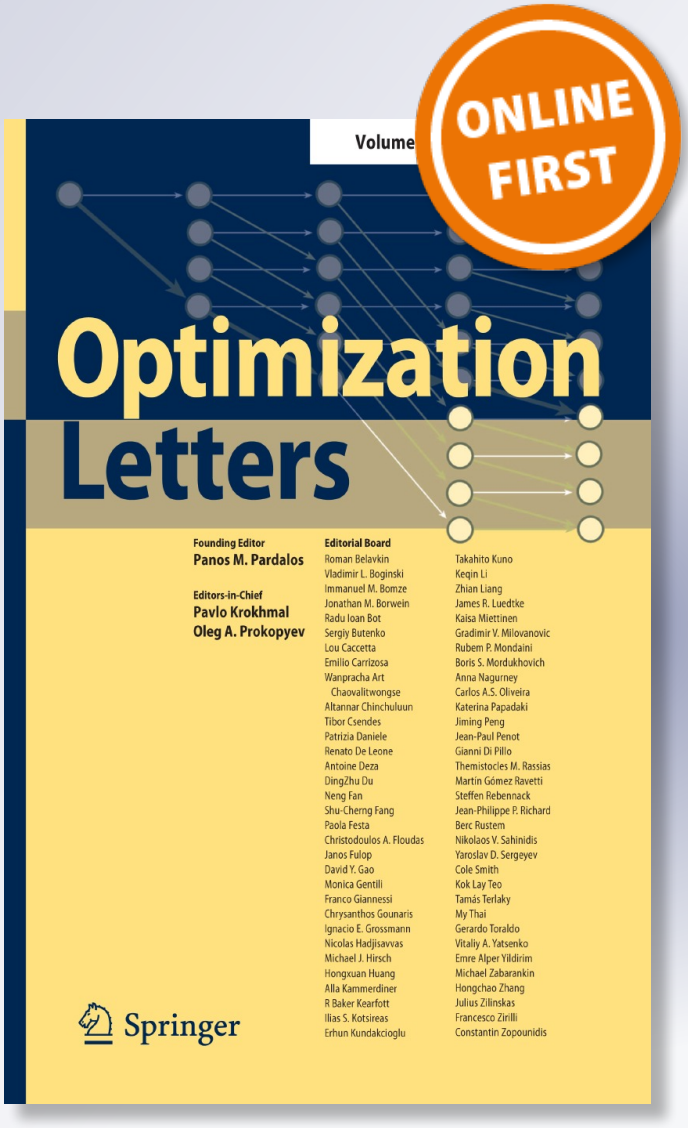
A new hybrid heuristic for replica placement and request distribution in content distribution networks

Tiago Neves, Luiz Satoru Ochi & Célio Albuquerque

Optimization Letters

ISSN 1862-4472

Optim Lett
DOI 10.1007/s11590-014-0770-6



Springer

Optim Lett
DOI 10.1007/s11590-014-0770-6

ORIGINAL PAPER

A new hybrid heuristic for replica placement and request distribution in content distribution networks

Tiago Neves · Luiz Satoru Ochi · Célio Albuquerque

Received: 18 October 2012 / Accepted: 2 July 2014

Abstract In Content Distribution Networks (CDN), in order to better serve clients, it is necessary to replicate contents at surrogate servers and distribute requests using such servers. The decisions of where to place replicated contents and how to distribute clients can be modeled as an optimization problem known as Replica Placement and Request Distribution Problem (RPRDP). In this paper we use a model that regards several realistic details that are not treated simultaneously in the literature, such as: constraints in server disk space and bandwidth, QoS requirements of requests and changes in the network conditions. Also, a new hybrid method, that uses exact and heuristic concepts simultaneously, is presented to solve the RPRDP. We compared the results obtained by the proposed algorithm with: a bound obtained by an exact offline approach, a solution of a real CDN provider and other hybrid heuristics. Results show that the proposed method outperforms the solution used in real CDNs and also in all other studied methods.

Keywords Replica placement · Request distribution · Content distribution network · Hybrid heuristic

T. Neves (✉)
Escola de Engenharia Industrial e Metalúrgica de Volta Redonda,
Universidade Federal Fluminense, Av. dos Trabalhadores 420, Volta Redonda, RJ, Brazil
e-mail: tneves@id.uff.br

L. S. Ochi · C. Albuquerque
Instituto de Computação , Universidade Federal Fluminense,
Rua Passo da Pátria 156, Niterói, RJ, Brazil

Published online: 24 July 2014

1 Introduction

A Content Distribution Network (CDN) is an overlay network [1] that is used to reduce the congestion by replicating contents in servers that are geographically close to clients. One of several problems related to CDN management is the Replica Placement Problem (RPP) [2] which consists in finding the best servers to place the contents in order to reduce the total traffic in the network.

The problem addressed in this paper is a generalization of the RPP called Replica Placement and Request Distribution Problem (RPRDP) which is a dynamic and online problem whose objectives are to find the best position for the content replicas and to distribute the requests across the servers, aiming at reducing the network traffic without violating the Quality of Service (QoS) requirements of the clients. Servers have limited capacity in bandwidth and disk space. The QoS constraints are given by a desired minimal bandwidth and a maximum delay in which a request is better served. In the beginning, the contents are positioned only in their origin servers. As requests arrive in the CDN system, contents may be replicated over the network and such requests may be redistributed across the servers taking into account QoS constraints. A server is allowed to handle a request partially or totally, as long as it has a replica of the desired content. A request may be served by several servers partially at the same time in order to satisfy clients' expectations. In this work, in order to handle the dynamics of requests and replications, time is divided in units called periods, and the optimization process works over all periods. Besides, clients may not have enough bandwidth to download a content in a single period, meaning that several periods are typically required to fully handle a single request. Moreover, since the overlay network may lack the resources to handle all requests within the desired QoS, some requests may have the QoS requirements partially fulfilled, but every time a request is not handled within the specified parameters a high cost is paid. A trade-off is established between the network bandwidth savings and the use of disk space in servers since replication is not free of costs and increases the use of disk space. Therefore, not only the QoS constraints but also such trade-off must be verified during the optimization process. RPRDP is a dynamic and online problem, meaning that costs of communication among servers can change, new contents and requests may come up, some contents may be purged and the future scenario is not known a priori. To the best of our knowledge, all this realistic details are not often addressed simultaneously in the literature.

There is a number of recent solutions proposed for CDN management problems and some of them are briefly described in this section. In [2–7] dynamic versions of CDN problems that do not consider quality perceived by the clients are studied. In [2] a version of the RPP, that appears in enterprise mobile networks, is presented. The authors present a mathematical formulation and a distributed heuristic. The distributed heuristic uses a forecast method on each server and, based on the forecasts, the servers determine the costs of replicating and maintaining contents. The cost of remote handling the requests are also analyzed. Based on these costs, the servers choose the best strategy to follow. The authors considered that one single period is enough to fully handle any request. This assumption may not be realistic for large contents since the request handling for such contents can take several hours depending on the content size. In [5] unplanned strategies for replica placement and request distribution

are compared with joint-optimization approaches. The results show that unplanned strategies have a greater values of Maximum Link Utilization when compared with an optimized approach with perfect future knowledge, but these values decrease as the storage capacity increases. The results also indicate that joint-optimization approaches could benefit from a good demand estimator, since the approach that has perfect future knowledge presented good results. In [3], the authors tackled the RPP and propose the use of a multicast tree to deliver the contents to end clients considering the delivery costs. In [4], the RPP is modeled as a Markovian process, which is able to deal with dynamic systems by using mathematical distributions. New requests are modeled as an incoming rate and the decision problem is to choose the best change to perform in an initial replica positioning in order to handle new incoming requests. In [7], the authors used concepts of Peer-to-Peer networks to solve the RPP. Servers communicate with each other, exchanging information about local traffic and load. The Servers use such information to locally decide on creating, deleting and migrating replicas, making the CDN infrastructure more efficient and fault tolerant. In [6], a centralized and a distributed approaches are proposed to solve the RPP. The algorithms take into account ratios that are calculated based on the replicas access frequency, communication and processing cost. The proposed algorithms are compared among themselves and the distributed one presented the best results.

Some papers deal with static versions of CDN problems. As the problem addressed here is dynamic we will not discuss these papers. However, the reader interested in static versions of problems related to CDN management is referred to [8–10].

Despite the attempts to optimize several aspects related to the RPP and aspects related to the CDN architecture, few papers consider the perceived quality at the end clients during the optimization process and, to the best of our knowledge, there is only one paper [11] that treats important issues like QoS constraints, the existence of multiple contents, network capacity and server load simultaneously. In [11], a mathematical formulation and several heuristics, including a heuristic used in real CDN, are presented to solve the RPRDP considering all mentioned characteristics. The heuristics split the RPRDP into two subproblems: the RPP and the Request Distribution Problem (RDP). The RDP consists in finding the best set of servers to handle each request. The RDP uses as input a replica positioning, thus, in each period, the heuristics proposed in [11] first define the replica positioning and then use such positioning to solve the RDP. This paper extends the work presented in [11], presenting a new hybrid heuristic that outperforms all methods exposed in [11] for the online version of the problem.

The remaining of the paper is organized as follows: Sect. 2 defines the problem. The new hybrid heuristic is presented in Sect. 3. The computational results are presented in Sect. 4 and Sect. 5 concludes the work.

2 Problem definition

The objective of the RPDRP is to find the best servers to place content replicas and to define how many and which servers will handle each request over the horizon so that CDN providers costs are minimized, servers constraints are not violated and QoS

requirements of the requests satisfied when possible. This problem is NP-Hard since it can be reduced to the Capacitated Facility Location Problem [12], which is NP-Hard. In order to better explain the problem, let the optimization horizon be divided in units called periods. Thus, the whole set of periods is the optimization horizon, denoted by T. Also let S be the set of available servers, R be the set of requests and C be the set of contents. Each content $c \in C$ has a size and a life time (submission period and removal period) associated to it. Each server $j \in S$ has a storage capacity and a bandwidth limit. There is an overlay network interconnecting the servers and between each pair of servers j and l there are two delay values (from j to l and from l to j) and one Round Trip Time (RTT), that corresponds to the sum of such delays. We assume that each client (1) establishes a connection with the CDN system, (2) makes a single request and (3) leaves the CDN system after his or her request is completely handled. Associated to each request $i \in R$ there are a desired content in C, an arrival period in T, a server $o_i \in S$ that represents the server to which the client is connected and a local delay ld_i, that represents the distance between o_i and the computer of the client. There are also a delay limit TD_i that represents the maximum delay tolerated and two values of bandwidth, minimum (BR_i) and maximum (BX_i), associated to each request $i \in R$. As clients have limited bandwidth, it is not always possible to deliver the desired content in a single period, meaning that several periods are typically required to fully handle a request. In this paper, we use the maximum bandwidth BX_i multiplied by the period length (δ) as demand (D_i) for request i, meaning that whenever possible, a client is served in its maximum bandwidth. If some part of the demand of a client i is not delivered in some period t, it means that i is not handled in its full capacity in period t and this missing part is called backlog of request i in period t. In order to fully handle clients' demands, multiple servers are allowed to handle the same request in the same period. There are costs associated for handling a request by each server on the CDN and there are also costs associated to the replication of the contents. Since the objective of the RPRDP is to reduce the operational costs of CDN providers and also improve the quality perceived by end users, the objective function of the problem can be expressed by (1):

$$Min \sum_{i \in R} \sum_{j \in S} \sum_{t \in T} c_{ijt} x_{ijt} + \sum_{i \in R} \sum_{t \in T} p_{it} b_{it} + \sum_{k \in C} \sum_{j \in S} \sum_{l \in S} \sum_{t \in T} h_{kjlt} w_{kjlt} \quad (1)$$

where x_{ijt} is a continuous variable that represents the fraction of the content asked by request i handled by server j in period t; b_{it} is an integer variable that represents the backlog of request i in period t, w_{kjlt} is a binary variable that assumes 1 if server j replicates content k from server l in period t; c_{ijt} represents the cost of handling request i by server j, in period t. These costs are calculated based on the fitness of each server, thus, servers that can better handle a request, considering its QoS requirements, have lower costs; p_{it} is the penalty for backlogging request i in period t. This penalty is greater than the cost for handling request i by any server in period t; h_{kjlt} is the cost that server j pays for downloading content k from server l in period t.

In the RPRDP every request must be fully handled, servers bandwidth and storage capacity must be verified and at least one replica of each content must exist in each period (except for those contents that were already removed, or not submitted yet).

Besides, the backlogged portion of a request must be handled in some of the periods ahead and servers are allowed to handle requests if, and only if, they have replicas of the desired contents. Moreover, contents downloaded by a server j from a server l in some period t will be available on server j only in period $t + 1$, meaning that one period is necessary to download a content. The reader interested in a more detailed description of the RPRDP is referred to [11], where a mathematical formulation that better describes the problem is presented.

3 Hybrid network heuristic

In order to improve the results obtained in [11], a new exact approach to RPP and a new exact approach to RDP are proposed in this paper. The two exact approaches and a demand estimators are combined into a new hybrid method, called Hybrid Network Heuristic (HNH) and its components are presented in the remaining of this section.

3.1 A mathematical formulation for the RPP

In [11], greedy heuristics were used to solve the RPP and it is known that greedy algorithms are not always competitive with mathematical formulations in terms of quality of the solutions. Thus we decide to propose a mathematical formulation to verify the viability of using such formulation instead of greedy algorithms.

The greedy algorithm used in *HC* heuristic [11], called *GAS* from now on, tries to insert contents in servers where there are more requests for them, always regarding disk space constraints. A relationship between RPP and the Generalized Assignment Problem (GAP) [13] can be established by changing the problem nomenclature, however, a minor change in the GAP formulation is necessary in order to better fit the RPP. In GAP, a single object must be placed in one knapsack. In RPP, a content must be placed in *at least* one server. A mathematical formulation for the RPP based on the GAP formulation is now presented. This formulation uses the following notation:

- m Number of knapsacks (servers);
- n Number of objects (contents);
- pft_{ij} The profit of object i when assigned to knapsack j (demand for content i in server j);
- wgt_{ij} The weight of object i when assigned to knapsack j (size of content i);
- cpt_j The capacity of knapsack j (disk space of server j);
- y_{ij} Binary variable that assumes 1 if object i is assigned to knapsack j and 0 otherwise.

The Adapted GAP (AGAP) formulation can be described as follows:

$$Max \sum_{j=1}^{m} \sum_{i=1}^{n} pft_{ij} y_{ij} \tag{2}$$

$$S.t.$$

$$\sum_{i=1}^{n} wgt_{ij} y_{ij} \leq cpt_j, \quad j \in M = \{1, \ldots, m\}, \tag{3}$$

$$\sum_{j=1}^{m} y_{ij} \geq 1, \quad i \in N = \{1, \ldots, n\}, \tag{4}$$

$$y_{ij} \in \{0, 1\}, \quad \forall i \in N, \forall j \in M. \tag{5}$$

The objective function (2) maximizes the profit given by the assignment of objects to knapsacks. Constraints (3) ensure that no knapsack becomes overloaded. Constraints (4) ensure that each object is placed in at least one knapsack and Constraints (5) are the integrality and non-negativity constraints. In RPP nomenclature, the objective function (2) maximizes the amount of demand handled by servers. Constraints (3) ensure that servers disk space capacity are not violated. Constraints (4) ensure that at least one replica of each content exists and Constraints (5) are the integrality and non-negativity constraints.

The *AGAP* formulation makes possible the use of an exact approach to solve the RPP. Experimental results show that AGAP can improve the quality of the solutions. Besides, the observed results clear indicates that the RPP can be treated as a variant of the Knapsack Problem thus, adaptations of good algorithms for solving the Knapsack Problem will be addressed in future studies. The results also corroborate the claims of [2,14] where greedy algorithms, such as *GAS*, are pointed as suitable solutions for problems similar to the RPRDP.

3.2 Demand estimator

In order to further improve the quality of the solutions, four different demand estimators were used in the computational tests: the Average Based Estimator, used in [11]; the Brown's Linear Double Exponential Smoothing, used in [2]; the Holt's Two-Parameters Double Exponential Smoothing (*HTDES*), exposed in [15]; and an estimator proposed by the authors of this paper. In the tests, *HTDES* estimator outperformed the others considering the average error in the forecasts and the number of best results obtained. Therefore, we will only focus on the *HTDES* estimator in this paper.

According to [15], exponential smoothing based estimators are very popular due to simplicity and accuracy. This class of estimators is able to provide very good forecasts, despite the fact that the calculations involved are quite simple and fast. Thus, exponential based estimators can be considered one of the most suitable estimators class for environments in which time is a critical factor. Another motivation for the use of this family of estimators is the successful use of the Brown's Linear Double Exponential Smoothing in [2] to forecast the demands of a problem similar to the RPRDP addressed in this paper.

Suppose that the current period is t and it is necessary to predict the demand of r periods ahead. The forecasts z_{t+r}^{p} obtained by *HTDES* method in period t for period $t + r$ are provided by the Eqs. (6)–(8).

$$z^{P}_{t+r} = z'_t + r z''_t, \tag{6}$$

$$z'_t = \alpha z_t + (1-\alpha)(z'_{t-1} + z''_{t-1}), \tag{7}$$

$$z''_t = \lambda(z'_t - z'_{t-1}) + (1-\lambda) z''_{t-1}, \tag{8}$$

In these equations, z_t is the real demand observed in period t, z'_t and z''_t are called exponentially smoothed values. The constants α and λ are called smoothing constants.

In this method two values are used to smooth the extreme values of the series using a different smoothing constant in each value. In other words, this method uses one constant to smooth the series level and another one to smooth the series trend.

Despite our efforts, it was not possible to find a single value for the smoothing constants of the method that could produce reasonable results as in [2], where one single value produced the best results. During the experiments, the best results were obtained when the values of α and λ were adjusted dynamically. Based on this fact, we decided to use the *backforecast* technique, as exposed in [15], to dynamically adjust the values of α and λ. The technique consists in testing several values for the constants, using demands of period $t-1$ trying to forecast demands of period t. As period t is the current period, its demands are already known and we can evaluate the forecast error for all values of the smoothing constants. The values that produce the smallest error are then used with demands of period t in order to forecast the demands of period $t+1$. It is important to mention that the use of *backforecast* improved considerably the performance of both Double Exponential Smoothing estimation methods (Brown's and Holt's). We used in the *backforecast* all values for α and λ in the interval $[0.1, 0.9]$ with step of 0.1 in both constants.

3.3 The network traffic model for the RDP

In this section we present a new exact method for solving the RDP that consists in solving a Minimum Cost Flow Problem (MCFP) [16] in a specific network. In this network, the vertexes represent requests, servers, flow source and flow sink. A specific demand is associated to each vertex depending on what it represents. Each arc in the network has a cost, a capacity and a class. The cost is the value paid for each unit of flow that traverses the arc and the capacity is the maximum amount of flow allowed in the arc. As the vertexes represent different entities of the problem, the relationships among them are not always equal. The difference in relationships is modeled by arc classes. A solution of the MCFP in this network is equivalent to a solution of the RDP provided by the mathematical formulation exposed in [11], referred as *RF* from now on. The network for the RDP can be built by the following steps:

1. Let $R* \subseteq R$ be the set of requests that are active (requests that just arrived or not fully handled) in the current period. Create a pair of vertexes i' and i'' for each request $i \in R*$;
2. Create a vertex for each server;
3. Create a vertex to represent a server with infinite capacity, i.e., a backlog server (SB);
4. Create a source vertex (FS) and a sink vertex for the flow (FF);

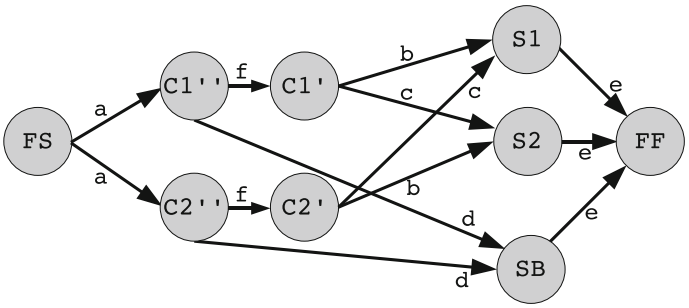


Fig. 1 Network flow model for the RDP

5. Establish demands for the vertexes as follows: FQ units in FS, $-FQ$ in FF and zero in the other vertexes, where $FQ = \sum_{i \in R*} D_i$, $D_i = \delta BX_i + b_{i(t-1)}$. In other words, FQ is the sum of the demands of active requests;
6. Create arcs linking FS to each vertex i''. These arcs belong to class a each one has cost zero and capacity D_i;
7. Create arcs linking the vertexes i'' to the vertexes i'. For each request, one arc will be created linking the two vertexes that represent this request. These arcs belong to the class f and have cost zero and capacity δBX_i;
8. Create arcs linking vertexes i' to the servers vertexes in order to model the request handling. These arcs can belong to two different classes. Arcs of class b will link requests to servers that have a replica of the desired content. These arcs have cost c_{ijt}/LC_i, where LC_i is the length of the content requested by i, and capacity δBX_i. Arcs of class c link requests to servers that do not have replicas of the desired content, modeling infeasible associations. These arcs have capacity δBX_i and a very high cost. Note that in this model there is only one arc linking a request to a server. In other words, a server can not be pointed by an arc of class b and by an arc of class c coming from the same request;
9. Link all vertexes i'' to the backlog server. The arcs created for such linking are associated to class d and have cost equal to p_i and infinite capacity;
10. Finally, it is necessary to create arcs linking all servers (including the backlog server) to the sink vertex. Such arcs have capacity set to the δMB_j, cost set to zero and belong to class e.

Figure 1 shows the network built following the mentioned steps for an instance with two clients' requests ($C1$ and $C2$) and two servers ($S1$ and $S2$). In this example $S1$ has the replica desired by $C1$ and $S2$ has the replica desired by $C2$. After all ten steps are concluded, we can solve the MCFP in the network using any suitable algorithm [16] and the solution provided is equivalent to the solution of the original RDP.

This network model is able to deal with backlogs of previous periods because it allows that an amount of traffic greater than the download capacity of the requests reaches the first request vertex. As the quantity of traffic FQ is equal to the sum of demands, all arcs of class a will be saturated, meaning that a traffic equal to the demand of client i will arrive in the vertex i''. There is a special case of demand when there are only a few bytes remaining to deliver. Suppose that a request i has a maximum

bandwidth of 100 bytes per second, the period duration is 2 seconds and the content desired by i is 410 bytes long. In this example we handle request i using 200 bytes as demand in 2 periods and 10 bytes in the third. Thus, in periods that the remaining bytes are less than δBX_i, we use the remaining bytes as demand. The class f arc, that links vertexes that represent the same request, has capacity set to the maximum download capacity of the requests. Thus, if a request has a demand greater than its capacity, the exceeding demand will pass through the arcs of class d, that represent the requests backlog. It is important to mention that the backlogged bytes of a request i will be as $b_{i(t-1)}$ in the next period. In order to clarify how the backlog works, let's use as example the same request used earlier. Now let's suppose that only 190 bytes can be delivered to such request in the first period. As there are 10 bytes missing on the demand of request i, then the backlog of i in the first period is $b_{i1} = 10$. Since the demand of a request is given by $D_i = \delta BX_i + b_{i(t-1)}$, $D_i = 210$ in the second period. Now let's suppose that 195 bytes can be handled in the second period. As it can be calculated, $b_{i2} = 15$ bytes. In the third period, as explained earlier, the amount of bytes that must be delivered is less than the capacity of request i and thus, $D_i = b_{i2} + missingPart = 25$. If D_i is fully delivered in the third period, request i can be removed since the whole content is delivered. In the case of some part of D_i is not delivered, this part will be used as b_{i3} in the fourth period and so on. The traffic that arrives in a vertex i' must be passed through arcs of class b or c. If there are no arcs of class b, a MCFP algorithm will transfer the traffic passing through the arcs of class f to the arcs of class d. The bandwidth constraints of servers are ensured by arcs of class e. When a server exceeds its capacity, a MCFP algorithm will transfer the traffic to other arcs of class b or even to arcs of class d if necessary. Although the arcs of class c will never be used in any optimal solution, their presence is necessary because some algorithms for the MCFP maintain the feasibility and pursue the optimality [16] during the optimization process and because we intend to use these arcs for speed up purposes in future studies.

The proposed network model allows the use of several graph based techniques to solve the problem and shows that the RDP is a variant of the Multi-Commodity Transportation Problem [16] where the facilities have multiple products, each client demands only one kind of product, and the capacity constraints are in the facilities/clients instead of in the transportation paths, which is usually found in the literature. To the best of our knowledge, this is the first network flow model proposed to the RDP.

The complexity analysis of the new network model is now presented. Let the number of servers be denoted by $|S|$, and the number of requests be denoted by $|R|$ in this complexity analysis. The model uses $2|R|$ vertexes to represent requests, $|S| + 1$ vertexes to represent all servers, including the backlog server, and 2 additional vertexes, the source and the sink. Thus the vertexes creation procedure takes $2|R| + |S| + 1 + 2$ steps, and has the worst case complexity $O(|R| + |S|)$. The model also creates $|R|$ arcs of class a, $|R|$ arcs of class f, $|R|$ arcs of class d and $|S| + 1$ arcs of class e. For the classes b and c together $|S||R|$ arcs are created, resulting in $3|R| + |S| + 1 + |S||R|$ arcs that is $O(|S||R|)$. Since the total number of steps to create the network is given by the number of arcs plus the number of vertexes, the network creation process complexity is given by $O(|R| + |S|) + O(|S||R|) = O(|S||R|)$ which is a polynomial complexity. Since the network creation is polynomial and there are polynomial algorithms to solve

Algorithm 1 HNH heuristic

```
1: pos = initial replica positioning
2: req = list of active requests
3: blg = list of backlogs of requests
4: for Each period of time do
5:    Update req
6:    SolveNetworkModel(pos, req, blg)
7:    dem = Estimated future demand
8:    pos = solution of RPP given by AGAP using dem as profit.
9: end for
```

the MCFP [16], it is clear that the RDP can be solved in polynomial time, and thus it belongs to P-class.

3.4 The hybrid network heuristic

By combining the two approaches proposed in Sect. 3, *AGAP* formulation presented in 3.1, the network flow model, introduced in 3.3, and the demand estimators, exposed in 3.2, a new hybrid heuristic, called Hybrid Network Heuristic (HNH), is proposed and described in Algorithm 1.

The *HNH* heuristic proceeds as follows: at each period the arriving requests are put in req along with not fully handled requests. Next, the network model is solved using the replica positioning and the list of backlogs of the previous period, as well as the updated list requests as input parameters. This method updates the req and blg lists. The req is updated by removing requests that are totally handled (no bytes left to deliver). The blg list is also updated in order to contain the new backlog values for the requests. The third step is to forecast the demand for the next period. The final step of each period is to solve the RPP. The AGAP formulation is used taking the forecast demand of the third step as input parameter. The new replica positioning determined by AGAP is stored in pos and it will be used for solving the network flow model in the next period. The main difference between *HC* [11] and *HNH* is that these two heuristics apply different techniques for each step. By applying *AGAP* and the HTDES estimator, *HNH* is able to obtain gaps lower than *HC*. A reduction on total computational times is also observed due to the use of the network traffic model.

It is important to emphasize that the heuristics presented in this section and in [11] must be fast. Since the time to provide a solution is a critical factor on real world, the algorithms used are generally simple [2,4,11] because, sometimes, sophisticated algorithms may take hours to run. However, it is also important to say that the heuristics presented in [11] and in this paper provide good results for the problem in spite of their simplicity.

4 Computational results

The algorithms presented in the previous sections were implemented in C++ using g++ version 4.3 and executed on a Quad-Core with 2.83 GHz/core, 8 Gigabytes of

Table 1 Instances characteristics

Characteristic	Smallest	Highest
# of requests	445	3,762
# of periods	15	35
# of servers	10	50
# of contents	5	15
Server disk (GB)	0.1	150
Server bandwidth (GB)	1.5	4
Client bandwidth (Mbps)	0.6	5.6

RAM using Linux (kernel 2.6). To solve all the formulations and network problems the solver CPLEX 11.2 [17] was used.

The instances used for computational tests are the same 60 instances used in [11]. These instances are available on the LABIC project [18] website, as well as a technical report explaining how such instances were created [19]. Table 1 summarizes the major characteristics of the instances, listed in the first column. Column two presents the smallest value for the characteristic found in the entire instances set. Column three exposes the highest value found in the entire set. It is important to mention that these instances were created based on the literature for similar problems and in real data of Brazilian Internet providers available by the time of their creation. Further details regarding the instances characteristics are documented in [19].

The remaining of this section is organized as follows: first, the time and gap results obtained by *AGAP* and *GAS* are analyzed. The gaps presented are calculated in relation to the *FD* formulation [11] using the following equation: $\text{gap} = (of(method) - of(FD))/of(FD)$, where of is the objective function value of the approach inside the parentheses. Notice that gap is not the same as GAP. The gap is the relative difference between two methods and GAP is the acronym for Generalized Assignment Problem. The time results comparing *RF* and the network traffic model are exposed in 4.2. The last results presented are the gaps and times of *HNH* and two heuristics presented in [11] (*OGHS* and *HC*).

4.1 Results for RPP approaches

This section compares the performances of *AGAP* and *GAS*. The average computational times for both approaches are exposed in Table 2. First column shows the instances size in number of servers. Columns two and three expose the average computational time, in seconds, and the standard deviation for *GAS* algorithm, respectively. Columns four and five show the same information for *AGAP*. The cells in boldface are the best average times. As it can be perceived, *AGAP* is more time consuming than *GAS*. However, the computational times for *AGAP* are also low even for the greater instances, making it a suitable option for solving the RPP.

Table 3 exposes the number of best results obtained for both replication approaches. Line two shows the number of cases in which *GAS* obtained the best results. Line three presents the number of cases in which *AGAP* outperformed *GAS*. Line four exposes

Table 2 Times: GAS × AGAP

# Servers	GAS		AGAP	
	Average (s)	Standard deviation	Average (s)	Standard deviation
10	**0.001**	0.000	0.017	0.007
20	**0.001**	0.000	0.027	0.013
30	**0.001**	0.000	0.023	0.010
50	**0.002**	0.003	0.062	0.030

Table 3 Number of best results: GAS × AGAP

Algorithm	# of best results
GAS	2
AGAP	16
Draws	42
Total	60

Table 4 Times: RF × network model

# Servers	RF		Network model	
	Average (s)	Standard deviation	Average (s)	Standard deviation
10	0.15	0.01	**0.09**	0.01
20	0.58	0.07	**0.30**	0.02
30	1.26	0.09	**0.62**	0.05
50	3.73	0.39	**1.61**	0.08

the number of cases in which the two algorithms presented the same result. The results show that *AGAP* can indeed produce solutions with better quality than *GAS* which is considered, to the best of our knowledge, the best heuristic in the literature for this problem. Since the average times of *AGAP* for the greater instances are less than 65 ms, as shown in Table 2, we consider it a suitable option and decided to use it as the part of the *HNH* heuristic. This decision is based on the fact that the Network Traffic model, presented in Sect. 3.3, is faster than the RF formulation (as exposed in Sect. 4.2), therefore we can spend a little more time in the solution of the RPP in order to obtain better results. Remark that, although the average gaps for the entire set of instances indicate that *AGAP* (5.42 %) outperforms *GAS* (5.54 %), these average gaps can not be considered statistically different.

4.2 Results for RDP approaches

The average computational times for request distribution approaches are exposed in Table 4. The results show that the Network Model is much less time demanding than *RF* formulation, meaning that the network model is the best option for solving the RDP.

Table 5 Computational times for *FD*, *HC* and *HNH*

# Servers	FD		HC		HNH	
	Average (s)	Standard deviation	Average (s)	Standard deviation	Average (s)	Standard deviation
10	10.48	3.89	0.20	0.01	**0.11**	0.01
20	139.33	62.41	0.73	0.04	**0.35**	0.03
30	110.74	36.71	1.64	0.14	**0.68**	0.06
50	394.10	132.31	4.54	0.48	**1.75**	0.08

As both methods are exact, there is no need to compare the quality of the solutions obtained since they provide the same result.

4.3 Results for RPRDP

This section presents the results obtained by HNH along with the results obtained by *FD* formulation and the *HC* and *OGHS* heuristics. Both *HC* and *OGHS* use the same framework of *HNH*. The difference is that they apply different techniques on each step. *HC* uses *RF* for solving the RDP, an estimator based on average demand and *GAS* for solving RPP. *OGHS* is an optimized version of the algorithm used in a real CDN provider. It uses *RF* for solving the RDP and, no demand estimator and an algorithm based on the Least Recently Used strategy for solving the RPP. The reader interested in more details on *HC* and *OGHS* is referred to [11]. Table 5 shows the average computational times and the standard deviations for *FD* formulation, *HC*, and *HNH* heuristics. The first column shows the size of the instances size in number of servers. Second and third columns show the average computational times, in seconds, and the standard deviation for *FD*, respectively. Columns 4 and 5 are related to *HC* heuristic and expose the same information of columns 2 and 3. Column 6 depicts the average computational times of *HNH* and column 7 shows the standard deviations of *HNH*. The average times of *OGHS* are not shown because they are very close to the times of *HC*.

It is clear that *HNH* is less time demanding than the other approaches. Although *AGAP* is more time consuming than *GAS*, the total execution times of *HNH* are lower because solving the network traffic model takes much less time than solving *RF* formulation, used by *HC* and *OGHS*. The computational times of *FD* formulation are much higher than the ones obtained by the hybrid methods and it also has higher values of standard deviation.

Table 6 exposes the average gaps, in %, of *OGHS*, *HC*, *HNH*, respectively. The table presents, for each heuristic, the average gaps and the standard deviations for each instance size. The values in boldface are the best results. The line "50" is marked with "*" meaning that the gaps, for these instances, are calculated based on the best solutions found by CPLEX. However, some of them are not optimal due to the scalability problem of *FD* formulation [11]. As *FD* formulation creates huge mathematical models, CPLEX was not able to find the optimal solution to all cases due to the lack

Table 6 Gaps: *OGHS* × *HC* × *HNH*

# Servers	OGHS		HC		HNH	
	Average (%)	Standard deviation	Average (%)	Standard deviation	Average (%)	Standard deviation
10	11.65	4.32	6.30	2.32	**4.55**	1.13
20	12.11	5.05	5.72	2.29	**4.18**	1.20
30	6.42	1.91	4.17	0.74	**3.65**	0.55
50*	10.68	5.66	5.99	2.52	**4.65**	1.53
Average	10.22	–	5.55	–	**4.26**	–

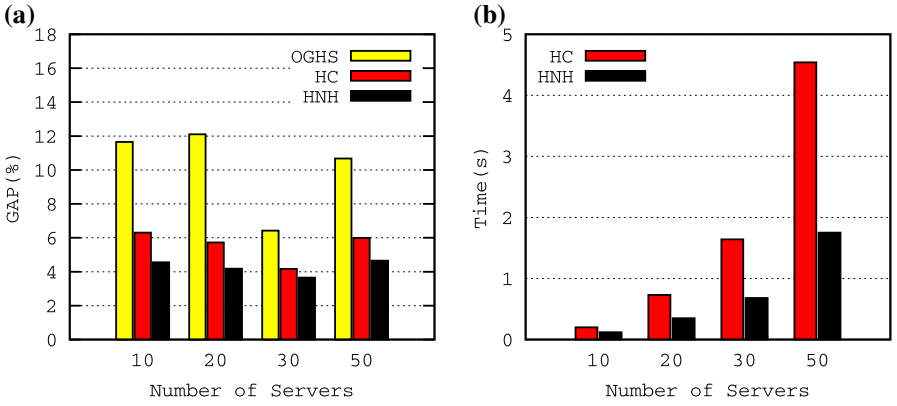


Fig. 2 Gap and time analysis

of memory in the computer used. In order to illustrate this, we asked CPLEX to write down the models for theses instances and we observed that, for those cases in which CPLEX was not able to find the optimal solutions, the files containing the models were more than 1 GB long, indicating that *FD* formulation may not be suitable for greater instances. Although *HNH* and *HC* do not have any knowledge about the future attributes, the average gaps of these heuristics are about 4.26 and 5.55 % of the optimal solution, respectively. The *HNH* heuristic has the best average gaps when compared to the other online approaches and also has the best average times, outperforming the other online approaches.

Figure 2 summarizes the obtained results. Figure 2a presents the average gaps for the instances of each size obtained by *HC*, *HNH*, and *OGHS* heuristics when compared to *FD*. Figure 2b shows the averages of computational times, expressed in seconds, for *HC* and *HNH*. Notice that the *HNH* heuristic outperforms the *HC* and the *OGHS* heuristic on average results. As mentioned before, the time results for *OGHS* are very similar to the ones obtained by *HC*.

5 Concluding remarks

This paper describes the Replica Placement and Request Distribution Problem. Besides that, a new hybrid heuristic *HNH* that outperforms all other heuristics for the online version of the problem is proposed. The new heuristic uses a network traffic model for solving the Request Distribution Problem and a mathematical formulation, based on the Generalized Assignment Problem, for solving the Replica Placement Problem. The *HNH* also uses a demand estimator based on Double Exponential Smoothing along with the backforecasting technique. The complexity of the new network traffic model is analyzed proving that the Request Distribution Problem belongs to P-class. The results show that *HNH* can achieve good solutions in much less computational time when compared with *FD*, which is applied to the offline version of the problem. *HNH* also outperforms *OGHS* and *HC* reducing the average gap in more than 1 % and the computational times in more than 50 %. All approaches consider several realistic details of networks like minimal bandwidth, maximum delay, better use of network capacity and servers load, which are often not treated simultaneously in the literature.

Acknowledgments The authors thank the support of CNPq, CAPES and FAPERJ.

References

1. Kurose, J., Ross, K.: Computer Networking: a Top-Down Approach Featuring the Internet. Addison-Wesley, Boston (2003)
2. Aioffi, W., Mateus, G., Almeida, J., Loureiro, A.: Dynamic content distribution for mobile enterprise networks. IEEE J. Sel. Areas Commun. **23**(10), 2022–2031 (2005)
3. Almeida, J.M., Eager, D.L., Vernon, M.K., Wright, S.J.: Minimizing delivery cost in scalable streamingcontent distribution systems. IEEE Trans. Multimed. **6**, 356–365 (2004)
4. Bartolini, N., Presti, F., Petrioli, C.: Optimal dynamic replica placement in content delivery networks. In: The 11th IEEE International Conference on Networks 2003, ICON 2003, pp. 125–130 (2003)
5. Sharma, A., Venkataramani, A., Sitaraman, R.K.: Distributing content simplifies ISP traffic engineering. In: Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, pp. 229–242 (2013)
6. Tenzakhti, F., Day, K., Ould-Khaoua, M.: Replication algorithms for the world-wide web. J. Syst. Archit. **50**, 591–605 (2004)
7. Wauters, T., Coppens, J., Dhoedt, B., Demeester, P.: Load balancing through efficient distributed content placement. In: Next Generation Internet Networks, pp. 99–105 (2005)
8. Bektas, T., Oguz, O., Ouveysi, I.: Designing cost-effective content distribution networks. Comput. Oper. Res. **34**, 2436–2449 (2007)
9. Huang, C., Abdelzaher, T.: Towards content distribution networks with latency guarantees. In: 12th IEEE International Workshop on Quality of Service, 2004. IWQOS 2004, pp. 181–192 (2004)
10. Zhou, X., Xu, C.Z.: Efficient algorithms of video replication and placement on a cluster of streaming servers. J Netw. Comput. Appl. **30**, 515–540 (2007)
11. Neves, T., Drummond, L., Ochi, L., Albuquerque, C., Uchoa, E.: Solving replica placement and request distribution in content distribution networks. Electron. Notes Discrete Math. **36**, 89–96 (2010). doi:10.1016/j.endm.2010.05.012
12. Wu, L.Y., Zhang, X.S., Zhang, J.L.: Capacitated facility location problem with general setup cost. Comput. Oper. Res. **33**, 1226–1241 (2006)
13. Martello, S., Toth, P.: Knapsack Problems. Wiley, New York (1990)
14. Tang, X., Xu, J.: On replica placement for QoS-aware content distribution. IEEE INFOCOM **2004**, 806–815 (2004)
15. Morettin, P.A., de Castro Toloi, C.M.: Modelos para Previsão de Séries Temporais, vol. 1. Instituto de Matemática Pura e Aplicada (1981)

16. Ahuja, R.K., Magnanti, T.L., Orlin, J.: Network Flows: Theory, Algorithms, and Applications, 1 edn. Prentice Hall, Englewood Cliffs, NJ (1993)
17. ILOG S.A.: CPLEX 11 User's Manual (2008)
18. LABIC: World Wide Web. http://labic.ic.uff.br/ (2005). Accessed 15 Feb 2012
19. Neves, T.: Synthetic instances for a management problem in content distribution networks. In: Technical report, UFF (2011) http://labic.ic.uff.br/ (url world wide web) (2011). Accessed 15 Feb 2012