

# Models for Internet Cache Location

Adam Wierzbicki  
*Institute of Telecommunications*  
*Warsaw University of Technology*  
*Warsaw, Poland 00-665*  
adamw@tele.pw.edu.pl

## Abstract

The paper discusses models of the Cache Location Problem from the perspective of Decision Support Systems. This approach leads to the design of models that allow the user (decision maker) to flexibly specify his preferences, and are precise in the search for solutions that are closest to these preferences. The models are designed using a multi-criteria approach and are specified using Mixed Integer Linear Programming. The paper considers what is the maximal size of cache location problems that can be solved optimally in realistic time using state-of-the-art MILP solvers. It is found that medium-sized problems can be solved optimally; the paper considers how to extend existing heuristics to solve larger problems using the newly developed models.

## 1 Introduction

Caches and replication servers store information transmitted over a computer network closer to the users of that information. In other words, caching and replication are used to distribute the content provided over computer networks. By deploying caches in a network, network administrators can manage the location of information that is frequently requested by clients. Among the well-known impacts of caching on network and system performance are: a reduction of network traffic, a reduction of average client delay, and a reduction of origin server load [12].

Companies that have located caches in the network can offer services to content providers, who can deliver their content through the caches owned by the company. These caches form an overlay network called a Content Delivery Network (CDN). The problem of information location to maximize performance or minimize the cost of a system that uses this information has been studied extensively since the early 1980s in several fields, including distributed databases, storage systems and CDNs [6, 1, 7]. This previous work solved the following decision problem: given a set of objects (files, pages of memory) and a set of potential locations, decide in which locations to place copies of the objects in order to optimize some objective and satisfy some additional constraints. A practical example of this general formulation is the problem of location of replicas of specific files on a number of nodes (for example, the caches of a CDN). We shall refer to this decision problem as the Replica Location Problem (RLP).

Recently, the subject of designing CDNs by choosing appropriate locations for caches in the network has received attention of researchers. The decision problem can be stated as follows: given a set of potential locations, decide in which locations to place caches, which will store copies of objects, in order to optimize some objective and satisfy some additional constraints. Observe that this decision problem is not concerned with the location of replicas of specific files, but with the location of the caches themselves. Also, the client demands are treated on a more aggregated level (not for individual files, but for files from a specific origin server). In [8], Krishnan *et al.* formulated the Cache Location Problem

(CLP) and studied heuristics and special-case optimal solutions of that problem. The authors showed that cache location had a significant impact on overall performance, and that commonly used rules-of-thumb for cache location were not always appropriate.

Most of the considered optimization problems related to cache or replica location are hard (the CLP was shown to be NP-hard, even on a tree topology, if multiple servers are considered [8]). This is the main reason for the focus of the majority of research in the area on heuristic solutions, their computational complexity and solution quality. This research direction has resulted in many useful algorithms for many different model formulations [11, 10]. However, all of these model formulations seem to share a common drawback: they do not allow to specify user preferences, and do not consider whether the model is realistic enough to give an accurate evaluation of the quality of the designed cache or replica location.

The aim of this paper is to approach the problem of cache and replica location from the perspective of Decision Support Systems (DSS). A Decision Support System is a program that supports a decision maker through the entire decision making process for a specific situation. Decision making is a major component of human life, and there are many types of decision making (for example, institutional or individual decisions, decision under risk, etc). Therefore there can be (and indeed are) many types of DSS. However, many of them need to handle three aspects of decision making [15]: (i) information about current situation and history; (ii) the relation between basic processes and actions or decisions, which is usually expressed using a mathematical model; (iii) the decision process, which has several phases, one of which is the choice among decision alternatives. Using these three aspects, we shall attempt to design a DSS for the problem of cache or replica location. Such an approach is useful, since it is more likely to result in a system that will be used to solve realistic problems. Also, we shall demonstrate how this approach leads to different models compared to the computationally-oriented previous work.

In the next section, we shall attempt to characterize the decision maker of the CLP and his preferences. Next, we shall consider the information aspect of the CLP and replica location. Apart from discussing the input data required for these decision problems, we shall consider the methods for gathering this information, and ways of dealing with the variability of this information over time. In the fourth section, we shall attempt to give models of cache location using Mixed Integer Linear Programming (MILP), but not limited to that approach, and compare these models with existing formulations. In section five, we consider ways of supporting the decision process, including methods of learning decision maker preferences and supporting his final choice of a decision among many alternatives. Section six gives a brief evaluation of the computational complexity of using MILP for cache location, and of extensions of existing heuristics to support the model modifications outlined in this article. The last section concludes.

## **2 The decision maker preferences for cache or replica location**

The decision maker (DM) who considers the problem of cache or replica location is usually the manager of a communication network. However, his preferences may be related to the interests of network users. The users of the network of the DM can be clients who request information from content providers, or the content providers themselves, who pay the DM for distributing their content closer to clients. The preferences of the DM may also be affected by the operating costs of the network. Finally, the DM preferences may be influenced by the cost of locating a new cache in the network, or by the cost of moving existing caches.

The DM could have two different kinds of preferences related to network users, one kind if he is the manager of a CDN, and another kind if he is the manager of a public caching infrastructure that does not charge for content distribution. The preferences of users are a function of the quality of service, which is affected by the communication delay

in receiving the content by a client. If the DM is a manager of a CDN, he is interested in maximizing revenue. The DM could therefore be influenced by the amount of money that each content provider pays for the services of a CDN. If the DM manages a public caching infrastructure, the interests of clients should influence him equally, and the DM could be interested in a fair treatment of the clients.

The cost of location of a new cache is a function of price of caching hardware and software. A DM could choose among the products of cache vendors and learn their prices before considering the decision problem of cache location. If the DM chooses a publicly available cache software, the price of the cache is the price of hardware from a selected vendor and with selected computational resources. Next, the DM can set a budget  $B$  and divide the budget by the price  $p$  of the selected cache product, which results in a maximal number of caches,  $m$ , that he can locate. Another approach would consider the price  $p$  of the chosen product as a fixed cost of cache location, while the price of a network connection for each cache is the variable cost of cache location. The budget set by the DM can then be used along with these costs to set a constraint on the expense of locating caches. Note that this constraint is not sufficient to express DM preferences, since obviously the DM would like to pay as little as possible for cache location. When a new cache is purchased, the price of a network connection must be considered. The cost of the network connection can depend on the cache location. The cost of moving existing cache is harder to quantify, and therefore it may suffice to minimize the number of caches which need to be moved.

The operating costs of a network are a function of the traffic required to communicate the content to clients. The DM may have different preferences with respect to traffic that is sent over different parts of his network (for example, traffic over a link connecting the DMs network to an Internet Service Provider is likely to be more expensive than intranet traffic).

The decision problem of cache or replica location could be considered by the DM when he is de-

signing his network. Then, cache or replica location could be connected to the planning of link capacities, network reliability, and other network design issues. On the other hand, the DM may already have an operating network and want to design a CDN by locating caches or replication servers. Finally, the DM may already have an operating network and installed caches, and could consider adding new caches, extending the capacity or changing the location of installed caches, after evaluating the performance of his caching system. The DM could also proceed in reverse order, by specifying a goal with respect to some preference, and then considering what changes are needed in the caching system to achieve that goal. It follows that the DM may be using the DSS periodically over time, and not only once when the caching system is first designed.

RLP problems might also be considered after the DM has located caches or replication servers, to determine what content should be stored by these servers. However, CDNs can use dynamic, on-line algorithms to determine the contents of replication servers or caches [4]. Cache content can also be determined entirely by requests of clients (if the CDN is not using push or prefetching), which results in the storage of most popular content in the caches.

In a real setting, the DM would consider all or most of the discussed preferences simultaneously. He would not start a decision process with a fixed set of preferences and knowledge of their relative importance. Also, the preferences of a DM could change over time. For these reasons, the DSS should take into account these preferences of a DM in an interactive way, and be sufficiently flexible to let the DM change his preferences during a session with the DSS.

### 3 The information aspect of the CLP

The information required as input to the decision problem of cache location is not large in volume, but not easy to obtain. In fact, many network administrators do not maintain such information and

Table 1: Input information of the CLP.

Variable name	Variable explanation
$z_c^s$	Demand from server $s \in V_S$ to client $c \in V_C$
$h_c^s$	Hit rate of demand from $s \in V_S$ to $c \in V_C$
$V_R$	Set of potential cache locations
$\pi_{uv}$	Communication path from $u \in V_S \cup V_R$ to $v \in V_C \cup V_R$
$G = \langle V, E \rangle$	Graph of network topology, $V = V_S \cup V_R \cup V_C$
$\delta'(e)$	Cost of communicating on link $e \in E$
$\delta'(u, v)$	Cost of communicating on path $\pi_{uv}$ , $u, v \in V$
$\delta''(e)$	Delay of communicating on link $e \in E$
$\delta''(u, v)$	Delay of communicating on path $\pi_{uv}$ , $u, v \in V$

are reluctant to make the effort to obtain it. However, others use the same or similar information on a daily basis to evaluate the performance of their network. It is not only possible, but beneficial for a network administrator to periodically gather this type of information. We shall attempt to describe the minimal amount of input information of the CLP. A summary of all input variables used by the CLP is given in Table 1.

The basic information of the cache or replica location problems are the client demands. For the CLP,  $z_c^s$  are demands from client  $c \in V_C$  for information from server  $s \in V_S$ . For the replica location problem, the demands  $z_c^{sf}$  are made for specific files  $f \in F$ . The sets  $V_C$  and  $V_S$  contain nodes of the network of the DM.

For the CLP (but not the RLP), each demand must be associated with a hit rate  $h_c^s$ , which is the proportion of traffic of the given demand that will be served by a cache. This value is needed as an input because the CLP treats demands in an aggregate way and does not consider the actual contents of a cache or the client demands to specific files of the server, as in the replica location problem. The hit rates can be most easily obtained from server logs. The need to have the hit rate as an input is a consequence of aggregating the demands for individual files into demands to servers. In the RLP, the location of individual files is the subject of the decision problem, and hit rates are either 0 or 1 depending on whether the file is located at the given cache or

not (which makes the dependence of the model on hit rate trivial). The price for such an approach is an increased number of individual demands, and, consequently, of decision variables and therefore problem complexity (see Section 6).

Next the DM has to specify the set of potential locations for a cache or replication server,  $V_R$ . This set contains nodes of the network where caches are to be located. Let  $\pi_{uv}$  be the path in the network of the DM connecting node  $u$  and  $v$ . The paths that will be the input to the cache or replica location problem should be determined by the routing algorithm used in the network. In other words, they should reflect how the information will be communicated in the network. Note that the paths need not be the outcome of shortest-path IP routing; they could be determined by another type of routing, such as Multi-protocol Label Switching (MPLS) or the Per Hop Behavior of DiffServ. What matters is that the paths should reflect the actual communication behavior of the network of the DM.

For each demand  $z_c^s$  and each potential cache location  $w \in V_R$ , we need to know the path  $\pi_{wc}$ , and, for the cache location problem, the path  $\pi_{sw}$ . Finally, we also need to know the path  $\pi_{sc}$ .

From the paths and nodes described above, a topological view of the relevant part of the network can be constructed. The network could be visualized, allowing the DM to interactively specify separate costs for each link on the paths where

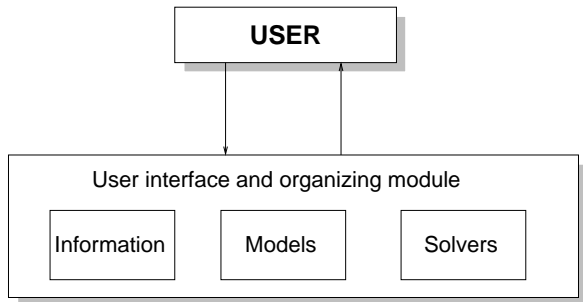


Figure 1: Modules of a Decision Support System.

the information will flow. This network topology can be described by a graph  $G = \{V, E\}$ , where  $V = V_C \cup V_S \cup V_R$ , and  $E = \bigcup_{u,v \in V} \pi_{uv}$  is the sum of all links on the routing paths. For each  $e \in E$ , the DM could specify a link cost  $\delta''(e)$  of sending a unit of data on that link. The total cost  $\delta''(u, v)$  of communicating a unit of information on the path  $\pi_{uv}$  can be obtained by adding all link costs of the path.

For each of the routing paths  $\pi_{uv}$ , the DM should specify an end-to-end delay  $\delta'(u, v)$  of sending a unit of data. This delay is the hardest to define and to measure. The DM could specify some link-level delays for each of the links of the path (depending on the link-layer technology) and processing delays for each node, and then add these delays for each path  $\pi_{uv}$ . However, the actual delay of network communications of the HTTP and FTP protocol is determined by the average available TCP bandwidth during the duration of a connection. This, in turn, depends on the amount of traffic that travels on the path. Such a delay is therefore asymmetric and non-additive, and can only be calculated by end-to-end measurements for all relevant pairs of nodes, under the assumption that we do not take into account the changes in traffic patterns that will result from cache location.

The end-to-end measurements could use a program that estimates the available TCP bandwidth [9] or measure end-to-end delay and loss using a sample of packets, and then use a formula that approx-

imates available TCP throughput [5]. (The first method is more exact, but may use more traffic.) To decrease the cost of measurements, a part of the delays (for the server-client paths that have significant traffic) can be obtained from server logs. The described measurements depend on seasonal traffic patterns, and therefore one specific time of day should be chosen for all measurements, and the measurements should be limited to working days. This implies that the amount of network resources used for measurements will not be large on a daily scale. Apart from the available TCP throughput, the end-to-end client latency includes the connection setup time of TCP, which is influenced by the packet transmission times on the path from the client to the server, and on the return path. This value (denoted by  $\delta^{lat}(u, v)$ ) can be estimated by the minimal packet round-trip time multiplied times  $3/2$ .

We believe that the advantage of using more relevant input information (average available bandwidth) outweighs the disadvantage of the variability of this information. The variability could be controlled using the following scheme: the results of measurements of available TCP bandwidth could be filtered by an exponential moving average. Next, instead of using the smoothed estimate, we define discretized bandwidth levels. The granularity of these levels can be used to control the sensitivity to changes of the smoothed estimate. To minimize possible oscillations when the smoothed estimate is close to a bandwidth level, a simple hysteresis algorithm could be employed. Thus, while we move down a level immediately when the smoothed estimate falls below the current level, we move up a level only if the estimate significantly exceeds the bandwidth corresponding to the next level.

Finally, the DM should also specify fixed cost  $p$  and a variable cost  $p'_w$  of cache location and a budget  $B$  for the expense of cache location, as was described in the previous chapter. The minimal amount of input information is: the client demands, the hit rates, the set of potential cache locations, the relevant costs of paths from each server to each potential cache location and from each potential cache location to each client.

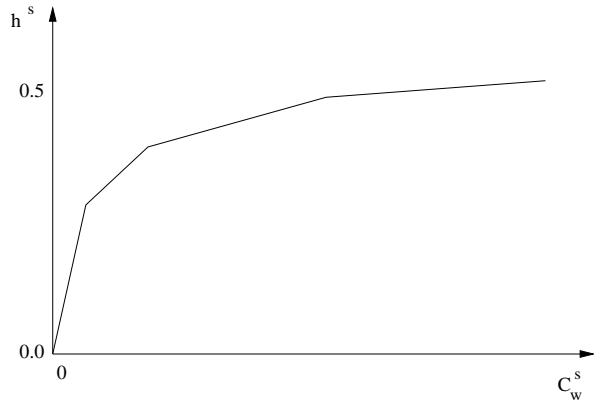


Figure 2: Approximated relationship of the number of cache clients and hit rate.

To sum up, the information used by the cache or replica location problems is of the type that can be useful to network administrators in network management. A DSS could contain a module dedicated to the gathering of such information. This module could be used by network administrators in daily management tasks. The basic modules of the DSS are shown on Figure 1. In the next section, we shall consider the models of cache location required by a DSS.

#### 4 Basic processes and models of the CLP

A natural approach for describing the relation between basic processes and decisions is to establish a mathematical model that uses some variables that are related to decisions as inputs and some characteristics of the resulting process as outputs. This model should be simple enough to easily find attractive decisions that result in good (or best) outcomes, but complex enough to capture essential behavior of the basic processes.

Cache location decisions can be described by two types of binary decision variables: the *location variables* and the *assignment variables*. For each demand  $z_c^s$  and each potential cache location  $w \in V_R$ , the location variable  $L_w = 1$  if a cache is located at the node  $w$ , and the assignment variable  $P_{cw}^s = 1$  if the demand  $z_c^s$  has been assigned to  $w$ . These two variables must be related by the common-sense

constraint:

$$\forall_{z_c^s > 0} \forall_{w \in V_R} P_{cw}^s \leq L_w \quad (1)$$

which prohibits the assignments of demands to nodes where no cache has been located. Another constraint that is common to all models requires that the assignment of demands to caches and the origin server is a function:

$$\forall_{z_c^s > 0} \sum_{w \in V_R \cup \{s\}} P_{cw}^s = 1 \quad (2)$$

For replica location, the location and assignment variables need to be considered separately for each file  $f \in F$ . The two constraints remain similar. Thus, there could be a larger number of both location variables and assignment variables for that problem. The CLP uses the assumption that a hit rate is given for every demand to aggregate the demands made by one client to one server for individual files. This is in fact an implicit model assumption (that the hit rate is not affected by the location decisions), that reduces the complexity of the model. Shortly, we shall consider how this assumption could be modified and partially relaxed.

Additional constraints could be added to the model, for example to express that demands can only be assigned to cache locations that are on the shortest path (in terms of hop count) from the client to the server of the demand. Such a constraint expresses the assumption that the caches use an L4 switch or similar technology to intercept client requests (transparent caching [12], also called Transparent En-Route Caching in [8]). When the routing in the network is known, such an assumption is easy to express by adding constraints that force certain assignment variables to be equal to zero (for the cache locations that do not lie on the shortest path from client to server). Such constraints will not be considered further in this paper. We will now turn to an outline of a multi-criteria approach to models of the CLP.

In section 2 it was said that the relative importance of the various preferences of a DM cannot be

known in advance of the decision process, and that the DM could change his preferences over time. For that reason, DM preferences are best expressed using a multi-criteria approach. The essence of any model for cache or replica location is its objective function. In a multi-criteria approach, the objective function is composed from several functions of the decision variables, called *criteria*, that express the preferences of a DM. For example, one of the criteria (minimized) is the cost of cache location, which can be expressed by the function:

$$Q_m = \sum_{w \in V_R} (p + p'_w) L_w$$

If the DM wishes to move old caches to new locations, he can be interested in minimizing the number of caches to be moved. This can be achieved by letting the DM specify old cache locations  $L'_w$  as an input. The number of caches which change locations (have to be moved) is expressed by the criterion:

$$Q''_m = \frac{1}{2} \left( \sum_{w \in V_R} |L_w - L'_w| - \left| \sum_{w \in V_R} L_w - \sum_{w \in V_R} L'_w \right| \right)$$

If the DM has specified old cache locations, the cost of new caches that have to be located is given by:

$$Q'_m = \max \left\{ \sum_{w \in V_R} (p + p'_w) L_w - \sum_{w \in V_R} (p + p'_w) L'_w, 0 \right\}$$

Among other criteria of the CLP, let us consider the time in which a client receives requested information from all servers. For each client demand, the delay of sending the demand from the assigned location is given by the expression  $R_c^s = \delta^{lat}(c, w) + z_c^s \sum_{w \in V_R \cup \{s\}} P_{cw}^s (\delta'(w, c) + (1 - h_c^s) \delta'(s, w))$ . This expression includes the latency of setting up a TCP connection from the client to the cache, but not from the cache to the server (this is not possible without knowing whether the request was a hit or a miss. However, caches could make persistent connections to popular servers.) The client delay is given by:

$$\forall_{c \in V} Q'_c = \sum_{s \in V_S} R_c^s$$

This criterion depends on the hit rate  $h_c^s$ . Recall that in the previous section the hit rate was described as an input information to the CLP. However, in reality the hit rate depends on the number of clients that are assigned to a cache. Since clients usually use their own local caches in the browser, the temporal reference locality of requests that arrive at a cache is the outcome of many clients requesting the same popular objects. It is clear that the use of a constant hit rate that depends on the particular demand is a simplification. To express the basic process of cache hits and misses more realistically, the hit rate should depend on the particular cache and on the number of clients who are assigned to the cache. This number can be expressed in the following way: for every cache  $w \in V_R$  and every server  $s \in V_S$ , let  $C_w^s = \sum_{c \in V} P_{cw}^s$  be the number of clients of that cache that request data from  $s$ . The hit rates  $h_w^s$  at the cache  $w$  of all demands from the server  $s$  would depend on  $C_w^s$ . The form of that relationship should be obtained from relevant studies [2, 14, 3], but it can be approximated by a piecewise linear function, as shown in Figure 2.

However, if this model enhancement is to be considered, then some of the model criteria would not be linear. Therefore the resulting model could be more difficult to solve. We shall return to this question in section 6. Until then, we shall continue to use the simplification of a constant hit rate given as input information.

The other criterion related to delay is the latency of communicating content of a certain server (which is owned by a content provider who is the client of the CDN) to all its clients. This criterion in the CLP is given by:

$$\forall_{s \in V_S} Q''_s = \sum_{c \in V} R_c^s$$

The model of the CLP formulated in [8] is equivalent to minimizing the objective function  $\sum_{s \in V_S} Q''_s$ , if the latency  $\delta^{lat}(u, v) = 0$  for all  $u, v \in V$  under the constraints (1) and (2).

The DM could use the client delay  $Q'_c$  or the server delay  $Q''_s$ , depending on his preferences: the admin-

istrator of a public caching infrastructure would use the client delay, while the administrator of a CDN would use the server delay. As will be described in the next section, the DSS should make it possible for the DM to exclude certain criteria from optimization or to allow them to be optimized, but without allowing them a great influence on the solution.

From the criteria related to network operating cost, the total traffic that flows over the network of the DM can be expressed by the formula:

$$Q_b = \sum_{s \in V_S} \sum_{c \in V} z_c^s \sum_{w \in V_R \cup \{s\}} P_{cw}^s (\delta''(w, c) + (1 - h_c^s) \delta''(s, w))$$

where  $\delta''(u, v)$  is the total link cost of a path  $\pi_{uv}$ , as described in section 3.

A number of other criteria could be added to the ones described above, depending on the type of content which will be distributed by the caches or on specific DM preferences. For example, for the distribution of streaming media caches could form an overlay network that implements application-layer multicast. As an example of criteria induced by DM preferences, consider the subject of searching for fair solutions: a measure of inequality could then be added as a criterion. For examples of model extensions of the CLP, the reader is referred to [13].

We have now expressed the basic DM preferences of cache installation cost, user delay, and network operating cost as criteria which depend on the decision variables of the model. Let all of these criteria be denoted by a vector  $\mathbf{Q}$ . It remains to formulate an objective function that would be composed of these criteria.

The first step is to establish a common scale of comparison of all the criteria. To do so, we need to find the ranges in which each of the criteria can vary. It is clear that these ranges are the interval from the "best" value of a criterion to its "worst" value. Therefore, the model should be solved for each of

the criteria separately, obtaining the vector  $\mathbf{q}^I$  of the "best" values. The vector  $\mathbf{q}^N$  of "worst" values can also be obtained from the results of optimization of each individual criterion. For each criterion, the "worst" value is chosen from the values of that criterion when any other criterion was optimized.

The subsequent steps that construct an objective function from the criteria depend on the how the model lets the DM express the relative importance of his preferences. This, in turn, is the subject of the next section, which deals with how the model can be structured to support the decision process of the CLP.

## 5 Supporting the decision process of the CLP

One of the simplest methods for aggregating scaled criteria into an objective function is the *weighted sum*. The DM expresses the relative importance of preferences using a vector of weights  $\mathbf{w}$ . The weights are multiplied by the scaled criteria, where the scaling uses the "best" and "worst" values obtained during single-criteria optimization. The scaling used depends on whether criteria are maximized or minimized, because the scaled criteria are always maximized. For the rest of this section, we shall assume that all criteria are minimized, which is true for all the described criteria of the CLP. Then, the weighted sum objective function is given by:

$$Q^{WS} = \sum_i \mathbf{w}_{(i)} \frac{|\mathbf{q}_{(i)}^N - \mathbf{Q}_{(i)}|}{|\mathbf{q}_{(i)}^N - \mathbf{q}_{(i)}^I|} \quad (3)$$

If the weighted sum is used, the final model of the decision problem of cache location can be formulated as follows:

$$\begin{aligned} & \text{Maximize} && (3) \\ & \text{Subject to} && (1), (2) \end{aligned}$$

Other methods from the theory of multi-criteria op-

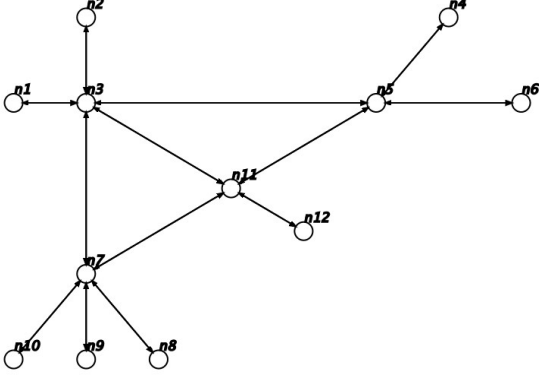


Figure 3: A simple topology.

timization is *goal programming* and the *reference point method* [15]. We shall discuss the latter approach in more detail.

The reference point method uses a different scaling function (a function that produces a vector of scaled criteria) than the weighted sum. This scaling function  $\sigma(\mathbf{Q}, \bar{\mathbf{q}}, \bar{\bar{\mathbf{q}}})$  takes as parameters two vectors called also *reference points*:  $\bar{\mathbf{q}}$ , the vector of *aspiration levels* and  $\bar{\bar{\mathbf{q}}}$ , the vector of *reservation levels*. The aspiration levels represent values of the criteria that should be achieved, while the reservation levels represent values that should be avoided. The formulas for the scaling function  $\sigma$  shall be given below; first, we shall express the objective function of the reference point method using the scaling function.

$$Q^{RP} = \min_i \sigma_i(\mathbf{Q}_{(i)}, \bar{\mathbf{q}}_{(i)}, \bar{\bar{\mathbf{q}}}_{(i)}) + \varepsilon \sum_i \sigma_i(\mathbf{Q}_{(i)}, \bar{\mathbf{q}}_{(i)}, \bar{\bar{\mathbf{q}}}_{(i)}) \quad (4)$$

The parameter  $\varepsilon$  is a small, positive number, usually 1%. If  $\varepsilon$  would be too large, the reference point method would become similar to the weighted sum. The scaling function is given by the formula:

$$\sigma_i = \begin{cases} 1 + \alpha \frac{\bar{\mathbf{q}}_{(i)} - \mathbf{Q}_{(i)}}{\bar{\mathbf{q}}_{(i)} - \mathbf{q}_{(i)}^I}, & \mathbf{q}_{(i)}^I \leq \mathbf{Q}_{(i)} \leq \bar{\mathbf{q}}_{(i)} \\ \frac{\bar{\bar{\mathbf{q}}}_{(i)} - \mathbf{Q}_{(i)}}{\bar{\bar{\mathbf{q}}}_{(i)} - \bar{\mathbf{q}}_{(i)}}, & \bar{\mathbf{q}}_{(i)} \leq \mathbf{Q}_{(i)} \leq \bar{\bar{\mathbf{q}}}_{(i)} \\ \beta \frac{\bar{\bar{\mathbf{q}}}_{(i)} - \mathbf{Q}_{(i)}}{\bar{\bar{\mathbf{q}}}_{(i)} - \mathbf{q}_{(i)}^N}, & \bar{\bar{\mathbf{q}}}_{(i)} \leq \mathbf{Q}_{(i)} \leq \mathbf{q}_{(i)}^N \end{cases}$$

Thus, the scaling function is piecewise linear. To express such a function using linear programming it should be concave (if it is to be maximized). Therefore the coefficients  $\alpha$  and  $\beta$  should be chosen once the DM has specified the reservation and aspiration levels. Then, the slope of the scaling function in the interval  $[\bar{\mathbf{q}}_{(i)}, \bar{\bar{\mathbf{q}}}_{(i)}]$  is known (it is equal to  $\frac{1}{|\bar{\mathbf{q}}_{(i)} - \bar{\bar{\mathbf{q}}}_{(i)}|}$ ).  $\beta$  can be chosen to be twice that slope, and  $\alpha$  – half the slope.

If the reference point method is used, the final model of the CLP can be described as:

$$\begin{aligned} & \text{Maximize} && (4) \\ & \text{Subject to} && (1), (2) \end{aligned}$$

### 5.1 Illustration of deficiency of weighted sum

To illustrate the difference between the weighted sum and the reference point method we shall use a simple example of cache location. Consider the simple network topology shown on Figure 3. All edges in the network have delays equal to 1 with the exception of  $(n_3, n_7)$  and  $(n_7, n_3)$ , which have delays equal to 2. Caches can be located at the nodes:  $V_R = \{n_3, n_5, n_7, n_{11}\}$ . There are two servers:  $V_S = \{n_1, n_9\}$ . The clients of these servers are located at the edge of the network. Their demand is given in Table 2.

Table 2: Demands in the simple topology.

	$n_1$	$n_2$	$n_4$	$n_6$	$n_8$	$n_{10}$	$n_{12}$	$\Sigma$
$n_1$	0	6	17	0	13	0	4	40
$n_9$	1	1	2	12	6	10	4	36

For this topology, all possible cache locations can be evaluated. Let us do so for  $m = 1$  and for a hit rate (equal for all demands) of  $h = 50\%$ . For each cache location, the DM has specified two criteria: the average delays for each server, that is, the functions  $Q_{n_9}''$  and  $Q_{n_1}''$ . On Figure 4, the possible solutions are plotted on a plane: the x axis shows the value of  $Q_{n_9}''$  and the y axis - of  $Q_{n_1}''$ .

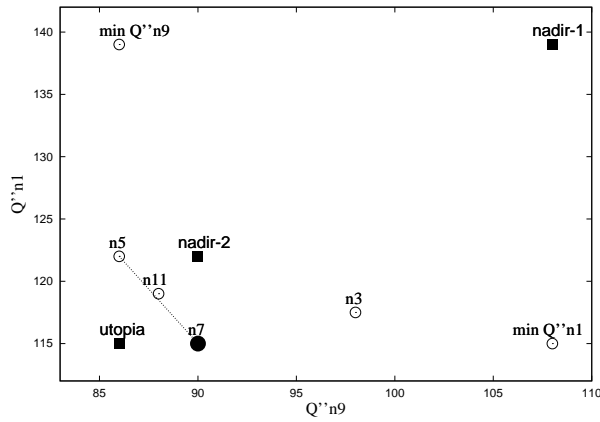


Figure 4: Results of locating 1 cache in the simple topology.

The three circles in the lower left corner of the figure are the results of locating the cache at one of the three nodes:  $n_{11}$ ,  $n_5$  and  $n_7$ . A little further to the right is a fourth circle which represents the solution obtained when the cache is located at  $n_3$ . The points labeled as "min  $Q''_{n_1}$ " and "min  $Q''_{n_9}$ " represent solutions obtained when these criteria were minimized individually. The point marked by a black circle represents the solution obtained by minimizing the weighted sum objective function with equal weights.

The solutions obtained for cache locations  $n_{11}$ ,  $n_5$  and  $n_7$  are so-called "Pareto-optimal" solutions, which means that we cannot improve the value of the delay of one server (for example,  $n_1$ ) without increasing the delay for the other server. In multi-criteria optimization, any of these points is an optimal solution to the problem. In this sense, the location of the cache at  $n_3$  is not optimal, since both criteria can be improved by moving to any of the Pareto-optimal solutions. Any of the two discussed methods – the weighted sum and the reference point method – will always result in a Pareto-optimal solution, regardless of the weights or reservation and aspiration levels specified by the DM.

By examining the Pareto-optimal solutions one sees that the solution for  $n_{11}$  lies "in between" the solutions for  $n_5$  and  $n_7$ . Which of the solutions is chosen can be a matter of DM preference – for example,

the DM may be receiving more money from the administrators of server  $n_1$  and consequently prefer to decrease the delay of that server. Let us assume as an example that the DM wishes to obtain a solution that is as fair as possible for the two criteria.

If the DM would use the weighted sum and vary the weights, he would never be able to find the solution  $n_{11}$ . The weighted sum would only find the two extreme solutions,  $n_5$  or  $n_7$ . The reason for this is that the solution  $n_{11}$  is located above the line that joins the two extreme solutions, as is shown on the figure. (Or inside the convex hull of the set of Pareto-optimal solutions). This phenomenon is called the Korhonen paradox, and a proof of the fact that the weighted sum will not find solutions inside the convex hull can be found in [15]. If the DM searches for fair solutions and uses the weighted sum with equal weights, he will find the solution that locates the cache at  $n_7$ , even though locating the cache at  $n_{11}$  is a choice that is closer to his preferences. If the DM would have other preferences than searching for fair solutions, the same phenomenon could occur – as long as there exist Pareto-optimal solutions inside the convex hull which may be closer to the DM's preferences.

On the other hand, using the reference point method the DM could find any Pareto-optimal solutions by changing the values of the reference point. The reference point method could be formulated without reservation levels (using only aspiration levels) and it would have the same property. This fact implies that the reference point method is a more precise way of searching for solutions that best fit the DM preferences. The difference in the solutions found by the weighted sum and reference point method for the same DM preferences can be arbitrarily large, and the chance that the Korhonen paradox will occur increases with the size of the problem.

Since the DM cannot specify his preferences in advance, it must be assumed that the DM will explore the space of Pareto-optimal solutions in an interactive way, by modifying weights or reference point values. In the reference point method, the reservation levels can be used by the DM to specify minimal values of certain criteria. As was indicated

in section 2, the DM may want to first specify a minimal performance level that the system should achieve, and then find a solution that fulfills that goal. He could also specify performance levels for several criteria, for example that a service provider should have a specified average delay, at the cost of locating at most one new cache, and moving existing caches. The reference point method makes that possible. Additionally, the DM could specify low aspiration values for certain criteria, indicating that these criteria have a low priority. By setting the aspiration value to the "worst" value of a criterion, the DM indicates that he will be satisfied with any solution with respect to that criterion that is better than the "worst" value. However, the criterion is still optimized. The DM could also remove the criterion entirely from consideration.

To allow the DM to explore the space of efficient solutions, the DSS should have an interactive, visual user interface. The interface does not have to visualize the location of current caches – it should, however, show the current criteria on a scale from the "worst" to the "best" value. This could be shown on a simple scale or on a graph that also plots the scaling function for that criterion (thus showing both the scaled and the unscaled value of a criterion on a single figure). The DSS could also remember the solutions that have already been explored by the DM.

## 6 Solving models for the CLP

All cache or replica location model formulations share one common feature: they use integer variables. Apart from one extension described in section 4 (the introduction of a variable hit rate into the CLP) all model formulations in this paper are linear. Therefore, a natural approach to the optimal solution of the CLP seems to be the use of Mixed Integer Linear Programming (MILP) approaches. A number of commercial or free solvers for this type of problems exist.

However, such an approach is useful only for small and medium problems. The complexity of MILP

formulations of the CLP depends on the number of non-zero demands and the size of the set of potential cache locations,  $V_R$ . Model complexity does not depend directly on the size of a topology. Additionally, the integer assignment variables for one demand are related by the second constraint of the CLP, which allows only one of them to be positive. This greatly reduces the complexity of branch-and-bound procedures. Taking this into consideration, the complexity of using MILP for cache location depends mainly on the number of demands. This number cannot be too large if the problem is to be solved optimally using MILP.

To establish the size of CLP models which can be solved optimally using MILP, a number of experiments were performed using the commercial MILP solver cplex, versions 7.1 and 6.6, on a SUN SPARC Ultra-4 with processor speed of 250MHz (not dedicated to the MILP computations). The Georgia Tech Internetwork Topology Models [16] (GT-ITM) was used to generate the network topologies. The topologies were generated from the "transit-stub" models to obtain graphs that more closely resemble the Internet than pure random construction. GT-ITM generates a transit-stub graph in stages, first a number of random backbones (transit domains), then the random structure of each backbone, then random "stub" graphs are attached to each node in the backbones. For the structure of both transit domains and stubs, the "locality" model was used. In this model, there is a low probability that an edge will connect a pair of nodes that are too far away on the plane.

The set  $V_R$  contained all nodes of a network. The demands were generated randomly in such a way that the sum of demands remained the same. The evaluation was made on 50 different topologies of each size, and for each topology we generated 10 different demand structures. For networks up to a 50 nodes and 500 demands all models were solvable in comfortable time (fast enough for an interactive system). Networks of 100 nodes and 500 demands were solvable in up to one hour.

However, certain MILP solvers (like cplex) offer the possibility of specifying an absolute tolerance

for the branch-and-bound procedure. The branch-and-bound algorithm calculates lower bounds on the value of the objective function for a solution that has certain integer variables replaced by real variables. The lower bound holds also for all integer solutions that would result from changing the real variables back into integer, therefore the branch-and-bound algorithm can use the lower bound to determine whether it should consider exploring the solutions in this particular subtree. The tolerance for the branch-and-bound procedure is a value that is used in the test on whether to branch into a subtree. To make that decision, the lower bound of the solution which is at the root of the subtree is compared with the objective value of the best known integer solution. If the difference is below the given tolerance, the solver will not consider the particular branch. Therefore the solver will always find a solution that is worse than the optimal one by at most the absolute tolerance specified. The tolerance can be used to shorten the time of calculations, while keeping the solution quality under pessimistic control.

The size of problems that can be considered with an increased tolerance is larger, but still limited by a practical concern: the size of the linear program itself. Depending on the architecture, MILP solvers have a limit on the maximal number of variables to be considered. For cplex on our platform, this number is 268435450. Even before that limit is reached, the size of the files that contain the linear program will be cumbersome. For a CLP that has 1000 demands and 1000 potential cache locations, the resulting linear program will require over a million constraints. The shortest constraint requires 7 characters, which means that the file will be at least 7000000 bytes long.

For very large problems, one of the many heuristics that have been developed for cache and replica location problems can be used. The greedy heuristic described in [8] has been shown to work well in two studies [8, 6]. However, this and similar heuristics need to be extended to take into account DM preferences. They could also easily be extended to consider non-linear models, for example the CLP with variable hit rate.

The extension of the greedy heuristic to use multi-criteria methods is straightforward. The heuristic can use the objective function (3) of the reference point method directly when it evaluates possible choices. To formulate function (3), the individual criteria must be optimized first, either by the heuristic or optimally (since for some of the criteria, only a subset of the decision variables is considered).

However, when the greedy heuristic uses the reference point objective function with  $\epsilon = 1\%$ , it does not perform well. This was also observed in [8] where the authors state that the greedy heuristic is not suitable for max-min objectives. The greedy heuristic works well for larger values of  $\epsilon$ , which implies that it performs similarly to a weighted sum (it will ignore certain Pareto-optimal solutions even if they are close to the preferences of the DM).

## 7 Conclusion

This paper considered models of the Cache Location Problem (which can also be used for the related Replica Location Problem, with small modifications) from the perspective of Decision Support Systems. This approach resulted in different models than previous work, which was mainly concerned with the computational complexity of solving models of the CLP. The paper also studied the subject of using MILP programming to express cache location and formulated the proposed models using this approach. Using the commercial solver cplex, the paper establishes sizes of problems that can be solved optimally using MILP and state-of-the-art solvers, in realistic time.

The need to take into consideration DM preferences requires a different approach to creating models of cache and replica location. The use of multi-criteria methods is natural in this context. As long as a linear model formulation is preserved, off-the-shelf commercial MILP solvers can be used to find optimal solutions for small and medium model instances. For larger problems, one of the many heuristics studied in the literature can be used, once it is modified to take into account DM preferences

and multi-criteria methods. The choice of an appropriate heuristic that is suitable to the reference point method is the subject of future work. Heuristics can also be extended to more realistic model variants, for example with a variable hit rate.

A Decision Support System for cache and replica location could be constructed using state-of-the-art results of research in the field of cache and replica location. Such a system would contain a module for gathering and visualizing information about the network of a DM, which could be useful by itself. The capability of planning cache location, evaluating and improving existing caching systems would not be difficult to add to such a system, once the problems of distributed collection of required information have been overcome. The DSS could also be used by the DM to demonstrate to customers of a CDN that they are receiving service that conforms to their Service Level Agreement.

## 8 Acknowledgments

The author wishes to thank the anonymous reviewers for their contribution, which have been very valuable.

## References

- [1] L. Dowdy and D. Foster. Comparative models of the file assignment problem. *Computing Surveys*, 14, 1982.
- [2] B. Duska, D. Marwood, and M. Feeley. Measured access characteristics of world-wide-web client proxy caches. *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.
- [3] A. Wolman et al. On the scale and performance of cooperative web proxy caching. *Operating Systems Review*, 34, 1999.
- [4] C. Lund et al. Competitive on-line algorithms for distributed data management. *SIAM J. on Computing*, 28, 1999.
- [5] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 1999.
- [6] M. Karlsson, C. Karamanolis, and M. Mahalingam. A unified framework for evaluating replica placement algorithms. Technical report, Hewlett-Packard Laboratories, 2002.
- [7] J. Kollias and M. Hatzopoulos. The file assignment problem. *Computing Surveys*, 15, 1983.
- [8] P. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transactions on Networking*, October 2000.
- [9] Carnegie Mellon University Pittsburgh Supercomputing Center (PSC). About the psc treno server. World Wide Web page, [http://www.psc.edu/networking/treno\\_info.html](http://www.psc.edu/networking/treno_info.html), 2000.
- [10] L. Qiu, V. Padmanabhan, and G. Voelker. On the placement of web server replicas. *Proceedings of IEEE Infocom*, 2001.
- [11] P. Radoslavov, R. Govindan, and D. Estrin. Topology-informed internet replica placement. *Proc. of the Sixth International Workshop on Web Caching and Content Distribution*, 2000.
- [12] J. Wang. A survey of web caching schemes for the internet. *ACM Computer and Communication Review*, pages 36–46, 1999.
- [13] A. Wierzbicki. Internet cache location and design of content delivery networks. In *Proc. International Workshop on Web Engineering, Networking'2002*. Lecture Notes on Computer Science, Springer Verlag, 2002.
- [14] A. Wierzbicki, M. Kurcewicz, and W. Sylwestrzak. Filtering algorithms for proxy caches. *Computer Networks and ISDN Systems*, 1998.

- [15] A. P. Wierzbicki, M. Makowski, and J. Wessels, editors. *Model-Based Decision Support Methodology with Environmental Applications*. Kluwer Academic Publishers, 2000.
- [16] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proc. IEEE Infocom*, pages 40–52, 1996.