A Benchmark Implementation for Evaluating the Performance of Power-aware Routing Algorithms in Practical Software-defined Networks

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

***Abstract*—The increase in demand for high network bandwidth has significantly increased the network power consumption and hence, capital expenditure and operational expenditure costs. Service providers are investigating various approaches to reduce operational and management costs, while delivering richer services across their networks. Recently, several centralized power-aware routing heuristic algorithms have been proposed leveraging the centralized control of the Software Defined Networking (SDN) architecture. However, a base solution for benchmarking the performance of these algorithms has not been developed yet. In this paper we propose an implementation of the centralized power-aware routing problem for SDN in GAMS. This implementation facilitates solving the problem using commercial packages and hence serves as a benchmark for accessing the performance of centralized power-aware routing algorithms. Experimental results demonstrate the efficiency of the developed implementation.**

I. Introduction

Backbone network infrastructures are becoming increasingly dense and chaotic as the number of subscribers and their demand for content rich data and is proliferating globally. Annual global IP traffic is on the rise and is expected to surpass the ZettaByte (ZB) [1] threshold by the end of this year, hitting the 2.3 ZB mark by 2020 [1]. This explosive increase in the demand for high network bandwidth has significantly increased the network power consumption and hence, Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) costs. Taking the search giant Google as an example, it consumes around 260 million watts of energy to power its global data centers, which is equivalent to power consumed by 200,000 homes [2]. Therefore, it is crucial to employ power saving mechanisms to cut down the power-related CAPEX and OPEX costs.

Backbone networks are generally over provisioned to satisfy stringent Quality of Service (QoS) network requirements and guarantee fault tolerance. Networking devices remain greatly under utilized and idle during off peak times. Traffic in backbone networks is highly unpredictable, thus employing power efficient protocols is unrealizable in traditional networks, without accurate evaluation and full overview of the network status. Software Defined Networking (SDN) is a new networking paradigm that offers full measurability and programmability to the network. Hence, it allows for fine grained optimization of network power consumption.

Backbone networks are comprised of network components manufactured by various vendors, imposing great challenges in managing, maintaining and upgrading network devices. SDN is characterized by three fundamental aspects: a clear separation of forwarding planes and control planes in networking devices, the abstraction of networking logic from hardware implementation onto software, and the presence of a central networking controller that coordinates forwarding decisions among networking devices. Abstraction of these devices has in-turn made them basic forwarding devices, with their built-in network intelligence moved to the controller.

In Software-defined Networks, devices handle incoming packets based on predefined rules called flow rules installed in their flow tables by the central controller. Flow tables are embedded in hardware as Ternary Content Addressable Memory (TCAM). Flow tables entries include source and destination MAC address, source and destination IP address, source and destination port. An action is taken by a device based on the best matching entry for an incoming packet. Network statistics are periodically acquired from the intermediate networking devices. This gives the controller a global network view of traffic matrices and topologies, enabling it to optimize performance of the entire network. Based on these statistics, network behavior can be dynamically tuned to adapt to varying network conditions by pushing rules to devices via a common protocol called OpenFlow [3].

Power efficiency is one of the performance measures that can be optimized in SDN. In particular, based on the network status available at the central controller, a centralized routing solution can be implemented to route traffic in such a way the number of active links and nodes is optimized. Furthermore, the link rates of active links can be optimized to operate at the least possible discrete rate to improve the power efficiency of the network while satisfying all traffic demands. However, in practice, routing flows on common links and nodes increases the size of their flow tables. Thus, the limited size of TCAM must be considered in power efficient routing.

In this work we propose an implementation of the generic

[1] 1 ZB = 1 Billion GB

power-aware routing problem with practical constraints in General Algebraic Modeling System (GAMS) [4]. The proposed implementation serves as a benchmark for evaluating the performance of power-aware routing algorithms in Software-defined Networks. We consider practical network topologies that were obtained from the Survivable Fixed Telecommunication Network Design Library (SNDLib) [5]. The implementation consists of four main components. First, modeling the topologies in MATLAB. Second, implementing the problem in GAMS. Third, interfacing MATLAB and GAMS. Fourth, solving the problem by CPLEX [6] and analyzing results in MATLAB. The implementation can be easily extended to model other network constraints and optimization objectives.

The rest of the paper is organized as follows. The system model and problem formulation are presented in Section II. An overview of the benchmark implementation is presented in Section III. MATLAB and GAMS implementations are discussed in IV and V, respectively. Experimental results illustrating the efficiency of the proposed implementation are presented in Section VI. Finally, conclusions are drawn in Section VII. The source codes are available at http://www.mohamadawad.com.

II. System Model and Problem Statement

We consider a software-defined network comprised of multiple networking devices known as Forwarding Elements (FEs) and a single centralized controller covering a certain geographical location. A similar system model has been considered in our previous works [7] [8] [9]. The network is represented as an undirected graph of FE's and undirected links denoted by $G(\mathcal{E},\mathcal{L})$. The set of FEs is symbolized by $\mathcal{E}=\{1,\cdots,e,\cdots,E\}$ and the set of undirected links between the FEs is symbolized by $\mathcal{L}=\{1,\cdots,l,\cdots,L\}$. The cardinality of FEs $E=|\mathcal{E}|$, represents total number of FE's in the network; while the cardinality of the set of links $L=|\mathcal{L}|$, which represents total links in the network. An undirected link passing through FE e is denoted by $\mathcal{L}_e$. It is important to specify direction for flow conservation purposes therefore, links originated at FE e are denoted by $\mathcal{L}_e^{\rightarrow}$, while links terminated at e are denoted by $\mathcal{L}_e^{\leftarrow}$. There are F data flows to be routed through the network, represented by the set $\mathcal{F}=\{1,\cdots,f,\cdots,F\}$. The f^{th} flow is routed from origin $o(f)$ to destination $d(f)$. The connected path flow f travels from the origin to the destination and is represented by $\mathcal{P}^f$.

The size of flow f determines the minimum rate to be installed on each link $l\in\mathcal{P}^f$ required to route the flow and is represented by $r^f>0$. A decision variable $r_l^f\geq 0$ indicates whether the flow $f\in\mathcal{F}$ with rate r^f travels on link $l\in\mathcal{L}$, otherwise if link $l\notin\mathcal{P}^f$ then $r_l^f=0$. Flow rates passing through an FE $e,\ e\in\mathcal{E}$, should comply with flow conservation constraints. For every flow $f,\ f\in\mathcal{F}$, and FE $e,\ e\in\mathcal{E}$,

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

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

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

The link rate r_l to be installed on a link l is selected from a set of discrete link rates $\overline{r}_l\in\overline{\mathcal{R}}=\{R_0,R_1,\cdots,R_{\max}\}$ and these rates are sorted in ascending order (i.e., $R_0<R_1<\cdots<R_{\max}$). Conversely, the selected discrete level $\overline{r}_l$ for link $l,\ l\in\mathcal{L}$, is denoted by

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

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

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

The number of flow rules installed on an FE $e\in\mathcal{E}$ is limited by the flow table size. Therefore, the number of flows in an FE e's table should not exceed its maximum number of rules T_e, which is expressed as

$$t_e=\frac{1}{2}\sum_{\substack{f\in\mathcal{F}\\ o(f)\neq e\\ d(f)\neq e}}\sum_{l\in\mathcal{L}_e}\mathbf{1}_{\{r_l^f>0\}}, \tag{5}$$

where $\mathbf{1}_{\{\cdot\}}$ is an indicator function. It is bounded by the size of the flow-rules table,

$$0\leq t_e\leq T_e. \tag{6}$$

Based on the above description, the problem of power-aware routing (P-aR) 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 power consumption. Formally, the problem is equivalent to the following mixed integer:

$$\begin{aligned}&\text{(P-aR)}\quad \min\ \sum_{l=1}^{L}\Gamma_l(\overline{r}_l)\\ &\text{subject to}\quad (1),\ (2),\ (3),\ (4),\ (5).\end{aligned}$$

The discreteness of the objective function in addition to limitation of flow table size make the problem not only NP hard, but also unapproximated [8].

III. An Overview of the Benchmark Implementation

In this section we present an overview of the proposed implementation of the P-aR problem in GAMS. This implementation allows us to solve the problem using CPLEX under real network topologies and settings. Therefore, it serves as benchmark for assessing the performance of heuristic algorithms developed for solving the P-aR problem.

GAMS models mathematical equations based on static parameters that should be provided prior to solving the problem. In our implementation, parameters and results are handled in MATLAB, while the problem is modeled in GAMS. Both MATLAB and GAMS exchange data through the GAMS Data Exchange (GDX) framework. The GDX framework is composed of two main functions. First, the "Write GDX (WGDX)" function, which writes MATLAB structures into GDX files. Second, "Read GDX (RGDX)" function, which reads data from GDX files that contain GAMS output data.

The proposed implementation is presented over two sections: MATLAB implementation and GAMS implementation, labeled IV and V, respectively.

IV. MATLAB Implementation

The MATLAB implementation mainly initializes data structures, interfaces MATLAB and GAMS, and reads the solution. The basics of building GDX file structures and the necessary MATLAB commands to interface MATLAB and GAMS are presented in sub-sections IV-A and IV-B, respectively. Once the GDX structures have been constructed, we describe writing GDX files in sub-section IV-C and calling GAMS to solve the model in sub-section IV-D. Finally, we present the process of reading the solution from GDX files in sub-section IV-E.

A. Basics of GDX Data Structures

Parameters sent to GAMS have to be prepared in a GDX data structure. A GDX data structure is composed multiple fields that have to be defined in a particular order, that is readable by the GAMS system. The fields of the GDX structure are as follows [10]:

- *Name:* A string field representing the name of the symbol in the GDX file.
- *Val:* The value matrix of the symbol being read or written. It can be in either full or sparse format.
- *Type:* A string input representing the type of the GDX symbol. There are different types of symbols, namely, set, parameter, scalar, variable or equation.
- *Form:* A string input representing the format of the *Val* matrix, which can be either "full" or "spares".
- *Unique Element Labels (UELS):* A cell array holding unique labels of elements in the set. GAMS data is referenced by labels instead of numbers; therefore, it is important to have a unique label for each element for indexing purposes and to better understand the output solution.

B. Building GDX Structures

Given the formats of the GDX fields, the MATLAB data elements can be transformed into GAMS compatible structures.

- Node names are conventionally represented in a single dimension MATLAB cell array. The following command constructs a structure of the set of nodes, with set name = "`Nodes`" and labels for each node are read from a cell array "`NodeNames`".

```
NodesStruct.name ='Nodes';
NodesStruct.uels = transpose(NodeNames);
```

- A network topology in MATLAB can be represented as a 2-dimensional binary adjacency matrix. Existence of a link between two nodes is indicated by a 1, and 0 otherwise. The following set of commands constructs a structure of topology to be sent to GAMS. Set name is set to `Topology`. Type is set to `parameter` because topology is a static matrix. `TopologyStruct.val` is assigned to the network graph saved the `G` matrix . We are copying the entire `G`, i.e., therefore `TopologyStruct.form=full`. The dimensions of `G` are `NumberOfNodes x NumberOfNodes`, which represent a full adjacency matrix. In addition, node names are loaded from the `NodesStruct` structure.

```
TopologyStruct.name = 'Topology';
TopologyStruct.type = 'parameter';
TopologyStruct.val = G;
TopologyStruct.form = 'full';
TopologyStruct.uels = {NodesStruct.uels,
    NodesStruct.uels};
```

C. Writing GAMS Data Exchange Files

Once the GDX structures have been constructed, the GDX file can be written using the WGDX function by calling the `wgdx('output_filename', Structure_1, Structure_2, ..)`. In the WGDX function call, we start by a string specifying the output file name, followed by all the structures that have to be written into the GDX file. In our implementation, we have set the output file name to `DiscreteMtoG`. We have to construct and send structures for all the sets and parameters that are to be loaded into the GAMS program.

```
wgdx('DiscreteMtoG', NodesStruct,..,
    LinksStruct, ..,TopologyStruct );
```

D. Calling GAMS from MATLAB

Given the GDX file syntax and P-aR problem GAMS model implemented in `Discrete.gms`[2], `gams` utility is used to call the solver and solve the problem modeled in GAMS. The model file, i.e., `Discrete.gms`, has to be stored in the same working directory of the MATLAB project. In the GAMS call `lo=3` is used to generate the output logs in the MATLAB console window. The argument `gdx=DiscreteGtoM` specifies the output GDX file name that will store GAMS parameters after termination.

```
system 'gams Discrete lo=3 gdx=DiscreteGtoM';
```

The output GDX file is populated with solution variables that have to be read back into the MATLAB program. The details of reading the GDX data into MATLAB using the RGDX function are discussed in the following sub-section.

[2]See Section V for details on modeling the P-aR problem in GAMS.

E. Reading GAMS Data Exchange File

Similar to writing, reading GDX data elements is handled as structures. Thus, we need to construct appropriate structures in order to read data from the solution GDX file.

To read the solution objective value, we build a structure using the `struct('name','z','form','full')` syntax. In this syntax, the parameter to be read in full format is `z`. After the structure is initialized, we call the `rgdx` function to read the data from the GDX file. We provide the `rgdx` function with the GDX file name and the structure name, `DiscreteGtoM` and `ObjectiveStruct`, respectively. Since we are only interested in the objective value, i.e., `ObjectiveRead.val`, it is assigned to a MATLAB variable `ObjectiveValue`.

```
ObjectiveStruct = struct('name','z','form','
    full');
ObjectiveRead = rgdx('DiscreteGtoM',
    ObjectiveStruct);
ObjectiveValue = ObjectiveRead.val;
```

V. GAMS Implementation

The implementation in this section focuses on modeling the P-aR problem in GAMS, sets solver options, reads the GDX files and calls the solver. In sub-section V-A we describe the solver options that guide the solver. Then, sub-sections V-B to V-E present the syntax of declaring GAMS data structures, i.e. sets, parameters and scalars. Moreover, the process of loading data from GDX files written by MATLAB is described in sub-section V-F. Subsequent to loading of data, sub-section V-G describes the declaration of decision variables used by optimization models, whereas, sub-section V-H describes the implementation of mathematical equations in GAMS. Finally, the procedure for calling an external solver is described in sub-section V-I.

A. Setting GAMS Options

GAMS options are optional parameters used to guide the solver in finding the optimal solution. A single GAMS file can encapsulate multiple models; however, options are processed at run-time and apply all models within the GAMS file. The following options are used in our model:

- `Option RESLIM = 500000;`

  This option specifies the maximum solving time, which we set to about 6 days.

- `Option MIP = CPLEX;`

  GAMS is capable of solving the model using various external solvers. Because the P-aR problem is a Mixed Integer Problem (MIP), this option specifies using CPLEX as the default solver for MIP.

- `Option optcr = 0.0;`

  This option specifies a relative termination tolerance used in solving MIP problems. The solver stops when the proportional difference between the solution found and the best theoretical objective function is guaranteed to be smaller than `optcr`. Setting it to 0.0 guarantees optimality of the solution.

B. Declaration of Sets

Sets representing the network elements are declared here. Specifically, the set of FEs ($\mathcal{E}$), links ($\mathcal{L}$), flows ($\mathcal{F}$), and Discrete Link Rates ($\overline{\mathcal{R}}$) are declared as follows,

```
Sets Nodes, Links, Flows, Rates;
```

C. Declaration of Aliases

In routing problems, the desired output of the solver is to obtain an optimal routing path for each flow. These routing paths are formed by a chain of nodes that represent sources, destinations and intermediate transshipment nodes that belong to a single set. Therefore, it is important to redefine the set using the `Alias` command to provide a secondary domain for the solver. In the following command, we redefine the nodes set to encompass a node e and all its neighbors represented in the set `Neighbors`.

```
Alias (Nodes,Neighbors);
```

D. Declaration of Parameters

Parameters represent a static and non-varying data. In GAMS, the domain over which parameters are valid is specified in brackets following the parameter. The following are declaration of parameters represent the topology, links, origin and destination nodes, flow tables capacities, and link rates.

- A binary matrix representing the network topology connecting the nodes.

  ```
  Parameter Topology(Nodes,Nodes);
  ```

- Links in the `Topology` matrix represent directional links. However, in our problem we deal with un-directed links. This is due to the fact that if a link is chosen for routing, both its uplink and downlink streams have to be used to establish connectivity between the two nodes.

  ```
  Parameter LinksMatrix(Nodes,Nodes,Links);
  ```

- The `FlowConserve` matrix defining the origins and destinations of flows is declared as follows,

  ```
  Parameter FlowConserve (Nodes,Flows);
  ```

  Positive values represent an originator for a flow f, while negative values indicate a destination node.

- A `FlowCapacity` matrix, setting the size of the flow table at each FE is declared in the following,

  ```
  Parameter FlowCapacity (Nodes);
  ```

  This parameter limits the number of flows installed at a node $e \in \mathcal{E}$ to the table size.

- The available link rates for each link are saved in the `LinksRates` matrix. Each link is considered to have three link rates: 100 Mbps, 1 Gbps and 10 Gbps [11]. The `ord` command is used for accessing the row corresponding to each link in order to set the available link rate.

```
Parameter LinksRates(Links, Rates) = 100(
    ord(Rates) eq 1) + 1000(ord(Rates) eq
    2) + 10000(ord(Rates) eq 3);
```

- The power consumption associated with selecting each discrete link rate are saved in a matrix named `Costs`. The values are based on measurements reported in [11].

```
Parameter Costs(Links, Rates) = 3.2(ord(
    Rates) eq 1) + 4.27(ord(Rates) eq 2) +
    7.7(ord(Rates) eq 3);
```

E. Declaration of Scalars

Scalars are parameters that represent a single value. The `LinkCapacity` scalar defines the global maximum link rate that can be installed on all links. This number is loaded from the GDX file and is set to 10 Gbps.

```
Scalar LinkCapacity;
```

F. Loading GDX Input File

The values of sets and parameters are loaded from the GDX file created in Section IV [3].

In order to load a GDX file into a GAMS program, we wrap the load command, i.e., `$LOAD`, with a pair of `$GDXIN` commands. Source GDX file, i.e., `DiscreteMtoG`, is set as an input file as follows: `$if not set gdxin $set gdxin DiscreteMtoG`. Data is loaded from the GDX file by using the `$LOAD` command followed by all the parameters to be loaded.

```
$if not set gdxin $set gdxin DiscreteMtoG
$GDXIN %gdxin%
$LOAD Nodes Links Flows FlowSizes Topology
    FlowConserve FlowCapacity LinkCapacity
    LinksMatrix
$GDXIN
```

G. Declaration of Variables

The decision variables hold the solution variables and remain unknown until the model is solved. A GAMS variable can be a single dimension or multiple dimensions combining various sets. Several decisions variables declarations are itemized in the following:

- A single dimension objective variable representing total network routing power consumption; i.e., the summation of all active links power consumption, i.e., the optimal value of the P-aR problem objective function.

```
Variable z;
```

- A binary multiple dimension variable representing the visited nodes and link rates used for routing the f^{th} flow.

```
Binary Variable X(Nodes,Neighbors,Flows,
    Rates);
```

- Similar to the previous variable, `Y(Links,Rates)` is a binary multiple dimension decision variable representing the discrete link rate activated on each link.

[3]Note that the GDX file to be loaded into the GAMS environment has to be in the same directory as the "`.gms`" file.

```
Binary Variable Y(Links,Rates);
```

- This variable indicates the flow rate passing through each link. Since flow sizes cannot be negative, we have placed a bound on the variable to be strictly positive.

```
Positive Variable LinkBandwidth(Links,
    Rates);
```

H. Declaration of Equations

GAMS equations are symbolic algebraic representation of the objective function and constraints. In the following, we model the P-aR problem equations in GAMS:

- Equation representing the objective function of the P-aR problem. We use the `sum(domain,variable)` syntax to iterate over all the involved variables domains.

```
Objective..
      z =e= sum ((Links,Rates), Y(Links,
          Rates)*Costs(Links,Rates));
```

- The flow conservation constraint modeled in equation (1) consists of two parts: flows leaving the node and flows entering the node. The difference between flows entering and leaving a node equals the flow rate either generated by or destined to the node. This corresponds to values saved earlier in the `FlowConserve` parameter.

```
ConserveFlow_Constraint(Nodes,Flows)..
sum ((Neighbors,Rates), X(Nodes,Neighbors,
    Flows,Rates)$Topology(Nodes,Neighbors)*
    FlowSizes(Flows))
- sum ((Neighbors,Rates), X(Neighbors,
    Nodes,Flows,Rates)$Topology(Neighbors,
    Nodes)*FlowSizes(Flows))
=e=   FlowConserve(Nodes,Flows);
```

- Total sum of flows passing through a link in both the uplink and downlink direction representing the constraint modeled in equation (2). This constraint sets the `LinkBandwidth` variable.

```
LinkBandwidth_Constraint(Nodes,Neighbors,
    Links,Rates)$(LinksMatrix(Nodes,
    Neighbors,Links) and ord (Nodes) < ord
    (Neighbors)) ..
sum (Flows, FlowSizes(Flows)*X(Nodes,
    Neighbors,Links,Rates)) + sum (Flows,
    FlowSizes(Flows)*X(Neighbors,Nodes,
    Links,Rates))
=e=   LinkBandwidth(Links,Rates);
```

- `LinkRate_Constraint` models Equation (3). This equation assigns one of the available discrete ink rates to a link, represented in the `LinksRates` parameter. The binary variable `Y` is a decision variable assigned by the solver in order to select one discrete level. The value 1 indicates that the level is selected and, 0 otherwise.

```
LinkRate_Constraint(Links,Rates)..
LinkBandwidth(Links,Rates) =l= LinksRates(
    Links,Rates)*Y(Links,Rates) ;
```

- This forcing constraint forces activating the link and its rate simultaneously. That is, if `LinkBandwidth > 0`, then the link has to be switched on.

```
Forcing_Constraint(Links,Rates)..
Y(Links,Rates) =l= LinkBandwidth(Links,
    Rates) ;
```

- In practice, a link operates at a single rate for both its uplink and downlink streams. i.e., $\sum_{\forall Rates} Y(Links, Rates) \leq 1 \quad \forall\, l \in \mathcal{L}$.

```
SingleRate_Constraint(Links)..
      sum(Rates,Y(Links,Rates)) =l= 1;
```

- Link rates have to be bounded by an upper bound that is defined by `LinkCapacity` value.

```
LinkCapacity_Constraint(Links, Rates)..
      LinksRates(Links,Rates) =l=
          LinkCapacity;
```

- The limited flow table constraint, i.e., (5) limits the number of flows passing through a link to the table size saved in `FlowCapacity`.

```
FlowRules_Constraint(Nodes)..
  sum(Flows, sum ((Neighbors, Rates),X(
      Nodes,Neighbors,Links,Rates)) + sum
      ((Neighbors, Rates),X(Neighbors,Nodes
      ,Links,Rates)))*(0.5$(sum (Neighbors,
       FlowConserve(Nodes,Flows) ) = 0) +
      (1$(sum (Neighbors,FlowConserve(Nodes
      ,Flows) ) ne 0 ))) =l=  FlowCapacity(
      Nodes);
```

I. Calling the Solver

GAMS does not solve optimization problems; however, it prepares the model and passes it to an external solver. The solver used here is `CPLEX` because the P-aR problem is a MIP. A selection of constraints can be chosen to be solved by listing them after the model name separated by a comma. However, if all the constraints are to be solved in the model, the command `all` is used to pass all the constraints on to the solver.

```
model routing /all/;
```

The `Solve` statement calls the solver and specifies the model to be solved, the type of the model and the variable being optimized, which are `routing`, `mip` and `z`, respectively. Note that the solver has been selected earlier to be `cplex` in subsection V-A.

```
Solve routing using mip minimizing z;
```

VI. Experimental Results

This section summarizes experimental results of the proposed benchmark implementation. The implementation was tested on a real network topology. Specifically, we consider the Abilene network instance available at SNDLib [5]. The topology of this network instance is shown in Figure 1. The network consists of 12 routers, 15 links and 132 flows. This high performance backbone network connects 11 regions across the United States with an average link density of 22.73% and an average node degree of 2.50. We use this network to demonstrate a working example of the optimization model, therefore we limit our evaluations to the first 10 flows of the network instance. The source, destination and size of the ten flows are tabulated in Table I. The discrete link rates used in this experiment were adopted from the Intel X540 Ethernet Controller Data-sheet [11]. The controller is capable of operating at three transmission rates 100 Mbps, 1 Gbps and 10 Gbps. The corresponding power consumptions of the 100 Mbps, 1 Gbps and 10 Gbps, are 3.2 W, 4.27 W and 7.7 W, respectively [11].

Flow f	Source	Destination	Flow Size (MB)
1	IPLSng	STTLng	113
2	CHINng	ATLAM5	115
3	HSTNng	STTLng	133
4	LOSAng	KSCYng	195
5	LOSAng	NYCMng	83
6	HSTNng	LOSAng	184
7	IPLSng	CHINng	61
8	LOSAng	STTLng	181
9	LOSAng	SNVAng	52
10	DNVRng	ATLAM5	166

Table I: Flows sources, destinations, and size.

Given the Abilene network topology and the ten flows, the P-aR problem was solved in GAMS based on the proposed implementation. Furthermore, the Dijkstra Shortest Path (SP) algorithm was used to route the ten flows. The power efficiency of the SP algorithm is verified by comparing its solution to the benchmark provided by GAMS.

Figure 2 shows the routing tree generated based on GAMS solution. The green links represent active links, while dotted links represent inactive links. The optimal number of links required to route the 10 flows is 10 out of 15 links. The network power consumption obtained by GAMS is 40.56 W because eight inks operate at 1000 Mbps, while two links operate at 100 Mbps. Links that operate at higher rates carry multiple flows of sizes ranging between 50 MB - 200 MB. As links have to comply with discrete link rates, if a flow size exceeds rate 1, i.e., 100 MB the next discrete level has to be activated to achieve sufficient bandwidth for routing. However, looking at the link connecting SNVAng and LOSAng, it is activated at the first discrete level, i.e., 100 Mbps. The first discrete rate is sufficient because it is routing only a single flow of size 52 MB, i.e., the 9^{th} in Table I.

Unlike GAMS, the SP algorithm routes flows to minimize the path length and hence it is power inefficient. Figure 3 shows links activated based on SP solution. Links colored blue are ON links, whereas doted links are OFF links. Clearly, we can see that only two links were saved by this routing method. Furthermore, the total induced routing power consumption of switching 13 links is 53.37 W. Therefore, the SP algorithm consumes additional 31.58% of the optimal benchmark. Other performance metrics can be computed based on the GAMS solution like link utilization, path length and rate utilization.

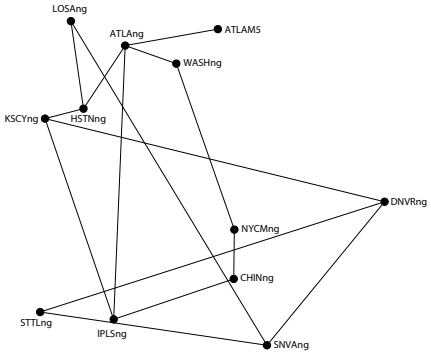

Figure 1: Abilene Network Topology

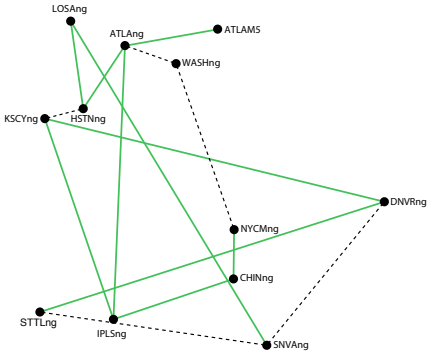

Figure 2: GAMS Topology

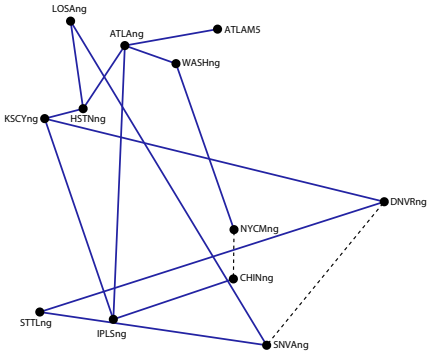

Figure 3: Shortest Path Topology

Link l	GMAS		SP	
	Rate $\overline{r}_l$	Γ_l	Rate $\overline{r}_l$	Γ_l
ATLAng - ATLAM5	1000	4.27	1000	4.27
HSTNng - ATLAng	1000	4.27	1000	4.27
IPLSng - ATLAng	1000	4.27	1000	4.27
IPLSng - CHINng	1000	4.27	1000	4.27
KSCYng - DNVRng	1000	4.27	1000	4.27
KSCYng - IPLSng	1000	4.27	1000	4.27
LOSAng - HSTNng	1000	4.27	1000	4.27
NYCMng - CHINng	100	3.20	1000	4.27
SNVAng - LOSAng	100	3.20	1000	4.27
STTLng - DNVRng	1000	4.27	1000	4.27
STTLng - SNVAng	OFF	0	1000	4.27
WASHng - ATLAng	OFF	0	100	3.20
WASHng - NYCMng	OFF	0	100	3.20

Table II: Active links, their link rates and power consumption solutions generated by GAMS and shortest-path solution.

VII. Conclusions

In this paper we presented a detailed benchmark implementation for benchmarking the performance of centralized power-aware routing heuristic algorithms. We model the power-aware routing problem in SDN as a MIP problem. We considered practical constraints, namely discreteness of link rates and limitation of flow rule table size. The model is implemented in GAMS and solved by CPLEX while model parameters were set in MATLAB. The implementation was tested on a real topology over which real flows are routed.

Acknowledgment

The authors would like to thank Dr. Ghanima Al-Sharrah with the Department of Chemical Engineering at Kuwait University for the very valuable discussion and support on modeling the optimization problems in GAMS. This project is partially funded by Kuwait Foundation for the Advancement of Sciences under project code: P314-35EO-01.

References

[1] C. V. N. Index, "Cisco visual networking index: Forecast and methodology 2015-2020," *White paper, CISCO (June 2016)*, 2016.

[2] J. Glanz and P. Sakuma, "Google details, and defends, its use of electricity," *The new york times*, vol. 8, 2011.

[3] O. S. Specification, "Version 1.2 (wire protocol 0x03)," *Open Network Foundation*, 2011.

[4] R. E. Rosenthal, "GAMS: a user's guide. 2008," *Washington, DC, USA: GAMS Development Corporation*, 2008.

[5] A. N. P. Instance. Sndlib–library of test instances for survivable fixed telecommunication network design. [Online]. Available: http://sndlib.zib.de

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

[7] 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*. IEEE, 2015, pp. 77–81.

[8] M. K. Awad, M. El-Shafei, T. Dimitriou, Y. Rafique, M. W. Baidas, and A. H. 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*, 2016.

[9] 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*, 2016.

[10] M. C. Ferris, R. Jain, and S. Dirkse, "GDXMRW: Interfacing gams and matlab," *Online: http://www. gams. com/dd/docs/tools/gdxmrw. pdf*, 2011.

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