OR Spectrum
DOI 10.1007/s00291-013-0358-z

REGULAR ARTICLE

The dynamic replica placement problem with service levels in content delivery networks: a model and a simulated annealing heuristic

Rainer Kolisch · André Dahlmann

Abstract Dynamically assigning copies (replicas) of internet files to regionally placed servers is a crucial planning problem for content delivery network providers. Their goal is to minimize placement, storage and delivery costs while fulfilling service-level agreements. Although there is considerable work on simplified versions of this problem such as the static case, the dynamic problem with all three cost types and service-level constraints has yet not been considered in the literature. This paper provides a mixed integer programming (MIP) formulation for the problem as well as a simulated annealing metaheuristic. A computational study is employed to assess the simulated annealing heuristic and the MIP employing a state-of-the-art solver.

Keywords Content delivery network · Content distribution network · Mixed integer program · Metaheuristic · Simulated annealing

1 Introduction

Content delivery networks (CDN) (see Verma 2002; Rabinovich and Spatschek 2002; Dilley et al. 2002; Bartolini et al. 2003; Pathan et al. 2008) enable quick delivery of content such as web pages, video or audio files through the internet. The main idea of CDN is, instead of having only one server, the so-called origin server, to copy (replicate) the data to several servers in the internet. Each customer request for content is handled from the nearest servers with a copy (replica) such that delivery

R. Kolisch (✉) · A. Dahlmann
TUM School of Management, Technische Universität München,
Munich, Germany
e-mail: rainer.kolisch@tum.de

A. Dahlmann
e-mail: andre.dahlmann@wi.tum.de

Published online: 12 January 2014

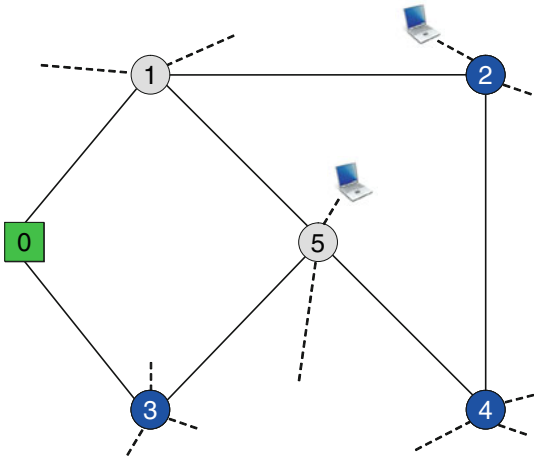

Fig. 1 Simplified structure of an overlay CDN

time is minimized. Supplying demand not only from one but several servers and using not one but a variety of routes in the internet fosters quick delivery times. CDN are mainly used by companies, so-called content providers, with a high and geographically distributed demand for their web contents. Typically, the CDN is not operated by the content provider but by specialized service companies, so-called CDN providers, such as Akamai. By using a network of servers (an overlay network on the internet, see Verma 2002; Rahul et al. 2008), CDN providers ensure that the web pages of the content providers can be accessed quickly by end-users (clients) throughout the world. A contract between a content provider and a CDN provider includes a service-level agreement (SLA) to ensure a guaranteed quality of service (QoS). The SLA prescribes that a given percentage of all requests has to be handled within a given time span, the permitted latency. For an existing CDN the provider has to decide which files are stored when and on which servers, taking into account the frequency, timing and geographical distribution of requests. An optimal placement has minimum costs while fulfilling the service-level agreements. Relevant costs are storage, placement and delivery costs. Since we assume that the content does not change over the planning horizon, we do not have to take into account update costs.

Let us explain a CDN with the example depicted in Fig. 1, where nodes represent servers, edges depict interconnections between servers and the entire graph forms the overlay network. We assume that all edges have a unit length of 1. The dotted edges are connections to nodes outside the CDN. The square node 0 represents the origin server of the content provider; the circle nodes depict replica servers where replicas can be placed. The two stylized laptops represent clients where each client demands a single unit of content. Each client's request is always aggregated at the nearest CDN server. Inside the CDN each request is then redirected to the nearest servers which stores a replica of the data (see Dilley et al. 2002; Bartolini et al. 2003; Vakali and Pallis 2003; Pallis and Vakali 2006). Let us consider a period where the content of server 0 is replicated at servers 2, 3 and 4. We then have storage costs which accrue for the servers 2, 3 and 4. Next, we consider the delivery of the content to the request

at servers 2 and 5. In order to fulfill each request as quickly as possible, the shortest path from a server with a replica to the request server is calculated (see Krishnan et al. 2000; Jamin et al. 2000, 2001; Korupolu et al. 2001; Karlsson and Mahalingam 2002; Chen et al. 2003; Aioffi et al. 2004; Tang and Xu 2005; Nguyen et al. 2005; Jeon et al. 2006). Such an individual fulfillment of demand, even if two or more requests from the same client arrive at the same replica server (see Cidon et al. 2002), is called a "unicast" approach. With unicast the request at server 2 will be served from server 2 itself and the request at server 5 will be served from server 3 or 4. The total distance is then 1. Finally, we have to consider the placement of the content on the replica servers. Let us assume that the content of the origin server 0 has to be transferred to the replica servers 2, 3 and 4. In a unicast approach, three transfers using the shortest paths from the origin server to each chosen replica server would be performed. In contrast, in a so-called "multicast" approach point-to-multipoint connections can be employed (see Frank et al. 1985; Paul 1998; Wittmann and Zitterbart 1999), where a used edge generates costs only once, no matter how often it is used. The object can be copied on every passed replica server and then be forwarded several times. In our example, the optimal multicast route is $0 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 2$ with a total distance of 4 in contrast to the three optimal unicast routes $0 \rightarrow 3$, $0 \rightarrow 3 \rightarrow 5 \rightarrow 4$ and $0 \rightarrow 1 \rightarrow 2$ with a total distance of 6. With a multicast approach the data flow during the placement uses edges in a subtree of the network. As in our example, this tree is typically different from the set of individual shortest paths. The multicast approach leads to a considerable reduction of placement costs (see Wolfson and Milo 1991).

In this paper, we consider the dynamic planning problem of the CDN provider where, given a forecast for the multi-period demand, the CDN provider has to decide for every period of the planning horizon where to place replicas, how to send the replicas via multicast to the chosen servers and how to deliver the content via unicast from the servers to the clients. We consider a single object to be content. An object represents the complete content of the customer, such as the web page including text, video and audio files. The remainder of this paper is organized as follows. In Sect. 2, we provide an overview of the relevant literature and detail how our work differs from the work done thus far. In Sect. 3, we present an MIP formulation of the dynamic replica placement problem in CDN with service-level constraints. We discuss how the problem relates to other known optimization problems and show that the problem is NP-hard. Moreover, we propose a lower bound which is tighter than the LP relaxation. Section 4 is devoted to a simulated annealing heuristic to solve the optimization problem. Section 5 presents a computational study where we employ a realistic set of test instances in order to gauge the simulated annealing heuristic as well as the solution of the MIP with different settings of the state-of-the-art solver CPLEX. Finally, we conclude with a summary in Sect. 6.

2 Literature review

The first treatment of a problem similar to the replica placement in content distribution networks is the classical file allocation problem (FAP, see Chu 1969; Dowdy and Foster 1982). The FAP considers the allocation of file copies in a distributed filesystem or

the assignment of file copies to storage nodes in a computer network. Later research focuses on web proxies and web caches which can be seen as the origin of content delivery networks (see Baentsch et al. 1997; Rabinovich and Spatschek 2002; Pathan et al. 2008). Typical problems are the placement of web proxies or web caches in the internet to reduce overall traffic in the network or access latency (see Li et al. 1999; Krishnan et al. 2000; Korupolu et al. 2001). First approaches assume very simple networks like line, ring, tree or other hierarchical topologies. Jamin et al. (2000) treat a placement problem with general topologies. Later, the work is generalized to the placement of web server replicas in CDNs by Qiu et al. (2001) and Jamin et al. (2001). Radoslavov et al. (2002) undertake a slightly refined study based on the work of Jamin et al. (2001). All these approaches assume a given number of replicas and address some variant of the minimum k-median (Li et al. 1999; Krishnan et al. 2000; Qiu et al. 2001) or the minimum k-center problem (Jamin et al. 2000, 2001) in order to minimize latency or bandwidth consumption. Hence, the number of replicas is not optimized. The first to minimize storage and delivery costs are Cidon et al. (2002). While they restrict the topology to a hierarchy tree, Kangasharju et al. (2002) extend the problem to general topologies. They consider multiple objects, that is, multiple customers, each with one object, and thus limited storage space on each server. However, they neither minimize the number of replicas nor incorporate placement costs. The objective is the minimization of the average distance a request must traverse. One of the first approaches towards QoS oriented content delivery is by Chen et al. (2002). For specific topologies and a single period, they minimize the number of replicas while meeting capacity constraints of the servers (load, bandwidth, storage) and latency constraints of the clients. This ensures that a maximum latency for the clients is met. To the best of our knowledge, Karlsson and Karamanolis (2003) are the first to model a dynamic, i.e., a multi-period version of the problem. Their model minimizes storage and placement costs for multiple objects, general topologies and specific guaranteed QoS-constraints. The drawback of their approach is the assumption of unit costs for storage and placement. Although Karlsson and Karamanolis (2003) report that the replica creation costs represent the costs of network resources used to create a replica, in their model these costs do not depend on the distance between origin and copy and not on the bandwidth used. Moreover, the delivery costs are not considered. In a successor paper, Karlsson and Karamanolis (2004) constrain the average latency to be less than or equal to a specific threshold. However, due to averaging, the approach cannot provide performance guarantees as required by SLAs. Tang and Xu (2005) also concentrate on QoS-requirements and add performance guarantees. They minimize storage costs and update costs which accrue if the data on the origin server changes. They neither consider the multi-period case nor the multi-object case. Aioffi et al. (2004) also consider the multi-period case. Their objective function minimizes the total traffic within the overlay network which is generated by delivery to the client, replica placement and replica update. Storage costs and service levels are not considered.

In contrast to the work done so far, we present a model for the simultaneous selection of replica servers, the transfer to the servers (placement) via multicast and the transfer from the servers to the clients (content delivery) via unicast. We consider service-level agreements and the dynamic case (as treated in Karlsson and Karamanolis 2003, 2004). While a number of papers such as Kalpakis et al. (2001), Xu et al. (2002),

Jia et al. (2003), and Unger and Cidon (2004), make similar assumptions about the multicast transfer as our paper, we provide the first mixed integer program which explicitly models multicast transfers. Unlike Tang and Xu (2005) we do not restrict the placement to a fixed distribution tree. In contrast to Li et al. (1999), Krishnan et al. (2000), Cidon et al. (2002), and Chen et al. (2002), our model is suited to any network topology and in contrast to Li et al. (1999), Krishnan et al. (2000), Jamin et al. (2000, 2001) and Qiu et al. (2001), we do not restrict the number of replicas. In order to solve this model, we propose a simulated annealing heuristic. We are the first to develop a metaheuristic for replica placement problems in content delivery networks. We are only aware of optimal algorithms for specific topologies (see Li et al. 1999; Krishnan et al. 2000; Cidon et al. 2002) or simple heuristics for more general topologies (see Jamin et al. 2001; Qiu et al. 2001; Radoslavov et al. 2002; Kangasharju et al. 2002; Aioffi et al. 2004; Tang and Xu 2005).

3 Model

3.1 Assumptions

In order to model the replica placement problem in content delivery networks, we make some assumptions. First, we consider only one object as in Li et al. (1999), Krishnan et al. (2000), Jamin et al. (2000, 2001), Qiu et al. (2001), Cidon et al. (2002), Chen et al. (2002), Tang and Xu (2005), and Jeon et al. (2006), such as the web page of a single customer. Due to this assumption we do not have to model scarce storage capacities of the servers and bandwidth of the edges. Consequently, the storage costs and the placement costs in our model are mainly opportunity costs for the use of scarce storage capacity (see Aggarwal and Rabinovich 1998; Jamin et al. 2001; Shi and Turner 2002; Chen et al. 2003; Tang and Xu 2005; Jeon et al. 2006) and limited bandwidth. Next, we assume that the content of the origin server does not change during the planning horizon. Due to this, we do not have to undertake an update of the placement due to changed content. Furthermore, we assume that the costs of the transfers are proportional to the length of the used path. We thus weight the network distances with cost coefficients for placement and delivery to obtain the corresponding cost values. In reality, it is the number of autonomous systems traversed (AS-hops, see Kangasharju et al. 2002), and not the distance, which determines the costs. However, there is a high correlation between the distance and the number of AS-hops. In general, our model can be easily adapted to any distance metric. Finally, we assume that deleting an object from a replica server can be done at zero cost (see Karlsson and Karamanolis 2004).

3.2 Building blocks

Network topology We consider a general network topology representing the CDN overlay network. Data flow is possible in both directions towards the edges of the network. As we need directed edges to model the problem, we substitute each undirected edge with two arcs with opposite directions. Let $\mathcal{V}$ be the set of nodes, $V := |\mathcal{V}|$, and

let $\mathcal{A}$ be the set of arcs of the network. $\delta_{i,j}$ is the weight of arc $(i, j) \in \mathcal{A}$ which gives the distance between the nodes.

Replica storage One node $v_0 \in \mathcal{V}$ represents the origin server, the one and only server which stores the original object in the dummy starting period $t = 0$. $\mathcal{V}^{\text{server}} \subseteq \mathcal{V}\backslash\{v_0\}$ is the set of replica servers, which are part of the content delivery network and can store a replica of the object. The cost for storing a replica in one period on a server is α. For an easier notation of the model, we require that the replica servers of the CDN $s \in \mathcal{V}^{\text{server}}$ form a connected subgraph in the topology. This typically holds for real CDN topologies because of the overlay network characteristic.

Replica placement We assume that the placement is done with infinite speed at the beginning of each period $t \in \mathcal{T}$. $\mathcal{T}$ denotes the set of all periods within the planning horizon and $T := |\mathcal{T}|$ denotes the number of periods. A replica server can receive a replica of the object in period t only from a replica server which stores a replica in period $t - 1$ or from the origin server v_0. The placement is done via multicast. Hence, the optimal routes are not obvious and we need to model the placement flow. In order to ease the notation of the model, we define two additional sets of nodes:

$$\mathcal{V}^{\text{pred}}(s) := \{i \in \mathcal{V}^{\text{server}} \cup \{v_0\} \mid (i, s) \in \mathcal{A}\}$$
$$\mathcal{V}^{\text{succ}}(s) := \{j \in \mathcal{V}^{\text{server}} \cup \{v_0\} \mid (s, j) \in \mathcal{A}\}$$

$\mathcal{V}^{\text{pred}}(s)$ is the set of immediate predecessor nodes of node s in the overlay network. $\mathcal{V}^{\text{succ}}(s)$ denotes the set of immediate successor nodes of node s in the overlay network.

Content delivery Let $\mathcal{V}^{\text{client}} \subseteq \mathcal{V}\backslash\{v_0\}$ be the set of clients which can request an object. The number of requests of a client $c \in \mathcal{V}^{\text{client}}$ in period $t \in \mathcal{T}$ is denoted as $r_{c,t}$. Each replica server $s \in \mathcal{V}^{\text{server}}$ has a load capacity C_s^L which is the maximum number of requests the server can process per period.

The service-level agreement is defined by the tuple (λ, q). In each period, all requests for the object have to be served. With q we denote the maximum latency to serve a request. λ defines the fraction of all requests to be served within q. The total number of requests $r_{c,t}$ of client c in period t can be split among several replica servers. A single request cannot be split. For example, assume that for the network of Fig. 1, replicas are placed on server 2, 3 and 4 and each server has a capacity of $C_s^L = 2$. Server 2 and 5 have requests of $r_c = 3$ each. A solution is then to serve the request of server 3 with two units from server 3 and one unit from server 4 while the request of server 2 is served with two units from server 2 and one unit from server 4.

For the corresponding constraints, we denote with $\mathcal{V}^{\text{server}}(c, q) \subseteq \mathcal{V}^{\text{server}}$ the set of replica servers where each server s can serve a request of client c within latency q if it stores a replica of the object. That is, in the overlay network there is for each $s \in \mathcal{V}^{\text{server}}(c, q)$ a shortest path with length $d_{c,s} \leq q$ from server s to client c which can be used for the delivery.

Variables For replica placement we employ the binary decision variable $x_{s,t}$ which equals 1 if server $s \in \mathcal{V}^{\text{server}}$ stores a replica of the object in period $t \in \mathcal{T}$ and 0 otherwise. To model the placement we denote with $0 \leq w_{s,t} \leq 1$ the variable which indicates for $w_{s,t} = 1$ that replica server $s \in \mathcal{V}^{\text{server}}$ receives a replica at the beginning

of period $t \in \mathcal{T}$. Note that $w_{s,t}$, although modeled as a continuous variable, will always be an integer due to the constraints of the model. The realized routes of the placement are modeled with the variable $z_{i,j,t} \in \{0, 1\}$. $z_{i,j,t} = 1$ indicates that arc $(i, j) \in \mathcal{A}$ is used for the placement in the beginning of period $t \in \mathcal{T}$. The variable $\tilde{z}_{i,j,t} \geq 0$ counts how many times arc $(i, j) \in \mathcal{A}$ is used for the placement with multicast in period $t \in \mathcal{T}$. Finally, for modeling the delivery to the clients we employ $y_{c,s,t} \geq 0$ which is the fraction of all requests from client $c \in \mathcal{V}^{\text{client}}$ in period $t \in \mathcal{T}$ which are served by replica server $s \in \mathcal{V}^{\text{server}}$. Due to the constraints the absolute number of requests $y_{c,s,t} \cdot r_{c,t}$ of client c served by replica server s in period t will always be an integer as long as $r_{c,t}$ is integer. Table 3 in the Appendix A provides a summary of the notations.

3.3 MIP formulation

We can now model the following mixed-binary linear program for the Dynamic Replica Placement with Service Level constraints (DRPSL):

$$\begin{aligned}\text{Min}\, \alpha \sum_{s \in \mathcal{V}^{\text{server}}} \sum_{t \in \mathcal{T}} x_{s,t} + \beta \sum_{(i,j) \in \mathcal{A}} \sum_{t \in \mathcal{T}} \delta_{i,j} \cdot z_{i,j,t} \\ + \gamma \sum_{c \in \mathcal{V}^{\text{client}}} \sum_{s \in \mathcal{V}^{\text{server}}} \sum_{t \in \mathcal{T}} d_{c,s} \cdot r_{c,t} \cdot y_{c,s,t} \qquad (1)\\ \text{subject to}\end{aligned}$$

$$\sum_{c \in \mathcal{V}^{\text{client}}} r_{c,t} \cdot y_{c,s,t} \leq C^L_s \cdot x_{s,t} \quad \forall s \in \mathcal{V}^{\text{server}},\ \forall t \in \mathcal{T} \qquad (2)$$

$$\sum_{s \in \mathcal{V}^{\text{server}}} y_{c,s,t} = 1 \quad \forall c \in \mathcal{V}^{\text{client}},\ \forall t \in \mathcal{T} \qquad (3)$$

$$\sum_{c \in \mathcal{V}^{\text{client}}} \sum_{s \in \mathcal{V}^{\text{server}}(c,q)} r_{c,t} \cdot y_{c,s,t} \geq \lambda \cdot \sum_{c \in \mathcal{V}^{\text{client}}} r_{c,t} \quad \forall t \in \mathcal{T} \qquad (4)$$

$$w_{s,t} \geq x_{s,t} - x_{s,t-1} \quad \forall s \in \mathcal{V}^{\text{server}},\ \forall t \in \mathcal{T} \qquad (5)$$

$$\sum_{i \in \mathcal{V}^{\text{pred}}(s)} \tilde{z}_{i,s,t} - w_{s,t} \leq \sum_{j \in \mathcal{V}^{\text{succ}}(s)} \tilde{z}_{s,j,t} \quad \forall s \in \mathcal{V}^{\text{server}},\ \forall t \in \mathcal{T} \qquad (6)$$

$$\sum_{i \in \mathcal{V}^{\text{pred}}(s)} \tilde{z}_{i,s,t} - w_{s,t} + M \cdot x_{s,t-1} \geq \sum_{j \in \mathcal{V}^{\text{succ}}(s)} \tilde{z}_{s,j,t} \quad \begin{matrix}\forall s \in \mathcal{V}^{\text{server}} \cup \{v_0\}, \\ \forall t \in \mathcal{T}\end{matrix} \qquad (7)$$

$$M \cdot z_{i,j,t} - \tilde{z}_{i,j,t} \geq 0 \quad \forall (i,j) \in \mathcal{A},\ \forall t \in \mathcal{T} \qquad (8)$$

$$x_{v_0,t} = 1 \quad \forall t \in \mathcal{T} \cup \{0\} \tag{9}$$

$$x_{s,0} = 0 \quad \forall s \in \mathcal{V}^{\text{server}} \tag{10}$$

$$w_{v_0,t} = 0 \quad \forall t \in \mathcal{T} \tag{11}$$

$$x_{s,t} \in \{0,1\} \quad \begin{array}{l} \forall s \in \mathcal{V}^{\text{server}} \cup \{v_0\}, \\ \forall t \in \mathcal{T} \end{array} \tag{12}$$

$$0 \leq w_{s,t} \leq 1 \quad \begin{array}{l} \forall s \in \mathcal{V}^{\text{server}} \cup \{v_0\}, \\ \forall t \in \mathcal{T} \end{array} \tag{13}$$

$$z_{i,j,t} \in \{0,1\} \quad \forall (i,j) \in \mathcal{A}, \ \forall t \in \mathcal{T} \tag{14}$$

$$\tilde{z}_{i,j,t} \geq 0 \quad \forall (i,j) \in \mathcal{A}, \ \forall t \in \mathcal{T} \tag{15}$$

$$y_{c,s,t} \geq 0 \quad \begin{array}{l} \forall c \in \mathcal{V}^{\text{client}}, \\ \forall s \in \mathcal{V}^{\text{server}}, \ \forall t \in \mathcal{T} \end{array} \tag{16}$$

The objective function (1) minimizes the sum of three cost terms. First, the storage costs α for each replica server and period if the server stores a replica of the object. Second, the costs for the placement which is for each period the sum of the lengths $\delta_{i,j}$ of all arcs (i, j) used in the multicast placement multiplied by cost coefficient β. Third, the costs for the delivery to the clients which is for every period the cost coefficient γ multiplied by the sum of the distances of all paths used for the transfers.

Constraints (2) ensure that each replica server in each period can only serve requests if it stores a replica of the object. Additionally, the load capacity C_s^L is defined as the maximum number of requests the server s can handle per period. Due to constraints (3) all requests of each client in each period have to be served. Constraints (4) ensure that in each period the service-level agreements are fulfilled, i.e., at least the fraction λ of all requests is sent from replica servers in the range of the maximum latency q.

The variable $w_{s,t}$ is set by constraints (5). We need it to model the multicast data flow of the placement in each period. The latter is done by the flow conservation constraints (6) and (7) which cover the following three cases.

First, if the considered server $s \in \mathcal{V}^{\text{server}}$ neither stores a replica in period $t-1$ nor in period t (i.e., $x_{s,t-1} = x_{s,t} = 0$), it follows through constraints (5) that $w_{s,t} = 0$. In this case (6) is reduced to

$$\sum_{i \in \mathcal{V}^{\text{pred}}(s)} \tilde{z}_{i,s,t} \leq \sum_{j \in \mathcal{V}^{\text{succ}}(s)} \tilde{z}_{s,j,t},$$

(7) is reduced to

$$\sum_{i \in \mathcal{V}^{\text{pred}}(s)} \tilde{z}_{i,s,t} \geq \sum_{j \in \mathcal{V}^{\text{succ}}(s)} \tilde{z}_{s,j,t}$$

and thus we have

$$\sum_{i \in \mathcal{V}^{\text{pred}}(s)} \tilde{z}_{i,s,t} = \sum_{j \in \mathcal{V}^{\text{succ}}(s)} \tilde{z}_{s,j,t}.$$

For server s the outflow equals the inflow, i.e., the server is just a transshipment point. Server s has not stored a replica in the preceding period and therefore cannot copy the object and send it to other servers on its own. As server s, furthermore, does not store a replica in period t, the object it receives is forwarded to other servers.

Second, if server $s \in \mathcal{V}^{\text{server}}$ has not stored a replica in period $t-1$ and receives a replica in period t, we have $x_{s,t-1} = 0$, $x_{s,t} = 1$ and due to constraints (5), $w_{s,t} = 1$. In this case constraints (6) and (7) set the outbound flow equal to the inbound flow minus 1 unit which is the unit stored on the server.

Third is the case if $x_{s,t-1} = 1$ and through constraints (5) $w_{s,t} = 0$. The server has a replica in period $t-1$ and thus can send copies to other servers in period t. Independently of $x_{s,t}$, constraints (6) force the outbound flow greater than or equal to the inbound flow while constraints (7) do not restrict the outbound flow in this case. Instead, the additional summand M allows the server to send copies of the object even without an inbound flow. Note that constraints (7) hold not only for the replica servers $s \in \mathcal{V}^{\text{server}}$, but also hold for the origin server v_0. Hence, constraints (7) allow the sending of copies from the origin server to replicate the object. It is easy to see that it is sufficient to set $M = |\mathcal{V}^{\text{server}}|$ in constraints (7) as this is an upper bound for the number of outgoing copies. This could be the case if only the origin server v_0 has the object and all replica servers receive a replica.

If for the replica placement there is a data flow $\tilde{z}_{i,j,t} > 0$ on arc (i, j) in period t, constraints (8) enforce $z_{i,j,t} = 1$ which indicates that arc (i, j) is employed for the multicast placement.

Constraints (9) assure that for each period the object is on the origin server while constraints (10) enforce that replica servers do not store a replica of the object at the beginning of the planning horizon. Constraints (11) set the w-variables for the origin server and all periods to 0 which is necessary to model the flows correctly.

The decision variables are defined in (12), (13), (14), (15) and (16).

3.4 Discussion

We now want to discuss some special cases of the DRPSL. If we set $\beta = \gamma = 0$ and $\alpha > 0$, the objective function (1) considers storage costs only. The goal is then to store as few replicas as possible while still meeting the demand under the service-level constraints (4). In this case, the model decomposes into T independent submodels. Hence, the model for each period t can be considered separately. If we additionally set $\lambda = 1$, i.e., the maximum latency holds for all requests, the problem can be formulated as a set covering problem (see Krarup and Pruzan 1983; Shi and Turner 2002).

By setting $\alpha = \beta = 0$ and $\gamma > 0$, we are minimizing only delivery costs. In this case, the optimal solution is trivial. We place a replica on every replica server in each period.

The consideration of storage costs and delivery costs only, i.e., $\alpha, \gamma > 0$ and $\beta = 0$, leads to a generalized facility location problem with service-level constraints. Constraints (5), (6), (7) and (8) are dropped. Constraints (2) and (3) are analogous to the constraints in a Capacitated Facility Location Problem (CFLP) that deal with the capacity of a facility and satisfy the demand of each customer (see Aikens 1985). Constraints (4) can be seen as the maximum distance between each facility and its assigned customers which has to be met for a certain percentage of all customers. As

there are no costs for deleting a replica, DRPSL also decomposes into one independent subproblem for each period.

When considering only storage and placement costs, i.e., setting $\alpha, \beta > 0$ and $\gamma = 0$, we search for a 'stable' placement with as few replicas as possible while serving the demand and fulfilling the service levels. A 'stable' placement means in this context a placement with as few changes as possible over time. This is due to the second summand of the objective function, the placement costs. Consecutive periods are connected by the placement costs.

3.5 Model complexity

In what follows, we show that the DRPSL can be restricted to two well-known NP-hard optimization problems, the uncapacitated facility location problem and the Steiner tree problem in graphs.

The uncapacitated facility location problem The DRPSL can be restricted to an uncapacitated facility location problem by setting $|\mathcal{T}| = 1$, $\beta = 0$, $\gamma = 1$, $C_s^L \equiv M$ and $\lambda = 0$. The set $\mathcal{V}^{\text{server}}$ corresponds to the set of possible locations for the facilities. Storing a replica on a replica server corresponds to opening a facility.

The Steiner tree problem in networks We restrict the DRPSL to the Steiner tree problem using the flow formulation of Voss (1990) by allowing only instances with $|\mathcal{T}| = 1$, $\alpha = 0$, $\gamma = 0$, $r_{c,t} \equiv 0$ and $\lambda = 0$. Then, we set $x_{s,t} = 1$ for all terminal nodes, i.e., the set of nodes that need to be connected by the Steiner tree. For one of these terminal nodes we set $x_{s,t-1} = 1$ as the 'origin' of the flow in the preceding period. Note that the solution of the Steiner tree problem is independent of the chosen starting node.

Proposition 1 *The DRPSL is NP-hard.*

Proof The uncapacitated facility location problem (see Krarup and Pruzan 1983) and the Steiner tree problem in graphs (see Karp 1972) are both known to be NP-hard. Since the uncapacitated facility location problem and the Steiner tree problem in graphs are special cases of the DRPSL, the latter has to be NP-hard. □

3.6 Lower bounds

Since the DRPSL is an NP-hard optimization problem, we are interested in a good lower bound in order to speed up branch-and-bound procedures or to calculate solution gaps for heuristics.

A straightforward LP-relaxation relaxes the binary decision variables $x_{s,t}$ and $z_{i,j,t}$ to be in [0, 1]. However, this relaxation provides bad lower bounds because the placement variable $x_{s,t}$ is very fragmented, i.e., many servers store a small fraction of the object. This has two effects. First, only a very low flow is necessary for the placement, especially as $z_{i,j,t}$ is also relaxed. Second, there are almost no delivery costs because each server c with a request in period t will store a fragment $0 < x_{x,t} \ll 1$. Thus, the absolute cost values for network usage in the objective function, i.e., the placement costs and delivery costs, are very low and far from realistic.

We improve the standard LP-relaxation by extending the model to the disaggregated or 'strong' formulation familiar from the facility location problem (see Krarup and Pruzan 1983). In order to do so, we add constraints (17) to the DRPSL.

$$y_{c,s,t} \leq x_{s,t} \qquad \forall c \in \mathcal{V}^{\text{client}},\ s \in \mathcal{V}^{\text{server}},\ t \in \mathcal{T} \tag{17}$$

Constraints (17) state for each period t and each client c that the client can only be supplied by a replica server s if the latter stores the object, i.e., $x_{s,t} = 1$. This is basically the same as in constraints (2) except for the load capacity C_s^L. The second difference is that constraints (2) aggregate over the clients c. Note that with M instead of C_s^L in constraints (2), i.e., without the capacity restriction, constraints (2) could be omitted from the disaggregated model.

We can further improve the lower bound by tightening the placement costs (second term) in the objective function (1) of the strong LP-relaxation. For a given LP-relaxation, we can calculate the minimum number of replicas as $r = \left\lceil \sum_{s \in \mathcal{V}} \text{server} \sum_{t \in \mathcal{T}} x_{s,t} \right\rceil$. Obviously, the number of replica placements within the entire planning horizon is $0 \leq r \leq |\mathcal{V}^{\text{server}}| \cdot T$, i.e., the minimum number of replica placements is 0 and for the maximum number of replica placements each replica server has one replica in each period. A lower bound of the placement costs can now be calculated as follows. Let us assume that $1 \leq r \leq T$ holds. In this case, we achieve minimum placement costs if we place in period $t = 1$ a single replica on the replica server closest to the origin server and leave the replica there until period T. In the example provided in Fig. 2, we add replica server 3 to the origin server 0 and obtain costs of $2 \cdot \beta$. Note that the thus defined graph with node set $\mathcal{V}' = \{0, 3\}$ and arc set $\mathcal{A}' = \{(0, 3)\}$ forms a subgraph of the overlay network. If we want to obtain a lower bound of the placement costs for $T < r \leq 2 \cdot T$, we extend the subgraph consisting of $\mathcal{V}'$ and $\mathcal{A}'$ by a node $j \in \mathcal{V} \backslash \mathcal{V}'$ and an associated edge (i, j) with a minimum distance connecting one node in $\mathcal{V}'$ with one node in $j \in \mathcal{V}^{\text{server}} \backslash \mathcal{V}'$. For the example we choose server 5 and obtain $\mathcal{V}' = \{0, 3, 5\}$ and arc set $\mathcal{A}' = \{(0, 3), (3, 5)\}$ with costs $(2+1) \cdot \beta$. Generally, we recursively calculate the distance $\hat{\delta}_u$ of the uth selected

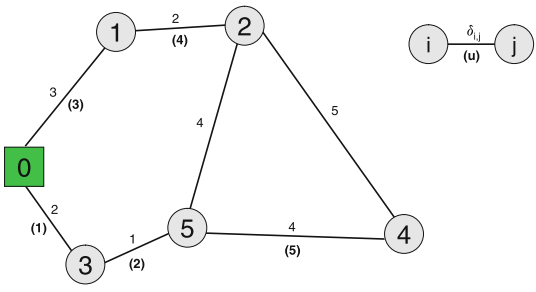

Fig. 2 Example for calculating the lower bound of the placement costs

edge and the set of u selected nodes $\mathcal{V}'(u)$ after initializing $\mathcal{V}'(0) = \{v_0\}$ as follows:

$$\hat{\delta}_u := \min\{\delta_{i,j} \mid i \in \mathcal{V}'(u-1),\ j \in \mathcal{V}^{\text{server}} \backslash \mathcal{V}'(u-1) : (i,j) \in \mathcal{A}\} \tag{18}$$

$$\mathcal{V}'(u) := \mathcal{V}'(u-1) \cup \{s\},\ s \in \{s \in \mathcal{V}^{\text{server}} \backslash \mathcal{V}'(u-1) \mid \delta_{i,s} = \hat{\delta}_u : i \in \mathcal{V}'(u-1)\} \tag{19}$$

The lower bound on the placement costs for r placed replicas is then

$$\underline{c}^{\beta}(r) = \sum_{u=1,\ldots,\lceil r\,/\,T\rceil} \hat{\delta}_u \tag{20}$$

We obtain the improved lower bound of the LP-relaxation by replacing the placement costs $\beta \cdot \sum_{(i,j)\in\mathcal{A}} \sum_{t\in\mathcal{T}} \delta_{i,j} \cdot z_{i,j,t}$ by $\underline{c}^{\beta}(r)$.

4 A simulated annealing heuristic for the DRPSL

Obtaining optimal solutions for practical replica placement problems with a computation time of a few minutes is not possible because the DRPSL is NP-hard. This necessitates heuristic approaches and in what follows we propose a simulated annealing (SA) metaheuristic. Simulated annealing (see Kirkpatrick et al. 1983) is a well-known metaheuristic inspired by the physical annealing process of solids. We adapt the standard simulated annealing scheme presented in Henderson et al. (2003) (see Algorithm 1).

4.1 Solution representation

As solution representation we choose a binary $|\mathcal{V}^{\text{server}}| \times T$ matrix. An entry $x_{s,t}$ equals 1, if replica server $s \in \mathcal{V}^{\text{server}}$ stores a replica of the object in period $t \in \mathcal{T}$, and 0 otherwise. The solution representation x is partial, as it does not define the solution for the placement z and the content delivery y. We will detail in Sect. 4.3 how a partial solution x is completed. In our initial solution, we set $x_{s,t} = 1$ for all $s \in \mathcal{V}^{\text{server}}$ and $t \in \mathcal{T}$. In doing so, we guarantee a feasible initial solution.

4.2 Neighborhood

The selection of a neighbor solution in neighborhood $N(x)$ works according to Algorithm 2, where exactly one replica is added or removed in one period. Thus, the size of the neighborhood is $|\mathcal{V}^{\text{server}}| \cdot T$. Let us first describe the selection of a neighbor for the case that the current solution is feasible. We randomly select a period t. With probability p we delete a replica from a server in period t, and with probability $(1-p)$ we add a replica to a server in period t. The selection of the server where a replica is removed (added) is made according to an approximation of the placement costs (see Algorithm 3). We describe the approximation for the case of removing a replica; the case of adding a replica works analogously. For each server s we assign rank $r_s \in \{-1, 0, 1, 2\}$. The

higher the value of the rank is, the more beneficial it is to remove the replica from the server. When assigning the rank, we distinguish the following four cases. First, if server s does not store a replica, we assign rank $r_s = -1$. Second, if server s stores a replica in each of the periods $t-1, \ldots, t+1$, we assign rank $r_s = 0$ because removing the replica in t increases the placement costs. Third, if server s stores a replica in t and in one of the adjacent periods $t-1$ or $t+1$, we assign rank $r_s = 1$ because placement costs do not change when removing the replica. Finally, if server s stores a replica in period t but not in the adjacent periods $t-1$ and $t+1$, we assign rank $r_s = 2$ since placement costs decrease when removing the replica in period t. We randomly select one of the servers with the highest rank and remove its replica.

If a replica is removed from a server, a new solution might be infeasible. In case of infeasibility, since a replica has been removed from one server in one period there is a single period t^{infeas} which causes the infeasibility. In order to find a feasible neighbor, we only consider a placement of an additional replica in period t^{infeas}. We randomly select one of the servers with the highest approximated value.

4.3 Solution evaluation

In order to evaluate a solution we have to calculate the objective function. Based on the solution representation x we can calculate the storage costs (first term of the objective function). However, in order to obtain the placement and the delivery costs (second and third term of the objective function), we have to complete x to the placement z and delivery y. In what follows, we describe how we calculate the three cost terms in the objective function by completing the solution.

Storage costs From the solution matrix x we can calculate the storage costs according to the first term of the objective function (1) straightforwardly to $\alpha \sum_{s \in \mathcal{V}^{\text{server}}} \sum_{t \in \mathcal{T}} x_{s,t}$.

Placement costs To calculate the placement costs for a given solution x, we need to know for each period the length of the multicast tree in the overlay network. As stated in Sect. 3.5, the problem of finding the optimal multicast tree in one period for a given placement can be formulated as a Steiner tree problem (STP) in graphs (see Voss 2006; Oliveira et al. 2006). To obtain the STP for a solution x, let us consider the placement problem of period t where we have three groups of servers: supply servers, demand servers and transshipment servers. Supply servers have a replica placement in period $t-1$ and thus can send a replica at the beginning of period t. Demand servers require a replica in period t. Transshipment servers neither have a replica in period $t-1$ nor in period t. Note that a server can be a supply and demand server at the same time. The placement problem is now transformed into a STP by aggregating all supply nodes into one virtual supply node. Arcs between supply nodes are deleted and if there are several arcs from the supply nodes to a demand or transshipment node the arc with smallest weight is used. Figure 3 provides an example where the two supply servers 2 and 5 are aggregated to the virtual supply node v.

For each period we have to solve one STP and thus in total we have to solve T STPs in order to obtain a solution for the placement problem. Since the STP is NP-hard and we have to solve it fast, we employ a heuristic, more precisely the KMB heuristic

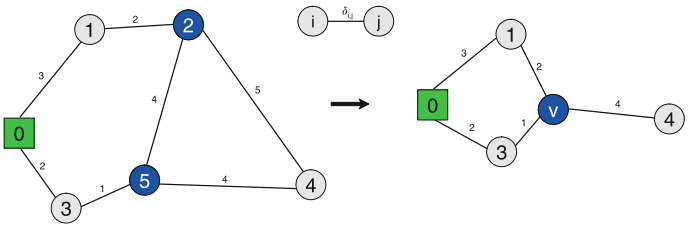

Fig. 3 Aggregation of two replica servers to a virtual node

of Kou et al. (1981). The KMB heuristic is based on minimum spanning trees and is known as one of the best heuristics for the STP. KMB has a performance guarantee of no more than twice the optimal objective function value. Computational experiments show that the objective function value is on average 5 % above the optimal value (see Oliveira and Pardalos 2005). At the start of the SA heuristic, we have to solve the STP for each period in order to obtain the placement costs. However, in each iteration of the SA, when there has been a change of the placement matrix in exactly one period t, we have to solve only the STP for the periods t and $t+1$ in order to actualize the placement costs.

Content delivery costs For a given request $r_{c,t}$ for all $c \in \mathcal{V}^{\text{client}}$ and $t \in \mathcal{T}$ we determine $y_{c,s,t}$, the fraction of all requests from client $c \in \mathcal{V}^{\text{client}}$ in period $t \in \mathcal{T}$ which is served by replica server $s \in \mathcal{V}^{\text{server}}$ when solving a transportation problem. We have employed the algorithm of Ahrens (1977) which is quite efficient with respect to runtime and the required data structure.

Infeasible solutions It might be that in some period t there are not enough replicas placed to deliver the object to the clients subject to the service-level constraints (4). In case of infeasibility, we add the penalty term (21) to the objective function.

$$\Delta^{\max} = \alpha \cdot 1 + \beta \cdot \max\left(d_{c,s}\right) + \gamma \cdot \overline{d}_{c,s} \cdot \max\left(r_{c,t}\right) \tag{21}$$

$\Delta^{\max}$ is an upper bound on the difference between two neighbor solutions which is made up as follows. First, at most one additional replica with cost α can be placed. Second, the maximum additional placement costs are the placement cost coefficient β times the maximum distance $d_{c,s}$ between a client c and a server s. Third, the additional delivery costs are the delivery cost coefficient γ multiplied with the maximum demand of a client $r_{c,t}$ and the average distance $\overline{d}_{c,s}$ between a client c and a server s. Note that the last term does not provide a true upper bound because we have taken the average distance $\overline{d}_{c,s}$ and not the maximum distance $\max(d_{c,s})$. We chose this approach because we do not want the acceptance probability of infeasible solutions to be too small in the course of the SA algorithm.

Cooling schedule We implemented the widely used geometric cooling schedule with $t_k = 0.9 \cdot t_{k-1}$ (see Aarts et al. 2005). Following Aarts et al. (2005), we set the initial temperature equal to the maximal cost difference between any two neighbor solutions, i.e., $t_0 = \Delta^{\max}$, and the number of evaluated solutions at each temperature level to the size of the neighborhood, i.e., $M_k \equiv |\mathcal{V}| \cdot T$. The stop criterion has been set to six consecutive temperature levels without a new best solution.

Table 1 Parameters and levels for the experimental test design (see also Appendix C Table 4)

Parameters	Experimental levels
ρ	0.9, 0.8, 0.75, 0.7, **0.66**, 0.6, 0.5, 0.4, 0.33, 0.3, 0.25, 0.2, 0.1
λ	0.75, 0.9, 0.95, **0.99**, 1.0
l	0.5, 0.75, 1.0, 1.5, **2.0**, 3.0
α	10, 50, 100, 500, **1,000**, 2,000
β	0, 0.1, **0.2**, 0.5, 1
γ	0, **0.01**, 0.05, 0.1, 0.5, 1, 2, 5

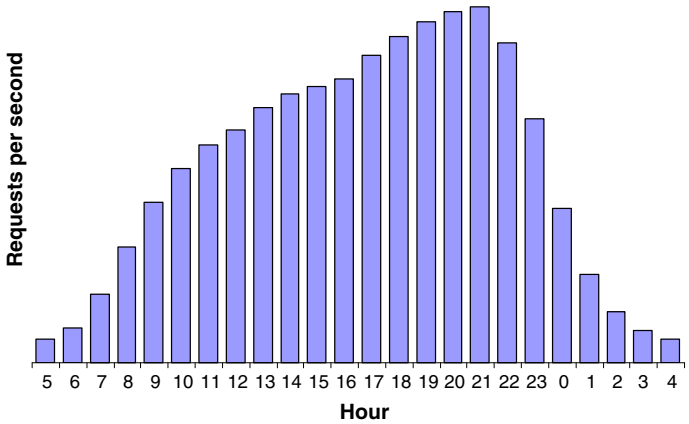

Fig. 4 Example for a request rate profile for Europe

5 Experimental investigation

5.1 Experimental setup

Test instances For our experimental study we generated a set of 38 test instances where we altered a base case subject to six parameters. The parameters and their levels are given in Table 1 where the levels of the base case are set in bold. Table 4 in Appendix C provides more detailed information on the test instances.

For the base instance, we considered a graph with $V = 50$ nodes which relates to the number of cities in the European Union that have at least one data center hosting servers of the CDN provider. As planning horizon, we have $T = 12$ periods of 2 h length. For problems of this size the DRPSL has 36,172 variables, 3,111 of them binary, and 35,535 constraints. To generate the graph, we employed the network generator BRITE (see Medina et al. 2001) using the Waxman model (Waxman 1988), one of the most commonly used models for generating graphs (see Jamin et al. 2000; Qiu et al. 2001; Shi and Turner 2002; Zhang et al. 2003; Tang and Xu 2005; Jeon et al. 2006).

For the time varying demand of the clients $r_{c,t}$, we used the total demand for objects hosted by Akamai within Europe (see Fig. 4) which is available on the homepage of Akamai (Akamai Technologies Inc. 2010). We disaggregated this data in order to get

request rates of single clients. Doing this we considered the shift of demand due to different time zones.

We have set a constant capacity for each server $s \in \mathcal{V}^{\text{server}}$ according to

$$C_s^L = \frac{1}{\rho \cdot |\mathcal{V}^{\text{server}}|} \cdot \max_{t \in \mathcal{T}} \left(\sum_{c \in \mathcal{V}^{\text{client}}} r_{c,t} \right), \tag{22}$$

where ρ is the average load per server in the period with maximum demand if we assume that each server has a replica of the object.

For the service-level constraint (4), we calculate the maximum latency q by varying the control parameter l according to

$$q = l \cdot \bar{\delta}_{i,j}, \tag{23}$$

where $\bar{\delta}_{i,j}$ denotes the average arc length of the network. The remaining systematically changed parameters are the percentage of requests which have to be served within the maximum latency λ, as well as the three cost coefficients α, β and γ.

Solution methods As solution method we use the proposed SA algorithm. We stopped the algorithm once there was no improvement of the global optimum for six temperature levels. Each instance has been solved ten times with different seed values and averages are reported. Furthermore, we apply CPLEX with time limits of 60, 300, 600, and 3,600 s, denoted as CPLEX(... s), as well as with a 10 % limit of the solution gap, denoted as CPLEX (10 %). The solution gap $\frac{(UB-LB)}{LB}$ of CPLEX is the percentage deviation of the best feasible solution UB from the LP-based lower bound LB for solving the tightened MIP-formulation (1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12), (13), (14), (15), (16) and (17). All problem instances are solved using one Intel Xeon core with 1.6 GHz on a workstation with 6 GB RAM. The simulated annealing metaheuristic and all of its components are implemented in Java 6.0. For CPLEX, we used version 10.1 at the default parameters.

5.2 Computation times

None of the instances could be solved to optimality when applying CPLEX (3,600 s). For CPLEX (10 %), we obtained an average runtime of 2.4 h (maximum runtime 44.9 h). The LP-relaxations are solved considerably faster. The weak LP-relaxation based on (1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12), (13), (14), (15) and (16) could be solved for all but one instance in less than 1 s (maximum 3 s). The strong LP-relaxation based on (1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12), (13), (14), (15), (16) and (17) required 11 s on average (maximum 35 s), improving the weak LP-relaxation by 27 % on average. The simulated annealing heuristic took on average 21 s (maximum 50 s). Figure 5 provides the average runtime of CPLEX (10 %) and the SA heuristic on a logarithmic scale with base 10 for all instances. On all but one ($i38$) instances SA has a shorter runtime than CPLEX (10 %). The impact of the parameter levels on the computation time is stronger for CPLEX 10 % than for the SA heuristic. We can observe two instances with extreme runtimes, $i23$ and $i38$.

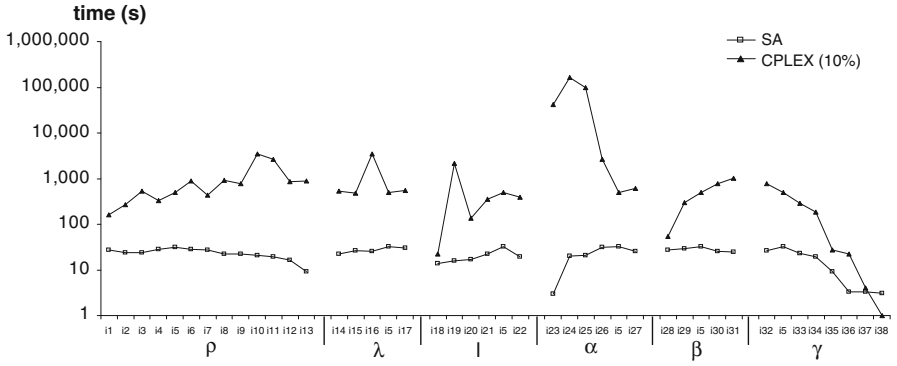

Fig. 5 Average solution times of CPLEX 10 % and SA

Table 2 Gaps of the solution methods

Solution method	MinΔ (%)	AvgΔ (%)	MaxΔ (%)
CPLEX (3,600 s)	0.04	6.14	44.31
CPLEX (10 %)	0.48	8.73	44.31
SA	0.56	8.96	46.39
CPLEX (600 s)	0.05	12.99	56.53
CPLEX (300 s)	0.07	34.20	79.89
CPLEX (60 s)	0.06	52.53	89.46

For instance $i23$ the storage cost coefficient α is at the lowest value of 0.5 while for $i38$ the delivery cost coefficient γ is at the highest value of 5. In both cases the best solutions have a high density on the placement matrix x, i.e., there are a large number of replica placements. Since the SA algorithm starts with a matrix $x_{s,t} \equiv 1$, the initial solution is already close to the optimal solution.

5.3 Solution gaps

The solution gaps are defined as $\Delta = \frac{z-\text{LB}}{z}$, where z is the objective function value of a particular solution method and LB is the lower bound derived from the strong LP-relaxation augmented by the tightening of the placement cost as described in Sect. 3.6. Table 2 provides the minimum, the average and the maximum solution gaps of the different methods ordered according to the average solution gap.

Figure 6 provides the solution gaps of the three selected solution methods CPLEX (60 s), CPLEX (3,600 s) and SA for all instances. In what follows, we discuss the impact of the parameters on these solution methods. Table 5 in the Appendix D provides the underlying data where the smallest gap for each instance is set in bold face. Note that CPLEX (60 s) did not find feasible solutions for the instances $i31$ and $i32$. The instances where the proposed SA heuristic has the highest solution gaps are for small ρ and thus high server capacities, medium latency constraints through a medium level of parameter l and medium delivery cost coefficient γ. We conjecture that the high gap for a low storage cost coefficient α stems from weak lower bounds

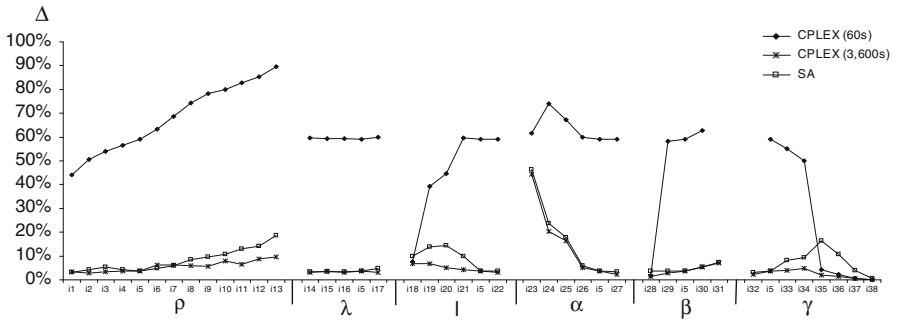

Fig. 6 Average solution gaps Δ for CPLEX (60 s), CPLEX (3,600 s) and SA

and not from a poor upper bound; we will discuss this issue below. Let us now analyze the impact of the six different parameters.

Impact of the capacity per server C_s^L For decreasing ρ and thus, via Eq. (22), increasing capacity C_s^L per server s, we see that the solution gap of all three methods increases. More capacity increases the solution space and thus makes it harder to find optimal or near-optimal solutions.

Impact of the percentage of requests λ *which have to be served with a maximum latency* λ does not have an impact on the gap of the three methods.

Impact of the maximum latency q Increasing the maximum latency q by increasing parameter l, see Eq. (23), has a different impact on the solution gap of each of the three methods. The solution gap of CPLEX (60 s) increases considerably. On the contrary, the solution gap of CPLEX (3,600 s) decreases mildly. Finally, the solution gap of SA increases first and then decreases. We conjecture that for problems with low maximum latency, the replicas have to be placed close to the demand and hence the set of feasible solutions is rather small, which makes problems easier to solve. In addition, problems with high maximum latency allowed are easy to solve because the latency constraints (4) are relaxed. The problems with a medium maximum latency are difficult to solve for the SA algorithm and to a lesser extent for CPLEX (3,600 s). CPLEX (60 s) does not have enough solution time to derive good solutions for any but the problems with small maximum latency and thus a small solution space. A similar observation of such a bell-shaped impact of resource scarcity on computation time for NP-hard optimization problems has been detected by De Reyck and Herroelen (1996) for the resource-constrained project scheduling problem.

Impact of the storage cost coefficient α An increasing α does not have a clear impact on the solution gap of CPLEX (60 s), while the solution gap of CPLEX (3,600 s) and SA decreases. We assume that the lower bound degrades for small α, because with fixed placement coefficient β the solution of the placement problem receives a higher weight in the objective function. Since the placement problem is hard to solve and we do not have a strong bound for it, the lower bound decreases.

Impact of the placement cost coefficient β Increasing β leads to a considerable increase in the solution gap of CPLEX (60 s). For $\beta = 1$, CPLEX (60 s) even fails to generate a feasible solution. The solution gap of CPLEX (3,600 s) and SA increases only moderately.

Impact of the delivery cost coefficient γ Increasing the delivery cost coefficient γ leads to a decrease in the solution gap for CPLEX (60 s) and CPLEX (3,600 s) which is rather steep for CPLEX (60 s). This happens because, as stated in Sect. 3.4, solving the DRPSL becomes trivial if only delivery costs are relevant. For the SA algorithm, we observe a bell-shaped impact of the delivery cost coefficient on the solution gap, which reveals that the SA algorithm has difficulties when solving instances with medium delivery costs.

6 Conclusions

In this work, we have addressed the problem of the dynamic placement of replicas in content delivery networks in order to minimize total costs consisting of storage, placement and delivery costs. We have modeled the problem as MIP which takes into account service-level agreements as well as unicast and multicast transfers for delivery and placement, respectively. Furthermore, we proposed a simulated annealing heuristic for the problem. We carried out an experimental study to show the performance of different settings of the state-of-the-art solver CPLEX and the SA heuristic. The results show that within less than 50 s the simulated annealing heuristic generates solutions of almost the same quality as CPLEX with 1 h computation time. Furthermore, the solution quality is much more stable in the sense that it is less impacted by varying levels of the problem parameters.

Further research could be directed in several directions. With respect to solution procedures one could try to develop better methods, e.g., hybridizing the MIP and the SA approach or refining the SA by, e.g., including other neighborhoods or improving the MIP model so as to speed up the search of off-the-shelf MIP-solvers. With respect to the model, one could generalize the model to multiple objects.

Appendix A: Notation

See Table 3

Table 3 Notation for the DRPSL

Sets	
$\mathcal{T}$	Set of periods
$\mathcal{V}$	Set of nodes in the network
$\mathcal{V}^{\text{server}}$	Set of replica servers in the network
$\mathcal{V}^{\text{client}}$	Set of clients in the network
$\mathcal{V}^{\text{server}}(c, q)$	Set of replica servers which can serve client $c \in \mathcal{V}_c$ within the maximum latency q
$\mathcal{A}$	Set of directed edges in the network
$\mathcal{V}^{\text{pred}}(s)$	Set of immediate predecessor nodes of node s in the network
$\mathcal{V}^{\text{succ}}(s)$	Set of immediate successor nodes of node s in the network
Parameter	
T	Number of periods in $\mathcal{T}$
V	Number of nodes $n \in \mathcal{V}$
$\delta_{i,j}$	Weight of edge $(i, j) \in \mathcal{A}$ (e.g., distance)

Table 3 continued

$d_{c,s}$	Distance if a request of client $c \in \mathcal{V}^{\text{client}}$ is served by replica server $s \in \mathcal{V}^{\text{server}}$ (i.e., the length of the shortest path)
C_s^L	Load capacity of replica server $s \in \mathcal{V}_s$ per period
$r_{c,t}$	Number of requests from client $c \in \mathcal{V}_c$ for the object in period $t \in \mathcal{T}$
q	Maximum allowed latency due to service-level agreements
λ	Fraction of all requests which have to be served within the maximum latency q
α	Cost coefficient for the storage of a replica on a server $s \in \mathcal{V}^{\text{server}}$ per period
β	Cost coefficient for the placement per distance unit
γ	Cost coefficient for the delivery per distance unit
Decision variables	
$x_{s,t}$	Indicates if server $s \in \mathcal{V}_s$ stores a replica in period $t \in \mathcal{T}$
$y_{c,s,t}$	Fraction of all requests from client $c \in \mathcal{V}_c$ in period $t \in \mathcal{T}$ which is served by replica server $s \in \mathcal{V}_s$
$w_{s,t}$	Indicates if a replica is newly stored on replica server $s \in \mathcal{V}_s$ in period $t \in \mathcal{T}$
$\tilde{z}_{i,j,t}$	Number of times edge $(i, j) \in \mathcal{A}$ is used for the placement at the beginning of period $t \in \mathcal{T}$
$z_{i,j,t}$	Indicates if edge $(i, j) \in \mathcal{A}$ is used for the placement at the beginning of period $t \in \mathcal{T}$

Appendix B: Pseudocodes

Algorithm 1 Simulated annealing scheme

```
Input: Initial temperature t_0 ≥ 0
Input: Temperature cooling schedule t_k
Input: Repetition schedule M_k that defines the number of iterations
       executed at each temperature t_k

Select an initial solution x
Set the temperature change counter: k ← 0
Set the currently best solution: x* ← x

while k < stopping criterion do
    for repetition counter m ← 0 to M_k do
        Generate a solution x' ∈ N(x)
        Calculate Δ_{x,x'} ← cost(x') − cost(x)
        if Δ_{x,x'} ≤ 0 then
            Accept new solution x' (x ← x')
            if cost(x') < cost(x*) then
                x* ← x'
                k ← 0
            end if
        else
            Accept new solution x' (x ← x') with
             probability exp(−Δ_{x,x'}/t_k)
        end if
    end for
    k ← k + 1
end while

return x*
```

Algorithm 2 Generate a neighbor of solution x in neighborhood $N(x)$

Input: Probability p to remove a server
Input: Solution x

if $t^{\text{infeas}} \neq NULL$ **then**
    $t \leftarrow t^{\text{infeas}}$
    $s \leftarrow$ chooseServerToAdd(x, t)
else
    $t \leftarrow random(\mathcal{T})$
    if $random[0, 1[< p$ **then**
        $s \leftarrow$ chooseServerToRemove(x, t)
    else
        $s \leftarrow$ chooseServerToAdd(x, t)
    end if
end if
$x_{s,t} \leftarrow \neg x_{s,t}$ ▷ negation of the boolean value

return x

Algorithm 3 Choice of a server to remove a replica from

function chooseServerToRemove(solution x, period t)
    for all $s \in \mathcal{V}^{\text{server}}$ **do**
        if $x_{s,t} = 0$ **then**
            $r_s \leftarrow -1$
        else
            $r_s \leftarrow 0$
            if $t > 1$ **and** $x_{s,t-1} = 0$ **then**
                $r_s \leftarrow r_s + 1$
            end if
            if $t < |\mathcal{T}|$ **and** $x_{s,t+1} = 0$ **then**
                $r_s \leftarrow r_s + 1$
            end if
        end if
    end for
    return A randomly chosen replica server s of the highest rank r
end function

Appendix C: Test instances

See Table 4

Table 4 Test instances (base case $i5$)

Instance	ρ	λ	l	α	β	γ
$i1$	0.9					
$i2$	0.8					
$i3$	0.75					
$i4$	0.7					
$i5$	**0.66**	**0.99**	**2**	**1,000**	**0.2**	**0.01**
$i6$	0.6					
$i7$	0.5					
$i8$	0.4					
$i9$	0.33					
$i10$	0.3					
$i11$	0.25					
$i12$	0.2					
$i13$	0.1					
$i14$		0.75				
$i15$		0.9				
$i16$		0.95				
$i17$		1				
$i18$			0.5			
$i19$			0.75			
$i20$			1.0			
$i21$			1.5			
$i22$			3			
$i23$				10		
$i24$				50		
$i25$				100		
$i26$				500		
$i27$				2,000		
$i28$					0	
$i29$					0.1	
$i30$					0.5	
$i31$					1	
$i32$						0
$i33$						0.05
$i34$						0.1
$i35$						0.5
$i36$						1
$i37$						2
$i38$						5

Appendix D: Computational results

See Table 5

Table 5 Solution gaps Δ for CPLEX (60 s), CPLEX (600 s), CPLEX (3,600 s) and SA

Instance	CPLEX (60 s)	CPLEX (600 s)	CPLEX (3,600 s)	SA
i1	44.10	3.25	3.30	**2.98**
i2	50.61	4.24	**2.95**	4.15
i3	53.89	5.97	**3.47**	5.29
i4	56.45	5.54	**3.81**	4.11
i5	59.15	4.18	**3.65**	3.73
i6	63.37	55.84	6.25	**4.67**
i7	68.54	7.51	6.23	**5.82**
i8	74.43	11.28	**6.05**	8.45
i9	78.24	10.34	**5.72**	9.63
i10	79.89	13.67	**7.84**	10.64
i11	82.72	10.01	**6.39**	13.00
i12	85.30	10.47	**8.71**	14.11
i13	89.46	12.90	**9.62**	18.65
i14	59.51	4.96	**3.10**	3.42
i15	59.44	5.42	3.70	**3.40**
i16	59.20	44.66	**3.23**	3.38
i5	59.15	4.18	**3.65**	3.73
i17	59.76	6.04	**2.97**	4.81
i18	7.54	7.12	**6.83**	9.76
i19	39.26	10.23	**6.71**	13.88
i20	44.65	5.59	**5.13**	14.44
i21	59.68	5.84	**4.10**	9.82
i5	59.15	4.18	**3.65**	3.73
i22	59.13	4.15	**3.18**	3.67
i23	61.72	46.48	**44.31**	46.39
i24	73.88	23.00	**20.31**	23.88
i25	67.29	21.20	**16.50**	17.88
i26	59.88	11.38	**5.16**	5.84
i5	59.15	4.18	**3.65**	3.73
i27	59.02	4.75	**2.37**	3.39
i28	1.58	1.48	**1.47**	3.61
i29	58.25	3.13	**2.92**	3.58
i5	59.15	4.18	**3.65**	3.73
i30	62.61	48.40	5.31	**5.23**
i31		57.53	**7.13**	7.53
i32		56.53	**2.22**	3.20
i5	59.15	4.18	**3.65**	3.73

Table 5 continued

Instance	CPLEX (60 s)	CPLEX (600 s)	CPLEX (3,600 s)	SA
i33	55.13	6.00	**3.96**	8.17
i34	50.09	7.08	**4.79**	9.32
i35	4.28	2.21	**1.93**	16.31
i36	2.12	1.53	**1.39**	10.71
i37	0.61	0.48	**0.42**	3.92
i38	0.06	0.05	**0.04**	0.56

References

Aarts E, Korst J,Wil M (2005) Simulated annealing. In Burke EK, Kendall G (eds) Search methodologies. Springer, New York

Aggarwal A, Rabinovich M (1998) Performance of dynamic replication schemes for an internet hosting service. In: Technical report, AT&T Labs. http://www.research.att.com/~misha/radar/tm-perf.ps.gz

Ahrens J (1977) A code for the transportation problem. In: Technical report, Universität Kiel, Kiel

Aikens CH (1985) Facility location models for distribution planning. Eur J Oper Res 22(3):263–279

Aioffi WM, Mateus GR, Almeida JM, Melo RC (2004) Dynamic content placement for mobile content distribution networks. In: Chi C-H , van Steen M, Wills C (eds) Web content caching and distribution, vol 3293. Lecture notes in computer science. Springer, Berlin, pp 19–33

Akamai Technologies Inc. (2010) http://wwwnui.akamai.com/datavis/visitors_feed.xml. Accessed 4 February 2010

Baentsch M, Baum L, Molter G, Rothkugel S, Sturm P (1997) Enhancing the web's infrastructure: from caching to replication. IEEE Internet Comput 1(2):18–27

Bartolini N, Presti F, Petrioli C (2003) Optimal dynamic replica placement in content delivery networks. In: The 11th IEEE international conference on networks, ICON 2003, pp 125–130

Chen Y, Katz R, Kubiatowicz J (2002) Dynamic replica placement for scalable content delivery. In: Druschel P, Kaashoek F, Rowstron A (eds) Peer-to-peer systems, vol 2429. Lecture notes in computer science. Springer, New York, pp 306–318

Chen Y, Qiu L, Chen W, Nguyen L, Katz R (2003) Efficient and adaptive web replication using content clustering. IEEE J Sel Areas Commun 21(6):979–994

Chu W (1969) Optimal file allocation in a multiple computer system. IEEE Trans Comput C-18(10):885–889

Cidon I, Kutten S, Soffer R (2002) Optimal allocation of electronic content. Comput Netw 40(2):205–218

De Reyck B, Herroelen W (1996) On the use of the complexity index as a measure of complexity in activity networks. Eur J Oper Res 91(2):347–366

Dilley J, Maggs B, Parikh J, Prokop H, Sitaraman R, Weihl B (2002) Globally distributed content delivery. IEEE Internet Comput 6(5):50–58

Dowdy LW, Foster DV (1982) Comparative models of the file assignment problem. ACM Comput Surv 14(2):287–313

Frank A, Wittie L, Bernstein A (1985) Multicast communication on network computers. IEEE Softw 2(3):49–61

Henderson D, Jacobson S, Johnson A (2003) The theory and practice of simulated annealing. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics (international series in operations research and management science). Springer, New York

Jamin S, Jin C, Jin Y, Raz D, Shavitt Y, Zhang L (2000) On the placement of internet instrumentation. In: 19th annual joint conference of the IEEE computer and communications societies, INFOCOM 2000. Proceedings of the IEEE, vol 1, pp 295–304

Jamin S, Jin C, Kurc A, Raz D, Shavitt Y (2001) Constrained mirror placement on the internet. In: 20th annual joint conference of the IEEE computer and communications societies, INFOCOM 2001. Proceedings of the IEEE, vol 1, pp 31–40

Jeon WJ, Gupta I, Nahrstedt K (2006) Mmc01-6: Qos-aware object replication in overlay networks. In: Global telecommunications conference, GLOBECOM '06. IEEE, pp 1–5

Jia X, Li D, Hu X, Wu W, Du D (2003) Placement of web-server proxies with consideration of read and update operations on the internet. Comput J 46(4):378–390
Kalpakis K, Dasgupta K, Wolfson O (2001) Optimal placement of replicas in trees with read, write, and storage costs. IEEE Trans Parallel Distrib Syst 12(6):628–637
Kangasharju J, Roberts J, Ross KW (2002) Object replication strategies in content distribution networks. Comput Commun 25(4):376–383
Karlsson M, Karamanolis C (2003) Bounds on the replication cost for QoS. In: Technical report, Hewlett Packard Labs, USA
Karlsson M, Karamanolis C (2004) Choosing replica placement heuristics for wide-area systems. In: Proceedings of the 24th international conference on distributed computing systems, pp 350–359
Karlsson M, Mahalingam M (2002) Do we need replica placement algorithms in content delivery networks. In: 7th international workshop on web content caching and distribution (WCW)
Karp R (1972) Reducibility among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of computer computations. Plenum Press, London
Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220:671–680
Korupolu MR, Plaxton CG, Rajaraman R (2001) Placement algorithms for hierarchical cooperative caching. J Algorithms 38(1):260–302
Kou L, Markowsky G, Berman L (1981) A fast algorithm for steiner trees. Acta Informatica 15(2):141–145
Krarup J, Pruzan PM (1983) The simple plant location problem: survey and synthesis. Eur J Oper Res 12(1):36–81
Krishnan P, Raz D, Shavitt Y (2000) The cache location problem. IEEE/ACM Trans Netw 8(5):568–582
Li B, Golin M, Italiano G, Deng X, Sohraby K (1999) On the optimal placement of web proxies in the internet. In: 18th annual joint conference of the IEEE computer and communications societies, INFOCOM '99. Proceedings of the IEEE, vol 3, pp 1282–1290
Medina A, Lakhina A, Matta I, Byers J (2001) Brite: universal topology generation from a user's perspective. In: Technical report, Boston University, Boston
Nguyen TV, Safaei F, Boustead P, Tung C (2005) Provisioning overlay distribution networks. Comput Netw 49(1):103–118
Oliveira CA, Pardalos PM, Resende MG (2006) Optimization problems in multicast tree construction. In: Resende MG, Pardalos PM (eds) Handbook of optimization in telecommunications. Springer, New York
Oliveira CAS, Pardalos PM (2005) A survey of combinatorial optimization problems in multicast routing. Comput Oper Res 32(8):1953–1981
Pallis G, Vakali A (2006) Insight and perspectives for content delivery networks. Commun ACM 49(1):101–106
Pathan M, Buyya R, Vakali A (2008) Content delivery networks: state of the art, insights, and imperatives. In: Buyya R, Pathan M, Vakali A (eds) Content delivery networks. Springer, New York, pp 3–32
Paul S (1998) Multicasting on the internet and its applications. Kluwer Academic Publishers, Norwell
Qiu L, Padmanabhan V, Voelker G (2001) On the placement of web server replicas. In: 20th annual joint conference of the IEEE computer and communications societies, INFOCOM 2001. Proceedings of the IEEE, vol 3, pp 1587–1596
Rabinovich M, Spatschek O (2002) Web caching and replication. Addison-Wesley Longman Publishing Co. Inc., Boston
Radoslavov P, Govindan R, Estrin D (2002) Topology-informed internet replica placement. Comput Commun 25(4):384–392
Rahul HS, Kasbekar M, Sitaraman RK, Berger AW (2008) Performance and availability benefits of global overlay routing. In: Buyya R, Pathan M, Vakali A (eds) Content delivery networks. Springer, New York, pp 251–274
Shi S, Turner J (2002) Placing servers in overlay networks. In: Symposium on performance evaluation of computer and telecommunication systems (SPETS)
Tang X, Xu J (2005) QoS-aware replica placement for content distribution. IEEE Trans Parallel Distrib Syst 16(10):921–932
Unger O, Cidon I (2004) Optimal content location in multicast based overlay networks with content updates. World Wide Web 7(3):315–336
Vakali A, Pallis G (2003) Content delivery networks: status and trends. IEEE Internet Comput 7(6):68–74
Verma DC (2002) Content distribution networks: an engineering approach. John Wiley , New York
Voss S (1990) Steiner-Probleme in Graphen. Anton Hain, Germany

Voss S (2006) Steiner tree problems in telecommunications. In: Resende MG, Pardalos PM (eds) Handbook of optimization in telecommunications. Springer, New York, pp 459–492

Waxman B (1988) Routing of multipoint connections. IEEE J Sel Areas Commun 6(9):1617–1622

Wittmann R, Zitterbart M (1999) Multicast communication: protocols and applications. Morgan Kaufmann Publishers Inc., Burlington

Wolfson O, Milo A (1991) The multicast policy and its relationship to replicated data placement. ACM Trans Database Syst 16(1):181–205

Xu J, Li B, Lee DL (2002) Placement problems for transparent data replication proxy services. IEEE J Sel Areas Commun 20(7):1383–1398

Zhang X, Wang W, Tan X, Zhu Y (2003) Data replication at web proxies in content distribution network. In: Zhou X, Orlowska ME, Zhang Y (eds) Web technologies and applications, vol 2642. Lecture notes in computer science. Springer, New York, pp 560–569