# Energy Efficient Content Distribution

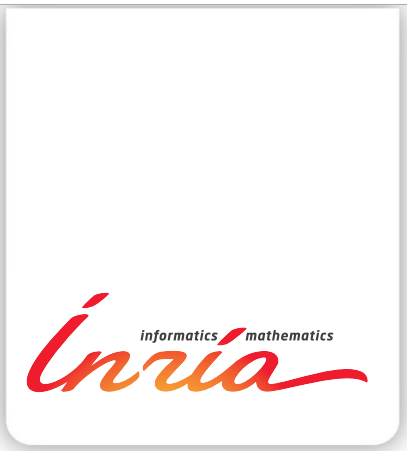
Julio Araujo, Frédéric Giroire, Yaning Y.L. Liu, Remigiusz Modrzejewski, Joanna Moulierac


# Energy Efficient Content Distribution

J. Araujo , F. Giroire , Yaning Liu , R. Modrzejewski, J. Moulierac

RESEARCH
REPORT
N° 8091
Mai 2013
Project-Team Mascotte

ISSN 0249-6399 ISRN INRIA/RR--8091--FR+ENG




# Energy Efficient Content Distribution *

J. Araujo † ‡, F. Giroire †, Yaning Liu §, R. Modrzejewski†, J. Moulierac†

Project-Team Mascotte

Research Report n° 8091 — version 2 — initial version Mai 2013 — revised version Mai 2013 — 27 pages

**Abstract:** To optimize energy efficiency, network operators try to switch off as many network devices as possible. Recently, there is a trend to introduce content caches as an inherent capacity of network equipment, with the objective of improving the efficiency of content distribution and reducing network congestion. In this work, we study the impact of using in-network caches and content delivery network (CDN) cooperation on an energy-efficient routing. We formulate this problem as Energy Efficient Content Distribution and propose an integer linear program (ILP) and an efficient heuristic algorithm to solve it. The objective is to find a feasible routing, so that the total energy consumption of the network is minimized subject to satisfying all the demands and link capacity. We exhibit the range of parameters (size of caches, popularity of content, demand intensity, etc.) for which caches are useful. Experimental results show that by placing a cache on each backbone router to store the most popular content, along with well choosing the best content provider server for each demand to a CDN, we can save about 20% of power in the backbone, while 16% can be gained solely thanks to the use of caches.

**Key-words:** Energy Efficiency, Integer Linear Programming, Content Delivery Network, Network Cache, Future Internet

* This work was partially supported by région PACA.

† {julio.araujo, frederic.giroire, remigiusz.modrzejewski, joanna.moulierac} @inria.fr - MASCOTTE Project, I3S (CNRS & UNS) and INRIA, INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex France.

‡ Supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil, under postdoctoral fellowship PDE 202049/2012-4.

§ yaning.liu @jcp-consult.com, JCP-Consult, France

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

# Distribution des Données Efficace en Énergie

**Résumé :** Pour optimiser l'efficacité énergétique dans un réseau, les opérateurs doivent éteindre un nombre maximum d'équipements réseau. Récemment, il a été propose de rajouter des caches à l'intérieur des nœuds réseaux dans l'objectif d'améliorer la distribution de contenus et de réduire la congestion des réseaux. Dans ce travail, nous étudions l'impact de l'utilisation de caches réseaux (in- network caches) et de leur coopération avec les Content Delivery Networks (CDN) sur l'énergie consommée par le routage. Nous modélisons ce problème, la Distribution de Données Efficace en Énergie, par un programme linéaire en nombres entiers et proposons une heuristique en temps polynomial pour le résoudre efficacement. L'objectif est de trouver un routage réalisable qui minimise la consommation énergétique du réseau tout en satisfaisant les demandes de contenus. Nous exhibons les valeurs des paramètres (tailles des caches, popularités des données, ...) pour lesquelles ces caches sont utiles. Des expérimentations montrent qu'en plaçant un cache sur chaque routeur d'un réseau backbone pour stocker le contenu le plus populaire, ainsi qu'on choisissant le meilleur serveur pour chaque demande à un CDN, environ 20% de l'énergie du backbone peut être sauvée, dont 16% du gain est du aux seuls caches.

**Mots-clés :** Efficacité énergétique, Programmation linéaire entière, Réseau de Livraison de Données, Caches des réseau, Internet future

# 1 Introduction

Energy efficiency of networking systems is a growing concern, due to both increasing energy costs and worries about $CO_2$ emissions. In [1] it is reported that Information and Communication Technology sector is responsible for up to 10% of global energy consumption. 51% of that is attributed to telecommunication infrastructure and data centers. Thus, saving power is important. Backbone network operators study the deployment of energy-efficient routing solutions. The general principle is to aggregate traffic in order to be able to turn off a maximum number of devices [2–5].

On the other hand, in order to reduce network load and improve quality of service, content providers and network operators try to disaggregate traffic by replicating their data in several points of the networks, reducing the distance between this data and their users. Recent years have seen, along the growing popularity of video over Internet, a huge raise of traffic served by Content Delivery Networks (CDNs). These kinds of networks operate by replicating the content among its servers and serving it to the end users from the nearest one. CDNs deliver nowadays a large part of the total Internet traffic: estimation ranges from 15% to 30% of all Web traffic worldwide for the single most popular CDN [6]. Chiaraviglio et al. [7,8] have shown how the choice of CDN servers impacts the backbone energy consumption. More precisely, they aim at turning off network devices by choosing, for each demand from a client to a content provider, the best server of this CDN while being energy aware.

Here, we go further on this idea by also considering the usage of caches on each of backbone routers, while still taking into account the choice of CDN servers. It is important to mention that there have been several proposals for developing global caching systems [9], in particular recently using in-network storage and content-oriented routing to improve the efficiency of content distribution by future Internet architectures [10–12]. Among these studies, we mention that in this paper we do not assume any specific technology for future Internet architectures, nor anything else that would require major overhaul of how the Internet works. Thus, there is no content routing among our caches. We assume that a cache serves a single city, taking all of its contents from the original provider. We consider that caches can be turned on or off. Thus, there is a trade-off between the energy savings they allow, by reducing network load, and their own consumption.

We propose an Integer Linear Programming (ILP) formulation to reduce energy consumption by using caches and properly choosing content provider servers, for each demand. We implemented this formulation on the ILP solver CPLEX [13] version 12 and made experiments on real, taken from SNDlib [14], and random, Erdős-Rényi [15], network topologies. We study the impact of different parameters: size of caches, demand intensity, network size, etc. In particular, we found that almost maximal energy gain can be achieved, in our scenarios, by caches of the order of 1 TB. Larger caches do not lead to significantly better gains. We discuss the increase of cache usage with network size. Experimental results show potential energy savings of around 20% by putting devices to sleep outside peak hours; introducing CDN to the network without caches gives 16% savings; introducing caches to network without CDN also gives around 16% savings. Furthermore, we observed that the impact of caches is more prominent in bigger networks. To be able to quantify this effect, we propose

an efficient heuristic. This heuristic, called Spanning Tree Heuristic, allows us to obtain acceptable solutions in a time orders of magnitude shorter than solving the model directly with CPLEX. Furthermore, the heuristic accepts a parameter controlling a speed/quality trade-off. This trade-off is also studied in this paper.

The main take away of our work is thus that, by storing the most popular content in caches at each router and by choosing the best content provider server, we may save around 20% of power in the backbone.

The paper is organized as follows. We discuss the related work in Section 2. We present the problem and its formulation in Section 3. Section 4 describes how we built the instances used in the experimentations.Finally, we present the experiments we did and discuss the results in Section 5.

## 2 Related Work

Reducing energy consumption of the backbone network has been approached before multiple times. A model where it was achieved by shutting down individual links, chosen by an algorithm similar to described in this work, is studied in [5]. An interesting way of performing this in a distributed manner is shown in [4]. Energy efficient CDNs have also been studied recently. Authors in [16] propose to reduce energy consumption in CDN networks by turning off CDN servers through considering user service level agreements. In order to optimize the power consumption of content servers in large-scale content distribution platforms across multiple ISP domains, a strategy is proposed in [17] to put servers into sleep without impact on content service capability. Our work is different from these works, since they do not consider in-network caches.

Network caches have been used in global caching systems [9]. In recent years, several Information Centric Networking architectures, such as Cache and Forward Network (CNF) [10], Content Centric Networking (CCN) [12] and Net-Inf [11], have exploited in-network caching. Their objectives are to explore new network architectures and protocols to support future content-oriented services. Caching schemes have been investigated in these new Internet architectures [10, 18, 19]. Similar to our work, these works also use in-network caches, however they do not consider energy savings.

Energy efficiency in content-oriented architectures [20–22] with an in-network caching has been studied recently. In [20], authors analyze the energy benefit of using CCN by comparison to CDN networks. A further work [22] considered the impact of different memory technologies on energy consumption. Adding network caches that work transparently with current Internet architecture has been studied, with linear power models, in [23], where caches are added to backbone routers and in [24], where it is found that optimal placement during peak hours is in the access network. These works focus on the energy efficiency considering data delivery and storage, however, they do not take into account the energy savings by turning on/off network links. Authors in [21] extend GreenTE [2] to achieve a power-aware traffic engineering in CCN network. It is different from our work, since we consider energy consumption of in-network caches that could be turned on or off, as well as a cooperation between network operators and content providers.

Most closely related to ours is the work from Chiaraviglio et al. [7, 8], which

enables the cooperation between network operators and content providers, to optimize the total energy consumption by an ILP formulation for both sides. In this paper, we also consider this cooperation to achieve such a total energy saving. Our work is an extension of this optimization problem formulation, through considering in-network caching.

# 3 Problem Modeling

What follows in this section is a discussion of model parameters, formal problem definition and a Mixed Integer Linear formulation used to solve it.

However, let us first informally recall the problem description and some assumptions we consider. Our goal is to save energy on a backbone network by aggregating traffic and turning off as many devices as possible. We consider that this network has a set of demands between pairs of routers and a set of demands to CDN servers in an overlay of this network. A demand to a CDN can be satisfied by any of its servers, which are placed in different routers of our backbone network. Thus, these demands have of course a single destination, but several possibilities of sources, one for each CDN server. Moreover, we consider that each backbone router of our network has a cache, with a limited amount of storage, that can only be used to satisfy demands to its router. Our goal is to satisfy all these demands, under the capacity contraints of CDN servers, caches and links, while minimizing the number of links and caches that are turned on.

## 3.1 Parameters

For in-network caches, it is still an open question: if and how they should be deployed. Therefore, we avoid making specific assumptions about the details. Once the question is answered, the model we propose can be updated to answer any possible specific concerns. However, the conclusions can change, if the actual parameters vary heavily from our estimates.

**Cache hit rate** A cache, located in each router, automatically caches the most popular content, potentially saving a fraction of any demand. Establishing this fraction is a non-trivial task. According to [25], content popularity follows a Zipf-like distribution. In their study, they computed the relation between cache size and hit rate for a trace of traffic towards YouTube. Note that this relation does not depend on the number of cache accesses, only on the relative size of the cache and all the content collection. This relation is shown on Figure 1, with the assumption that an average video is 100 megabytes. The figure shows results for two algorithms: *least recently used*, a classic caching algorithm, and *static most popular*, a simple algorithm proposed by the authors. For example with a cache of around 800GB the expected hit rate is around 17.7% using LRU and around 32.5% using the static algorithm, thus saving an equivalent fraction of traffic.

As the situation changes frequently, both regarding to the volume of popular content and available storage, we leave this fraction as a parameter of the model: $\alpha$ – the maximal part of any demand that can be served from a cache. Network operator can establish it empirically, by means of measurements. Typically, we take $\alpha \in [0.2, 0.35]$.

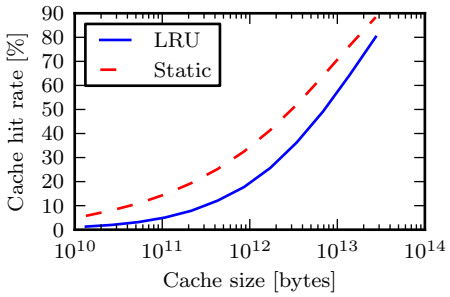



Figure 1: Cache hit ratio for YouTube trace, assuming average video size 100MB, following the results in [25].

**Cache power usage** In our model we deal with two types of equipment: links and caches. In practice, main energy drain of links are port cards and amplifiers. As can be seen in Powerlib [26], power requirements of single port cards suitable for long haul networks are well over 100 Watts, while other backbone cards can be as few as a quarter of that. For the caches, the main concern is fast mass storage. This has improved recently, with current SSD models offering 1TB of storage accessed at 10Gbps consuming below 10 Watts of power, for example [27].

Again, as the practical values in the time of implementation are hard to predict, we make this ratio another parameter of the model: $\beta$ – the power consumption of a cache divided by the power consumption of a link. Typically we take $\beta \in [0.1, 1]$.

## 3.2 Problem definition

We use a typical model, from the perspective of a backbone provider, where aggregated traffic between cities is expressed as a demand matrix. We augment this matrix to represent not only cities, but also content providers. This is motivated by the fact that content providers generate traffic that can easily be equal to that of a city.

Let us first formally define the optimization problem we are dealing with. We call it Energy Efficient Content Distribution. In this problem, we model the network by a graph $G = (V, E)$, for which we have a link capacity function $c : E \to \mathbb{R}_+$ and city to city demands $\tilde{\mathcal{D}}^t_s, \forall s, t \in V$. Moreover, we are given a set of content providers $P$. For each content provider $p \in P$, the subset of vertices of $V(G)$ containing its servers is given by the function $\mathfrak{L}_p \subseteq V(G)$. Each server placed in location $l \in \mathfrak{L}_p$ of a content provider $p$ has a capacity $C(\mathfrak{s}^l_p)$. We are also given demands from cities to content providers given by the function $\tilde{\mathfrak{D}}^p_s$, for every $s \in V, p \in P$. We consider that the data is replicated at each node of $\mathfrak{L}_p$. Finally, each node $v \in V(G)$ in the network has a cache of bandwidth capacity $b(v)$.

The goal of our problem is to find a feasible routing in $G$ satisfying all the demands $\tilde{\mathcal{D}}^t_s$ and $\tilde{\mathfrak{D}}^p_s$ under the capacity constraints $c(u, v)$, $C(\mathfrak{s}^l_p)$ and $b(v)$ that minimizes the total energy consumption of the network. By total energy consumption, we mean the energy used by the links plus the energy used by the caches. For each cache, despite of a fixed energy cost of turning it on, we also

consider an increased usage of energy in terms of its load.

### 3.3 Mixed Integer Linear Programming Formulation

First recall that our goal is to turn off links and caches in order to minimize the amount of energy used in the underlying network. Consequently, we use a variable $x_{uv}$ to indicate if the link $uv$ is turned on or off, for every $\{u, v\} \in E$. The model is normalized as to say that every link uses 1 unit of energy. We denote this unit $l_c$. We use a variable $y_v$ to indicate if the cache at router $v$ is turned on or off, for every $v \in V$. We say that a cache uses at most $\beta$ units of energy. Finally, we recognize that mass memory access can constitute a significant energy cost. Thus, we use a variable $z_v$ to indicate the load (fraction of used bandwidth) of the cache in router $v$. We assume that an idle cache uses fraction $\gamma$ of $\beta$ and its power consumption grows linearly with load to reach $\beta$ once fully utilized. The objective function is then written formally as:

$$\min \sum_{\{u,v\}\in E} x_{uv} + \sum_{v\in V} \beta\gamma y_v + \sum_{v\in V} \beta(1-\gamma) z_v.$$

Denote $\tilde{\mathcal{D}}$ and $\tilde{\mathfrak{D}}$ as the demands posed in the problem instance, respectively from other cities and content providers. A cache in a source router $s$, when turned on, allows to save a portion of any demand up to $\alpha$, call these savings respectively $\mathcal{C}$ and $\mathfrak{C}$. We will consider *reduced demands*, denoted $\mathcal{D}$ and $\mathfrak{D}$, which are the *input demands* with the caching savings subtracted:

$$\begin{aligned}
\mathcal{D}_s^t &= \tilde{\mathcal{D}}_s^t - \mathcal{C}_s^t \quad ,\forall s,t \in V,\\
\mathcal{C}_s^t &\le \alpha \tilde{\mathcal{D}}_s^t \quad ,\forall s,t \in V,\\
\mathfrak{D}_s^p &= \tilde{\mathfrak{D}}_s^p - \mathfrak{C}_s^p \quad ,\forall s \in V, p \in P,\\
\mathfrak{C}_s^p &\le \alpha \tilde{\mathfrak{D}}_s^p \quad ,\forall s \in V, p \in P.
\end{aligned}$$

Then, we record the load of the cache:

$$\sum_t \mathcal{C}_s^t + \sum_p \mathfrak{C}_s^p = z_s b(s) \quad ,\forall s,t \in V, \forall p \in P.$$

Finally, the load cannot exceed the capacity and needs to be zero if cache is off:

$$z_s \le y_s \quad ,\forall s \in V.$$

Each possible source $s \in V$ demands from each provider $p \in P$ an amount of data flow $\mathfrak{D}_s^p \ge 0$. The provider has a set of servers of $\mathfrak{s}_p^l$ located in a subset of nodes of the network $l \in \mathfrak{L}_p \subseteq V$. Each of those servers sends $\mathfrak{S}_p^{l,s}$ flow units, to collectively satisfy the demand:

$$\sum_{l\in\mathfrak{L}_p} \mathfrak{S}_p^{l,s} = \mathfrak{D}_s^p \quad ,\forall s \in V, p \in P.$$

Each server $\mathfrak{s}_p^l$ has a constrained capacity $C(\mathfrak{s}_p^l)$, which limits the demands it can satisfy:

$$\sum_{s\in V} \mathfrak{S}_p^{l,s} \le C(\mathfrak{s}_p^l) \quad ,\forall p \in P, l \in \mathfrak{L}_p.$$

| | Popularity | Server capacity | Server locations |
|---|---|---|---|
| CDN1 | 40 | 0.3 | Berlin Hamburg Duesseldorf Frankfurt Muenchen Nuernberg |
| CDN2 | 20 | 0.45 | Berlin Duesseldorf Frankfurt Muenchen |
| CDN3 | 15 | 0.6 | Berlin Frankfurt |
| CDN4 | 15 | 0.5 | Hamburg Frankfurt Muenchen |
| CDN5 | 10 | 0.2 | Berlin Duesseldorf Frankfurt Hamburg Muenchen Nuernberg Osnabrueck |

Table 1: Content Distribution Networks assumed for the *germany50* network.

Now we set flow constraints. By $f^s_{u,v}$ we denote the flow on edge $\{u, v\}$ corresponding to demands originating from $s$.

$$\sum_{v \in N_u} f^s_{v,u} - \sum_{z \in N_u} f^s_{u,z} =$$

$$= \begin{cases} -\sum_{p \in P} \mathfrak{D}^p_s - \sum_{t \in V} \mathcal{D}^t_s & u = s \\ \mathcal{D}^u_s + \sum_{\{p \in P | u \in \mathfrak{L}_p\}} \mathfrak{S}^{u,s}_p & \text{otherwise} \end{cases} \quad ,\forall s, u \in V.$$

Finally we consider capacities of links, denoted $c(uv)$. The constraints involve both kinds of flows and the on/off status of the links:

$$\sum_{s \in V} (f^s_{u,v} + f^s_{v,u}) + \leq c(uv) x_{uv} \quad ,\forall \{u, v\} \in L.$$

The variable types are:

$$\begin{aligned} x_{uv} &\in \{0, 1\} && ,\forall \{u, v\} \in E \\ y_v &\in \{0, 1\} && ,\forall v \in V \\ z_v &\in \mathbb{R}_+ && ,\forall v \in V \\ \mathfrak{S}^{l,s}_p &\in \mathbb{R}_+ && ,\forall p \in P, l \in \mathfrak{L}_p, s \in V \\ \mathfrak{C}^p_s &\in \mathbb{R}_+ && ,\forall p \in P, s \in V \\ \mathcal{C}^t_s &\in \mathbb{R}_+ && ,\forall s, t \in V \\ f^s_{p,u} v &\in \mathbb{R}_+ && ,\forall p \in P, s, u, v \in V \\ f^s_{t,u} v &\in \mathbb{R}_+ && ,\forall s, t, u, v \in V \end{aligned}$$

Note that the problem admits also a fractional relaxation, where we have $x_{uv \in}[0, 1], \forall \{u, v\} \in E$ and $y_v \in [0, 1], \forall v \in V$, which is useful in heuristic described in this paper. It is in P, as there are $O(|V|^2|E|)$ variables and constraints in the formulation and there are polynomial-time algorithms to solve such linear programs. [Remik: Julio wanted to provide a reference. . . ]

## 3.4 Spanning tree heuristic

Since CPLEX was not able to solve the ILP model described in the last section for bigger instances, we describe here a polynomial-time heuristic to our

problem. For instance, for a random example with 150 cities and 300 links, CPLEX was not able to produce any feasible solution within two hours, while the proposed algorithm can give a good solution in two minutes.

Our heuristic is an iterative algorithm that, at each step $i \geq 0$, computes an optimal (fractional) solution $s_i$ for the relaxation of our model and fix value of some variables of the model corresponding to the usage of links and caches (i.e. the integral variables $x_{uv}$ and $y_v$). When we say that we *fix* a variable $x$ to a value $c \in \{0, 1\}$ at step $i$, we mean that we add a constraint $x = c$ to the model used to compute $s_j$, for all $j > i$.

At the first step 0, our heuristic computes a solution $s_0$ of relaxation of the model. Then, by setting the weight of each edge to be the value of its corresponding variable in $s_0$, a maximum spanning tree $T$ of the input network graph $G$ is computed and all the variables $x_{uv}$ of all the edges $uv \in E(T)$ are fixed to one.

After this initialization step, the heuristic solves, at each step $i > 0$, the relaxation of the model (which will already have several variables with fixed values) to get an optimal solution $s_i$. Then, if some other variables $x_{uv}$ or $y_v$ have value $v \in \{0, 1\}$ in the solution $s_i$, these variables are fixed to this value $v$. Finally, at least one most loaded device is fixed to be turned on. To speed up the process, the heuristic has a parameter $\mathcal{S}$. At each step $i$, we also fix $\mathcal{S}$ fraction of the highest value variables $x_{uv}$ or $y_v$ whose values $v$ are in $0 < v < 1$ to one. Once all the integer variables are fixed, the relaxation is solved one last time. This gives us a valid solution to the Integer Linear Program.

The heuristic has been implemented in Java and it can be downloaded as open source[1]. Note that we use CPLEX to solve the relaxations of the model at each step of the heuristic. The performance of this heuristic is analyzed in Section 5.

**On the complexity of the heuristic algorithm** The model we propose has a polynomial number of constraints on the size of the input. It is well-known that its relaxation can then be solved in polynomial time. The number of devices whose variables have to be set to 0 or 1 by our heuristic is $n$ caches (one at each node) plus $m$ edges. The first iteration puts $n-1$ edge variables to 1. When a variable is set to 0 or 1, it is not reconsidered during the algorithm execution. Hence, the number of relaxations solved, i.e. of steps of the heuristic, is bounded by $m + 1$.

Note that we indicated the number of iterations and the execution times in seconds for varied values of $\mathcal{S}$ in Section 5.1.2 and for networks of varying size in Section 5.5.

# 4 Instance generation

The Survivable fixed telecommunication Network Design (SND) Library contains a set of real network topologies, which we use as a base for most of our instances. In particular, we have decided to use three networks with considerably different sizes:

- *Atlanta* – $|V| = 15, |E| = 22$, unidentifiable cities

[1] `https://github.com/lrem/GreenContentDistribution`

- *Nobel-EU* – $|V| = 28, |E| = 41$, major European cities
- *Germany50* – $|V| = 50, |E| = 88$, major German cities

We added the position of the Content Distribution Network servers. Usually, Content Distribution Networks locate their servers in Internet Exchanges and major Points of Presence, to minimize the network distance to the end users. Locations of such points are publicly known. Thus, for topologies with clearly identifiable cities, we have ready sets of candidate locations for CDN servers. Otherwise (Atlanta network), we put them manually at cities which minimize the route lengths.

In order to obtain demands, we assume that in average populations among the cities behave similarly. Thus, the total amount of demands originating from a city is proportional to its population. If cities cannot be identified, the population is assumed to be distributed uniformly. There is roughly the same amount of demands toward each content per thousand people everywhere. Similar uniformity is assumed for demands between cities, bigger cities attract more demands than the small ones. A more formal description follows.

We used a population model to build the traffic matrices of the demands between cities. Then, we augmented matrices with the demands towards content providers. Obtaining exact figures about CDN market shares and operational details is out of scope of this study. Still, we explored the publicly available information, e.g. [6], to come up with a list of the major providers. Each of the networks is assigned a *popularity*, which is based on market share either claimed by the company or media. The number of servers is heterogeneous and we try to arrange it into distinct classes in regard to popularity/server capacity proportion, i.e. there can be networks with many small servers, or few strong ones.

Table 1 exemplifies CDN specification used in the *germany50* network. Server capacity means what part of total demand towards a network can be satisfied by the infrastructure in a single location. For example, just two servers with capacity 0.5 can satisfy all demands towards CDN4.

The process of obtaining the instance files has been automated by a set of Python scripts, which we make freely available[2]. First, one needs to provide a list of English names of the cities in the topology. Population data is pulled automatically from Wikipedia.

Then, having the list of cities and their populations, user specifies the total amount of demands $d$ originating from the most populous city $m$ and percentage of traffic directed towards CDN $o$. It is more convenient to specify demands in relation to link bandwidth $l$ in the form of demand ratio $r$, such that $d = {}^{l}/_{r}$. The remaining $100 - o$ percentage is directed towards other cities. Total population $t$ of all cities is computed. For a city with population $a$ its aggregate demand towards another city with population $b$ is equal to:

$$d \cdot \frac{a}{m} \cdot \frac{b}{t-a} \cdot (100 - o)\%$$

Another part of input is CDN configuration. It is comprised of a list of entries containing name, popularity, server capacity and server locations. Total popularity $P$ of all CDNs is computed. For a city with population $a$ its aggregate

[2] `http://www-sop.inria.fr/members/Remigiusz.Modrzejewski/software.html`

demand towards an CDN with popularity $p$ is equal to:

$$d \cdot \frac{a}{m} \cdot \frac{p}{P} \cdot o\%$$

# 5 Results

In this section, we first validate our heuristic. We show that it is able to find good solutions for small and medium-sized networks by comparing with optimal solutions given by the model. We implemented the formulation on the ILP solver CPLEX version 12. We then show that the heuristic is fast and is able to quickly find solutions for large networks for which CPLEX was not able to find any feasible solutuion.

Then, we investigate the potential energy savings of our solution on realistic networks. We exhibit the impact of the cache, CDN and network parameters, such as cache size, number of CDN servers, or route lengths. Note that, as described in Section 3.3, energy consumption is given in normalized energy units equal to energy used by one link, denoted $l_c$.

When directly solving the ILP, by default we set a limit on the execution time to five minutes per instance.

## 5.1 Validation of the Heuristic Algorithm

In order to validate the Spanning Tree Heuristic, we compare its performance to solving the integer model directly with CPLEX. First we show the differences in several examples. Then, we focus on showing the impact of the parameter $\mathcal{S}$, which governs the speed/quality trade-off, on three chosen examples.

### 5.1.1 Comparison of the heuristic and the ILP

Table 2 displays performance comparison between Spanning Tree Heuristic and solving the ILP directly by CPLEX version 12. It compares the values of objective function and wall clock time taken by the computation on an Intel i7-powered computer. The $\Delta$ columns mean, respectively, by what percentage the solution found by the heuristic is worse and how much time is saved by using it. The heuristic parameter $\mathcal{S}$ is set to 0.2. It is discussed in the next section.

First, notice that for very small networks it is feasible to solve the ILP optimally. This is exemplified by the 15-node Atlanta network. The optimal solution is found within two seconds. Interestingly, the running time grows for lower traffic. This is entirely because rising the lower bound, which has to be equal to the objective value to state the solution is optimal, becomes much harder. Solutions given by the heuristic are close to the optimum, while the time needed to find them is much shorter. Still, for networks of this size, we would strongly recommend solving the model directly.

For networks up to 30 nodes it is still feasible to find optimal solutions. However, the cost of obtaining the solution is rather high, while closing the gap to the lower bound becomes impractical for low traffic. Thus, we limited the CPLEX execution time to half an hour. On the other hand, Spanning Tree

| Topology | $\|V\|$ | Total energy[$l_c$] | | $\Delta$ | Computation time[s] | | $\Delta$ |
|---|---|---|---|---|---|---|---|
| | | Model | Heuristic | | Model | Heuristic | |
| Atlanta (high traffic) | 15 | 18.8$\star$ | 19.0 | 1% | 1.5 | 0.6 | 60% |
| Atlanta (medium traffic) | 15 | 16.6$\star$ | 18.6 | 12% | 5.2 | 0.6 | 88% |
| Atlanta (low traffic) | 15 | 14.1$\star$ | 14.4 | 2% | 34.4 | 0.6 | 98% |
| Nobel-EU (high traffic) | 28 | 31.4$\star$ | 35.1 | 12% | 1075 | 1.8 | 99.8% |
| Nobel-EU (medium traffic) | 28 | 28.4 | 32.2 | 13% | 1800 | 1.3 | 99.9% |
| Nobel-EU (low traffic) | 28 | 27.9 | 30.2 | 8% | 1800 | 1.1 | 99.9% |
| Germany50 (high traffic) | 50 | 69.7 | 69.0 | -1% | 300 | 8.5 | 97% |
| Germany50 (medium traffic) | 50 | 54.2 | 61.6 | 14% | 300 | 5.0 | 98% |
| Germany50 (low traffic) | 50 | 50.0 | 56.2 | 12% | 300 | 2.9 | 99% |
| Random | 150 | No solution | 203.7 | — | 7200 | 127.8 | — |

Table 2: Comparison of results given by the Spanning Tree Heuristic (labelled *Heuristic*) and by solving the model directly with CPLEX (labelled *Model*). The $\star$ symbol denotes optimal solutions.

HEURISTIC provides its solutions in under two seconds. Again, by choosing the heuristic, we accept only a slight increase in consumed energy. Precisely, to obtain a solution within 12% of the optimum, we save 99.8% of the computation time.

In medium-sized networks, such as Germany50, finding exact solution becomes impractical. Thus, we set a limit of 5 minutes to obtain near-optimal results. This allows SPANNING TREE HEURISTIC to obtain a slightly better solution than the ILP, while taking only 3% of the running time, in the case of high traffic. In the other cases it is still not far quality-wise, while taking negligible time.

Finally, we take a big random instance. The topology is a two-connected Erdös-Renyi graph, with 150 nodes, an average degree of four and one CDN with fifteen servers. Each city issues demands only to seven other cities. The overall traffic level is medium (demand ratio 4.0), as these kind of instances are prone to bottlenecks, which could render higher traffic levels unrouteable. It is infeasible to directly obtain any integer solution of the model. After two hours CPLEX was not able to propose even a trivial solution (e.g. turning on all the devices). SPANNING TREE HEURISTIC, in just above two minutes, gives a solution that is 35.8% over the trivial lower bound of a minimal connected network.

To conclude, we say that the SPANNING TREE HEURISTIC is clearly the better choice for big networks. For small to medium-sized ones, its results are always reasonably good, while its running time is very short. Therefore, it is a viable choice whenever the computation time is an issue.

### 5.1.2 Speed/quality trade-off of the Spanning Tree Heuristic

As stated in Section 3.4, the parameter $\mathcal{S}$ governs an execution speed vs quality of solution trade-off for the SPANNING TREE HEURISTIC. We investigate its influence in this section.

First, recall that $\mathcal{S}$ determines the fraction of undecided variables to be fixed to an integer value within an iteration. Each iteration at least one variable is set to one, so setting $\mathcal{S}$ to zero means turning on devices one by one. It is easy to see how increasing $\mathcal{S}$ reduces the number of iterations. To comprehend how it can decrease the quality of obtained solution, imagine a simple example, that represents a fragment of an instance. Take two cities with two disjoint paths and one demand between them. Let the value of that demand be equal to bandwidth of a link. One valid solution of the relaxation can be splitting the demand in half and routing both halves along both paths. The optimal integer solution for this case is all the flow going through one route, the links of the other turned off. If $\mathcal{S} = 0$, then after the first step one link will be turned on. The only possible solution of the relaxation will route all the traffic through the path containing this link. Thus, the solution found by SPANNING TREE HEURISTIC will be optimal. However, if $\mathcal{S} > 0$ and two links are turned on in the first step, then it is possible the two links will be on different paths. Thus, the integer solution will have some unnecessary links turned on. In the extreme case of $\mathcal{S} = 1$ all devices will always be turned on.

Figure 2 shows the impact on three examples. In all cases, the x axis determines the value of parameter $\mathcal{S}$. The left column plots the value of the objective function in integer solutions. The right column shows computational

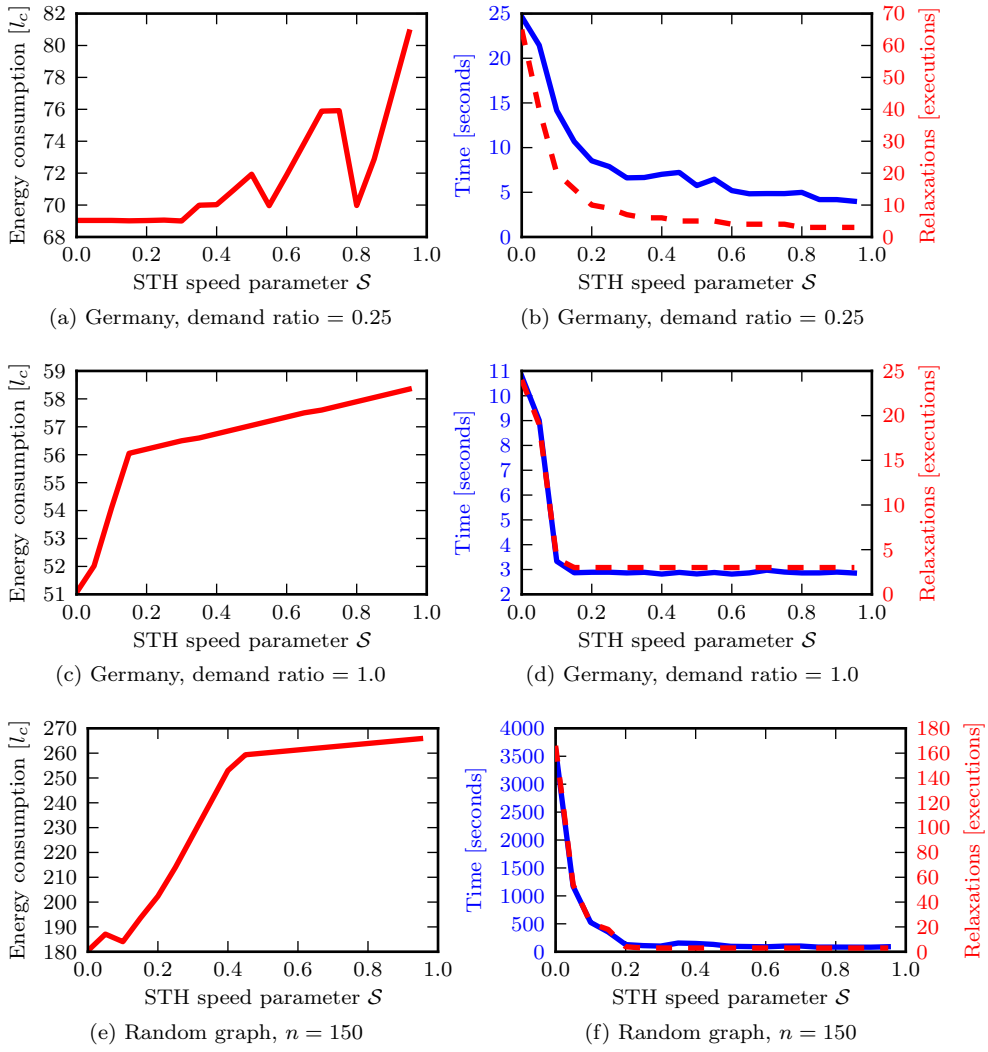
costs, both in terms of wall clock time in seconds (solid blue lines) and number of relaxations solved (dashed red lines).

First, look into an instance based on maximum traffic sustainable in the *Germany50* network. Solutions obtained are displayed on plot 2a. Recall that the value found by a solver for this instance was 69.7 energy units. Taking between 24 and 8 seconds, Spanning Tree Heuristic with $\mathcal{S} \leq 0.3$ obtains solutions with 69.0 units. This means it is in this case both faster by an order of magnitude and gives a marginally better solution. Note that even at $\mathcal{S} = 1$ not all devices are turned on. This is because, after freezing the spanning tree, some devices get turned off before all the undecided ones are turned on. Looking at plot 2b, we see that the number of relaxations solved and the running time are falling drastically for $\mathcal{S} \leq 0.2$. Then, they decrease more slowly, with 6 seconds at $\mathcal{S} = 0.5$ and 4 seconds at $\mathcal{S} = 1.0$.

Second, we assign to the same network a small load, that sill does not allow for routing on a spanning tree (which would be a trivial case for the heuristic). With model given directly to a solver, we have obtained in 5 minutes a solution with value 50. Plot 2c shows that the best solution found by Spanning Tree Heuristic is still one unit worse and can deteriorate by almost eight further units. On the other hand, the maximum time taken by Spanning Tree Heuristic is 10.8 seconds. For $\mathcal{S} = 0.1$ it is already 3.3 seconds, reaching 2.8 at $\mathcal{S} = 1$.

Finally, we present results for the same random graph as in the previous section. Looking at plot 2e, we see that there is significant but steady increase in energy consumption until $\mathcal{S} = 0.4$. At that point, the value objective function is nearly saturating, at 1.44 times the value for $\mathcal{S} = 0$. On the other hand, plot 2f shows that there is a sharp decrease in computational cost until $\mathcal{S} = 0.2$. As the objective value at that point is not far from the best known value, we deduce that this is a reasonable value of $\mathcal{S}$ for fast solving of big instances. Note that when solving the model directly, CPLEX 12 is not able to produce any integer solution within reasonable timespan of two hours. The only lower bound on the objective value we know comes from the fact, that the network needs to be connected. Thus, there are at least 149 links needed. This means that the heuristic, with $\mathcal{S} = 0$, is at most within 20.8% from the optimal solution.

As we have seen in the above examples, the Spanning Tree Heuristic is a good alternative to solving the model directly for big networks. Furthermore, even when it is possible to obtain a solution directly from the model, it may be possible that the heuristic provides a solution of similar quality in a shorter time.




(a) Germany, demand ratio = 0.25

(b) Germany, demand ratio = 0.25

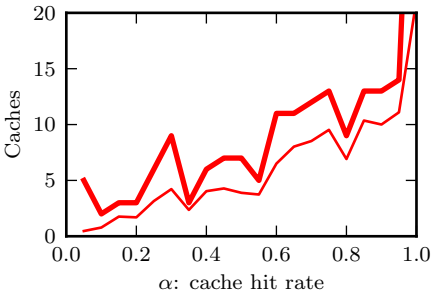
(c) Germany, demand ratio = 1.0

(d) Germany, demand ratio = 1.0

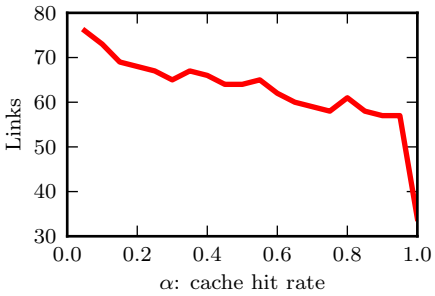
(e) Random graph, $n = 150$

(f) Random graph, $n = 150$

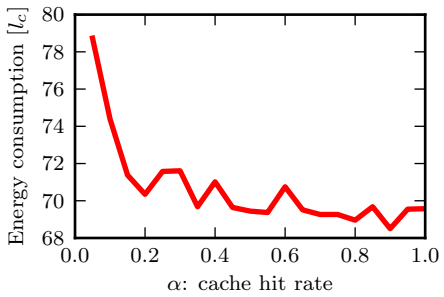
Figure 2: Impact of the parameter $\mathcal{S}$, left column plots the energy consumption of obtained designs, while right column plots the computational cost.




(a) Cache usages; thick line marks the number of caches turned on, thin lines below their total load.




(b) Link usages.




(c) Total energy consumption.

Figure 3: Results in function of parameter $\alpha$ for Germany.

## 5.2 Impact of cache parameters

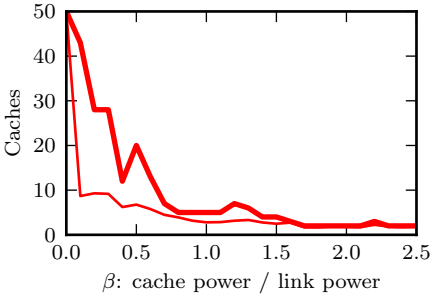
In this section, we exemplify the impact of parameters of the cache. We look into how the obtained network designs differ on changing values of the cache hit rate $\alpha$ and of the cache power usage $\beta$.

First, we look at the effects of changing the parameter $\alpha$, shown in Figure 3. Recall, that it limits what part of any single demand can be served from a cache. Increasing the significance of caches results in more being used and energy being saved. However, note that once about 15% of traffic can be cached, further gains are highly diminished. This, according to Section 3.1, is equivalent to about 800GB or just 100GB depending on the cache algorithm used.

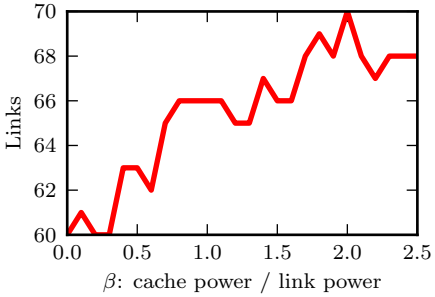
Cache bandwidth, plotted in Figure 6, has a similar effect as $\alpha$. It is additionally reinforced by the fact that, with the same amount of provided data, it renders load proportionally smaller. Thus, the difference has little practical meaning.

Finally, we look at the effects of changing maximum cache power usage, shown on Figure 4. As seen on plot 4a, this has a much stronger influence on the number of caches turned on. In fact, this is the scenario in which the minimum cache usage was seen – only two caches. As expected, both total energy consumption and number of used links, depicted on plots 4c and plots 4b, raise along $\beta$.

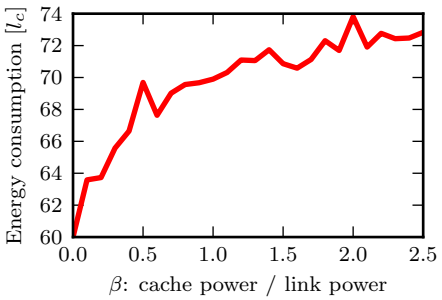
Figure 5 compares energy consumption of networks without caches and with caches of different parameters. Traffic is set to the highest level which is feasible




(a) Cache usages; thick line marks the number of caches turned on, thin lines below their total load.

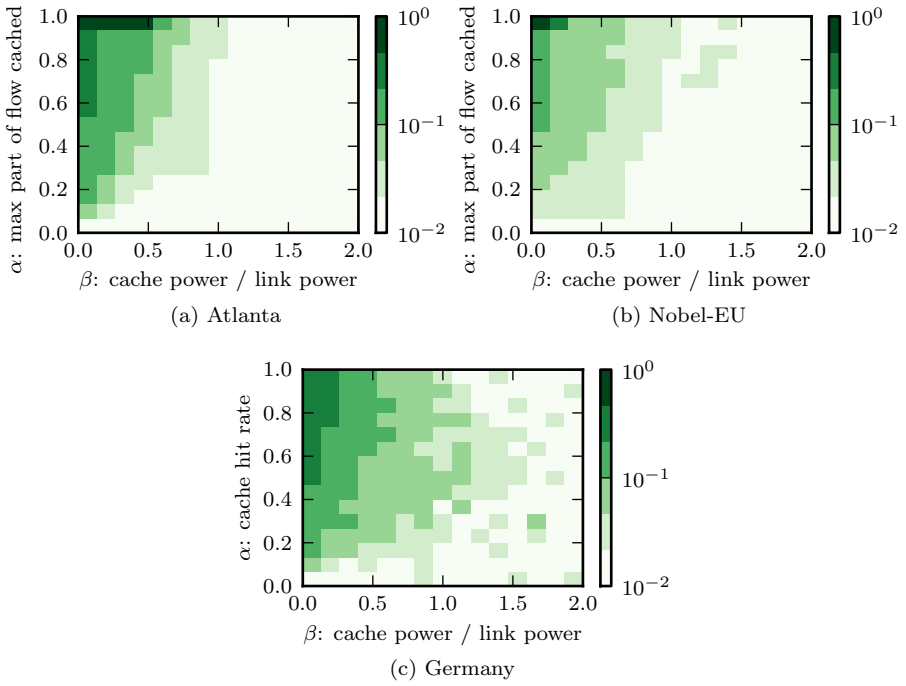



(b) Link usages.




(c) Total energy consumption.

Figure 4: Results in function of parameter $\beta$.

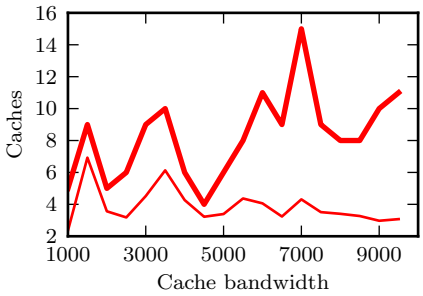



(a) Atlanta

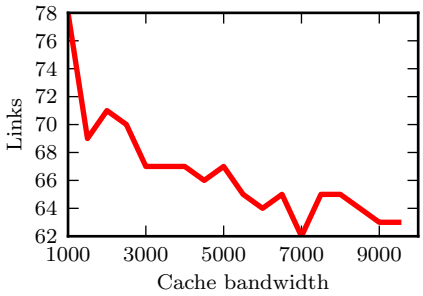
(b) Nobel-EU

(c) Germany

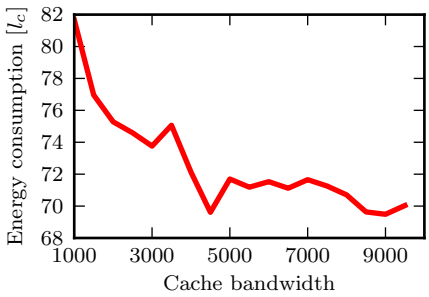
Figure 5: Part of energy consumption saved by introducing caches with various parameters.




(a) Cache usages; thick line marks the number of caches turned on, thin lines below their total load.




(b) Link usages.




(c) Total energy consumption.

Figure 6: Results in function of cache bandwidth. Note that link bandwidth = 10000.

without caches. The difference, divided by consumption in the network without caches, is shown as a color map. It can be seen as a two-dimensional product of plots 3 and 4.

We can see from them a similar impact of the parameters among different networks. In case of both smaller instances, we see practically no gains with caches that use no less energy than a link, no matter how much traffic can it save. Unsurprisingly, in all the instances, potential savings resemble the distance from the point $(\alpha = 1, \beta = 0)$, with an especially high effect when the caches consume no energy.

## 5.3 Impact of CDN parameters

Then, we investigate the impact of the cooperation with CDN. Figure 7 shows the evolution of energy consumption in function of what part of all demands are directed towards CDN networks. The demand ratio for this plot is set to 0.33, to make routing without caches nor cooperating content providers feasible. Results both with and without caches are compared. First, we discuss results for Germany. As we can see, introducing cooperating content providers to a network without caches is highly beneficial. In the extreme case when all traffic would be served by CDNs, energy consumption would decrease by 27.4%. At today's claimed values this number is still 16.4%. Then, introducing caches to a network without CDN gives 16.7% savings. There remain 8.0% savings at

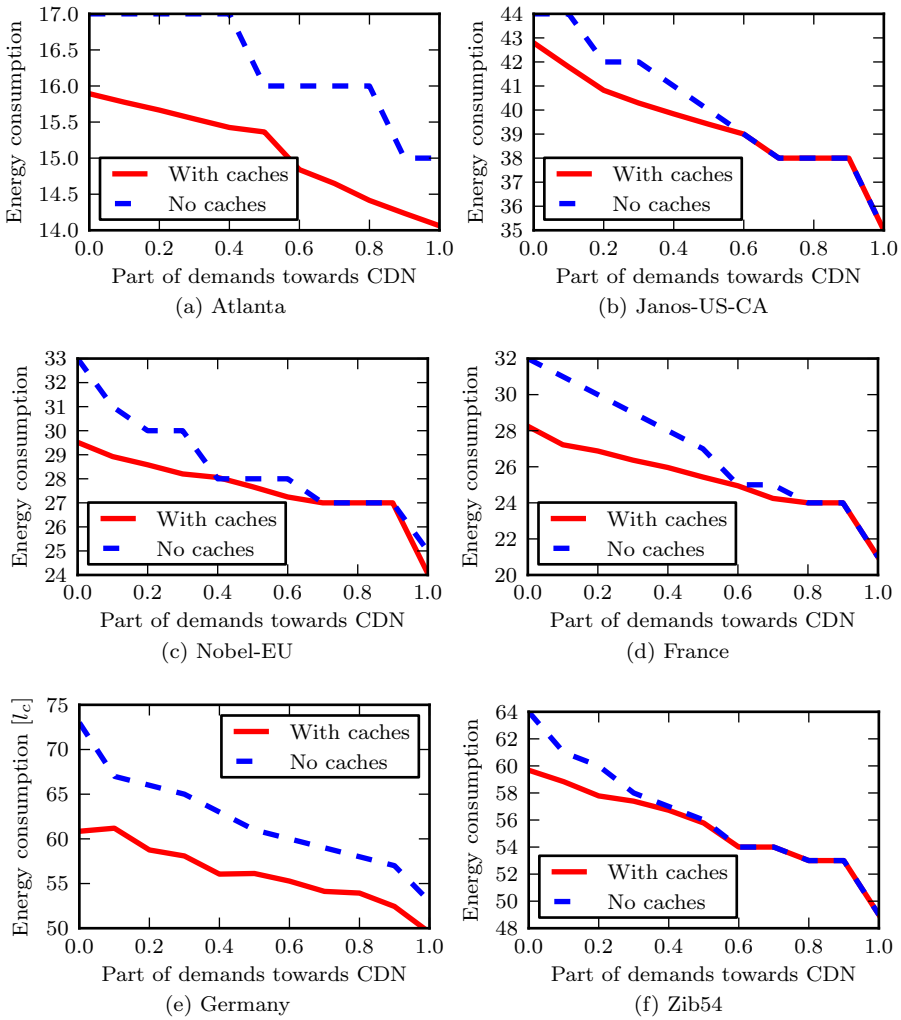



Figure 7: Impact of caches and CDN cooperation on energy consumption

.

| Instance | Cache | CDN 0.5 | CDN 1.0 | Cache + CDN 0.5 |
|---|---|---|---|---|
| Atlanta | 6.5% | 5.9 % | 11.8% | 9.6% |
| Janos-US-CA | 2.7% | 9.0% | 20.5% | 9.45% |
| Nobel-EU | 10.6% | 15.2% | 24.2% | 16.2% |
| France | 11.8% | 15.6% | 34.4% | 20.5% |
| Germany | 16.7% | 16.4% | 27.4% | 23.1% |
| Zib54 | 6.74% | 12.5% | 23.4% | 12.9% |

Table 3: Savings in different scenarios for different networks.

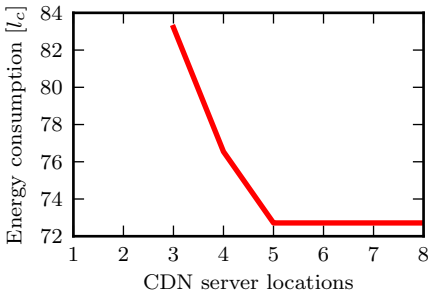



Figure 8: Impact of the number of CDN server locations in Germany.

today's CDN popularity. What may be a bit surprising, there are still 6.6% savings by introducing caches when 100% of traffic is served by the Content Delivery Networks. Finally, comparing network without CDN nor caches, to network with 50% of traffic served by CDN and with enabled caches, we save 23.12% of energy. Table 3 shows the numbers, for cases described above, for all the networks. Savings are always compared to the case without caches nor CDN.

Figure 8 investigates how many location choices are needed to achieve good savings. In this scenario, for the sake of clarity, there is only one CDN. Its servers are potentially located in: Berlin, Frankfurt, Muenchen, Hamburg, Dortmund, Stuttgart, Leipzig and Aachen. In each data point, only the first $n$ servers from this list are enabled. Each server is able to provide all the demands alone, 50% of all traffic is served by the CDN. It is infeasible to route with less than 3 locations. As we can see, increasing the number of possible choices from 3 to 5 yields around 13% of energy savings. Further increases have little effect. Thus, in this network of 50 cities, it is optimal to have 5 server locations.

## 5.4 Impact of traffic level

In this section, we look into the potential reduction of energy consumption of the networks in our model, both with and without usage of the caches, exploiting the variance in network traffic over time. The parameters used throughout this section are: $\alpha = 0.35$, $\beta = 0.1$ and cache bandwidth is half of a link.

Figure 9 shows energy consumption in function of demand ratio, that is the inverse of traffic level. As we can see, in all the networks, enabling caches makes routing feasible under much higher loads than before. For example in the case of Germany, we can accommodate an increase in demands by one third. Then, as traffic decreases, we can save energy by turning off some devices. The right column of Table 4 states relative difference between energy consumption of a network under highest possible load and half of that load, with caches enabled.

For a range of demand values, it is feasible to route without caches, but at a higher total energy cost. Note that half of maximum sustainable load is in all cases within this range. The left column of Table 4 shows the highest difference of power consumption accommodating the same traffic with and without caches.

As can be seen, there is a point after which there are no additional savings with falling traffic. This is when the routing is feasible on a spanning tree, using no caches. Turning off any additional device would disconnect the network.

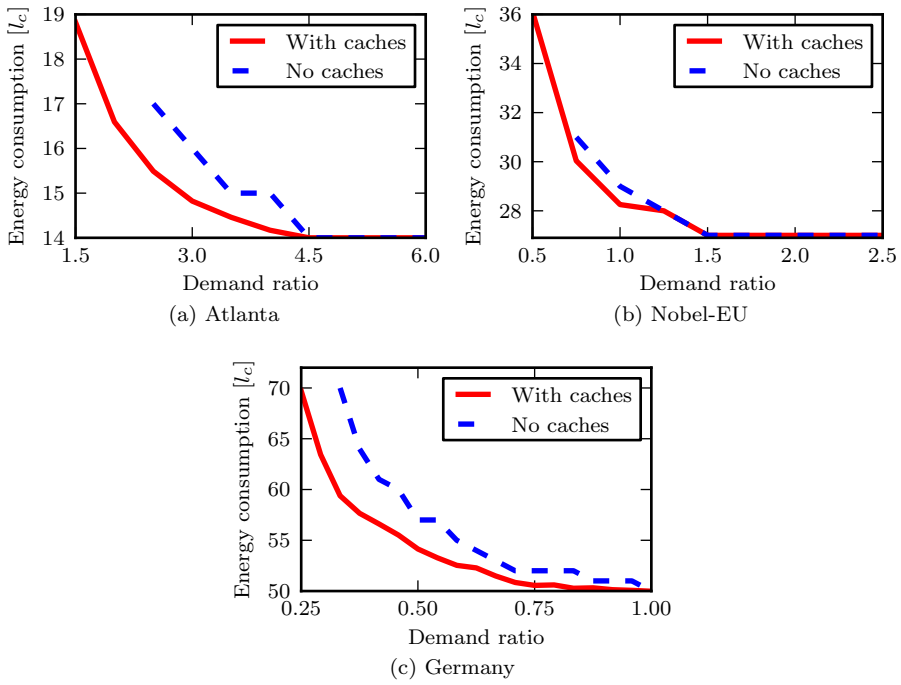



Figure 9: Comparison of energy consumption with and without caches in the model.

| Network | Nodes count | Maximum energy saved due to caches | Total energy savings (load=50%) |
|---|---|---|---|
| Atlanta | 15 | 8.9% | 21.3% |
| Nobel-EU | 28 | 3.2% | 21.7% |
| Germany | 50 | 16.7% | 22.3% |

Table 4: Potential energy savings

What is interesting is the fact that caches have a much higher effect in the *germany50* than the smaller instances. We attribute that to longer routes, which mean higher energy cost to transfer the data through the network. This effect is investigated in Section 5.5.

Figure 10 plots the results into kinds of devices being on. As expected, turning on some caches allows us to turn off some links.

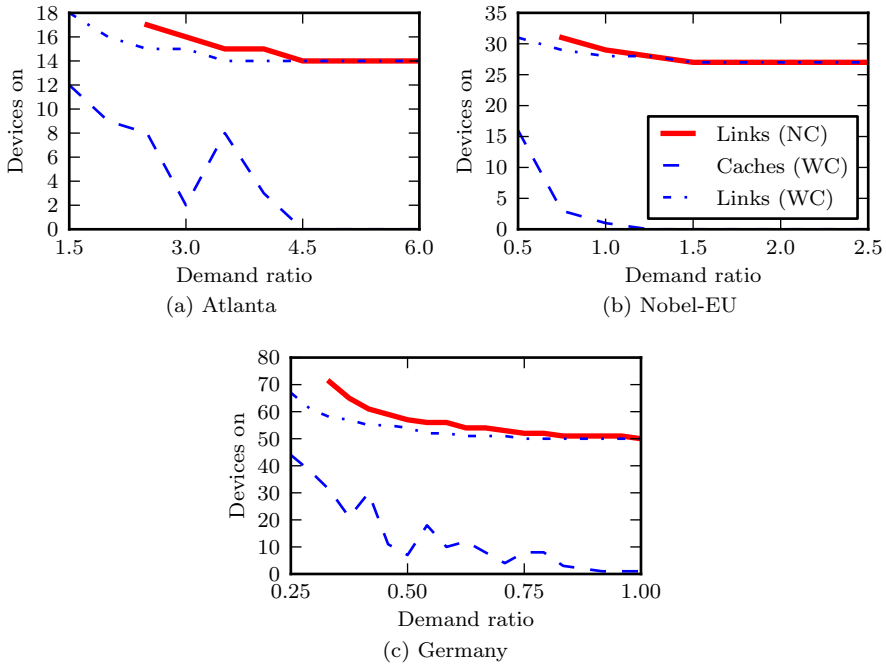



Figure 10: Breakdown of devices used in each case.

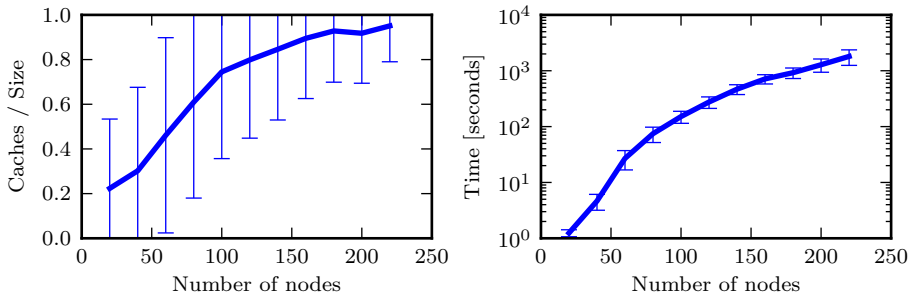
## 5.5 Impact of network size

We have seen varying usage of caches in the studied networks. An explanation for that is the difference of route lengths in the diverse networks. Energy is saved by serving from a cache close to the user. Savings depend on how long would be the route traversed by the data, if it was served from content provider. A longer route yields higher reductions. However, in the biggest network we used, the *germany50*, the average route length is only 4. Furthermore, when looking at a distance traveled by an average bit of data, this length is only 2.6. We claim that in bigger networks we could see higher utility of caches.

To estimate the impact of route length, we look into results on Erdős-Rényi graphs. Recall that in these graphs, the route lengths grow logarithmically in respect to the graph size. As we need many big networks to demonstrate the effect, obtaining integer solutions directly from a solver would be impractical. Therefore, the results presented are computed using the Spanning Tree Heuristic.

Figure 11a shows the number of caches used divided by the number of cities in two-connected Erdős-Rényi graphs of increasing sizes. The average degree of each graph is 4, each city emits 7 demands to random other cities, cache parameters are $\alpha = 0.35$, $\beta = 0.1$ and $\gamma = 0.5$. Each data point is an average over at least two thousand instances, error bars represent standard deviation.

As we can see, with no other parameters changing, usage of caches clearly grows with increasing network sizes. In a network of size 20, having average route length around 2.3, average number of caches on is only 4.47 (22.3%), while in networks of size 220, of average route length around 4.2, there are on




(a) Cache usage divided by network size (b) Execution time (note the log scale)

Figure 11: Spanning Tree Heuristic on Erdős-Rényi graphs.

average 209.2 (95.1%) caches turned on. Caches see an average usage over 50% for networks of size at least 80, where the average route length is only around 3.4. This size can correspond to small networks comprised of both core and metropolitan parts, or just big core networks.

Figure 11b displays the computation times. The value of $\mathcal{S}$ is 0.2. The execution time grows quickly. This is not due to the number of heuristic iterations, between 20 and 220 nodes the number of relaxations solved only doubles. However, at $n = 220$ a single relaxation takes 6 minutes on average. Thus, the time needed to find the fractional routing is the critical part of the computational cost.

# 6 Conclusions and further research

In this work, we addressed to the problem of energy saving in backbone networks. To the best of our knowledge, this is the first work to consider that impact of in-router caches, along with assigning servers of Content Delivery Networks to demands, in an energy-efficient routing.

We have proposed a new Integer Linear Programming model for saving energy in backbone networks by disabling links and caches of this network and a polynomial-time heuristic for this problem. We compared the performance of the solutions proposed by our heuristic against those found by CPLEX. In small to medium-sized instances the solutions given by the heuristic are close to that of the integer program. Being faster by orders of magnitude, it allows to find good solutions for bigger networks, where CPLEX was not able to produce any feasible solution in hours.

We studied instances based on real network topologies taken from SNDLib. The total energy savings found oscillate around 20% for realistic parameters. Part of energy saved solely due to introduction of caches is up to 16% in our instances.

As a future work, the model could be extended to enable the usage of a single cache to satisfy the demands of multiple cities, i.e. to let a cache satisfy demands to different routers and not only to its own router. The energy savings will probably grow in this model, however it would be interesting to study how this solution could be deployed.

One could also look at different network architectures. This work considered

only the backbone. A next step could be introducing access networks, leading to larger instances. As the savings due to caches grow with network size, they should be substantially higher in this case. This could also motivate study of new mechanisms, e.g. layered caching.

# References

[1] M. Webb, "Smart 2020: Enabling the low carbon economy in the information age," *The Climate Group London*, 2008.

[2] M. Zhang, C. Yi, B. Liu, and B. Zhang, "Greente: Power-aware traffic engineering," in *ICNP'10: IEEE International Conference on Network Protocols*, 2010.

[3] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing isp network energy cost: Formulation and solutions," *IEEE/ACM Transactions on Networking*, vol. 20, pp. 463–476, April 2011.

[4] A. P. Bianzino, L. Chiaraviglio, M. Mellia, and J.-L. Rougier, "Grida: Green distributed algorithm for energy-efficient ip backbone networks," *Computer Networks*, vol. 56, no. 14, pp. 3219–3232, August 2012.

[5] F. Giroire, D. Mazauric, and J. Moulierac, "Energy efficient routing by switching-off network interfaces," *Energy-Aware Systems and Networking for Sustainable Initiatives*, p. 207, 2012.

[6] Akamai. [Online]. Available: http://www.akamai.com/html/riverbed/akamai_internet.html

[7] L. Chiaraviglio and I. Matta, "Greencoop: Cooperative green routing with energy-efficient servers," in *First International Conference on Energy-Efficient Computing and Networking (e-Energy)*, April 2010.

[8] ——, "An energy-aware distributed approach for content and network management," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2011, pp. 337 –342.

[9] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," in *IEEE INFOCOM*, April 2009, pp. 2631 –2635.

[10] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose, "The cache-and-forward network architecture for efficient mobile content delivery services in the future internet," in *Innovations in NGN: Future Network and Services*, May 2008, pp. 367 –374.

[11] C. Dannewitz, "Netinf: An information-centric design for the future internet," in *Proceedings of 3rd GI/ITG KuVS Workshop on The Future Internet*, May 2009.

[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of ACM CoNEXT*, December 2009.

[13] CPLEX. [Online]. Available: http://www.ibm.com/software/integration/optimization/cplex-optimization-studio/

[14] SNDLib. [Online]. Available: http://sndlib.zib.de

[15] P. Erdős and A. Rényi, "On the evolution of random graphs," in *Publication of The Mathematical Institute of The Hungarian Academy of Sciences*, 1960, pp. 17–61.

[16] V. Mathew, R. K. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in *Proceedings IEEE INFOCOM*, March 2012, pp. 954 –962.

[17] C. Ge, N. Wang, and Z. Sun, "Optimizing server power consumption in cross-domain content distribution infrastructures," in *IEEE International Conference on Communications Workshops (ICC)*, June 2012.

[18] Z. Li, G. Simon, and A. Gravey, "Caching policies for in-network caching," in *21st International Conference on Computer Communications and Networks (ICCCN)*, August 2012, pp. 1 –7.

[19] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking (ICN)*. ACM, 2012, pp. 55–60.

[20] K. Guan, G. Atkinson, D. C. Kilper, and E. Gulsen, "On the energy efficiency of content delivery architectures," in *IEEE International Conference on Communications Workshops (ICC)*, june 2011, pp. 1–6.

[21] Y. Song, M. Liu, and Y. Wang, "Power-aware traffic engineering with named data networking," in *Seventh International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, Dec. 2011, pp. 289–296.

[22] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson, "In-network caching effect on optimal energy consumption in content-centric networking," in *IEEE International Conference on Communications Workshops (ICC)*, June 2012.

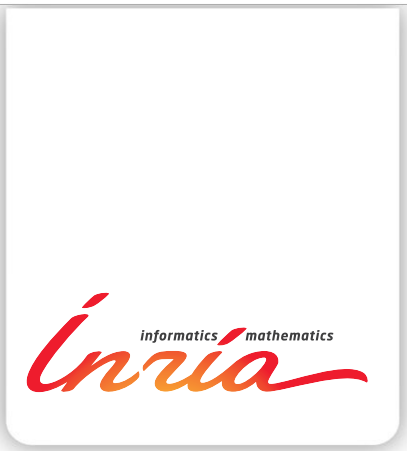
[23] U. Mandal, C. Lange, A. Gladisch, P. Chowdhury, and B. Mukherjee, "Energy-efficient content distribution over telecom network infrastructure," in *Transparent Optical Networks (ICTON), 2011 13th International Conference on*. IEEE, 2011, pp. 1–4.

[24] C. Jayasundara, A. Nirmalathas, E. Wong, and C. A. Chan, "Energy efficient content distribution for vod services," in *Optical Fiber Communication Conference*. Optical Society of America, 2011.

[25] G. Haßlinger and O. Hohlfeld, "Efficiency of caches for content distribution on the internet," in *Teletraffic Congress (ITC), 2010 22nd International*. IEEE, 2010, pp. 1–8.

[26] W. Van Heddeghem and F. Idzikowski, "Equipment power consumption in optical multilayer networks-source data," Technical Report IBCN-12-001-01 (January 2012), available at http://powerlib. intec. ugent. be, Tech. Rep., 2012.

[27] OCZ Technology Group. [Online]. Available: http://www.ocztechnology.com/ocz-revodrive-3-x2-pci-express-ssd.html




**RESEARCH CENTRE**
**SOPHIA ANTIPOLIS – MÉDITERRANÉE**

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803