Acta Cybernetica — *online–first paper version* — pages 1–17.

Integer Programming Based Optimization of Power Consumption for Data Center Networks

Gergely Kovásznai[a] and Mohammed Nsaif[b]

Abstract

With the quickly developing data centers in smart cities, reducing energy consumption and improving network performance, as well as economic benefits, are essential research topics. In particular, Data Center Networks do not always run at full capacity, which leads to significant energy consumption. This paper experiments with a range of optimization tools to find the optimal solutions for the Integer Linear Programming (ILP) model of network power consumption. The study reports on experiments under three communication patterns (near, long, and random), measuring runtime and memory consumption in order to evaluate the performance of different ILP solvers. While the results show that, for near traffic pattern, most of the tools rapidly converge to the optimal solution, CP-SAT provides the most stable performance and outperforms the other solvers for the long traffic pattern. On the other hand, for random traffic pattern, Gurobi can be considered to be the best choice, since it is able to solve all the benchmark instances under the time limit and finds solutions faster by 1 or 2 orders of magnitude than the other solvers do.

Keywords: integer programming, optimization, power consumption, Data Center Network, solvers

1 Introduction

Data Centers Networks (DCNs) are becoming increasingly significant in daily routine because of the fast growth of modern information technologies such as the Internet of Things, Big Data, Cloud Computing, and Mobile Sensing Networks [17, 16]. DCNs aim for high reliability and stability with several redundant links and enough capacity. The network devices usually work at full capacity 24 hours a day, consuming much energy. However, network equipment is underutilized most of the time, causing extremely low network energy efficiency. As a result, this problem attracts

[a]Department of Computational Science, Eszterházy Károly Catholic University, Eger, Hungary, E-mail: kovasznai.gergely@uni-eszterhazy.hu, ORCID: 0000-0001-8455-0218

[b]Department of Information Technology, University of Debrecen, Hungary, E-mail: mohammed.nsaif@mailbox.unideb.hu, mohammed.nsaif@uokufa.edu.iq, ORCID: 0000-0001-6768-4644

DOI: 10.14232/actacyb.299115

many researchers to figure out techniques that save energy while maintaining network performance. For the current DCNs, there are two techniques to save power consumption: device sleep [1, 7] and adaptive link rate [19]. The device sleeping technique is based on turning on/off the switches and links that are under a utility value in a dynamic manner. Whereas the adaptive link rate technique is based on assigning the fair bandwidth value for each flow passing through the links to control the ports' clock rate (operating frequency), leading to lower power consumption.

Because switches are considered to be one of the major devices in DCNs, this paper focuses on devices' sleep techniques to save power. This technique appeared in 2010; Heller et al. [7] introduced three types of optimizers to save power consumption: formal model, greedy bin-packing, and topology-aware heuristic. The topology-aware heuristic shows good results in saving up to 50%. It is based on elastic topology, which increases/decreases the size of the topology according to the size of the traffic.

Our current paper builds upon our last contribution in [14], which proposed an Integer Linear Programming (ILP) model for traffic and energy-aware routing in Software-Defined Networking (SDN) based on link utility information, and could decide many pathways simultaneously. Additionally, it proposed a link utility-based heuristic algorithm called FPLF, which had the ability to save energy up to 10% when the traffic load is high (e.g., during rush hour) and 63.3% when the load is low (e.g., at night). Our current paper aims to explore and examine other ILP solving tools that can solve convex and non-convex optimization problems, which we can use in real-time action to find an optimal solution for a large number of injected flows, instead of FPLF-heuristic solutions.

The rest of the paper is arranged as follows. In Section 2, we review recent papers and studies that use an ILP formulation to optimize power consumption. The power optimization problem for DCNs is explained in Section 3. Our ILP model is described in Section 3.1. Our experiments, benchmarks, ILP solving tools, our portfolio solver, and the experimental results are detailed in Section 4. Finally, we summarize all our key contributions and outline some future directions in Section 5.

2 Related Work

This section outlines the robustness and limitations of recent Integer Linear Programming (ILP) approaches that address the power consumption decrease challenge for network routing algorithms.

The authors in [7] developed three methods to calculate a minimum-power network subset; one of them uses a formal model. The objective function consists of two binary variables for each switch and link. The constraints represent Multi-Commodity Network Flow, Power Minimization, and Flow Split. At the same time, the model's input parameters include the traffic matrix, the switch power model, and the topology. The model outputs a subset of the original topology and per-flow route information. While the study focuses on the number of nodes that the model can manage in the topology, network performance is not taken into consideration.

In contrast, our current model calculates the minimum number of links that satisfy the traffic matrix, based on the utilization matrix.

The authors in [18] proposed a 0-1 ILP model to minimize the power of idle line cards and integrated chassis by switching them to sleep, under link utilization and packet delay constraints. Meanwhile, the study proposes two heuristic algorithms for the same purpose and reports on experiments executed in two scenarios: synthetic topology and real-life topology named CERNET. In the same context, another study in [2] proposes an ILP model with a multi-objective function to minimize the sum of the energy consumption of switches and links. The study limits the links' maximum and minimum utility to manage the trade-off between the power consumption and the network performance. Nevertheless, neither study presents any experimental result with the ILP model, and the algorithms were experimented outside of any DCN.

[11] proposed a data center scheduling algorithm called FLOWP, besides an ILP formula, with the aim of optimizing power consumption and Quality of Service (QoS). The formula considers a minimum threshold for the efficiency of links and switches. The results show that QoS is improved compared to the approach in [7]. However, similar to [18], the study does not show any experimental result with the ILP model. Experiments are conducted on a heuristic algorithm only.

Our contribution in [14] presents an ILP model that has the ability to manage multiple paths to save power consumption and to balance the load at overloaded times. The study experiments with the model using the optimization tool LINGO [10]. The results show that LINGO could not find the optimal solution in a reasonable time for high number of flows sent simultaneously. Our current study shows that more powerful optimization tools can find solutions in a reasonable time.

Finally, we mentioned that various ILP formulations have been proposed to address the traffic-aware energy consumption challenges. Nevertheless, in many of them, the results of optimal solutions do not scale to a large number of links, nodes, line cards, switches, or *flows* [14]. Some of these ILP formulations are designed for appointed DCN topologies, i.e., fat-tree and bicubic. On the other hand, some of them are independent of topology structures. All those facts encouraged the authors to explore a wide range of state-of-the-art optimization tools and to compare their experimental results in the current study.

3 Problem Statement and Proposed Solutions

Although the ILP model in [14] can calculate optimal multi-path ways and can manage the current status of the network, the model was solved by using the optimization toolkit LINGO, which was costly when the network size and number of flows were large. It took more than 160 minutes to accommodate only 120 bursts of flows in the topology (a fat-tree topology with $k = 4$) at one time. This fact motivated the authors to propose a heuristic routing algorithm called Fill Preferred Link First (FPLF) to find feasible solutions.

Figure 1 shows part of the results from [14]. Based on the results, the left figure

shows that LINGO cannot find a solution in reasonable time when sending more than 100 simultaneous flows, and the runtime dramatically increases. On the other hand, the right figure shows how the number of active links increases proportionally to the flows. Therefore, the authors of the current study think that it might be possible to find more powerful optimization tools that can find the optimal solution for a higher number of flows in reasonable time.

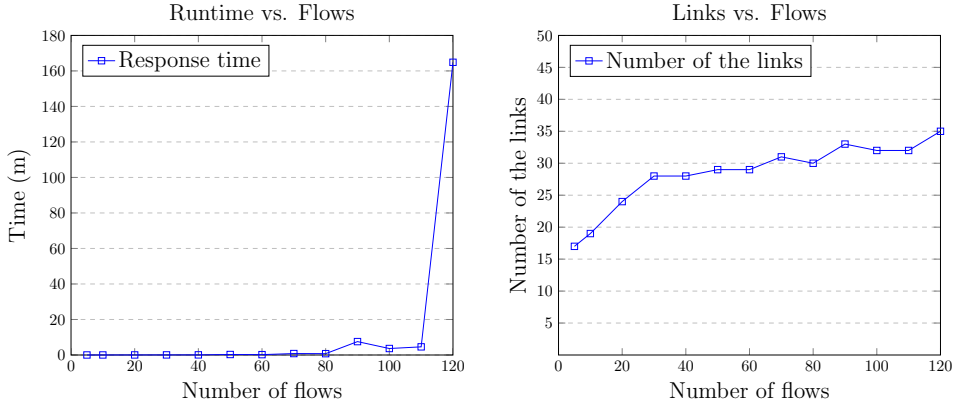

Figure 1: Correlation between the number of flows and the runtime of LINGO, and between the number of flows and the number of active links, respectively.

Our contributions in the current paper are summarized as follows:

- Developing NEO-DCN, our network optimizer tool for DCNs.
- Generating more realistic benchmarks to experiment with the proposed ILP model for three different traffic patterns.
- Evaluating the performance of several solvers with the ILP model on three different benchmarks.

3.1 DCN Models and Constraints

The DCN is modeled as an undirected graph $\mathrm{G} = (\mathbb{S}, \mathbb{E})$, where $\mathbb{S} = \{s_1, s_2, \ldots, s_n\}$ is a set of switches and $\mathbb{E} \subseteq \{e_{ij} \mid s_i, s_j \in \mathbb{S}\}$ is a set of links. The traffic is represented as a set of flows $\mathbb{F}$, where each flow $f = (f.S, f.D, \lambda_f) \in \mathbb{F}$ consists of a source $f.S \in \mathbb{S}$, a destination $f.D \in \mathbb{S}$ and a bit rate $\lambda_f \in \mathbb{N}$.

The power consumption of a DCN is based on the SDN network equipment $\mathbb{S}$ and $\mathbb{E}$. Therefore, the Network Power Consumption (NPC) model is directly related to the number of active switches and the number of links. The computation formula for NPC is shown in (1).

$$\mathrm{NPC} = P_{\mathrm{switch}} \sum_{s_i \in \mathbb{S}} B_i + P_{\mathrm{link}} \sum_{e_{ij} \in \mathbb{E}} L_{ij}. \tag{1}$$

B_i and L_{ij} denote the state of a corresponding switch and link, respectively, where the value 1 represents the active state, and 0 the passive state. The base power consumption of switches and links are denoted by P_{switch} and P_{link}.

Links and traffic correlation constraint: This constraint considers the correlation between the traffic volume and the links. Therefore, the constraint defines the relationship between the traffic volume T_{ij} and the link state L_{ij} to increase the utility of the link as much as possible.

Links and flows correlation constraint: This constraint represents the correlation between links and flows, such that a link should be active if and only if a flow passes through it.

Utility constraint: This constraint computes the utility of all the topology's links, and limits the link utility to less than or equal to the link's bandwidth BW_{ij}.

Path conservation constraint: This constraint installs the path from the source $f.S$ to the destination $f.D$ for each flow f.

Flow conservation constraint: This constraint guarantees for any flow f that the incoming and outgoing flows of the intermediate switches between the source $f.S$ and the destination $f.D$ should be equal, in order to avoid packet loss.

Network loop avoidance constraint: Since this model computes an acyclic graph in this context of routing, it is impossible to start at a switch s and to follow a directed path that returns to s. Thus, this constraint helps to avoid looping between switches.

3.2 ILP Formulation

This section describes how the DCN optimization model specified in Section 3.1 can be formulated as an ILP model, which computes the minimum number of links for a given traffic utilization, under the following conditions:

- The optimization model's parameters refer to a snapshot of the network state. This means that the model considers the case of the network state in a specific unit of time.
- The model starts with a standard multi-commodity flow problem. The constraints include flow conservation, link capacity, demand satisfaction, and the total number of active links.
- Splitting a single flow into packets across multiple links in the topology could save energy by increasing overall link utilization. However, due to varied path delays, reordered packets at the destination can degrade the performance. As

a result, we incorporate restrictions into our formulation based on the entire flow.

In this ILP model, all the variables are Boolean, i.e., they are restricted to the range $\{0, 1\}$. We will use the following Boolean variables:

- L_{ij} denotes if the link e_{ij} is active;
- $FR(f, i, j)$ denotes if the flow f passes through the link e_{ij}.

Additional integer constants are used in the model:

- BW_{ij} represents the bandwidth of the link e_{ij};
- T_{ij} represents the input traffic volume over the link e_{ij}.

The model in [14] employed only the second operand from (1), to minimize the number of the links as shown in (2).

$$\min \left(\sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij} \right). \tag{2}$$

The above objective function works against the following constrains:

Links and traffic correlation constraint:

$$\frac{T_{ij}}{BW_{ij}} \leq L_{ij}, \quad \forall e_{ij} \in \mathbb{E}, \tag{3}$$

expressing that the traffic volume must not exceed the bandwidth of a link.

Links and flows correlation constraint:

$$FR\,(f, i, j) \leq L_{ij}, \quad \forall f \in \mathbb{F},\ \forall e_{ij} \in \mathbb{E}, \tag{4}$$

meaning that flows should pass only through active links.

Utility constraint:

$$\sum_{f \in \mathbb{F}} \big(FR\,(f, i, j) + FR\,(f, j, i) \big) \cdot \lambda_f \leq BW_{ij} - \mathrm{T}_{ij}, \quad \forall e_{ij} \in \mathbb{E}, \tag{5}$$

where a flow's packet rate is counted according to the undirected nature of the network graph.

Path conservation constraint:

$$\sum_{i=1}^{n} FR\,(f, f.S, i) = 1, \quad \sum_{i=1}^{n} FR\,(f, i, f.D) = 1, \quad \forall f \in \mathbb{F}. \tag{6}$$

Flow conservation constraint:

$$\sum_{\substack{i=1 \\ i \neq f.S}}^{n} FR(f,i,j) = \sum_{\substack{i=1 \\ i \neq f.D}}^{n} FR(f,j,i), \quad \forall\ f \in \mathbb{F},\ j \in \mathbb{S}. \tag{7}$$

Network loop avoidance constraint:

$$FR(f,i,j) + FR(f,j,i) \leq 1, \quad \forall f \in \mathbb{F},\ i,j \in \mathbb{S}. \tag{8}$$

4 Implementation and Experiments

4.1 Benchmarks

The communication patterns affect performance and power consumption [9]. Based on the fact that the traffic in a data center swings between peak traffic (e.g., at daytime) and low traffic (e.g., at nighttime) [3], the traffic matrix for a DCN follows the sine wave in (9).

$$\text{Traffic Rate } = \frac{1}{2} \cdot \text{max rate} \cdot (1 + \sin(t)) \tag{9}$$

This paper explores three types of the sine-wave traffic matrix: near, long, and mixed (i.e., random). The benchmarks build upon what we described in Section 3.2. Each benchmark is a snapshot of the DCN at the time interval t_i , $1 \leq i \leq n$. This means that each benchmark captures the state of the traffic matrix at a specific time.

Near traffic pattern: The traffic is restricted between the servers that reside in the same PODs (Point of Delivery) of the topology, i.e., the servers that are connected through the edge layer switches only. The benchmarks aggregate the flows to a minimum number of links inside the same POD.

Long traffic pattern: The traffic is restricted between the servers that reside in different PODs of the topology, i.e., the servers that are connected through the edge, aggregation, and core layer switches. The model saves less power due to balancing the load between switches and using multiple paths to keep the QoS at an acceptable level, depending on the utility of links at that time.

Random traffic matrix: The traffic matrix for this pattern is a mixture of the above patterns, in order to explore how many links we can save with a random sine-wave pattern.

4.2 ILP Solving Tools

LINGO provides a collection of built-in solutions to handle a wide range of optimization problems. Unlike many modeling products, all of the LINGO solvers are

directly connected to the modeling environment. Instead of using slower intermediary files, this seamless connection enables LINGO to transmit the issue to the right solver immediately in memory. This direct connection also reduces the compatibility issues between the solver and the modeling language components. LINGO is supported by LINDO Systems Inc. [10]. It is free and available for students and interested researchers.

Google's OR-Tools [15] is an open-source toolkit for solving optimization problems in general. Via the Python package `ortools`, one can access several optimization tools, including the following ones. Gurobi [6] provides one of the most powerful commercial solvers for a wide range of optimization problems, including ILP problems, and it is free to use for academics and students. CP-SAT[1] is a constraint programming solver that uses SAT methods, and it is part of the OR-Tools package. SCIP [4] is one of the fastest non-commercial solvers for Mixed Integer Programming (MIP) and, also, an open-source framework for constraint integer programming. CBC [5] (Coin-or Branch and Cut) is an open-source MIP solver. We will run Gurobi, CP-SAT, SCIP, and CBC from Python code inside our portfolio solver neO-DCN, as detailed in Section 4.3.

4.3 neO-DCN Portfolio Solver for DCN Optimization

The proposed ILP model has been implemented in our tool neO-DCN, which is a variant of our open-source tool neO [8] that we adapted to our DCN model. neO-DCN is publicly available at `https://github.com/kovasz/neO-DCN`.

neO-DCN is a portfolio solver, meaning that it can execute different ILP solvers, which were mentioned in Section 4.2, in parallel. The parallel execution is implemented by instantiating `ProcessPool` from the `pathos.multiprocessing` [12] Python module, which can run jobs with a non-blocking and unordered map.

The OR-Tools package provides two solver interfaces that we can use for ILP solving: (1) the `MPSolver` interface for MIP solvers such as Gurobi, SCIP and CBC, and (2) the `CPSolver` interface for Google's Constraint Programming solver CP-SAT. Both interfaces allow adding ILP constraints in the form

$$\texttt{solver.Add(}w_1\texttt{ * }x_1\texttt{ + ... + }w_n\texttt{ * }x_n\texttt{ <= }c\texttt{)}$$

where each w_i and c is an integer, and x_i a Boolean variable. Note that although the interface allows to use relational operators other than $\leq$, neO-DCN translates all constraints to "AtMost" constraints for normalization purposes.

For neO-DCN, we introduced a JSON input format to read data about the configuration of the network as well as the current configuration of flows. The benchmark files that we used in our experiments apply this input format and can be found in the repository of neO-DCN.

[1] `https://developers.google.com/optimization/cp/cp_solver`

4.4 Experimental Results

In our experiments, we are dealing with the real-world DCN topology same DCN topology as in [14]. Figure 2 shows the topology containing 20 switches and 16 hosts, and numerous links between those nodes. The bandwidth of each link is uniformly set to 1 Gbps.

The ILP solvers that we mentioned in Section 4.2 were run on the benchmark instances with a wall clock time limit of 1200 seconds. LINGO was run as a Windows desktop application, while the other ILP solvers as part of NEO-DCN on Linux. During the experiments, we measured the runtime of the solvers and, also, monitored the memory consumption of solvers by using the memory profiler `mprof`. While applying `mprof` to NEO-DCN was successful, we were not able to apply it to LINGO. This is why we will not provide memory consumption data for LINGO in the subsequent sections.

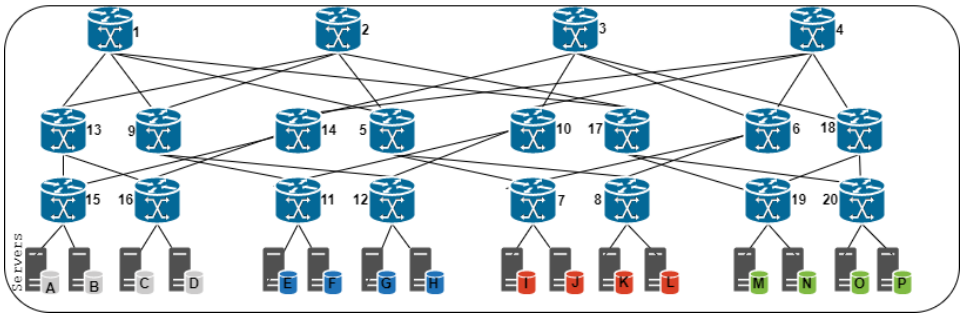

Figure 2: DCN topology (fat tree) for experiments.

4.4.1 Near Traffic Pattern

All mentioned solvers converge to the optimal solution for this traffic pattern very fast. They reduce the topology in Figure 2 to a minimum-tree DCN topology in Figure 3 by setting all unneeded links to off-state. Figure 3 shows the four scenarios of the near sine-wave pattern. In the first scenario, we burst roughly 1 Gbps distributed over 20 flows from servers A, B to C, D, i.e, unidirectional traffic. The optimizer aggregates all flows in six links. On the other hand, in the second scenario, the number of flows are increased to 30 and the traffic to 1.3 Gbps. The topology changes because the model balances the traffic over the links, and the number of links becomes 8 instead of 6. In the third scenario, part of the traffic is bidirectional, while we keep the traffic volume at 1.3 Gbps. We burst the same number of flows, 30, distributed as follows: (1) 10 flows from servers A to C, (2) 10 flows from B to D, and (3) 10 flows in a reverse direction, from server C to A. The optimizers output the minimum number 10 of the 12 links for the sake of aggregating as many flows as possible in one path. In the fourth scenario, we keep all the characteristics of the third scenario, except that we increase the traffic

volume to roughly 2 Gbps by sending new traffic from server D to B. As a result, the minimum number of active links becomes 40.

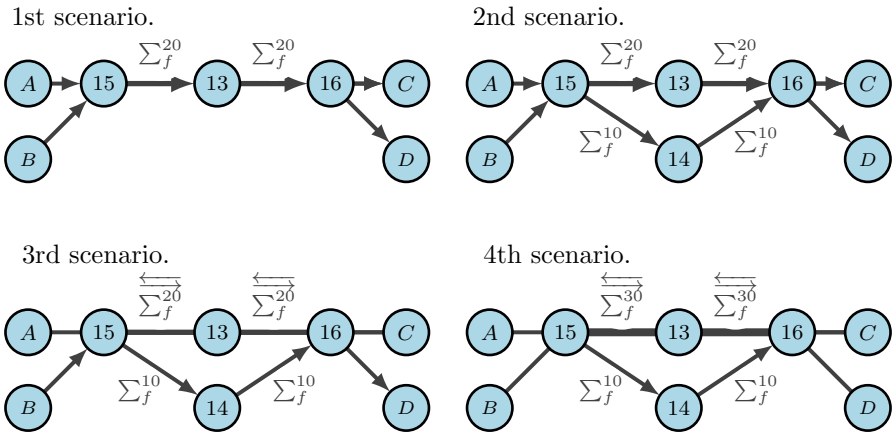

Figure 3: Four near-traffic scenarios, where directed links indicate flow direction. Undirected links represent bidirectional flows.

Table 1 shows some of the results reported by the ILP solvers, including the optimum value (e.g., minimum number of active links). In the table, the runtime of each solver is given in seconds. The results show that the benchmark instances in all scenarios can be solved in reasonable time by any of the solvers. However, Gurobi and CP-SAT outperform the other solvers by almost 1 order of magnitude on this benchmark.

Table 1: Runtimes (s) of ILP solvers for near traffic pattern

Scenario	**Flows**	**Opt.**	LINGO	Gurobi	CP-SAT	SCIP	CBC
1	20	6	2.11	0.44	0.42	0.63	0.53
2	30	8	4.21	0.53	0.53	0.73	3.03
3	30	10	3.21	0.53	0.52	0.93	0.62
4	40	14	5.01	0.73	0.72	1.33	3.43

4.4.2 Long Traffic Pattern

In this benchmark, we separate the servers from different PODs into two groups: sender and receiver. Besides that, we set the number of flows to a constant value of 24. Then, we gradually increase the traffic volume roughly from 1 Gbps to 5 Gpbs, which were captured in 14 benchmark instances, and we burst them into the DCN. The idea behind this benchmark is to demonstrate how the subset of active links changes according to the traffic demand when the number of flows is constant.

Table 2 shows the traffic volume in Gbps for each benchmark instances, and the corresponding optimum values (e.g., minimum number of active links).

While 6 active links are sufficient to use for the initial time interval (representing low traffic), one needs to activate all the 48 links in the topology for the last time interval (representing high traffic). In the table, one can definitely see higher runtimes than those in the near-traffic experiment. LINGO, SCIP, and CBC even timed out (TO) for some of the benchmark instances, due to the higher level of difficulty of solving, caused by a high load of traffic.

Table 2: Runtimes (s) of ILP solvers for long traffic pattern

Time Interval	**Traffic**	**Opt.**	LINGO	GUROBI	CP-SAT	SCIP	CBC
1	0.91	6	2.81	0.53	0.42	0.62	0.63
2	0.92	12	2.74	0.43	0.43	0.62	0.63
3	1.5	18	2.93	0.53	1.93	0.62	1.73
4	2.15	26	2.96	0.53	28.88	1.32	3.13
5	2.3	28	2.97	0.53	22.57	25.28	19.56
6	2.4	30	2.85	0.53	17.45	54.63	5.53
7	2.7	32	2.73	0.53	12.25	14.76	11.35
8	3	36	27.73	1.74	16.76	146.02	101.21
9	3.4	36	10.83	2.44	11.35	29.31	80.16
10	3.6	40	597.18	141.96	17.69	122.61	406.6
11	3.9	40	566.44	162.26	13.05	951.43	TO
12	4.2	44	TO	425.5	36.6	291.98	TO
13	4.5	44	TO	144.51	28.28	TO	TO
14	4.8	48	TO	575.49	45.01	TO	TO

The solvers' runtimes are visualized in Figure 4. Notice that the vertical axis, that represents the runtimes in seconds, is log-scaled. Up to the 7th time interval, when the traffic volume is 2.7 Gbps, all the solvers can find the optimum in a short time and, in particular, GUROBI and LINGO seem to provide a stable performance. For a higher volume of traffic, however, all the solvers loose efficiency very rapidly, except for CP-SAT, which keeps a surprisingly stable performance all the way to the very last time interval.

Figure 5 visualizes the memory consumption of the different ILP solvers. Recall that we could not apply memory profiling to LINGO. As the chart shows, all the solvers consume a moderate amount of memory for each benchmark instance. Most importantly, for CP-SAT, which was proved to be the fastest solver on this benchmark, memory consumption seems to be constant-like.

4.4.3 Random Traffic Matrix

In the benchmark with random traffic matrix, we inject different burst sizes and random flows into the DCN. The generated benchmark instances consist of 5, 10,

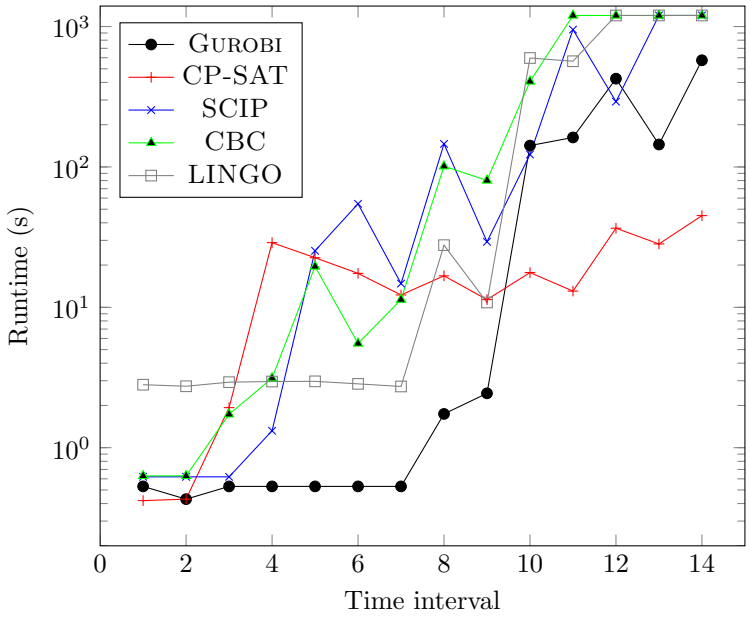

Figure 4: Runtimes of ILP solvers for long traffic pattern.

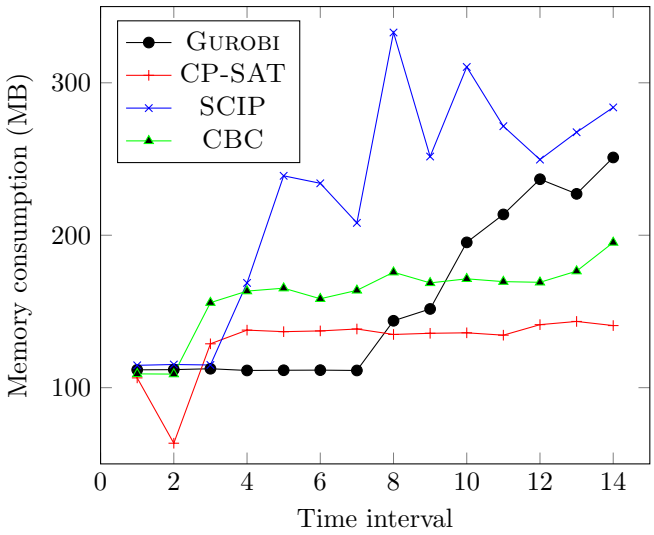

Figure 5: Memory consumption of ILP solvers for long traffic pattern.

20, ..., 200 flows, respectively, where each flow f is given by a random source host $f.S$, a random destination hosts $f.D$ and a random packet rate λ_f.

Table 3 gives several details for each benchmark instance. The number of flows gradually increases from 5 to 200. Note that benchmark instances up to 110 flows are satisfiable (SAT), while the ones above that are unsatisfiable (UNSAT), meaning that there does not exist any solution for them. For the SAT instances, the table shows the optimum values (e.g., minimum number of active links).

Table 3: Runtimes (s) of ILP solvers for random traffic

Flows	**Traffic**	**Result**	**Opt.**	LINGO	GUROBI	CP-SAT	SCIP	CBC
5	0.08	SAT	15	0.79	0.24	0.24	0.22	0.24
10	0.21	SAT	19	1.34	0.23	0.33	0.32	0.44
20	0.49	SAT	24	2.22	0.33	0.63	0.53	0.74
30	0.61	SAT	28	3.22	0.53	1.23	0.83	0.83
40	0.92	SAT	28	4.37	0.63	1.63	1.13	1.14
50	1.06	SAT	29	16.89	1.13	3.13	25.68	11.75
60	1.35	SAT	29	12.38	1.54	5.54	7.74	158.31
70	1.35	SAT	31	49	4.54	22.47	185.03	206.83
80	1.67	SAT	30	45.53	3.85	62.25	97.99	158.70
90	1.86	SAT	33	452.84	32.2	83.36	TO	TO
100	2.42	SAT	32	219.28	6.85	45.02	1160.16	329.72
110	2.25	SAT	33	275.24	34.92	TO	TO	481.98
120	2.71	UNSAT		TO	1.93	TO	7.44	42.51
130	8	UNSAT		TO	2.14	TO	8.25	41.3
140	7.71	UNSAT		TO	2.24	TO	10.25	69.56
150	7.84	UNSAT		TO	2.34	TO	9.15	78.88
160	8	UNSAT		TO	2.64	TO	7.44	63.04
170	8	UNSAT		TO	2.83	TO	10.55	116.14
180	7.81	UNSAT		TO	3.04	TO	9.35	136.88
190	7.99	UNSAT		TO	3.14	TO	16.16	87.49
200	8	UNSAT		TO	4.65	TO	13.66	115.13

Table 3 and Figure 6 show the solvers' runtimes for each benchmark instance. Notice that the vertical axis of the chart is log-scaled.

Only GUROBI is able to solve all the benchmark instances under the time limit. LINGO and CP-SAT times out on all the UNSAT instances, while all the other solvers are able to recognize the UNSAT case in quite a reasonable timeframe. CBC and SCIP time out on 1 and 2 SAT instances, respectively, which consist of a high number of flows.

Regarding runtime, GUROBI outperforms all the other solvers by 1 or 2 orders of magnitude, especially when comparing to LINGO that was used as an underlying solver in [14].

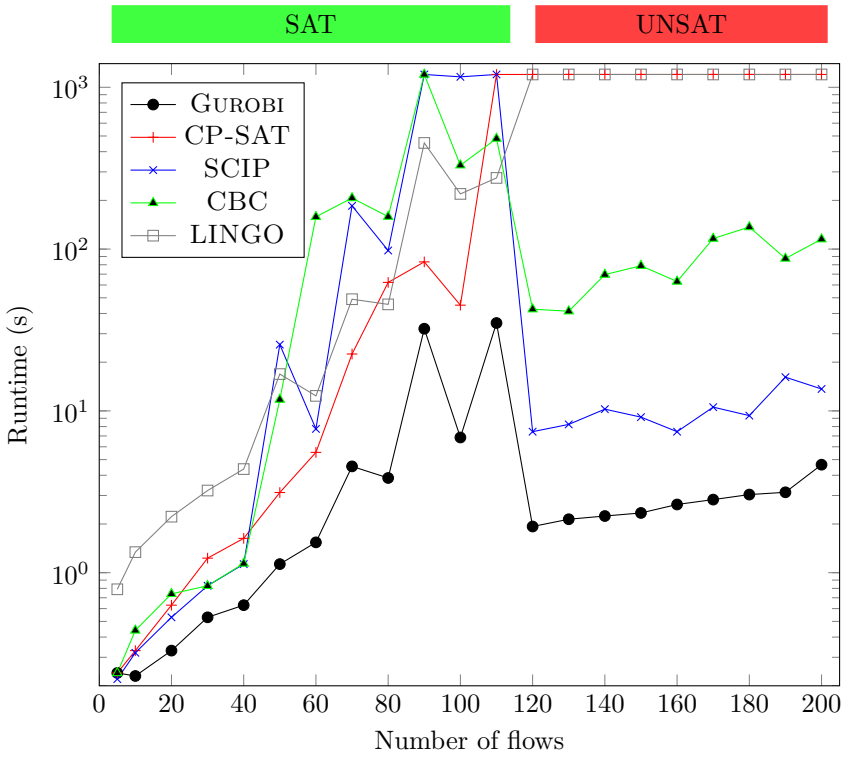

Figure 6: Runtimes of ILP solvers for random traffic.

The memory consumption we have recorded is shown in Figure 7. Gurobi, as the fastest solvers for the current benchmark, consumes a moderate amount of memory.

5 Conclusion and Future work

With the aim of optimizing the power consumption of a real DCN topology called the fat tree, we proposed an ILP model in our previous paper [14] and reported on experiments with the optimization toolkit LINGO. For our current paper, we have implemented the same model for other ILP solvers. We report on comparative experiments with them on a wide range of traffic benchmarks for three different communication patterns: (1) the near traffic pattern results show that Gurobi and CP-SAT outperform the other solvers regarding runtime for most traffic instances; (2) the long traffic pattern results show that above a traffic volume of 2.7 Gbps all the solvers dramatically loose efficiency, except for CP-SAT, which keeps a good performance and roughly constant memory consumption; (3) the random

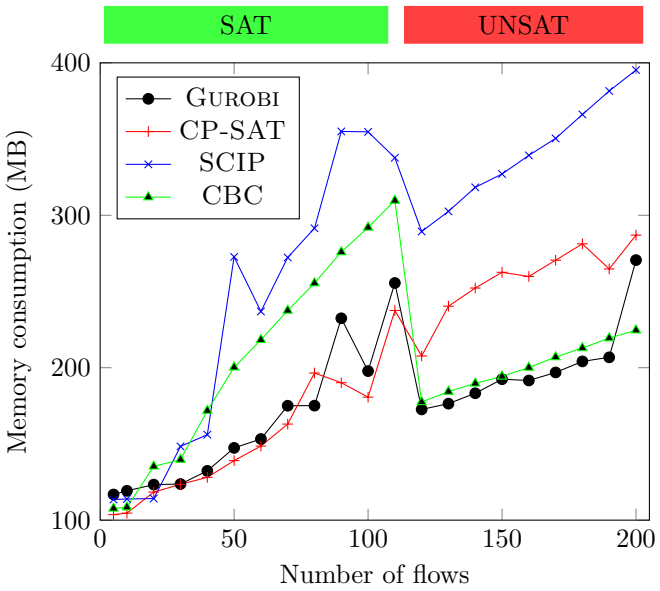

Figure 7: Memory consumption of ILP solvers for random traffic.

traffic results show that, for most of the traffic instances, GUROBI outperforms the other solvers regarding both runtime and memory. We can conclude that, for most of the benchmark instances, most of the solvers outperform LINGO regarding runtime. Consequently, it was definitely worth experimenting with those solvers, with GUROBI and CP-SAT in particular, as part of our new contribution.

As future work, it would be worth investigating how much ILP solvers scale for certain generalizations of the DCN model, such as using heterogeneous power consumption values for switches and links. In an ongoing work, we are upgrading the ILP model to save as much power as possible by adding more parameters, such as flow type [13]. Additionally, we are planning to experiment with pseudo-Boolean solvers as well.

References

[1] Al-Tarazi, Motassem and Chang, J Morris. Performance-aware energy saving for data center networks. *IEEE Transactions on Network and Service Management*, 16(1):206–219, 2019. DOI: 10.1109/TNSM.2019.2891826.

[2] Assefa, Beakal Gizachew and Ozkasap, Oznur. Framework for traffic proportional energy efficiency in software defined networks. In *Proceedings of the IEEE International Black Sea Conference on Communications and Network-*

ing (BlackSeaCom), pages 1–5. IEEE, 2018. DOI: 10.1109/BlackSeaCom.2018.8433618.

[3] Assefa, Beakal Gizachew and Özkasap, Öznur. A survey of energy efficiency in SDN: Software-based methods and optimization models. *Journal of Network and Computer Applications*, 137:127–143, 2019. DOI: 10.1016/j.jnca.2019.04.001.

[4] Bestuzheva, Ksenia et al. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, 2021. URL: http://nbn-resolving.de/urn:nbn:de:0297-zib-85309.

[5] Forrest, John et al. coin-or/cbc: Release releases/2.10.7, 2022. DOI: 10.5281/zenodo.5904374.

[6] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. URL: https://www.gurobi.com/documentation/9.5/refman/, 2022.

[7] Heller, Brandon, Seetharaman, Srinivasan, Mahadevan, Priya, Yiakoumis, Yiannis, Sharma, Puneet, Banerjee, Sujata, and McKeown, Nick. Elastictree: Saving energy in data center networks. In *NSDI'10: Proceedings of the 7th USENIX conference on Networked systems design and implementation*, Volume 10, pages 249–264, 2010.

[8] Kovásznai, G., Gajdár, K., and Kovács, L. Portfolio SAT and SMT solving of cardinality constraints in sensor network optimization. In *Proceedings of the 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 85–91. IEEE, 2019. DOI: 10.1109/SYNASC49474.2019.00021.

[9] Li, Xiaolin, Lung, Chung-Horng, and Majumdar, Shikharesh. Green spine switch management for datacenter networks. *Journal of Cloud Computing*, 5(1):1–19, 2016. DOI: 10.1186/s13677-016-0058-8.

[10] LINDO Systems Inc. Lingo the modeling language and optimizer. URL: http://www.lindo.com, 2020.

[11] Luo, Juan, Zhang, Song, Yin, Luxiu, and Guo, Yaling. Dynamic flow scheduling for power optimization of data center networks. In *Proceedings of the Fifth International Conference on Advanced Cloud and Big Data (CBD)*, pages 57–62. IEEE, 2017. DOI: 10.1109/CBD.2017.18.

[12] McKerns, Michael M, Strand, Leif, Sullivan, Tim, Fang, Alta, and Aivazis, Michael AG. Building a framework for predictive science. *arXiv preprint arXiv:1202.1056*, 2012. DOI: 10.48550/arXiv.1202.1056.

[13] Nsaif, Mohammed, Kovásznai, Gergely, Abboosh, Mohammed, Malik, Ali, and Fréin, Ruairí de. ML-based online traffic classification for SDNs. In *Proceedings of the IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, pages 217–222, 2022. DOI: 10.1109/CITDS54976.2022.9914138.

[14] Nsaif, Mohammed, Kovásznai, Gergely, Rácz, Anett, Malik, Ali, and de Fréin, Ruairí. An adaptive routing framework for efficient power consumption in software-defined datacenter networks. *Electronics*, 10(23), 2021. DOI: 10.3390/electronics10233027.

[15] Perron, Laurent and Furnon, Vincent. OR-Tools. URL: https://developers.google.com/optimization/, 2019.

[16] Rabee, Furkan, Al-Haboobi, A, and Nsaif, Mohammed Ridha. Parallel three-way handshaking route in mobile crowd sensing (PT-MCS). *Journal of Engineering and Applied Sciences*, 14:3200–3209, 2019. DOI: 10.36478/jeasci.2019.3200.3209.

[17] Rabee, Furkan, Nsaif, Mohammed, and Al-Haboobi, Ali. Reliable compression route protocol for mobile crowd sensing (RCR-MSC). *Journal of Communications*, 14:170–178, 2019. DOI: 10.12720/jcm.14.3.170-178.

[18] Wang, Rui, Jiang, Zhipeng, Gao, Suixiang, Yang, Wenguo, Xia, Yinben, and Zhu, Mingming. Energy-aware routing algorithms in software-defined networks. In *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE, 2014. DOI: 10.1109/WoWMoM.2014.6918982.

[19] Xu, Guan, Dai, Bin, Huang, Benxiong, Yang, Jun, and Wen, Sheng. Bandwidth-aware energy efficient flow scheduling with SDN in data center networks. *Future Generation Computer Systems*, 68:163–174, 2017. DOI: 10.1016/j.future.2016.08.024.