Energy Efficient Content Distribution

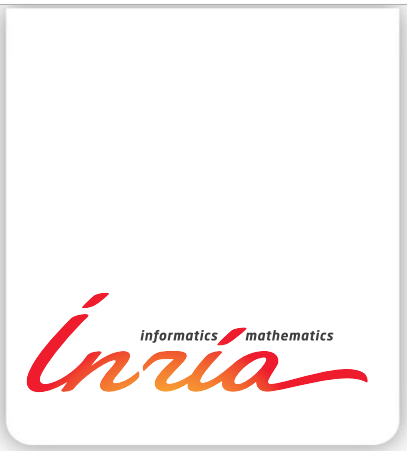
Julio Araujo, Frédéric Giroire, Yaning Y.L. Liu, Modrzejewski Remigiusz, Joanna Moulierac

Energy Efficient Content Distribution

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

RESEARCH
REPORT
N° 8091
Octobre 2012
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 1 — initial version Octobre 2012 — revised version Octobre 2012 — 24 pages

Abstract: To optimize energy efficiency in 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 CDN cooperation on an energy-efficient routing. We formulate this problem as Energy Efficient Content Distribution. 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. Experiment 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 a total up to 23% of power in the backbone, while 16% can be gained solely thanks to 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 proposé 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. 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, jusqu'à 23% de l'énergie du backbone peut être sauvée, dont 16% du gain est dû 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 there 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 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 use energy and 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 well choosing content provider servers, for each demand. We implemented this formulation on the ILP solver CPLEX version 12 and made experiments on real, taken from SNDlib [13], and random, Erdős-Rényi [14], 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 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. Experiment results show potential energy savings of about 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.

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 a total 23% 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. In the appendix, we show two heuristics.

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 [15] 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 [16] 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, 17, 18]. Similar to our work, these works also use in-network caches, however they do not consider energy savings.

Energy efficiency in content-oriented architectures [19–21] with an in-network caching has been studied recently. In [19], authors analyze the energy benefit of using CCN by comparison to CDN networks. A further work [21] considering the impact of different memory technologies on energy consumption is studied. 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 [20] 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.

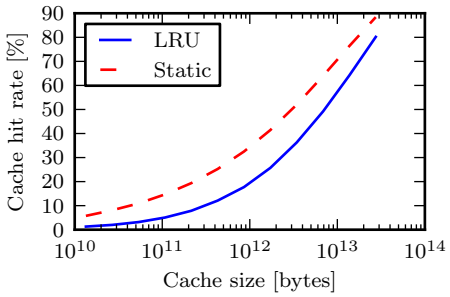


Figure 1: Cache hit ratio for YouTube trace, assuming average video size 100MB, following the results in [22].

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.

3.1 Parameters

For in-network caches, it is still an open question: if and how they will be deployed. Therefore, we try to avoid making specific assumptions about the details. Once the question is answered, the model can be updated to answer any possible specific concerns. 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 [22], 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 cache 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: α – 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 $\alpha \in [0.2, 0.35]$.

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 [23], 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 [24].

Again, as the practical values in the time of implementation are hard to predict, we make this ratio another parameter of the model: β – power consumption of a cache divided by 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 each content provider p has a capacity $C(\mathfrak{s}^l_p)$, for every $p \in P, v \in \mathfrak{L}_p$. 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 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 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 β 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 γ of β and its power consumption grows linearly with load to reach β 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 α, 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:

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

Then, we record the load of the cache:

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

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

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

Each possible source $s \in V$ demands from each provider $p \in P$ an amount of data flow $\mathfrak{D}_s^p \geq 0$. 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:

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

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

$$\sum_{v \in N_u} f_{v,u}^s - \sum_{z \in N_u} f_{u,z}^s =$$

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

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

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

	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.

The variable types are:

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

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 heuristics 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.

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:

- *Atlanta* – $|V| = 15, |E| = 22$, unidentifiable cities
- *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, 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 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[1]. 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:

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

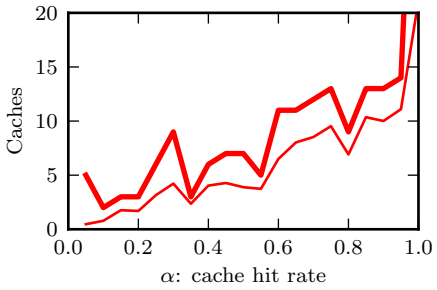
Another part of input is CDN configuration. It is comprised of a list of entries containing name, popularity, server capacity and server locations. Total popularity P of all CDNs is computed. For a city with population a its aggregate demand towards an CDN with popularity p is equal to:

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

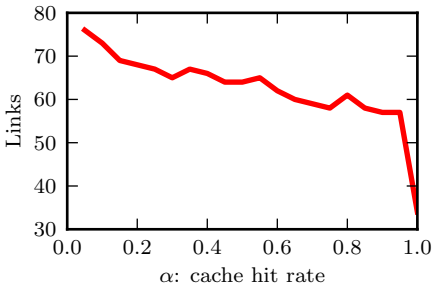
5 Empirical results

We implemented the formulation on the ILP solver CPLEX version 12. For large networks, we set a limit on the execution time. This section summarizes the empirical results obtained by solving our model, explains the impact of the parameters and shows the potential energy savings.

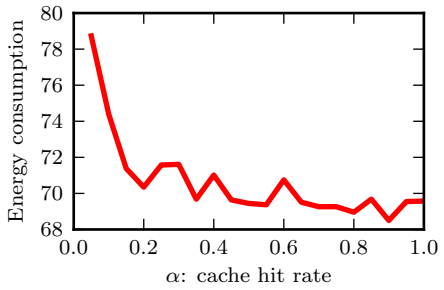
[1] `http://www-sop.inria.fr/members/Remigiusz.Modrzejewski/software.html`



(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 2: Results in function of parameter α for Germany.

5.1 Cache parameters

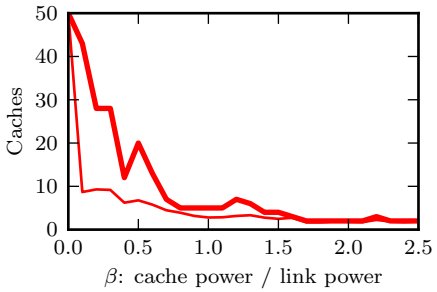
In this section, we exemplify the impact of parameters of the cache. We look into how the obtained network designs differ on changing values of the cache hit rate α and of the cache power usage β.

First, we look at the effects of changing the parameter α, shown in Figure 2. 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.

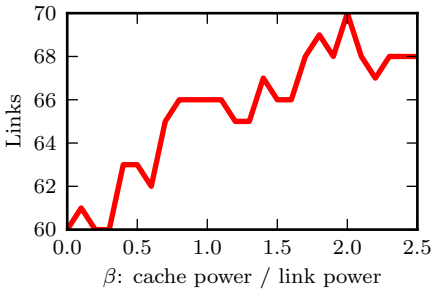
Cache bandwidth, plotted in Figure 5, has a similar effect as α. It is additionally reinforced by the fact that, with the same amount of provided data, it renders load proportionally smaller. Thus, the difference has little practical meaning.

Finally, we look at the effects of changing maximum cache power usage, shown on Figure 3. As seen on plot 3a, 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 3c and plots 3b, raise along β.

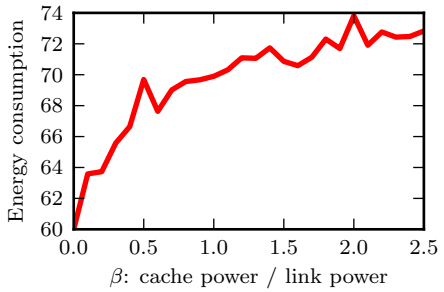
Figure 4 compares energy consumption of networks without caches and with caches of different parameters. Traffic is set to the highest level which is feasible



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

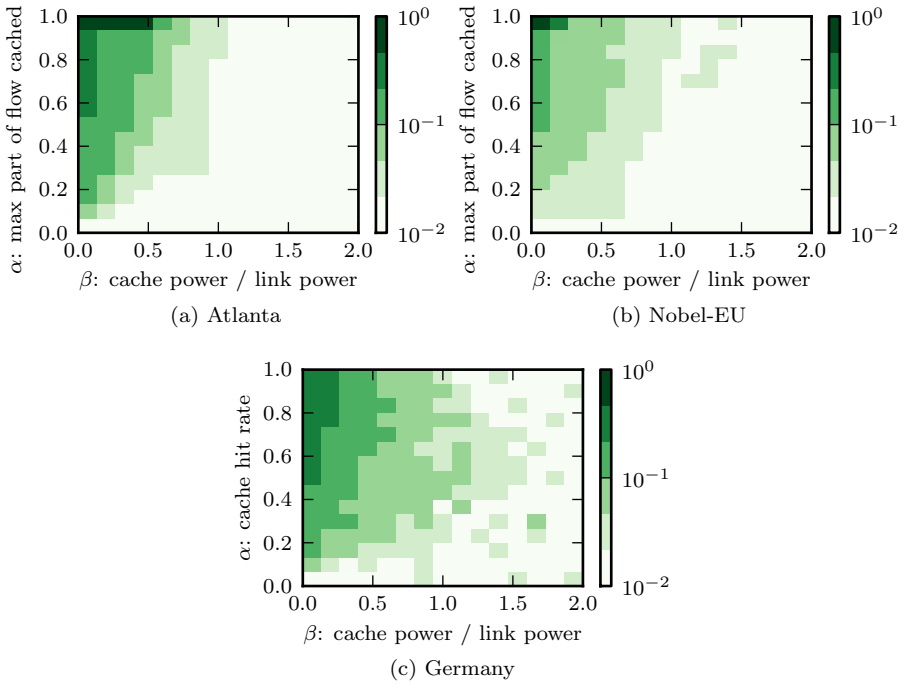


(b) Link usages.



(c) Total energy consumption.

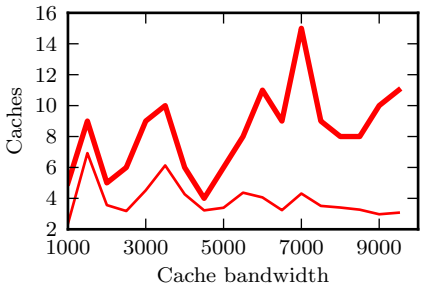
Figure 3: Results in function of parameter β.



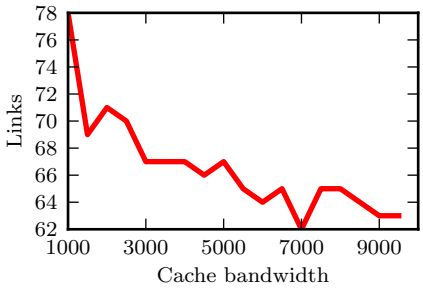
(a) Atlanta (b) Nobel-EU

(c) Germany

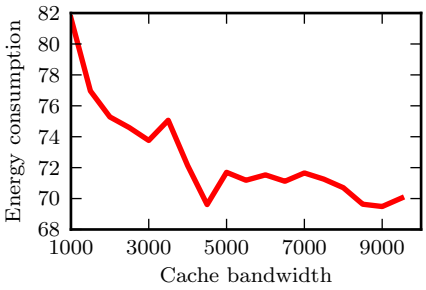
Figure 4: Part of energy consumption saved by introducing caches with various parameters.



(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 5: 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 2 and 3.

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.

5.2 CDN parameters

Then, we investigate the impact of the cooperation with CDN. Figure 6 shows the evolution of energy consumption in function of what part of all demands are directed towards CDN networks. The traffic has been reduced for this plot 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 idealized 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

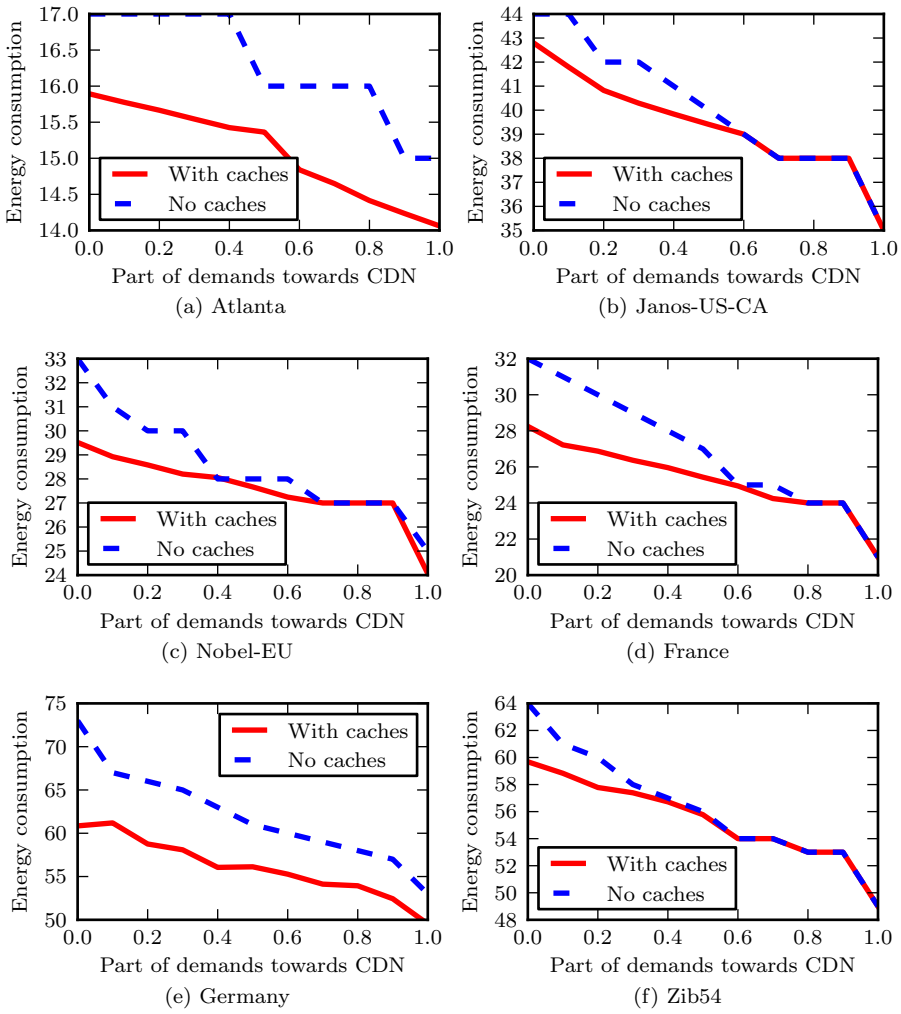


Figure 6: Impact of caches and CDN cooperation on energy consumption

.

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 2: Savings in different scenarios for different networks.

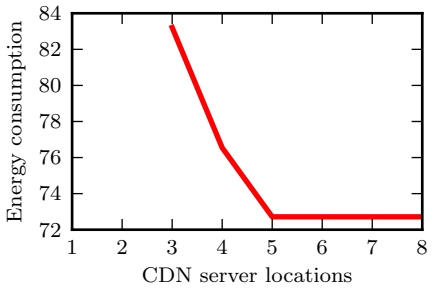


Figure 7: Impact of the number of CDN server locations in Germany.

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 2 shows the numbers, for cases described above, for all the networks. Savings are always compared to the case without caches nor CDN.

Figure 7 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, it is optimal to have 5 server locations in this network of 50 cities.

5.3 Impact of traffic level

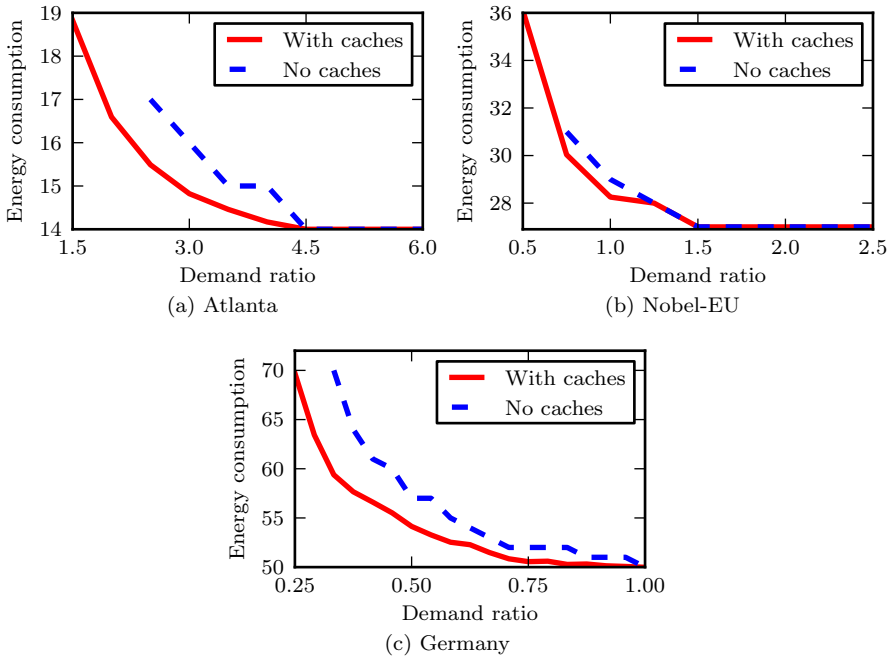
In this section, we look into the potential reduction of energy consumption of the networks in our model, both with and without usage of the caches. The parameters are: $\alpha = 0.35$, $\beta = 0.1$ and cache bandwidth is half of a link.

Figure 8 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 3 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 3 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.

What is interesting is the fact that caches have a much higher effect in the



(a) Atlanta (b) Nobel-EU

(c) Germany

Figure 8: Comparison of energy consumption with and without caches in the model.

Network	Maximum energy saved due to caches	Total energy savings at load = 50% max
Atlanta	8.9%	21.3%
Nobel-EU	3.2%	21.7%
Germany	16.7%	22.3%

Table 3: Potential energy savings

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 5.4.

Figure 9 plots the results into kinds of devices being on. As expected, turning on some caches allows us to turn off some links.

5.4 Impact of network size

We have seen different usage of caches in different networks. There is an explanation for that. First note that energy savings of serving from cache depend on how long would be the route traversed by the data if it was served from the network. Then, notice that even in the biggest network we used, the *germany50*, the average route length is only 4. We claim that in bigger networks we could see higher utility of caches.

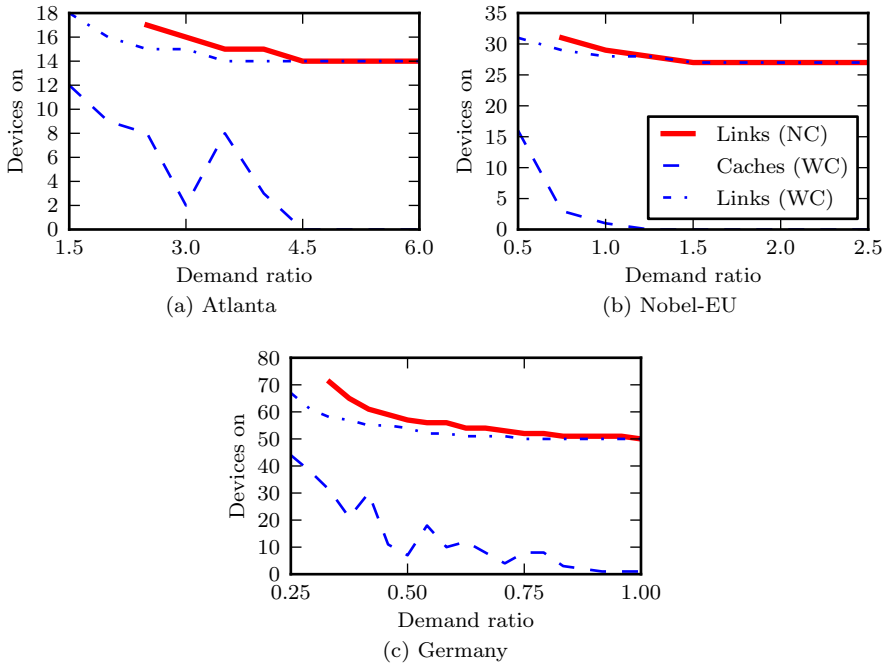


Figure 9: Breakdown of devices used in each case.

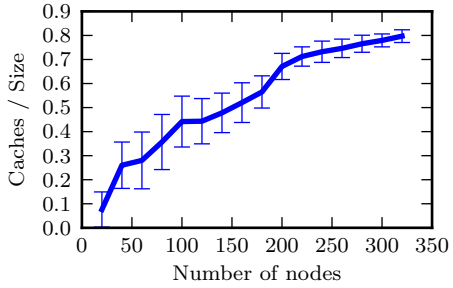


Figure 10: Cache usage divided by network size for Erdős-Rényi graphs.

To test this claim 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 would be impractical. Therefore, the results presented are for a fractional relaxation of the model.

Figure 10 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 $\alpha = 0.5$, $\beta = 1.0$ and $\gamma = 0.5$. Each data point is an average over 200 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 cache usage is only 7.7%, while in networks of size 320, of average route length around 4.5, it is 80.3%. Caches see an average usage over 50% for networks of size at least 160, where the average route length is only around 3.9.

6 Conclusions and further research

We have proposed a new model for saving energy in backbone networks by disabling equipment, taking into account in-router caches. It has been validated by solving instances based on real network topologies. The total energy savings found oscillate about 25% for realistic parameters. Part of energy saved solely due to introduction of caches is up to 16% in our instances.

In future work, the model can be extended by enabling sharing a single cache between multiple cities. Another interesting direction is modelling the problem from perspective of content provider. We can 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] SNDLib. [Online]. Available: http://sndlib.zib.de

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

Appendix

6.1 Heuristics

In this section, we describe two heuristic approaches based on the LP-rounding technique. We attack the problem with them from different sides. First, the *least loaded heuristic* begins with a network with all equipment turned on, gradually turning off the least loaded elements. On the other hand, the *spanning tree heuristic* begins with everything turned off and turns on additional equipment until a routing becomes feasible.

6.2 Fixing extremities

In both heuristics we face a fractional relaxation gradually approaching an integer solution. By intuition this process should be monotonic, especially in the extremities. If we are turning off devices with zero load, or turning on ones with full load, we are not changing the solution. Furthermore, we can say a similar thing of the opposite extremity. If a fractional load on a device is zero, then after turning on some additional devices it should still remain zero. An analogous thing should apply for the symmetric case. As we found out, it is not always the case.

We have proposed fixing the values to the integer once a variable approaches one of the extremities. In our tests it has proved both to highly reduce the computational cost and produce, in most cases, slightly better results than the versions without fixing. The procedure is outlined as Algorithm 1.

Algorithm 1: Fix Extremities

input : s: solution for Mixed Integer Linear Programming model for Energy Efficient Content Distribution

output: Mixed Integer Linear Programming model for Energy Efficient Content Distribution

foreach $a \in \{x_v, \forall v \in V\} \cup \{y_v, \forall v \in V\}$ **do**
  if $a < \varepsilon$ **then** add to s: $a = 0$
  if $a > 1 - \varepsilon$ **then** add to s: $a = 1$

return *s with the additional constraints*

6.3 Least Loaded Heuristic

This is a natural extension of the heuristic proposed in [5]. A routing is found using fractional relaxation of the linear program described in the previous section. Then, in each step, the least loaded device in the network is permanently turned off and a new routing is found. If the routing was not feasible, then the device is turned on instead. The algorithm ends when each device has either been turned on or off. Note that, without changing the result, all devices with zero load can be turned off at any moment. The procedure is shown more formally in Algorithm 2.

Algorithm 2: Least Loaded Heuristic

input : fractional relaxation of ILP model for ENERGY EFFICIENT CONTENT DISTRIBUTION
output: an integer solution to the model or INFEASIBLE

$s \leftarrow$ solution of the model
if *s is infeasible* **then**
  return INFEASIBLE
$A = \{x_v, \forall v \in V\} \cup \{y_v, \forall v \in V\}$
while $\exists_{a \in A}\ 0 < a < 1$ **do**
  Fix Extremities(s)
  choose $o = \min\{a \in A \,|\, 0 < a < 1\}$
  add to the model: $o = 0$
  $s \leftarrow$ solution of the model
  if *s is infeasible* **then**
    change in the model: $o = 1$
return *current solution to the model*

6.4 Spanning tree heuristic

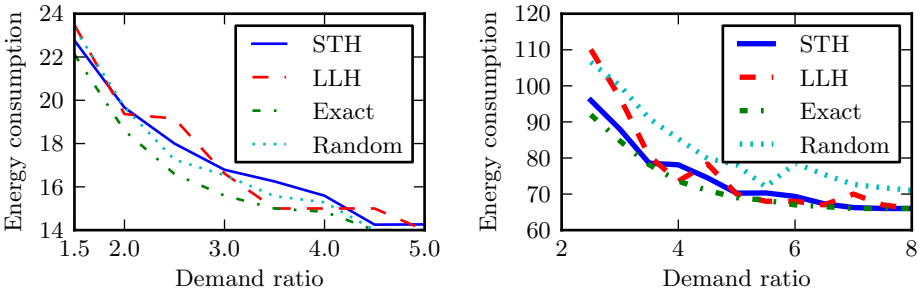
Algorithm 3: Spanning Tree Heuristic

input : fractional relaxation of ILP model for ENERGY EFFICIENT CONTENT DISTRIBUTION
output: an integer solution to the model or INFEASIBLE

$s \leftarrow$ solution of the model
if *s is infeasible* **then**
  return INFEASIBLE
$T =$ heaviest spanning tree, with x_{uv} as weight of each edge uv
add to the model: $x_{uv} = 1$ for each $uv \in T$
$A = \{x_v, \forall v \in V\} \cup \{y_v, \forall v \in V\}$
while $\exists_{a \in A}\ 0 < a < 1$ **do**
  Fix Extremities(s)
  choose $o = \max\{a \in A \,|\, 0 < a < 1\}$
  add to the model: $o = 1$
  $s \leftarrow$ solution of the model
return *current solution to the model*

This algorithm also bases on solving the fractional relaxation of the ILP. However, here we are greedily building a list of devices that need to be turned on for the routing to be feasible. This enables an important optimization: as the network has to be connected, we can constraint the model to contain a spanning tree in the first step. Once there are enough devices constrained to be turned on, the energy-optimal routing will not use any other ones and we can end the algorithm. Similarly to previous algorithm, this can also be sped up by immediately constraining all the integer values. The procedure is shown in Algorithm 3.

Due to some surprising results, displayed in section 5, we have believed that



(a) Atlanta (b) TA2

Figure 11: Comparison of energy consumption of networks obtained with different algorithms.

this heuristic has a tendency to turn on more caches that are actually used. Thus, we tried a slight modification. After obtaining an integer solution, we tried an additional step of turning off caches. This was analogical to Algorithm 2, but took into account only caches. In all test cases, it was found that the results were exactly equal with and without this step. In other words, no caches turned on by this heuristic could be turned off, without making routing infeasible.

6.5 Algorithms comparison

In this section we compare performance of both heuristics proposed here with exact solution and a random algorithm. The random algorithm is equivalent to Algorithm 2, with the exception that instead of the least used equipment, a random one is chosen to be decided on. This has the advantage, that if the right sequence of random choices is made, an optimal solution can be obtained. The random algorithm is repeated multiple times, in our setup 100 times, and the best solution is chosen. Note, that those repetitions can be done in parallel, as they are independent. We have also tried randomized rounding, where instead of choosing the best best equipment according to the relaxation, a random one is chosen weighted by the relaxation's result. This, when repeated many times, proved to give slightly better results than the greedy versions, but not worth the additional computation cost.

The results are plotted in Figure 11. Two network instances are chosen for comparison: Atlanta, for which an exact solution can be obtained in a short time, and TA2, the biggest instance in SNDLib. In the latter case, the solver is given limited time (in this case 1 hour on a quad core 2.7GHz CPU) to obtain each solution. The x axis of the plots is demand ratio, as described in previous sections. Other parameters are: $\alpha = 0.35$, $\beta = \gamma = 0.25$ and cache bandwidth is half of a link.

The heuristic results are in all cases, with the exception of Algorithm 2 in TA2 under very high traffic, remarkably close to those obtained directly from the solver. Thus, they allow, without seriously compromising the results, significant time savings. All computations for the first plot took 7 minutes with exact algorithm, 1 minute with Algorithm 2 and 18 seconds with Algorithm 3. For the TA2 instance it was 25 minutes with Algorithm 3, 3 hours with Algorithm 2

and 12 hours with the solver, which also could not provide exact solutions.

The random algorithm appears to be clearly inferior. It obtained better solutions than both heuristics in some cases in the smaller network, but at a computational cost that is 11 times higher than getting the exact solutions. In the bigger network it proved to both give inferior results and take significantly more time than any other algorithm. Note, that the running time of the random algorithm is approximately the time of Algorithm 2 multiplied by the number of repetitions.

6.6 Comparisons

The two scenarios have shown, among a number of parameters, the difference between two traffic profiles. The low traffic assumed is 60% of peak traffic, a conservative assumption. Now, for every set of parameters, we take the difference between total energy consumption under those parameters between the high and low traffic scenarios. We omit the ones, where routing was infeasible. We divide the difference by the energy usage in the higher scenario. This way we get relative energy savings between the scenarios. Mean savings are 23% with standard deviation 5%.

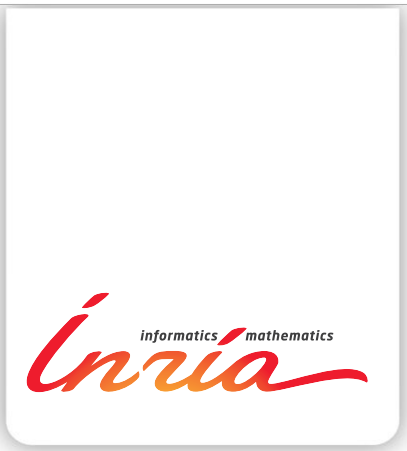
We use similar formula to compare the results of both heuristics. The Least Loaded Heuristic gave on average better results than the Spanning Tree Heuristic, but the difference is only 1% with standard deviation 6%.

6.7 Computational cost

The basic operation in each heuristic is optimization of the fractional relaxation (which itself is polynomial time in both number of cities and links). This analysis will be expressed in terms of this operation. Denote n the number of cities and m the number of links in the network.

The Least Loaded Heuristic makes its decisions one by one. These decisions can mean both turning the element off and on. Furthermore, it often happens that until the last elements the relaxed solution remains fractional. Thus, the number of operations performed usually is close to $n + m$.

On the other hand, the Spanning Tree heuristic starts by turning on $n - 1$ links and 1 additional element after the first optimization. As it continues to add elements, at some point it reaches a minimal network in which routing is feasible. At this point the usage of all additional elements in the fractional relaxation is zero, so the algorithm ends. Thus, the number of operations is $y - n$, where y is the number of elements turned on in the solution. This means that in average, for the high traffic scenario, the number of operations performed was only 31.5 with standard deviation of 13.9. This corresponds well to execution times seen in practice, where Algorithm 2 usually around 4 times more time to run than Algorithm 3 in the high traffic scenario. The discrepancy is, as expected, much higher in the case of lower traffic values.



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