Energy-aware Routing in Software-Defined Network using Compression

Frédéric Giroire[1], Nicolas Huin[1], Joanna Moulierac[1] and Truong Khoa Phan[2]

[1]CNRS, Laboratoire I3S, UMR 7172, UNS, CNRS, Inria, COATI, 06900 Sophia Antipolis, France
[2]Department of Electronic and Electrical Engineering University College London
Email: joanna.moulierac@unice.fr

Software-defined Networks (SDN) is a new networking paradigm enabling innovation through network programmability. Over past few years, many applications have been built using SDN such as server load balancing, virtual-machine migration, traffic engineering and access control. In this paper, we focus on using SDN for energy-aware routing (EAR). Since traffic load has a small influence on the power consumption of routers, EAR allows putting unused links into sleep mode to save energy. SDN can collect traffic matrix and then computes routing solutions satisfying QoS while being minimal in energy consumption. However, prior works on EAR have assumed that the SDN forwarding table switch can hold an infinite number of rules. In practice, this assumption does not hold since such flow tables are implemented in Ternary Content Addressable Memory (TCAM) which is expensive and power-hungry. We consider the use of wildcard rules to compress the forwarding tables.
In this paper, we propose optimization methods to minimize energy consumption for a backbone network while respecting capacity constraints on links and rule space constraints on routers. In details, we present two exact formulations using Integer Linear Program (ILP) and introduce efficient heuristic algorithms. Based on simulations on realistic network topologies, we show that using this smart rule space allocation, it is possible to *save almost as much power consumption as the classical EAR approach.*

Keywords: Software defined networks; Energy-aware Routing; Forwarding tables; Compression with wildcards; Algorithms; Green Networking.

Received April 2017

1. INTRODUCTION

The environmental footprint of the Information and Communication Technology (ICT) sector is a growing concern. According to [1], the sector's own emissions are expected to increase to 1.43 billion tons carbon dioxide equivalent, with 43% attributed to data centers and telecommunication networks. The reduction of CO2 emissions and the economic savings associated are thus an important issue in the scientific community.

Recent studies show that the traffic load of routers only has a small influence on their energy consumption [2, 3, 4]. Instead, the dominating factor is the number of active elements on routers such as ports, line cards, base chassis, etc. Therefore, in order to minimize the energy consumption, fewer network elements should be used while preserving connectivity and QoS.

Software Defined Network is a rising networking paradigm that proposes a centralized management of the network, in contrast with the current decentralized networks. The approach consists in separating the control plane from the data plane. The routers and switches become simple forwarding devices while one or multiple controllers do the heavy lifting by computing paths and instructing the routers how to handle the packets in the network using the OpenFlow protocol [5]. Reshaping the traffic is thus easier on an SDN since the controllers have a total knowledge of the topology and its usage. This flexibility eases the deployment of green policies on the network. The traffic can be easily aggregated on a subset of the network with a change in the Forwarding Information Base (FIB) of the switches and the unused link shut down.

To store the forwarding rules given by the controller, the switches use Ternary Content Address Memory (TCAM), an expensive and power hungry memory type. Moreover, the OpenFlow rules are more complex than their legacy counterpart. While the legacy rules match on only two fields, the OpenFlow rules can match on 40 fields in its version 1.3. The size of the forwarding tables is thus a scarce resource in an SDN, the maximum number of rules in a switch can vary from 750 to several thousands in current hardware [6, 7]. This limit can even be lower if we consider the use of IPv6 traffic. The capacity of the forwarding table is a non-negligible constraint of the deployment of SDN.

In this paper, we use *Software Defined Networks to deploy an Energy Aware Routing (EAR)* which routes the demands on the network *respecting the capacity constraints of the links and of the forwarding tables* while minimizing the energy consumption of the network. We use *wildcard rules to reduce the size of the forwarding tables.* These wildcard rules aggregate rules with the same action on corresponding fields. We study in particular two kinds of compression of forwarding tables: *default port compression*, using a wildcard forwarding all packets to a default port, and *multi-field compression*, using additional wildcards aggregating all flows with a field having a specific value (for example, all flows going to a specific destination). We name the problem considered here, Energy Aware Routing with Compression (EARC).

A short version of this work has been presented in [8]. To our best knowledge, this previous publication was the first work that defines and formulates the optimizing rule space problem in SDN for EAR. We provided an Integer Linear Programs to solve optimally EARC for the default port compression, a heuristic algorithm to provide solutions for large networks, and preliminary results about their compression and scalability properties. However, the previous proposition only used a very simple compression. In this work, we used the compression methods we proposed in [9] and tested on a data center networks. We adapted them to solve more efficiently the EARC problem.

In brief, our added contributions to the EARC problem are:

- We considered multi-field compression to optimize rule space in SDN for EAR.
- We provide a new Integer Linear Program to solve optimally EARC for two levels multi-field compression in Section 4.
- As EAR (and thus EARC) is known to be NP-hard [10], we consider heuristic algorithms for EARC that are effective for large network topologies in Section 5. The algorithm has three main modules: a compression module in charge of compressing the routing table, a routing module responsible for finding a route for each demand satisfying the capacity constraints and an energy module deciding which network link to turn off. In particular, we study several solutions for the compression problem.
- In Section 6, we compare the different solutions of the compression problem. We validate them on random forwarding tables as well as on network tables originating from simulations on networks of the SNDlib library [11].
- Using real-life data traffic traces from SNDlib, we quantify energy savings achieved by our approaches. Moreover, we also present other QoS aspects such as routing length and link load of EAR solutions in Section 7.

2. RELATED WORK

2.1. Classical Energy-aware Routing

Starting from the pioneering work of Gupta [12], the idea of power proportionality has gained a growing attention in networking research area. The power consumption of network equipment has been shown to be largely independent of the network load in [3, 2, 4]. Indeed, these papers show that the baseline power consumption with no traffic represents around 80% of the global power consumption when the router is fully-loaded. Therefore, the part induced by the traffic is roughly negligible compared to the fact to switch on an equipment. Therefore, people suggested putting network components to sleep in order to save energy.

Although power savings are worthwhile, performance effects must be minimal, and fault tolerance must be satisfied. Several algorithms have been proposed to find feasible routing solutions while satisfying QoS constraints and being minimal in power consumption. For instance, the authors in [3, 2] use mixed integer programming to optimize router power in a wide area network. Furthermore, other works on saving energy for data centers have also been presented [13, 14]. In general, these works show that up to 50% of network energy can be saved while maintaining the ability to handle traffic surges and guaranteeing QoS.In [15], the authors present a two-stage routing scheme where a simple distributed multipath finding algorithm is firstly performed to guarantee loop-free routing, and then a distributed routing algorithm is executed for energy-efficient routing in each node among the multiple loop-free paths. In [16], the authors propose a new framework for energy minimization in Data Center Networks and model the energy-saving problem with both VM assignment and network routing with respect to energy conservation.

2.2. Energy-aware Routing with Software Defined Network

Some recent works [17, 18, 19, 20] consider the problem of optimizing the power consumption in SDN using an energy-aware traffic engineering approach that minimizes the number of links. They present formulation of the optimization problem involving routing requirements for control and data plane communications. They evaluate their propositions through simulations. They ensure some performance constraints that are crucial for the correct operation of SDN such as bounded delay for the control plane traffic and load balance between controllers. [21] presents a general state of the art of energy efficiency approaches in software defined networking. However, these works do not address the problem of limited rule space.

2.3. Limited Rule Space in OpenFlow Switches

To support a vast range of network applications, OpenFlow rules are more complex than forwarding rules in traditional IP routers. For instance, access-control requires matching on source - destination IP addresses, port numbers, and protocol [22] whereas a load balancer may match only on source and destination IP prefixes [23]. These complicated matching can be well supported using TCAM since all rules can be read in parallel to identify the matching entries for each packet. However, as TCAM is expensive and extremely power-hungry, the on-chip TCAM size is typically limited. Several works have tackled the distribution of the forwarding policies on a network considering the table size constraints. In [24] and [25], the authors propose similar solutions in which the set of end points policies of the network is divided and then spread over the network so that every packet is affected by all the policies. However, the routing policies are not taken into consideration. In both [26] and [27], routing policies are dealt with by changing the path of the flows to take advantages of the table space from all the switches of the network. This type of solutions (including the one proposed in this paper) do however change the path used to route a flow and thus impact the QoS of the network. We go further than only changing paths by considering compression. In order to evaluate the impact on QoS of changing paths, we study the induced network delays of our method in Section 7. [28, 29] use compression methods to deal with limited rule space, as we do in our work. In [28] the authors introduce XPath which identifies end-to-end paths using path ID and then compresses all the rules and pre-install the necessary rules into the TCAM memory. This solution requires contacting the controller for every new flow entering the network to obtain the corresponding path ID. In [29], the authors suggest to following the concept of lost prefix matching with priorities for compression, using the Espresso [30] heuristic. They show that their algorithm leads to 17% savings only. In this paper, we use different compression methods, which are very efficient, as we use wildcards to aggregate rules based on several fields. This leads to savings over 80% in terms of numbers of rules for practical network forwarding tables. The compression solution using multi-field compression was tested with experiments on a small SDN platform for data center networks in [9, 31]. The experiments show that this is a realistic solution and that the impact of compression on failure rate and delay is negligible. In the present work, we extend this work by proposing new solutions for the combined problem of routing, compressing and minimizing the energy consumption, in the new context of ISP networks. In [32], a solution to enable energy-aware routing in a scenario of progressive migration from legacy to SDN hardware is presented. Since in real life, turning off network devices is a delicate task as it can lead to packet losses, the proposed solution, SENAtoR, also provides several features to safely enable energy saving services: tunneling for fast rerouting, smooth node disabling and detection of both traffic spikes and link failures.

2.4. Energy Savings with OpenFlow

Since, as stated, the power consumption of router is mostly independent of traffic load, people suggested putting network components to sleep to save energy. OpenFlow is a promising method to implement EAR in a network. Without setting entries manually, the SDN controller can collect traffic matrix, performs routing calculation and then installs new routing rules on routers using OpenFlow. For instance, the authors in [13] have implemented and analyzed ElasticTree on a prototype testbed built with production OpenFlow switches. The idea is to use OpenFlow to control traffic flows so that it minimizes the number of used network elements to save energy. Similarly, the authors in [14] have set up a small testbed using OpenFlow switches to evaluate energy savings for their model. OpenFlow switches have also been mentioned in existing work as an example of the traffic engineering method to implement the EAR idea [7]. However, as we can see, the testbed setups with real OpenFlow switches are quite small. For instance, in [13], 45 virtual switches onto two 144-port 5406 chassis switches are used; or in [14], there is a testbed with ten virtual switches on a 48-port Pronto 3240 OpenFlow-enabled switch. We argue that when deploying EAR in real network topologies, much more real OpenFlow switches should be used and they have to handle a large number of traffic flows. In this situation, limited rule space in switches becomes a serious problem since we can not route traffic as expected. Therefore, we present in next sections a novel optimization method to overcome the rule placement problem of OpenFlow for EAR.

3. PROBLEM DEFINITION

Energy Aware Routing with Compression (EARC) We consider a backbone network as a *directed* graph $G = (V, A)$. The nodes in V describe routers and the arcs in A represent connections or links between those routers. The links have a limited capacity. We denote by C_{uv} the capacity of a link (u, v). The nodes have a limited memory space to store rules and we note C_u the maximum number of rules can be installed at router u. We denote by D^{st} the demand of traffic flow from node s to node t such that $D^{st} \geq 0, s, t \in V, s \neq t$. The *objective is to find a feasible routing for all traffic flows, respecting the capacity and the rule space constraints and being minimal in energy consumption.* We name the problem *Energy Aware Routing with Compression (EARC).* Note that EAR refers to the same problem without rule space constraints. The energy consumption of the network is given by the number of active links in our model. We consider that routers have to stay powered on in backbone networks as they are as they are entry and exit points of the traffic.

Power Model. Campaigns of measures of power consumption (see, e.g., [3] and [4]) show that a network device consumes a large amount of its power as soon as it is switched on. Following this observation, on/off power models have been proposed and studied [33]. Later, researchers and hardware constructors have proposed more energy proportional hardware models [34]. To encompass those different models, see [35] for a discussion, we use a hybrid power model in which the power of an active link uv is expressed as

$$P_{uv}^{\text{ON}} + \frac{\text{BW}_{uv}}{C_{uv}} P_{uv}^{\text{MAX}}$$

where P_{uv}^{ON} represents the energy used when the link uv is switched on, BW_{uv} the bandwidth that is carried on uv, and P_{uv}^{MAX} the energy consumed by uv when it is fully capacitated, i.e., when the amount of carried bandwidth equals the transport capacity (C_{uv}) of link uv. We assume that links can be put into sleep mode, by putting to sleep both endpoint interfaces. Routers cannot be put into sleep mode, as there are the sources/destinations of network traffic.

Compression problem A forwarding table is composed of multiple entries that match flows with corresponding action(s). In OpenFlow 1.0, the matching can be done on 12 fields from the packet header. For each field, the matching rule can use a specific value or a wildcard (noted $*$) that can accept any value. The action associated with a matching rule can be to drop the packet, modify the header, or forward to a specific port.

In the following, we consider the action to be limited to a forward to outgoing ports. We also limit the use of the wildcard to the source and destination addresses. However, our solution also applies if other fields are considered such as ToS (Type of Service) field or transport protocol. We compress a table by using either the aggregation by source (i.e $(s, *, p)$), by destination (i.e $(*, t, p)$) or by the default rule (i.e $(*, *, p)$). When only the default rule is used, we talk of *default port compression*, and, when all the wildcard may be used, of *multi-field compression.*

An example is given in Table 1. Table 1(a) represents the original table. Table 1(b) provides the result using default port compression and Table 1(c) the one using multi-field compression, that is when all three types of wildcard rules can be used to obtain the optimal compressed table. *Since multiple entries can correspond to the same couple of source and destination, rules are considered in the order of the table. A rule on top of the table has priority over a rule below.* Indeed, in Table 1(c), if we exchange the priorities of the rules $(1, *, Port-6)$ and $(*, 4, Port-4)$, a flow from source 1 to destination 4 is no longer forwarded through Port-4 like in the original table.

4. INTEGER LINEAR PROGRAMMING

We propose two Integer Linear Programs to solve the EARC problem. In the first one, *EARC-LP-Default*, only the default port compression is allowed, while multi-field compression is used in the second, *EARC-LP-Multi*. The first program thus is less powerful but runs faster. We were able to obtain optimal solutions for small networks using both ILP.
The following notations are used in both formulations:

- $x_{uv} \in \{0, 1\}$, where $x_{uv} = 1$ if the link (u, v) is active or not.
- $\mathcal{D}$: the set of all traffic demands to be routed.
- $D^{st} \in \mathcal{D}$: demand of traffic flow from s to t.
- C_{uv}: capacity of a link (u, v).
- C_u: maximum number of rules can be installed at router u.
- $N^+(u)$ and $N^-(u)$: set of outgoing and incoming neighbors of u, respectively.

4.1. EARC with default port Compression (EARC-LP-Default)

In this version of the problem, a flow can be routed following the FIB that contains only *perfect match* rules (rules with no wildcard that match exactly the source and destination address), or via the default port. The following notations are used in the formulation of the ILP:

- $f_{uv}^{st} \in \{0, 1\}$, where $f_{uv}^{st} = 1$ if (s, t) that is routed on the link (u, v) by a distinct rule. We call f_{uv}^{st} as normal flow.
- $g_{uv}^{st} \in \{0, 1\}$, where $g_{uv}^{st} = 1$ if (s, t) that is routed on the link (u, v) by a default rule. g_{uv}^{st} is called default flow to distinguish from the normal flow f_{uv}^{st}.
- $f_{uv} \in \mathbb{R}^+$: sum of the flows routed on the link (u, v).
- $k_{uv} \in \{0, 1\}$, where $k_{uv} = 1$ if the default port of the router u is p_v.

Flow	Output port
(0, 4)	Port-4
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 5)	Port-4
(1, 6)	Port-6
(2, 4)	Port-4
(2, 5)	Port-5
(2, 6)	Port-6

(a) Without compression

Flow	Output port
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 6)	Port-6
(2, 5)	Port-5
(2, 6)	Port-6
$(*, *)$	Port-4

(b) Default port compression

Flow	Output port
(1, 5)	Port-4
(2, 6)	Port-6
$(1, *)$	Port-6
$(*, 4)$	Port-4
$(*, *)$	Port-5

(c) Multi-field compression

TABLE 1: Examples of routing tables: (a) without compression, (b) with default port compression (only $(*, *)$ rule), (c) with multi-field compression (three possible aggregation rules), giving the routing table with minimum number of rules.

We want to minimize the power consumption of the network (1).

$$\min \sum_{(u,v)\in A} \left(P_{uv}^{\text{ON}} x_{uv} + P_{uv}^{\text{MAX}} \frac{f_{uv}}{C_{uv}} \right) \tag{1}$$

The flow conservation constraints (2) express that the total flows entering and leaving a router are equal (except the source and the destination nodes). Note that a normal flow entering a router can become a default flow on an outgoing link and vice versa.

$$\sum_{v\in N(u)} f_{vu}^{st} + g_{vu}^{st} - g_{uv}^{st} - f_{uv}^{st} = \begin{cases} -1 & \text{if } u = s, \\ 1 & \text{if } u = t, \\ 0 & \text{else} \end{cases}$$
$$\forall u \in V, (s,t) \in \mathcal{D} \tag{2}$$

A flow cannot be routed as both a default flow and a normal one (3).

$$f_{uv}^{st} + g_{uv}^{st} \leq 1 \qquad \forall (u,v) \in A, (s,t) \in \mathcal{D} \tag{3}$$

Link capacities constraints are given by Equation (4) and no flow is allowed to be forwarded on a disabled link.

$$f_{uv} = \sum_{(s,t)\in\mathcal{D}} D^{st}(f_{uv}^{st} + g_{uv}^{st}) \leq C_{uv} x_{uv} \qquad \forall (u,v) \in A \tag{4}$$

The total number of rules in the table of a router is equal to the sum of the normal flow forwarded from the router. It cannot exceed the table capacity C_u minus the reserved rule for the default port (5).

$$\sum_{(s,t)\in\mathcal{D}} \sum_{v\in N(u)} f_{uv}^{st} \leq C_u - 1 \qquad \forall u \in V \tag{5}$$

Finally, we limit the number of default port to one per router (6). A demand can only be forwarded on an edge as a default flow if is the edge of the default port (7).

$$\sum_{v\in N(u)} k_{uv} \leq 1 \qquad \forall u \in V \tag{6}$$

$$g_{uv}^{st} \leq k_{uv} \qquad \forall (u,v) \in A, (s,t) \in \mathcal{D} \tag{7}$$

4.2. EARC with multi-field Compression

In this version, we consider that the forwarding table contains several wildcard rules. These rules can match any flow that comes from a source s (i.e., $(s, *, p)$) or goes to a destination t (i.e., $(*, t, p)$). The following notations are used for the formulation of the ILP:

- $\mathcal{S}$, set of all sources.
- $\mathcal{T}$, set of all destinations.
- $f_{uv}^{st} \in \{0, 1\}$, where $f_{uv}^{st} = 1$ if the flow (s,t) is routed on the link (u, v).
- $f_{uv} \in \mathbb{R}^+$: sum of the flows routed on the link (u, v).

or each router u, we also define the following sets of variables used for the compression of the tables, similar to the ones defined in Section 5.1.2. We use the notation p_v to define the port connected to the router v.

- $r_{stp_v}^u \in \{0, 1\}$, where $r_{stp_v}^u = 1$ if the rule (s, t, p_v) exists.
- $gs_{tp_v}^u \in \{0, 1\}$, where $gs_{tp_v}^u = 1$ if the wildcard rule $(*, t, p_v)$ exists.
- $gt_{sp_v}^u \in \{0, 1\}$, where $gt_{sp_v}^u = 1$ if the wildcard rule $(s, *, p_v)$ exists.
- $d_p^u \in \{0, 1\}$, where $d_{p_v}^u = 1$ if p_v if the default port.
- $o_{st}^u \in \{0, 1\}$, where $o_{st}^u = 1$ if the wildcard rule for the source s has higher priority than the one for the destination t.
- $\overline{o_{ts}^u} \in \{0, 1\}$, where $\overline{o_{ts}^u} = 1$ if the wildcard rule for the destination t has higher priority than the one for the source s. By definition, $o_{st}^u = 1 - \overline{o_{ts}^u}$.

We want to minimize the power consumption of the network (8).

$$\min \sum_{(u,v)\in A} \left(P^{\text{ON}}_{uv} x_{uv} + P^{\text{MAX}}_{uv} \frac{f_{uv}}{C_{uv}} \right) \tag{8}$$

Flow conservation is ensured via the following set of constraints:

$$\sum_{v\in N^+(u)} f^{st}_{uv} - \sum_{v\in N^-(u)} f^{st}_{vu} = \begin{cases} 1 & \text{if } u = s, \\ -1 & \text{if } u = t, \\ 0 & \text{else} \end{cases} \qquad \forall u \in V, (s,t) \in \mathcal{D} \tag{9}$$

The sum of the flows on a link cannot exceed its capacity. Moreover, if the link is disabled, no flow can be forwarded on it (10).

$$f_{uv} = \sum_{(s,t)\in\mathcal{D}} D^{st} f^{st}_{uv} \leq C_{uv} x_{uv} \qquad \forall (u,v) \in A \tag{10}$$

The following sets of constraints are similar to the ones presented for Comp-LP. The principal change is that the table is no longer an input of the problem and depend on the routing of the demands. Thus, if a demand (s,t) is forwarded on the link (u,v), there must exists a corresponding rule on the router u (11). Moreover, a non aggregated rule can only exists if the demand (s,t) is forwarded on the link (u,v) (12).

$$r^u_{stp_v} + gs^u_{tp_v} + gt^u_{sp_v} + d^u_{p_v} \geq f^{st}_{uv} \qquad \forall (u,v) \in A, (s,t) \in \mathcal{D} \tag{11}$$

$$r^u_{stp_v} \leq f^{st}_{uv} \qquad \forall (u,v) \in A, (s,t) \in \mathcal{D} \tag{12}$$

The total number of rules in the table of a router u cannot exceed its capacity C_u

$$\sum_{v\in N^+(u)} \left(d^u_{p_v} + \sum_{(s,t)\in\mathcal{D}} r^u_{stp_v} + \sum_{t\in\mathcal{T}} gs^u_{tp_v} + \sum_{s\in\mathcal{S}} gt^u_{sp_v} \right) \leq C_u \qquad \forall u \in V \tag{13}$$

We limit to one the number of default port (14), the number of wildcard rules for one source (15) and for one destination (16).

$$\sum_{v\in N^+(u)} d^u_{p_v} \leq 1 \qquad \forall u \in V \tag{14}$$

$$\sum_{v\in N^+(u)} gs^u_{tp_v} \leq 1 \qquad \forall u \in V, t \in \mathcal{T} \tag{15}$$

$$\sum_{v\in N^+(u)} gt^u_{sp_v} \leq 1 \qquad \forall u \in V, s \in \mathcal{S} \tag{16}$$

For a given demand (s,t) routed on a link (u,v_1), if a matching wildcard rule exists with different port than p_{v_1}, the table must contain the unaggregated rule (s,t,p_{v_1}) or another wildcard rule, $(s,*,p_{v_1})$ or $(*,t,p_{v_1})$, with a higher priority (it means that it will appeared before in the forwarding table). This is to impose that the flow will be effectively routed through the right port p_{v_1}.

$\forall u \in V, (s,t) \in \mathcal{D}, v_1 \neq v_2 \in N^+(u)$:

$$r^u_{stp_{v_1}} + gs^u_{tp_{v_1}} \geq gt^u_{sp_{v_2}} - 1 + f^{st}_{up_{v_1}} \tag{17}$$

$$r^u_{stp_{v_1}} + \overline{o^u_{ts}} \geq gt^u_{sv_2} - 1 + f^{st}_{up_{v_1}} \tag{18}$$

$$r^u_{stp_{v_1}} + gt^u_{sp_{v_1}} \geq gs^u_{tp_{v_2}} - 1 + f^{st}_{up_{v_1}} \tag{19}$$

$$r^u_{stp_{v_1}} + o^u_{st} \geq gs^u_{tp_{v_2}} - 1 + f^{st}_{up_{v_1}} \tag{20}$$

Finally, we remove cyclic order dependencies between rules.

$$1 \leq o^u_{s_1t_1} + \overline{o^u_{t_1s_2}} + o^u_{s_2t_2} + \overline{o^u_{t_2s_1}} \leq 3 \qquad \forall u \in V, s_1 \neq s_2 \in \mathcal{S}, t_1 \neq t_2 \in \mathcal{T} \tag{21}$$

Both linear programs run for small networks. In particular, we were able to obtain optimal solutions for the atlanta network from SNDLib, which has 15 nodes and 22 bi-directional links, see Section 7. However, the running time increases very quickly as the Energy Aware Routing problem is NP-Hard [10]. Thus, we propose efficient heuristic algorithms for larger networks in the following section.

5. HEURISTIC ALGORITHMS

As the linear programs proposed in the previous section do not run for medium and large networks, we propose here efficient heuristic algorithms. The problem can be decomposed into three sub-problems:

- First, the *compression problem* consists in reducing the size of a single table by using aggregation rules: the default rule for default port compression, and, additionally, source or destination rules for the multi-field compression.
- Second, the *routing problem* goal is to compute and assign a path in the network for each demand, while respecting the link and forwarding table capacities.
- Last, the *energy saving problem* goal is to shut down a maximum number of links while maintaining a valid routing in the network for all the flows.

The heuristic algorithm is thus composed of three different modules designed to solve these sub-problems. For the compression module, we propose multiple heuristics for the two levels of compression (default port and multi-field compression).

5.1. Compression module

We propose several solutions to solve the compression problem: First, *Comp-Default*, giving optimal solutions for the default port compression. We then provide an integer linear program, *Comp-LP*, which gives optimal solutions for the multi-field compression. However, as the problem is NP-Hard (see [36] for a proof), the program does not scale to large tables (also see Section 6 for compression time and a discussion). We thus provide two heuristic algorithms for the compression problem, *Comp-Greedy* and *Comp-Direction*.

5.1.1. *Default Rule (Comp-Default)*

When using only the default port compression, finding the optimal solution is simple. The algorithm finds the most occurring port p^* in the forwarding table, remove all the rules with p^*, and add the default rule $(*,*,p^*)$ at the end of the table.

5.1.2. *Integer Linear Programming (Comp-LP)*

We first define the following notations. We then formulate the problem as an Integer Linear Program. Note that a large number of constraints are similar to the one of EARC-LP-Multi. We yet decided to provide here the full program to avoid confusion, even if it is at the cost of some repetitions for the reader.

- $\mathcal{R}$, set of rules in the forwarding table
- $\mathcal{S}$, set of sources in the forwarding table
- $\mathcal{T}$, set of destinations in the forwarding table
- $\mathcal{P}$, set of ports of the router
- $r_{stp} \in \{0,1\}$, where $r_{stp} = 1$ if the rule (s,t,p) exists
- $gs_{tp} \in \{0,1\}$, where $s_{tp} = 1$ if the wildcard rule $(*,t,p)$ exists
- $gt_{sp} \in \{0,1\}$, where $t_{sp} = 1$ if the wildcard rule $(s,*,p)$ exists
- $d_p \in \{0,1\}$, where $d_p = 1$ if p is the default port of the table
- $o_{st} \in \{0,1\}$, where $o_{st} = 1$ if the wildcard rule for the source s has higher priority than the one for the destination t.
- $\overline{o_{ts}} \in \{0,1\}$, where $\overline{o_{ts}} = 1$ if the wildcard rule for the destination t has higher priority than the one for the source s. By definition, $o_{st} = 1 - \overline{o_{ts}}$.

We want to minimize the total number of rules in the compressed table (22).

$$\min \quad \sum_{(s,t,p)\in\mathcal{R}} r_{stp} + \sum_{p\in P}\left(\sum_{t\in T} gs_{tp} + \sum_{s\in S} gt_{sp}\right) \tag{22}$$

All original rules must have a corresponding rule in it (23).

$$r_{stp} + gs_{tp} + gt_{sp} + d_p \geq 1 \qquad \forall (s,t,p)\in\mathcal{R} \tag{23}$$

There can be at most one default port (24), one wildcard rule per source (25) and one wildcard rule per destination (26) in the table.

$$\sum_{p\in\mathcal{P}} d_p \leq 1 \tag{24}$$

$$\sum_{p\in\mathcal{P}} gs_{tp} \leq 1 \qquad \forall t\in\mathcal{T} \tag{25}$$

$$\sum_{p\in\mathcal{P}} gt_{sp} \leq 1 \qquad \forall s\in\mathcal{S} \tag{26}$$

For every rule (s,t,p) in the original table, if a matching wildcard rules exists with a different port, i.e., $(s,*,p' \neq p)$ or $(*,t,p' \neq p)$, either the original rule (s,t,p) or a matching wildcard rule with the right port exists with a higher priority, (27) to (30).

$$r_{stp} + gs_{tp} \geq gt_{sp'} \qquad \forall (s,t,p)\in\mathcal{R}, p'\in\mathcal{P}\setminus\{p\} \tag{27}$$

$$r_{stp} + \overline{o_{ts}} \geq gt_{sp'} \qquad \forall (s,t,p)\in\mathcal{R}, p'\in\mathcal{P}\setminus\{p\} \tag{28}$$

$$r_{stp} + gt_{sp} \geq gs_{tp'} \qquad \forall (s,t,p)\in\mathcal{R}, p'\in\mathcal{P}\setminus\{p\} \tag{29}$$

$$r_{stp} + o_{st} \geq gs_{tp'} \qquad \forall (s,t,p)\in\mathcal{R}, p'\in\mathcal{P}\setminus\{p\} \tag{30}$$

Finally, we remove the cyclic order dependencies between rules.

$$1 \leq o_{s_1t_1} + \overline{o_{t_1s_2}} + o_{s_2t_2} + \overline{o_{t_2s_1}} \leq 3 \qquad \forall s_1 \neq s_2 \in \mathcal{S}, t_1 \neq t_2 \in \mathcal{T} \tag{31}$$

5.1.3. *Most savings heuristic (Comp-Greedy)*

For this heuristic algorithm, we add wildcard rules for sources or destination in a greedy way, based on the highest potential compression ratio. The *potential compression ratio* of a source s (or destination t) is equal to the number of rules with the most repeated port p among all the rules with s (or t) over the total number of rules with s (or t). At each step, we compute the potential compression ratio of all sources and destinations. We then add the wildcard rule corresponding to the source or destination with the highest potential compression ratio, and we remove all the rules matching the wildcard rule. Note that at each step, the compression ratios of the other sources can be affected. We thus recompute them at each step.

5.1.4. *Direction Based Heuristic (Comp-Direction or Comp-Dir in short)*

We present a second heuristic which is a 3-approximation of the compression problem [36].

This heuristic computes three different compressed routing tables and, then, chooses the smallest one. An example can be seen in Figure 2. We consider the routing table given in Fig. 2(a).

Flow	Out. port
(0, 4)	Port-4
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 5)	Port-4
(1, 6)	Port-6
(2, 4)	Port-4
(2, 5)	Port-5
(2, 6)	Port-6

(a) Original table

Flow	Out. port
$(0,4)$	Port-4
$(1,5)$	Port-4
$(2,4)$	Port-4
$(2,5)$	Port-5
$(0,*)$	Port-5
$(*,*)$	Port-6

(b) Source and default

Flow	Out. port
(1, 4)	Port-6
$(1,5)$	Port-4
$(0,6)$	Port-5
$(*,4)$	Port-4
$(*,5)$	Port-5
$(*,*)$	Port-6

(c) Dest. and default

Flow	Out. port
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 6)	Port-6
(2, 5)	Port-5
(2, 6)	Port-6
$(*,*)$	Port-4

(d) Default only

TABLE 2: The three compressed tables of the Comp-Direction heuristic on the example presented in (a): (b) Compression using source aggregation rule and then default rule; (c) Compression using destination aggregation rule and then default rule; (d) Default port compression. The heuristic selects the smallest table among (b), (c) and (d).

To compute the first compressed table, the algorithm considers all the sources one by one. For each source s, it finds the most occurring port p^*, and replace all the matching rules with $(s, *, p^*)$. If there is a coincidence of the number of times a port occurs, we choose one port randomly for the matching rule. The remaining rules $(s, t, p \neq p^*)$ stay unchanged and have priority over the wildcard rule. Once all the sources have been considered, we do a pass over all the wildcard rules. We aggregate them by finding the most occurring port in the wildcard rules and by setting it as the default port. The default port rule has the lowest priority of all the rules. The first compressed table is given in Fig. 1(b).

The second compressed routing table is obtained with the same method, but when considering an aggregation by destinations ($(*, t, p^*)$ rules) and not by sources. The second compressed table can be seen in Fig. 1(c).

The third and last one (Fig. 1(d)) is the result of a single aggregation using the best default port.

5.2. Routing module

The *routing module takes as input a sub-network H given by the energy savings module and tries to find a feasible routing.* Its principle is to try to *spread the flows over the sub-network* as much as possible in order to avoid to overload a link or a routing table.

The module uses a *shortest-path algorithm* with an *adaptive metric in a residual graph H^R, defined in the following.* At the beginning of the algorithm $H^R = H$. We route the demands one by one. Consider we have already routed k demands and that we are in step k. The residual graph is H^R_k. For a demand between two nodes s and t of load d, we create a new subgraph H'^R_k by first removing all arcs with capacities lower than d. We then consider routers with a full forwarding table. For such a router u, we distinguish two cases. If its forwarding table does not contain any wildcard rule which could route the demand (s, t), we remove the router u from the residual graph. On the contrary, if such a wildcard rule exists[3], we keep the router u in the residual graph, but we remove all outgoing links corresponding to a port different than p.

We then compute a route by finding a shortest path between s and t in H'^R_k with the set of weights defined below.

The weight w_{uv} of a link depends (1) on the total flow using the link corresponding to demands previously routed, and (2) on the table's usage of router u also corresponding to demands previously routed. We note w^c_{uv} the weight corresponding to the link capacity and w^r_{uv} the weight corresponding to the rule capacity. They are defined as follows:

$$w^c_{uv} = \frac{f_{uv} + d}{C_{uv}}$$

where C_{uv} is the capacity of the link (u, v) and f_{uv} the total flow on (u, v). The more the link is used, the heavier the weight is. This favors the use of links with lower load, leading to load-balancing. And

$$w^r_{uv} = \begin{cases} \frac{|R_u|}{S_u} & \text{if } \nexists \text{ wildcard rule for } (s, t, v) \\ 0 & \text{otherwise} \end{cases}$$

where S_u is the maximum table size of router u and R_u is the set of rules for router u. The weight is proportional to the usage of the table.

Finally, the weight w_{uv} of a link (u, v) is given by:

$$w_{uv} = 1 + \alpha w^c_{uv} + \beta w^r_{uv} \qquad (32)$$

The importance of links is given by the parameter α and the one of tables by β. If α is larger than β, we give more weight to links.

If a path is found for a demand, we build the residual graph H^R_{k+1} for the next step of the algorithm. For each

[3]It should be of the form $(s, *, p)$, $(*, t, p)$ or $(*, *, p)$, where p represents also the output port of u towards v. Note that several wildcard rules may be present in the forwarding table of u. In this case, we consider the first one in the order of priority, as it will be the one routing the demand (s, t)

arc (u, v) of the path, we add the rule (s, t, v) to the router u if no corresponding wildcard rule exists. If the table is full, one of the compression methods previously described is used to reduce the size of the table[4].We also reduce the capacity of the arc by d from the one in H_k^R. If no path is found, the routing module returns that no feasible routing was found.

Setting the parameters of routing module. The metric to find a path for each demand (Equation 32) combines link usage and link capacity with table capacity and table usage. We compared several choices of metrics in [37] for different networks. The results in terms of energy savings are similar for metrics 3:1 and 1:3 (with slightly better results for the first one), and they are both always better than metric 1:1. Therefore, we choose the metric 3:1 for the remaining of the paper.

5.3. Energy savings Module

The energy savings module uses a greedy approach to select the links to switch off. It tries to remove links that are less loaded and to accommodate their traffic on other links in order to reduce the total number of active links.

The algorithm is simple. We start with the full network. We launch the routing module to try to find a feasible routing for all the demands. If such a routing exists, we try to remove the edge with the lowest load. We then re-launch the routing module on the network without the considered edge. If a feasible routing is found, we continue and try to switch off another edge. If no feasible routing is found, we put back the edge, and we try to remove the edge with the second lowest load. An edge, which was selected and could not be removed, is not considered anymore. The algorithm stops when all edges have been selected once.

6. COMPRESSION OF FORWARDING TABLES

We test here the compression module. The goal is to evaluate typical compression ratios, compression times and to compare the different solutions proposed in Section 5.1: the solution using default port compression (Comp-Default) and the three solutions using the multi-field compression: the optimal one from the Linear Program (Comp-LP), when it is possible to compute it, the Direction Based Heuristic (referenced as Comp-Direction or Comp-Dir in short), and last, the Most savings heuristics (Comp-Greedy). As instances, we consider random routing tables as well as network routing tables coming from simulations on SNDlib instances [11].

[4]Note that, if at some point, the compression method cannot further compress the table, the algorithm should remember this fact to avoid relaunching a useless compression each time and increasing the execution time.

6.1. Random tables

In this section, we focus on the compression of random tables. The following parameters are used to generate the random tables studied:

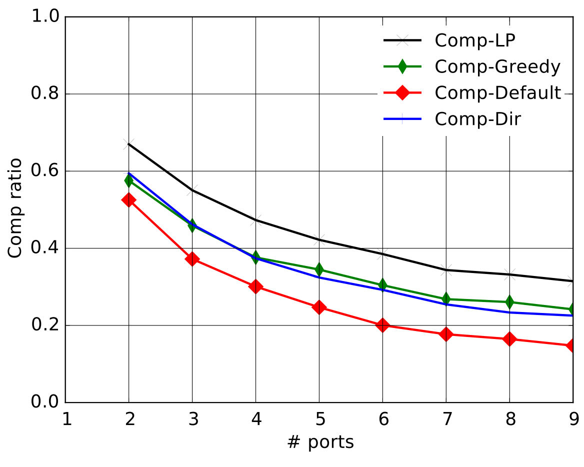
- the number of sources and destinations n
- the number of ports of the switch p
- the density of the corresponding matrix $0 \le d \le 1$

For a pair source-destination, there is an entry in the table with probability d, and in this case, the exiting port is chosen uniformly at random among the p ports.

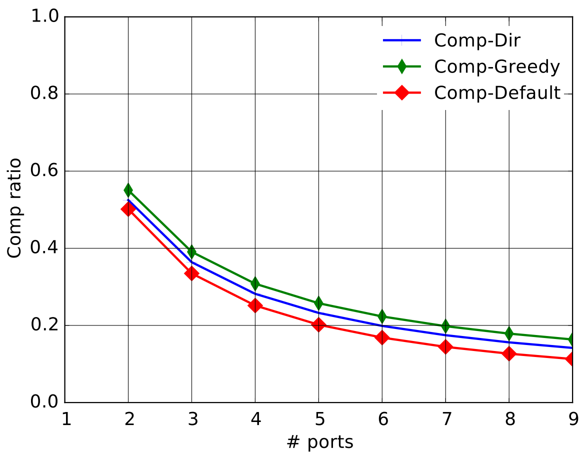
We show the average compression ratio of the solutions proposed in Section 5.1 as a function of the parameters used to build the random matrices. We vary the number of ports in the experiments of Figure 1, the number of network nodes (corresponding to the number of sources and destinations) in Figure 2, and the table density in Figure 3. Each point represents the average of the results for 10 random forwarding tables for the comparison with the LP and 20 for the heuristics.

Gap from optimal for small tables. For small routing tables, we are able to compute the optimal compressed tables using the linear program (see Figure 1(a), Figure 2(a) and Figure 3). As an example, in Figure 1(a), we compare Comp-LP and the other three solutions on a set of random tables with $n = 15$ sources/destinations, a density of 0.5 with a number of ports between 2 and 9. Without surprise, the ILP compresses better than the other 3 solutions with 68% ratio at only two ports to 32% with nine ports. The two heuristics present the same compression with a ratio of 59% at two ports and 23% at nine ports. Finally, the only use of the default port yields to the worst compression as it compresses 53% of the rules with 2 ports and only 15% at nine ports. Similarly, the difference of compression ratio in Figure 3 is between 4 and 10% when comparing the optimal solution with the Comp-Greedy and Comp-Direction heuristics. Default port is the less efficient solution with a compression ratio around 23%, when the compression ratio of Comp-Greedy and Comp-Direction heuristics is around 30%. In Figure 2(a), we vary the number of network nodes between 5 and 11. The global comparison between solutions is similar, except that, when there is a small number of network nodes, Comp-Greedy does not behave well and provides worse results than Comp-Default. The explanation is that, for small tables, Comp-Greedy adds source and destination aggregation rules that are not necessary, as a default rule works well. Because of the order between source and destination rules, most of these rules cannot be aggregated when we add the default rule, leading to an inefficiency. The problem disappears for larger numbers of network nodes (larger than 10), and thus would not appear for ISP networks which have more network nodes.

Comparison between heuristics for larger tables. However, the ILP does not scale well for larger



(a) ~ 112 rules $(n = 15, d = 0.5)$



(b) $\sim 101\,250$ rules $(n = 450, d = 0.5)$

FIGURE 1: Compression ratio as a function of the number of ports for the four compression methods.

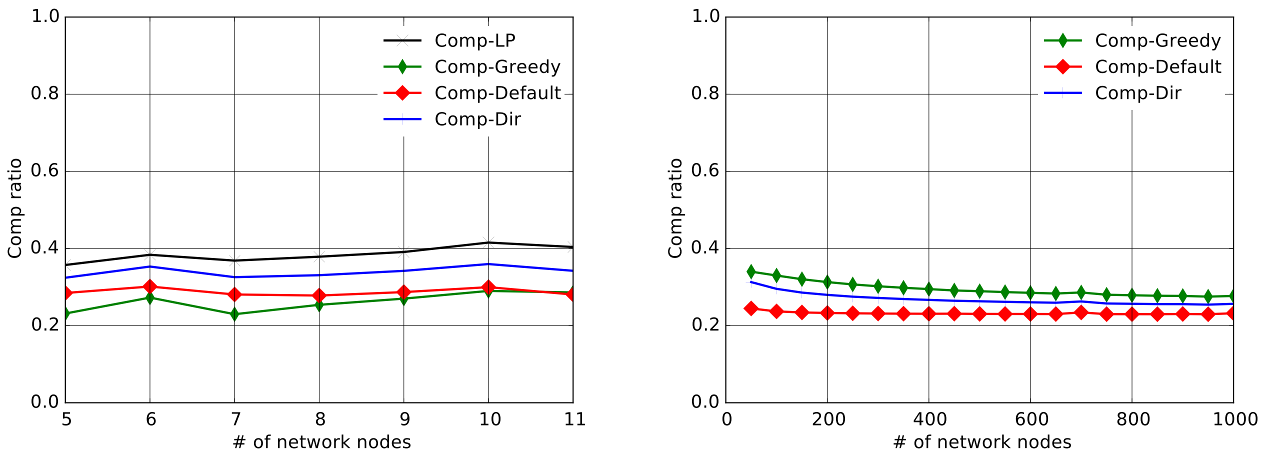


FIGURE 2: Compression ratio as a function of the number of network nodes (that is the number of sources and destinations) for the four compression methods.

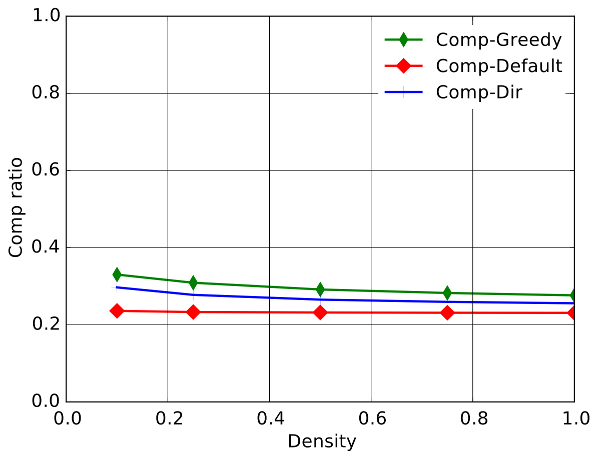


FIGURE 3: Compression ratio as a function of the forwarding tables density for the four compression methods.

tables. In Figure 1(b), we only compare the two heuristics and Comp-Default on tables with $n = 450$ sources/destinations and a density of 0.5. First, we notice that the two heuristics Comp-Greedy and Comp-Direction obtain the best results. However, the Comp-Default solution is not far behind with a ratio between 49% and 11% for the random tables. We will see later that the difference is significantly higher for real network tables. Comp-Greedy behaves better than Comp-Direction, with a compression ratio between 55% and 16% to be compared to a compression ratio between 52% and 14% for Comp-Direction.

Impact of the parameters. The compression ratio is very sensitive to the number of ports, see Figure 1. The compression ratio varies from 55% to 18% when the number of ports varies from 2 to 9 for a random matrice with around 100,000 rules. Similar results are observed for small tables with variations from 70% to 35%. *We observe higher compression ratio for smaller numbers of ports.* This is expected as, for example, the impact of setting a default port is higher when the number of ports is lower. For two ports, using a default port saves at least 50% of the rules.

Conversely, the *density and the size (number of network nodes) of the forwarding tables do not have an important impact on the compression ratio.* For the experiments in Figures 2, the compression ratio varies only by a few percents when the number of network nodes increases from 5 to 11, and then from 50 to 1000; and similarly, when the density goes from 0.1 to 1, even if it represents a 10-fold increase of the number of rules in the table (Figure 3). However, density and size of the forwarding tables have an impact on the compression time as discussed below.

Compression time. We study the time to compress forwarding tables. This time depends mostly on the number of entries in the forwarding table, as presented in Figure 4. The compression time using linear programming (Comp-LP) is a lot higher than the one using heuristic algorithm: around 1000 s for only 125 rules, when it takes a lot less than 1 ms for the heuristics. We thus had to present the results for Comp-LP independently in Figure 4(a) with a different log-scale ($[0, 10^7]$), compared to ($[0, 10^4]$) for Figure 4(b). We observe that the compressing time of Comp-LP increases exponentially with the number of rules. It reaches the limit of one hour we had set for tables with slightly more than 150 rules. Note that a network with 10 nodes cannot have more than 90 entries in a routing table (in the extreme case of one central node seeing all the possible flows). Thus, we know that we can use Comp-LP for networks with a number of nodes reaching 10 as all traffic usually is not routed through a single node. In fact, we show in Section 7, that LP runs on the SNDlib Atlanta network with 15 nodes, but that it is not usable for larger networks.

On the contrary, the *compression time of the heuristic algorithms is very low* and does not increase exponentially, but linearly with the number of rules. A large network with 100 nodes cannot have more than 10,000 entries in a routing table. A forwarding table of this size is compressed in less than 10 ms (around 10 ms for Comp-Greedy, 1 ms for Comp-Direction, and less than 1 ms for Comp-Default). It is even possible to compress a routing table of size 1M rules (for a network of more than a thousand nodes) in around a 1s for Comp-Greedy and less than 10 ms for Comp-Direction and Comp-Default. The heuristic algorithms for compression can thus be used for very large networks and have a very low execution time.

6.2. Network tables

We now compare the solutions on tables from routing on backbone networks using the routing and compression module presented in Section 5.1 and 5.2. We use four of the SNDlib instances shown in Figure 5:

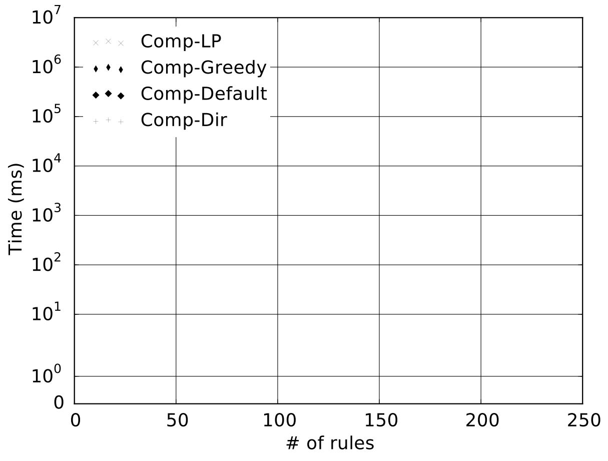
- atlanta network with 15 nodes and 44 directed links,
- germany50 network with 50 nodes and 176 directed links,
- zib54 network with 54 nodes and 216 directed links, and
- ta2 network with 81 nodes and 162 directed links.

For each network, we compute a routing of all demands without considering a limit on the number of rules. We then extract the forwarding tables for all routers. We then compress each of them with the different compression solutions. Since the ILP does not scale, we only compare it with the other solutions on the atlanta network, see Figure 6(a). On the other three networks, we compare the EARC-H-Direction, EARC-H-Greedy and EARC-H-Default solutions, see respectively Figures 6(b),(c)(d) for germany50, ta2, and zib54 networks.

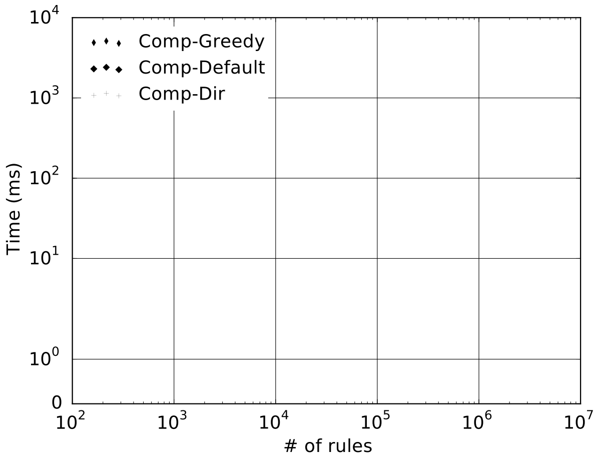
Compression Rates. The first global observation is that the solutions achieve *higher compression rates for network tables than for random tables*, with median values *around 80% for all networks.* This shows the efficiency of the algorithms for practical cases. The explanation of this phenomenon is that real network tables have a larger number of repeating ports traffic originating from a source or going to a destination, than random matrices.

We remark that some tables show a compression ratio near 100% for all solutions for zib54 and ta2. These tables correspond to the two routers with only one outgoing port (the two routers in black in Figure 5). Thus, only the default port can be used to route all the demands.

Comparison of the solutions. In the atlanta network, we see that the difference of efficiency between the heuristics, Comp-Direction, and Comp-Greedy,

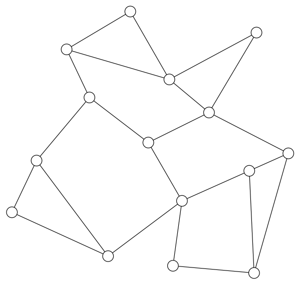


(a) Linear Program

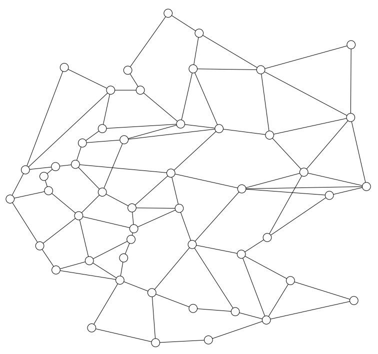


(b) Heuristic Algorithms

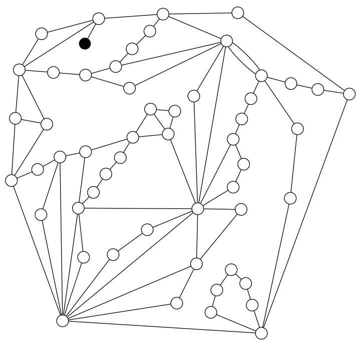
FIGURE 4: Compression times of forwarding tables as a function of the number of rules in the tables for four methods of compression (two different scales for Time).



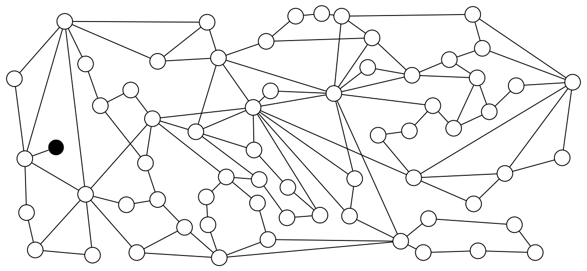
(a) atlanta



(b) germany50



(c) zib54

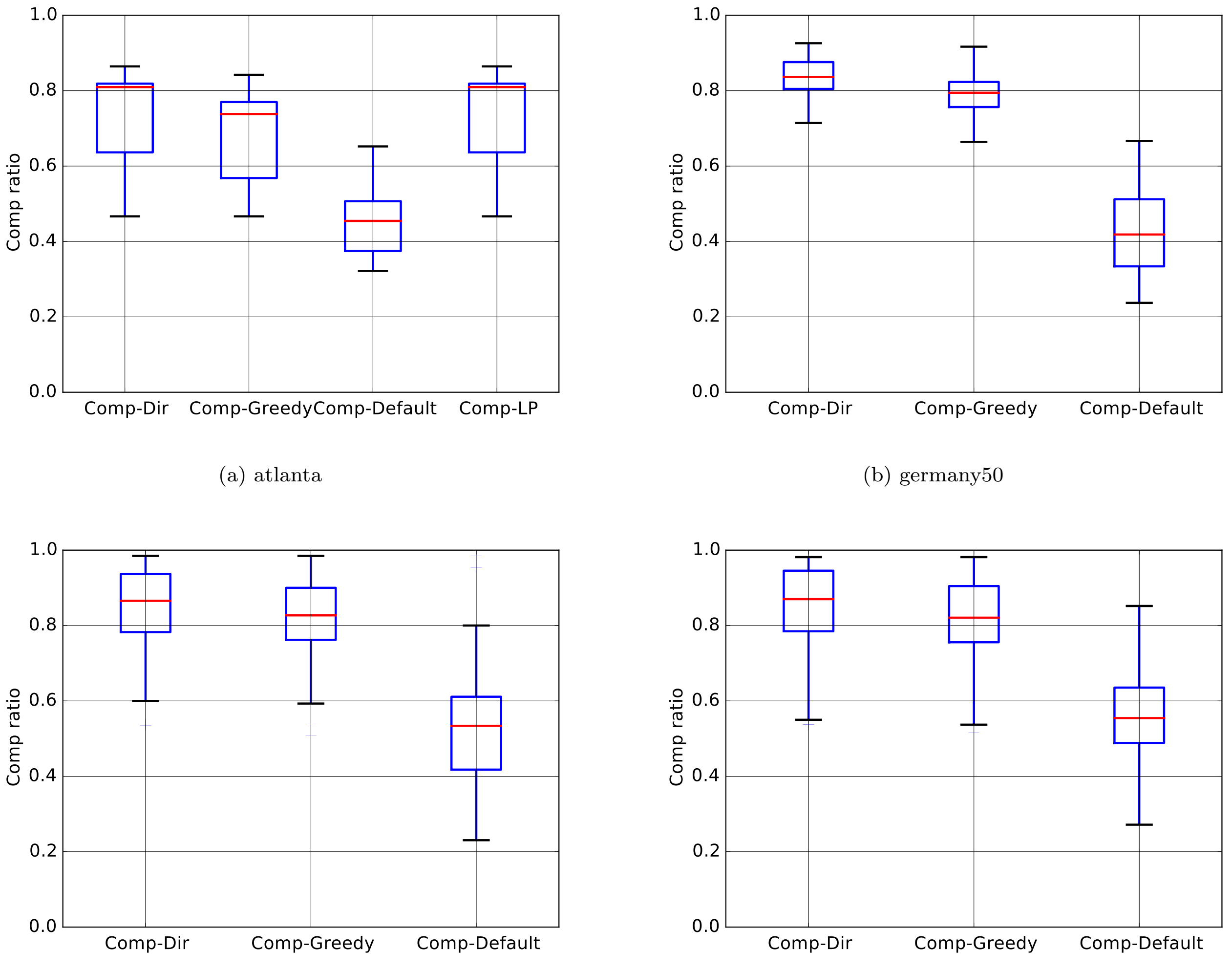


(d) ta2

FIGURE 5: The four SNDlib topologies used. Each edge corresponds to two directional links.

and the linear program for compression, Comp-LP, is smaller than in the case of random tables. The compression rate of Comp-Direction is almost the same as the one of Comp-LP, with a median ratio of 81%. *As real network tables are easier to compress than random tables*, we thus can suppose that the results of the heuristics on larger networks should be good. And, in fact, we obtain very high compression rates: the median is 83% for germany, 86% for ta2 and zib54.

Last, we observe that the *difference between the two levels of compression is more significant for real network tables than for random tables.* The median ratios of the Comp-Default solution are about 30% lower than the one from the Comp-Greedy heuristics. This shows the



(a) atlanta (b) germany50

(c) ta2 (d) zib54

FIGURE 6: Compression ratio of the three different heuristics on 4 different SNDlib topologies.

importance of considering multi-field compression.

The two heuristics using multi-field compression, *EARC-H-Direction and EARC-H-Greedy, show similar results on all networks.* While the Comp-Greedy heuristic provides better compression ratios on random tables, *the advantage for real network tables is for the Comp-Direction heuristic*: the median ratio is 4% higher for germany50, ta2, and zib54, and 8% for atlanta. We use both heuristics in the simulations of next section in which we obtain results for the EARC problem on practical network instances.

7. ENERGY SAVINGS

In this section, we study the energy saved over multiple periods of time and the four following networks: atlanta, germany50, ta2, and zib54. We compare the results obtained for the different solutions proposed to solve the EARC problem, the EAR problem without compression and classical routing (CR) without energy.

For the power parameters, we look at the powerlib database [38] that collects representative data for major network devices such as routers, switches, transponders. In this database, the scope of the values for the maximal power is huge, going, as an example, from 10W to 9000W for IP router components. Therefore, in order to present results that do not rely on a specific equipment from a specific vendor, we choose for the parameters of the power model a classical On-Off power model. Several other papers are dealing with this same power model, and among others, we can cite the very well-known and most-cited paper in this area: [3].

The different solutions are summarized in Table 3. Unless specified, the limit of the forwarding table is 750 rules. The number corresponds to the capacity of a NEC PF5820 switch. We considered a typical daily pattern of traffic as shown in Figure 7. Data come from a typical France Telecom link. For each network considered, we rescale the traffic based on the traffic matrices provided by SNDib. We then divide the

Compression kind	Name	Short name in figures	Routing	Energy	Compression algo
default port	EARC-LP-Default		LP (Sec. 4.1)		
multi-field	EARC-LP-Multi		LP (Sec. 4.2)		
default port	EARC-H-Default	EARC-Default	Heur		Opt. Comp-Default (Sec. 5.1.1)
multi-field	EARC-H-LP		Heur		LP Comp-LP (Sec. 5.1.2)
multi-field	EARC-H-Greedy	EARC-Greedy	Heur		Heur Comp-Greedy (Sec. 5.1.3)
multi-field	EARC-H-Direction	EARC-Dir	Heur		Heur Comp-Direction (Sec. 5.1.4)
none	EAR		yes	yes	none (but no limit on the number of rules)
none	EAR-with-limit		yes	yes	none (with limit on the number of rules)
none	Classic Routing	CR	yes	no	none (but no limit on the number of rules)

TABLE 3: Names of the solutions to solve the EARC problem (and of the EAR problem without compression for comparison).

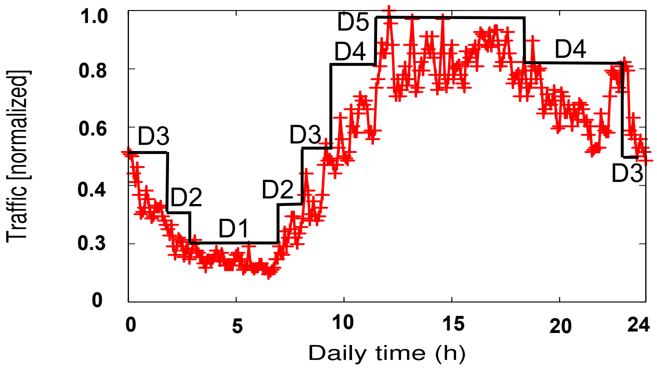


FIGURE 7: Daily traffic in multi-period

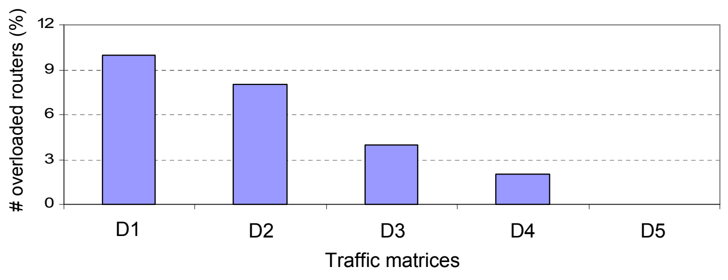
day into five periods, with different levels of traffic as shown in Figure 7. D1 represents the off-peak hours with the least amount of traffic on the network and D5 the peak hours. We choose a small number of periods as network operators prefer to carry out as few as possible changes of configurations of their network equipments to minimize the chance of introducing errors or producing routing instability. Moreover, most of the energy savings can be achieved with a very small number of configurations, see for example [39]. Energy savings is computed as the number of links to sleep divided by the total number of links of the network ($|E|$).

The need for more space In Figure 8, we show the number of overloaded routers (with more than 750 installed rules) when applying the heuristic proposed in [10] for Energy Aware Routing. This EAR heuristic does not take into account the table size constraint. As a result, we see that for almost every traffic patterns (except for D5 on germany50), an EAR needs more than 750 rules to be deployed. In germany50, up to 10% of the devices are overloaded. For zib54, this number goes up to 11% and 16% for ta2. This confirms that in order to be able to deploy energy policies on an SDN, the table size problem needs to be resolved.

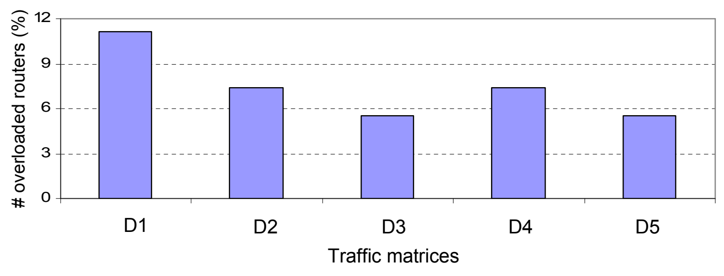
Optimal vs. Heuristic solution We compare for a small network, atlanta (15 links and 44 links), the solutions using linear programming and heuristic algorithms. We considered solutions for different rule capacities on routers: 100, 750 and 2000 rules.

Both linear programs, LP-Default and LP-Multi, proposed in Section 4 can be run on the atlanta network (but not on larger networks such as germany50, zib54 and ta2). As expected, LP-Multi, which solves the problem using more complex wildcards, has a longer execution time as it has more variables: around 8 min for 750 and 2000 rule capacities, to be compared with 23 s and 33 s for EARC-LP-Default. For 100 rule capacities, we observe a sharp increase as both ILPs take around 11 minutes to find the optimal solution. Both LPs find the same optimal solution, with a savings of 52.27%. This is due to the fact that Atlanta is a small network with nodes of small degrees. The need for compression is not high, and both levels of compression achieve the same results.

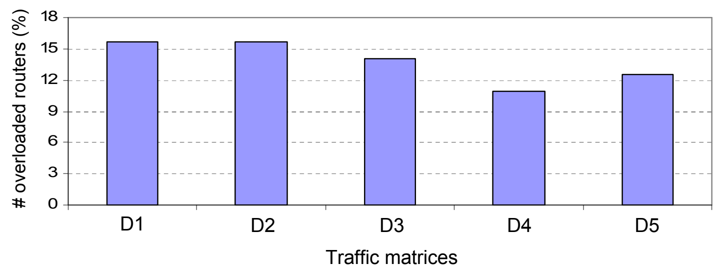
We ran two heuristic algorithms with two different compression modules proposed in Section 5.1, EARC-H-Direction with the Comp-Direction heuristic and EARC-H-LP, which solves optimally the compression problem each time a table has to be compressed when flows are routed. Both heuristics provide solutions of the same values, 40.91% of energy savings. However, the computation times are a lot higher for EARC-H-LP: more than 30 s compared to 14 ms for the heuristic. The computation time will prevent us from using EARC-H-LP on larger networks. As a matter of fact, we recall the compression time of tables using the LP increases exponentially, see Figure 4. However, we observe that both solutions provide the same level of energy savings



(a) Germany50 network



(b) Zib54 network



(c) Ta2 network

FIGURE 8: Number of overloaded routers in three networks with unlimited rule-space algorithm

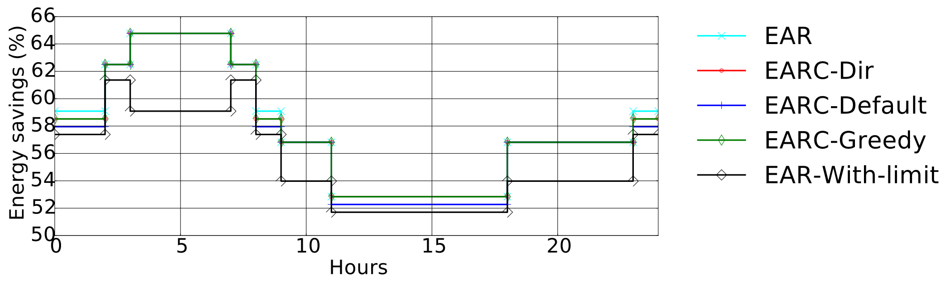
TABLE 4: Energy savings (in %) and computation times (in millisecond) for the ILP and the heuristics on the atlanta network

Rule capacity	EARC-LP-Default		EARC-LP-Multi		EARC-H-Direction		EARC-H-LP	
	savings	time	savings	time	savings	time	savings	time
100	52.27	641 940	52.27	694 302	40.91	∼ 14	40.91	3381
750	52.27	33 830	52.27	486 759	40.91	∼ 14	40.91	3311
2000	52.27	23 640	52.27	487 386	40.91	∼ 14	40.91	3300

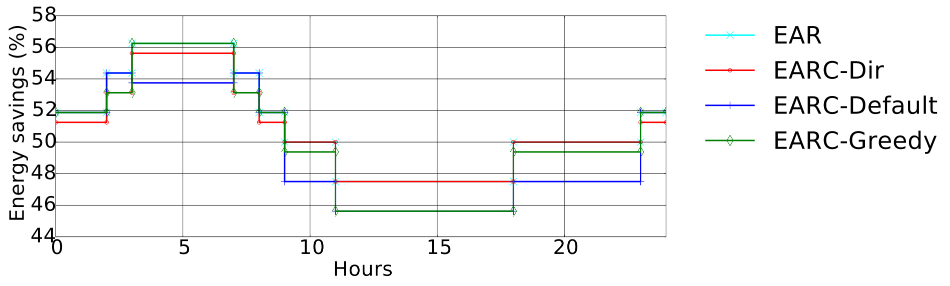
for the three rule capacities. The EARC-H-Direction heuristic is very efficient, and we use it to get solutions for larger networks. In particular, the running times for the whole set of demands is around 2 seconds for Germany50 and zib54 (between 1450 ms and 2785 ms), and between 3753 ms and 5921 ms for ta2 network. Therefore, the time to find a route for a single demand is quite small (0.97ms for the worst case).

Energy savings during the day In Figure 9, we compare the multiple solutions proposed for the compression module. We also check the possibility of an SDN routing without compression (corresponding to a simple *EAR*). The ILP is not considered in the comparison as the networks are too big to be optimally resolved in an acceptable time.

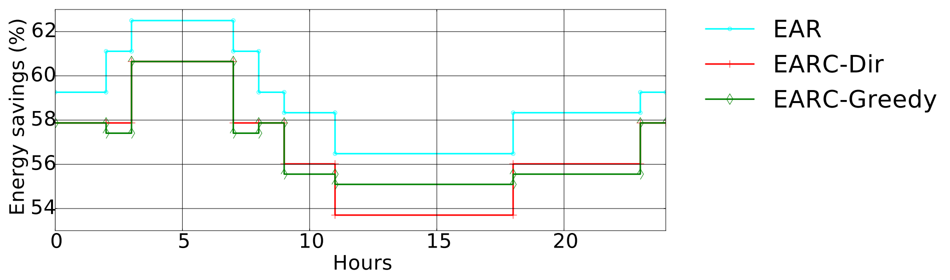
Importance of compression. First, we see that, as the networks grow in size, not all heuristics give a valid SDN routing. No compression is needed to find a valid SDN routing on germany50. However, *it is impossible to find a routing satisfying the capacity constraint for zib54 and ta2 without using a compression algorithm.*



(a) germany50



(b) zib54



(c) ta2

FIGURE 9: Energy savings of the different heuristics during the day with a limit of 750 rules.

Moreover, *multi-field compression should be used to find a valid routing for ta2.* Indeed, it is impossible to find a valid routing for ta2 while using only default port compression.

Results of the heuristics. On germany50, all heuristics give similar results between 52% for the peak hours and up to 65% during the night. They are all small within a margin of about 2% from one another. The EARC-H-Greedy and EARC-H-Direction heuristics show the best results and no compression gives the worst ones in all periods.

For the zib54 network, the difference between the heuristics is more visible. Between 46% and 56% is saved during the day. Once again, either the EARC-H-Greedy or EARC-H-Direction heuristics gives the best results depending on the periods. The only exception is during the D2 periods, where the EARC-H-Default compression shut about 1% more links than the other two heuristics.

Finally, in the ta2 network, the EARC-H-Greedy heuristic saves slightly more energy than the EARC-H-Direction one as the former saves almost 2% more than the latter.

The amount of saved energy by the heuristics for each network is different. The explanation is that the order in which each link is shut down depends on its charge. A small change in the routing thus can affect the total energy saved.

EAR vs. EARC. We compare the results of the proposed solutions with the one of the classic EAR approach in which no limit on the number of rules is considered. We show that *by using an efficient way to route demands and compress forwarding tables, it is possible to save almost as much power consumption as the EAR approach* (curve named *No Limit* in Figure 9). Indeed, we see that for the zib54 network, we succeeded to save the same amount of energy when using the best of all solutions. The solution EARC-H-Direction alone is very close to the EAR one. Only half a percent of energy is lost for some periods of time. On germany50, the results of the heuristics are almost as good. For some periods of time, no solutions can do as well as EAR, but the difference again is only of half a percent. In general, the results of EARC-H-Greedy are withing 1% of the one of EAR. For the network ta2, the difference between EAR and our solutions is higher but stays with 2%.

Path lengths As we shut down links, we remove some shortest paths in the network, and thus raise the minimum delay between nodes. To study this effect, we look at the length of the paths in our EAR solutions and compare it to a routing obtained not using the energy saving module. For these comparisons, we use the Direction heuristic.

In Figure 10, we show the distribution of the stretch ratio of the path used in EARC-H-Direction compared to a classic routing (CR). The first observation is that the behavior is similar for the three topologies: the median stretch is about 2 in the off-peak hour period (corresponding to the demand D1) and decreases to about 1.3 in the peak hours (demand D5). The explanation is that, as expected, in the off-peak hours, a large number of links can be switched off, and the paths are the longest. For larger demands, more links are on, and the stretch decreases.

Note that the median value is not very high. However, the third quartile value of the off-peak hours is quite high: 7, 6 and 5.25 for germany50, zib54, and ta2, respectively. These values are mostly due to paths of small lengths stretched all the way through the network to attain their destination (corresponding for example to nodes linked by a switched-off edge). Nevertheless, we show below that these somehow large values of stretch do not cause a problem of too large delays on the networks.

Delays In Figures 11, 12 and 13, we show the delay of the paths in the three networks, for both the classical routing and an EARC solution (EARC-H-Direction). We consider an optical network in which the delay is proportional to the distance [40], and we used the distances given by the geographical coordinates in SNDlib for the germany50 network. We got an average value of 1.8 ms per link. Since the coordinates are not given for the other two topologies, we used the same average value for zib54 and ta2.
The delays for the classical routing are similar for the three networks with a median of 8 ms and a maximum of 15 ms during all periods. For the EARC solution, the values are much higher. Larger delays are shown during the off-peak hours as expected. The germany50 network shows the largest delays among the three topologies. The explanation is that more energy can be saved for this network. Its median delay is between 11 ms and 16 ms, and the maximum delay is below 50 ms. The delay on the two larger networks is slightly less impacted as fewer links can be turned off. The maximum delay observed on zib54 and ta2 is about 40 ms and the medians fluctuate between 14 ms and 9 ms for zib54 and 14 ms and 10 ms for ta2.

Note that the *maximum delay observed is always below* 50 ms. This is an important fact, as this value is often chosen by Service Level Agreements (SLAs) as the maximum allowed delay for a route in a network [41]. Thus, even if new routes computed by our algorithms may sometimes display a high value of stretch, this will not be a problem for network operators.

Link load When we turn off links, we aggregate the flows on the remaining links. The load of them is thus increased. In Figure 14, we *compare the link load of all network links (switched off and switched on) for energy aware routing and for classical routing.* In Figure 15, cumulative distributions are given considering only the switched on links. Results are provided only for off-peak traffic (D1) and rush hour traffic (D5) in the first figure for clarity reason, while all five demand matrices are considered in the other.

The first observation is the percentage of links with a null load (switched off links), e.g., for germany50 62% with the demand D1 and 54% with D5. The load on the remaining links is highly increased: for germany50 again, we see that no link has a load higher than 15% for the CR for the D1 (higher than 50 % for D5) , when 45% of the links have a load higher than this value for EARC (40% for D5). Similarly, for zib54 and ta2, more than 80% of the links have a load smaller than 10% for D1 for CR and of 30% for D5, when more than 50% of the switched on links have a load higher than 50% for EARC.

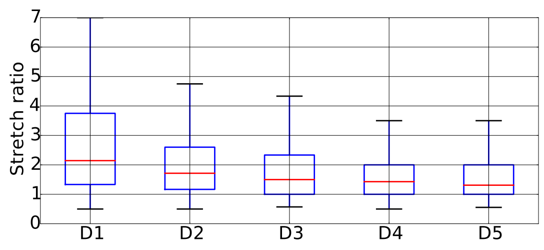
Note that, in the germany50 network, there is a very notable difference between the D1 and D5 period. In the off-peak hours, only 30% of the switched-on links are above 75% compared to 52% in the peak hours. In the other two networks, the difference between periods as the difference as the range of energy savings is smaller. For zib54, in the off-peak hours, the maximum link utilization is 86% and the minimum is 2% while in the peak hours, 19% of the links are above this usage and the minimum is 6%.

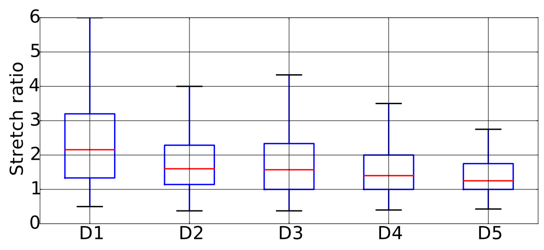
8. DISCUSSION

8.1. Generalization to Larger Number of Fields

In fact, our method can be generalized to any number of fields. Indeed:

1. The LPs can be generalized to a larger number of fields. With the current efficiencies of both state of the art linear solvers such as Cplex and computers, such LPs would run only for 2 fields. This is why we use this particular (simpler case) as example of our method in the paper.
2. The compression algorithm can itself be generalized to any number of fields as explained in [42]. We used an algorithm giving a 3-approximation algorithm when considering 2 fields. In fact, there exists a $2^f - 1$ approximation algorithm, where


(a) germany50


(b) zib54

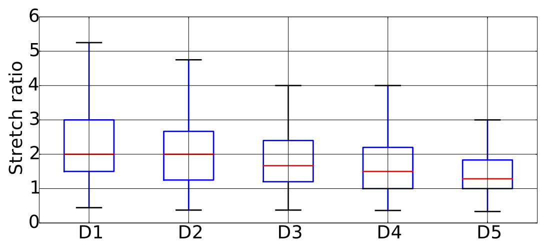
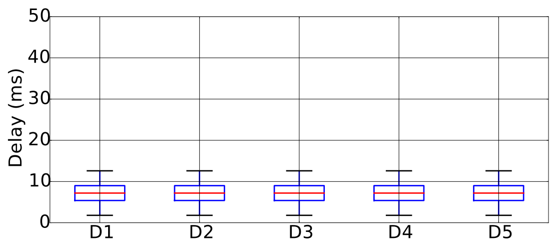

(c) ta2

FIGURE 10: Stretch ratio of the paths given by a EARC solution (EARC-H-Direction) compared to the one given by a classic routing (without energy savings) on the germany50 network with different traffic matrices.


(a) Classic Routing

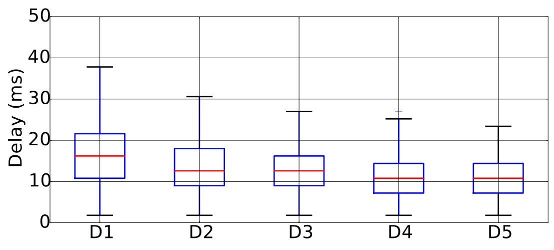

(b) EARC

FIGURE 11: Average delay by path on the germany50 network

f is the number of fields. The general idea is to compress on each possible subsets of fields using a greedy algorithm, and then select the best compression over all subset of fields.

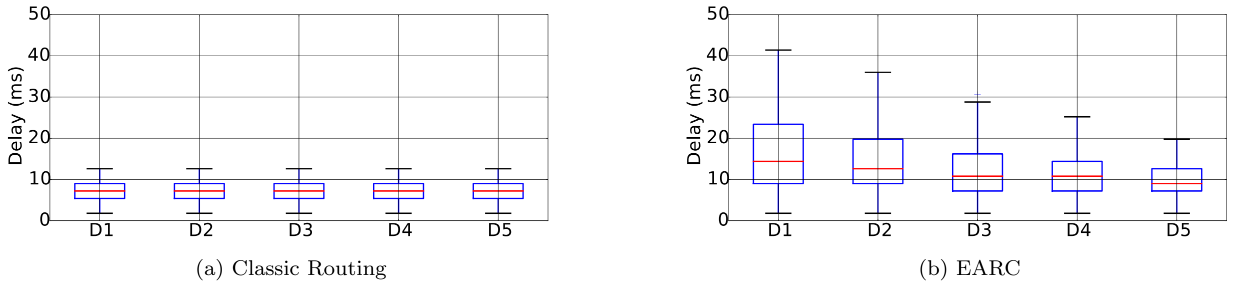
8.2. Prefix Aggregation

In this paper, we performed exact matching for the rules, and put wildcards on the whole source or destination field. Note that IP prefix aggregation can be added to our model after having compressed the table. It will allow to reduce more the number of entries. This second-step compression would be applied on the entries that contain no wildcard, but only on source or destination not both.

9. CONCLUSION

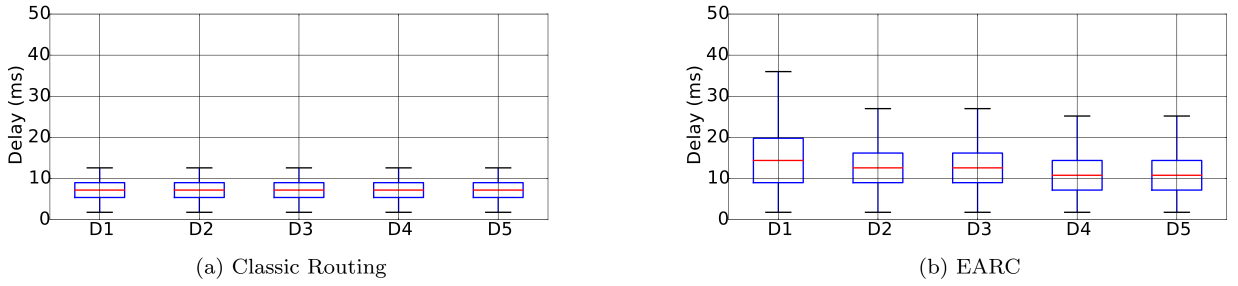
To our best knowledge, this is the first work considering rule space constraints of OpenFlow switch in energy-aware routing (EAR). We argue that, in addition to capacity constraint, the rule space is also important as it can change the routing solution and affects QoS. We proposed solutions using forwarding table compression, defining the problem of energy-aware routing with compression (EARC) for SDN networks. We succeed in modeling the problem using Integer Linear Programs, even for complex compression for which a flow may be routed according to two packet header fields. We also provide efficient heuristic algorithms for large networks.

Based on simulations with real traffic traces, we show that using wildcard rules our smart rule allocation can *achieve high energy efficiency for a backbone network while respecting both the capacity and the rule space constraints. Thanks to forwarding table compression, the energy savings are almost as high as in the case of classic EAR without a limit on the number of forwarding rules.* We also evaluate the impact of the proposed solutions on path delay. We show that, if the delay is inevitably increased, the *maximum delay*



(a) Classic Routing (b) EARC

FIGURE 12: Average delay by path on the zib54 network.



(a) Classic Routing (b) EARC

FIGURE 13: Average delay by path on the ta2 network.

always stays below typical values given by Service Level Agreements.

REFERENCES

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

[2] Chiaraviglio, L., Mellia, M., and Neri, F. (2012) Minimizing isp network energy cost: formulation and solutions. *IEEE/ACM Transactions on Networking (TON)*, **20**, 463–476.

[3] Chabarek, J., Sommers, J., Barford, P., Estan, C., Tsiang, D., and Wright, S. (2008) Power awareness in network design and routing. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, Phoenix, AZ, USA, April, pp. 457–465. IEEE.

[4] Mahadevan, P., Sharma, P., and Banerjee, S. (2009) "A Power Benchmarking Framework for Network Devices". *IFIP NETWORKING*, Aachen, Germany, may, pp. 795–808. Springer Berlin Heidelberg.

[5] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008) Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, **38**, 69–74.

[6] Hp 2920 switch series.

[7] Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., and Banerjee, S. (2011) Devoflow: Scaling flow management for high-performance networks. *SIGCOMM Comput. Commun. Rev.*, **41**, 254–265.

[8] Giroire, F., Moulierac, J., and Phan, K. (2014) Optimizing rule placement in software-defined networks for energy-aware routing. *IEEE Global Communications Conference (GLOBECOM)*, Austin, United States, December, pp. 2523–2529. IEEE.

[9] Rifai, M., Huin, N., Caillouet, C., Giroire, F., Lopez-Pacheco, D., Moulierac, J., and Urvoy-Keller, G. (2015) Too Many SDN Rules? Compress Them with MINNIE. *2015 IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, USA, Dec, pp. 1–7. IEEE.

[10] Giroire, F., Mazauric, D., and Moulierac, J. (2012) Energy efficient routing by switching-off network interfaces. In Kaabouch, N. and Hu, W.-C. (eds.), *Energy-Aware Systems and Networking for Sustainable Initiatives*, chapter 10, June, pp. 207–236. IGI Global, Hershey, Pennsylvania (USA).

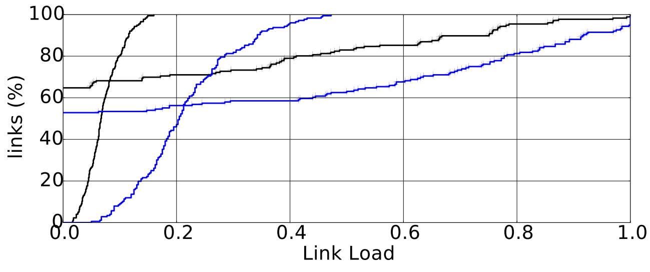
[11] Orlowski, S., Pióro, M., Tomaszewski, A., and Wessäly, R. (2010) SNDlib 1.0–Survivable Network Design Library. *Networks*, **55**, 276–286.

[12] Gupta, M. and Singh, S. (2003) Greening of the internet. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Karlsruhe, Germany, August, pp. 19–26. ACM.

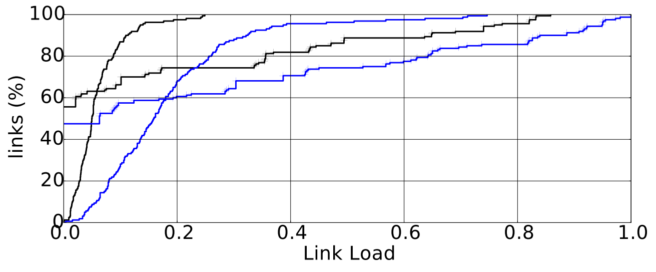
[13] Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., and McKeown, N. (2010) Elastictree: Saving energy in data center networks. *NSDI*, San Jose, CA, USA, April, 28-30, pp. 249–264. USENIX Association.

[14] Wang, X., Yao, Y., Wang, X., Lu, K., and Cao, Q. (2012) Carpo: Correlation-aware power optimization in data center networks. *INFOCOM, 2012 Proceedings IEEE*, Orlando, FL, USA, March, pp. 1125–1133. IEEE.

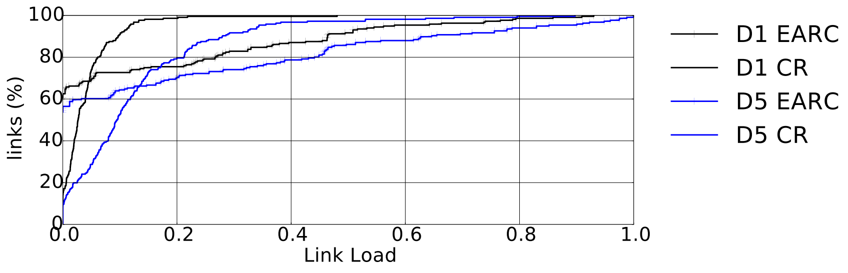
[15] Zhou, B., Zhang, F., Wang, L., Hou, C., Anta, A. F.,



(a) germany50

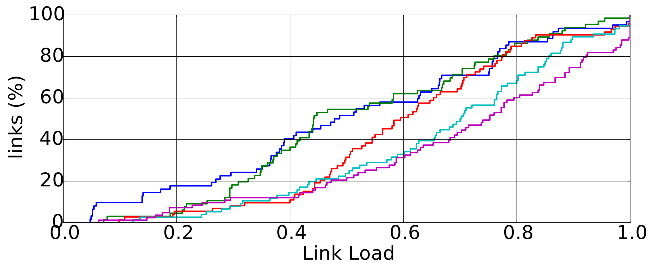


(b) zib54

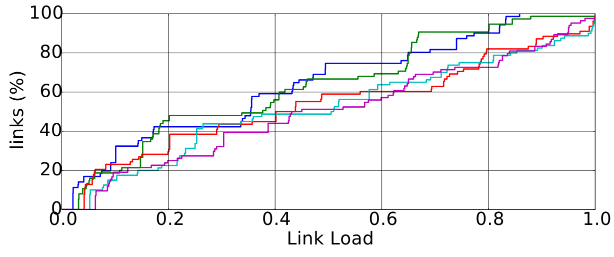


(c) ta2

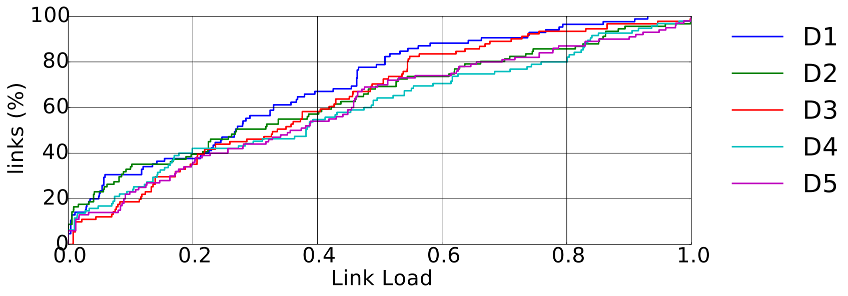
FIGURE 14: Comparison of the cumulative distribution function of the link load for Energy Aware Routing with Compression (EARC) and classic routing (CR). Results for off peak traffic (D1) and rush hour traffic (D5) are provided.



(a) germany50



(b) zib54



(c) ta2

FIGURE 15: Cumulative distribution function of the link load of the *switched on links* using EARC for the five demand matrices (D1 is off peak traffic and D5 is rush hour traffic.)

Vasilakos, A. V., Wang, Y., Wu, J., and Liu, Z. (2016) Hdeer: A distributed routing scheme for energy-efficient networking. *IEEE Journal on Selected Areas in Communications*, **34**, 1713–1727.

[16] Wang, L., Zhang, F., Aroca, J. A., Vasilakos, A. V., Zheng, K., Hou, C., Li, D., and Liu, Z. (2014) Greendcn: A general framework for achieving energy efficiency in data center networks. *IEEE Journal on Selected Areas in Communications*, **32**, 4–15.

[17] Fernandez-Fernandez, A., Cervello-Pastor, C., and Ochoa-Aday, L. (2016) Achieving energy efficiency: An energy-aware approach in SDN. *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, USA, Dec, pp. 1–7. IEEE.

[18] Wang, R., Jiang, Z., Gao, S., Yang, W., Xia, Y., and Zhu, M. (2014) Energy-aware routing algorithms in software-defined networks. *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, Sydney, Australia, June, pp. 1–6. IEEE Computer Society.

[19] Özbek, B., Aydoğmuş, Y., Ulaş, A., Gorkemli, B., and Ulusoy, K. (2016) Energy aware routing and traffic management for software defined networks. *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, South Korea, June, pp. 73–77. IEEE.

[20] Awad, M. K., Rafique, Y., Alhadlaq, S., Hassoun, D., Alabdulhadi, A., and Thani, S. (2016) A greedy power-aware routing algorithm for software-defined networks. *2016 IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2016*, Limassol, Cyprus, December 12-14, pp. 268–273. IEEE.

[21] Assefa, B. G. and Ozkasap, O. (2015) State-of-the-art energy efficiency approaches in software defined networking. *ICN 2015*, **x**, 268.

[22] Casado, M., Freedman, M. J., Pettit, J., Luo, J., Gude, N., McKeown, N., and Shenker, S. (2009) Rethinking enterprise network control. *IEEE/ACM Trans. Netw.*, **17**, 1270–1283.

[23] Wang, R., Butnariu, D., and Rexford, J. (2011) OpenFlow-based Server Load Balancing Gone Wild. *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Boston, MA, USA, March, pp. 12–12. USENIX Association.

[24] Kanizo, Y., Hay, D., and Keslassy, I. (2013) Palette: Distributing tables in software-defined networks. *INFOCOM, 2013 Proceedings IEEE*, Turin, Italy, April, pp. 545–549. IEEE.

[25] Kang, N., Liu, Z., Rexford, J., and Walker, D. (2013) Optimizing the "one big switch" abstraction in software-defined networks. *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, New York, NY, USA, December CoNEXT '13, pp. 13–24. ACM.

[26] Nguyen, X.-N., Saucez, D., Barakat, C., and Turletti, T. (2015) OFFICER: A general Optimization Framework for OpenFlow Rule Allocation and Endpoint Policy Enforcement. *INFOCOM*, Kowloon, Hong Kong, April, pp. 478–486. IEEE.

[27] Cohen, R., Lewin-Eytan, L., Naor, J., and Raz, D. (2014) On the effect of forwarding table size on sdn network utilization. *INFOCOM*, Toronto, Canada, April, pp. 1734–1742. IEEE.

[28] Hu, S., Chen, K., Wu, H., Bai, W., Lan, C., Wang, H., Zhao, H., and Guo, C. (2015) Explicit path control in commodity data centers: Design and applications. *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, CA, USA, May NSDI'15, pp. 15–28. USENIX Association.

[29] Braun, W. and Menth, M. (2014) Wildcard compression of inter-domain routing tables for openflow-based software-defined networking. *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, Budapest, Hungary, Sept, pp. 25–30. IEEE Computer Society.

[30] Theobald, M., Nowick, S. M., and Wu, T. (1996) Espresso-hf: A heuristic hazard-free minimizer for two-level logic. *Proceedings of the 33rd Annual Design Automation Conference*, New York, NY, USA, June DAC '96, pp. 71–76. ACM.

[31] Rifai, M., Huin, N., Caillouet, C., Giroire, F., Moulierac, J., Pacheco, D. L., and Urvoy-Keller, G. (2017) Minnie: An SDN world with few compressed forwarding rules. *Computer Networks*, **121**, 185–207.

[32] Huin, N., Rifai, M., Giroire, F., Pacheco, D. L., Urvoy-Keller, G., and Moulierac, J. (2017) Bringing energy aware routing closer to reality with SDN hybrid networks. *2017 IEEE Global Communications Conference, GLOBECOM*, Singapore, December, pp. 1–7. IEEE.

[33] Saravanan, K. P., Carpente, P. M., and Ramirez, A. (2015) Exploring multiple sleep modes in on/off based energy efficient hpc networks. *2015 33rd IEEE International Conference on Computer Design (ICCD)*, New York, NY, USA , Oct, pp. 54–61. IEEE.

[34] Niccolini, L., Iannaccone, G., Ratnasamy, S., Chandrashekar, J., and Rizzo, L. (2012) Building a power-proportional software router. *Presented as part of the 2012 USENIX Annual Technical Conference*, Boston, MA, USA, June, pp. 89–100. USENIX Association.

[35] Idzikowski, F., Chiaraviglio, L., Cianfrani, A., Vizcaíno, J. L., Polverini, M., and Ye, Y. (2016) A survey on energy-aware design and operation of core networks. *IEEE Communications Surveys & Tutorials*, **18**.

[36] Giroire, F., Havet, F., and Moulierac, J. (2015) Compressing two-dimensional routing tables with order. *7th Network Optimization Conference (INOC)*, Varsaw, Poland, May, pp. 1–8. Electronic Notes in Discrete Mathematics.

[37] Giroire, F., Huin, N., Moulierac, J., and Phan, K. (2016) Energy-aware routing in software-defined networks with table compression (using wildcard rules). Research report 8897. Inria, Sophia Antipolis.

[38] Van Heddeghem, W., Idzikowski, F., Vereecken, W., Colle, D., Pickavet, M., and Demeester, P. (2012) Power consumption modeling in optical multilayer networks. *Photonic Network Communications*, **24**, 86–102.

[39] Araújo, J., Giroire, F., Moulierac, J., Liu, Y., and Modrzejewski, R. (2016) Energy efficient content distribution. *Computer Journal*, **59**, 192–207.

[40] Choi, B.-Y., Moon, S., Zhang, Z.-L., Papagiannaki, K., and Diot, C. (2007) Analysis of point-to-point packet delay in an operational network. *Computer networks*, **51**, 3812–3827.

[41] Giroire, F., Nucci, A., Taft, N., and Diot, C. (2003) Increasing the robustness of ip backbones in the absence of optical level protection. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, San Franciso, CA, USA, March, pp. 1–11. IEEE.

[42] Giroire, F., Havet, F., and Moulierac, J. (2018) On the complexity of compressing two dimensional routing tables with order. *Algorithmica*, **80**, 209–233.