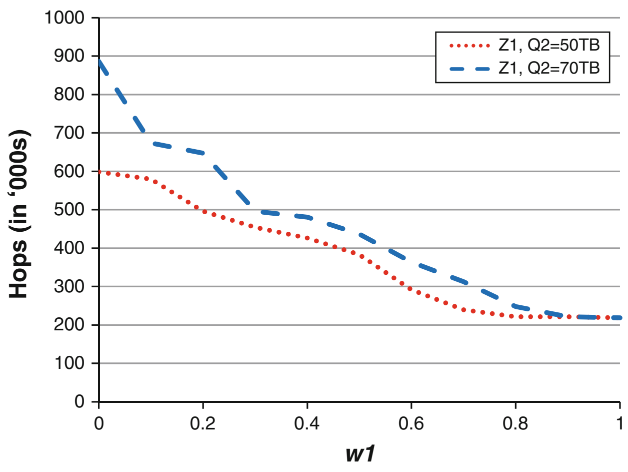

Fig. 1 Graph of hops by weight w_1

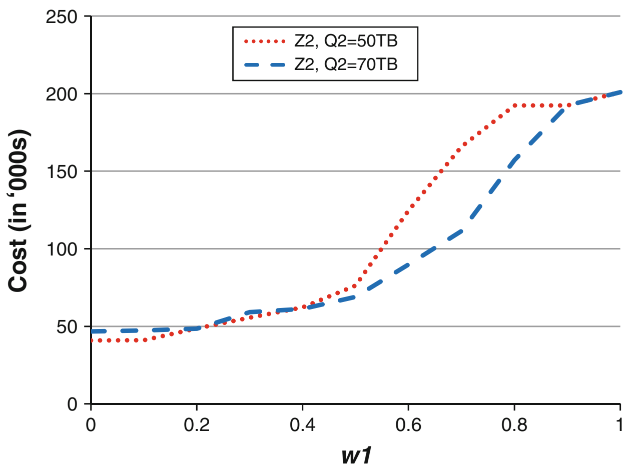

Fig. 2 Graph of cost by weight w_1

threshold had significant impact, this was not a significantly different pricing strategy but was more an inducement for CDN users to utilize more bandwidth in an effort to reduce transmission cost. In this section, we examine the effects of a dual pricing strategy which might be designed to reward or attract smaller customers (lower bandwidth usage) while still providing quantity discount opportunities for the larger customers. Alternatively, it could represent a current and a proposed pricing schedule that are part of an analysis preceding a contemplated price change.

We assume that the first pricing schedule shown above is still in effect, but that additionally we consider a pricing schedule whereby $f_l = (\$.35, \$.20, \$.05)$ per gigabyte and $q_l = (0, 60, 100)$ terabytes per month (storage costs are not changed because, though just as important as transmission costs, they are typically less volatile and competitive than transmission costs). In other words, the base price for bandwidth usage below the volume price threshold is lower by \$.05 per gigabyte. However, in this schedule the threshold to qualify for a volume discount on bandwidth is raised from 50 to 60 terabytes per month. We keep the other parameters at their original levels to facilitate comparison. The results of the original pricing schedule and the new pricing schedule are shown in Table 5.

Several points can be inferred from the table for this particular content provider with their pattern of request volume. First, when cost is deemed to be much more important than hops as indicated by weight vectors such as (0,1) or (.1,.9), the original pricing schedule is better. The lower base bandwidth cost of the new pricing schedule does not benefit this content provider, as their request volume is such that they are better served by adopting a plan with fewer copies and more required bandwidth which takes advantage of the volume discount price. However, because the threshold for volume price is increased in the new schedule, they would have to push the bandwidth beyond the level required in the original plan, resulting in higher cost (about $4,500 per month).

Next, when performance is deemed to be more important, as represented by weight vectors such as (1, 0) or (.9, .1), the new pricing schedule seems to be better. With no increase in hops, we are able to achieve slightly lower costs by about $2,000 per month.

Finally, when cost and hops are viewed to be about equally important, the decision is much less clear and requires the content provider to determine which outcome would be preferable. At a weight vector of (.5,.5), cost is slightly lower for the new price schedule, while hops are somewhat lower for the original price schedule. Also, the preference point at which the content provider would fall out of the volume discount range changes from (.6, .4) in the original plan to (.4, .6) in the new plan.

As with any multi-criteria solution, in the end the best solution is defined by how the decision maker views the tradeoffs between cost and performance. However, we believe this example further illustrates the type of information which this caching model can provide. For the CDN provider, such feedback, when coupled with knowledge of the general preferences of their customers for low cost versus high performance, could be beneficial in deciding how to price their CDN services. For the content provider, this information could be invaluable in choosing from a tiered pricing structure from their chosen CDN provider, in negotiating a service-level agreement, or in choosing between CDN providers.

7 Experimental results

In any study where a model must be instantiated with typical parameters, the model results are dependent on those parameters. Even in a study such as this one where the parameters are chosen to be typical of the industry, the results are still specific to those choices. In an effort to offer broader managerial insights, we conclude with an experiment which uses a broad range of parameter values.

The CDN model has two distinct sets of parameters. The parameters r_{ik}, d_{ij}, b_k, o_k, and s_j are network descriptors which define the CDN provider's available servers and their corresponding locations and capacities, and the content provider's caches and their size and popularity. The parameters f_l, q_l, and c_j are the bandwidth cost, quantity discount threshold, and storage cost, and reflect the pricing policy set by the CDN provider. In this experiment, we will utilize the network descriptors to generate several networks to serve as our base test problems, and then systematically vary the two cost parameters and the quantity discount threshold to allow us to analyze the effectiveness of varying pricing policy combinations. Specifically, we generated 20 hypothetical base network configurations using random values selected from uniform ranges for each of the network parameters. The uniform ranges were constructed to represent low-to-high value ranges for each of the network structure parameters, so that the 20 test problems represent a varied but structurally valid set of hypothetical networks.

Next, we defined three levels (low, medium, and high) for each of the two cost parameters and the threshold parameter, yielding 27 potential organizational pricing policy strategies. These cost parameters may again be ranges, such as in the case of storage costs at the servers, since each server will have a different storage cost. For example, the "low storage cost" range is from $2 to $4 per gigabyte per month, the "medium storage cost" range is from $4 to $6 per gigabyte per month, and the "high storage cost" range is from $6 to $9 per gigabyte per month. Then, we solved each of the 20 test problems for the 27 controlled combinations of cost parameters, with

Table 5 A comparison of two price schedules

(w_1, w_2)	Original pricing schedule					New pricing schedule				
	Z_1 (hops in 1,000s)	Z_2 (in '000s)	V_1	V_2	V_3	Z_1' (hops in 1,000s)	Z_2' (in '000s)	V_1'	V_2'	V_3'
(0, 1)	598.64	40.85	0	1	0	654.02	45.30	0	1	0
(.1 ,.9)	579.44	41.08	0	1	0	587.50	45.31	0	1	0
(.2, .8)	496.18	48.58	0	1	0	537.16	49.33	0	1	0
(.3, .7)	453.88	55.66	0	1	0	530.52	50.13	0	1	0
(.4, .6)	426.36	62.21	0	1	0	409.52	72.86	1	0	0
(.5, .5)	382.98	76.22	0	1	0	363.86	87.45	1	0	0
(.6, .4)	291.74	124.49	1	0	0	312.58	109.24	1	0	0
(.7, .3)	239.22	165.80	1	0	0	263.38	141.30	1	0	0
(.8, .2)	221.08	192.27	1	0	0	239.22	163.86	1	0	0
(.9, .1)	221.08	192.27	1	0	0	221.08	190.43	1	0	0
(1, 0)	218.92	200.97	1	0	0	218.92	199.05	1	0	0

three different decision maker preference structures (weight sets) for each combination, representing preference given to hops, preference given to cost, and equal preference. This gives 27 × 3 = 81 experimental combinations or cells with 20 model runs per cell (one for each of the hypothetical network configurations), yielding a total of 1,620 runs. While the test networks were generated manually, the combination of a Visual Basic code generating program for creating the CPLEX code and utilization of the Concert optimizer in CPLEX for automating the solution of the models allowed us to make the task manageable. In the experimental cells, we report the average and a margin of error at 95% confidence level across the 20 test problems, for cost in Table 6 and for hops in Table 7.

In general, we observe that the margin of error is relatively small compared to the average values reported in Tables 6 and 7, implying that the 95% confidence intervals are quite narrow. While some of the outcomes are predictable, such as average CDN cost getting relatively higher as we move from low-cost parameter combinations in the upper left corner of Table 6 toward high-cost combinations at the bottom right of Table 6, other insights are perhaps more interesting. Several of these are apparent from the tables:

- The impact of changing preferences has an asymmetric effect on costs. Moving from a baseline of equal preferences (.5, .5) to an emphasis on cost reduction (.2, .8), the relative decrease in cost is much less than the relative increase in cost which occurs when we move from equal preference to a preference on hops (.8, .2). This can be seen from the first three columns of Table 6 where, across all levels of f_l and q_l, the objective function impact of moving from an equal weighting strategy to an emphasis on cost is an average cost reduction of \$16,500 or approximately 28%, while a change to emphasis on hops produced an average cost increase of \$45,433, or slightly over 77%. This asymmetry is also present in columns 4–6 and columns 7–9 and can be clearly seen on the graph in Fig. 3.
- The impact of changing preferences is much more symmetric in terms of the resulting number of hops. Looking at the first three columns of Table 7, moving from a baseline of equal preference (.5, .5) to an emphasis on cost (.2, .8) caused an average increase in hops of 140,408, or about 48%. An equivalent change in preference on hops to (.8, .2) reduced hops by an average of 132,319 or approximately 45%. This linear trend can also be seen in the graph in Fig. 4.
- This asymmetry on cost and symmetry on hops implies that as we emphasize hops reduction, we will pay an increasingly expensive premium to achieve that reduction. Stated another way, the cost to hops ratio is not

Table 6 95% Confidence interval (average ± margin of error) for cost (in '000s) across 20 runs

	Low C_j			Medium C_j			High C_j		
	(.2, .8)	(.5, .5)	(.8, .2)	(.2, .8)	(.5, .5)	(.8, .2)	(.2, .8)	(.5, .5)	(.8, .2)
Low f_l, Low q_l	35.76 ± 1.37	55.21 ± 4.33	101.13 ± 5.25	49.84 ± 2.21	82.19 ± 5.02	155.42 ± 7.87	79.19 ± 3.72	123.61 ± 8.54	235.39 ± 9.75
Low f_l, Med q_l	37.22 ± 1.88	54.58 ± 4.91	101.83 ± 5.06	52.26 ± 2.86	84.46 ± 4.91	154.60 ± 8.33	81.72 ± 4.35	129.46 ± 10.17	235.40 ± 10.13
Low f_l, High q_l	38.91 ± 1.93	54.25 ± 4.74	100.36 ± 4.13	53.35 ± 2.60	83.45 ± 6.73	154.39 ± 7.97	83.79 ± 3.96	129.80 ± 10.18	236.96 ± 9.32
Med f_l, Low q_l	41.12 ± 1.31	58.56 ± 3.63	104.20 ± 5.61	55.74 ± 2.14	86.40 ± 6.13	155.72 ± 9.52	85.34 ± 3.63	130.81 ± 8.29	240.81 ± 9.66
Med f_l, Med q_l	42.91 ± 1.92	58.14 ± 4.70	103.88 ± 5.27	58.36 ± 2.84	87.41 ± 6.18	156.82 ± 8.53	88.36 ± 4.45	134.45 ± 10.18	238.56 ± 11.85
Med f_l, High q_l	44.47 ± 2.24	57.63 ± 3.67	102.99 ± 4.46	60.50 ± 2.74	83.31 ± 6.09	153.64 ± 7.92	90.60 ± 4.01	137.15 ± 10.05	240.88 ± 9.61
High f_l, Low q_l	45.26 ± 1.64	61.70 ± 4.06	105.28 6.06	61.07 ± 2.05	90.07 ± 5.69	160.88 ± 9.16	91.45 ± 3.62	131.84 ± 11.34	246.04 ± 8.97
High f_l, Med q_l	47.50 ± 2.47	62.35 ± 3.96	108.44 ± 4.59	63.95 ± 2.81	90.53 ± 6.20	159.41 ± 9.28	94.81 ± 4.23	134.29 ± 9.86	243.98 ± 12.96
High f_l, High q_l	46.93 ± 3.01	66.43 ± 3.49	109.64 ± 4.78	66.50 ± 3.07	87.59 ± 5.87	159.90 ± 8.04	97.77 ± 4.21	135.89 ± 11.60	239.43 ± 12.29
Average	42.23	58.76	104.19	57.95	86.16	156.75	88.11	131.92	239.72

Table 7 95% Confidence interval (average ± margin of error) for number of hops (in '000s) across 20 runs

	Low C_j			Medium C_j			High C_j		
	(.2, .8)	(.5, .5)	(.8, .2)	(.2, .8)	(.5, .5)	(.8, .2)	(.2, .8)	(.5, .5)	(.8, .2)
Low f_l, Low q_l	398.35 ± 18.23	267.77 ± 29.57	154.68 ± 16.97	405.04 ± 25.23	262.13 ± 24.97	152.62 ± 17.41	391.65 ± 17.76	267.50 ± 22.65	152.79 ± 16.86
Low f_l, Med q_l	404.53 ± 18.71	281.86 ± 26.67	155.10 ± 17.97	403.17 ± 24.40	263.11 ± 23.66	154.25 ± 17.48	391.12 ± 17.25	261.60 ± 20.79	153.50 ± 16.77
Low f_l, High q_l	411.75 ± 19.50	292.77 ± 23.20	158.22 ± 16.31	410.83 ± 24.45	272.79 ± 25.75	154.91 ± 16.85	389.61 ± 19.65	266.59 ± 24.99	153.06 ± 16.78
Med f_l, Low q_l	414.42 ± 22.38	279.32 ± 24.58	157.67 ± 15.55	409.84 ± 26.32	267.69 ± 25.91	156.97 ± 17.24	392.34 ± 17.93	264.56 ± 23.51	152.79 ± 16.86
Med f_l, Med q_l	419.28 ± 22.22	296.09 ± 24.12	160.79 ± 17.23	410.60 ± 26.09	273.71 ± 26.85	157.44 ± 17.82	393.86 ± 16.87	265.30 ± 21.51	155.25 ± 16.88
Med f_l, High q_l	442.87 ± 22.15	311.12 ± 22.71	163.58 ± 15.50	412.68 ± 24.64	296.06 ± 26.24	161.35 ± 16.81	396.63 ± 18.67	266.43 ± 24.69	154.52 ± 16.17
High f_l, Low q_l	439.39 ± 28.39	293.37 ± 26.01	162.74 ± 12.95	417.09 ± 26.90	274.64 ± 25.09	156.97 ± 17.13	393.49 ± 18.33	274.57 ± 26.38	153.31 ± 17.70
High f_l, Med q_l	442.72 ± 26.39	305.91 ± 23.07	161.66 ± 16.41	418.60 ± 26.13	285.91 ± 28.61	160.71 ± 19.51	396.61 ± 17.31	278.48 ± 21.86	155.70 ± 17.76
High f_l, High q_l	518.48 ± 37.74	299.91 ± 22.03	162.81 ± 17.56	427.85 ± 24.65	306.15 ± 27.48	161.36 ± 16.73	399.40 ± 18.54	284.94 ± 27.57	159.26 ± 15.33
Average	432.42	292.01	159.69	412.86	278.02	157.40	393.86	270.00	154.46

uniform across the range of preferences. As we strive to reduce the number of hops by giving it greater weight in the model, the relative cost required to achieve this hops reduction increases, resulting in very high monthly costs for the lowest level of hops (See Fig. 5). Looking toward the upper left and lower right corners of Tables 6 and 7 (where the lowest and highest hops and cost tend to occur), in order to reduce hops by 60%, we must increase monthly expenditures by 6.69 times (from $35,760 to $239,430).

- Storage cost fluctuations seem to have a much greater effect on cost than on hops. In Table 6, if we pick any preference structure, such as (.5, .5) in columns 2, 5, and 8, and look at the increase in cost as we increase C_j, the increase is significant. As we go from *Low* C_j to *Medium* C_j, across all levels of f_l and q_l the average cost increases by 46%. Going from columns 5 to 8, average cost increases by 53%. The corresponding changes in storage cost levels result in hop decreases of 5 and 3%, respectively. This same behavior is true for the other

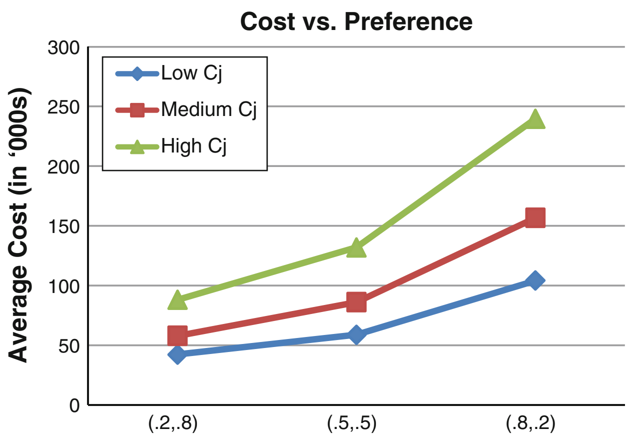

Fig. 3 Cost versus preference

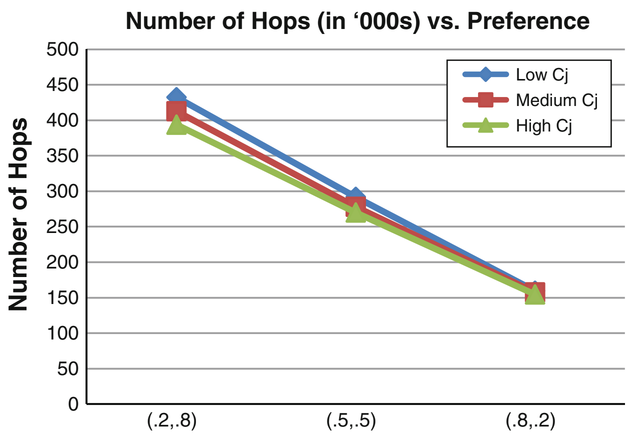

Fig. 4 Number of hops versus preference

preference structures. Therefore, moving in reverse from higher storage cost to lower storage cost, we may be able to reduce storage costs and therefore overall cost significantly without incurring any substantial increase in hops. As a result, organizations working on a thin margin may benefit by working diligently to secure the lowest possible storage rates from their CDN provider.

- Hops are much more sensitive to bandwidth charges when minimizing hops is the secondary objective as opposed to the primary objective, but only when storage cost is low. In Table 7, columns 1 and 3, suppose we compare row 1 (*Low* f_l, *Low* q_l) to row 9 (*High* f_l, *High* q_l). In column 1, hops increase by 30% as we move from the lower to higher bandwidth price combination. In column 3, where we have a preference structure that places more emphasis on hops, the corresponding increase in hops is only 5%, illustrating that when we prioritize hops we are able to find solutions which better mitigate the high bandwidth charges. If we look at columns 4 and 6 for medium storage cost, the changes do not display the same behavior and are basically equal (5.6 and 5.7%). This is also true for high storage cost in columns 7 and 9, where the increases are 2 and 4%. While the ability to reduce bandwidth costs when emphasis is placed on hops is to be expected, the degree of dependence on low storage cost for this effect is less intuitive but likely stems from the fact that the solution needs the flexibility of being able to place numerous cached copies (at a relatively low cost) in order to reduce hops and thereby mitigate the effect of high bandwidth charges. As storage cost increases, this flexibility is removed. This implies that when choosing a CDN vendor, one very important factor is to make sure that the storage cost structure of the vendor is low enough to provide adequate flexibility to accommodate aggressive hops and bandwidth control.

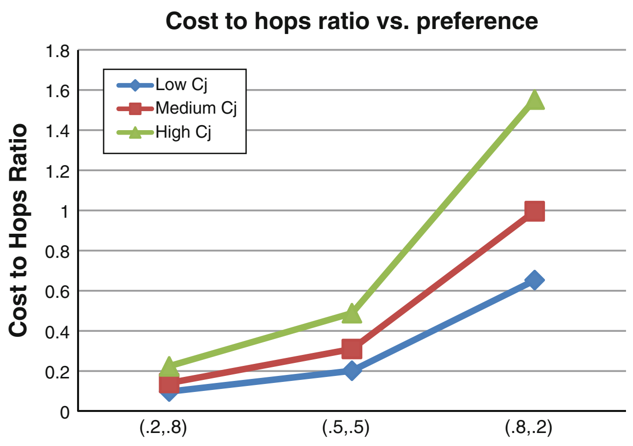

Fig. 5 Cost to number of hops versus preference

While other inferences may be drawn from these experimental results, in the end any conclusions which we draw are still influenced to some degree by the structure of the experimental networks and the chosen parameters. The main contribution continues to be demonstration of the insight that the modeling process can provide to a CDN provider and a content provider, once their unique cost and demand structures have been encoded.

8 Conclusions

Developing a caching plan which simultaneously provides good performance, as measured by low latency and hops, and low storage and bandwidth costs is a formidable task. When the CDN environment is further complicated by service-level agreements which include bandwidth volume pricing levels, the task is even more difficult. In this paper, we have presented a model for analyzing the multiple objectives of cost and performance within the CDN caching problem. This model not only treats these two objectives but also allows for the inclusion of bandwidth volume discount pricing. We utilize a case example to depict the feasibility of solving this model for a CDN of reasonable size and illustrate the depth of information related to preference tradeoffs. The example clearly defines the nature of the tradeoffs as the decision maker's emphasis switches between hops and cost. When cost is emphasized and hops are given no weight, fewer cached copies of the collections were used resulting in lower storage costs and in forwarded request bandwidth volume which rise into a volume discount pricing tier where there are more favorable bandwidth rates. When the weights emphasize hops, we see the lowest possible hops but at the largest possible cost. In this case, more copies are cached to reduce the necessary hops, resulting in more storage cost and a bandwidth usage in the most expensive price range (below any volume price levels) for those requests which must be forwarded. For all of the compromise solutions in between, we see solutions which achieve more of a balance. In a variation of this example with a higher threshold for the volume pricing discount, when cost is emphasized the solution pushes the request volume to the highest volume discount level in order to achieve the lowest cost, and as we increase w_1 and lower w_2 gradually putting more emphasis on hops, the request volume decreases into the second tier of volume pricing, and eventually into the first or lowest tier of volume pricing.

In a second example which featured two base bandwidth prices with different thresholds for discounts, we observed a clear dominance by one of the pricing schemes when cost is emphasized, with a shift to the other pricing policy being clearly preferred when hops are emphasized. This shows that for the CDN provider, the caching model could be beneficial in deciding how to price their CDN services. For the content provider, the model solutions could be invaluable in choosing from a tiered pricing structure from their chosen CDN provider, in negotiating a service-level agreement, or in choosing between CDN providers. In a final experiment with a broad set of possible network and cost parameters, we provided some general insights into cost/hops tradeoffs.

Balancing multiple objectives such as cost and performance in a complex decision environment like CDN caching is a challenging task. We believe that a model such as this one, further customized to accommodate any additional CDN factors that might be unique to a particular situation, could provide valuable assistance to a CDN decision maker or content provider.

References

1. Adb El-Wahed WF, Lee SM (2006) Interactive fuzzy goal programming for multi-objective transportation problems. OMEGA 34:158–166
2. Almeida JM, Eager DL, Vernon MK, Wright SJ (2004) Minimizing delivery cost in scalable streaming content distribution systems. IEEE Trans Multimed 6(2):356–365
3. Bekas T, Cordeau J, Erkut E, Laporte G (2008) Exact algorithms for the joint object placement and request routing problem in content distribution networks. Comput Oper Res 35:3860–3884
4. Bektas T, Oguz O, Ouveysi I (2007) Designing cost-effective content distribution networks. Comput Oper Res 34:2436–2449
5. Bit AK, Biswal MP, Alam SS (1992) Fuzzy programming approach to multicriteria decision making transportation problem. Fuzzy Sets Syst 50:135–141
6. Bit AK, Biswal MP, Alam SS (1983) Fuzzy programming approach to multiobjective solid transportation problem. Fuzzy Sets Syst 57:183–194
7. Bit AK, Biswal MP, Alam SS (1993) An additive fuzzy programming model for multiobjective transportation problem. Fuzzy Sets Syst 57:313–319
8. Chang NB, Wen CG, Chen YL (1997) A fuzzy multi- objective programming approach for optimal management of the reservoir watershed. Eur J Oper Res 99:289–302
9. Cidon I, Kutten S, Soffer R (2002) Optimal allocation of electronic content. Comput Netw 40(2):205–218
10. Coffin MA, Taylor BW III (1996) Multiple criteria R&D project selection and scheduling using fuzzy logic. Comput Oper Res 23:207–220
11. Cohon JL (2003) Multiobjective programming and planning. Dover Publications, Inc., Mineola
12. Comer DE (2009) Computer networks and internets, 5th edn. Prentice-Hall, New Jersey
13. Datta A, Datta K, Thomas H, VanderMeer D (2003) World wide wait: a study of internet scalability and cache-based approaches to alleviate it. Manage Sci 9:1425–1444
14. Kangasharju J, Roberts J, Ross KW (2002) Object replication strategies in content distribution networks. Comput Commun 25(4):376–383
15. Kumar C (2009) Performance evaluation for implementations of a network of proxy caches. Decis Support Syst 46:492–500
16. Kumar C, Norris JB (2008) A new approach for a proxy-level web caching mechanism. Decis Support Syst 46:52–60
17. Laoutaris N, Zissimopoulos V, Stavrakakis I (2004) Joint object placement and node dimensioning for internet content distribution. Inf Process Lett 89(6):273–279
18. Laoutaris N, Zissimopoulos V, Stavrakakis I (2005) On the optimization of storage capacity allocation for content distribution. Comput Netw 47(3):409–428
19. Lee ES, Li RJ (1993) Fuzzy multiple objective programming and compromise programming with Pareto optimum. Fuzzy Sets Syst 53:275–288
20. Li L, Lai K (2000) A fuzzy approach to the multiobjective transportation problem. Comput Operat Res 27:43–57
21. Nguyen T, Chou CT, Boustead P (2003) Resource optimization for content distribution networks in shared infrastructure environment. In: Proceedings of the Australian telecommunications networks applications conference. Available at http://www.atnac2003.atcrc.com/ORALS/NGUYEN-resource.pdf [accessed 10-19-09]
22. Pallis G, Vakali A (2006) Insight and perspectives for content delivery networks. Commun ACM 49:101–106
23. Pathan A, Buyya R (2008) A taxonomy and survey of content delivery networks, GRIDS Laboratory, University of Melbourne. Online; available: http://www.gridbus.org/cdn/reports/CDN-Taxonomy.pdf [accessed 10-19-09]
24. Plastria F (2002) Formulating logical implications in combinatorial optimization. Eur J Oper Res 140:338–353
25. Qiu L (2001) On the placement of web server replicas. In: Proceedings of the 20th IEEE INFOCOM conference, pp 1587–1596
26. Ryoo J, Panwar SS (2001) File distribution in networks with multimedia storage servers. Networks 38(3):140–149
27. Streaming Media (2009) Video CDN Pricing in Q1 2009. Available at http://www.blog.streamingmedia.com/the_business_of_online_vi/2009/06/video-cdn-pricing-drops-in-q1-but-not-by-much-other-contract-trends-noticed.html [accessed 10-19-09]
28. Topaloglu S, Selim H (2007) Nurse scheduling using fuzzy multiple objective programming. Lecture notes in computer science. Springer, Berlin, pp 54–63
29. Xu J, Li B, Lee DL (2002) Placement problems for transparent data replication proxy services. IEEE J Sel Areas Commun 20(7):1383–1398
30. Xuanping Z, Weidong W, Xiaopeng T, Yonghu Z (2003) Data replication at web proxies in content distribution network. Lecture notes in computer science, vol 2642. Springer, Berlin, pp 560–569
31. Yager R (1978) Competitiveness and compensation in decision making. Inoa College Report RRY, New Rochelle, pp 78–14
32. Yang M, Fei Z (2003) A model for replica placement in content distribution networks for multimedia applications. In: Proceedings of IEEE international conference on communications, pp 557–561
33. Zimmermann HJ (1978) Fuzzy programming and linear programming with several objective functions. Fuzzy Sets Syst 1:45–55
34. Zimmermann HJ (1985) Applications of fuzzy set theory to mathematical programming. Inf Sci 34:29–58
35. Zimmermann HJ (1987) Fuzzy sets, decision making and expert systems. Kluwer, Dordrecht
36. Zimmermann HJ (1996) Fuzzy set theory and its applications. Kluwer, Boston