RESEARCH ARTICLE

Power-efficient routing for SDN with discrete link rates and size-limited flow tables: A tree-based particle swarm optimization approach

Mohamad Khattar Awad1 | Mohammed El-Shafei1 | Tassos Dimitriou1 | Yousef Rafique1 | Mohammed Baidas2 | Ammar Alhusaini1

1Computer Engineering Department, College of Computing Sciences and Engineering, Kuwait University, Kuwait City, Kuwait
2Electrical Engineering Department, College of Engineering and Petroleum, Kuwait University, Kuwait City, Kuwait

Correspondence
Mohamad Khattar Awad, Computer Engineering Department, College of Computing Sciences and Engineering, Kuwait University, Firdous Street, Khaldiya, Kuwait City, Kuwait.
Email: mohamad@ieee.org

Funding Information
Kuwait Foundation for the Advancement of Sciences, Grant/Award Number: P314-35EO-01

Summary
Software-defined networking is a promising networking paradigm for achieving programmability and centralized control in communication networks. These features simplify network management and enable innovation in network applications and services such as routing, virtual machine migration, load balancing, security, access control, and traffic engineering. The routing application can be optimized for power efficiency by routing flows and coalescing them such that the least number of links is activated with the lowest link rates. However, in practice, flow coalescing can generally overflow the flow tables, which are implemented in a size-limited and power-hungry ternary content addressable memory (TCAM). In this paper, a set of practical constraints is imposed to the software-defined networking routing problem, namely, size-limited flow table and discrete link rate constraints, to ensure applicability in real networks. Because the problem is NP-hard and difficult to approximate, a low-complexity particle swarm optimization–based and power-efficient routing (PSOPR) heuristic is proposed. Performance evaluation results revealed that PSOPR achieves more than 90% of the optimal network power consumption while requiring only 0.0045% to 0.9% of the optimal computation time in real-network topologies. In addition, PSOPR generates shorter routes than the optimal routes generated by CPLEX.

KEYWORDS
network optimization, power-aware routing, software-defined networks

1 INTRODUCTION

Software-defined networking (SDN) is an emerging networking paradigm that decouples the control plane and the data forwarding plane.1 The SDN architecture is organized into 3 layers (as depicted in Figure 1): application layer, infrastructure layer, and control layer. The application layer consists of network services and applications, eg, routing, load balancing, and security. The infrastructure layer is formed of simple forwarding elements (FEs) that forward packets based on flow-rules installed in their respective flow table. The control layer holds the network intelligence and consists of a central controller (CC) that collects various performance metrics from all network devices to form a wide overview of the network status. Depending on the network status, the CC computes network management decisions based on the application layer services and applications and implements them as flow-rules at the FEs. The communication among the
different layers is facilitated by the northbound and southbound application programming interfaces.

Software-defined networking is intrinsically dynamic, secure, cost efficient, and power efficient mainly due to centralized control and programmability. The instantaneous measure of the network status and feedback available at the CC along with its ability to remotely program FEs makes dynamically reconfiguring the network and automatically responding to network events possible. In addition, SDN provides a central point of control to distribute, implement, and update security policies across the network as well as to manage threats. Moreover, SDN allows network administrators to remotely access and configure devices thus reducing the operational expenditure. Simplifying the network devices, eg, routers, switches, firewalls, and load balancers, by moving the intelligence to the CC also reduces their cost. Software-defined networking flexibility facilitates optimization of the network applications and services for power efficiency.

There has been increasing concern regarding the power inefficiency of communication networks and its negative impact on the climate change via greenhouse gas emissions. The growing number of smart devices connected to the Internet with the increasing demand for cloud computing and data centers has led to tremendous growth in information and communications technology (ICT) energy consumption worldwide. The use of ICT has been estimated to contribute 2% to 2.5% of the greenhouse gas emissions. The collective electricity consumption of ICT has increased from approximately 4% in 2007 to 4.7% in 2012 and is growing at a rate of 6.6% per year worldwide. Telecommunication networks, which constitute a major sector of ICT, contribute almost 50% of the power consumption of ICT due to the operation of telecommunication networks. Energy efficiency of both wireless and wired access networks has been the focus of several international initiatives and projects, such as the GreenTouch consortium and the European Union funded projects “Energy-Aware Radio and neTwork technologies” (EARTH) and “low Energy Consumption NETworks” (ECONET).

The scope of our work involves an investigation on how to leverage SDN for power savings in wired networks while considering practical constraints. In particular, we optimize the routing application of SDN to route traffic flows such that the network power is conserved while considering the discreteness of the link rates and the size limitation of the flow tables. The routing application solves the following routing problem: given a set of traffic flows, their sources, their destinations, and their rate requirements, determine the integral routes of the flows that minimize the network power consumption. The power consumption is minimized subject to the following constraints: one of the discrete link rates is activated per link, the flow table occupancy is limited to its size, and traffic flow demands are provisioned. In practical networks, both Ethernet and optical, links operate at one of a set of increasing discrete rates. In a previous work, we showed that considering the discreteness of the link rates is crucial to exploit the power saving margin of wired networks. In addition, in practice, flow tables are implemented in ternary content addressable memory (TCAM) cells, which are power hungry and limited in size. Thus, the routing application should limit the number of flow-rules installed at each FE such that it does not exceed the TCAM size.

Recent studies have focused on developing a power-efficient routing algorithms for software defined networks. In a study, Wang et al developed a low-complexity greedy algorithm to route flows by aggregating them over the least number of links while considering packet delay and link utilization constraints. Özbek et al presented a greedy routing algorithm to minimize the network power consumption by switching off inactive devices, while provisioning traffic demands. Similarly, in another study, Markiewicz et al presented a heuristic method with 4 different strategies to switch on the minimum number of network devices and links, such that traffic is routed and network power consumption is minimized. Despite the significant power saving these algorithms achieve, they overlook 2 major practical constraints in SDN: the discreteness of link rates and the limitation of the flow table size. Such algorithms fail in practice for 2 reasons. First, the computed link rates are continuous whereas in practice, FEs activate one of its discrete link rates, eg, 10 Mbps, 100 Mbps, or 1 Gbps. Thus, the computed rates

![FIGURE 1 Software-defined networking architecture. API, application programming interface](image)
must be rounded to the nearest discrete rate; as a result, the claimed power saving cannot be achieved. Second, a key approach in these algorithms is traffic aggregation to reduce the number of active links and devices; therefore, it is possible for these algorithms to overload the flow table of a FE connecting 2 subdomains, possibly resulting in link failures.

Alternatively, several other works have considered either one or the other of these practical constraints. Nam et al\textsuperscript{15} presented an OpenFlow-based routing algorithm for data centers using SDN. Although discrete link rates were considered in this work, the limitation on the size of flow tables was not considered. Furthermore, algorithms designed for power-aware routing in data centers generally exploit the symmetry of the data center topologies and are not applicable in regular networks that do not exhibit similar properties. In a previous study,\textsuperscript{16} we presented a Benders-based routing algorithm for networks with discrete link rates; however, the limited flow table size was not considered. In another study,\textsuperscript{17} Tang et al\textsuperscript{17} presented a flow allocation algorithm in which the flows are allocated on a set of precomputed paths through links with discrete link rates. However, their algorithm requires an optimization package that is commonly not available at the CC and assumes an infinite size of flow table. Other works addressed the issue of the limited size of the flow table. For example, in a previous study,\textsuperscript{18} Girroir et al\textsuperscript{18} proposed a greedy routing heuristic algorithm to minimize the energy consumption in backbone networks while optimizing the flow-rule placement in FEs. However, they did not consider the availability of different link rates and assumed that all links are operating at their capacity. Rifai et al\textsuperscript{19} presented a compression technique to minimize the memory required in the TCAM of FEs. Maqbool et al\textsuperscript{20} presented a technique for implementing a virtual TCAM to enhance the flow-rule storage capabilities. The deployment of these techniques\textsuperscript{19,20} is not trivial as it requires manual configuration of each FE and cannot be implemented remotely.

In this current work, we consider the practical limitations of algorithms available in the literature and propose a low-complexity power-efficient routing algorithm subject to the practical constraints described earlier, namely, the discreteness of the link rates and the limitation of the flow-rule table size. The proposed algorithm operates at the application layer of SDN and does not require any manual configuration at the infrastructure layer. The algorithm is developed based on the particle swarm optimization (PSO) algorithm, which is a population-based metaheuristic algorithm.\textsuperscript{21} Specifically, the PSO algorithm is extended to be tree based,\textsuperscript{22,23} wherein each particle is initialized to hold a tree representing 1 minimal spanning tree of the network. The proposed algorithm operates over several phases: initialization, merging, flow-rules constraints satisfaction and termination. The proposed algorithm is a low-complexity heuristic algorithm and does not require an optimization package at the CC. Our major contributions are as follows:

1. We present a mixed-integer linear programming formulation of the power-efficient routing problem (PeRP) in SDN with practical constraints, ie, the discrete link rates and the size-limited flow tables.
2. We show that the considered routing problem is NP-hard and hard to approximate.
3. We present a low-complexity heuristic that effectively solves the routing problem with a small optimality gap in comparison to a solution provided by the CPLEX commercial optimization package.\textsuperscript{24}
4. We present extensive performance evaluation results that demonstrate the effectiveness of the proposed algorithm in real networks with diverse properties.

The remainder of the paper is organized as follows. We present the network model and problem formulation in Section 2. An overview of the basic PSO algorithm is given in Section 3. The PSO-based and power-efficient routing (PSOPR) algorithm is presented in Section 4. Performance evaluations are given in Section 5. We conclude our work in Section 6. In the Appendix, we show that this problem is NP-hard and hard to be approximated. The performance evaluation codes are available at http://www.mohamadawad.com.

2 NETWORK MODEL AND PROBLEM FORMULATION

We consider a software-defined network consisting of a single CC and \(N\) FEs. We extend our model presented in Awad et al\textsuperscript{16} to consider the limited size of flow tables. The set of FEs is denoted by \(\mathcal{N} = \{1, \ldots, n, \ldots, N\}\), and the set of links connecting them is denoted by \(\mathcal{L} = \{1, \ldots, l, \ldots, L\}\). Thus, the network can be represented by the graph \(G(\mathcal{N}, \mathcal{L})\). The subset of links \(\mathcal{L}\) originated at FE \(n\) is denoted by \(\mathcal{L}_n^+\), whereas the subset of links terminated at \(n\) is denoted by \(\mathcal{L}_n^-\), and their union is \(\mathcal{L}_n\).

A set of flows \(\mathcal{F} = \{1, \ldots, f, \ldots, F\}\) is to be routed through the network, each with a rate requirement denoted by \(r^f\). A given flow \(f\) is originated at \(o(f)\) and terminated at \(d(f)\). The CC computes the optimal integral routing path denoted by \(P^f\) to route flow \(f\) from \(o(f)\) to \(d(f)\) and then updates the forwarding table of the FEs on this path. A flow-rule is installed at each FE on the path indicating the action to be taken when a packet belonging to flow \(f\) is received. In this work, we consider integral routing where each flow is routed on a single path. Integral routing is important when the number of flow-rules installed at each FE is limited and when the frames are not arbitrarily divisible.\textsuperscript{25}
Let $r_f^l$ be the decision variable reflecting the data rate required on link $l \in L$ by flow $f \in F$; i.e., when $l \in \mathcal{P}^f$, $r_f^l = r_f$. Alternatively, when $f$ does not travel through $l$, $r_f^l = 0$. This relationship can be captured using an indicator decision variable $\mathbb{1}\left(r_f^l\right)$ by the equality

$$r_f^l = \mathbb{1}\left(r_f^l\right) r_f.$$  

(1)

Here, $\mathbb{1}\left(r_f^l\right)$ takes a value of 1 if flow $f$ passes through link $l$ and is 0 otherwise. That is,

$$\mathbb{1}\left(r_f^l\right) = \begin{cases} 1 & r_f^l > 0 \\ 0 & r_f^l = 0 \end{cases}.$$  

(2)

An FE on the path of a given flow adds a flow-rule entry to its flow-table defining a matching rule and its corresponding action. Flow tables are implemented in TCAM that are of limited size and supports hundreds to few thousands of entries. Each FE maintains a TCAM-based forwarding table of maximum size $T_n$. Let $t_n$ be the length of the FE $n$’s table. The number of flow-rules installed, i.e., table occupancy, at a given FE $n, n \in \mathcal{N}$, can be written as

$$t_n = \frac{1}{2} \sum_{f \in F} \sum_{l \in L} \mathbb{1}\left(r_f^l\right),$$  

(3)

$t_n$ is bounded by the size of flow table, i.e.,

$$0 \leq t_n \leq T_n.$$  

(4)

This equation constitutes the limited flow table size constraint. The data rate on a given link $l, l \in L$, is equivalent to the sum of flows passing through it,

$$r_l = \sum_{f \in F} r_f^l.$$  

(5)

The CC activates one of the discrete operating rates $\tilde{r}_l \in \mathcal{R} = \{R_0, R_1, \cdots, R_{\text{max}}\}$ on link $l, l \in L$ to satisfy the routed traffic rate through link $l$, $r_l$. Therefore, the operating rate $\tilde{r}_l$ for link $l, l \in L$, is given by

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

(6)

The power consumption of a given discrete rate $\tilde{r}_l \in \mathcal{R}$ for a link $l, l \in L$, is given by

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

(7)

Furthermore, flows passing through a FE $n, n \in \mathcal{N}$, must satisfy the following flow conservation constraints. For every flow $f, f \in F$, and for FE $n, n \in \mathcal{N}$,

$$\sum_{l \in L_n^+} r_f^l - \sum_{l \in L_n^-} r_f^l = \begin{cases} 0 & n \neq \alpha(f) \text{ and } n \neq d(f) \\ r_f^l & f \in \mathcal{F}, n = \alpha(f) \\ -r_f^l & f \in \mathcal{F}, n = d(f). \end{cases}$$  

(8)

The PeRP in software-defined networks with discrete link rates and limited flow tables can be formulated as

$$\text{(PeRP)} \quad \min \sum_{l = 1}^{L} \Gamma_l(\tilde{r}_l) \quad \text{(9a)}$$  

subject to

$$r_l = \sum_{f \in F} r_f^l, \quad l \in L$$  

(9b)

$$\sum_{l \in L_n^+} r_f^l - \sum_{l \in L_n^-} r_f^l = \begin{cases} 0 & n \neq \alpha(f) \text{ and } n \neq d(f) \\ r_f^l & f \in \mathcal{F}, n = \alpha(f) \\ -r_f^l & f \in \mathcal{F}, n = d(f). \end{cases}$$  

(9c)

$$\tilde{r}_l = \begin{cases} R_0 & 0 < r_l \leq R_0, \\ R_1 & R_0 < r_l \leq R_1, \\ \vdots & \vdots \\ R_{\text{max}} & R_{\text{max}} - 1 < r_l \leq R_{\text{max}}. \end{cases} \quad \forall l \in L$$  

(9d)

$$t_n = \frac{1}{2} \sum_{l \in L} \sum_{f \in F} \mathbb{1}(r_f^l)$$  

(9e)

$$0 \leq t_n \leq T_n, \quad n \in \mathcal{N}$$  

(9f)

The mathematical formulation in Equation 9a is equivalent to a mixed-integer programming problem with a nonconvex discrete–cost step function. The integral routing constraint and discreteness of the objective function make the problem NP-hard. Furthermore, the flow-rule constraint adds to the complexity of the power efficient routing in SDN. In the Appendix, we show that PeRP is not only NP-hard but also cannot be approximated.

Before introducing the proposed algorithm in Section 4, an overview of the original PSO algorithm is given in the following section, Section 3.

### 3 | BASIC PSO ALGORITHM

Particle swarm optimization, which is a population-based metaheuristic algorithm developed by Eberhart and Kennedy,
was inspired by the behavior of bird flocking.\textsuperscript{21} In PSO, every particle mimics a bird that searches for food, i.e., the "solution," at a certain velocity by learning from the experience of the individual and its companions. The movement for each particle is controlled by the following system of equations.

\begin{equation}
\begin{aligned}
\mathbf{v}_{id}^{iter+1} &= \omega \times \mathbf{v}_{id}^{iter} + c_1 \times \text{rand}_1 \times (\mathbf{p}_{id} - \mathbf{x}_{id}^{iter}) \\
&+ c_2 \times \text{rand}_2 \times (\mathbf{g}_{best} - \mathbf{x}_{id}^{iter}) \\
\mathbf{x}_{id}^{iter+1} &= \mathbf{x}_{id}^{iter} + \mathbf{v}_{id}^{iter+1}
\end{aligned}
\end{equation}

Equations 10 and 11, respectively, update the \( i^{th} \) particle velocity \( \mathbf{v}_{id}^{iter+1} \) and position \( \mathbf{x}_{id}^{iter+1} \) at the end of iteration \( Iter \) based on its previous position \( \mathbf{x}_{id}^{iter} \), previous velocity \( \mathbf{v}_{id}^{iter} \), best position \( \mathbf{p}_{id} \) that the particle has achieved, and the global best position \( \mathbf{g}_{global-best} \) among all particles. Furthermore, in Equations 10 and 11, the constant \( \omega \) is the inertia weight, \( \text{rand}_1 \) and \( \text{rand}_2 \) are mutually independent random numbers between 0 and 1. The constants \( c_1 \) and \( c_2 \) are learning factors.

The PSO algorithm minimizes a cost or fitness function \( \text{func}(\cdot) \) over all positions. The pseudocode of PSO is given\textsuperscript{26} in Algorithm 1. The algorithm initializes \( M \) particles and iterates until the termination conditions are satisfied. Reaching maximum number of iterations \( maxIter \) or convergence are common termination conditions. In each iteration, the particles positions, velocities, \( \mathbf{p}_{id} \) and \( \mathbf{g}_{global-best} \) are updated. At termination, the global best position is returned as a solution.

### 4 PARTICLE SWARM OPTIMIZATION-BASED AND POWER-EFFICIENT ROUTING ALGORITHM

In this section, we introduce a PSOPR algorithm for software-defined networks. We extend the PSO algorithm to a tree-based PSO algorithm to solve the PeRP.\textsuperscript{22} In PSOPR, every particle has 6 characteristics: first, the set of links forming the subgraph of the particle, \( T_{par} \); second, a set of the FEs forming the subgraph of the particle, \( T_{n} \); third, a subgraph source \( s \), which indicates the root FE of the subgraph; fourth, a routing table that stores the routing paths for all flows, \( P_{f} \forall f \in F \); fifth, the network power consumption, which denotes the total power consumption of the network, \( W_{par} \); and sixth, the flow-rules table, which stores the number of flow-rules installed at each FE, \( t_{n} \forall n \in N \). The proposed algorithm consists of several phases, which are described in the following.

1. Initialization: Initialize a population of \( O \) particles, where each particle's subgraph is equivalent to a minimal spanning tree.
2. Merging: This phase iterates for \( i_{\text{max}} \) iterations. In each iteration, the particle with the least power consumption \( W_{par} \) is identified and is considered to be the best solution of the \( i^{th} \) iteration, denoted by \( B_{i} \). Each particle sub-graph is merged with the \( B_{i} \)'s sub-graph, and replaced with the merged sub-graph if and only if the network power consumption is reduced by merging.
3. Limited flow table constraints satisfaction: Identify particles violating the limited flow table size constraints and reroute the flows on alternative paths to satisfy these constraints. Alternative paths are computed using the K-shortest-path (KSP) algorithm. Particles that do not satisfy the flow-rules constraints after rerouting are eliminated from the population.
4. Termination: Among the remaining particles, the particle with the minimal power consumption \( W_{par} \) is considered the global best GB solution and the best solution to the PeRP.

The particles subgraphs in the initialization phase are minimal spanning trees; however, in the merging phase, trees
are merged and the resulting topology might contain cycles. Therefore, after merging, we refer to it by “subgraph” rather than “tree.” Although cycles can be removed from the subgraph, they are necessary for finding alternative paths and rerouting traffic to satisfy the size limited flow-rules table constraints. In the following subsections, we present several routines to support each of the algorithm phases.

### 4.1 Initialization

The pseudocode in Routine 2 presents the initialization phase for a population of \(O\) particles. Each particle is initialized with a subgraph that is a minimal spanning tree derived from the network graph \(G(N, L)\). A random source node \(s\) is selected from the set of FEs \(N\) to initialize the spanning tree and is added to set of nodes forming the tree \(T_{n}^{\text{par}}\). The function \(\text{rand}()\) selects 1 element at random from a set. Next, a one-hop link is randomly selected from any of the tree nodes to any of the other adjacent nodes until all nodes are added to the tree, ie, \(N \setminus T_{n}^{\text{par}} = \emptyset\) (see lines 6 to 15). Once a minimal spanning tree is formed, the \(F\) flows are routed on the minimal spanning tree using the shortest path (SP) Algorithm, line 17. Based on the computed routing paths, other particle characteristics, such as flow-rules count \(t_{\text{par}}^{n}\) and network power consumption \(W_{\text{par}}^{n} = \sum_{l=1}^{L} \Gamma(l(f))\), are updated.

### 4.2 Merging

The pseudocode in Routine 3 presents the merging phase for the PSOPR. After initializing a population of \(O\) particles,

---

**Routine 2: Particles initialization**

**Data:** Population size \(O\), Network Topology  
**Result:** \(O\) particles  
/* Initialization of the particle subgraph sets of links and FEs */  
1 \(T_{n}^{\text{par}} = \{\}\)  
2 \(T_{n}^{\text{par}} = \{\}\)  
3 for \(par = 1\) to \(O\) do  
   /* Random selection of tree source */  
   4 \(s \leftarrow \text{rand}(N)\)  
   /* Add it to the tree elements */  
   5 \(T_{n}^{\text{par}} = T_{n}^{\text{par}} \cup \{s\}\)  
   repeat  
      /* Initialize the set of candidate links */  
      6 \(C = \{\}\)  
      foreach \(\tau \in T_{n}^{\text{par}}\) do  
         /* Find set of links \(C_{\tau}\) from \(\tau\) to \(\tau'\): \(\tau' \notin T_{n}^{\text{par}}\)  
         7 \(C_{\tau} = L_{\tau}^{n} \cap L_{\tau'}^{n} : \tau' \notin T_{n}^{\text{par}}\)  
         /* Select a link at random and add it to the tree */  
         8 \(C = \bigcup_{\tau} C_{\tau}\)  
         /* Add the new element \(\tau'\) corresponding to \(l'\) to the set of tree nodes */  
         9 \(T_{n}^{\text{par}} = T_{n}^{\text{par}} \cup \{l'\}\)  
      until \(N \setminus T_{n}^{\text{par}} = \emptyset\);  
   /* Route flows on tree links \(T_{n}^{\text{par}}\) */  
   10 foreach \(f \in F\) do  
      /* Compute the remaining particle characteristics */  
      11 \(P_{f}^{n} \leftarrow \text{SP}(T_{n}^{\text{par}}, s(f), d(f))\)  
   Update \(t_{\text{par}}^{n}\)

---

**Routine 3: Merging**

**Data:** \(O\) particles, \(i_{\text{max}}\)  
**Result:** merged \(O\) particles  
1 for \(i = 1\) to \(i_{\text{max}}\) do  
   /* Identify particle with minimum power consumption */  
   2 \(B_{i} \leftarrow \arg \min_{\text{par}}(W_{\text{par}})\) \(\forall \text{par} = 1 \ldots O\)  
   /* Merge set of sub-graphs links */  
   3 \(T_{i}^{\text{par}} = \bigcup_{\text{par}} T_{i-1}^{\text{par}}\)  
   /* Route all \(F\) flows on the merged sub-graph */  
   4 foreach \(f \in F\) do  
      /* Replace sub-graph with merged one */  
      5 \(P_{f}^{i} \leftarrow \text{SP}(T_{i}^{\text{par}}, s(f), d(f))\)  
      \(W_{\text{par}}^{i} = \sum_{l=1}^{L} \Gamma(l(f))\)  
   if \(W_{\text{par}}^{i} < W_{\text{par}}^{\text{par}}\) then  
      /* Compute the rest of particle characteristics */  
      6 Update \(T_{\text{par}}^{i}\)  
      Update \(t_{\text{par}}^{i}\)  

these particles sub-graphs are merged with the subgraph of the lowest power consumption. The merging process repeats for \(i_{\text{max}}\) iterations. The particle with minimum power consumption \(W_{\text{par}}^{i}\) is considered to be the best solution, \(B_{i}\), of the \(i\) th iteration. Each particle’s subgraph is merged with the
subgraph of \( B_i \) to form a new subgraph referred to by the merged subgraph. The merged subgraph links are denoted by \( T^m_{i, \text{par}} \). Subgraphs are merged using the logical OR operator, \( \lor \), where a link is active in the merged subgraph if it is active in at least 1 of the 2 subgraphs being merged, i.e., \( T^m_{i, \text{par}} = T^{par}_{i} \lor T^{R}_{i, \text{par}} \). It was mentioned earlier that the merged subgraph is not necessarily a spanning tree. If the merged subgraph power consumption is less than that of the original particle’s subgraph, then the merged subgraph replaces the original one, and the characteristics are updated; otherwise, the original subgraph is retained (see lines 8 to 12).

**Routine 4: Constraints satisfaction**

\begin{verbatim}
Data: Particles violating the limited flow table constraints, \( T_n \forall n \in \mathcal{N} \)
Result: Particles satisfying the limited flow table constraints.

1 forall the Particles containing any \( n \) with \( t^\text{par}_n > T_n \) do
   /* Find set of nodes with flow-rules violation */
   \( \mathcal{V} = \{ n : t^\text{par}_n > T_n \} \forall n \in T^\text{par}_n \)
   /* Find the element \( n \) with maximum violation */
   \( v = \arg \max_n |t^\text{par}_n - T_n| \forall n \in \mathcal{V} \)
   /* Find a set of \( K \) shortest paths to reroute flow \( f \) */
   \( \mathcal{K}\mathcal{P}^f = \text{KSP}(T^\text{par}_{i, v}, s(f), d(f), K) \)
   /* Select the shortest path that does pass through the element \( v \) */
   \( \bar{p}^f \leftarrow \text{Shortest path in } \mathcal{K}\mathcal{P}^f : \text{it does not contain } v \)
   if \( \bar{p}^f \neq \text{null} \) then
      /* \( \bar{p}^f \) does not pass through \( v \) */
      \( p^f \leftarrow \bar{p}^f \)
   else
      /* All \( K \) paths pass through \( v \) */
      Keep original SP \( p^f \)

11 if \( t^\text{par}_n \leq T_n \forall n \in T^\text{par}_n \) then
   /* Update the rest of the particle characteristics */
   Update \( t^\text{par}_n \)
   Update \( W^\text{par} \)
12 else
   /* Rerouting did not satisfy all flow-rules constraints */
   remove particle \( \text{par} \) from population
\end{verbatim}

4.3 Limited flow table constraints satisfaction

The pseudocode in Routine 4 presents the rerouting process to satisfy the flow-rules constraints \( t_n \leq T_n \forall n \in \mathcal{N} \). Forwarding elements violating the flow-rules constraint, i.e., their flow tables contains flow-rules greater than its size, \( t_n > T_n \), form the set \( \mathcal{V} \). Flows are rerouted using the KSP algorithm to avoid traversing through a FE \( v \) with the maximum flow-rules violation \( |t_n - T_n| \). Specifically, KSP(\( T^\text{par}_{i, v}, s(f), d(f), K \)) computes \( K \) SPs for the \( f \)th flow from source \( s(f) \) to destination \( d(f) \). Among these paths, the path not passing through \( v \) is selected to route flow \( f \). If no path found, then the SP is selected. If the limited flow table constraints are satisfied for all FEs, then the particle’s characteristics are updated; otherwise, the particle is removed from the population.

**Routine 5: Termination**

\begin{verbatim}
Data: Particles satisfying PeRP constraints
Result: Global Best solution, GB

1 forall the Particles in population do
   /* Find the particle with least power consumption */
   \( \text{par} = \arg \min_{\text{par}} W^\text{par} \)
2 GB \( \leftarrow \text{par} \)
3 Turn off links not in the GB sub-graph set of links \( T^\text{gb}_i \)
4 Route flows according to \( p^f \forall f \in \mathcal{F} \)
5 Set link rates according to equation (6).
\end{verbatim}

4.4 Termination

All remaining particles in the population after the constraints satisfaction phase satisfy all of the problem PeRP constraints; hence, all routing solutions and link rates are valid. The last step is to find the solution with the lowest power consumption \( W^\text{par} \), which is the GB solution. In Section 5, we evaluate the performance of the proposed algorithm in real networks of diverse properties.

5 PERFORMANCE EVALUATION

In this section, we present the simulation performance results of the proposed algorithm in real networks of diverse properties. For all our simulation experiments, we considered real-network topologies from the library of test instances for survivable fixed telecommunication network design (SNDlib). Four networks with diverse properties were selected (as shown in Figure 2): Abilene, Nobel-US, Nobel-Germany, and Polska. The properties of these networks are tabulated in Table 1. The link density of a topology is given by \( L / \frac{N(N-1)}{2} \). All simulations were performed in
TABLE 1  Properties of network topologies

<table>
<thead>
<tr>
<th>Network Topology</th>
<th>FEs</th>
<th>Links</th>
<th>Flows</th>
<th>Link Density, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abilene</td>
<td>12</td>
<td>15</td>
<td>132</td>
<td>22.73</td>
</tr>
<tr>
<td>Nobel-US</td>
<td>14</td>
<td>21</td>
<td>91</td>
<td>23.08</td>
</tr>
<tr>
<td>Nobel-Germany</td>
<td>17</td>
<td>26</td>
<td>121</td>
<td>19.12</td>
</tr>
<tr>
<td>Polska</td>
<td>12</td>
<td>18</td>
<td>66</td>
<td>27.27</td>
</tr>
</tbody>
</table>

Abbreviation: FEs, forwarding elements.

TABLE 2  Link rates and their power consumption

<table>
<thead>
<tr>
<th>Discrete Rate</th>
<th>Link Rate</th>
<th>Power Consumption, Watts</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>100 Mbps</td>
<td>3.20</td>
</tr>
<tr>
<td>R1</td>
<td>1 Gbps</td>
<td>4.27</td>
</tr>
<tr>
<td>R2</td>
<td>10 Gbps</td>
<td>7.70</td>
</tr>
</tbody>
</table>

MATLAB on a 3.5 GHz Intel Xeon E5 computer with 16 GB RAM. The performance results of the proposed algorithm were compared against the optimal performance results generated by IBM CPLEX 12.4. To allow CPLEX to find the optimal solution, the stopping criterion was set to reach an absolute gap of 0. The absolute gap refers to the distance between best solution generated by CPLEX and the optimal solution. The proposed algorithm is the first to solve the problem under consideration with the considered practical constraints; therefore, the comparison was limited to CPLEX.

The flows had randomly selected source \( o(f) \) and destination \( d(f) \) FEs, as well as random traffic demand \( r_f \), \( \forall f \in \mathcal{F} \) following a uniform distribution in a range of [200, 300] Mbps. The link rates and their corresponding power consumption ratings are based on the measurement results reported in a study, as listed in Table 2. These values do not include the power consumed by the line cards and the chassis. We assume that each line card is connected to a single link. Therefore, the power consumption of the line cards and chassis can be added as constants to the link rate power rating and do not affect the performance results. In all experiments, the number of merging iterations \( i_{\text{max}} \) is set to 10% of the population size \( O \), i.e., \( i_{\text{max}} = 0.1 \times O \).

The presented results represent averages over 50 experiment runs. Each of the considered topologies has a minimum size of flow table that guarantees feasibility of the problem. The problem is not solvable with a size of flow table less than this minimum size where the feasible region is tight. Moreover, there is a maximum size of flow table in which the flow table size constraint becomes relaxed. In our experiments, the size of flow table starts at the minimum value required for the problem to be solvable and ends at a value larger than the maximum value that relaxes the constraint.

The performance of the PSOPR is evaluated over different feasible regions ranging from tight to loose. In Sections 5.1 to 5.4, we present the performance evaluation findings of PSOPR over the 4 topologies and compare it to CPLEX for a...
fixed number of particles, $O = 500$. However, in Section 5.5, we present the performance results of PSOPR over Nobel-US for a range of population size $O$.

5.1 Abilene

The Abilene network is a high-performance backbone network that was developed in the late 1990s for education, research, and commercial Internet service providers. Figure 3 shows the power consumption of the Abilene network under PSOPR and CPLEX and the corresponding optimization error percentage. The confidence interval of PSOPR over the 50 runs versus the flow table size is shown in Figure 3A. The average power consumption falls in the lower end of the confidence interval for most of the runs. The minimum flow table size required for the problem to be solvable is 56, and the maximum number of flow table size beyond which the constraint is relaxed is 84. In comparison to CPLEX, the PSOPR average optimization error is less than 10% for a given flow table size in the range 56 to 84, as shown in Figure 3B. However, as the size of flow table exceeds 84, the flow table size constraint is relaxed, and the PeRP problem is optimally solved to 0% error. The average error over the considered range of flow table size is 6.77%. The average execution time for CPLEX is 104.56 seconds, whereas the average execution time of PSOPR with $O = 500$ is 0.2086 second.

Figure 4A presents the average path length of Abilene for PSOPR and CPLEX. As reflected in Figure 4A, the average path length increases with the increase of flow table size. A larger flow table size allows the routing application to route flows on common links and reduces the network power consumption, as shown in Figure 3A. This comes at the cost of longer paths and delays, as shown in Figure 4A. However, it is clear that PSOPR outperforms CPLEX in generating shorter paths lengths: the average path length of PSOPR is 2.77 hops while that of CPLEX is 3.81 hops. Note that the path length

![Figure 3](image-url) A, Power consumption of Abilene and B, optimization error of particle swarm optimization–based and power-efficient routing (PSOPR) relative to CPLEX

Figure 5A shows the power consumption of Nobel-US and Nobel-Germany. These networks are funded by the European Commission and involved major telecommunication network operators and equipment manufacturers. Hence, these networks were carefully constructed to correspond to realistic planning

![Figure 4](image-url) A, Average path length and B, average rate utilization in Abilene. PSOPR, particle swarm optimization–based and power-efficient routing

![Figure 5](image-url) A, Power consumption of Nobel-US and B, optimization error of particle swarm optimization–based and power-efficient routing (PSOPR) relative to CPLEX

5.2 Nobel-US

Nobel-US and Nobel-Germany are international research networks originating from the European project Neighborhood Oriented Brokerage Electricity Monitoring system. These networks are funded by the European Commission and involved major telecommunication network operators and equipment manufacturers. Hence, these networks were carefully constructed to correspond to realistic planning

*The average rate utilization reflects the ratio of flow rate on a link to the activated link rate on the same link, averaged over network links. The average rate utilization is computed according to $\frac{\sum_{n \in L} (r_n / \bar{r}_n)}{L}$. 
scenarios. The power consumption of the Nobel-US network when operating under the proposed PSOPR and CPLEX is shown in Figure 5A. The flow table size ranges from 35 to 59 rules. The average execution time for CPLEX is 840 minutes, whereas for PSOPR, it is 2.256 seconds. The PSOPR power consumption approaches the optimal consumption level as the size of the flow table increases; however, a small gap of 0.98% remains. PSOPR provided solutions 22 340.4 times faster than CPLEX, while the average error of PSOPR for a given size of flow table is less than 10%; moreover, PSOPR maintains an average error of 6.53% as compared to the results of CPLEX over the considered range of flow table size (Figure 5B).

The results demonstrate the effectiveness of the proposed PSOPR in generating shorter paths than CPLEX, as shown in Figure 6A. Furthermore, the average rate utilization averages at 35% for PSOPR and 66.5% for CPLEX. PSOPR not only generates a solution within 10% of the optimal in 0.0045% of CPLEX computational time but also generates shorter paths. For large flow table sizes, ie, $T_n \geq 57$, the PSOPR and

![FIGURE 6](image)

**FIGURE 6** A, Average path length and B, average rate utilization in Nobel-US. PSOPR, particle swarm optimization–based and power-efficient routing

![FIGURE 7](image)

**FIGURE 7** A, Power consumption of Nobel-Germany and B, optimization error of particle swarm optimization–based and power-efficient routing (PSOPR) relative to CPLEX

![FIGURE 8](image)

**FIGURE 8** A, Average path length and B, average rate utilization in Nobel-Germany. PSOPR, particle swarm optimization–based and power-efficient routing

![FIGURE 9](image)

**FIGURE 9** A, Power consumption of Polska and B, optimization error of particle swarm optimization–based and power-efficient routing (PSOPR) relative to CPLEX

![FIGURE 10](image)

**FIGURE 10** A, Average path length in hops and B, average rate utilization in Polska. PSOPR, particle swarm optimization–based and power-efficient routing

CPLEX solutions are comparable; however, the average rate utilization of PSOPR is approximately half that of CPLEX,
Figure 6B. This finding indicates that better load balancing is provided by PSOPR compared to CPLEX.

5.3 | Nobel-Germany

Routing flows on the Nobel-Germany topology is a complex task because it has 26 links, 17 FEs, and 121 flows. Figure 7A shows the average power consumption of both PSOPR and CPLEX. PSOPR shows a narrow confidence interval over the tight feasible region, and it gets wider as the size of flow table increases. This behavior is expected because a limited number of routing solutions is available over the tight feasible region, and the converse is true over the relaxed region. The PSOPR error is less than 10%, over the tight feasible region and decreases to 0% for a large size of flow table, i.e., the flow table size constraint is relaxed. PSOPR provides solutions that are 5756.9 times faster than CPLEX (270 min vs 2.814 sec) and has an average error of 8.86%, as shown in Figure 7B. Although the PSOPR solution matches the solution of CPLEX, it outperforms CPLEX in terms of path lengths (3.47 hops for PSOPR vs 5.59 hops for CPLEX), as per Figure 8A, and average rate utilization (38.8% for PSOPR vs 61.34% for CPLEX), see Figure 8B.

5.4 | Polska

Polska is an industrial background synchronous digital hierarchy transport network of the Polish Telecom developed in the early 1990s. For a flow table size ranging from 25 to 61, Figure 9A shows the average power consumption of Polska. The corresponding average error of PSOPR in comparison to CPLEX is shown in Figure 9B. The results reveal tight confidence intervals than other topologies that become even tighter as the size of flow table increases. This behavior occurs via the close-to-symmetrical shape of the topology, wherein randomly generated minimal spanning trees are highly correlated. This behavior also justifies the fixed error observed for $T_n > 50$, where most particles in this range have highly correlated minimal spanning trees and hence the same power consumption. The average execution time for PSOPR ($O = 500$) is 0.912 second, whereas for CPLEX, it is 70 minutes. PSOPR provided routing solutions 4605.3 times faster than CPLEX, with errors less than 10% for a given flow table and an average error of 6%, as shown in Figure 9B.

The difference between the average path length of routes generated by CPLEX and PSOPR, as shown in Figure 10A, is not as large as that in the other 3 networks due to the shape of the Polska topology. Randomly generated spanning trees are correlated, and the path length between a given pair of nodes is comparable among the different particles. Nevertheless, PSOPR generates routes with an average path length of 2.51 hops, whereas routes generated by CPLEX are, on average, 3.06 hops long. Despite the small error that PSOPR achieves relative to CPLEX, i.e., 6% (shown in Figure 9B), the average rate utilization of PSOPR (39.4%) is less than that of CPLEX (58%) as shown in Figure 10B.

The execution time of PSOPR in the 4 considered networks of Abilene, Polska, Nobel-US, and Nobel-Germany is given by 0.2086, 0.912, 2.256, and 2.814 seconds, respectively, and can be related to the number of links in the network which is given by 15, 18, 21, and 26, respectively. The larger the number of links in the network is, the larger the time required to compute the SPs and the KSP; thus, the larger the execution time.

![FIGURE 11](image-url)  
Power consumption of Nobel-US under both particle swarm optimization–based and power-efficient routing (PSOPR) and CPLEX vs size of flow table. The population size of PSOPR $O$ increases from 100 to 1000.
5.5 Effect of population size on the performance of PSOPR in Nobel-US

In this subsection, we evaluate the impact of population size \( O \) on the performance of PSOPR in Nobel-US. The population size \( O \) takes on the following values: 100, 200, 300, 400, 500, and 1000 particles. We evaluate the power consumption, optimization error, and execution time for all experiments.

Figure 11 presents the power consumption of Nobel-US corresponding to routing solutions generated by both CPLEX and PSOPR. With 100 particles, the number of minimal spanning trees of PSOPR and the chance of finding an energy

FIGURE 12 Optimization error of particle swarm optimization–based and power-efficient routing in comparison to CPLEX for a number of population sizes \( O \) vs the size of the flow table

FIGURE 13 The execution time of CPLEX and particle swarm optimization–based and power-efficient routing for a number of population sizes.
The flow-rules constraint is relaxed. Furthermore, the average path length of the computed routes ranges from 62.07% to 82.02% of the optimal, and the average rate utilization falls in the range 52.36% to 67.93% of the optimal.

In our future work, we plan to optimize the routing and load balancing applications of SDN for power efficiency in wireless mobile networks.

**ACKNOWLEDGEMENTS**

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

**REFERENCES**


AUTHORS BIOGRAPHIES

Mohamad Khattar Awad, earned the BASc in electrical and computer engineering (communications option) from the University of Windsor, Ontario, Canada, in 2004 and the MASc and PhD in electrical and computer engineering from the University of Waterloo, Ontario, Canada, in 2006 and 2009, respectively. From 2004 to 2009, he was a research assistant in the Broadband Communications Research Group (BBCR), University of Waterloo. From 2009 to 2010, he was a lecturer at the Australian College of Kuwait and Adjunct Professor at the American University of Kuwait (AUK). In 2010, he joined AUK as an assistant professor of Electrical and Computer Engineering. Since 2012, he has been with Kuwait University as an Assistant Professor of Computer Engineering.

Dr Awad’s research interest includes wireless and wired communications, software-defined networks resource allocation, wireless networks resource allocation, and acoustic vector-sensor signal processing. He received the Ontario Research & Development Challenge Fund Bell Scholarship in 2008 and 2009, the University of Waterloo Graduate Scholarship in 2009, and a fellowship award from the Dartmouth College, Hanover, NH in 2011. In 2015, he received the Kuwait University Teaching Excellence Award. Dr Awad serves on the editorial board of the IEEE Transactions on Green Communications and Networking (TGCN). More information about him can be found on his webpage http://www.mohamadawad.com

Mohammed El-Shafei received his Bachelor of Engineering in Computer Engineering from American University of Kuwait in 2013. In 2016, he received his Masters of Science in Computer Engineering from Kuwait University. His search interest includes artificial intelligence and embedded systems.

Tassos Dimitriou is currently affiliated with the Department of Computer Engineering at Kuwait University (KU) and the Research and Academic Computer Technology Institute (CTI), Greece. Prior to that, he was an associate professor at Athens Information Technology, Greece, where he was leading the Algorithms and Security group, and adjunct professor in Carnegie Mellon University, USA, and Aalborg University, Denmark.

Dr Dimitriou conducts research in areas spanning from the theoretical foundations of cryptography to the design and implementation of leading edge efficient and secure communication protocols. Emphasis is given in authentication and privacy issues for various types of networks (adhoc, sensor nets, RFID, smart grid, etc), security architectures for wireless and telecommunication networks, and the development of secure applications for networking and electronic commerce. His research in the above fields has resulted
in numerous publications, some of which received distinction, and numerous invitations for talks in prestigious conferences.

Dr. Dimitriou is a senior member of IEEE, ACM, a Fulbright fellow, and distinguished lecturer of ACM. More information about him can be found in the web page http://tassosdimitriou.com/

**Yousef Rafique** received his Bachelor of Engineering in Computer Engineering from Kuwait University, Kuwait, in 2013. Currently, he is working towards his Master of Science Degree in Computer Engineering at Kuwait University. His research interests include energy optimization, software-defined networks, and cellular networks.

**Mohammed Baidas** received the BEng (first class honors) degree in communication systems engineering from the University of Manchester, Manchester, United Kingdom, in 2005; the MSc degree with distinction in wireless communications engineering from the University of Leeds, Leeds, United Kingdom, in 2006; the MS degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 2009; and the PhD degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2012. He is currently an assistant professor with the Department of Electrical Engineering, Kuwait University, Kuwait. His research interests include resource allocation and management in cognitive radio systems, game theory, and cooperative communications and networking.

**Ammar Alhusaini** is an assistant professor in the Computer Engineering Department at Kuwait University. He received a BS degree in computer engineering from Kuwait University and MS and PhD degrees in computer engineering from the University of Southern California. His research interest includes scheduling and resource management in grids, networks, and cloud environments. He is a member of IEEE, IEEE Computer Society, and ACM.

---

**APPENDIX**

**Hardness Results**

We consider a decision version of the power-efficient routing problem (PeRP) problem in which we ask whether it is possible to satisfy the flows without the total network power consumption exceeding a given bound. The following 2 theorems attempt to capture the hardness of the problem.

**Theorem 1.** PeRP is NP-hard.

**Proof.** We will show that PeRP is NP-hard via a reduction from the F-vertex disjoint path (F-VDP) problem which is known to be NP-complete. An instance of the F-VDP problem consists of a graph G and F pairs of vertices \{o(f), d(f)\}. The problem asks to determine if there exist F-VDPs \{P_1, P_2, \ldots, P_F\} of G such that each path \(P_f\) connects \(o(f)\) with \(d(f)\).

We can reduce an instance of the F-VDP problem to an instance of the PeRP as follows. We construct an SDN network from the input graph G, and we define a set of F flows, each with a rate of 1, \(r_f = 1\ \ \forall f \in F\), whose origin and destination coincide with the designate vertices of \(f\) and \(d(f)\), respectively. Additionally, we set the size of the table at each forwarding element (FE) equal to 1, ie, \(T_n = 1\ \ \forall N \in N\). Hence, the FE can only be used once to forward data towards some destination. To complete the reduction, we define only one operating rate \(R_0\) for each link. However, for the links originating from some \(o(f)\) or terminating at some \(d(f)\), we set the associated power consumption equal to 0. To see this, assume that there exist F-VDPs from each \(o(f)\) to the corresponding \(d(f)\) in the original graph G. Use such a path to transfer 1 unit of flow in the associated software-defined network. This is possible as the capacity constraint of each FE is not violated. Additionally, the total power consumption incurred on each path is equal to 2\(\gamma_0\) (due to the start and end vertices), for a total of 2\(\gamma_0\) for all disjoint paths. Conversely, if it is possible to ship F units of flow from each \(o(f)\) to \(d(f)\) without exceeding a total power of 2\(\gamma_0\) units, then each path must incur a cost of at most 2\(\gamma_0\), accounting for the starting and ending nodes of each flow. These paths must necessarily be vertex-disjoint as in each FE only 1 flow is admissible. Hence, this suggests that the existence of F disjoint paths in the underlying graph G.

In what follows, we show that the PeRP cannot even be approximated. Therefore, designing a heuristic like particle...
swarm optimization–based and power-efficient routing that works well in practice becomes necessary.

**Theorem 2.** For any constant \( \rho > 1 \), there is no polynomial time \( \rho \)-approximation algorithm for PeRP, unless \( P = NP \).

**Proof.** The F-edge disjoint paths (F-EDP) problem consists of a graph \( G \) and \( F \) pairs of vertices \( \{o(f), d(f)\} \). The problem asks to determine if there exist F-EDP \( \{P^1, P^2, \cdots, P^f, \cdots, P^F\} \) of \( G \) such that each path \( P^f \) connects \( o(f) \) with \( d(f) \). Here, we will show that if there exists a \( \rho \)-approximation algorithm \( A \) for the PeRP problem then the F-EDP problem can be decided in polynomial time. This would prove that \( P = NP \) as F-EDP is also NP-complete.31

So, assume that such an approximation algorithm \( A \) exists. Given an instance of the F-EDP problem, we construct an SDN network consisting of \( F \) flows, each with a rate of 1, whose origin and destination coincide with the designated vertices \( \{(o(f), d(f))\} \). For each link, we define 2 operating rates \( R_0 = 1 \) and \( R_1 = F \), with associated power costs \( \gamma_0 = 1 \) and \( \gamma_1 = L\rho + 1 \), where \( L \) is the number of links (edges) in the F-EDP instance. We complete the reduction by requiring all FEs to have a table of size \( F \); hence, an FE can be used to forward all \( F \) flows through it, if necessary.

Now, run the hypothetical approximation algorithm on the instance of PeRP. If there are \( F \) disjoint paths for the \( \{(o(f), d(f))\} \) pairs of the EDP problem then the network routing problem has a solution of total power at most \( L \). To see why notice that each link can be used at most once, hence operating at rate \( R_0 \) with cost 1. Thus, the cost of the optimal solution \( q^* \) cannot be more than \( L \). As \( A \) is a \( \rho \)-approximation algorithm for the PeRP problem, the cost of the solution found by it will be at most \( \rho |q^*| \leq \rho L \). On the other hand, if there are no edge-disjoint paths, then at least one of the links will operate at rate \( R_1 \), incurring a cost of at least \( \gamma_1 = L\rho + 1 \). Hence, the algorithm \( A \) can be used to determine if the F-EDP problem admits a solution in polynomial time. \( \square \)