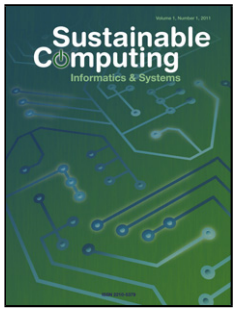


Title: Energy-aware Routing for Software-defined Networks with Discrete Link Rates: A Benders Decomposition-*based* Heuristic Approach

Author: Mohamad Khattar Awad Yousef Rafique Rym A. M'Hallah

PII: S2210-5379(16)30125-1
DOI: http://dx.doi.org/doi:10.1016/j.suscom.2016.11.003
Reference: SUSCOM 158

To appear in:

Received date: 7-8-2016
Accepted date: 17-11-2016

Please cite this article as: Mohamad Khattar Awad, Yousef Rafique, Rym A. M'Hallah, Energy-aware Routing for Software-defined Networks with Discrete Link Rates: A Benders Decomposition-*based* Heuristic Approach, *<![CDATA[Sustainable Computing: Informatics and Systems]]>* (2016), http://dx.doi.org/10.1016/j.suscom.2016.11.003

Energy-aware Routing for Software-defined Networks with Discrete Link Rates: A Benders Decomposition-*based* Heuristic Approach

Mohamad Khattar Awad[†], *Member, IEEE,* Yousef Rafique[†], *Student Member, IEEE,* Rym A. M'Hallah[‡]
[†] Department of Computer Engineering, College of Computing Sciences and Engineering, Kuwait University
[‡] Department of Statistics and Operations Research, College of Science, Kuwait University
E-mail: mohamad@ieee.org, yousef.rafique.kw@ieee.org, rym.mhallah@ku.edu.kw

Abstract

The energy efficiency of wired networks has received considerable attention over the past decade due to its economic and environmental impacts. However, because of the vertical integration of the control and data planes in conventional networks, optimizing energy consumption in such networks is challenging. Software-defined networking (SDN) is an emerging networking paradigm that decouples the control plane from the data plane and introduces network programmability for the development of network applications. In this work, we propose an energy-aware integral flow-routing solution to improve the energy efficiency of the SDN routing application. We consider discreteness of link rates and pose the routing problem as a Mixed Integer Linear Programming (MILP) problem, which is known to be NP complete. The proposed solution is a heuristic implementation of the Benders decomposition method that routes additional single and multiple flows without resolving the routing problem. Performance evaluations demonstrate that the proposed solution achieves a close-to-optimal performance (within 3.27% error) compared to CPLEX on various topologies with less than 0.056% of CPLEX average computation time. Furthermore, our solution outperforms the shortest path algorithm by 24.12% to 54.35% in power savings.

Index Terms

Energy-aware routing, Software-defined Networks, Network Optimization.

I. Introduction

THE telecommunication sector is one of the fastest growing sectors, having reached 7 billion mobile subscriptions by the end of 2015 and 7-fold growth in Internet penetration, from 6.5% to 43%, between 2000 and 2015 [1]. According to the same report, the proportion of households with Internet access has also increased from 18% in 2005 to 46% in 2015. The growing number of users, increasing demand and expansion of wireless communications have led to tremendous growth in energy consumption in ICT worldwide. Studies have shown that more than 4.7% of worldwide energy is being consumed by ICT [2], [3]. Major organizations in the U.S. spend millions of dollars annually on power consumption [4]. For example, eBay, Akami, Microsoft and Google annually consume 0.6×10^5 MWh, 1.7×10^5 MWh, 6×10^5 MWh and 6.3x105 MWh, respectively, which costs $3.7, $10M, $36M and $38M. The Massachusetts Institute of Technology campus network infrastructure consumes 7×10^5 MWh annually, which costs $62 million. In the U.K., the total energy consumption of British Telecom has reached 2.6 TWh, which is 10% of total power consumption in the U.K. [5]. Similarly, the total energy consumption of Deutsche Telecom in Germany was approximately 3 TWh in 2007 [5]. Moreover, ICT is responsible for approximately 2% of global CO_2 emissions, which corresponds to approximately 66% of CO_2 emissions in Germany, 100% of the CO_2 emissions caused by international air traffic, and 25% of the CO_2 emissions produced by passenger cars world-wide [6]. With the increasing demand for telecommunication services, energy consumption in this sector is becoming a major concern from both an economic and environmental perspective. This increasing demand calls for more energy efficient and eco-friendly communication networks.

Reconfiguring conventional networks to implement energy efficient policies is challenging due to the vertical integration of the control and data planes at each networking device. The control plane optimizes data handling decisions whereas the data plane forwards data according to the decisions of control plane; both are integrated at each device, thus limiting the ability to achieve adaptive control. In addition, a change in data handling policies requires configuring each device using low-level commands that are often vendor specific [7]. To simplify this complexity and allow for innovation in networking, a new networking paradigm, software-defined networking (SDN), was developed to sunder the control and data planes. The data plane remains at the networking devices, whereas the control plane is logically displaced to a central controller. Based on the network status reported by network devices, the central controller adaptively optimizes and remotely configures the entire network [8].

Two major approaches were proposed in the literature for energy saving, namely, rate adaptation, also known as speed scaling, and powering down. The speed-scaling approach reduces the energy consumption of networks by scaling the power consumption of a network element to the amount of traffic it carries [9]. The power down approach conserves energy by switching off unused

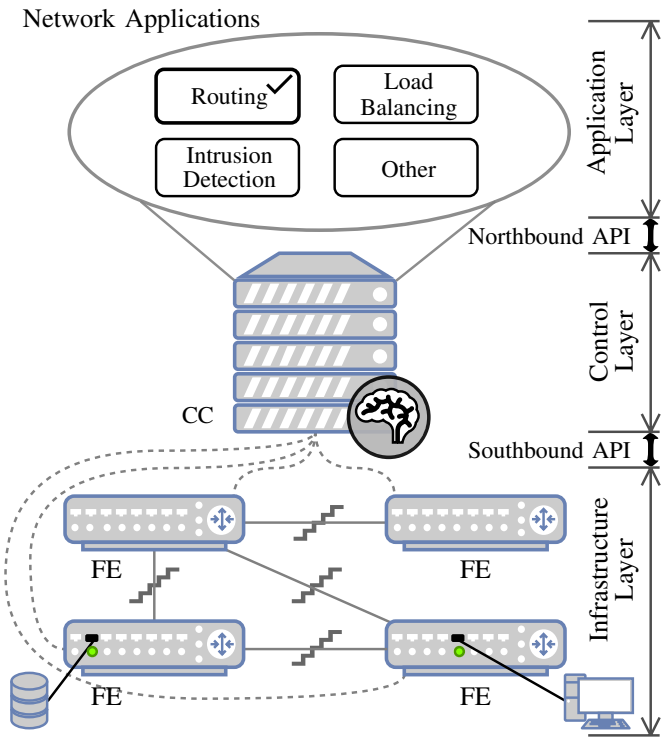


Figure 1: The SDN three-layer architecture comprising the application layer, control layer and infrastructure layer. The control layer consists of the central controller, "the brain of the network", and interfaces the above and below layer through the northbound and southbound APIs, respectively.

network elements, which operate either at the full rate or zero rate [10]. In [9], Andrews et al. studied the network-wide integral routing problem with the objective of satisfying a set of demands' rate requirements while minimizing energy consumption. They reported that optimizing routes highly depends on the characteristics of the energy function. In particular, they proposed a constant approximation algorithm for sub-additive functions and showed that no bounded approximation exists for super-additive functions. Andrews et. al. extend their work to consider the powering down approach in [10]. They showed that ON-OFF oscillation can be reduced via routing and presented a logarithmic approximation for both energy consumption and end-to-end delay. Nedevschi et al. [11] showed that both speed scaling and power down approaches can bring significant energy saving with negligible end-to-end delay and packet loss. However, the realization of these approaches in conventional networks is challenging, as achieving network-wide energy optimization requires an accurate estimation of the network status; this estimation includes link utilization, a complete traffic matrix and network topology. Therefore, the adoption of these approaches was limited to distributed and per-link implementations, e.g., IEEE 802.3.az, which does not exploit the full energy saving margin of entire networks.

Unlike conventional networks, software-defined networks are intrinsically programmable and status measurable. Under SDN, the central controller receives all traffic flow routing requests; thus, it is aware of the traffic matrix. In addition, forwarding elements measure various network status metrics, e.g., link utilization, connectivity, and delay, and report them to the central controller. Based on the received report, the central controller constructs a network-wide overview of the network status. Therefore, the central controller consolidates traffic routes to power off as many links as possible and scales all link rates to their utilization across the entire network while provisioning traffic requests.

The scope of this work is to design an energy-efficient routing algorithm for the routing application of software-defined networks. This network application resides in the application layer of SDN as shown in Figure 1. The proposed algorithm allows the controller to configure routes and link rates remotely by updating the network devices flow tables and setting their ports operating rates. This algorithm approximately solves the following SDN routing problem: Given the network topology, a set of data flows and their traffic demands, the available discrete operating rates of each link, find the integral routing paths of data flows and the operating rates of all links so that network energy consumption is minimized and traffic demands are provisioned. In integral routing, each flow is routed on a single path from its source to its destination. This routing is of practical importance in scenarios where either frames are not arbitrarily divisible [12] or packet ordering is not preferable [9]. The power consumption of ports is modeled by a discrete step-increasing discontinuous function of their discrete operating rates [13]. In our previous study [14], we show that discrete discontinuous rates are able to model practical scenarios and brings significant energy savings.

Due to the programmability and configuration flexibility introduced by SDN, several recent studies have proposed energy optimization algorithms for software-defined networks. Markiewicz et al. propose [15] an algorithm that reconfigures the network to maximizes the number of OFF links. Similarly, in [16] a greedy algorithm was presented to reroute traffic so that the number of OFF links is maximized. These algorithms disable unused links to conserve energy while operating ON links at their highest rate. They overlook the energy savings that can be achieved by adjusting the link rates to match their utilization. Zhu et al. [17] develop an energy-aware component in an open-sourced network management platform for data centers. The energy aware-component adopts priority-based shortest path routing and exclusive flow scheduling. Wang et al. [18] present a fast topology-aware heuristic scheme for multi-resource energy-efficient routing in cloud data centers. Energy-aware routing and flow scheduling schemes for data centers leverage the structural regularity and symmetricity of data center topologies as well as the data centers' special traffic characteristics [15]. Therefore, these schemes are inapplicable in corporate or operator networks, which do not exhibit similar topology or traffic characteristics in the data center networks. Wang et al. [19] investigate energy saving via integral routing and rate adaptation in networks with discrete link rates. They proposed a routing solution that transformed the discrete rate function into a continuous one and relaxed the integral routing constraint to convexify the problem. The solution achieves a constant approximation performance for uniform demand and the bounded adjacent steps ratio of the discrete rate function. However, these stringent requirements are difficult to satisfy in current networks. Tang et. al. [20] studied the energy-efficient flow allocation problem in networks with discrete link rates. Given a set of candidate paths, they developed two greedy algorithms and one linear programming (LP)-based algorithm to allocate fractions of data flows on the candidate paths; one greedy algorithm is for allocating a single flow, whereas the other two algorithms are for allocating multiple flows. The greedy algorithms outperformed a shortest path baseline solution, whereas the LP-based algorithm provided a close-to-optimal solution. However, it is not clear how different algorithms can be implemented in one network or how candidate paths are pre-computed or updated. Furthermore, the LP-based algorithm solves a series of linear programs, which requires access to a commercial optimization package at the network controller; hence, it is inapplicable in controllers with limited computational power.

In this paper, we propose a unified routing heuristic solution that can handle single and multiple data flow routing requests in software-defined networks. The solution does not require pre-computation of candidate paths or the support of a commercial solver. Furthermore, it routes demands of nonuniform size and sets link rates with general energy consumption step-increasing functions. It is a relaxation of the Benders-type cutting plane generation procedure [21], [22]. In particular, the *original* energy-efficient routing problem is formulated as a mixed integer linear programming problem (MILP). The polyhedron of the routing feasible space is represented by a large set of inequalities; however, only a limited number of inequalities is considered in the relaxed version of the problem. The routing solution obtained by solving the *relaxed* problem is validated by solving the *constraint generation* problem which identifies the most violated constraints of the *original* problem. The *relaxed* problem is augmented by the identified constraint and solved iteratively until none of the constraints of the *original* problem is violated. We develop two algorithms, the Link Rate Control (LRC) algorithm and Violated Constrain Generation (VCG) algorithm for solving the *relaxed problem* and *constraint generation* problem, respectively. The LRC algorithm is an extension of the heuristic presented in [22] for solving the *relaxed problem*. The VCG algorithm solves the dual of the *constraint generation* problem. The contributions of this work are summarized as follows:

- We formulate the energy-efficient integral routing problem of software-defined networks with discrete link rates as MILP and develop a solution to minimize network energy consumption while satisfying traffic requirements. The proposed solution is a heuristic implementation of the Benders-type cutting plane generation procedure and consists of two algorithms, LRC and VCG.
- We perform extensive simulation experiments to evaluate the performance of the proposed algorithm in well-known network topologies available at the library of test instances for Survivable fixed telecommunication Network Design (SNDlib) [23]. The solution is validated by comparing the results obtained by CPLEX with those obtained by shortest path algorithm.

The paper is organized as follows. The network model and problem formulation are presented in Section II. In Section III, a Benders decomposition-based solution is developed, and the two algorithms for solving the *relaxed* problem and the *constraint generation* problem are presented. Performance evaluation results are presented in Section IV, and conclusions are drawn in Section V. The simulation codes are available at http://www.mohamadawad.com.

II. Network Model and Problem Formulation

We consider an SDN architecture proposed by the Open Networking Foundation (ONF) as shown in Figure 1 [24]. ONF is an organization dedicated to the standardization and promotion of SDN. Three layers constitute this architecture: the infrastructure layer, the control layer and the application layer. The infrastructure layer consists of simple network devices referred to by forwarding elements (FEs). These elements forward packets based on forwarding rules installed in their forwarding tables. In addition, they collect network information and report it to the central controller (CC). The control layer serves as a liaison between the application and infrastructure layers through its northbound and southbound application programming interfaces (APIs). One of the first southbound APIs is OpenFlow [24]. The application layer consists of network applications designed to control and optimize the network. Examples of these applications are routing algorithms, resource allocation schemes, load

balancing schemes, mobility management schemes and intrusion detection systems access. Network applications access a global view of the network which is maintained by the CC to control and optimize the network operation. The network application control instructions are represented by flow rules and pushed by the CC to FEs' forwarding tables.

The network is formed by a single CC and a finite number of FEs. The CC computes the optimal integral route for each demand based on the proposed solution and pushes the results to the FEs, which implement these routes as flow rules in their forwarding tables [25]. The network is modeled by an undirected graph $G(\mathcal{E},\mathcal{L})$, where $\mathcal{E}=\{1,\cdots,e,\cdots,E\}$ is the set of FEs and $\mathcal{L}=\{1,\cdots,l,\cdots,L\}$ is the set of bi-directional links connecting them. $E=|\mathcal{E}|$ and $L=|\mathcal{L}|$ denote the cardinality of the set of FEs and set of links. The set $\mathcal{L}$ does not include links connecting the FEs to the central controller. The subset of links $\mathcal{L}$ originated at FE e is denoted by $\mathcal{L}_e^{\rightarrow}$, whereas the subset of links terminated at e is denoted by $\mathcal{L}_e^{\leftarrow}$; their union is $\mathcal{L}_e$. There is a set of F data flows, represented by $\mathcal{F}=\{1,\cdots,f,\cdots,F\}$, to be routed through the network. The origin and destination FEs of the f^{th} flow are represented by $o(f)$ and $d(f)$, respectively. A given flow $f\in\mathcal{F}$ is routed from $o(f)$ to $d(f)$ on a path $\mathcal{P}^f$, which is the set of connected links forming the route of f. It requires a rate $r^f>0$ on each link $l\in\mathcal{P}^f$.

The data rate required to support a given flow f is denoted by r^f. Let $r_l^f\geq 0$ be the decision variable reflecting the data rate required on link $l\in\mathcal{L}$ by flow $f\in\mathcal{F}$. When $l\in\mathcal{P}^f,\ r_l^f=r^f$. By contrast, when f does not travel through $l,\ r_l^f=0$. The flow rates through an FE $e,\ e\in\mathcal{E}$, must satisfy the flow conservation constraints. For every flow $f,\ f\in\mathcal{F}$, and FE $e,\ e\in\mathcal{E}$,

$$\sum_{l\in\mathcal{L}_e^{\rightarrow}} r_l^f-\sum_{l\in\mathcal{L}_e^{\leftarrow}} r_l^f=\begin{cases}0 & e\neq o(f) \text{ and } e\neq d(f)\\ r^f & e=o(f)\\ -r^f & e=d(f).\end{cases} \tag{1}$$

Consequently, the sum of the rates routed through a given link $l,\ l\in\mathcal{L}$, can be written as

$$r_l=\sum_{f\in\mathcal{F}} r_l^f. \tag{2}$$

These rates constitute a vector $\mathbf{r}=[r_l]_{l\in\mathcal{L}}$ of size L.

To support the routed flow r_l through link $l,\ l\in\mathcal{L}$, the CC activates one of the available discrete link rates $\overline{r}_l\in\overline{\mathcal{R}}=\{R_0,R_1,\cdots,R_{\max}\}$, where the available rates are sorted in ascending order (i.e., $R_0<R_1<\cdots<R_{\max}$). Stated differently, the activated link rate $\overline{r}_l$ for link $l,\ l\in\mathcal{L}$, is given by

$$\overline{r}_l=\begin{cases}R_0 & 0<r_l\leq R_0,\\ R_1 & R_0<r_l\leq R_1,\\ \vdots & \vdots\\ R_{\max} & R_{\max-1}<r_l\leq R_{\max}\end{cases}. \tag{3}$$

An activated discrete operating rate $\overline{r}_l\in\overline{\mathcal{R}}$ for a link $l,\ l\in\mathcal{L}$ engenders energy consumption

$$\Gamma_l(\overline{r}_l)=\begin{cases}\gamma_0 & \text{if } \overline{r}_l=R_0,\\ \gamma_1 & \text{if } \overline{r}_l=R_1,\\ \vdots & \vdots\\ \gamma_{\max} & \text{if } \overline{r}_l=R_{\max}\end{cases}. \tag{4}$$

The set of discrete rates activated on the L links in the network are denoted by the vector $\overline{\mathbf{r}}=[\overline{r}_l]_{l\in\mathcal{L}}$ of size L.

Based on the above description, the problem of routing flows and setting discrete link rates consists of finding the flow rates $r_l^f,\ l\in\mathcal{L},f\in\mathcal{F}$ that minimize network energy consumption. Formally, the problem is equivalent to the following

mathematical program:

$$\min \sum_{l=1}^{L} \Gamma_l(\overline{r}_l) \tag{5a}$$

$$\text{subject to} \quad r_l = \sum_{f \in \mathcal{F}} r_l^f \qquad l \in \mathcal{L} \tag{5b}$$

$$\sum_{l \in \mathcal{L}_e^{\rightarrow}} r_l^f - \sum_{l \in \mathcal{L}_e^{\leftarrow}} r_l^f = \begin{cases} 0 & f \in \mathcal{F}, e \in \mathcal{E}, e \neq o(f), e \neq d(f) \\ r^f & f \in \mathcal{F}, e = o(f) \\ -r^f & f \in \mathcal{F}, e = d(f). \end{cases} \tag{5c}$$

$$\overline{r}_l = \begin{cases} R_0 & 0 < r_l \leq R_0, \\ R_1 & R_0 < r_l \leq R_1, \\ \vdots & \vdots \qquad\qquad l \in \mathcal{L} \\ R_{\max} & R_{\max -1} < r_l \leq R_{\max}. \end{cases} \tag{5d}$$

The objective function in (5a) minimizes energy consumption at all links. The constraint (5b) defines the link rate required to support all flows passing through it. Flow conservation is guaranteed in constraint (5c). The discrete link rate corresponding to the sum of flows through link l is defined in constraint (5d).The mathematical formulation in (5) is equivalent to a mixed-integer programming problem with a non-convex discrete-cost step increasing function. The integral routing constraint and discreteness of the objective function make the problem NP hard [26].

III. Bender's Decomposition-based Solution

In this section, we define a polyhedron of the feasible space defined by Equations (5b) to (5d) and represent it by a set of inequalities. Then, we adopt a heuristic implementation of the Benders approach to iteratively solve the problem.

Let $\mathfrak{R} \subset \mathbb{R}_+^L$ be a polyhedron representing the set of all feasible discrete link rates $\overline{\mathbf{r}}$, where $\mathbb{R}_+$ is the set of real numbers [27]. The $\mathfrak{R}$ polyhedron can be defined as follows:

$$\mathfrak{R} = \{\overline{\mathbf{r}} \in \mathbb{R}_+^L |\ \overline{\mathbf{r}} \geq \mathbf{r} \text{ and } \mathbf{r} \text{ satisfying constraints (5b) to (5d)}\}.$$

Based on this definition of the feasible space, a solution $\overline{\mathbf{r}}$ is feasible if and only if $\overline{\mathbf{r}} \in \mathfrak{R}$. Therefore, the problem given in equations (5b) to (5d) can be re-written as follows

$$\min \sum_{l=1}^{L} \Gamma_l(\overline{r}_l) \tag{6a}$$

$$\text{subject to} \quad \overline{\mathbf{r}} \in \mathfrak{R} \tag{6b}$$

$$\overline{r}_l \in \overline{\mathcal{R}} \quad \forall l \in \mathcal{L}. \tag{6c}$$

The polyhedron $\mathfrak{R}$ can be represented by a set of linear inequalities [21], [27]. For *any* linear, non-negative design link costs $\boldsymbol{\lambda} = \{\lambda_1, \cdots, \lambda_l, \cdots, \lambda_L\}$, $s^f(\boldsymbol{\lambda})$ is the cost to route flow f, $f \in \mathcal{F}$, from $o(f)$ to $d(f)$. That is, if f is a unique flow on a network whose link costs are $\boldsymbol{\lambda} \in \mathbb{R}_+^L$, then $s^f(\boldsymbol{\lambda})$ would be the cost of the optimal path for f.

Based on Theorem 5 in [28, chapter 6], which is a consequence of Farkas and Minkowski's Lemma [28, Appendix 1], the polyhedron representation of the feasible space $\mathfrak{R}$ can be captured by the following set of "metric inequalities":

$$\sum_{l \in \mathcal{L}} \lambda_l \overline{r}_l \geq \sum_{f \in \mathcal{F}} r^f s^f(\boldsymbol{\lambda}) \qquad \forall \boldsymbol{\lambda} \geq 0. \tag{7}$$

The theorem, which first appeared in [29], has been widely applied in the area of multi-commodity flow routing [21], [27], [28], [30], [31]. It has been shown that the "metric inequalities" are a special case of the well known Benders inequalities [32].

The inequalities in (7) imply that the lowest routing cost in terms of $\boldsymbol{\lambda}$, is to route flows on their individual shortest paths. Subsequently, a lower bound for $\sum_{l \in \mathcal{L}} \lambda_l \overline{r}_l$ is the sum of shortest path costs $s^f(\boldsymbol{\lambda})$ multiplied by their required rates r^f [30], [31]. Thus, for any value of $\boldsymbol{\lambda} \geq 0$, the discrete rates $\overline{\mathbf{r}}$ of the activated links are feasible and belong to $\mathfrak{R}$ if and only if the constraints in (7) are satisfied. With this representation of the feasible space, the problem in equations (6a) to (6c) can be

reformulated as follows

$$(\text{OP}) \quad \min \sum_{l=1}^{L} \Gamma_l(\overline{r}_l) \tag{8a}$$

$$\text{subject to} \quad \sum_{l\in\mathcal{L}} \lambda_l \overline{r}_l \geq \sum_{f\in\mathcal{F}} r^f s^f(\boldsymbol{\lambda}) \qquad \forall \boldsymbol{\lambda} \geq 0 \tag{8b}$$

$$\overline{r}_l \in \overline{\mathcal{R}} \quad l \in \mathcal{L}. \tag{8c}$$

Despite the accurate representation of the feasible space provided by these "metric inequalities" in (8b), their number is exponential. This result suggests that a constraint generation-based approach should be adopted to solve the *original* problem in OP [21], [31]. The algorithm iteratively solves a *relaxed* version RP^j of OP in the j^{th} iteration. RP^j considers only a subset of the constraints in (8b). Given the current solution $\overline{\mathbf{r}}$, it finds the most violated constraint of the problem (8) and adds it to RP^j; i.e., it appends it to the constraints generated in iterations $1, \cdots, j-1$. In the j^{th} iteration, the violated constraint is found by solving the *constraint generation* problem (CGP^j). The algorithm iterates until an optimal solution $\overline{\mathbf{r}}^*$ that does not violate any of the *original* problem constraints is found. Figure 2 shows a flow chart of the proposed solution major steps as well as problems RP^j and CGP^j.

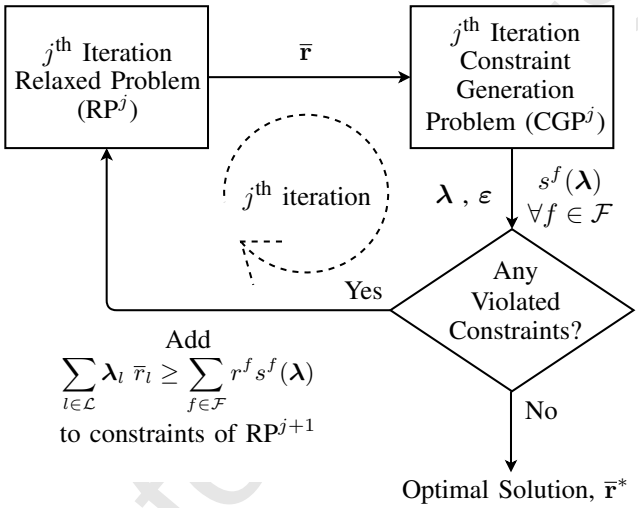


Figure 2: A flow chart of the major steps of the proposed solution, the problems solved in each iteration, and the variables passed among them.

Let $\mathcal{C}^j$ be the set of metric inequalities considered in the j^{th} iteration, where the vector of link costs $\boldsymbol{\lambda}$ is augmented by the superscript $(\cdot)^c$ to indicate the constraint index, i.e., $\boldsymbol{\lambda}^c, \ \forall c \in \mathcal{C}^j$. Therefore, RP^j becomes

$$\left(\text{RP}^j\right) \quad \min \sum_{l=1}^{L} \Gamma_l(\overline{r}_l) \tag{9a}$$

$$\text{subject to} \quad \sum_{l\in\mathcal{L}} \boldsymbol{\lambda}_l^c \, \overline{r}_l \geq \sum_{f\in\mathcal{F}} r^f s^f(\boldsymbol{\lambda}^c) \quad \forall c \in \mathcal{C}^j \tag{9b}$$

$$\overline{r}_l \in \overline{\mathcal{R}}_l \quad \forall l \in \mathcal{L}. \tag{9c}$$

Starting with a non-feasible initial solution, the proposed algorithm identifies a violated constraint and adds it to the problem to be solved in the following iteration RP^j. In the following iteration, RP^j is solved and a solution $\overline{\mathbf{r}}$ is obtained, i.e., the set of discrete rates on all links. The feasibility of the solution is evaluated by checking if all flows are simultaneously routed without violating any of the original problem (5) constraints [31]. Specifically, the actual link rates do not exceed the activated discrete rates and flow conservation holds. Let the decision variable $\varepsilon_l \geq 0$ be the excess rate on link $l, \forall l \in \mathcal{L}$ generated by routing flows with rates higher than the activated discrete link rate. In other words, given the set of discrete rates $\overline{\mathbf{r}}$, ε_l captures the rate violation on each link l. The CGP^j problem amounts to routing all flows in such a way that the rate violation on all

network links is minimized. The CGP^j is posed as follows [31]:

$$\big(\text{CGP}^j\big) \quad z^j = \min_{\mathbf{r}} \sum_{l=1}^{L} \varepsilon_l \tag{10a}$$

$$\text{subject to} \quad r_l - \varepsilon_l \leq \overline{r}_l \quad \forall l \in \mathcal{L} \tag{10b}$$

$$\sum_{l \in \mathcal{L}_e^{\rightarrow}} r_l^f - \sum_{l \in \mathcal{L}_e^{\leftarrow}} r_l^f = \begin{cases} 0 & f \in \mathcal{F}, e \in \mathcal{E}, e \neq o(f), e \neq d(f) \\ r^f & f \in \mathcal{F}, e = o(f) \\ -r^f & f \in \mathcal{F}, e = d(f) \end{cases} \tag{10c}$$

$$\varepsilon_l \geq 0 \quad l \in \mathcal{L}. \tag{10d}$$

Ideally, for a given feasible rate $\overline{\mathbf{r}}$, the rate violation $\varepsilon_l = 0$ for all $l \in \mathcal{L}$; resulting in the CGP^j problem having a zero objective function value. Stated differently, when $z^j = 0$, the solution $\overline{\mathbf{r}}$ received from RP^j is feasible. Alternatively, $z^j > 0$ because one or more excess variables are strictly positive, the current solution of RP^j is infeasible and violates at least one of the constraints in (8b). Therefore, if for the j^{th} iteration CGP^j is solved and $z^j > 0$, the most violated constraint is generated and appended to the set of RP^j's constraints to be considered in $j^{\text{ss1s21ss2 th}} + 1$ iteration, i.e., $|\mathcal{C}^{j+1}| = |\mathcal{C}^j| + 1$. In what follows, the parameters $\boldsymbol{\lambda}$ and $s^f(\boldsymbol{\lambda})$ associated with the most violated constraint are derived.

The dual problem of CGP^j uses the dual variables α_l and β_e^f associated with equations (10b) and (10c), respectively. The dual problem is given by:

$$\big(\text{DCGP}^j\big) \quad z^j_{\text{Dual}} = \max \sum_{f \in \mathcal{F}} r^f \beta^f_{e=o(f)} - \sum_{l \in \mathcal{L}} \alpha_l \, \overline{r}_l \tag{11a}$$

$$\text{subject to} \quad \beta_e^f - \beta_{e'}^f - \alpha_l \leq 0 \quad \forall l = (e, e') \in \mathcal{L}, \ \forall f \in \mathcal{F} \tag{11b}$$

$$0 \leq \alpha_l \leq 1 \quad l \in \mathcal{L}. \tag{11c}$$

The third subset of (10c) equations corresponding to destination nodes constitute redundant constraints. Subsequently, their associated dual variables $\beta^f_{e=d(f)}$ can be set to any value; thus, it was set to 0. Therefore, $\forall f \in \mathcal{F}, \ \beta^f_{e=d(f)} = 0$. In addition, the subset of (10c) equations with a zero right hand-side do not contribute to the objective function of the dual. Thus, their associated dual variables $\beta_e^f, \ f \in \mathcal{F}, \ e \in \mathcal{E}, \ e \neq o(f)$ and $e \neq d(f)$, can be set to any value that satisfies the dual constraints. Because the dual problem is a maximization problem and setting all dual variables to zero yields an optimal solution whose value is 0, it is possible to augment the dual problem with $\beta^f_{e=o(f)} \geq 0$ without affecting the optimal solution to the problem. At optimality, the values of β_e^f are those of the reduced costs of their associated variables in CGP^j, which is a minimization problem; thus, it must be positive. In Lemma 1, α_l is related to the excess $\varepsilon_l \ \forall l \in \mathcal{L}$.

Lemma 1: If the allocated rate on a given link $\sum_{f \in \mathcal{F}} r_l^f$ exceeds the link rate $\overline{r}_l$, i.e., $\varepsilon_l > 0$, the dual variable $\alpha_l = 1$. However, if $\varepsilon_l = 0$, $\alpha_l = 0$.

Proof: Let υ_l denote the slack variable associated with the l^{th} equation (10b) of CGP^j; that is,

$$r_l - \varepsilon_l + \upsilon_l = \overline{r}_l \quad \forall l \in \mathcal{L}. \tag{12}$$

An optimal solution to CGP^j, which is a minimization problem, has zero reduced costs for the basic variables and positive reduced costs for non-basic variables. In addition, the reduced cost of $\upsilon_l \forall \ l \in \mathcal{L}$ corresponds to the value of the dual variable α_l associated with equation (10b) of CGP^j. In the matrix corresponding to constraints in equations (10b) and (10c) of CGP^j, for every $l \in \mathcal{L}$, the columns of ε_l and υ_l are equal in absolute value but opposite in sign. Thus, the indirect costs of ε_l and υ_l are equal in absolute value but opposite in sign. In addition, the coefficient of ε_l in the objective function, i.e., in equation (10a), is 1, whereas that of υ_l is 0. Therefore, the reduced cost of ε_l is the sum of 1 and of the indirect cost of υ_l. Stated differently, the reduced cost of ε_l is $1 - \alpha_l$.

In any basic solution of CGP^j, $\varepsilon_l = 0$ or $\upsilon_l = 0$. In fact, when the flow rate capacity of link l is exceeded, $\varepsilon_l > 0$ and $\upsilon_l = 0$. By contrast, when there is no excess capacity on link l, i.e., $r_l \leq \overline{r}_l$, the excess capacity $\varepsilon = 0$ and $\upsilon_l \geq 0$. In the former case, $\varepsilon_l = 0$ and is non-basic, whereas $\upsilon_l \geq 0$ and is basic. Therefore, the reduced cost of υ_l is zero, i.e., $\alpha_l = 0$. In the latter case, $\varepsilon_l > 0$ and is basic; thus, the reduced cost of ε_l is zero. In other words, $1 - \alpha_l = 0$; subsequently, $\alpha_l = 1$. ■

Let z^j denote the solution value of CGP^j, and z^j_{Dual} be the solution value of DCGP^j. Duality theory implies that $z^j_{\text{Dual}} \leq z^j$. When either CGP^j or DCGP^j has an optimal solution, the other does. In addition, at optimality $z^j_{\text{Dual}} = z^j$. When $z^j > 0$, the current solution of CGP^j is infeasible; however, it is feasible if and only if $z^j_{\text{Dual}} \leq 0$. Replacing z^j_{Dual} by its expression yields $\sum_{f \in \mathcal{F}} r^f \beta^f_{e=o(f)} - \sum_{l \in \mathcal{L}} \alpha_l \, \overline{r}_l \leq 0$. Therefore, it is a valid cut to the relaxed master problem RP^j, and can be rearranged to

$$\sum_{l \in \mathcal{L}} \alpha_l \, \overline{r}_l \geq \sum_{f \in \mathcal{F}} r^f \beta^f_{e=o(f)}. \tag{13}$$

Recall that $\boldsymbol{\lambda}$ is a vector of any linear and non-negative design link costs, and thus can be set to the dual variables α_l, i.e., $\boldsymbol{\lambda}_l = \alpha_l \ \forall l \in \mathcal{L}$. In addition, it was shown in [31] that

$$\beta^f_{e=o(f)} = s^f(\boldsymbol{\lambda}) \quad \forall f \in \mathcal{F}. \tag{14}$$

Hence, the constraint in (13) can be re-written as

$$\sum_{l \in \mathcal{L}} \boldsymbol{\lambda}_l \ \overline{r}_l \geq \sum_{f \in \mathcal{F}} r^f s^f(\boldsymbol{\lambda}), \tag{15}$$

which is equivalent to one of the constraints in (7). Therefore, given the solution $\overline{\mathbf{r}}$ obtained by solving RP^j, DCGP^j is solved to test the feasibility of $\overline{\mathbf{r}}$. If z^j_{Dual} equals zero, the solution $\overline{\mathbf{r}}$ is feasible and optimal. However, if $z^j_{\text{Dual}} > 0$, $\overline{\mathbf{r}}$ is infeasible and the constraint in equation (15) is appended to the set of constraints of RP^j, which is solved in the next iteration as RP^{j+1}. A synthesis of the proposed solution is presented below.

- *Step 1 - Initialization, $j = 0$*: At initialization, the proposed solution starts with a zero link rate, i.e., $\overline{\mathbf{r}} = \mathbf{0}$, and hence any routing would introduce the same excess on any traversed link. Therefore, we set the initial $\boldsymbol{\lambda}$ to $\mathbf{1}$ on all links and perform shortest path routing to compute $s^f(\boldsymbol{\lambda}) \ \forall f \in \mathcal{F}$. After computing the excess rate ε_l on each link, the VCG algorithm is invoked to generate the first most violated constraint. The constraint is added to the RP^j set of constraints $\mathcal{C}^j$.
- *Step 2 - Solve RP^j, $j = j+1$*: Invoke the LRC algorithm (presented in Subsection III-B) to solve the RP^j problem, which generates the link discrete rates $\overline{\mathbf{r}}$.
- *Step 3 - Solve DCGP^j*: Evaluate the feasibility of $\overline{\mathbf{r}}$ using the VCG algorithm, which is presented in Subsection III-A. If the solution is feasible, output $\overline{\mathbf{r}}^* = \overline{\mathbf{r}}$ as an optimal discrete link rate solution. However, if the solution is not feasible, generate a violated constraint, append it to the constraints of RP^j, and return to *Step 2*.

In the following two subsections we present the VCG and LRC algorithms for solving DCGP^j and RP^j, respectively.

A. Violated Constraint Generation (VCG) Algorithm

The pseudo-code of the VCG algorithm is described in Algorithm 1. Given a set of discrete rates $\overline{\mathbf{r}}$ and data flows requirements, VCG routes the F flows in such a way that the network excess rate, i.e., $\varepsilon^{\text{net}} = \sum_{l=1}^{L} \varepsilon_l$, is minimized and data flow requirements are satisfied. The algorithm solves DCGP^j as well as computing $\boldsymbol{\lambda}$ and $s^f(\boldsymbol{\lambda})$.

Starting with a solution $\overline{\mathbf{r}}$ and routing $\mathcal{P}^f \ \forall f \in \mathcal{F}$, the excess rate on each link $\varepsilon_l \ \forall l \in \mathcal{L}$ and network excess ε^{net} are evaluated. To minimize ε^{net}, the algorithm tries to eliminate the excess on links sequentially in descending order of excess by rerouting some of the flows on alternative paths. In particular, the link with the maximum ε_l is identified as $\hat{l}$ and added to the set of visited links $\mathcal{V}$. The largest set of flows traversing through $\hat{l}$ are added to the set of removable flows $\mathcal{S}$, such that $\sum_{\forall f \in \mathcal{S}} r^f_{\hat{l}} \geq \varepsilon_{\hat{l}}$. An attempt is then made to eliminate excess $\varepsilon_{\hat{l}}$ on this link $\hat{l}$ by rerouting the flows in the set $\mathcal{S} = \{1, \cdots, S\}$ on alternative paths. It is only an attempt because flows are not rerouted unless ε^{net} is reduced (line 15). The alternative paths are computed based on the "K-Shortest-Paths" algorithm presented in [33]. Let θ^s_k be the excess in the network when one of the flows s in $\mathcal{S}$ is routed in addition to all other flows $\mathcal{F} \setminus \mathcal{S}$ on one of its "K-Shortest-Paths" denoted by $\mathcal{K}^s$ (line 12). The alternative path $\hat{k}^s$ that introduces the least θ^s_k for each flow $s \in \mathcal{S}$ is selected for rerouting (line 13), and the network excess after rerouting is computed $\varepsilon^{\text{net}}_{\text{reroute}} = \sum_{l \in \mathcal{L}} \varepsilon_l$.

In line 15, if the resulting $\varepsilon^{\text{net}}_{\text{reroute}}$ is less than the previously computed network excess ε^{net} before rerouting, we update the paths $\mathcal{P}^f$ with the rerouting paths $\hat{k}^s \ \forall f \in \mathcal{S}$. Alternatively, the original routes in $\mathcal{P}^f_{temp}$ are restored and the possibility of eliminating excess on another link $l \in \mathcal{L} \setminus \mathcal{V}$ is evaluated.

After minimizing excess on all links, if the network excess is eliminated $\varepsilon^{net} = 0$, the discrete link rates $\overline{\mathbf{r}}$ are feasible. However, if $\varepsilon^{net} > 0$, $\overline{\mathbf{r}}$ is infeasible and a constraint is generated by computing its parameters $\boldsymbol{\lambda}$ and $s^f(\boldsymbol{\lambda})$ based on Lemma 1. Specifically, links with $\varepsilon_l > 0$ are assigned a link cost $\boldsymbol{\lambda}_l = 1$ ($\alpha_l = 1$), whereas links with $\varepsilon_l = 0$ are assigned a link cost of $\boldsymbol{\lambda}_l = 0$ ($\alpha_l = 0$). Furthermore, the path cost $s^f(\boldsymbol{\lambda})$ is computed by adding all link costs on the path $\mathcal{P}^f \ \forall f \in \mathcal{F}$.

B. Link Rate Control (LRC) Algorithm

Constraints generated by the VCG algorithm guide the LRC algorithm in satisfying data flows requirements while minimizing energy consumption. The LRC algorithm updates the vector of link rates $\overline{\mathbf{r}}$ to satisfy constraints considered in the current iteration of the RP^j, i.e., $\mathcal{C}^j$. A feasible solution $\overline{\mathbf{r}}$ generated by LRC is feasible with respect to the limited set of constraints in RP^j, and does not necessarily satisfy the *original* problem constraints. The feasibility of a solution with respect to the *original* problem constraints is validated by the VCG algorithm.

The feasibility of $\overline{\mathbf{r}}$ given the limited set of constraints $\mathcal{C}^j$ is measured by [22]:

$$\varphi(\overline{\mathbf{r}}) = \sum_{c \in \mathcal{C}^j} \max \left(0, \sum_{f \in \mathcal{F}} r^f s^f(\lambda^c) - \sum_{l \in \mathcal{L}} \overline{\mathbf{r}}_l \lambda^c_l \right). \tag{16}$$

Algorithm 1: Violated Constraint Generation (VCG) Algorithm.

Data: $\overline{\mathbf{r}}$
Result: ε, $\boldsymbol{\lambda}$ and $s^f(\boldsymbol{\lambda})$
1 $\mathcal{V} = \emptyset$
2 **while** $|\mathcal{V}| \neq |\mathcal{L}|$ **do**
/* Store original paths of all flows */
3 $\mathcal{P}^f_{temp} \leftarrow \mathcal{P}^f \ \forall f \in \mathcal{F}$
4 Compute $\varepsilon_l \ \forall \ l \in \mathcal{L}$
5 Compute ε^{net}
/* Find link with largest excess */
6 Set $\hat{l} \leftarrow \underset{l \in \mathcal{L} \setminus \mathcal{V}}{\operatorname{argmax}} \ \varepsilon_l$
7 $\mathcal{V} = \mathcal{V} \cup \{\hat{l}\}$
/* Find flows that eliminate $\varepsilon_{\hat{l}}$ */
8 $\mathcal{S} = \emptyset$
9 Add largest $r^f_{\hat{l}}$ flows to $\mathcal{S}$ such that $\sum_{\forall f \in \mathcal{S}} r^f_{\hat{l}} \geq \varepsilon_{\hat{l}}$
/* Attempt to find alternative paths leading to a decrease in ε^{net} */
10 **for** $s \in \mathcal{S}$ **do**
11 **for** $k \in \mathcal{K}^s$ **do**
12 $\theta^s_k = \sum_{l \in \mathcal{L}} \left[\sum_{f \in \mathcal{F} \setminus \mathcal{S} \bigcup s} (r^f_l) - \overline{\mathbf{r}}_l \right]$
13 $\hat{k}^s = \underset{k \ \in \ \mathcal{K}^s}{\operatorname{argmin}} \ \theta^s_k$
14 Re-route on $\hat{k}^s \quad \forall s \in \mathcal{S}$ and compute $\varepsilon^{\text{net}}_{\text{reroute}}$
/* Check if re-route leads to a reduction of ε^{net} */
15 **if** $\varepsilon^{net}_{reroute} < \varepsilon^{net}$ **then**
16 Update $\mathcal{P}^f \leftarrow \hat{k}^s \ \forall f \in \mathcal{S}$
17 Set $\varepsilon^{net} \leftarrow \varepsilon^{\text{net}}_{\text{reroute}}$
18 **else**
19 $\mathcal{P}^f \leftarrow \mathcal{P}^f_{temp} \ \forall f \in \mathcal{S}$
20 **if** $\varepsilon^{net} > 0$ **then**
/* Compute violated constraint parameters */
21 Set $\boldsymbol{\lambda}_l \ \leftarrow \ 1$ for links l with $\varepsilon_l > 0$ and $\boldsymbol{\lambda}_l \ \leftarrow \ 0$, otherwise.
22 Compute $s^f(\boldsymbol{\lambda}) = \sum_{\forall l \in \mathcal{P}^f} \boldsymbol{\lambda}_l \ \ \forall f \in \mathcal{F}$
23 Generate constraint $\sum_{l \in \mathcal{L}} \boldsymbol{\lambda}_l \ \overline{r}_l \geq \sum_{f \in \mathcal{F}} r^f s^f(\boldsymbol{\lambda})$
24 **else**
25 $\overline{\mathbf{r}}$ is feasible and no constraint generated.

The larger the value of $\varphi(\overline{\mathbf{r}})$, the larger the infeasibility of $\overline{\mathbf{r}}$, and vice versa. A solution $\overline{\mathbf{r}}$ satisfying all constraints of RP^j gives a feasibility measure of zero, $\varphi(\overline{\mathbf{r}}) = 0$ [22].

The LRC algorithm consists of two phases. In the first phase (line 1 to line 10), it tries to improve the feasibility of the solution obtained in the last iteration by increasing the link rate on each link to one of the available higher rates. Link Rates are increased so that the largest feasibility improvement is achieved with the least energy cost. This improvement in feasibility relative to the energy cost increase is captured by the following profitability measure:

$$\Upsilon(l, y_l) = \frac{\varphi(\overline{\mathbf{r}}) - \varphi(\overline{\mathbf{r}}')}{\Gamma(\overline{\mathbf{r}}') - \Gamma(\overline{\mathbf{r}})}, \tag{17}$$

where $\overline{\mathbf{r}}'$ is equivalent to $\overline{\mathbf{r}}$ except for one link l where the link rate is increased to one of the higher link rates $y_l \in \overline{\mathcal{R}}$ such that $y_l > \overline{\mathbf{r}}_l$. Then the link-rate pairs $(l', y_{l'})$ with the maximum profitability ratio are identified and added to the set $\mathcal{M} = \{(l', y_{l'})\}$. If only one pair is identified, the link rate of link l' is set to $y_{l'}$. However, if more than one link is identified, the link rate of the link l' is set to the largest link excess $\varepsilon_{l'}$. Therefore, higher priority is given to increasing the rate of a link that can achieve not only the largest profitability but also the largest reduction in excess.

In the second phase, the LRC algorithm reduces the link rates $\overline{\mathbf{r}}$ on all links while maintaining feasibility (line 12 to line 20). For each link $l \in \mathcal{L}$, the $\overline{\mathbf{r}}_l$ is reduced to one of the lower rates $y_l \in \{\overline{\mathcal{R}} \mid y_l < \overline{\mathbf{r}}_l\}$, whereas the rates of other links remain the same; the updated set of rates is denoted by $\overline{\mathbf{r}}'$. If feasibility is maintained with a lower rate, the difference in energy cost

between $\overline{\mathbf{r}}$ and $\overline{\mathbf{r}}'$ is computed for each possible lower rate. The pairs of links and rates achieving the maximum reduction in energy cost $\Delta\Gamma(\cdot,\cdot)$ are selected (line 18). Among the selected pairs, the rate of the link with the smallest excess is reduced to the level that maximizes energy conservation. In this phase, a higher priority is given to reducing the rate on links with the least excess ε_l. The LRC algorithm returns the solution $\overline{\mathbf{r}}$ to the VCG algorithm that evaluates its feasibility.

Algorithm 2: Link Rate Control (LRC) Algorithm

Data: ε_l, $\boldsymbol{\lambda}_l^c$, and $s^f(\boldsymbol{\lambda}^c)\ \forall c \in \mathcal{C}^j$ and $\forall l \in \mathcal{L}$
Result: $\overline{\mathbf{r}}$
/* Increase a link rate on one of the links to improve feasibility of $\overline{\mathbf{r}}$ */
1 **if** $\varphi(\overline{\mathbf{r}}) \neq 0$ **then**
2 $\mathcal{M} = \emptyset$;
3 **foreach** $l \in \mathcal{L}$ **do**
4 $\overline{\mathbf{r}}' \leftarrow \overline{\mathbf{r}}$;
 /* Update current rate to a higher rate */
5 **foreach** $y_l \in \overline{\mathcal{R}} \mid y_l > \overline{\mathbf{r}}_l$ **do**
6 $\overline{\mathbf{r}}_l{}' = y_l$;
7 compute $\Upsilon(l, y_l) = \frac{\varphi(\overline{\mathbf{r}}) - \varphi(\overline{\mathbf{r}}')}{\Gamma(\overline{\mathbf{r}}') - \Gamma(\overline{\mathbf{r}})}$;
 /* Find the set of (link, rate) pairs with maximum profitability ratio */
8 $\mathcal{M} \cup \{(l', y_{l'})\} = \underset{\forall l, \forall y_l}{\operatorname{argmax}}\ \Upsilon(l, y_l)$;
 /* Among links found, choose the one with the largest $\varepsilon_{l'}$ */
9 $\hat{l} \leftarrow \underset{l' \mid (l', y_{l'}) \in \mathcal{M}}{\operatorname{argmax}}\ \varepsilon_{l'}$;
 /* Update rate of link $\hat{l}$ */
10 $\overline{\mathbf{r}}_{\hat{l}} \leftarrow y_{\hat{l}}$;
/* Reduce link rates while maintaining feasibility */
11 $\mathcal{Q} = \emptyset$;
12 **foreach** $l \in \mathcal{L}$ **do**
13 $\overline{\mathbf{r}}' \leftarrow \overline{\mathbf{r}}$;
 /* Update current rate to a lower rate */
14 **foreach** $y_l \in \overline{\mathcal{R}} \mid y_l < \overline{\mathbf{r}}_l$ **do**
15 $\overline{\mathbf{r}}'_l = y_l$;
16 **if** $\varphi(\overline{\mathbf{r}}'_l) = 0$ **then**
17 compute $\Delta\Gamma(l, y_l) = \Gamma(\overline{\mathbf{r}}) - \Gamma(\overline{\mathbf{r}}'_l)$;
 /* Find set of (link, rate) most energy efficient pairs */
18 $\mathcal{Q} \cup \{(l', y_{l'})\} = \underset{\forall l, \forall y_l}{\operatorname{argmax}}\ \Delta\Gamma(l, y_l)$;
 /* Among links found, choose the one with the least $\varepsilon_{l'}$ */
19 $\hat{l} \leftarrow \underset{l' \mid (l', y_{l'}) \in \mathcal{Q}}{\operatorname{argmax}}\ \varepsilon_{l'}$;
 /* Update rate of link $\hat{l}$ */
20 $\overline{\mathbf{r}}_{\hat{l}} \leftarrow y_{\hat{l}}$;

IV. Performance Evaluation

We carried out extensive computational experiments to evaluate the performance of the proposed solution. In this section, we provide a detailed description of our findings.

A. Experimental Setup

The proposed solution is compared to both CPLEX solver and Dijkstra's shortest path algorithm (SP). The proposed solution and SP were implemented in MATLAB; however, the problem in equations (5a) to (5d) was modeled in GAMS [34] and solved by CPLEX-12 solver [35]. CPLEX parameters were set to solve the problem to optimality; in specific, the relative termination tolerance was set to zero, i.e., `optcr=0.0`, and the maximum solving time was set to 500000 seconds, i.e., `reslim = 500000`. In the sequel we refer to the CPLEX solution as the "optimal solution". All computational experiments were performed on the same machine with 16 GB RAM and a 3.5 GHz 6-Core Intel Xeon E5 processor.

We considered well-known network topologies available at the SNDlib library [23] with various properties. The number of flows, FEs, and links as well as the average number of FEs and link density of these network topologies are tabulated in Table I. Furthermore, these topologies are shown in Figure 3. The origin $o(f)$ and destination $d(f)$ of each flow f in a given network were randomly selected. Furthermore, the flow rate was randomly generated following a uniform distribution between 50 to 200, i.e., $r^f \in [50, 200]\ \forall f \in \mathcal{F}$. Each experiment was repeated 10 times; a new seed was used in each experiment. The results presented here are averages of the ten runs results. Links operate at one of the available discrete link rates; these rates and their corresponding power consumption are given in Table II. The discrete rates used in our evaluation are based on the rates of Intel Ethernet Controller X540 [36]. The energy consumption is normalized to the time unit; therefore, the terms power and energy are used interchangeably in the sequel.

Table I: Properties of Network Topologies

Network	Flows	FEs	Links	Avg FE Degree	Link Density
Abilene	132	12	15	2.50	22.73 %
Atlanta	210	15	22	2.93	20.95 %
Polska	66	12	18	3.00	27.27 %
Nobel-US	91	14	21	3.00	23.08 %
Nobel-Germany	121	17	26	3.06	19.12 %
New York	240	16	49	6.12	40.83 %

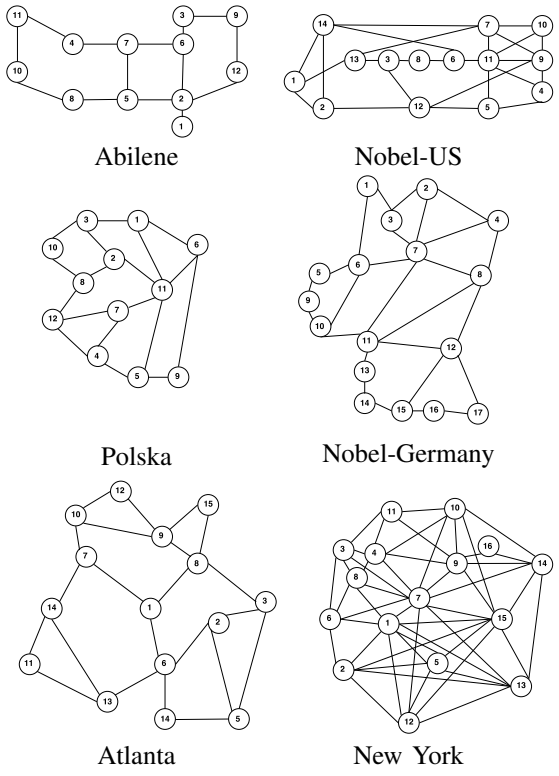
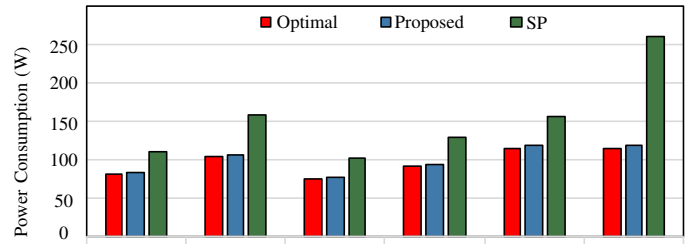


Figure 3: Network Topologies

Table II: Link Rates and their Power Consumption

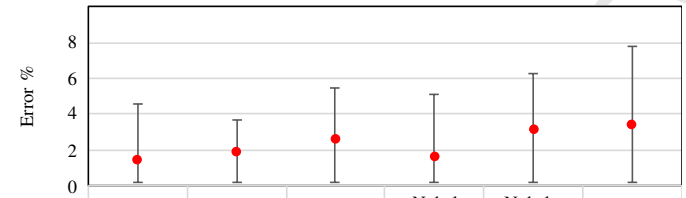
Link Rate	Power Consumption
100 Mbps	3.20 W
1 Gbps	4.27 W
10 Gbps	7.70 W

B. Network Power Consumption



	Abilene	Atlanta	Polska	Nobel - US	Nobel - Germany	NewYork
Optimal	82.28	105.28	74.59	91.53	114.95	114.57
Proposed	83.33	107.11	76.47	92.90	118.40	118.30
SP	110.36	159.26	101.28	130.17	156.25	259.27

(a)



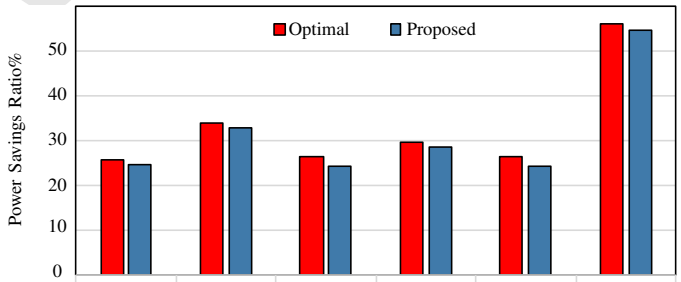
	Abilene	Atlanta	Polska	Nobel -US	Nobel-Germany	NewYork
Average Error %	1.33%	1.76%	2.50%	1.51%	3.01%	3.27%

(b)

Figure 4: Network Total Power Consumption (a) and percentage error of the proposed solution results in relation to the optimal (b).

The network power consumption of the proposed solution, CPLEX and SP are shown in Figure 4-(a). SP routes each demand individually on the shortest hop path irrespective of the power consumption, leading to least power-efficient routing. The results demonstrate that the proposed solution achieves a power consumption close to the optimal achieved by CPLEX. The error difference between the proposed solution and the optimal falls, on average, in the range 0% - 3.27% of the optimal as shown in Figure 4-(b). The error is proportional to the number of links in the topology. Although Polska has fewer links than Atlanta and Nobel-US, the proposed solution is shows a higher error rate for Polska because of its higher link density compared to Atlanta and Nobel-US.

C. Power Saving Compared to SP



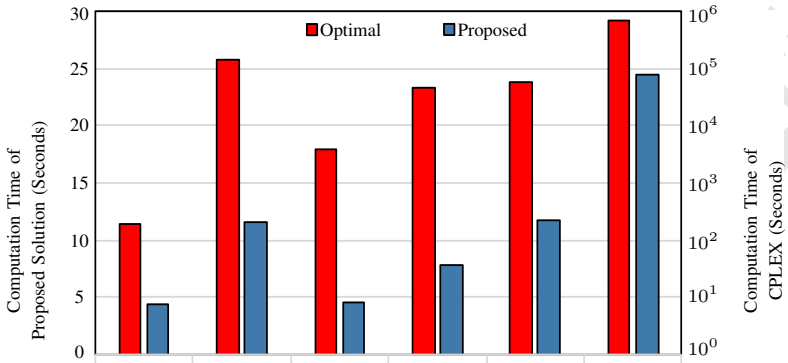
	Abilene	Atlanta	Polska	Nobel - US	Nobel - Germany	NewYork
Optimal	25.44%	33.88%	26.15%	29.64%	26.32%	55.79%
Proposed	24.47%	32.72%	24.22%	28.58%	24.12%	54.35%

Figure 5: Power saving compared to SP network power consumption.

The power saving ratio of the proposed solution and of CPLEX compared to SP are shown in Figure 5. The proposed solution brings significant power saving compared to SP; the power saving ranges from 24.22% to 54.35%. It reduces the power consumption of a highly connected network, New York, with 40.83% link density by 54.35%, whereas the reduction in

power consumption is 24.22% for the least connected network, Nobel-Germany, with 19.12% link density. The power saving of the remaining networks with comparable link density, Abilene, Polska, Noble-US and Atlanta, depends on the number of links in the network. The larger the number of links, the larger the power saving. SP routes each flow individually on the shortest path between the origin and destination; therefore, it enables more links than necessary for routing the flows. Conversely, the proposed solution coalesces flows to reduce the number of enabled links; thus, the network power efficiency is increased. In network topologies with high connectivity or link density, the proposed solution has a higher chance of grouping flows and reducing network power consumption and vice versa.

D. Computation Time

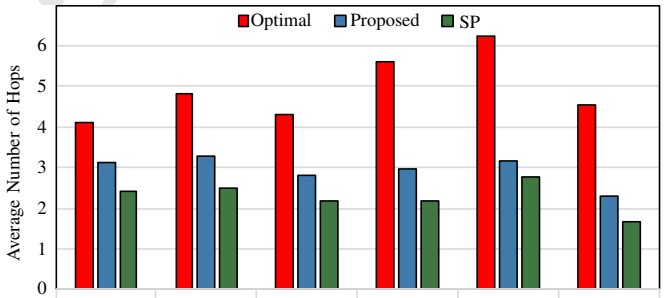


	Abilene	Atlanta	Polska	Nobel - US	Nobel - Germany	NewYork
Optimal	78.85	19684.89	988.28	7532.06	9587.72	76007.12
Proposed	4.32	11.49	4.61	7.87	11.66	24.39

Figure 6: Computation time of the proposed solution and CPLEX.

Figure 6 shows the computation times of the proposed solution and of CPLEX. The SP computation time is not shown because it is negligible. The average computation time of the proposed solution is 10.72 seconds, whereas the average computation time of CPLEX is 5.27 hours. The results demonstrate that the computation time of the proposed solution is proportional to the number of links in the network. It is interesting to observe that for highly connected networks like New York with 49 links, 40.83% link density and 240 demand, the proposed solution converges to the optimal in 24.39 seconds, whereas CPLEX generated the optimal after 21.11 hours.

E. Average Number of Hops



	Abilene	Polska	Nobel -US	Nobel-Germany	Atlanta	NewYork
Optimal	4.12	4.32	5.58	6.24	4.82	4.55
Proposed	3.12	2.80	2.98	3.15	3.27	2.23
SP	2.42	2.16	2.17	2.78	2.49	1.69

Figure 7: Average number of hops in various network topologies.

The average number of hops per path in the considered topologies is shown in Figure 7. SP provides a lower bound to the average number of hops. Despite the power efficiency achieved by the proposed solution, which may require longer paths to coalesce flows, the proposed solution outperforms CPLEX when considering path length in all networks. The average path length of routes generated by the proposed solution is, on average, 28% higher than that of SP, whereas the average path

length of CPLEX is 116.11% higher than that of SP. The proposed solution makes attempts to reroute flows on paths computed based on Yen's K-Shortest path routing algorithm; see Algorithm 1 Line 9. Therefore, it starts with the shortest path and iteratively routes flows through longer routes until a feasible solution is identified. In the following two subsections, we study the performance of the proposed solution under the following two scenarios. First, an additional flow is added to the set of flows in an existing network. Second, multiple flows are added to the set of existing flows. In both scenarios, the existing routing solution has to be updated to route both existing and new flows. This scheme represents a real scenario where flow routing requests simultaneously arrive at the central controller.

F. Single Flow and Multiple Flow Routing

The analysis in the previous subsections focused on concurrently routing the complete set of demands in each network; however, in this section we study the performance of the proposed solution when either a single flow or multiple flows are added to the network. The proposed solution is able to accommodate an additional flow without re-solving the whole routing problem. VCG generates the necessary violated constraints and appends them to the set of existing constraints. Then, LRC updates the link rates to satisfy the extended set of constraints.

For single flow experiments, we routed 50% (66 flows) of the Abilene network flows and iteratively added a single flow until all 132 flows were routed. Meanwhile, for the multiple flow-routing computational experiments, 72 flows were routed and an additional 10 flows were added for each iteration. In these experiments, we evaluated network power consumption, and the additional time required to update the routing solution.

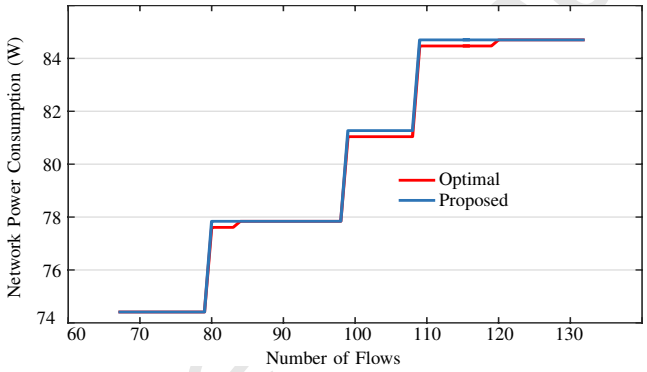


Figure 8: Total network power consumption of Abilene versus number of flows in the network. A single flow is added in each iteration.

Figure 8 shows that the proposed solution is able to achieve a routing power consumption close the optimum generated by CPLEX without re-solving the entire problem. For some flows, e.g., 70 to 75, the existing link rates are sufficient to route the additional five flows, and hence, the power consumption is fixed. However, when flow 80 is added one of the links had to be activated, and an additional 3.8 W was consumed.

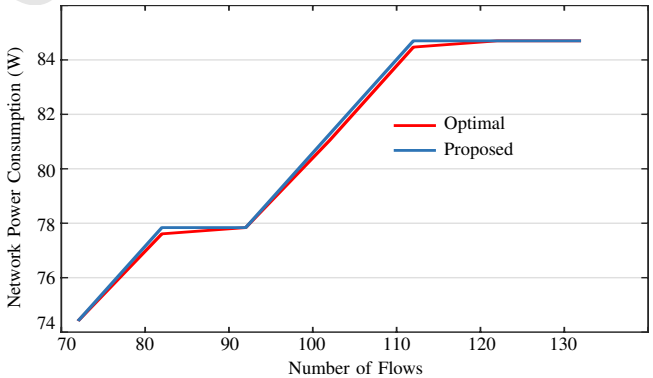


Figure 9: Total network power consumption of Abilene versus number of flows in the network. Multiple flows added in each iteration.

Figure 9 shows the network power consumption of the Abilene network when starting with an initial routing solution of 72 flows and adding 10 flows per iteration. The achieved power consumption closely follows the optimal generated by CPLEX.

An increase in power consumption is observed when adding the first 10 flows, i.e., 72 to 82 flows. However, the power consumption is fixed when the second 10 flows are added, i.e., 82 to 92 flows because the spare capacity of the active discrete link rates are sufficient to accommodate the additional 10 flows.

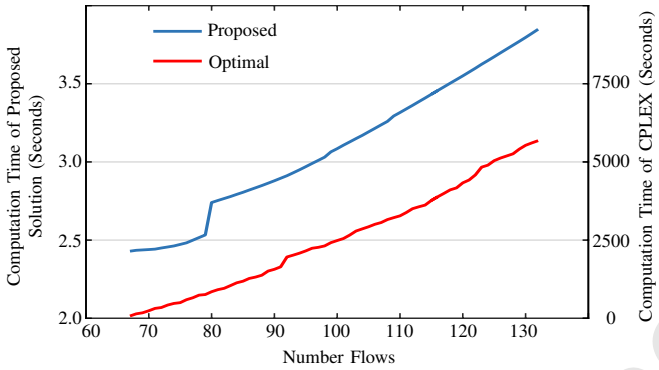


Figure 10: Computation time of the proposed solution and CPLEX when a single flow is added to the network.

Figure 10 presents the computation time of both the proposed solution and CPLEX when a single flow is added at a time. Due to the large difference in order, we plotted their computation times on different axes. The average computation time of the proposed solution is 22 milliseconds for each additional flow, whereas that of CPLEX is 1.43 seconds. Furthermore, the proposed solution routed the additional 66 flows in 1.43 seconds, whereas CPLEX routed them in 1.58 hours because routes were recomputed for all flows when a single flow was added.

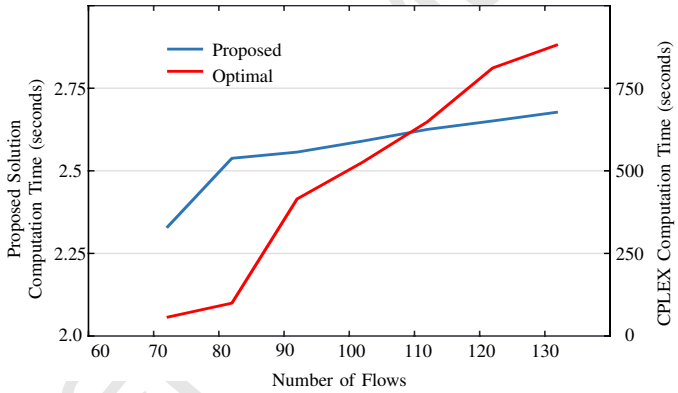


Figure 11: Computation time of the proposed solution and CPLEX when multiples of ten flows are added to the network.

Figure 11 shows the computation time required to route additional multiple flows added to the network; 10 flows were added in each iteration. The average computation time of the proposed solution is 58 milliseconds to route each set of additional 10 flows; however, the average computation time of CPLEX for the additional 10 flows is 2.3 minutes. The proposed solution routed the additional 72 flows in 0.35 seconds, whereas CPLEX routed them in 825 seconds.

V. Conclusions

In this paper, we investigated how to optimize the energy efficiency of the routing network application in software-defined networks. We considered practical constraints of real networks; namely, integral routing and discrete link rates. We modeled the routing problem as a MILP problem and proposed a heuristic solution that consists of two algorithms, LRC and VCG. The proposed solution routes a single or multiple flows which is desired in dynamic networks. The performance of the proposed solution was evaluated on six real networks with various topological structures and properties. In all experiments, the proposed solution showed supremacy over the shortest path algorithm and achieved power saving of 54.35%. Furthermore, our algorithm achieves a close-to-optimal performance when compared to CPLEX; the error is less than 3.27%.

Acknowledgment

This project was funded partially by Kuwait Foundation for the Advancement of Sciences under project code: P314-35EO-01.

REFERENCES

[1] B. Sanou, "ICT facts & figures – the world in 2015," Telecommunication Development Bureau, International Telecommunication Union, www.itu.int/ict, 2015.

[2] E. Gelenbe and Y. Caseau, "The impact of information technology on energy consumption and carbon emissions," *Ubiquity*, vol. 2015, no. June, pp. 1:1–1:15, Jun. 2015. [Online]. Available: http://doi.acm.org/10.1145/2755977

[3] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide ict electricity consumption from 2007 to 2012," *Comput. Commun.*, vol. 50, pp. 64–76, Sep. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2014.02.008

[4] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 123–134. [Online]. Available: http://doi.acm.org/10.1145/1592568.1592584

[5] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys Tutorials*, vol. 13, no. 2, pp. 223–244, Second 2011.

[6] G. Fettweis and E. Zimmermann, "ICT energy consumption-trends and challenges," in *Proceedings of the 11th International Symposium on Wireless Personal Multimedia Communications (WPMC'08)*, vol. 2, no. 4, Lapland, Finland, September 2008, p. 6.

[7] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *CoRR*, vol. abs/1406.0440, 2014. [Online]. Available: http://arxiv.org/abs/1406.0440

[8] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, May 2015.

[9] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.

[10] ——, "Routing and scheduling for energy and delay minimization in the powerdown model," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–5.

[11] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation." in *NSDI*, vol. 8, 2008, pp. 323–336.

[12] Y. Shi, F. Zhang, J. Wu, and Z. Liu, "Randomized oblivious integral routing for minimizing power cost," *Theoretical Computer Science*, vol. 607, Part 2, pp. 221 – 246, 2015, frontiers of Algorithmics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397515006283

[13] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," in *NETWORKING 2009*. Springer, 2009, pp. 795–808.

[14] M. K. Awad, G. Neama, and Y. Rafique, "The impact of practical network constraints on the performance of energy-aware routing schemes," in *Service Operations And Logistics, And Informatics (SOLI), 2015 IEEE International Conference on*, Nov 2015, pp. 77–81.

[15] A. Markiewicz, P. N. Tran, and A. Timm-Giel, "Energy consumption optimization for software defined networks considering dynamic traffic," in *IEEE 3rd International Conference on Cloud Networking (CloudNet'14)*, Luxembourg, Oct 2014, pp. 155–160.

[16] R. Wang, Z. Jiang, S. Gao, W. Yang, Y. Xia, and M. Zhu, "Energy-aware routing algorithms in software-defined networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, June 2014, pp. 1–6.

[17] H. Zhu, X. Liao, C. de Laat, and P. Grosso, "Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform," *Journal of Network and Computer Applications*, vol. 63, pp. 110 – 124, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516000473

[18] L. Wang, A. F. Anta, F. Zhang, J. Wu, and Z. Liu, "Multi-resource energy-efficient routing in cloud data centers with networks-as-a-service," *CoRR*, vol. abs/1501.05086, 2015. [Online]. Available: http://arxiv.org/abs/1501.05086

[19] L. Wang, A. Fernández Anta, F. Zhang, C. Hou, and Z. Liu, *Theory and Applications of Models of Computation: 9th Annual Conference, TAMC 2012, Beijing, China, May 16-21, 2012. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Energy-Efficient Network Routing with Discrete Cost Functions, pp. 307–318. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29952-0_32

[20] J. Tang, B. Mumey, Y. Xing, and A. Johnson, "On exploiting flow allocation with rate adaptation for green networking," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 1683–1691.

[21] V. Gabrel, A. Knippel, and M. Minoux, "Exact solution of multicommodity network optimization problems with general step cost functions," *Operations Research Letters*, vol. 25, no. 1, pp. 15–23, 1999.

[22] ——, "A comparison of heuristics for the discrete cost multicommodity network optimization problem," *Journal of Heuristics*, vol. 9, no. 5, pp. 429–445, 2003.

[23] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC'07)*, Spa, Belgium, April 2007. [Online]. Available: http://sndlib.zib.de

[24] (2016, June) Open networking foundation. [Online]. Available: https://www.opennetworking.org/about/onf-overview

[25] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *INFOCOM, 2013 Proceedings IEEE*, April 2013, pp. 2211–2219.

[26] L. Wang, A. F. Anta, F. Zhang, C. Hou, and Z. Liu, "Routing for energy minimization with discrete cost functions," *Computing Research Repository (CoRR'13)*, vol. abs/1302.0234, 2013.

[27] M. Minoux, "Discrete cost multicommodity network optimization problems and exact solution methods," *Annals of Operations Research*, vol. 106, no. 1, pp. 19–46, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1014554606793

[28] M. Gondran, M. Minoux, and S. Vajda, *Graphs and algorithms*. John Wiley & Sons, Inc., 1984.

[29] M. Iri, "On an extension of the maximum-flow minimum-cut theorem to multicommodity flows," *Journal of the Operations Research Society of Japan*, vol. 13, no. 3, pp. 129–135, 1971.

[30] M. Stoer and G. Dahl, "A polyhedral approach to multicommodity survivable network design," *Numerische Mathematik*, vol. 68, no. 1, pp. 149–167, 1994.

[31] M. Mrad and M. Haouari, "Optimal solution of the discrete cost multicommodity network design problem," *Applied Mathematics and Computation*, vol. 204, no. 2, pp. 745–753, 2008.

[32] A. M. Costa, J.-F. Cordeau, and B. Gendron, "Benders, metric and cutset inequalities for multicommodity capacitated network design," *Computational Optimization and Applications*, vol. 42, no. 3, pp. 371–392, 2009.

[33] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[34] R. E. Rosenthal, *GAMS–A User's Guide*, Dec 2014.

[35] *IBM ILOG CPLEX 12.4 User's Manual*, IBM ILOG, 2012.

[36] *Intel Ethernet Controller X540 Datasheet Rev. 2.7*, Intel Corporation, March 2014.