An Energy Efficient Integral Routing Algorithm for Software-defined Networks

Ghadeer Naji Neama, *Student Member*, *IEEE*, Mohamad Khattar Awad, *Member*, *IEEE*
Department of Computer Engineering, College of Computing Sciences and Engineering, Kuwait University, Kuwait
E-mail: ghadeer.neama.kw@ieee.org, mohamad@ieee.org

***Abstract*—The exponential growth of the Information and Communication Technology (ICT) sector have led to a significant increase in energy consumption, higher electricity bills, and negative environmental and economical impacts. Several researchers, network providers, and manufacturers have been investigating different approaches to improve the energy efficiency of communication networks. Software-defined Networks (SDN) is emerging as a new networking framework that separates data plane from control plane in order to simplify network management, reduce operational costs (OPEX), and facilitate innovation. In this work, we address the centralized integral routing problem in SDN. We propose a greedy heuristic algorithm called Energy Efficient Integral Routing (EEIR) algorithm to minimize power consumption in SDN backbone networks while respecting discreteness of link rates. The performance of EEIR has been evaluated in real topologies, and compared to both optimal and shortest path solutions. Experimental results have shown a significant power saving that is as large as 44.42% can be achieved. Compared to optimal solution, EEIR provides a solution with an optimality gap in the range 7.52% -12.67%.**

***Index Terms*—Energy-aware routing, Software-defined Networks, Network Optimization.**

I. Introduction

Information and communication technology (ICT) is considered as an important knowledge sharing base. As this base is continuing to grow, e.g., widespread deployment of data centers and increasing number of Internet users, the energy consumption is becoming a major concern. Electricity costs of ICT has increased up from 4% to 4.7% during a period from 2007 to 2012 [1], [2], and this fraction is expected to be doubled by 2020 [3]. Moreover, ICT is one of the contributors to overall green house gases (GHG) emissions, being responsible for generating more than 2% of carbon emissions world wide [4], [5].

Software-defined networking (SDN) emerges as a network architecture that separates control and data functionalities with the promise to simplify network management, enhance resource utilization, reduce operation costs, and support innovation of the network applications and services for power efficiency. SDN architecture comprises of three main layers, namely application layer, control layer, and infrastructure layer. The application layer represents business services and applications, e.g., access control, traffic and security monitoring, and energy efficient networking. SDN applications communicate network requirements and behavior to SDN control layer through the north bound interface (NBI). The control layer contains the centralized controller that executes routing algorithm to compute the logic of forwarding packets and installs it in the forwarding table for all Forwarding Elements (FEs) in the network.

Recently, several works have been investigating how to reduce power consumption through energy efficient routing schemes. Wang *et. al.* [6], proposed two greedy algorithms to save power of idle network devices, while putting them into sleep mode. The proposed algorithms were shown to be effective in minimizing power consumption of unnecessary network devices, e.g., link chassis, while maintaining minimum number of operating devices under maximum link utilization and maximum packet delay constraints. Razmnoush and Bakhshi [7], presented an on-line centralized approach to route and manage all traffic flows in backbone networks while switching off maximum number of links. The solution is composed of two stages, namely routing and monitoring traffic flows. In routing stage, algorithm tries to route all traffic flows through minimum number of on-links and nodes, whereas in monitoring stage, algorithm makes sure that links utilization is bounded by under-utilization and over-utilization thresholds. Giroire *et. al.* [8], proposed a greedy heuristic algorithm to optimize energy consumption in backbone SDN networks. Algorithm was implemented to route flows in network with respect to capacity constraint on links, and rule space constraint on routers. The previously mentioned algorithms provided close to optimal solutions, and have been shown effective in saving energy consumed by networks. However, they overlooked the discreteness of link rates and considered the cost function to be continuous. In practice, each link can operate at one of the discrete link rates; thus, deploying the former algorithms is impractical.

Unlike the aforementioned solutions, we consider a practical constraint, that is the discreteness of link rates. In our previous work [9], we evaluated the performance of energy aware routing algorithms in wired networks under both continuous and discrete link rates assumptions. We demonstrated that considering discrete link rates is important to achieve higher power savings in real networks. In this work, we aim to solve the centralized routing problem in SDN while respecting discrete link rates, maximum link capacity, and flow conservation constraints.

- We propose an efficient greedy heuristic algorithm that finds near optimal solution to the centralized routing

problem, called Energy Efficient Integral Routing (EEIR) algorithm.
- Using real network traffic matrices, we compare our algorithm performance with results obtained by GAMS/CPLEX solvers [10], [11], and shortest path approach.

The remainder of the paper is organized as follows. System model and routing problem formulation are presented in Section II. Energy efficient integral routing (EEIR) algorithm is proposed in Section III. Numerical results and discussions are presented in Section IV. Finally, conclusions are drawn in Section V.

II. Network Model and Problem formulation

A. Network Model

A similar problem formulation was treated in [9], [12]–[14]. The software-defined network topology is given as a bidirectional graph $G(\mathcal{V},\mathcal{E})$, where $\mathcal{V}=\{1,2,\cdots,V\}$ represents a set of vertices and $\mathcal{E}$ represents a set of bidirectional edges. $\mathcal{N}(u)$ models the set of neighbors for each node $u\in\mathcal{V}$. Let $C_{(u,v)}$ represent the capacity of the link connecting node u to node v. Each link $(u,v)\in\mathcal{E}$ can operate at one of the available discrete transmission rates $\mathcal{R}=\{R_0,R_1,R_2,\cdots,R_{\max}\}$, where $R_{max}\leq C_{(u,v)}$. The power consumption for each link (u,v), denoted by $g(z_{(u,v)})$, depends on its transmission rate $z_{(u,v)}\in\mathcal{R}$ and is given by

$$g(z_{(u,v)})=\begin{cases}P_0 & z_{(u,v)}=R_0\\ P_1 & z_{(u,v)}=R_1\\ \vdots & \vdots\\ P_{\max} & z_{(u,v)}=R_{\max}\end{cases} \tag{1}$$

Let $\mathcal{D}$ represent the set of all traffic demands in the network, and (s,t) denote a traffic demand going from source node $s\in\mathcal{V}$ to destination node $t\in\mathcal{V}$, and $t\neq s$. Let d^{st} be the demand bandwidth exchanged between source node s and destination node t. Total flow on all links, routers, besides available transmission rates on each link are available to the central controller. Therefore, the controller computes the optimal integral path for each demand in $\mathcal{D}$, from source node to destination node.

B. Problem Formulation

In this section, the problem of integral centralized routing in SDN is formulated in order to satisfy available discrete transmission rates, satisfy flow conservation constraint and satisfy total link capacity constraint. The objective of centralized routing is to find the optimal integral routes for all demands in $\mathcal{D}$ with the goal of minimizing total power consumption of the network. Let $x_{(u,v)}\in\{0,1\}$, be the binary variable that takes the value of 1 if the link (u,v) is active, i.e. powered ON, and zero otherwise. The network energy consumption is a function of ON links, their transmission rates, and the corresponding power consumption. Therefore, the objective function can be written as

$$\sum_{(u,v)\in\mathcal{E}} x_{(u,v)}\cdot g(z_{(u,v)}). \tag{2}$$

Let $f^{st}_{(v,u)}\in\{0,1\}$, denotes the binary variable that indicates whether a flow (s,t) is going out of node v and entering node u. In other words, $f^{st}_{(v,u)}$ takes the value of 1 if there is a flow from source node v to destination node u, and zero otherwise. Based on this definition, the flow conservation constraint is stated as follows

$$\sum_{v\in\mathcal{N}(u)} d^{st}(f^{st}_{(v,u)}-f^{st}_{(u,v)})=\begin{cases}-d^{st}, & \text{if } u=s\\ d^{st}, & \text{if } u=t\\ 0 & \text{otherwise}\end{cases}$$
$$\forall u\in\mathcal{V},(s,t)\in\mathcal{D}. \tag{3}$$

Operating discrete transmission rate $z_{(u,v)}$ on link (u,v), should satisfy total traffic flow $l_{(u,v)}$ requirements, i.e., all d^{st} routed through this link.

$$l_{(u,v)}=\sum_{(s,t)\in\mathcal{D}} d^{st}(f^{st}_{(u,v)}\vee f^{st}_{(v,u)})\leq z_{(u,v)} \qquad \forall(u,v)\in\mathcal{E}. \tag{4}$$

In addition, the selected transmission rate $z_{(u,v)}$ is restricted to one of the available discrete rates in $\mathcal{R}$, i.e.,

$$z_{(u,v)}\in\{R_0,\ldots,R_{max}\} \qquad \forall(u,v)\in\mathcal{E}. \tag{5}$$

Furthermore, maximum transmission rate R_{max} should satisfy the link capacity constraint, that is

$$R_{max}\leq C_{(u,v)}x_{(u,v)} \qquad \forall(u,v)\in\mathcal{E}. \tag{6}$$

Given the previous definitions, our discrete integral centralized routing problem can be formulated as follows:

$$\min \sum_{(u,v)\in\mathcal{E}} x_{(u,v)}\cdot g(z_{(u,v)}) \tag{7}$$
$$\text{s.t. equations (3), (4), (5), (6).} \tag{8}$$

The presented problem formulation in (7) falls in the class of Mixed Integer Linear Programming (MILP) problems. Generally speaking, MILP problems are known to be difficult to solve. Furthermore, discreteness of objective function, and existence of binary variables $x_{(u,v)}$ and $f^{st}_{(u,v)}$ make the problem NP-hard and can not be solvable in polynomial time; thus, we propose a lightweight greedy algorithm to solve this problem.

III. Proposed Energy Efficient Integral Routing Algorithm

Optimizing the number of active links has been widely used in green networking, and has shown to be effective in reducing the power consumption [15]–[17]. In this section we propose an algorithm that aims to reduce the power consumption by switching OFF unnecessary links while maintaining routes feasibility for all demands in the network.

We extended the algorithm developed in [15] to solve the problem under consideration. The EEIR starts by generating

set of shortest paths $\mathcal{S}$ for all demands in $\mathcal{D}$ while all links are powered ON. After computing shortest path for each demand in $\mathcal{D}$, the algorithm turns OFF unused links and updates the set of ON links $\mathcal{E}_{ON}$. Iteratively, EEIR selects a candidate link, $(u,v)'$, with the highest residual capacity $r_{(u,v)}$ and tries to turn it OFF, while satisfying all demands in the network. If a traffic demand (s,t) is affected by turning a link $(u,v)'$ OFF, the algorithm generates set of alternative shortest paths $\mathcal{K}_{(s,t)}$ to reroute that demand. However, if link $(u,v)'$ cannot be turned OFF, the algorithm stores link $(u,v)'$ in $\mathcal{E}_{fixed}$. The pseudo-code of EEIR is given in Algorithm 1. The algorithm goes through three major steps, which are described as follows:

- Step 1 (lines 2-4): EEIR generates a shortest path $SP_{(s,t)}$ for each demand in $\mathcal{D}$ using Dijkstra's algorithm, and stores it in the set of shortest paths $\mathcal{S}$. Shortest path is selected based on minimum number of hops. In the same step, the bandwidth d^{st} is reserved along the shortest path $SP_{(s,t)}$ for each demand.
- Step 2 (lines 5-19): In this step, the algorithm turns OFF as many links as possible and saves energy while maintaining feasible routes for all demands in the network. Given the computed shortest path for each demand, EEIR computes total flow $l_{(u,v)}$ on each link (u,v). Next, EEIR computes the least transmission rate $z_{(u,v)}$ required to support $l_{(u,v)}$. Furthermore, EEIR calculates residual capacity using the values of total flow on each link, and removes unused links. For instance, if total flow $l_{(u,v)} = 0$ on link (u,v), the link is unused, transmission rate $z_{(u,v)}$ is set to $zero$, and the link is removed from $\mathcal{E}_{ON}$. However, if total flow $0 < l_{(u,v)} \leq R_1$, $z_{(u,v)}$ is set to link operating rate R_1, and the residual capacity for this link is $r_{(u,v)} = R_1 - l_{(u,v)}$ Gbps.
- Step 3 (lines 20-32): EEIR repetitively selects a link $(u,v)'$ with the highest residual capacity $r_{(u.v)}$ such that $(u,v)' \in \mathcal{E}_{ON}$ and $(u,v)' \notin \mathcal{E}_{fixed}$, and checks if it is feasible to lower the transmission rate of the selected link one level down. In each iteration, the algorithm runs a greedy heuristic function, called $Reroute(\cdot)$ and given in Algorithm 2, to determine the feasibility of degrading the transmission rate of the selected link $(u,v)'$ one level. The function tries to reroute one or multiple demands necessary to take rate of selected link one level down. Selecting demands to be rerouted depends on value of excess rate $e_{(u,v)'}$ over degraded link rate $z'_{(u,v)'}$. If rerouting is feasible, the transmission rate of link $(u,v)'$ is degraded one level, and any link $(u,v)'$ that has $z'_{(u,v)'} = 0$ is removed from $\mathcal{E}_{ON}$ (lines 22-29). However, if rerouting is infeasible, link $(u,v)'$ is stored in $\mathcal{E}_{fixed}$. Step 3 is repeated until it is not possible to degrade the transmission rate of any of the remaining links, i.e. $\mathcal{E}_{ON} = \mathcal{E}_{fixed}$, and thus EEIR terminates.

$Reroute(\cdot)$ function is used to check whether taking the transmission rate $z'_{(u,v)'}$ of the selected link $(u,v)'$ one level down is feasible, i.e., the degraded operating rate on link $(u,v)'$ and all other ON links in $\mathcal{E}_{ON}$ can still satisfy all demands in the network.

Algorithm 1: EEIR

1 $\mathcal{E}_{ON} \leftarrow \mathcal{E}$, $\mathcal{E}_{fixed} \leftarrow \phi$
2 **foreach** $(s,t) \in \mathcal{D}$ **do**
3 Generate shortest path $SP_{(s,t)}$ and reserve bandwidth d^{st} through $SP_{(s,t)}$;
4 Add $SP_{(s,t)} \in \mathcal{S}$;
5 **foreach** $(u,v) \in \mathcal{E}$ **do**
  /* Compute total $flow$ on each link */
6 $l_{(u,v)} \leftarrow \sum_{(s,t)\in\mathcal{D}} d^{st} \quad \forall (u,v) \in SP_{(s,t)}$
7
  /* Find the least transmission rate required to support $l_{(u,v)}$, find residual capacity, and remove unused links */
8 **if** $l_{(u,v)} = 0$ **then**
9 $z_{(u,v)} \leftarrow 0$;
  /* Turn OFF link (u,v) */
10 $\mathcal{E}_{ON} \leftarrow \mathcal{E}_{ON} \setminus \{(u,v)\}$;
11 **else if** $0 < l_{(u,v)} \leq R_1$ **then**
12 $z_{(u,v)} \leftarrow R_1$;
13 $r_{(u,v)} \leftarrow R_1 - l_{(u,v)}$;
14 ⋮
15 **else if** $R_{\max -1} < l_{(u,v)} \leq R_{\max}$ **then**
16 $z_{(u,v)} \leftarrow R_{\max}$;
17 $r_{(u,v)} \leftarrow R_{\max} - l_{(u,v)}$;
18 **while** $|\mathcal{E}_{ON}| > |\mathcal{E}_{fixed}|$ **do**
  /* Find link with highest residual capacity */
19 $(u,v)' \leftarrow \max_{(u,v)\notin\mathcal{E}_{fixed}} r_{(u,v)} \ \forall (u,v) \in \mathcal{E}_{ON}$
  /* Call $Reroute(\cdot)$ to check if rerouting is feasible */
20 **if** $Reroute(\mathcal{E}_{ON}, (u,v)') = true$ **then**
  /* Take transmission rate one level down */
21 **if** $z_{(u,v)'} = R_{\max}$ **then**
22 $z'_{(u,v)'} \leftarrow R_{\max -1}$;
23 ⋮
24 **else if** $z_{(u,v)'} = R_1$ **then**
25 $z'_{(u,v)'} \leftarrow 0$;
  /* Remove link $(u,v)'$ */
26 $\mathcal{E}_{ON} \leftarrow \mathcal{E}_{ON} \setminus \{(u,v)'\}$;
27 **else**
  /* Transmission rate of this link cannot be degraded */
28 $\mathcal{E}_{fixed} = \mathcal{E}_{fixed} \cup \{(u,v)'\}$;
29 Terminate;

At initialization, $Status$ is set to true. The $Reroute(\cdot)$ function consists of two main steps that are described as follows:

In step 1 (lines 2-10), the function computes the excess rate $e_{(u,v)'}$ over degraded link rate $z'_{(u,v)'}$. Given $e_{(u,v)'}$, the function finds the set of demands $\mathcal{D}'$ such that $\sum_{(s,t)\in\mathcal{D}'} d^{st} \geq e_{(u,v)'}$. Note that only demand(s) that allow switching the transmission rate to a lower level are rerouted. The function searches for each $SP_{(s,t)}$ of $(s,t) \in \mathcal{D}$, that uses the selected link $(u,v)'$ and adds the demands bandwidth d^{st} for all affected demands to residual capacity of each link in $SP_{(s,t)}$.

Algorithm 2: $Reroute(\mathcal{E}_{ON}, (u,v)')$

Input: $(\mathcal{E}_{ON}, (u,v)')$
Output: $Status$
1 $Status \leftarrow true$;
/* Lowering the rate one level down */
2 **if** $z_{(u,v)'} = R_{\max}$ **then**
3 $\quad z'_{(u,v)'} \leftarrow R_{\max-1}$;
4 $\quad\vdots$
5 **else if** $z_{(u,v)'} = R_1$ **then**
6 $\quad z'_{(u,v)'} \leftarrow 0$;
$\quad$/* Remove link $(u,v)'$ */
7 $\quad\quad \mathcal{E}_{ON} \leftarrow \mathcal{E}_{ON} \setminus \{(u,v)'\}$;
/* Find excess rate over degraded link rate $z'_{(u,v)'}$ */
8 $e_{(u,v)'} = l_{(u,v)'} - z'_{(u,v)'}$;
/* Find set of demands $\mathcal{D}'$ required to switch link $(u,v)'$ to a lower rate */
9 $\mathcal{D}' = \{(s,t) : \sum_{(s,t)\in\mathcal{D}'} d^{st} \geq e_{(u,v)'}\}$
10 **foreach** $(s,t) \in \mathcal{D}'$ **do**
11 $\quad$ **if** $SP_{(s,t)}$ *contains link* $(u,v)'$ **then**
$\quad\quad$/* Add demand bandwidth d^{st} to residual capacity */
12 $\quad\quad r_{(u,v)'} \leftarrow r_{(u,v)'} + d^{st}$;
/* Reroute on alternative routes */
13 **foreach** $(s,t) \in \mathcal{D}'$ **do**
14 $\quad$ Find $\mathcal{K}_{(s,t)}$ shortest paths to reroute (s,t)
15 $\quad$ **foreach** $P_k \in \mathcal{K}_{(s,t)}$ **do**
16 $\quad\quad Status \leftarrow false$;
17 $\quad\quad$ **if** $r_{(u,v)'} \geq d^{st}\ \forall (u,v) \in P_k$ **then**
18 $\quad\quad\quad r_{(u,v)'} \leftarrow r_{(u,v)'} - d^{st}$;
19 $\quad\quad\quad$ Replace $SP_{(s,t)} \in \mathcal{S}$ with P_k;
20 $\quad\quad\quad Status \leftarrow true$;
21 $\quad\quad\quad$ break;
22 $\quad$ **if** $Status = false$ **then**
23 $\quad\quad$ break;
24 **return** $Status$;

In step 2, the function calls $k\text{-}shortestpaths$ algorithm [18] to generate a set of alternative shortest paths $\mathcal{K}_{(s,t)}$ for each demand in $\mathcal{D}'$. In the same step, the function checks if it is feasible to route the affected demand(s) through the new shortest path P_k; i.e., the link rate on all links $(u,v) \in P_k$ are sufficient to route demand bandwidth d^{st} and existing demands. The function then updates the residual capacity of each link in P_k, and replaces the existing path $SP_{(s,t)}$ by P_k. However, if any link in P_k does not have sufficient residual capacity, the path P_k is ignored. The function returns $false$ if it cannot reroute all selected demands.

Table I: Properties of Considered Network Topologies

Network	Number of nodes	Number of links
pdh	11	34
di-yuan	11	42
dfn-bwin	10	45
dfn-gwin	11	47

IV. Experimental Results

In this section, we provide the performance evaluation results of EEIR.

A. Experimental Setup

We evaluated the performance of the proposed algorithm in four real backbone networks. The considered topologies are pdh, di-yuan, dfn-bwin, and dfn-gwin available at SNDlib [19]. Both dfn-bwin and dfn-gwin topologies are used to connect several universities and research foundations. Pdh is a highly connected network that covers 11 industrial regions. Di-yuan is a metropolitan optical city network. Table I, lists the properties of all considered networks, and Figure 1 depicts their topologies.

The number of traffic demands ranges from 30 to 43. Each demand is established between random source s and destination t, while its bandwidth d^{st} is uniformly distributed in the range $[50, 300]$ Mbps. Link operate at one of the available discrete transmission rates: 100 Mbps, 1 Gbps, and 10 Gbps with power consumption of 3.2 Watts, 4.27 Watts, and 7.7 Watts, respectively [20].

The proposed algorithm and shortest path are implemented in Matlab and all experiments are performed on 2.8 GHz Intel core i5 PC with 8 GB RAM. However, optimal solution is modeled in GAMS, solved by CPLEX, and is performed on 3.5 GHz 6-Core Intel Xeon E5 processor, running MAC OS X 10.11 with 16 GB RAM. The presented results reported in this work are averaged over 15 runs.

B. Performance Results

In this section, we analyze the performance findings of EEIR over the four topologies and compare it against CPLEX and shortest path solutions.

Figure 2 shows average power consumption for routing all traffic demands using CPLEX, EEIR, and shortest path (SP) algorithms over the four networks. We can observe that average power consumption of using SP is higher than both optimal and EEIR solutions. This observation is expected because SP optimizes routes based on hop count rather than energy consumption. However, the optimal and EEIR solutions try to switch OFF the maximum possible number of links, resulting in reducing the power consumption. We can observe that EEIR has very close performance to CPLEX performance in terms of average network power consumption; 56.38% vs.

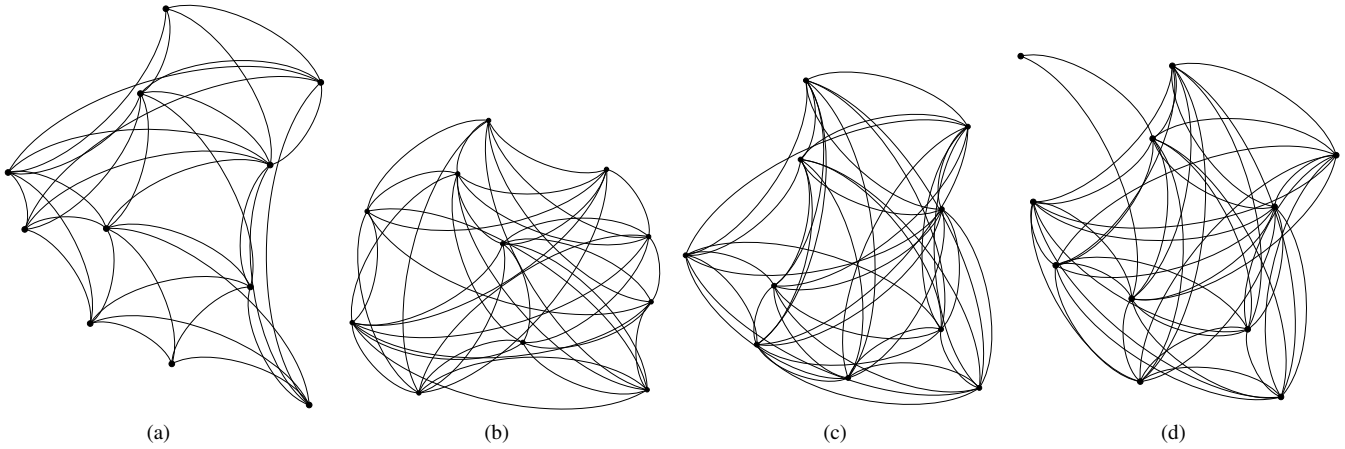

Figure 1: Considered network topologies: (a) pdh, (b) di-yuan, (c) dfn-bwin, and (d) dfn-gwin

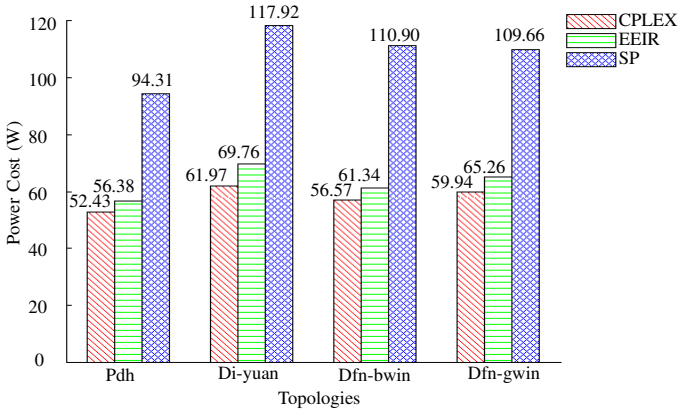

Figure 2: A comparison of the average network power consumption for routing all traffic demands considering optimal, proposed, and SP solutions.

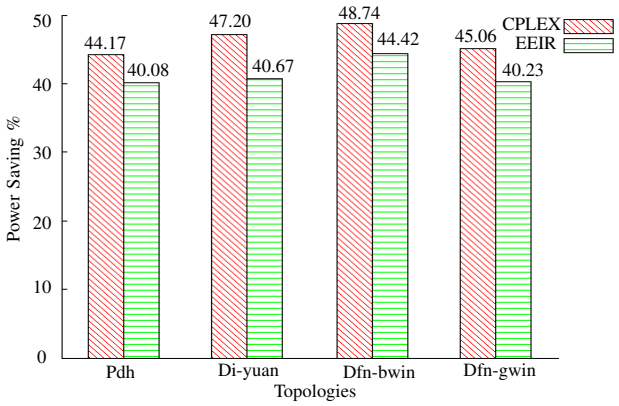

Figure 3: A comparison of the energy savings of the optimal vs. the proposed solutions.

52.43% for pdh, 69.76% vs. 61.97% for di-yuan, 61.34% vs. 56.57% for dfn-bwin, and 65.26% vs. 59.94% for dfn-gwin.

Figure 3 shows the power savings achieved by both optimal and EEIR solutions relative to SP, i.e., the percentage of power saving that can be achieved by switching OFF some links. Results demonstrate that both optimal and EEIR solutions effectively reduce power consumption. Compared to CPLEX power saving, i.e., 44.17% - 48.74%, on average EEIR can achieve a significant power saving in the range 40.08% - 44.42%.

Figure 4 compares the average running time of EEIR and CPLEX. The average computation time consumed by CPLEX to find a solution is 1 hour for pdh, and more than 24 hours for the other three networks. However, EEIR finds a solution within 1.28 seconds, 3.73 seconds, 2.71 seconds, and 2.37 seconds for pdh, di-yuan, dfn-bwin, and dfn-gwin, respectively. Clearly, EEIR proves to be computationally efficient taking at most 3 seconds to generate a routing solution. Compared to CPLEX, EEIR solves the centralized integral routing problem much faster. Note that the proposed algorithm is implemented in Matlab, a faster performance can be achieved by implementing the algorithm in C or C++.

Figure 5 shows the average path length of routes generated by CPLEX, EEIR, and SP solutions. As expected SP has the shortest path length since it routes demands based on minimum number of hops. SP optimizes the path length and does not consider reducing power consumption. Both solutions, EEIR and optimal, optimize the energy consumption by switching OFF links, which reflects an increase in path length compared to SP. It is clear that average path length of EEIR is almost half of the that obtained by CPLEX for all considered networks.

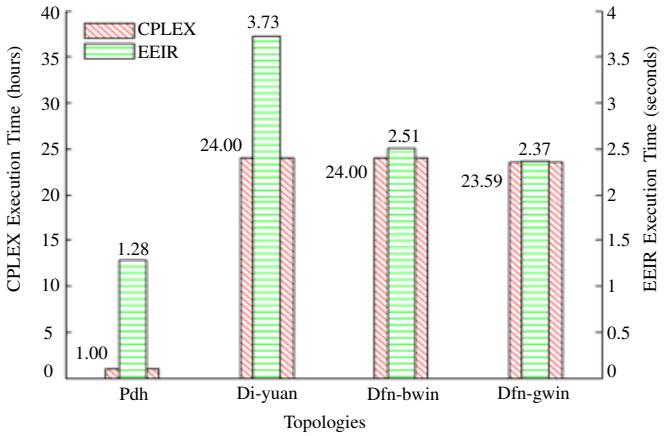

Figure 4: Average execution time of CPLEX, and proposed algorithm.

EEIR shows shorter average path length than CPLEX because it is a shortest path-based algorithm.

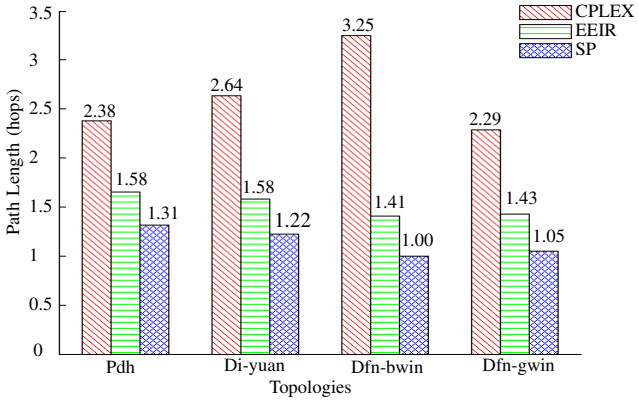

Figure 5: Average path length (hops) in all considered topologies.

V. Conclusions And Future Work

We investigated the integral routing problem with the aim to minimize total power consumption under practical constraints, i.e., discrete link rates, maximum link capacity, and flow conservation constraints. We proposed an efficient energy aware and centralized routing algorithm called Energy Efficient Integral Routing (EEIR), that minimizes power consumption while respecting real network constraints. The performance of the proposed algorithm was evaluated over four real network scenarios. Results showed that EEIR could potentially achieve near optimal solution, with a gap of 12.67% at most, in only 3.73 seconds, in comparison to CPLEX.

In future work, we plan to focus on improving the performance of EEIR to optimize routing in partially deployed Software-defined Networks.

Acknowledgment

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

References

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

[2] Global Action Plan, "An inefficient truth," *Global Action Plan Report*, Dec 2007, http://globalactionplan. org. uk.

[3] W. Vereecken, W. Van Heddeghem, D. Colle, M. Pickavet, and P. Demeester, "Overall ict footprint and green communication technologies," in *Proceedings of the 4th International symposium on Communications, Control and Signal Processing (ISCCSP'10)*. IEEE, 2010, p. 6. [Online]. Available: http://dx.doi.org/10.1109/ISCCSP.2010.5463327

[4] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing power consumption in backbone networks," in *IEEE International Conference on Communications*, June 2009, pp. 1–6.

[5] I. T. U. (ITU). (2016) Ict and energy efficiency. http://www.itu.int/en/action/climate/Pages/energy_efficiency.aspx.

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

[7] R. Razmnoush and B. Bakhshi, "Green traffic engineering in sdn," in *24th Iranian Conference on Electrical Engineering (ICEE'16)*, May 2016, pp. 693–698.

[8] F. Giroire, J. Moulierac, and T. K. Phan, "Optimizing rule placement in software-defined networks for energy-aware routing," in *IEEE Global Communications Conference (GLOBECOM'14)*, 2014, pp. 2523–2529.

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

[10] GAMS Development Corporation, "General Algebraic Modeling System (GAMS) Release 24.4.1," Washington, DC, USA, 2014. [Online]. Available: http://www.gams.com

[11] J. T. Linderoth and A. Lodi, "Milp software," *Wiley encyclopedia of operations research and management science*, vol. 5, pp. 3239–3248, 2011.

[12] M. K. Awad, M. El-Shafei, T. Dimitriou, Y. Rafique, M. Baidas, and A. Alhusaini, "Power-efficient routing for sdn with discrete link rates and size-limited flow tables: A tree-based particle swarm optimization approach," *International Journal of Network Management*, pp. e1972–n/a, 2017. [Online]. Available: http://dx.doi.org/10.1002/nem.1972

[13] Y. Rafique, M. K. Awad, and G. Neama, "A benchmark implementation for evaluating the performance of power-aware routing algorithms in practical software-defined networks," in *4th IEEE International Conference on on Software Defined Systems (SDS)*, Nov 2017.

[14] M. K. Awad, Y. Rafique, and R. A. M'Hallah, "Energy-aware routing for software-defined networks with discrete link rates: A benders decomposition-based heuristic approach," *Sustainable Computing: Informatics and Systems*, pp. 31–41, March 2016.

[15] G. Lin, S. Soh, K.-W. Chin, and M. Lazarescu, "Efficient heuristics for energy-aware routing in networks with bundled links," *Computer Networks*, vol. 57, no. 8, pp. 1774–1788

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

[17] R. Carpa, O. Gluck, L. Lefevre, and J.-C. Mignot, "Improving the energy efficiency of software-defined backbone networks," *Photonic Network Communications*, vol. 30, no. 3, pp. 337–347, 2015.

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

[19] 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

[20] *Intel Ethernet Controller X540 Datasheet Rev. 2.7*, Intel Corporation, March 2014, http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ethernet-x540-datasheet.pdf.