IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 July 2022, accepted 1 August 2022, date of publication 8 August 2022, date of current version 15 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3197217

RESEARCH ARTICLE

Green Multistage Upgrade of Switches and Controllers for Bundled-Links SDNs

LELY HIRYANTO[1], (Member, IEEE), SIETENG SOH[1], (Member, IEEE), KWAN-WU CHIN[2], DUC-SON PHAM[1], (Senior Member, IEEE), AND MIHAI M. LAZARESCU[1], (Member, IEEE)

[1]School of Electrical Engineering, Computing, and Mathematical Sciences, Curtin University, Perth, WA 6102, Australia
[2]School of Electrical, Computer, and Telecommunications Engineering, University of Wollongong, Wollongong, NSW 2500, Australia

Corresponding author: Lely Hiryanto (lely.hiryanto@postgrad.curtin.edu.au)

This work was supported by the Australian Government through the Department of Foreign and Trade.

ABSTRACT The switches in legacy networks perform *both* control and data forwarding operations. In contrast, in a software defined network (SDN), a controller is responsible for network control and each SDN-switch (*s*-switch) only forwards data packets. Advantageously, controllers can be upgraded easily to run state-of-the-art network configuration and management solutions. In this context, this paper addresses the novel problem of upgrading a legacy network into an SDN over multiple stages that span months or years. More specifically, it aims to minimize energy consumption by optimizing switch upgrades, controller placement, and traffic routing subject to an operator's budget, traffic delay tolerance, and controller capacity. We formulate the problem as a mixed integer program (MIP), and develop a heuristic solution that ensures a single path is used between switches and up to two link-disjoint paths are used between an *s*-switch and its controller. Our simulation results show that increasing an operator's budget and the number of upgrade stages reduce the energy consumption of tested networks by 68.42%. Further, our heuristic solution yields energy saving that is within 5% away from the optimal value. In addition, deploying controllers at strategic locations saves more energy than placing them at arbitrary locations. Lastly, our heuristic solution guarantees the delay requirement of traffic demands and runs up to 307 times faster than a prior solution.

INDEX TERMS Controller placement, energy saving, green routing, hybrid SDN, network planning, switch upgrade.

I. INTRODUCTION

Software Defined Networks (SDNs) simplify network resource management. To elaborate, in legacy networks, *legacy*-switches or *l*-switches have both control and forwarding tasks. They use distributed network control, whereby each *l*-switch routes traffic according to local network information. Moreover, they contain both control and data plane. Further, the software used to manage network resources is usually fixed, which complicates network management when there is a need to upgrade *l*-switches to meet more sophisticated user requirements [1]. In contrast, an SDN decouples network control and forwarding tasks from legacy switches so that they become simple forwarding devices, where network control is performed by a centralized software-based entity called an SDN controller [1]–[3]. The controller uses global network information to generate rules to control the forwarding behavior of *s*-switches. As a result, an SDN has network programmability and better manageability, which help an operator to reduce its reliance on specific equipment vendors and adopt open networking, e.g., OpenFlow [4]. Hence, the aforementioned benefits have led to SDN technologies being adopted by enterprise networks such as Google [5], Microsoft public cloud [6], NTT cloud gateway [7], and IBM public cloud [8] to name a few.

A fundamental problem faced by operators is network planning, where they have to decide on a schedule to upgrade their network into an SDN over months or years. In this respect, an operator must consider three main issues: (i) available budget (in $), (ii) maturing SDN technology,

The associate editor coordinating the review of this manuscript and approving it for publication was Bijoy Chand Chand Chatterjee.

and (iii) depreciation in the cost of network equipment over time. These issues motivate operators to upgrade their *l*-switches over multiple periods or *stages* (in years). Specifically, at each stage, an operator replaces one or more *l*-switches with *s*-switches [2]. This thus creates a so called *hybrid* SDN containing a mix of *l*-switches and *s*-switches. In addition, an operator may deploy one or more controllers to manage these *s*-switches. A key design problem is thus to determine the minimum number of controllers, their location, and association between controllers and *s*-switches. This so called controller placement problem is non-trivial [9]. Further, an operator requires controllers to share the load of processing requests from existing and newly upgraded *l*-switches.

Recently, saving energy has become an important issue to network operators [3], [10], [11]. Current networks are known to over-provision network resources, e.g., link bandwidth, to satisfy traffic demands during peak hours; thus, networks are often under-utilized during off-peak periods [12]. Consequently, as shown in reference [12] some network resources, e.g., links, can be powered down to conserve energy. In this regard, an enabling technology is IEEE 802.11AX [13]. It creates so called *bundled*-links containing multiple physical cables. Advantageously, IEEE 802.1AX allows a network operator to scale the bandwidth or the number of cables in each link as per traffic demand [12]. More importantly, during off-peak hours, each unused IEEE 802.3az cable can be switched off [14]. Another approach to reduce energy consumption is via traffic rerouting. For example, the work in [12] optimizes traffic routing in legacy networks to maximize the number of unused cables that can be switched-off. To this end, an SDN architecture helps facilitate energy saving solutions, among others, energy-aware traffic routing [15]. More specifically, the centralized controller of an SDN has access to global network information that it can then use to compute a route that uses the minimum number of network resources such as links [16]. The controller then informs *s*-switches on the said route to forward traffic and turn off any unused links or/and cables. Apart from that, controller locations have a significant impact on energy consumption [11]. This in turn motivates research into energy-aware controller placement solutions [11] and traffic routing between *s*-switches and their associated controller [10].

This paper addresses a novel network planning problem called *Green Multi-Stage Upgrade of Switches and Controllers* (GMSU-SC). As shown in Fig. 1, the problem considers upgrading a legacy network that supports Open Shortest Path First (OSPF) and bundled-links to a *green* SDN over multiple planning stages. It aims to minimize energy consumption by maximizing the total number of unused cables that can be switched-off. GMSU-SC contains three sub-problems: (a) switch upgrade, (b) controller placement, and (c) traffic rerouting. Sub-problems (a) and (b) consider an operator's maximum budget and the depreciation rate of switch upgrade and controller deployment cost at each stage. Moreover, sub-problem (b) considers *control* traffic demand, termed control-packet-per-second or *cpps*, and the maximum processing capacity of each controller. Sub-problem (c) aims to increase the total number of unused cables. GMSU-SC also considers (i) single path routing of *data* traffic between switches, (ii) the use of up to two link-disjoint paths to route *control* traffic between an *s*-switch and a controller; one path is used to route traffic, while the other serves as a backup path, (iii) the Maximum Link Utilization (MLU) of each link, and (iv) only *s*-switches can turn off unused cables. Each path in (i) and (ii) must satisfy a given delay constraint.

We will now use Fig. 2 to illustrate GMSU-SC. The example network has nine nodes and 24 directed links. Each link has two cables, meaning the network has a total of 48 cables. The MLU of each cable is 80%, and has a propagation delay of one second. There are five data traffic demands, each routed via its shortest path; see the dashed-red lines in Fig. 2a. Assume an operator wishes to upgrade the network over two stages. Further, the operator requires a delay tolerance of no higher than 10%, and each controller can process packets from a maximum of three *s*-switches. Assume the budget in the *first* stage can be used to upgrade two *l*-switches, e.g., switches 6 and 8, and deploy one controller that is co-located with switch 6; see Fig. 2b. Both switches are associated to the controller. The control traffic from switch 8 to controller 6 is routed via path (8, 5, 6), shown as solid-blue lines. It uses (8, 9, 6) as a backup path (shown as a dotted-green line). Further, path (6, 5, 8) is used to route control traffic from node 6 to 8, while path (6, 9, 8) serves as a backup. Fig. 2b also shows that *flow 5* is rerouted from path (6, 3, 1) to (6, 5, 1). Assume each link that carries data and/or control traffic, e.g., link (5, 6), uses only one of its two cables. Thus, the unused cable of each such links can be powered-off. In contrast, there are eight links that are adjacent to *s*-switches 6 and 8, e.g., link (3, 6), that do not carry any traffic. Therefore, all the cables of these links can be switched-off. Thus, Fig. 2b contains $2 \times 8 + 4 = 20$ unused cables that can be switched-off. Assuming unused cables do not consume energy, and cables with traffic consume the same amount of energy, the energy saving for the example in Fig. 2b is $20/48 \times 100\% = 41.67\%$. In the *second* stage, see Fig. 2c, we assume a budget that can upgrade three more *l*-switches, namely switches 1, 2 and 4, and deploy a new controller. Assume the controller is co-located with *s*-switch 2 and manages switches 2 and 4, while switch 1 is managed by controller 6. Notice that the *s-c* traffic between switch 4 and controller 2 has no backup paths. This is because each path that is link-disjoint with paths (2, 4) and (4, 2) has a delay that exceeds the stated delay tolerance. The resulting hybrid SDN as shown in Fig. 2c has 38 unused cables. This equates to $38/48 \times 100\% = 79.17\%$ in energy saving. Thus, the average energy saving over the two stages is $(41.67\% + 79.17\%)/2 = 60.42\%$.

This paper makes the following *contributions*: (i) to the best of our knowledge, this paper is the first to consider the *joint* problem of *upgrading switches*, *controller placement*,

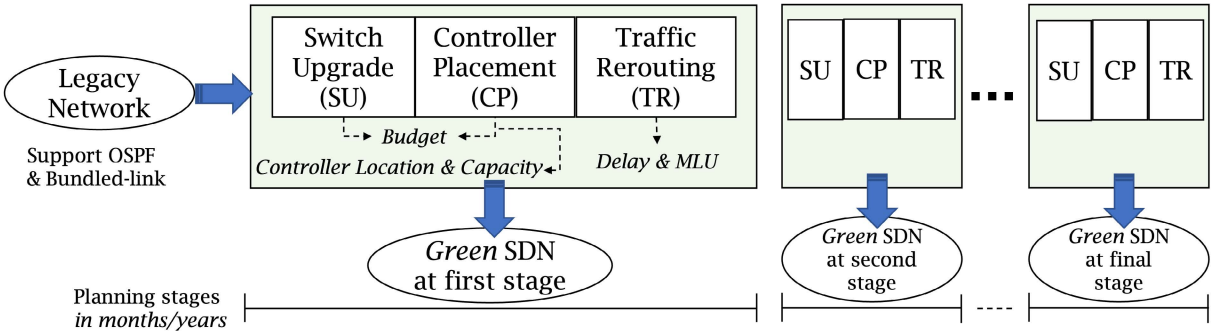


FIGURE 1. A diagram of GMSU-SC that shows the process of upgrading a network over planning stages that span months/years into an SDN. It further shows three sub-problems in each stage: switch upgrade, controller placement, and traffic rerouting. Further, for each sub-problem, an operator has to consider its budget, controller location and capacity, and also traffic delay and link utilization when routing traffic.

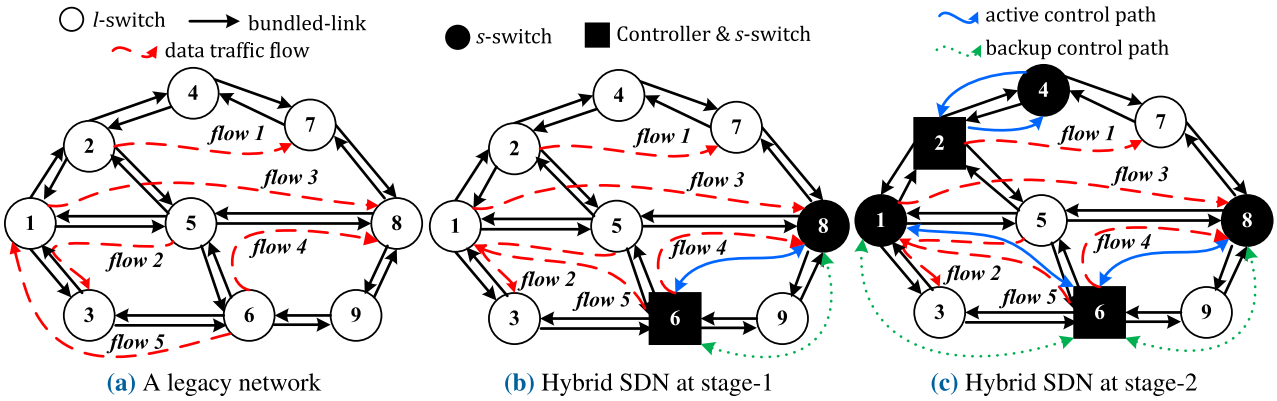


FIGURE 2. An illustration of GMSU-SC for two stages. At stage-1, *l*-switch 6 and 8 are upgraded and associated to a controller co-located with *s*-switch 6. At stage-2, new *s*-switch 1 is associated to controller 6, while new *s*-switch 2 and 4 are associated to a new controller that is co-located with *s*-switch 2.

and *traffic rerouting* to save energy over multiple planning stages. Our simulation shows that the proposed controller placement approach increases energy saving by up to 20.02% as compared to placing controllers at arbitrary locations, (ii) it outlines a novel mixed integer program (MIP) that can be used to determine the optimal solution for small-size networks, and (iii) it outlines the first complexity analysis of GMSU-SC, which is shown to be NP-Hard. Further, it proposes a heuristic solution for use in large-size networks, and also analyzes its run-time and space complexity.

The remainder of this paper is organized as follows. Section II discusses related works on energy saving, switch upgrade, controller placement, and traffic routing. Section III outlines our notation, problem formulation, and its complexity analysis. We present a heuristic solution in Section IV and evaluate its performance in Section V. Section VI concludes the paper.

II. RELATED WORK

This section discusses relevant works on energy saving in legacy networks, pure SDNs, and hybrid SDNs. Additionally, we include related works on hybrid SDNs, where the network upgrade is carried out over multiple stages. These works, however, do not aim to save energy.

A number of works aim to maximize energy saving in legacy networks that use IEEE 802.1AX links. For example, the authors of [12] propose an energy-aware routing to maximize the number of unused cables that can be switched off. They consider multi-path routing, MLU and delay requirements. The work in [11] proposes a controller placement solution that minimizes the energy consumption of links in a *pure* SDN, where each *l*-switch is upgraded to an *s*-switch. It proposes an energy aware controller placement solution called Improved Genetic Controller Placement Algorithm (IGCPA) for *pure* SDNs. Further, IGCPA uses GreCo [10] to associate switches and controllers in order to minimize the number of unused links. GreCo routes each data and control traffic demand via a single path. Moreover, each path used to route control messages cannot be longer than a given maximum path delay.

Works such as [16]–[19] have considered energy saving in hybrid SDNs. They assume OSPF routing for all *l*-switches and consider links with a single cable. Among them, the authors of [16]–[18] minimize the power consumption of *hybrid* SDNs by optimizing network traffic routing with

a given set of *s*-switches. On the other hand, the work in [19] minimizes the power consumption of a network by optimizing both switch upgrade and shortest path traffic routing. However, the work in [16]–[19] does not address controller placement. In contrast, reference [20] jointly optimizes switch upgrades and controller placement under a given budget constraint over a single stage. It aims to optimize the number of flows passing *s*-switches and *s*-*c* traffic propagation delays.

A number of works have considered network upgrade over multiple stages; e.g., [21]–[23], and [24]. The work in [21] considers switch upgrade and controller placement. Its goal is to minimize flow-setup delay, path failure, and controllers load given a limit on the number of controllers and their capacity. The authors of [22] aim to maximize traffic flow passing through *s*-switches by replacing *l*-switches with *s*-switches over multiple stages. They consider switch upgrade cost and changes in data traffic (in bytes) over multiple stages and budget constraints. Further, the work in [22] uses shortest paths to route data traffic. On the other hand, our recent work in [23] and [24] jointly addresses switch upgrade and data traffic routing to maximize the number of unused cables. Both works consider a maximum budget at each upgrade stage, MLU, and a given delay tolerance. Other constraints relate to the decreasing and increasing rate of upgrade cost and data traffic, respectively. Further, the work in [24] proposes the use of non-link-disjoint or link-disjoint paths. Our previous works and the work in [22], however, do not address controller placement.

Table 1 summarizes the aforementioned related works. Observe that there are no works on energy saving that *jointly* consider switch upgrade, controller placement and traffic routing. We emphasize that a solution that considers a subset of these features independently is unlikely to yield an optimal energy saving solution. Our work, similar to prior research such as [12], [23], [24], aims to maximize energy saving by switching off the maximum number of unused cables. However, reference [12] does not consider network upgrade. On the other hand, unlike the prior work in [16]–[19] and [22]–[24], we consider controller placement when carrying out switch upgrade and traffic routing. Different to works such as [11] and [21], our goal is to minimize energy consumption under the following constraints: 1) maximum budget that covers *both* the cost to upgrade a set of *l*-switches and deploy a set of controllers at each stage, and 2) control traffic routing via active and backup paths that are link-disjoint and satisfy a given maximum path delay and MLU threshold.

III. PROBLEM FORMULATION

The next section describes the system model. Table 2 summarizes our notations. After that, Section III-B presents a mathematical model of GMSU-SC. Finally, Section III-C analyzes the complexity of our problem.

TABLE 1. A summary of related works.

Scheme	References	Scope			
		ES	SU	TR	CP
Legacy Networks	[12]	✓	×	✓	×
Pure SDNs	[10], [11]	✓	×	✓	✓
Hybrid SDNs (one stage)	[16], [17], [19]	✓	✓	✓	×
	[20]	×	✓	✓	✓
Hybrid SDNs (multi stage)	[22]	×	✓	✓	×
	[21]	×	✓	✓	✓
	[23], [24]	✓	✓	✓	×
	Our work	✓	✓	✓	✓

ES: Energy Saving SU: Switch Upgrade
TR: Traffic Routing CP: Controller Placement

A. SYSTEM MODEL AND NOTATIONS

In the next section, we first present a model of our network, followed by switch-controller association, traffic flow, routing, budget, and energy saving.

1) NETWORK

Let $G^0(V, E)$ be a *legacy* network that uses Open Shortest Path First (OSPF) [25]. The network has $|V|$ nodes that represent the location of *l*-switches, where $|.|$ is the cardinality of a set. There are $|E|$ *directed* links. The set E contains both link (u, v) and (v, u), for $u, v \in V$ and $u \neq v$. Let b_{uv} be the number of cables or the *bundle size* of each link $(u, v) \in E$. All cables have capacity γ (in bytes), meaning the total capacity of link (u, v) is $c_{uv} = \gamma \times b_{uv}$. We assume the legacy network $G^0(V, E)$ has sufficient link capacity to carry the highest traffic volume ω_d^T for each demand via its shortest path.

For each stage t, an operator upgrades a set of *l*-switches to *s*-switches and deploys one or more controllers to manage *s*-switches. We use V^t and C^t respectively to denote the set of *s*-switches and controllers that have been installed up to stage t, respectively. Let $G^t(V, E)$ be the hybrid SDN after undergoing an upgrade at stage $t \leq T$, where the term T, with $T \geq 1$, is the planning horizon. The duration of each stage t is determined by the lifetime of a network device; e.g., three to five years [22]. An *s*-switch v is a switch located at node $v \in V$, denoted by $\mathcal{S}(v)$. Each *s*-switch is able to communicate with both *l*-switch and *s*-switch, e.g., a hybrid OpenFlow switch [26]. We use $\mathcal{C}(v)$ to denote a controller at node v. For each controller $\mathcal{C}(v)$, node v must contain an *s*-switch, i.e., each controller must be *co-located* with an *s*-switch; this reduces traffic delay and ensures high communication reliability [27]. We call an *s*-switch that is co-located with a controller a *co*-switch. Hence, for each *co*-switch v, node v represents both a controller $\mathcal{C}(v) \in C^t$ and an *s*-switch $\mathcal{S}(v) \in V^t$. For brevity, unless it is necessary to differentiate between a node v and an *s*-switch or a controller at node v, this paper uses v and $\mathcal{S}(v)$ as well as v and $\mathcal{C}(v)$ interchangeably.

TABLE 2. Notations and definitions.

1.	**Network**
$G^0(V,E)$	A legacy network with $\|V\|$ nodes and $\|E\|$ links.
T	The total number of upgrade stages.
$G^t(V,E)$	An upgraded legacy network.
$c_{uv}(b_{uv})$	The capacity (bundle size) of link $(u,v) \in E$.
π_{uv}	The propagation delay of link $(u,v) \in E$.
$V^t(C^t)$	The set of s-switches (controllers) at stage t.
2.	**Switch-Controller Association**
$q^t_{v,a}$	An indicator if switch v is associated with controller a at stage t.
A^t_a	The set of s-switches associated to controller a at stage t.
$\hat{\omega}^t_v$	The number of control packets per second originated from switch v at stage t.
$\hat{\mu}_v$	The increase rate of $\hat{\omega}^t_v$ per stage.
θ	Controller capacity.
3.	**Traffic Flow**
$(s_d, \tau_d, \omega^t_d)$	A traffic demand in D^t from source s_d to destination τ_d with traffic size ω^t_d.
D^t	The set of s-s and s-c traffic demands in $G^t(V,E)$.
μ_d	The increase rate of traffic demand d per stage.
β	The average size of a control packet.
4.	**Routing**
$\delta_{\max,d}$	The maximum path delay requirement for demand d.
$\mathcal{P}_d$	The set of alternative paths within $\delta_{\max,d}$ for demand d
R^t_d	The set of paths to route each demand d in D^t.
5.	**Budget**
B (B^t)	The maximum budget over T stages (at each stage t).
$\rho(\hat{\rho})$	The decrease rate in switch (controller) cost per stage.
p^t_v	The upgrade cost for switch $v \in V$ at stage t.
$\hat{p}^t$	The deployment cost of a controller at stage t.
6.	**Energy Saving**
$U_{\max}$	MLU threshold.
n^t_{uv}	The number of *on*-cables for link (u,v) at stage t.
ε^t	The energy saving at stage t.
ε_T	The average energy saving over T stages.

2) SWITCH-CONTROLLER ASSOCIATION

At any stage t, each s-switch v must be associated to a controller $a \in C^t$. Let $q^t_{v,a}$ be a binary indicator that is set to one when switch v is associated to a controller a at stage t. Let $A^t_a \subseteq V^t$ be a set of all s-switches that are associated to controller a at stage t. Each *co*-switch is permanently associated to its controller. However, the association of other s-switches may change in subsequent stages. Define the set A^t to record all switch-to-controller associations at stage t; formally, we have $A^t = \{A^t_a \mid a \in C^t\}$. Let $\hat{\omega}^t_v$ denote the number of control-packet-per-second or *cpps* that originates from s-switch v at stage t. The value of $\hat{\omega}^t_v$ increases at a rate of $\hat{\mu}_v \geq 0$ per stage, i.e., $\hat{\omega}^{t+1}_v = \lceil \hat{\omega}^t_v \times (1+\hat{\mu}_v) \rceil$, for $t < T$. We assume each controller generates the same *cpps* to each of its associated s-switch at every stage. In addition, each controller has limited capacity. As per [28], [29], and [21], we assume all controllers have the same capacity θ (in *cpps*). Consequently, for a given controller, at each stage t, the total *cpps* from all its associated s-switches must be less than θ, i.e., $\sum_{v \in A^t_a} \hat{\omega}^t_v \leq \theta$.

3) TRAFFIC FLOW

There are three types of traffic: (i) *switch-switch* or *s-s*, (ii) *switch-controller* or *s-c*, and (iii) *controller-controller* or *c-c*. Traffic *s-s* contains data packets between s-switches, a pair of l-switches, or an s-switch and an l-switch. Traffic *s-s* originates from an s-switch or an l-switch. Note that l-switches rely on OSPF to determine the least cost path between switches. Traffic type *s-c* corresponds to control packets between an s-switch and its controller. For example, an s-switch generates *s-c* traffic when requesting forwarding rules for a new *s-s* traffic flow or when it sends its status [30]. On the other hand, a controller generates *s-c* traffic when it sends forwarding rules and maintenance information [30]. Both *s-s* traffic and *s-c* traffic share the same communication link or *in-band channel*. Lastly, traffic *c-c* represents control packets between controllers. This paper ignores *c-c* traffic because it is typically routed via a dedicated control network to ensure fast and reliable communication [29], [31].

A traffic demand is a single commodity with a source node, a destination node, and volume. Let $D^t = \{(s_d, \tau_d, \omega^t_d) \mid \forall d \in \{1, \ldots, |D^t|\}\}$ represent a set of *s-s* and *s-c* traffic demands at stage t. Each demand $d \in \{1, \ldots, |D^t|\}$ contains a source node $s_d \in V$, a destination node $\tau_d \in V$, and traffic volume $\omega^t_d > 0$ (in bytes). The traffic volume for each demand d increases with each successive stage $t < T$ at a rate of $\mu_d \geq 0$. Thus, we have $\omega^{t+1}_d = \omega^t_d \times (1+\mu_d)$. Let $D^t_{\text{ss}} \subset D^t$ and $D^t_{\text{sc}} \subset D^t$ denote the sets of *s-s* and *s-c* demands at stage t, respectively. We have $D^t_{\text{ss}} \cup D^t_{\text{sc}} = D^t$, $D^t_{\text{ss}} \cap D^t_{\text{sc}} = \phi$, and demand $d > |D^t_{\text{ss}}|$ refers to an *s-c* demand. Note that $D^0_{\text{ss}} = D^1_{\text{ss}}$ is a given set of *s-s* traffic demands in $G^0(V,E)$, i.e., aggregated traffic demands at a given time, and ω^0_d is the initial traffic volume of demand d. We assume the total number of *s-s* demands remains the same over T stages, i.e., $|D^t_{\text{ss}}|$ is constant. Each *s-c* demand in D^t_{sc} exists only if there is an association between s-switch $s_d \in V^t$ and controller $\tau_d \in C^t$ at stage t, i.e., $q^t_{s_d,\tau_d} = 1$ and $q^t_{\tau_d,s_d} = 1$. For each association, there are two possible *s-c* demands in D^t_{sc}: (i) from a switch to controller, i.e., $(s_d, \tau_d, \omega^t_d)$, and (ii) from a controller to a switch, i.e., $(s_{d+1}, \tau_{d+1}, \omega^t_{d+1})$, where $s_d = \tau_{d+1} \in V^t$ and $\tau_d = s_{d+1} \in C^t$. Let β denote the average size

of a control packet (in bytes). Thus, the traffic volume of each *s-c* demand is computed as $\omega_d^t = \hat{\omega}_{s_d}^t \times \beta$, and $\omega_{d+1}^t = \omega_d^t$.

4) ROUTING

Let P_d denote a set of alternative paths that can be used to route a demand d, and $P_{d,i}$ is the i-th path, i.e., $P_d = \{P_{d,i} \mid \forall i \in \{1, \ldots, |P_d|\}\}$. We use π_{uv} (in seconds) to represent the propagation delay of link (u, v). The delay over path $P_{d,i}$, denoted by $\delta_{d,i}$ (in seconds), is computed as the sum of propagation delays over all its links, i.e., $\delta_{d,i} = \sum_{(u,v)\in P_{d,i}} \pi_{uv}$. We ignore transmission and queuing delay. This is because propagation delay dominates when a network spans a large geographical distance [32]. We have $\delta_{d,i} = 0$ for each demand d from a *co*-switch to its controller, and vice-versa. The reason is because the switch and controller are located at the same node.

Let $\delta_{\min,d}$ be the minimum delay among all paths in P_d, and $\delta_{\max,d} = \lceil \sigma \times \delta_{\min,d} \rceil$ be the maximum path delay, for a delay multiplier $\sigma \in [1.0, 2.0]$. A path $P_{d,i} \in P_d$ is called the *shortest path* if its delay is equal to the minimum delay $\delta_{\min,d}$. We assume users can tolerate delays up to $\delta_{\max,d}$. Let $\mathcal{P}_d \subset P_d$ denote a set of paths; each of which has delay within $\delta_{\max,d}$. Define $R_d^t \subseteq \mathcal{P}_d$ as a set of paths that are used to route demand d at stage t. Each *s-s* demand $d \in D_{ss}^t$ is routed via a single path; thus we have $|R_d^t| = 1$. On the other hand, each *s-c* demand $d \in D_{sc}^t$ uses up to two (s_d, τ_d) link-disjoint paths; thus, we have $|R_d^t| = 1$ or $|R_d^t| = 2$. For $|R_d^t| = 2$, one path, called the *active* path is used to route demand d, while the other serves as *backup*. Note that there is no path for each *s-c* demand d between a *co*-switch and its controller, i.e., for this case we have $R_d^t = \phi$. We use $R^t = \{R_d^t \mid d \in \{1, \ldots, |D^t|\}\}$ to denote the set of paths to route *s-s* and *s-c* demands at stage t.

5) BUDGET

Let B denote the total budget (in \$) that can be spent over T stages, and $B^t \leq B$ is the allocated budget at each stage t. We set $B^t = B/T$. Any unused budget in stage t, denoted by ΔB^t, can be spent in subsequent stages, i.e., $B^{t+1} = B^{t+1} + \Delta B^t$. We use p_v^t and $\hat{p}^t$ to denote the upgrade cost of switch v and the deployment cost of each controller at stage t, respectively. Both costs include the purchase price and installation cost. The upgrade cost of a switch corresponds to its processing capability; i.e., whether it is an edge or core switch [22]. Thus, we assume a switch with larger *cpps* is more expensive than a switch with lower *cpps*. All controllers have the same deployment cost and capacity θ. We use ρ and $\hat{\rho}$ to denote the per-stage depreciation rate of switch upgrade cost and controller deployment cost, respectively; here rate ρ and $\hat{\rho}$ satisfy $0 \leq \rho, \hat{\rho} < 1$. Thus, the upgrade cost of a switch v at stage t is $p_v^t = p_v^0 \times (1 - \rho)^{t-1}$. Similarly, the cost to deploy each controller at stage t is $\hat{p}^t = \hat{p}^0 \times (1 - \hat{\rho})^{t-1}$. Note that p_v^0 and $\hat{p}^0$ are the initial cost of upgrading each switch v and deploying a controller, respectively.

6) ENERGY SAVING

Let *c*-link be a link (u, v) that is adjacent to at least one *s*-switch, i.e., $u \in V^t$ and/or $v \in V^t$; otherwise the link is an *l*-link. Further, we call a cable that carries traffic an *on*-cable; otherwise the cable is an *unused* cable. As per [17] and [19], each unused cable in a *c*-link can be switched-off by powering off its line card. Without loss of generality, this paper assumes that *only unused* cable in *c*-link can be switched-off. The reason is because we assume each *l*-switch does not comply with the IEEE 802.3az [33] standard. As per [12], we assume each *on*-cable consumes the same amount of energy. Thus, this paper computes network energy usage from the total number of *on*-cables. Alternatively, energy saving can be computed from the total number of unused cables that can be switched-off, called *off*-cables. Let f_{uv}^t denote the total volume of traffic demands carried by link (u, v) at stage t. Further, let n_{uv}^t, for $0 \leq n_{uv}^t \leq b_{uv}$, be the required number of *on*-cables to carry traffic volume f_{uv}^t. Also, let $U_{\max}$ be the maximum link utilization threshold, for $0 \leq U_{\max} \leq 1$. Thus, we have $n_{uv}^t = \lceil f_{uv}^t / (\gamma \times U_{\max}) \rceil$.

Formally, the energy saving at stage t, denoted by ε^t, is computed as

$$\varepsilon^t = \frac{\sum_{(u,v)\in E} (b_{uv} - n_{uv}^t)}{\sum_{(u,v)\in E} b_{uv}}. \tag{1}$$

In other words, the energy saving ε^t is a ratio between the total number of powered-*off* cables or *off*-cables and the total number of cables. Note that we set $n_{uv}^t = b_{uv}$ in Eq. (1) for each *l*-link because we assume cables in *l*-link cannot be switched-off. The average energy saving over T stages is then computed as

$$\varepsilon_T = \frac{1}{T} \sum_{t=1}^{T} \varepsilon^t. \tag{2}$$

B. MATHEMATICAL MODEL

Given a legacy network $G^0(V, E)$ that supports OSPF, our optimization problem, i.e., GMSU-SC, upgrades a subset of *l*-switches with a set of *s*-switches V^T and deploys a set of controllers C^T over $T \geq 1$ stages. The aim is to maximize energy saving as per Eq. (2) by shutting down the maximum number of unused cables. We formulate GMSU-SC as a mixed integer linear program, called MIP-SC. Its objective, see Eq. (3), is to minimize the number of *on*-cables over T stages. This objective is equivalent to maximizing the energy saving as quantified by Eq. (2). Formally,

$$\min \sum_{t=1}^{T} \sum_{(u,v)\in E} n_{uv}^t$$
$$\text{s.t. } (4) - (22). \tag{3}$$

GMSU-SC considers three main constraints, i.e., (i) *Budget*: the maximum budget B^t at each stage t must be sufficient to upgrade switches and deploy controllers, (ii) *Controller*

placement: a controller must be co-located with an *s*-switch, each *s*-switch must be associated to exactly one controller, and the total switches being associated to a controller cannot exceed its maximum capacity, and (iii) *Traffic routing*: each traffic must be routed via a single path that must satisfy delay tolerance and MLU requirements, and each s-c traffic, when possible, has two link-disjoint paths: an *active* path to route traffic, and a *backup* path. Next, we elaborate these three constraints in detail.

1) BUDGET CONSTRAINTS

Constraint (4) guarantees that the total cost to upgrade *l*-switches and to deploy controllers at each stage t does not exceed the maximum budget B^t. Let x_v^t ($\hat{x}_v^t$) be an indicator that is set to one when an *s*-switch (controller) is deployed at node v at stage t. Constraint (5) and (6) ensure that each switch and each controller are deployed only once. Constraint (7) enforces all cables in *l*-links to be turned on. Note, only unused cables of *c*-links can be turned off.

$$\sum_{v\in V}(p_v^t x_v^t + \hat{p}^t \hat{x}_v^t) \le \sum_{k=1}^{t} B^k - \sum_{k=1}^{t-1}\sum_{v\in V}(p_v^k x_v^k + \hat{p}^k \hat{x}_v^k), \tag{4}$$

$$\sum_{t=1}^{T} x_v^t \le 1, \tag{5}$$

$$\sum_{t=1}^{T} \hat{x}_v^t \le 1, \tag{6}$$

$$n_{uv}^t = \max\left(n_{uv}^t, b_{uv}\left(1 - \sum_{k=1}^{t} x_u^k - \sum_{k=1}^{t} x_v^k\right)\right). \tag{7}$$

2) CONTROLLER PLACEMENT CONSTRAINTS

Next, constraint (8) guarantees that each controller is co-located with an *s*-switch. Constraint (9) ensures that each controller is associated to its *co*-switch. In constraint (10), each *s*-switch must be associated to exactly one controller. As per constraints (11) and (12), each controller must be associated to at least one *s*-switch and the total generated control packets is no larger than the capacity of the controller. Constraint (13) ensures that an *s*-*c* association exists, i.e., $q_{v,a}^t = 1$, if and only if *l*-switch v is upgraded to an *s*-switch and a controller is deployed at node a.

$$\sum_{k=1}^{t} \hat{x}_v^k \le \sum_{k=1}^{t} x_v^k, \tag{8}$$

$$\sum_{k=1}^{t} x_v^k \sum_{k=1}^{t} \hat{x}_v^k \le q_{v,v}^t, \tag{9}$$

$$\sum_{a\in V} q_{v,a}^t = \sum_{k=1}^{t} x_v^k, \tag{10}$$

$$\sum_{v\in V} q_{v,a}^t \ge \sum_{k=1}^{t} \hat{x}_a^k, \tag{11}$$

$$\sum_{v\in V}(\hat{\omega}_v^t q_{v,a}^t) \le \theta, \tag{12}$$

$$q_{v,a}^t \le \min\left(\sum_{k=1}^{t} x_v^k, \sum_{k=1}^{t} \hat{x}_a^k\right). \tag{13}$$

3) TRAFFIC ROUTING CONSTRAINTS

Constraint (14) sets indicator $\mathcal{I}_d$ to one if $d \in \{1, \ldots, |D^t|\}$ is an *s*-*s* demand or an *s*-*c* demand; the latter applies only when *s*-switch s_d is associated to controller τ_d, for $s_d \neq \tau_d$. Otherwise, the indicator is set to zero. Recall that d is an *s*-*s* (*s*-*c*) demand if $d \le |D^t|$ ($d > |D^t|$). Constraints (15) to (20) are binding only if we have $\mathcal{I}_d = 1$. Let K_d be the number of paths used to route demand d. We set $K_d = 1$ for each *s*-*s* demand. For an *s*-*c* demand, we set $K_d = 2$ if set $\mathcal{P}_d$ contains two-link-disjoint paths; otherwise, it is $K_d = 1$. Constraint (15) ensures flow conservation for both *s*-*s* and *s*-*c* traffic. For each *s*-*c* demand d with $K_d = 2$ paths, constraint (16) ensures the two paths are link-disjoint. Constraints (17) and (18) enforce every selected path i to meet its maximum delay and capacity $c_{uv} \times U_{\max}$ of each link (u, v) on the path, respectively. We use $n_{uv,\text{all}}^t$ to denote the required number of *on*-cables per link when all selected paths for each *s*-*c* demand d carry the same amount of control packets ω_d^t. Constraints (19) and (20) guarantee only one path, indicated by $y_{d,i}^t = 1$, that routes either a *s*-*s* or *s*-*c* demand. Constraint (21) limits the number of *on*-cables n_{uv}^t to the maximum *on*-cables $n_{uv,\text{all}}^t$, which is not larger than the bundle size of each link $(u, v) \in E$.

$$\mathcal{I}_d = \begin{cases} 1, & d \in \{1, \ldots, |D_{\text{ss}}^t|\} \\ q_{s_d,\tau_d}^t + q_{\tau_d,s_d}^t, & d > |D_{\text{ss}}^t| \end{cases}, \tag{14}$$

$$\sum_{i=1}^{K_d}\sum_{(u,v)\in E} y_{d,uv,i}^t - \sum_{i=1}^{K_d}\sum_{(v,u)\in E} y_{d,vu,i}^t = \begin{cases} \mathcal{I}_d, & u = s_d \\ -\mathcal{I}_d, & u = \tau_d \\ 0, & u \neq s_d, \tau_d \end{cases}, \tag{15}$$

$$\sum_{i=1}^{K_d} y_{d,uv,i}^t \le \mathcal{I}_d, \tag{16}$$

$$\sum_{(u,v)\in E}(y_{d,uv,i}^t \pi_{uv}) \le \delta_{\max,d}\mathcal{I}_d, \tag{17}$$

$$\sum_{d=1,\mathcal{I}_d=1}^{|D^t|}\sum_{i=1}^{K_d}(y_{d,uv,i}^t \omega_d^t) \le (n_{uv,\text{all}}^t / b_{uv}) U_{\max} c_{uv}, \tag{18}$$

$$\sum_{i=1}^{K_d} y_{d,i}^t = \mathcal{I}_d, \tag{19}$$

$$\sum_{d=1,\mathcal{I}_d=1}^{|D^t|}\sum_{i=1}^{K_d}(y_{d,i}^t y_{d,uv,i}^t \omega_d^t) \le (n_{uv}^t / b_{uv}) U_{\max} c_{uv}, \tag{20}$$

$$0 \le n_{uv}^t \le n_{uv,\text{all}}^t \le b_{uv}, \tag{21}$$

Finally, constraint (22) defines decision variables x_v^t, $\hat{x}_v^t$, $q_{v,a}^t$, $\mathcal{I}_d$, $y_{d,uv,i}^t$, and $y_{d,i}^t$ to be binary.

$$x_v^t, \hat{x}_v^t, q_{v,a}^t, \mathcal{I}_d, y_{d,uv,i}^t, y_{d,i}^t \in \{0, 1\}. \tag{22}$$

Note that constraint (4) - (22), except (5) and (6), apply to each stage $t \in \{1, \ldots, T\}$. Constraint (5), (6), and (8)–(12) apply to all nodes in V. Constraint (7), (18), (20), and (21) exist for each link $(u, v) \in E$. Constraint (13) is for each node $v \in V$ and every node $a \in V$. Constraint (14) and (19) apply to each demand $d \in \{1, \ldots, |D^t|\}$. Finally, constraint (15) - (17) consider all demands. Moreover, constraint (15), (16), and (17) are also evaluated for every node $u \in V$, each link $(u, v) \in E$, and each of K_d paths, respectively.

C. PROBLEM COMPLEXITY

GMSU-SC is related to an NP-hard Multiple Knapsack Problem (MKP) problem [34]. Briefly, MKP considers T knapsacks and m items. Each knapsack has a maximum weight capacity, and each item has a profit and weight. The objective of MKP is to select T-disjoint subsets of items that maximize the total profit such that the total weight of each subset is no larger than its knapsack's capacity. Notice that the following parameters as used in GMSU-SC and MKP, respectively, are equivalent: (i) T upgrade stages and T knapsacks, (ii) $|V|$ l-switches and m items, (iii) maximum budget B^t at each stage t and capacity of each knapsack, (iv) upgrade cost p_v^t and weight of each item, and (v) *off*-cables adjacent to the switch and profit of each item. Further, the objective of GMSU-SC, i.e., selecting T-disjoint subsets of s-switches that maximize the number of *off*-cables, is equivalent to selecting T-disjoint subsets of items that maximize the total profit. However, GMSU-SC considers the following additional parameters that do not exist in MKP: (i) depreciation rate of each switch upgrade cost, (ii) controller deployment cost, its location, and its association to s-switches, and (iii) paths to route s-s and s-c demands. In this case, MKP is equivalent to a special case of GMSU-SC when (i) there is no depreciation rate for each switch upgrade cost and controller deployment cost, (ii) the locations of controllers and their association to s-switches are given, and (iii) the paths to route s-s and s-c traffic demands are fixed, i.e., the number of *on*-cables for each link is known. Thus, GMSU-SC is at least as hard as MKP. The next section describes a heuristic solution for GMSU-SC.

IV. HEURISTIC SOLUTION

This section outlines **HGMSU-SC**. Specifically, Section IV-A contains the details of HGMSU-SC. After that Section IV-B presents an analysis of HGMSU-SC.

A. DETAILS OF HGMSU-SC

As shown in Fig. 3, HGMSU-SC consists of three parts: 1) routing initialization, 2) network upgrade, and 3) traffic rerouting; see Algorithm 1. Part 1) generates the initial routing for all s-s traffic demands in set D_{ss}^0. It uses the said routing to obtain the total number of unused cables that are adjacent to each l-switch $u \in V$, denoted by w_u. Each switch's *off*-cables, i.e., w_u, is the input to Part 2). Specifically, Part 2) determines the deployment of s-switches and controllers that maximize the number of *off*-cables over T stages. Part 3) increases, if possible, the number of *off*-cables obtained in Part 2). The details of Part 1), 2), and 3) are given in Section IV-A1, IV-A2, and IV-A3, respectively.

Algorithm 1 HGMSU-SC

Input: $G^0(V, E)$, T, B, D_{ss}^0, p_v^0, $\hat{p}_v^0$, $U_{\max}$, μ_d, ρ, $\hat{\rho}$, σ
Output: V^t, C^t, A^t, D^t, R^t, n_{uv}^t, ε^t

▷ *Part 1: routing initialization*
1: **for** $(d \in \{1, \ldots, |D_{\text{ss}}^0|\})$ **do**
2: Generate set $\mathcal{P}_d$
3: $R^0 = R^0 \cup \mathcal{P}_{d,1}$
4: $f_{uv}^T = f_{uv}^T + \omega_d^T$ for each $(u, v) \in R_d^0 \in R^0$
5: **end for**
6: $n_{uv}^T = \lceil f_{uv}^T / (\gamma \times U_{\max}) \rceil$ for each $(u, v) \in E$
7: Compute w_u for each $u \in V$ using Eq. (23)
▷ *Part 2: network upgrade*
8: **for** $(t \in \{1, \ldots, T\})$ **do**
9: $\{V^t, C^t, A^t, D^t, R^t, \Delta B^t, L\}$ = **Deployment**$(V^{t-1}, C^{t-1}, A^{t-1}, D^{t-1}, R^{t-1}, B^t)$
10: $B^{t+1} = B^{t+1} + \Delta B^t$
▷ *Part 3: traffic rerouting*
11: R^t = **GTE-SC**(V^t, R^t, L)
12: Compute ε^t using Eq. (1)
13: $\varepsilon_T = \varepsilon_T + \varepsilon^t / T$
14: **end for**

1) ROUTING INITIALIZATION

Part 1) of Algorithm 1 initially routes each s-s demand as per OSPF. It then uses the initial route of all s-s demands to calculate each switch's total *off*-cables. More specifically, Line 2 of HGMSU-SC uses Yen's algorithm [35] to generate the first k shortest paths $\mathcal{P}_d$ for each s-s demand $d \in \{1, \ldots, |D_{\text{ss}}^0|\}$ in increasing order of their delay, i.e., $\mathcal{P}_d = (\mathcal{P}_{d,1}, \mathcal{P}_{d,2}, \ldots, \mathcal{P}_{d,k})$, where path $\mathcal{P}_{d,1}$ has the shortest delay. Let R^0 be a set containing the shortest path for all s-s demands, i.e., $R^0 = \{\mathcal{P}_{d,1} \mid \forall d \in \{1, \ldots, |D_{\text{ss}}^t|\}\}$. Line 3 adds the shortest path $\mathcal{P}_{d,1}$ of each s-s demand d to set R^0. Line 4 computes the largest required capacity, denoted by f_{uv}^T for each link $(u, v) \in E$, when each demand d is routed by the OSPF protocol. Recall that the traffic at the last stage, i.e., ω_d^T, has the maximum volume. Line 6 then calculates the number of *on*-cables n_{uv}^T of link (u, v) based on f_{uv}^T. Line 7 computes the total number of *off*-cables w_u that are adjacent to each switch $u \in V$ as follows:

$$w_u = \sum_{(u,v) \in E} (b_{uv} - n_{uv}^T), \quad u \in V. \tag{23}$$

The motivation for using $b_{uv} - n_{uv}^T$ in Eq. (23) is due to the following observation:

Observation 1: Upgrading a set of l-switches with the highest total number of unused cables at the earliest possible stage can maximize energy saving over T stages.

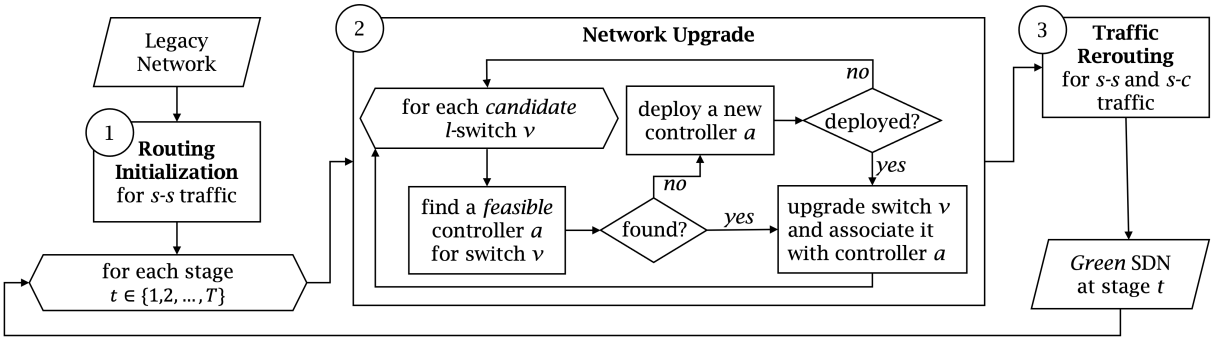


FIGURE 3. A flowchart of HGMSU-SC. It consists of three parts: routing initialization, network upgrade, and traffic rerouting.

To explain Observation 1, recall that the volume of demand d grows at a rate of $\mu_d \geq 0$ for each subsequent stage $t \in [2, \ldots, T]$. Thus, if a link (u, v) that has n^T_{uv} number of *on*-cables is able to carry traffic demand at stage T, these *on*-cables are also sufficient to carry traffic demands at any stage $t < T$. This implies that we have $n^t_{uv} \leq n^{t+1}_{uv}$ and $(b_{uv} - n^t_{uv}) \geq (b_{uv} - n^{t+1}_{uv})$ for each link (u, v). In this case, the number of unused cables $(b_{uv} - n^t_{uv})$ at stage t includes the $(b_{uv} - n^T_{uv})$ unused cables that can remain off in stage $t + 1$. An l-switch u with the highest w_u also has the highest number of unused cables at stage $t < T$.

2) NETWORK UPGRADE

For each stage t, Line 9 of Algorithm 1 calls **Deployment**() to upgrade a set of l-switches into s-switches that can turn off the maximum possible number of unused cables, and associate each of them to a controller. The function **Deployment**() contains three main steps: (i) find a *feasible* controller, see Definition 1 below, from set C^t for each *candidate* l-switch upgrade, (ii) deploy a new controller if Step (i) fails to find a *feasible* controller, (iii) upgrade the switch and associate it with the *feasible* controller found in Step (i) or (ii). Note that Step (ii) assumes there is sufficient budget to upgrade one l-switch and deploy one controller. Further, in Step (ii), if the deployed controller is co-located with an s-switch u, the function disassociates u from its previous controller. Finally, if Step (ii) fails, the function considers the next candidate l-switch.

Definition 1: A controller a is a *feasible* controller for an s-switch v if (i) it has sufficient capacity to manage s-switch v, i.e., $\sum_{u \in A^t_a} \hat{\omega}^T_u + \hat{\omega}^T_v \leq \theta$, and (ii) there is at least a *feasible* path, see Definition 2, from v to a and from a to v.

Definition 2: A path $\mathcal{P}_{d,i} \in \mathcal{P}_d$ for s-c traffic demand d between switch v and controller a is *feasible* if adding traffic volume ω^T_d to the current total volume f^T_{uv} of each link $(u, v) \in \mathcal{P}_{d,i}$ does not exceed its MLU, i.e., $f^T_{uv} + \omega^T_d \leq c_{uv} \times U_{\max}$.

Note that Definition 1 uses the $\hat{\omega}^T_v$ value of each switch v to ensure that the association between s-switch v and controller a is feasible at each stage $t \leq T$. A controller is always feasible for its co-located switch. Further, we use f^T_{uv} and ω^T_d in Definition 2 to ensure that each *feasible* path can carry traffic demands at any stage $t \leq T$. Finally, if there are two *feasible* paths $\mathcal{P}_{d,i}, \mathcal{P}_{d,j} \in \mathcal{P}_d$ that are link-disjoint, they must be used to route the s-c demand d. Thus, set R^t_d may contain only one *feasible* path or two-link-disjoint *feasible* paths. Hence, we call set R^t_d as a set of *feasible* paths to route s-c demand d at stage t. Algorithm 2 shows the details of the function **Deployment**().

Algorithm 2 Deployment()

Input: $V^{t-1}, C^{t-1}, A^{t-1}, D^{t-1}, R^{t-1}, B^t$
Output: $V^t, C^t, A^t, D^t, R^t. \Delta B^t, L$

1: $V^t = V^{t-1}$, $C^t = C^{t-1}$, and $A^t = A^{t-1}$
2: $D^t = D^{t-1}$, $R^t = R^{t-1}$, and $X = V - V^t$
3: **for** (each switch $v \in X$ that has $p^t_v \leq B^t$, $w_v > 0$, and $\max_{v \in X}\{w_v/p^t_v\}$) **do**
4: **if** (**SelectC**($\mathcal{S}(v)$) finds a *feasible* controller $a \in C^t$) **then**
5: $\{V^t, X, \Delta B^t, L\}$ = **UpgradeS**($\mathcal{S}(v)$)
6: $\{A^t, D^t, R^t\}$ = **Associate**($\mathcal{S}(v), \mathcal{C}(a)$)
7: **else if** (**LocateC**(B^t, v) returns a 2-tuple (a, u)) **then**
8: $\{C^t, \Delta B^t\}$ = **DeployC**($\mathcal{C}(a)$).
9: **if** (node a contains an s-switch) **then**
10: Find $x \in C^t$ for $a \in A^t_x$
11: $\{A^t, D^t, R^t\}$ = **Disassociate**($\mathcal{S}(a), \mathcal{C}(x)$)
12: $\{A^t, D^t, R^t\}$ = **Associate**($\mathcal{S}(a), \mathcal{C}(a)$)
13: **end if**
14: $\{V^t, X, \Delta B^t, L\}$ = **UpgradeS**($\mathcal{S}(u)$)
15: $\{A^t, D^t, R^t\}$ = **Associate**($\mathcal{S}(u), \mathcal{C}(a)$)
16: **else** $X = X - v$
17: **end if**
18: **end for**
19: $n^T_{uv} = \lceil f^T_{uv}/(\gamma \times U_{\max}) \rceil$ for each $(u, v) \in E$

First, Line 1 of Algorithm-2 initializes V^t, C^t, and A^t with the set of s-switches, controllers, and associations from stage $t - 1$, respectively. Line 2 initializes sets D^t and R^t with all traffic demands and the paths used to route them in stage $t - 1$. Note that V^0, C^0, A^0, and D^0_{sc} are empty sets. Recall that D^0 contains only s-s traffic demands. Line 2 also uses set X to store all l-switches that yet to be upgraded.

Next, Line 3 of Algorithm-2 selects one *candidate* l-switch v to be upgraded at stage t, using the following selection criteria: (i) its upgrade cost is within budget, i.e., $p_v^t \leq B^t$, (ii) it has weight $w_v > 0$, and (iii) it has the largest unused cables per cost unit, i.e., the largest w_v/p_v^t. For each selected candidate l-switch v, Line 4 uses **SelectC**($\mathcal{S}(v)$) to find a *feasible* controller $a \in C^t$ that can be associated with switch v. Specifically, **SelectC**($\mathcal{S}(v)$) prioritizes a *feasible* controller a with two-link-disjoint *feasible* paths between switch v and controller a. The function **Deployment**() considers whether **SelectC**() finds a *feasible* controller.

If **SelectC**() finds a *feasible* controller $a \in C^t$ for switch v, Line 5 of Algorithm 2 first uses **UpgradeS**($\mathcal{S}(v)$) to upgrade switch v. The function performs the following five steps: (i) add v to set V^t, (ii) remove v from set X, (iii) reduce budget B^t by the upgrade cost of switch v, i.e., $B^t = B^t - p_v^t$, (iv) reduce the number of *off*-cables w_u by the number of *off*-cables that are adjacent to u and v, i.e., $w_u = w_u - (b_{uv} - n_{uv}^T)$, for each l-switch neighbor $u \in V - V^t$ of v, and (v) put every c-link (u, v) with non-zero traffic flow in a set L. Line 6 then uses **Associate**($\mathcal{S}(v), \mathcal{C}(a)$) to associate the upgraded switch v to its *feasible* controller $a \in C^t$. The function performs the following five tasks: (i) add s-switch v into set A_a^t, (ii) generate two s-c demands d and $d+1$, i.e., (v, a, ω_d^t) and (a, v, ω_{d+1}^t), respectively, and add them into set D^t, (iii) add their respective *feasible* paths R_d^t and R_{d+1}^t to set R^t, (iv) determine *active* path $R_{d,i}^t \in R_d^t$ and $R_{d+1,i}^t \in R_{d+1}^t$, and (v) increase traffic volume f_{uv}^T and f_{ij}^T of each link (u, v) in active path $R_{d,i}^t \in R_d^t$ and each link (i, j) in active path $R_{d+1,i}^t \in R_{d+1}^t$ by ω_d^T and ω_{d+1}^T, i.e., $f_{uv}^T = f_{uv}^T + \omega_d^T$ and $f_{ij}^T = f_{ij}^T + \omega_{d+1}^T$, respectively. In task (iv), each *active* path is the path that uses the smallest additional *on*-cables, i.e., $\min_{R_{j,i}^t \in R_j^t}\{\sum_{(u,v)\in R_{j,i}^t}(\lceil f_{uv}^T/(\gamma \times U_{\max})\rceil - \lceil (f_{uv}^T - \omega_j^T)/(\gamma \times U_{\max})\rceil)\}$ for $j \in \{d, d+1\}$ and $|R_j^t| = 2$. Further, task (v) uses ω_d^T and f_{uv}^T to ensure that each *active* path can carry s-c traffic demands at any stage $t \leq T$. Note that tasks (iv) and (v) are performed only if switch v and controller a are co-located at different nodes, i.e., $v \neq a$, $R_d^t \neq \phi$, and $R_{d+1}^t \neq \phi$.

If **SelectC**() fails to find a *feasible* controller, Line 7 of Algorithm-2 uses **LocateC**(B^t, v) to select one node in set $\{V - C^t\}$ to deploy a new controller. More specifically, **LocateC**() returns a 2-tuple (a, u), where a is the node at which the new controller is deployed and u is the node at which an l-switch is to be upgraded. To determine whether to place a controller at node a, it uses *one* of the following options, which it iterates through in order: (a) at node $a \neq v$ that has an s-switch, (b) at node $a = v$, or (c) at node $a \neq v$ that has an l-switch.

LocateC() returns (a, v), (v, v) or (a, a) for option (a), (b) or (c), respectively. Thus, for option (a) and (b), the available budget B^t must be sufficient to deploy one controller and an s-switch at node v, i.e., $B^t \geq \hat{p}^t + p_v^t$. For option (c), budget B^t is used to deploy an s-switch and a controller at node a, and thus, we have $B^t \geq \hat{p}^t + p_a^t$. It means, for option (c), the l-switch v is not upgraded; recall that each controller must be co-located with an s-switch. Except for option (b), **LocateC**() greedily selects a controller a that has the largest ratio w_a/p_a^t and the largest number of two-link-disjoint paths with all l-switches in $V - V^t$. In addition, for option (a), the function also selects a controller a that has two-link-disjoint paths with l-switch v. It selects a controller randomly for any remaining tie. **LocateC**() returns False when none of the three options (a)–(c) are feasible. In this case, Line 16 of **Deployment**() removes v from set X so that Line 3 considers the next candidate l-switch in X.

Next, if **LocateC**() successfully returns (a, v), (v, v) or (a, a) for options (a), (b), or (c), respectively, Lines 8–15 aim to complete the following tasks:

- **A 2-tuple** (a, v): (i) use the function **DeployC**($\mathcal{C}(a)$) in Line 8 to deploy controller a, (ii) use Line 10 to find s-switch a's previous *feasible* controller $x \in C^t$ where $a \in A_x^t$, (iii) use the function **Disassociate**($\mathcal{S}(a), \mathcal{C}(x)$) in Line 11 to dis-associate s-switch a from controller x, (iv) associate s-switch $\mathcal{S}(a)$ with controller $\mathcal{C}(a)$ using **Associate**($\mathcal{S}(a), \mathcal{C}(a)$) in Line 12, (v) upgrade l-switch v using **UpgradeS**($\mathcal{S}(v)$) in Line 14, and (vi) associate s-switch v with controller a using **Associate**($\mathcal{S}(v), \mathcal{C}(a)$) in Line 15.
- **A 2-tuple** (v, v): (i) use the function **DeployC**($\mathcal{C}(v)$) in line 8 to deploy controller v, (ii) use the function **UpgradeS**($\mathcal{S}(v)$) in Line 14 to upgrade l-switch v, and (iii) use the function **Associate**($\mathcal{S}(v), \mathcal{C}(v)$) in Line 15 to associate s-switch $\mathcal{S}(v)$ with controller $\mathcal{C}(v)$.
- **A 2-tuple** (a, a): (i) use the function **DeployC**($\mathcal{C}(a)$) in line 8 to deploy controller a, (ii) use the function **UpgradeS**($\mathcal{S}(a)$) in Line 14 to upgrade l-switch a, and (iii) use the function **Associate**($\mathcal{S}(a), \mathcal{C}(a)$) in Line 15 to associate s-switch $\mathcal{S}(a)$ with controller $\mathcal{C}(a)$.

The details of **DeployC**() and **Disassociate**() are as follows. **DeployC**($\mathcal{C}(a)$) performs the following two steps: (i) add controller a into set C^t, and (ii) reduce budget B^t by deployment cost $\hat{p}^t$. **Disassociate**($\mathcal{S}(a), \mathcal{C}(x)$) carries out the following three tasks: (i) remove any s-c demand d generated for s-switch a's previous association with controller x from set D^t, (ii) decrease the total volume f_{uv}^T of every link (u, v) in active path $R_{d,i}^t \in R_d^t$ by the demand's volume ω_d^T, and (iii) remove switch a from set A_x^t.

Lines 3–18 are repeated until (i) all switches in X have been upgraded in Line 5 or Line 14 or removed in Line 16, i.e., $X = \phi$, or (ii) each remaining switch $v \in X$ has cost larger than the remaining budget B^t, i.e., $p_v^t > \Delta B^t$, or (iii) there is no unused cables to turn-off, i.e., $w_v = 0$. Line 19 computes the number of *on*-cables n_{uv}^T according to the current volume f_{uv}^T for each link (u, v). Line 10 of Algorithm 1 then adds the budget ΔB^t to B^{t+1}.

3) TRAFFIC REROUTING

Line 11 of HGMSU-SC uses the function **GTE-SC**(V^t, R^t, L) to increase the number of unused cables that can be powered off by s-switches. We adopt traffic rerouting algorithm

proposed in our prior work in [23] for function **GTE-SC**(). Briefly, the algorithm performs four main steps:

(a) Find a c-link $(u, v) \in L$ with a cable that has the smallest used capacity $r_{uv} = (f^T_{uv} - \gamma \times U_{\max} \times \lfloor f^T_{uv}/\gamma \times U_{\max} \rfloor)$.
(b) Store each path $R^t_{d,i} \in R^t_d$ that passes link (u, v) into set Q_{uv} and switch off one of the link's *on*-cables.
(c) Find *routable path* $\mathcal{P}_{d,j}$ in set $\{\mathcal{P}_d - R^t_{d,i}\}$. Briefly, path $\mathcal{P}_{d,j}$ is called *routable* if each link (u, v) in the path has sufficient capacity to carry an additional traffic volume of ω^T_d, i.e., $(f^T_{uv} + \omega^T_d) \leq (\gamma \times n^T_{uv} \times U_{\max})$.
(d) Recalculate the *off*-cables w_u for each l-switch u in set $\{V - V^t\}$.

Note that Step (b) and (c) are repeated for each c-link $(u, v) \in L$. Further, Step (c) is repeated for each path $R^t_{d,i} \in Q_{uv}$ or until all paths that use the powered-off cable in c-link (u, v) are rerouted onto their alternative path, i.e., $r_{uv} = 0$.

The function **GTE-SC**() extends Step (c) to maintain the existing two-link-disjoint paths for any s-c traffic demand d, i.e., $|R^t_d| = 2$ paths. Here, Step (c) considers the following three cases when finding the *routable* path $\mathcal{P}_{d,j}$: (i) path $\mathcal{P}_{d,j}$ is the backup path in set R^t_d, (ii) path $\mathcal{P}_{d,j}$ is link-disjoint with the two paths in set R^t_d, i.e., $\mathcal{P}_{d,j} \notin R^t_d$, or (iii) path $\mathcal{P}_{d,j}$ is not in set R^t_d and $|R^t_d| = 1$ path. For case (i), the extended Step (c) sets path $R^t_{d,i}$ as backup while path $\mathcal{P}_{d,j}$ as *active*. On the other hand, for cases (ii) and (iii), it replaces path $R^t_{d,i}$ with path $\mathcal{P}_{d,j}$. For any case, the updated Step (c) decreases and increases the total volume f^T_{uv} for each link (u, v) in path $R^t_{d,i}$ and $\mathcal{P}_{d,j}$, respectively, by ω^T_d. Line 12 of Algorithm 1 computes the energy saving ε^t using Eq. (1). Finally, Line 13 computes the average energy saving over T stages.

B. ALGORITHM ANALYSIS

We conclude this section with the run-time and space complexity analysis of HGMSU-SC presented in in Proposition 1 and Proposition 2, respectively.

Proposition 1: The time complexity of HGMSU-SC is $O(|V|^2|E|^2)$.

Proof: The routing initialization in Lines 1–7 of Algorithm 1 has a worst case time complexity of $O(k|D^0_{ss}||V|(|E|+|V|log|V|))$. Here, Line 2 of Algorithm 1 uses Yen's algorithm [35] that has run-time complexity $O(k|D^0_{ss}||V|(|E| + |V|log|V|))$. Lines 3 and 4 require $O(1)$ and $O(|D^0_{ss}||E|)$, respectively. Both Lines 6 and 7 have a worst case time complexity of $O(|E|)$. The function **Deployment**() in Line 9 of Algorithm 1 requires $O(k|V|^2|E|)$ because (i) the initialization of all sets in Lines 1–2 of Algorithm 2 needs $O(|D^T|)$, (ii) functions **SelectC**(), **UpgradeS**(), **Associate**(), **LocateC**(), and **DeployC**() require $O(k|V||E|)$, $O(|V||E|)$, $O(k|V||E|)$, $O(k|V|^2|E|)$, and $O(1)$, respectively, (iii) Line 10 of Algorithm 2 takes $O(|V|)$, (iv) function **Disassociate**() needs $O(|D^T_{sc}| + |E|)$, and (v) Line 19 takes $O(|E|)$. Line 10 of Algorithm 1 requires $O(1)$. The function **GTE-SC**() in Line 11 of Algorithm 1 takes $O(k|D^T||E|^2)$. More specifically, Step (a) of **GTE-SC**() needs $O(E)$ to find a c-link with the smallest used capacity. Step (b) requires $O(|D^T||E|)$ to check every path in set R^t that uses the c-link. Step (c) takes $O(k|E|)$ to find a *routable* path among the generated k shortest path for each demand. Step (b) is repeated $O(|E|)$ times and thus, it requires $O(|D^T||E|^2)$. On the other hand, Step (c) is repeated $O(|E|)$ times, each of which is repeated $O(|D^T|)$ times. Thus, Step (c) takes $O(k|D^T||E|^2)$. Step (d) requires $O(|V|)$. Line 12 of Algorithm 1 needs $O(|E|)$ while Line 13 requires $O(1)$. Repeating Lines 8–14 of Algorithm 1 in T times, Lines 9–13 require $O(Tk|V|^2|E| + Tk|D^T||E|^2)$. Thus, HGMSU-SC has the worst case run-time of $O(k|D^0_{ss}||V|(|E|+|V|log|V|)+Tk|V|^2|E|+Tk|D^T||E|^2) = O(Tk|D^T||E|^2) = O(|V|^2|E|^2)$. Note that we consider $|D^0_{ss}| \leq |V|^2$, $|D^T_{sc}| \leq |V|^2$, $|D^T| \leq 2|V|^2$, $|E| \leq |V|^2$, and parameters k and T are constants. □

Proposition 2: The space complexity of HGMSU-SC is $O(V|^2|E|)$.

Proof: There are 13 input variables for HGMSU-SC based on the algorithms presented in Section IV-A1, IV-A2, and IV-A3: (i) parameters k, T, B, $U_{\max}$, ρ, $\hat{\rho}$, σ, and ε_T require constant space, i.e., $O(1)$, (ii) network $G^0(V, E)$ with $|V|$ nodes and $|E|$ links takes $O(|V| + |E|)$, (iii) the initial cost parameters p^0_v and $\hat{p}^0_v$ and their cost at each stage t needs $O(|V|)$, and (iv) the set of initial demands D^0_{ss} requires $O(|D^0_{ss}|) = O(|V|^2)$ as $|D^0_{ss}| \leq |V|^2$. Thus, all input variables of HGMSU-SC have a space requirement of $O(1) + O(|V| + |E|) + O(T|V|) + O(|V|^2) = O(|E| + |V|^2) = O(|V|^2)$ because $|E| \leq |V|^2$. For its outputs, HGMSU has 16 types of variables: (i) variable types B^t, ΔB^t, and ε^t, as used in Algorithm 1 and Algorithm 2, require $O(T)$ space, (ii) the set V^t, C^t, X, and A^t, as used in Algorithm 1 and Algorithm 2, for all stages $t \in \{1, 2, \ldots, T\}$ have a space requirement of $O(|V|)$, (iii) the set D^t used by Algorithm 1 and Algorithm 2 for all stages $t \in \{1, 2, \ldots, T\}$ takes $O(T(|D^t_{ss}| + |D^t_{sc}|)) = O(T|V|^2)$ because $|D^t_{ss}| = |D^0_{ss}| \leq |V|^2$ and $|D^t_{sc}| \leq |V|^2$, (iv) there are $|E|$ variables of type n^t_{uv} and f^t_{uv} that are used in Algorithm 1 and function **GTE-SC**() in Line 11 of Algorithm 1. Thus, their space requirement is $O(T|E|)$, (v) the set r_{uv} and Q_{uv} that are used by **GTE-SC**() occupy $O(|E|)$ space, (vi) the variable type w_v that is used in Algorithm 1 and Algorithm 2 is for all $|V|$ nodes and thus, its space complexity is $O(|V|)$, (vii) the set L used in Algorithm 1, Algorithm 2, and **GTE-SC**() is for all links and hence, we have $O(|E|)$, (viii) the set $\mathcal{P}_d$ of Algorithm 1, Algorithm 2, and **GTE-SC**() needs $O(|D^0||E|) = O(|V|^2|E|)$ space, and (ix) the set R^t, which is used in Algorithm 1, Algorithm 2, and **GTE-SC**(), for all stages $t \in \{1, 2, \ldots, T\}$ consume $O(T|D^t||E|) = O(T|V|^2|E|)$. Thus, the space requirement of all output variables for HGMSU-SC occupy $O(T + |V| + T|V|^2 + T|E| + |E| + |V| + |E| + |V^2||E| + T|V|^2|E|) = O(|V|^2|E|)$ because T and k are constants. Overall, the space complexity of HGMSU-SC is $O(|V|^2 + |V|^2|E|) = O(|V|^2|E|)$. □

V. EVALUATION

We have implemented HGMSU-SC in C++ and used Gurobi [36] to solve MIP-SC. All of our experiments are conducted on a 64-bit Windows machine with an

Intel-core-i7 CPU @3.60 GHz and 16 GB of memory. Further, we use five actual network topologies, which are also used in [23] and [24]; see Table 3. For Abilene and GÉANT, we use their actual traffic matrices. For DFN, ITC Deltacom and TATA, we use the gravity model [37] to generate traffic flows as there are no public traffic matrices.

Each experiment uses the following parameter values. Each link has a bundle size of $b_{uv} = 4$ cables. The cable capacity is $\gamma = 2.5$ Gbps, and links have a Maximum Link Utilization (MLU) threshold of $U_{max} = 80\%$. We use an equal depreciation cost rate of $\rho = \hat{\rho} = 40\%$ for each switch and controller. Further, each path uses a delay multiplier $\sigma = 1.1$. Thus the maximum delay of each path for demand d is $\delta_{\max,d} = \lceil 1.1 \times \delta_{\min,d} \rceil$. This means each path can have a delay of up to 10% longer than its corresponding shortest path delay $\delta_{\min,d}$. The traffic of each s-s demand d increases at a rate of $\mu_d = 22\%$. For each switch, its initial upgrade *cost* and *cpps*, is randomly assigned from one of the following (*cost*, *cpps*) values: ($50K, 100K), ($100K, 200K), and ($150K, 300K). More specifically, for each switch, we draw a random number from the Normal distribution $\mathcal{N}(2, 0.5)$ and round the number to the nearest integer. A switch v that draws a value of one, two or three is assigned ($p_v^0 =$ $50K, $\hat{\omega}_v^1 =$ 100K *cpps*), ($100K, 200K) or ($150K, 300K), respectively. Each switch's *cpps* increases at a rate of $\hat{\mu}_v = 22\%$ per stage. Control packets have size $\beta =$ 160 bytes [28]. For each controller, we set its capacity θ to 7800K *cpps* and use an initial deployment cost of $\hat{p}^0 =$ $25K. Following [28], the capacity $\theta = 7800$K *cpps* is calculated from the controller access bandwidth of 10 Gbps and control packet size $\beta = 160$ bytes, i.e., $\theta = 10$ Gbps/(160 bytes $\times$ 8 bits). Similar to the work in [20], the cost to deploy a controller is cheaper than the upgrade cost of an s-switch.

The following sections are organized as follows. First, Section V-A evaluates the scalability of MIP-SC and HGMSU-SC solutions in terms of their running time in CPU seconds. Section V-B and Section V-C then analyze the effect of increasing budgets on energy saving and the total number of deployed s-switches and controllers over the T stages. Next, Section V-D studies the impact of controller placement on network performances, e.g., energy saving and path delay. Finally, Section V-E compares the performances of HGMSU-SC against an existing technique, i.e., the Improved Genetic Controller Placement Algorithm (IGCPA) [11].

A. RUNNING TIME

This experiment studies the run time performance of MIP-SC and HGMSU-SC. It uses a budget of $B = 1.2$M and $T = 3$ stages. As shown in Table 3, the run time of MIP-SC is significantly longer than HGMSU-SC for the tested five networks. MIP-SC requires 14.46, 27542.88, 73670.16, and 111293.31 seconds, while HGMSU-SC takes only 0.17, 1.28, 21.05, and 256.34 seconds to produce the results for Abilene, GÉANT, DFN, and Deltacom, respectively. The results show that on average 80.44% of HGMSU-SC's running time is used for generating k alternative paths using Yen's algorithm [35]. Further, MIP-SC failed to produce results for TATA because the optimizer ran out of memory. Thus, the performance comparisons between MIP-SC and HGMSU-SC in the following sections use only Abilene, GÉANT, DFN, and Deltacom.

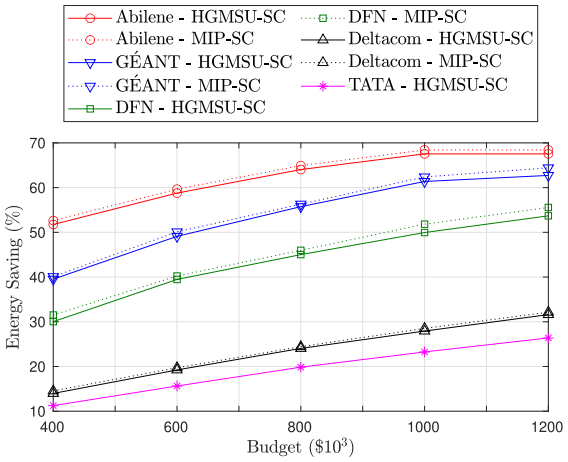


FIGURE 4. Energy saving ε_T of MIP-SC and HGMSU-SC for various budget B values.

Note that this experiment does not consider different values of budget B, number of stages T, and the other parameters, e.g., the initial upgrade cost of each switch p_v^0 and controller $\hat{p}^0$, and their depreciation rates ρ and $\hat{\rho}$. The reason is because these factors will not significantly affect the running time of HGMSU-SC that mainly depends on the number of links $|E|$ and nodes $|V|$; see Proposition 1. Similarly, the running time of MIP-SC is affected mainly by the values of $|E|$ and $|V|$; see Section III-B. Thus, different values of other parameters will not significantly change the running time of MIP-SC.

B. IMPACT OF BUDGET

An operator's budget has an impact on the number of upgraded switches, and the numbers of deployed controllers. To study this relationship, we consider a budget value of $B \in$ {$400K, $600K, $800K, $1M, $1.2M}, and $T = 3$ stages. Referring to Fig. 4, we see that MIP-SC and HGMSU-SC produce more energy saving ε_T for larger budget values. The energy saving of HGMSU-SC for Abilene, GÉANT, DFN, and Deltacom is only up to 1.67%, 2.62%, 4.71%, and 3.91% off from the optimal energy saving computed by MIP-SC. For example, using a budget of $B =$ $400K, MIP-SC and HGMSU-SC respectively produce energy saving ε_T of 52.63% and 51.75% for Abilene. Their energy saving increases to 68.42% and 67.54% when using a larger budget of $B =$ $1.2M. The increasing energy saving in Fig. 4 is reasonable because a large budget allows MIP-SC and HGMSU-SC to upgrade more l-switches.

Let $\mathcal{V}^T$ denote the percentage of l-switches that have been upgraded into s-switches over T stages, i.e., $\mathcal{V}^T = |V^T|/|V| \times 100\%$. Fig. 5 shows that a larger budget B value allows MIP-SC and HGMSU-SC to upgrade more switches.

TABLE 3. Running time of MIP-SC and HGMSU-SC.

Name	$\|V\|$	$\|E\|$	$\|D_{ss}^0\|$	Running Time (CPU seconds)	
				MIP-SC	HGMSU-SC
Abilene	12	30	132	14.46	0.17
GÉANT	23	74	466	27542.88	1.28
DFN	58	174	3306	73670.16	21.047
Deltacom	113	322	12656	111293.31	256.34
TATA	145	372	20880	N/A	413.27

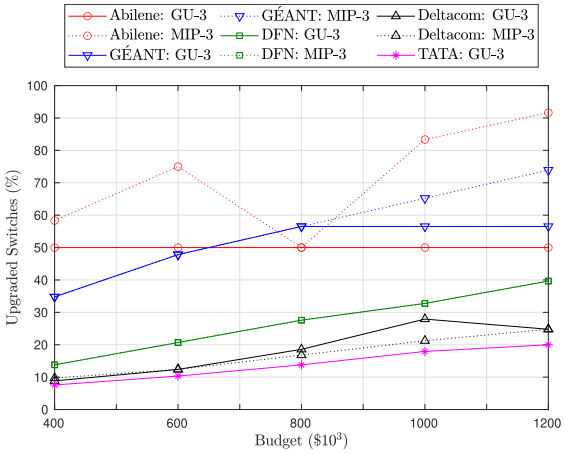


FIGURE 5. The percentage of upgraded *l*-switches $\mathcal{V}^T$ for MIP-SC and HGMSU-SC given various budget B values.

As an example, for Abilene, GÉANT, DFN, and Deltacom, MIP-SC increases the percentage $\mathcal{V}^T$ from 58% to 91.67%, 34.78% to 73.91%, 13.79% to 39.66%, and 9.74% to 24.78%, respectively when the budget B is increased from \$400K to \$1.2M. However, for Abilene, when the budget B increases from \$600K to \$800K, MIP-SC decreases the percentage $\mathcal{V}^T$ from 75% to 50%. Recall that MIP-SC and HGMSU-SC aim to maximize energy saving ε_T, and thus larger budget B value *does not always* increase the number of upgraded switches. One possible reason is because maximizing ε_T may require the algorithms to upgrade more *l*-switches at earlier stages. In this case, there is less remaining budget to upgrade switches in the later stages. Thus, in total, there are less number of switches that can be upgraded. Recall that the switch upgrade cost at later stages is cheaper than the cost at earlier stages.

As shown in Fig. 5, HGMSU-SC upgrades similar number of switches as compared to MIP-SC for DFN and Deltacom. Further, HGMSU-SC does not upgrade more switches for Abilene and GÉANT even when it has remaining budget. The percentage $\mathcal{V}^T$ for both networks remains constant at 50% and 56.52% when budget B is increased from \$400K and \$800K to \$1.2M, respectively. The reason is because HGMSU-SC will upgrade a switch only if it can turn off unused cables. For both networks, the respective budget B value \$400K and \$800K is sufficient to upgrade all *l*-switches that have unused cables.

Fig. 6 shows that larger budget allows MIP-SC to deploy more controllers. As an example, for Abilene and GÉANT with budget B = \$1.2M, MIP-SC deploys up to $|C^T|$ = 11 controllers for 11 *s*-switches and $|C^T|$ = 13 for 17 *s*-switches, respectively. Thus, 100% and 56.52% of the *s*-switches in Abilene and GÉANT respectively are co-located with a controller. This helps minimize the number of cables used to route *s-c* traffic demands. Moreover, MIP-SC is able to deploy more controllers for both networks because controller deployment cost is low. To show the case when the controller deployment cost is expensive, we rerun MIP-SC for Abilene and GÉANT using B = \$1.2M and set the controller deployment cost to $\hat{p}^0$ = \$300K. We find that the number of controllers $|C^T|$ produced by MIP-SC decreases from 11 to five controllers and from 13 to two controllers for Abilene and GÉANT, respectively. The respective energy saving of Abilene and GÉANT also reduces from 68.42% to 59.65% and 64.41% to 53.94%. The decreasing energy saving for Abilene and GÉANT is inline with the decrease in their respective number of upgraded switches. Specifically, the percentage of upgraded switches $\mathcal{V}^T$ of Abilene and GÉANT decreases from 91.67% to 50% and 73.91% to 60.87%, respectively. These results are reasonable because MIP-SC has to reduce the number of upgraded switches in order to deploy five and two controllers for Abilene and GÉANT, respectively. For DFN and Deltacom, a budget of B = \$1.2M is sufficient for MIP-SC to deploy only up to three and two controllers, respectively. This is because there are a number of *l*-switches with unused cables that are yet to be upgraded by MIP-SC for both networks.

On the other hand, Fig. 6 shows that HGMSU-SC consistently deploys one controller when the budget is not larger than B = \$1.2M. The reason is because the controller has sufficient capacity to manage all *s*-switches. As an example, budget B = \$1.2M can be used by HGMSU-SC to upgrade $\mathcal{V}^T$ = 50% and $\mathcal{V}^T$ = 20% of *l*-switches for Abilene and TATA, respectively. Our experiment shows that 20.99% and 82.05% of the controller capacity θ = 7800K is used to handle the traffic load (in *cpps*) from *s*-switches in Abilene and TATA, respectively. Recall that HGMSU-SC deploys a controller only if there is no *feasible* controller as per Definition 1 for any of the upgraded *l*-switches. To show the effect of smaller capacity θ = 1100K on the number of deployed controllers $|C^T|$, we rerun HGMSU-SC for Abilene

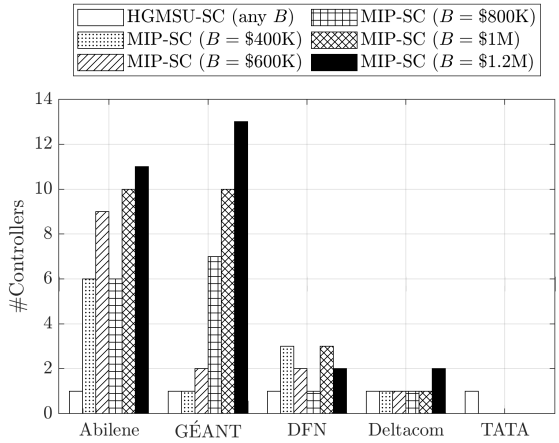


FIGURE 6. Total number of deployed controllers $|C^T|$ for MIP-SC and HGMSU-SC for various budget B values.

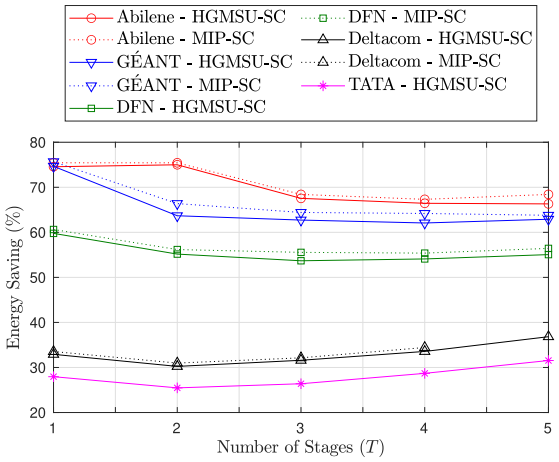


FIGURE 7. Energy saving ε_T of MIP-SC and HGMSU-SC for various stage T values.

and TATA using $B = \$1.2\text{M}$. We find that HGMSU-SC deploys more controllers, which are $|C^T| = 2$ and $|C^T| = 6$ controllers for Abilene and TATA, respectively. The energy saving and the number of upgraded switches for Abilene remains the same. The reason is because a budget of $B = \$1.2\text{M}$ is sufficient to upgrade 50% of Abilene's l-switches as well as deploying its two controllers. However, the energy saving ε_T for TATA decreases slightly from 26.39% to 25.49%, and the percentage of upgraded l-switches $\mathcal{V}^T$ also decreases from 20% to 18.62%. The results for TATA are reasonable because for the given budget, HGMSU-SC must reduce the number of upgraded l-switches in order to deploy its six controllers.

C. EFFECT OF INCREASING STAGES

This experiment investigates the impact of using a longer planning horizon T on the energy saving and the number of deployed switches and controllers. We use a budget of $B = \$1.2\text{M}$ and vary T from one to five. Note that MIP-SC fails to obtain a result for Deltacom when there are $T = 5$ stages. As shown in Fig. 7 the energy saving ε_T for Abilene, GÉANT, and DFN decreases for larger T values. For these three networks, a budget of $B = \$1.2\text{M}$ can be used to upgrade a larger percentage of l-switches at earlier stages. Further, as the later stages have a higher traffic volume, smaller number of remaining l-switches in later stages have fewer unused cables that can be turned off. This means upgrading these switches does not significantly increase ε_T. For example, as shown in Fig. 7, the energy saving ε_T produced by MIP-SC and HGMSU-SC for Abilene decreases from 75.44% to 68.42% and 74.56% to 66.32%, respectively. For Deltacom and TATA, however, there are more s-switches to upgrade in later stages. The energy saving ε_T produced by MIP-SC and HGMSU-SC for Deltacom increases from 33.54% to 34.43% and 32.92% to 33.56%, respectively, when T increases from one to four. Similarly, the saving ε_T produced by HGMSU-SC for TATA increases from 27.96% to 31.55% for $T = 1$ to $T = 5$. Fig. 7 further shows that for Abilene, GÉANT, DFN, and Deltacom, HGMSU-SC produces an energy saving ε_T that is up to 3.08%, 4.07%, 3.36%, and 2.38%, respectively, off from the result produced by MIP-SC.

Fig. 8 shows the percentage $\mathcal{V}^T$ of upgraded l-switches for different number of stages T. MIP-SC does not always upgrade more switches for larger T values. Recall that the goal of MIP-SC is not to maximize the percentage $\mathcal{V}^T$ but to maximize energy saving ε_T. In contrast, the percentage $\mathcal{V}^T$ computed by HGMSU-SC never decreases with increasing stages T. As an example, for Abilene, HGMSU-SC upgrades the same number of l-switches, i.e., $\mathcal{V}^T = 50\%$ of switches, when the value of T increases from one to five. The reason is because budget $B = \$1.2\text{M}$ is sufficient for HGMSU-SC to upgrade all l-switches that are adjacent to each unused cable in a single stage. For the same budget, however, MIP-SC and HGMSU-SC require more than one stage to upgrade all l-switches with unused cables for GÉANT, DFN, and Deltacom. Thus, the percentage $\mathcal{V}^T$ increases for larger T values. For example, MIP-SC and HGMSU-SC increase the percentage $\mathcal{V}^T$ of Deltacom from 13.27% to 31.86% and 13.27% to 30.97%, respectively, when T increases from one to four.

In terms of the number of controllers $|C^T|$, larger number of s-switches makes MIP-SC and HGMSU-SC deploy more controllers. As an example, for DFN, MIP-SC deploys $|C^T| = 2$ controllers, which increases significantly to $|C^T| = 26$ controllers when T increases from one to five. The reason is because the cost to deploy a controller is cheaper at later stages. Moreover, a budget of $B = \$1.2\text{M}$ is sufficient to upgrade all l-switches, which helps turn off unused cables at earlier stages. The number of controllers deployed by HGMSU-SC for DFN, however, increases marginally from $|C^T| = 1$ controller to only $|C^T| = 2$ controllers. This is because HGMSU-SC deploys a new controller only if no existing controllers satisfy Definition 1. For Deltacom and TATA, however, there are more l-switches to upgrade in later stages. MIP-SC and HGMSU-SC produce the same

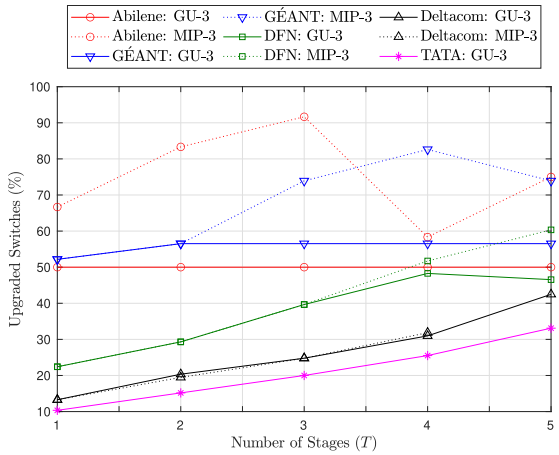


FIGURE 8. The percentage of upgraded *l*-switches $\mathcal{V}^T$ for MIP-SC and HGMSU-SC for various stage T values.

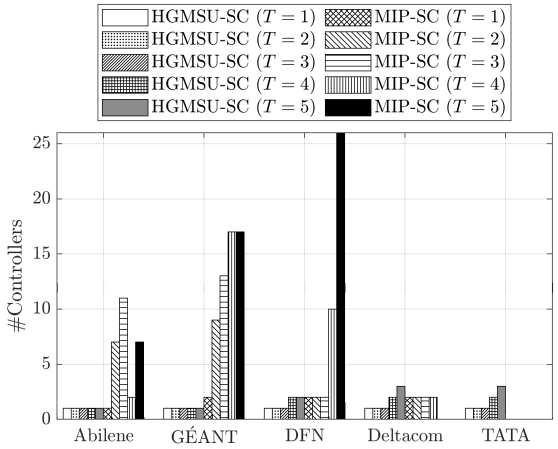


FIGURE 9. Total deployed controllers $|C^T|$ for MIP-SC and HGMSU-SC given various stage T values.

number of controllers $|C^T| = 2$ for Deltacom using value $T = 4$. For value $T = 5$, HGMSU-SC produces $|C^T| = 3$ controllers. For TATA, HGMSU-SC deploys $|C^T| = 1$ to $|C^T| = 3$ controllers for an increasing number of stages from value $T = 1$ to $T = 5$.

D. EFFECT OF CONTROLLER PLACEMENT

This section aims to evaluate the impact of using HGMSU-SC, where it deploys controllers at strategic locations, on 1) energy saving, 2) number of *s-c* traffic demands with two link-disjoint paths, 3) path delay of *s-s* and *s-c* traffic, and 4) number of deployed *s*-switches and controllers.

We compare HGMSU-SC against HGMSU-SC^a - a version of HGMSU-SC that uses *arbitrary* controller placement, i.e., deploy controllers and associates each *s*-switch to a controller arbitrarily. More specifically, we modify Algorithm 2 for HGMSU-SC^a such that (i) function **SelectC()** selects any existing *feasible* controller arbitrarily, and (ii) function **LocateC()** deploys a new controller at any arbitrary node. We use a budget value $B \in$ {\$400K, \$1.2M}, planning

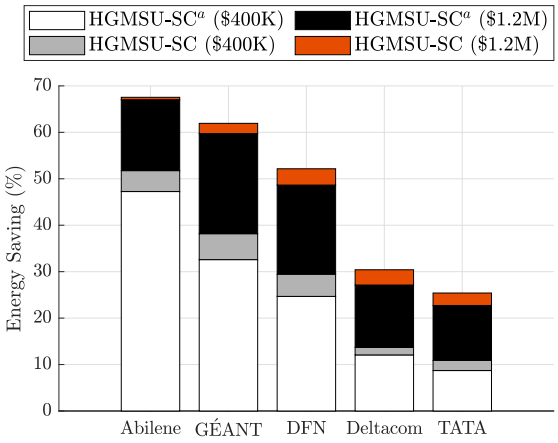


FIGURE 10. Energy saving of HGMSU-SC and HGMSU-SC^a.

horizon with $T = 3$ stages, and set the controller capacity to $\theta = 1100$K *cpps*. For each network and budget, the results for HGMSU-SC^a are averaged over ten runs with a maximum standard deviation of 3.64. Note that to save space, except for energy saving, we do not present other results, i.e., the number of *s-c* traffic demands with two link-disjoint paths, the path delay, and the number of deployed *s*-switches and controllers, in figures.

1) ENERGY SAVING

Fig. 10 shows that using selective controller placement produces higher energy saving than using arbitrary controller placement. More specifically, HGMSU-SC produces 8.7% and 0.78% higher energy saving than HGMSU-SC^a for Abilene with budget $B =$ \$400K and $B =$ \$1.2M, i.e., $\varepsilon_T = 51.75\%$ and $\varepsilon_T = 67.54\%$ for HGMSU-SC versus $\varepsilon_T = 47.25\%$ and $\varepsilon_T = 67.02\%$ for HGMSU-SC^a, respectively. Similarly, for budget $B =$ \$400K ($B =$ \$1.2M), HGMSU-SC produces 14.63% (3.56%), 16.23% (6.73%), 11.83% (10.81%), and 20.02% (10.57%) higher saving than HGMSU-SC^a for GÉANT, DFN, Deltacom and TATA, respectively. The reason is because the selective controller placement of HGMSU-SC aims to optimize the location of each deployed controller and the switch-controller association to maximize energy saving.

2) NUMBER OF *s-c* TRAFFIC THAT USES TWO LINK DISJOINT PATHS

Our simulation shows that HGMSU-SC is able to route more *s-c* traffic via two link-disjoint paths than HGMSU-SC^a. For example, for DFN and Deltacom with budget $B =$ \$1.2M, 19.05% (7.81%) and 7.14% (0.77%) of *s-c* traffic for HGMSU-SC (HGMSU-SC^a) are routed via two-link-disjoint paths, respectively. The reason is because HGMSU-SC always prioritizes an association that has two-link-disjoint paths between an *s*-switch and a controller. Furthermore, HGMSU-SC selects a node that not only has the largest number of unused cables per upgrade cost unit,

but also has the largest number of two-link-disjoint paths to l-switches to co-locate a new controller.

3) PATH DELAY OF *s-s* AND *s-c* TRAFFIC

Our simulation shows that HGMSU-SC and HGMSU-SC^a produce a similar increase in path delay. For example, on average, HGMSU-SC (HGMSU-SC^a) increases the path delay for GÉANT and Deltacom by 1.3% (1.11%) and 0.11% (0.13%), respectively, when using budget B = \$400K. We find similar results for budget B = \$1.2M.

4) NUMBER OF DEPLOYED *s*-SWITCHES AND CONTROLLERS

HMGSU-SC and HGMSU-SC^a deploy similar number of s-switches and controllers. More specifically, for budget value B = \$400K, except for TATA where HGMSU-SC upgrades one switch more than HGMSU-SC^a, both solutions upgrade the same number of switches for each other network. Further, they deploy the same number of controllers $|C^T|$ = 2 for all networks. For budget value B = \$1.2M, HGMSU-SC upgrades one switch more than HMGSU-SC^a for Deltacom and TATA, and one switch less for each other network. In terms of controllers, HMGSU-SC^a deploys one controller more than HMGSU-SC for GÉANT. Both solutions deploy the same number of controllers for each other network. The simulation results are reasonable because the aim of our selective controller placement is not to minimize the number of deployed s-switches nor controllers.

In summary, our simulation shows that HGMSU-SC as compared to HMGSU-SC^a (i) produces higher energy savings and route more control traffic demands via link-disjoint paths, and (ii) marginally affects the number of deployed switches and controllers, and path delays. It is worth noting that HGMSU-SC spends up to 7.24% less than HMGSU-SC^a. Further, on average, for HGMSU-SC, a controller has up to 12% lower load (in cpps) than HMGSU-SC^a. Finally, both solutions have a similar running time.

E. HGMSU-SC VERSUS IGCPA

This section compares HGMSU-SC against IGCPA [11]. Briefly, IGCPA aims to strategically deploy a set of controllers of a *pure* SDN in T = 1 planning stage. Its goal is to maximize energy saving. IGCPA uses GreCo [10] to associate each s-switch to one of the deployed controllers, and to route network traffic. Following GreCo, in IGCPA, (i) each *s-c* traffic is routed via a single path, (ii) maximum path delay constraint is applied *only* for *s-c* traffic routing, and (iii) each link contains only one cable.

To ensure fair comparisons, we use the following setup: (i) T = 1 stage, MLU threshold $U_{\max}$ = 100%, controller capacity θ = 1100K *cpps*, maximum delay $\delta_{\max,d} = \lceil 1.1 \times \delta_{\min,d} \rceil$ for each *s-c* demand d, and bundle size b_{uv} = 4 cables for each link (u, v), (ii) the budget B in HGMSU-SC for each network is set sufficiently large to upgrade *all* switches and deploy controllers to manage them, and (iii) for each network, we use HGMSU-SC to find the number of controllers needed to manage *all* s-switches. Then, we deploy the same number of controllers for IGCPA, i.e., three, five, 11, 21, and 28 controllers for Abilene, GÉANT, DFN, Deltacom, and TATA, respectively. Note that we omit parameters such as the depreciation rate ρ and $\hat{\rho}$ of each switch and controller cost per stage and the increase rate μ_d of every traffic demand d per stage. This is because there is only T = 1 planning stage.

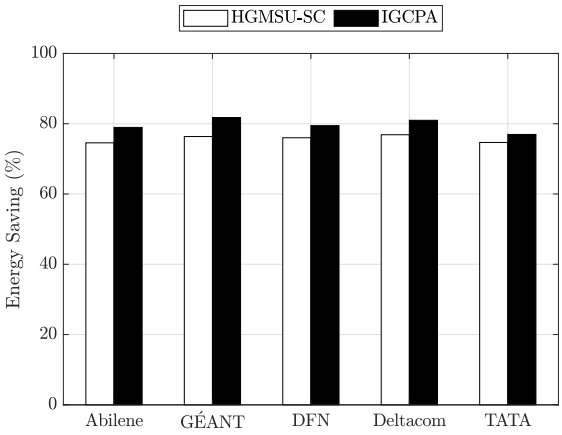


FIGURE 11. Energy saving of HGMSU-SC and IGCPA.

Fig. 11 shows that HGMSU-SC produces smaller energy saving than IGCPA. More specifically, HGMSU-SC produces energy saving ε_T of 74.56%, 76.35%, 76%, 76.86%, and 74.66% for Abilene, GÉANT, DFN, Deltacom, and TATA respectively. IGCPA, on the other hand, saves 5.55%, 6.61%, 4.34%, 5.08%, and 2.97% higher energy than HGMSU-SC for the respective five networks. These results are reasonable because each *s-s* traffic d in HGMSU-SC must be routed via a path with delay no longer than $\delta_{\max,d}$. In contrast, IGCPA does not consider the maximum delay constraint for *s-s* traffic. The reason is because GreCo [10] that is used in IGCPA does not have a maximum delay constraint on *s-s* traffic. Our simulation results show that using IGCPA, 29.17%, 54.06%, 57.13%, 30.2%, and 46.46% of *s-s* traffic demands for Abilene, GÉANT, DFN, Deltacom, and TATA, respectively, use paths with a delay longer than $\delta_{\max,d}$. Note that, for these five networks, these paths have a delay up to 98.28% longer than the delay of their original shortest path. In contrast, for the five networks, HGMSU-SC generates respectively *only* 6.06%, 9.23%, 10.5%, 6.95%, and 18.83% paths that are longer than their corresponding shortest path. More importantly, each path in HGMSU-SC is within its maximum delay requirement of $\delta_{\max,d}$. Our simulation shows that IGCPA requires a significantly higher CPU time to produce results as compared to HGMSU-SC. For example, IGCPA takes 32312.91 and 121092.48 seconds to produce results for Deltacom and TATA, respectively. In contrast, HGMSU-SC requires only 254.36 and 394.26 seconds for the two respective networks.

VI. CONCLUSION

This paper studies network planning solutions, namely MIP-SC and HGMSU-SC, which are used to upgrade a legacy network into an SDN over multiple stages. The computed solution aims to minimize the energy cost of an upgraded network by strategically upgrading *l*-switches and installing SDN controllers. The results show that (i) increasing an operator's budget and planning stages result in more upgraded *l*-switches leading to higher energy savings up to 68.42%, (ii) HGMSU-SC produces energy saving that is within 4.71% from optimal, (iii) MIP-SC deploys more controllers than HGMSU-SC in order to maximize energy saving, (iv) HGMSU-SC that uses a selective controller placement results in higher energy saving as compared to an arbitrary controller placement, and (v) HGMSU-SC runs significantly faster than an existing solution while producing competitive results in energy saving and ensuring each traffic is routed via a path within a given delay constraint. For future work, we plan to include (i) communication between controllers when performing controller placement and traffic routing for GMSU-SC, (ii) a scenario when traffic demands are *dynamic*, i.e., the traffic changes over time, and (iii) an alternative solution that use machine learning techniques such as in [38] and [39] for GMSU-SC and its extension for dynamic traffic demands.

REFERENCES

[1] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," *J. Netw. Comput. Appl.*, vol. 156, Apr. 2020, Art. no. 102563.

[2] Sandhya, Y. Sinha, and K. Haribabu, "A survey: Hybrid SDN," *J. Netw. Comput. Appl.*, vol. 100, pp. 35–55, Dec. 2017.

[3] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3259–3306, 4th Quart., 2018.

[4] *OpenFLow*. Accessed: Sep. 2021. [Online]. Available: https://opennetworking.org/sdn-resources/customer-case-studies/openflow

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, and J. Zolla, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM*, Hong Kong, Aug. 2013, pp. 3–14.

[6] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM*, Hong Kong, Aug. 2013, pp. 15–26.

[7] S. Natarajan, A. Ramaiah, and M. Mathen, "A software defined cloud-gateway automation system using OpenFlow," in *Proc. IEEE 2nd Int. Conf. Cloud Netw. (CloudNet)*, San Francisco, CA, USA, Nov. 2013, pp. 219–226.

[8] S. Cash, V. Jain, L. Jiang, A. Karve, J. Kidambi, M. Lyons, T. Mathews, S. Mullen, M. Mulsow, and N. Patel, "Managed infrastructure with IBM cloud openstack services," *IBM J. Res. Develop.*, vol. 60, nos. 2–3, pp. 6–12, Mar./May 2016.

[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 473–478, Sep. 2012.

[10] A. Ruiz-Rivera, K. W. Chin, and S. Soh, "GreCo: An energy aware controller association algorithm for software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 4, pp. 541–544, Apr. 2015.

[11] Y. Hu, T. Luo, N. C. Beaulieu, and C. Deng, "The energy-aware controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 741–744, Apr. 2017.

[12] G. Lin, S. Soh, K.-W. Chin, and M. Lazarescu, "Power-aware routing in networks with quality of services constraints," *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 1, pp. 122–135, Jan. 2016.

[13] *802.1AX-2020—Link Aggregation*. Accessed: Sep. 2021. [Online]. Available: https://1.ieee802.org/tsn/802-1ax-rev

[14] M. Rodriguez-Perez, M. Fernandez-Veiga, S. Herreria-Alonso, M. Hmila, and C. Lopez-Garcia, "Optimum traffic allocation in bundled energy-efficient Ethernet links," *IEEE Syst. J.*, vol. 12, no. 1, pp. 593–603, Mar. 2018.

[15] B. G. Assefa and Ö. Özkasap, "A survey of energy efficiency in SDN: Software-based methods and optimization models," *J. Netw. Comput. Appl.*, vol. 137, pp. 127–143, Jul. 2019.

[16] Y. Wei, X. Zhang, L. Xie, and S. Leng, "Energy-aware traffic engineering in hybrid SDN/IP backbone networks," *J. Commun. Netw.*, vol. 18, no. 4, pp. 559–566, Aug. 2016.

[17] H. Wang, Y. Li, D. Jin, P. Hui, and J. Wu, "Saving energy in partially deployed software defined networks," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1578–1592, May 2016.

[18] J. Galán-Jiménez, "Legacy IP-upgraded SDN nodes tradeoff in energy-efficient hybrid IP/SDN networks," *Comput. Commun.*, vol. 114, pp. 106–123, Dec. 2017.

[19] N. Huin, M. Rifai, F. Giroire, D. L. Pacheco, G. Urvoy-Keller, and J. Moulierac, "Bringing energy aware routing closer to reality with SDN hybrid networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 4, pp. 1128–1139, Dec. 2018.

[20] Z. Guo, W. Chen, Y.-F. Liu, Y. Xu, and Z.-L. Zhang, "Joint switch upgrade and controller deployment in hybrid software-defined networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1012–1028, May 2019.

[21] T. Das and M. Gurusamy, "Multi-objective control plane dimensioning in hybrid SDN/legacy networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2929–2942, Sep. 2021.

[22] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "One step at a time: Optimizing SDN upgrades in ISP networks," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.

[23] L. Hiryanto, S. Soh, K.-W. Chin, and M. Lazarescu, "Green multi-stage upgrade for bundled-link SDNs with budget and delay constraints," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1410–1425, Sep. 2021.

[24] L. Hiryanto, S. Soh, K.-W. Chin, D.-S. Pham, and M. M. Lazarescu, "Multi-path routing in green multi-stage upgrade for bundled-links SDN/OSPF-ECMP networks," *IEEE Access*, vol. 9, pp. 99073–99091, 2021.

[25] *OSPF Standardization Report*. Accessed: Sep. 2021. [Online]. Available: https://www.ietf.org/rfc/rfc2329.txt

[26] ONF. (Sep. 2013). *OpenFlow Switch Specification: Version 1.3.3 (Wire Protocol 0 × 04)*. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.3.pdf

[27] T. Das and M. Gurusamy, "Resilient controller placement in hybrid SDN/legacy networks," in *Proc. IEEE Globecom*, Abu Dhabi, UAE, Dec. 2018, pp. 1–7.

[28] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.

[29] M. T. I. U. Huque, W. Si, G. Jourjon, and V. Gramoli, "Large-scale dynamic controller placement," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 1, pp. 63–76, Mar. 2017.

[30] S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Towards optimal network planning for software-defined networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2953–2967, Dec. 2018.

[31] B. Görkemli, S. Tatlıcıoğlu, A. M. Tekalp, S. Civanlar, and E. Lokman, "Dynamic control plane for SDN at scale," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2688–2701, Dec. 2018.

[32] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The controller placement problem in software defined networking: A survey," *IEEE Netw.*, vol. 31, no. 5, pp. 21–27, Sep./Oct. 2017.

[33] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro, "IEEE 802.3az: The road to energy efficient Ethernet," *IEEE Commun. Mag.*, vol. 48, no. 11, pp. 50–56, Nov. 2010.

[34] P. Toth and S. Martello, *Knapsack Problems: Algorithms and Computer Implementations*. Hoboken, NJ, USA: Wiley, 1990.

[35] J. Y. Yen, "Finding the *k* shortest loopless paths in a network," *Manag. Sci.*, vol. 17, no. 11, pp. 712–716, Jul. 1971.

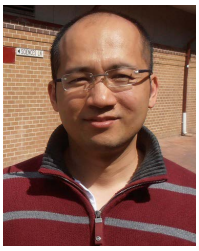
[36] *Gurobi Optimizer Reference Manual*. Accessed: Nov. 2019. [Online]. Available: http://www.gurobi.com

[37] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 93–96, Oct. 2005.

[38] G. Stampa, M. Arias, D. Sánchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," 2017, *arXiv:1709.07080*.
[39] P. Sun, Z. Guo, J. Lan, J. Li, Y. Hu, and T. Baker, "ScaleDRL: A scalable deep reinforcement learning approach for traffic engineering in SDN with pinning control," *Comput. Netw.*, vol. 190, May 2021, Art. no. 107891.

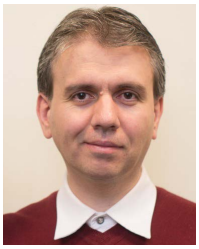
LELY HIRYANTO (Member, IEEE) received the bachelor's degree in computer science from Tarumanagara University, Indonesia, in 2001, and the master's diploma, Master of Science, and Ph.D. degrees in computer science from Curtin University, in 2015, 2016, and 2022, respectively. Her research interests include software defined networking, energy-aware traffic engineering, and network optimization.

SIETENG SOH (Member, IEEE) received the B.S. degree in electrical engineering from the University of Wisconsin–Madison, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Louisiana State University, Baton Rouge, in 1989 and 1993, respectively. From 1993 to 2000, he was with Tarumanagara University, Indonesia. He is currently a Senior Lecturer with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia. He has published over 100 international journals and conference papers. His current research interests include algorithm design, network optimization, and network reliability.



KWAN-WU CHIN received the Bachelor of Science degree (Hons.) and the Ph.D. degree with commendation from Curtin University, Australia, in 1997 and 2000, respectively. He was a Senior Research Engineer at Motorola, from 2000 to 2003. In 2004, he joined the University of Wollongong as a Senior Lecturer. He is currently an Associate Professor. To date, he holds four United States of America (USA) patents, and has published more than 180 journals and conference papers. His research interests include medium access control protocols for wireless networks, and resource allocation algorithms/policies for communications networks.

DUC-SON PHAM (Senior Member, IEEE) received the Ph.D. degree from the Curtin University of Technology, in 2005. He is currently a Senior Lecturer with the Discipline of Computing, Curtin University, and Perth, WA, Australia. His current research interests include sparse learning theory, large-scale data mining, convex optimization, and advanced deep learning with applications to computer vision and image processing. He was a recipient of the Young Author Best Paper Award 2010 for a publication in IEEE TRANSACTIONS ON SIGNAL PROCESSING.



MIHAI M. LAZARESCU (Member, IEEE) received the B.S. (Hons.) and Ph.D. degrees in computer science from Curtin University, Perth, Australia, in 1996 and 2000, respectively. He was a Senior Member of the IMPCA Research Institute for ten years. He is currently the Computing Discipline Lead and an Associate Professor with Curtin University. He has published over 80 papers in refereed international journals and conference proceedings in the areas of artificial intelligence, machine vision, data mining, and network reliability.