Load-Optimization in Reconfigurable Networks: Algorithms and Complexity of Flow Routing

Wenkai Dai, Klaus-Tycho Foerster, David Fuchssteiner, Stefan Schmid.
Faculty of Computer Science, University of Vienna

ABSTRACT

Emerging reconfigurable data centers introduce unprecedented flexibility in how the physical layer can be programmed to adapt to current traffic demands. These reconfigurable topologies are commonly hybrid, consisting of static and reconfigurable links, enabled by e.g. an Optical Circuit Switch (OCS) connected to top-of-rack switches in Clos networks. Even though prior work has showcased the practical benefits of hybrid networks, several crucial performance aspects are not well understood. For example, many systems enforce artificial segregation of the hybrid network parts, leaving money on the table.

In this paper, we study the algorithmic problem of how to jointly optimize topology and routing in reconfigurable data centers, in order to optimize a most fundamental metric, maximum link load. The complexity of reconfiguration mechanisms in this space is unexplored at large, especially for the following cross-layer network-design problem: given a hybrid network and a traffic matrix, jointly design the physical layer and the flow routing in order to minimize the maximum link load.

We chart the corresponding algorithmic landscape in our work, investigating both un-/splittable flows and (non-)segregated routing policies. We moreover prove that the problem is not submodular for all these routing policies, even in multi-layer trees, where a topological complexity classification of the problem reveals that already trees of depth two are intractable.

However, networks that can be abstracted by a single packet switch (e.g., nonblocking Fat-Tree topologies) can be optimized efficiently, and we present optimal polynomial-time algorithms accordingly. We complement our theoretical results with trace-driven simulation studies, where our algorithms can significantly improve the network load in comparison to the state of the art.

CCS CONCEPTS

• Networks → Network architectures; **• Theory of computation → Design and analysis of algorithms**.

ACM Reference Format:
Wenkai Dai, Klaus-Tycho Foerster, David Fuchssteiner, Stefan Schmid.. 2020. Load-Optimization in Reconfigurable Networks: Algorithms and Complexity of Flow Routing. In *PERFORMANCE '20, November 02–06, 2020, Milan, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/123456789

1 INTRODUCTION

Data centers nowadays empower everyday life in aspects such as business, health, industry, but also science and social interactions. With the rise of related data-intensive workloads as generated by machine learning, artificial intelligence, and the distributed processing of big data in general, data center traffic is growing very fast [49, 56]. Much of this traffic is internal to data centers, evoking considerable interest in data center design problems [50, 67].

Herein the emergence of a *programmable physical layer*, enabled by optical circuit switches [24, 38, 65], free-space optics [12, 30], or beamformed wireless connections [33, 34], leads to intriguing new possibilities, as leveraging fully electrically packet switched networks "*is increasingly cost prohibitive and likely soon infeasible*" [47, 48], see also the recent report by Microsoft [22]. In other words, electrical chips are unlikely to deliver sufficient performance for next-generation networks, and in turn we must rely on programmable optical topologies for increased bandwidth, connectivity, and power-efficiency [5].

Extensive past work has already shown significant benefits of such reconfigurable data center networks [28], but the underlying complexity is not well understood [11]. For example, many works artificially restrict their flow routing policies to be segregated between programmable and static network parts, aiming to place elephant flows on reconfigurable links [27].

Whereas some general algorithmic results exist w.r.t. latency [26, 30] or specific traffic patterns [10, 63], complexity questions of network-design for the objective of load-optimization are mostly uncharted. An exception is the work by Yang et al. [70], which focuses on the hardness induced by wireless interference, and the work by Zheng et al. [73], who study intractability on general graphs.

At the same time, link load is a most central performance metric [15, 33, 35, 59], and flow routing in traditional networks has been investigated for decades already [3]. We are thus motivated by the desire to take the first steps towards fundamentally understanding the network-design problem for load-optimization in data center networks, jointly considering flow routing and (interference-free) physical layer programmability enabled by, e.g., optical circuit switches.

1.1 Contributions

This paper initiates the network-design study of load-optimization in reconfigurable networks with optical circuit switches, leveraging the flexibility of emerging programmable physical layers for flow routing. We investigate multiple problem dimensions, from splittable to unsplittable flows, to fully flexible (non-segregated) versus segregated routing policies. Our results not only include efficient algorithms and complexity characterizations, but also simulations on real-world workloads:

(1) *Complexity:* We prove strong NP-hardness for non-segregated and segregated routing on trees of any height greater or equal

than two, for un-/splittable flow models. Moreover, all four problem settings are not submodular w.r.t. load-optimization, preventing common approximation techniques.

(2) *Algorithms:* In turn, we give polynomial-time optimal algorithms for the hybrid switch model of Venkatakrishnan et al. [62], which applies to non-blocking data center interconnects as, e.g., Fat-Trees. To this end we leverage a combination of subset matching results and topology-specific insights.

(3) *Evaluations:* Our workload-driven simulations (using Facebook, pFabric, and high-performance computing traces) show that our algorithms significantly improve on state of the art methods, decreasing the maximum load by 1.6× to 2.0×.

As a contribution to the research community and in order to ensure reproducibility of our results, we will make the source code of our algorithms as well as experimental artefacts publicly available.

Overview. We start with a formal model and preliminaries in §2, followed by complexity (§3) results for trees and algorithms for the hybrid switch model (§4). We then investigate the performance of our algorithms with trace-driven evaluations in §5. Lastly, we discuss related work in §6 and conclude in §7.

2 MODEL AND PRELIMINARIES

Network model.

Let $N=(V,E,\mathcal{E},C)$ be a *hybrid* network [43, 62] connecting the n nodes $V=\{v_1,...,v_n\}$ (e.g., top-of-the-rack switches), using static links E (usually connected by electrical packet switches). The network N also contains a set of reconfigurable (usually optical) links $\mathcal{E}$. The graph $(V,E\cup\mathcal{E})$ is a bidirected[1] graph such that two directions of each bidirected link $\{v_i,v_j\}\in E$ (resp. $\{v_i,v_j\}\in\mathcal{E}$), where $v_i,v_j\in V$, work as two (anti-parallel) directed links (v_i,v_j) and (v_j,v_i) respectively. We use the symbol $\overrightarrow{E}$ (resp. $\overrightarrow{\mathcal{E}}$) to denote the set of corresponding directed links of E (resp. $\mathcal{E}$). Moreover, a function $C:\overrightarrow{E}\cup\overrightarrow{\mathcal{E}}\mapsto\mathbb{R}^+$ defines capacities for both directions of each bidirected link in $E\cup\mathcal{E}$. Note that $(V,E\cup\mathcal{E})$ can be a multi-graph, e.g., when a reconfigurable link in $\mathcal{E}$ also connects two endpoints of a static link in E.

Reconfigured Network.

We say that a hybrid network N is *reconfigured* by a reconfigurable switch S if some reconfigurable links $M\subseteq\mathcal{E}$, which must induce *a matching*,[2] are *configured* (implemented) by S to enhance the static network (V,E). The set of configured (bidirected) links M, i.e., a matching, is called *a reconfiguration* of N. The enhanced network obtained by integrating the configured links M with the static links E of the hybrid network N is called a *reconfigured network*, i.e., $N(M)=(V,E\cup M)$. The static network (V,E) of the hybrid network N before reconfiguration can also be thought as a reconfigured network denoted by $N(\emptyset)$.

Hardware. Our results also apply to non-optical switches and links, as long as they match the theoretical properties described in the model. As such, we will only talk about reconfigurable switches and reconfigurable links, simply implying any appropriate technology that matches our model.

Topologies. Our network model does not place a restriction on the underlying static topology and hence can be applied generally. Notwithstanding, for our hardness results in §3, already simple trees suffice, whereas our positive algorithmic results cover many data center topologies, as we elaborate from §4 on.

Traffic Demands. The resulting network should serve a certain communication pattern, represented as a $|V|\times|V|$ communication matrix $D:=\left(d_{ij}\right)_{|V|\times|V|}$ (*demands*) with non-negative real-valued entries. An entry $d_{ij}\in\mathbb{R}^+$ represents the traffic load (frequency) or a demand from the node v_i to the node v_j. With a slight abuse of notation, let $D(v_i,v_j)$ also denote a demand from v_i to v_j hereafter.

Routing Models. For networking, *unsplittable* routing requires that all flows of a demand must be sent along a single (directed) path, while *splittable* routing does not restrict the number of paths used for the traffic of each demand; For a reconfigured network, segregated routing requires flows being transmitted on either static links or configured links, but non-segregated routing admits configured links to be used as shortcuts for flows along static links [24, 65]. Hence, there are *four different routing models*: *Unsplittable & Segregated (US)*, *Unsplittable & Non-segregated (UN)*, *Splittable & Segregated (SS)*, and *Splittable & Non-segregated (SN)*.

2.1 Load Preliminaries

"As minimizing the maximum congestion level in all links is a desirable feature of DCNs [33, 35]*, the objective of our work is to minimize the maximum link load"*

Yang et al. [70], to appear at ACM SIGMETRICS 2020

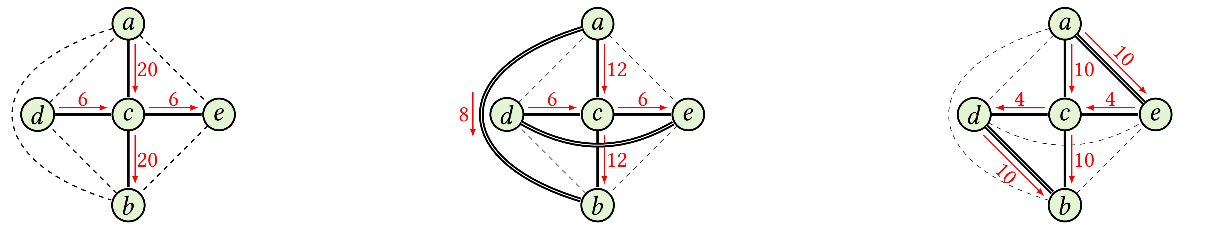
Load Optimization. Given a reconfigured network $N(M)$ and demands D, let $f:\overrightarrow{E}\cup\overrightarrow{M}\mapsto\mathbb{R}^+$ be *a feasible flow* serving demands D in $N(M)$ under a routing model $\tau\in\{\text{US,UN,SS,SN}\}$. *The load of each directed link* $e\in\overrightarrow{E}\cup\overrightarrow{M}$ induced by the flow f is defined as $L(f(e)):=f(e)/C(e)$. Then, for a feasible flow f in $N(M)$, *the maximum load* is defined as $L_{\max}(f):=\max\left\{L(f(e)):e\in\overrightarrow{E}\cup\overrightarrow{M}\right\}$, and there must be *an optimal flow* f_{opt} to serve D such that its maximum load is minimized for all feasible flows in $N(M)$. Such an optimal flow is called *a load-optimization flow* in $N(M)$.[3] For a reconfigured network $N(M)$, with slight abuse of notation, let f_{opt}^{M} denote an arbitrary load-optimization flow in $N(M)$ and we define a function $L_{\text{min-max}}(N(M)):=L_{\max}\left(f_{\text{opt}}^{M}\right)$.

Load-Optimization Reconfiguration Problem. Given a hybrid network N, a routing model $\tau\in\{\text{US,UN,SS,SN}\}$, and demands D, the *τ-load-optimization reconfiguration problem* is to find *an optimal reconfiguration $M\subseteq\mathcal{E}$* to generate *an optimally reconfigured network $N(M)$* such that $U_{\text{min-max}}(N(M))$ is minimized for all valid reconfigurations $M_i\subseteq\mathcal{E}$ of N. The *τ-load-optimization reconfiguration problem* is also abbreviated as the *τ-reconfiguration problem* henceforth. We lastly need to find a load-optimization flow for the optimally reconfigured network.

[1]Symmetrical connectivity is the standard industry assumption for static cabling, however for reconfigurable links as well. Outside highly experimental hardware, e.g. [30], off-the-shelf products use full-duplex connections [14, 51] and this model assumption is hence prevalent, even in Free-Space Optics [12] proposals.

[2]In other words, no two links in M are adjacent or share an endpoint, enforced by hardware constraints in practice (exclusive connections between ports).

[3]We note that in other works with analogous definitions, load might also be denoted by utilization, and load-optimization by load-balancing.



(a) A hybrid network N **(b) Reconfiguration $M_1 = \{\{a,b\},\{d,e\}\}$** **(c) Reconfiguration $M_2 = \{\{d,b\},\{a,e\}\}$**

Figure 1: Illustration of a hybrid network N with five nodes $\{a,b,c,d,e\}$ and four static links (drawn solid). A reconfigurable link (dashed) can be created between every pair of nodes, except c, as long as the set of reconfigurable links forms a matching. In this example, the capacity of each link is 20, and the task is to route the five demands $D(a,b) = 8$, $D(a,c) = 6$, $D(c,b) = 6$, $D(d,b) = 6$ and $D(a,e) = 6$, s.t. the maximum link load L is minimized. We utilize the splittable non-segregated (SN) model, which allows arbitrary routing on the (reconfigured) network, and we indicate the flow size on each link except for zero flow. In Fig 1a, we just consider the static topology, which induces a maximum link load of $20/20 = 1$. A first improvement strategy would be to greedily choose the biggest demand ($D(a,b) = 8$) and create the corresponding reconfigurable link, as in Fig. 1b. Then only a further reconfigurable link between d and e can be created, resulting in a maximum load of $12/20 = 0.6$. In contrast, an optimal solution has a maximum link load of $10/20 = 0.5$, shown in Fig. 1c.

Example. To illustrate the τ-load-optimization reconfiguration problem, we give a small example in Figure 1. Fig. 1a depicts the network before any reconfiguration, with five nodes $V = \{a,b,c,d,e\}$, four static (bidirected) links E: $\{d,c\}$, $\{b,c\}$, $\{a,c\}$ and $\{e,c\}$ and six reconfigurable (bidirected) links $\mathcal{E}$: $\{a,d\}$, $\{d,b\}$, $\{b,e\}$, $\{a,e\}$, $\{a,b\}$ and $\{d,e\}$.

We consider the routing model $\tau = \text{SN}$ and a capacity function $\forall e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} : C(e) = 20$, with the six demands $D(a,b) = 8$, $D(a,c) = 6$, $D(c,b) = 6$, $D(d,b) = 6$ and $D(a,e) = 6$. In Fig. 1a, each flow can only be routed along static links, creating a link load of $20/20 = 1$ on, e.g., (a,c) with three demands of size 8,6,6 from a. In order to improve the maximum link load, one could, e.g., greedily add reconfigurable links in order to reduce the maximum load, such as $\{a,b\}$ in Fig. 1b. Now, the demand $D(a,b) = 8$ is routed directly, reducing the maximum load to just 0.6. Yet, only one further reconfigurable link can be chosen, $\{d,e\}$, without violating the matching constraints. In this situation, any further rerouting does not decrease the maximum link load. For example, when attempting to alleviate the load of 0.6 on (c,b), the load on (a,c) will increase, and vice versa, in the best case cancelling each other's load increase.

Notwithstanding, we can improve the maximum load further. To this end, we select $\{a,e\}$ and $\{d,b\}$ as reconfigurable links, as shown in Fig. 1c. At first, this might seem counter-intuitive, as $D(a,e)$ and $D(d,b)$ are only of size 6 each, leaving a load of 0.7 on the links (a,c) and (c,b). However, the demand $D(a,b) = 8$ can be routed indirectly, via the path $\{a,e,c,d,e\}$, yielding an optimal maximum link load of 0.5.

3 COMPLEXITY

In this section we consider the underlying complexity of the load-optimization problem in reconfigurable networks. We begin with the investigation of NP-hardness, where we study segregated routing (§3.1) and non-segregated routing (§3.2). For all four routing models, we prove NP-hardness for trees of any height of two or greater.

Yang et al. [70] considered the case of unsplittable segregated routing on trees and weak NP-hardness, i.e., for large demand sizes. Our NP-hardness results also hold for small demand sizes and we moreover extend the previous result [70] to trees of height one. To show hardness, we can consider special cases, where all directed links have the same capacity of $\gamma \in \mathbb{R}^+$. In particular, we set $\gamma = 1$ in all our NP-hard proofs s.t. the load of each link equals the flow size on itself, but our proofs work for arbitrary γ.

We then prove in §3.3 that all four routing models are not submodular, i.e., resist common approximation schemes. Venkatakrishnan et al. [62] considered different objective functions and showed submodularity for the hybrid switching model, resulting in approximation algorithms which therefore cannot be applied here.

3.1 Segregated Routing

We start with the case of segregated routing w.r.t. NP-hardness. The following and some later proofs will make use of the strongly NP-hard 3-Partition problem, which we define first:

Definition 3.1 (3-Partition [29]). Given a finite set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$, and a size function: $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that each $s(a)$ satisfies $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$, can we partition A into m disjoint sets $A_1,\ldots,A_m$, such that for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$, where $|A_i| = 3$?

Theorem 3.2. *The τ-load-optimization reconfiguration problem, where $\tau \in \{US, SS\}$, are strongly NP-hard when the given hybrid network N, before reconfiguration, is a tree of height $h \geq 2$.*

Proof. We first consider the US-load-optimization reconfiguration problem. Given an instance of 3-Partition (A,B,s), we construct an instance of the US-reconfiguration problem as follows: the constructed tree T has the node r as its root, and for each element $a_i \in A$, r has a direct child $s_i \in S$. The root r also has m subtrees T_i for $1 \leq i \leq m$. For each subtree T_i, its root is the node r_i, which is a direct child of r, and the root r_i has $3m$ direct child nodes $F^i = \{f_1^i,\ldots,f_{3m}^i\}$. The tree T represents the static (bidirected) links E and nodes V of the hybrid network N. For the set of all reconfigurable links $\mathcal{E}$, if two nodes $u \in V$ and $v \in V$ are not connected by a static link in T, then there is a reconfigurable (bidirected) link $\{u,v\} \in \mathcal{E}$. Regarding demands D, for each $a_i \in A$, we define $D(f_i^k, s_i) = s(a_i)$ for each $1 \leq k \leq m$. Clearly, the constructed tree only has a height of two. For each $e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$, the capacity $C(e) = 1$. We claim that after N being reconfigured, there is no directed link $e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$ that has load higher than $(m-1)B$ if and only if there exists a valid 3-Partition for the set A.

Before reconfiguration, each static link $(r_i,r) \in \overrightarrow{E}$ has a flow size of mB. Assume A has a 3-partition $A_1,...,A_m$, where $\sum_{a \in A_i} s(a) = B$ for $1 \le i \le m$. For each $A_i = \{a_j,a_k,a_f\}$, where $1 \le i \le m$, we connect $s_j,s_k,s_f \in S$ to the corresponding nodes f_j^i,f_k^i,f_f^i in the subtree T_i, by the configured links. Thus, the flow size on each static link $(r_i,r) \in \overrightarrow{E}$ is decreased by B for $1 \le i \le m$, i.e., $(m-1)B$.

On the other hand, we assume that we could find a set of configured (bidirected) links $M \subseteq \mathcal{E}$ such that each static (directed) link $(r_i,r) \in \overrightarrow{E}$ for $1 \le i \le m$ does not have a flow size more than $(m-1)B$. Note that each element $a_i \in A$ has $B/4 < s(a_i) < B/2$. Due to $D(f_i^k,s_i) = s(a_i)$ for $1 \le k \le m$, for any two configured links in M, they cannot convey flows more than B. Moreover, if more than three (bidirected) links are configured between nodes S and the children nodes F^i in the subtree T_i, then there must be another subtree T_j, where $j \ne i$, whose children F^j have at most two configured links connecting to S, since the set of configured links must be a matching M. Thus, for each subtree T_i, where $1 \le i \le m$, there must be exactly three configured (bidirected) links between three nodes in F^i and three nodes in S. It is known that the direction from F^i to S in each of these three configured links should have a flow of size B. Let $s_j,s_k,s_f \in S$ be these three nodes in S connected to F^i by M, which exactly correspond to a partition $A_i \subseteq A$. Therefore, a valid 3-partition can be obtained.

To prove the hardness of the SS-reconfiguration problem, we use the same construction and claim as for the US-reconfiguration problem above. If A has a 3-partition $A_1,...,A_m$, where $\sum_{a \in A_i} s(a) = B$ for $1 \le i \le m$, then we have proven a valid solution M exists for the US-reconfiguration problem, which is also a solution for SS-reconfiguration since routing for unsplittable flows is a special case of the splittable flow variant. Recall that in the segregated model, a configured (directed) link $(u,v) \in \overrightarrow{\mathcal{E}}$ (resp. $(v,u) \in \overrightarrow{\mathcal{E}}$) can only carry flows for $D(u,v)$ (resp. $D(v,u)$). By this setting, we know configured links can only be between leaf nodes of T. If a reconfigurable (bidirected) link $\left\{s_i,f_i^k\right\}$, where $k \in \{1,...,m\}$, is configured, then all flows of the demand $D(f_i^k,s_i)$ can go through $\left(f_i^k,s_i\right) \in \overrightarrow{\mathcal{E}}$ even under a splittable model due to $D(f_i^k,s_i) \le (m-1)B$. On the other hand, if we can find a set of configured links $M \subseteq \mathcal{E}$ such that each link $(r_i,r) \in \overrightarrow{E}$, where $1 \le i \le m$, does not have a flow size more than $(m-1)B$, then we know each configured (directed) link $(f_i^k,s_i) \in \overrightarrow{M}$ must carry all flows of its demand $D(f_i^k,s_i)$. This corresponds to an unsplittable model, otherwise there must be a static (directed) link $(r_i,r) \in \overrightarrow{E}$, where $i \in \{1,...,m\}$, which has a flow size more than $(m-1)B$. The same conclusion can be drawn for the SS-reconfiguration problem. □

3.2 Non-Segregated Routing

For the non-segregated routing model, we obtain

- weak NP-hardness for trees of height $h=1$ in the UN model,
- strong NP-hardness for trees of height $h \ge 2$ in the UN model,
- strong NP-hardness for trees of height $h \ge 2$ in the SN model.

The proofs are deferred to appendices A.1 and A.2.

3.3 Non-Submodularity

The submodularity of objective functions plays an important role in approximating optimization problems [61], as by Venkatakrishnan et al. [62] for hybrid switch networks. However their objective function does not consider load-balancing and therefore does not apply in our setting, as we show next.

Definition of submodularity. We recall the definition of submodularity [31]: A function $f: 2^B \mapsto \mathbb{R}$, where 2^B is a power set of a finite set B, is submodular if it satisfies that for every $X,Y \subseteq B$ with $X \subseteq Y$ and every $x \in B \setminus Y$,

$$f(X \cup \{x\}) - f(X) \ge f(Y \cup \{x\}) - f(Y).$$

Overview. In this section, we investigate the submodularity of the objective function Φ of a τ-reconfiguration problem, which minimizes the maximum load of reconfigured networks $N(M)$, i.e., $L_{\text{min-max}}(N(M))$, for all valid reconfigurations M of a given hybrid network N. Moreover, we are also interested in the submodularity of the objective function Ω that maximizes the gap of the minimized maximum load between the given hybrid network N before reconfiguration and reconfigured networks $N(M)$ for reconfigurations M of N. We will show that both functions Φ and Ω are not submodular functions. To prove the functions to be not submodular, we present special instances as counter-examples.

Theorem 3.3. *For a τ-reconfiguration problem, the objective function Φ that minimizes $L_{\text{min-max}}(N(M))$ for all reconfigurations M of N, is not submodular when $\tau \in \{US, UN, SS, SN\}$.*

Proof. For a hybrid network $N = (V,E,\mathcal{E},C)$, we define the set of nodes $V = U \cup Q$, where $U = \{u_i : i = 1,2,3\}$ and $Q = \{q_i : i = 1,2,3\}$. For static (bidirected) links E, we have six options: $\{u_1,u_2\} \in E$, $\{u_1,u_3\} \in E$, $\{q_1,q_2\} \in E$, $\{q_1,q_3\} \in E$, and $\{u_2,q_2\} \in E$, $\{u_3,q_3\} \in E$. For each $i \in \{1,2,3\}$, there is a reconfigurable (bidirected) link $\{u_i,q_i\} \in \mathcal{E}$. W.l.o.g., each (directed) link in $\overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$ has the same capacity $\gamma \in \mathbb{R}^+$. Let our objective function $\Phi: 2^{\mathcal{E}} \mapsto \mathbb{R}^+$ be a function defined by an equation $\Phi(M) = L_{\text{min-max}}(N(M))$, where $M \in 2^{\mathcal{E}}$ is a reconfiguration (matching). Recall the definition of submodularity. We define $X = \{\{u_2,q_2\}\} \subseteq \mathcal{E}$, $Y = \{\{u_1,q_1\},\{u_2,q_2\}\} \subseteq \mathcal{E}$ and $x = \{u_3,q_3\} \in \mathcal{E}$. When τ = SS, SN, we define demands as follows: $D(u_3,q_3) = 3$, $D(u_2,q_2) = 3$, $D(q_2,u_2) = 3$ and $D(u_1,q_1) = 3$. When τ = SS, we have $\Phi(X \cup \{x\}) = \frac{9}{4\gamma}$, $\Phi(X) = \frac{3}{\gamma}$, $\Phi(Y \cup \{x\}) > \frac{3}{2\gamma}$, and $\Phi(Y) \le \frac{9}{4\gamma}$. Thus, the Inequality (1) shows that the function Φ is not submodular.

$$\Phi(X \cup \{x\}) - \Phi(X) = \frac{-3}{4\gamma} < \Phi(Y \cup \{x\}) - \Phi(Y) > \frac{-3}{4\gamma} \qquad (1)$$

When τ = SN, we have $\Phi(X \cup \{x\}) = \frac{9}{4\gamma}$, $\Phi(X) = \frac{3}{\gamma}$, $\Phi(Y \cup \{x\}) = \frac{9}{5\gamma}$, and $\Phi(Y) = \frac{9}{4\gamma}$. Thus, the Inequality (2) shows that the function Φ is not submodular.

$$\Phi(X \cup \{x\}) - \Phi(X) = \frac{-3}{4\gamma} < \Phi(Y \cup \{x\}) - \Phi(Y) = \frac{-1.8}{4\gamma} \qquad (2)$$

When τ = US, we modify our above constructed network N by adding one more node d and one more static link $\{d,q_2\} \in E$, while reconfigurable links $\mathcal{E}$ are unchanged. We define new demands as follows: $D(u_3,q_3) = 3$, $D(u_2,d) = 3$, and $D(u_1,q_1) = 3$. Now, we have $L_{\text{min-max}}(N(\emptyset)) = 6$. When τ = US, we know $\Phi(X \cup \{x\}) = 3$, $\Phi(X) = 6$,

$\Phi(Y\cup\{x\})=3$, and $\Phi(Y)=3$. Thus, the Inequality (3) shows that the function Φ is not submodular.

$$\Phi(X\cup\{x\})-\Phi(X)=-3/\gamma<\Phi(Y\cup\{x\})-\Phi(Y)=0 \qquad (3)$$

When $\tau=$UN, we extend the above constructed network N for US by adding one more static link $\{d,q_1\}\in E$ and removing the static link $\{q_1,q_2\}\in E$, while reconfigurable links $\mathcal{E}$ are unchanged. We define new demands as follows: $D(u_3,q_3)=3$, $D(u_2,d)=3$, and $D(u_1,d)=3$. Now, we know $\Phi(X\cup\{x\})=3/\gamma$, $\Phi(X)=6/\gamma$, $\Phi(Y\cup\{x\})=3/\gamma$, and $\Phi(Y)=3/\gamma$. Thus, the function Φ is not submodular. □

Theorem 3.4. *For a τ-reconfiguration problem, the objective function Ω that maximizes $L_{min\text{-}max}(N(\emptyset))-L_{min\text{-}max}(N(M))$ for all reconfigurations M of N, is not submodular, when $\tau\in\{US, UN, SS, SN\}$.*

Proof. For a hybrid network $N=(V,E,\mathcal{E},C)$, we define nodes $V=U\cup Q\cup P$, where $U=\{u_i:i=1,2,3\}$, $P=\{p_i:i=1,2,3\}$ $Q=\{q_i:i=1,2,3\}$; For each $i\in\{1,2,3\}$, we set two static links $\{u_i,q_i\}\in E$ and $\{p_i,u_i\}\in E$, and a reconfigurable link $\{u_i,q_i\}\in\mathcal{E}$, and two demands $D(u_i,q_i)=3$ and $D(p_i,q_i)=3$. Let our objective function $\Omega:2^{\mathcal{E}}\mapsto\mathbb{R}^+$ be defined by an equation $\Omega(M)=\omega-L_{\text{min-max}}(N(M))$, where a reconfiguration (matching) $M\in 2^{\mathcal{E}}$ and $\omega=L_{\text{min-max}}(N(\emptyset))$. With loss of generality, for each (directed) link in $\overrightarrow{E}\cup\overrightarrow{\mathcal{E}}$, it has the capacity γ. Recall the definition of submodularity. We set $X=\{\{u_1,q_1\}\}$, $Y=\{\{u_1,q_1\},\{u_2,q_2\}\}$ and $x=\{u_3,q_3\}$. For each routing model $\tau\in\{$US, UN, SS, SN$\}$, we always have $\Omega(X\cup\{x\})=\omega-6/\gamma$, $\Omega(X)=\omega-6/\gamma$, $\Omega(Y\cup\{x\})=\omega-3/\gamma$, and $\Omega(Y)=\omega-6/\gamma$. Hence, Ω is not submodular. □

4 HYBRID SWITCH NETWORKS

As we saw before, already simple tree networks of height ≥ 2 are intractable, and optimizations leveraging submodularity are not possible. This raises the question if we can obtain *optimal* and *polynomial-time* algorithms for meaningful settings.

4.1 Non-Blocking Data Center Topologies

Common data center topologies have trees of height 2 as subgraphs or minors and hence seem like bad candidates for efficient algorithms at first glance. However, already early designs adapted from telecommunications such as Clos [17] topologies have a so-called non-blocking property, which we can use to our advantage. An interconnecting topology C is non-blocking, if the servability of a flow from v_1 to v_2 via C only depends on the utilization of the links (v_1,C) and (C,v_2) : "**such an interconnect behaves like a crossbar switch**" [71]. Non-blocking interconnects have become popular data center topologies [4] in particular in the form of folded Clos networks or Fat-Trees [41], depicted in Fig. 2a: the actual topology inside the interconnect (marked in a blue rectangle) is immaterial and we only need to consider the links incident to the nodes.[4]

As thus, for our purposes, we can abstract the data center interconnect C (which can be understood as a packet switch) by a single center node c, leaving our previous intractability considerations behind. We hence turn our attention to *hybrid switch networks* as considered by of Venkatakrishnan et al. [62], which are represented by a packet and a circuit switch connected to all nodes, see Fig. 2b.

[4]We note that the non-blocking property can also be restricted to keep distributed routing schemes in mind, we refer to Yuan [71] for an in-depth discussion.

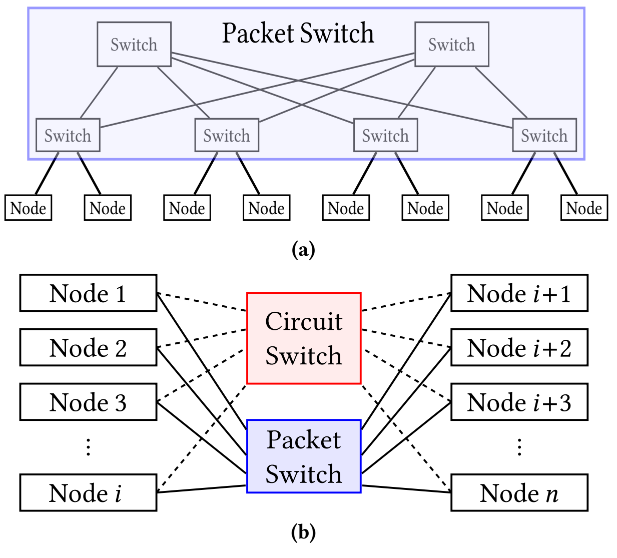


Figure 2: Illustration of a Fat-Tree network in 2a and a hybrid switch network in 2b, as in [62, Figure 1]. Due to the non-blocking property of network in 2a, we can abstract the interconnect enclosed by the blue box as a packet switch, as depicted in 2b. Additionally, a hybrid switch network also contains a reconfigurable circuit switch (e.g. an OCS) that provides a matching of the nodes, to be optimized for the demands D. Hence, the augmentation of a Fat-Tree by an OCS can also be investigated from the viewpoint of hybrid switch networks

Routing in hybrid switch networks is straightforward (only one path exists for each node pair in the packet switched network), but the addition of a circuit switch adds a large degree of freedom: First, the number of possible matchings grows exponentially, and second, we have to decide for each flow which path to take as well. Notwithstanding, the special structure of hybrid switch networks allows us to solve reconfiguration and routing efficiently.

We structure our approach as follows. We first introduce an auxiliary problem in §4.2 and a constant-time triangle graph algorithm in §4.3, which we then leverage for our optimal algorithm in §4.4. We lastly discuss performance bounds and extensions in §4.5.

4.2 Red-Target Matching

Later in our algorithms, we will mark nodes (in red) which need to be connected to e.g. the OCS in order to satisfy a load threshold. Not all reconfigurable connections are suitable for such a task, and we will identify them in the next subsection. Given such red nodes and corresponding links, the question is if the red nodes can be matched accordingly, which we denote as Red-Target Matching:

Definition 4.1 (Red-Target Matching (RTM)). Given a graph $G=(V,E)$ and a coloring $l:V\mapsto\{r,b\}$, the question is to find a matching M of G such that each node $v\in V$ having the color $l(v)=r$ is an endpoint of an edge of M.

Lemma 4.2. *The RTM problem (Definition 4.1) can be solved by a maximum-weight matching algorithm in polynomial time.*

Proof. For a given graph $G = (V,E)$, if an edge $e \in E$ has both endpoints of color b (blue), then it can be removed directly. For each edge $e \in E$ having both endpoints of color r (red), we set the weight $w(e) = 2$, and for each edge $e \in E$ that has only one endpoint of color r, we set $w(e) = 1$. Let the number of red-colored nodes in G be n. If we can find a matching M of the weight n, then all of these n red nodes are contained in M. There is no matching that can have a weight more than n, otherwise the number of red-colored nodes in G is more than n. Therefore, if RTM problem has a solution then a maximum-weight algorithm can always find a valid matching M for RTM. If the maximum-weight matching has a weight of less than n, then RTM has no solution. Lastly, a maximum-weight matching is solvable in polynomial time, e.g., by the Blossom algorithm [21], which has a running time $O(|E||V|^2)$. □

4.3 Selection of Suitable Reconfigurable Links

In the studied hybrid switch networks, reconfigurable links can be created between any pair of nodes connected to the packet switch, e.g. via an OCS. While we will select the (matching) subset of reconfigurable (bidirected) links in the next subsection, we herein identify the benefit of adding specific reconfigurable links.

Lemma 4.3. *Given a reconfigured network $N(M)$, which is a triangle on three nodes $V = \{a,b,c\}$ with the only configured (bidirected) link $\{a,b\} \in \mathcal{E}$ and two static (bidirected) links $\{c,a\},\{c,b\} \in E$, then for demands D, a load-optimization flow in $N(M)$ can be computed in a constant time under routing models $\tau \in \{US,SS,SN\}$.*

Proof. In the triangle $N(M)$, there are at most six demands in D and six directed links $\overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$. For each demand, e.g., $D(a,b)$, the directed link (a,b) is called the shortcut of $D(a,b)$, and the directed path (a,c,b) from a to b is called the indirect path of $D(a,b)$.

For the segregated routing model, demands $D(c,a)$, $D(a,c)$, $D(c,b)$ and $D(b,c)$ can be only sent on their shortcuts (c,a) (a,c), (c,b) and (b,c) respectively, which are static links in $\overrightarrow{E}$, and then we only need to consider $D(a,b)$ and $D(b,a)$. Moreover, for the unsplittale routing model, each demand, e.g., $D(a,b)$, can only be sent on a single path: either its shortcut (a,b) or its indirect path (a,c,b).

When τ = US, $D(a,b)$ can only be sent through either (a,b) or (a,c,b), and the similar argument can be applied to $D(b,a)$. In terms of the given capacity function C, a load-optimization flow can be decided by searching these four different routing possibilities for $D(a,b)$ and $D(b,a)$, which is in a constant time.

When τ = SS, the respective load values for directed links (a,c), (c,b) and (a,b) only depends on how the demand $D(a,b)$ divides its traffic between the indirect path (a,c,b) and its shortcut (a,b), while an analogous argument can be applied to the demand $D(b,a)$. Thus, a load-optimization flow can be decided in a constant time.

When τ = SN, a load-optimization flow in $N(M)$ can be computed in constant time as well. Due to space constraints, the detailed proof is shown in Lemma A.5 in the Appendix. □

It remains to utilize the single triangle algorithms in a larger context (Lemma 4.4) and to determine the resulting runtime (Lemma 4.5):

Lemma 4.4. *Given a hybrid switch network N on leaves $V' = \{v_1,...,v_n\}$, where we denote the central packet switch by a node c,*

```
Algorithm 1: Preprocess Triangles
  Input  : Hybrid switch network $N = (V,E,\mathcal{E},C)$,
           demands $D$ and a routing model $\tau \in \{US,SS,SN\}$;
  Output : A set of 2-tuple $S^\Delta$ ;
1 $S^\Delta := \emptyset$;
2 for each reconfigurable (bidirected)
    link $\{v_i,v_j\} \in \mathcal{E}$, where $v_i,v_j \in V$ are leaf nodes in $N$ do
3     define the triangle on nodes $\{v_i,v_j,c\}$, where a configured
        (bidirected) link $\{v_i,v_j\}$ and two static (bdirected)
        link $\{\{v_i,c\},\{v_j,c\}\}$, and the new demands $D'$;
4     let $D'(v_i,v_j) = D(v_i,v_j)$ and $D'(v_j,v_i) = D(v_j,v_i)$;
5     foreach $u \in \{v_i,v_j\}$ do
6         Let $D'(u,c) = \sum_{v \in V \setminus \{v_i,v_j\}} D(u,v)$;
7         let $D'(c,u) = \sum_{v \in V \setminus \{v_i,v_j\}} D(v,u)$;
8     for a routing model $\tau \in \{US,SS,SN\}$,
        by Lemma 4.3, compute a load-otimization flow $f^\Delta_{ij}$ in
        the triangle $\{v_i,v_j,c\}$ for demands $D'$ in a constant time;
9     compute the minimized maximum load: $\mu^\Delta_{ij} := L_{max}(f^\Delta_{ij})$;
10    $S^\Delta = S^\Delta \cup \left\{\left(\mu^\Delta_{ij}, f^\Delta_{ij}\right)\right\}$;
11 return $S^\Delta$;
```

demands D, and a routing model $\tau \in \{US,SS,SN\}$, for any reconfiguration M' of N, let f be an arbitrary flow *serving D in the reconfigured network $N(M')$. Let $\{v_i,v_j\} \in M'$ be any configured (bidirected) link in M', where $v_i,v_j \in V'$. For the triangle on $\{v_i,v_j,c\}$, let E^Δ_{ij} denote the six (directed) links of this triangle, i.e., $E^\Delta_{ij} \subset \overrightarrow{E} \cup \overrightarrow{M}$, and let μ^Δ_{ij} be the minimized maximum load computed by Algorithm 1 for the triangle $\{v_i,v_j,c\}$. We then obtain:*

$$\max\left\{L(f(e)) : e \in E^\Delta_{ij}\right\} \geq \mu^\Delta_{ij}. \tag{4}$$

Proof. Let the given hybrid switch network N have nodes $V = V' \cup \{c\}$, where c is the central packet switch node and $V' = \{v_1,...,v_n\}$ are leaf nodes (leaves). Recall that a reconfiguration M' must be a matching. Thus, in the reconfigured network $N(M')$, for each configured (bidirected) link $\{v_i,v_j\} \in M'$, where $v_i,v_j \in V'$, the node v_i (resp. v_j) only connects to nodes c and v_j (resp. v_i). Let f be an arbitrary flow serving D in $N(M')$. Any partial flow of f that start from the node $u \in \{v_i,v_j\}$ and end at a node $v \in V \setminus \{v_i,v_j\}$ must go through the center c to leave the triangle $\{v_i,c,v_j\}$, and the size of these flows must be $D'(u,c)$ defined in Algorithm 1; on the other hand, any partial flow (sub-flow) of f that starts from a node $v \in V \setminus \{v_i,v_j\}$ and ends at the node $u \in \{v_i,v_j\}$ must go through the center c to enter the triangle $\{v_i,c,v_j\}$ and the size of these sub-flows must be $D'(c,u)$ defined in Algorithm 1. Therefore, the local sub-flows of f inside the triangle $\{v_i,v_j,c\}$ satisfy the demands D' defined in Algorithm 1. Since μ^Δ_{ij} denotes the maximum load of a local load-optimization flow serving D' in $\{v_i,v_j,c\}$, then by the correctness of Lemma 4.3, Inequation (4) holds directly. □

Lemma 4.5. *In Algorithm 2, a load-optimization flow f_{final} serving D in the reconfigured hybrid switch network $N(M)$ under a routing* *model $\tau \in \{US,SS,SN\}$ can be constructed in a runtime of $O(n^2)$, where the number of demands D is at most n^2.*

Proof. We note that a local load-optimization flow f_{ij}^{Δ} for each triangle $\{v_i,v_j,c\}$ where $\{v_i,v_j\} \in \mathcal{E}$, has been computed in Algorithm 1 according to τ and D.

Let M be an optimal reconfiguration for the hybrid switch network N and demands D. For each configured (bidirected) link $\{v_i,v_j\} \in M$, the set S^{Δ} returned by Algorithm 1 contains a load-optimization flow for the triangle $\{v_i,v_j,c\}$ and D'. We first construct the related sub-flow serving an arbitrary demand $D(v_i,v_j)$ in f_{final}, where v_i and v_j are leaf nodes. First, if $\{v_i,v_j\} \in M$, then the flow for $D(v_i,v_j)$ (resp. $D(v_j,v_i)$) is already given in f_{ij}^{Δ} contained in S^{Δ}. Second, if there are two configured links $\{v_i,v_k\}$ and $\{v_j,v_l\}$ in M, then the flow of $D(v_i,v_j)$ obtained by merging the sub-flow of size $D(v_i,v_j)$ in f_{ik}^{Δ} serving $D'(v_i,c)$ and the sub-flow of size $D(v_i,v_j)$ in f_{jl}^{Δ} serving $D'(c,v_i)$ on the joint center c. If only v_i is contained in a configured link $\{v_i,v_l\} \in M$, then the sub-flow serving $D(v_i,v_j)$ in f_{final} is obtained by extending the sub-flow of size $D(v_i,v_j)$ in f_{il}^{Δ} serving $D'(v_i,c)$ from the destination c to the node v_j. Moreover, for the demand $D(v_i,c)$ (resp. $D(c,v_i)$), if v_i is contained in a configured link $\{v_i,v_k\} \in M$, then the sub-flow serving $D(v_i,c)$ (resp. $D(c,v_i)$) in f_{final} can be found in the local flow f_{ik}^{Δ} in S^{Δ} directly; otherwise we send its flow directly on the static (directed) link (v_i,c) (resp. (c,v_i)) in f_{final}.

Lastly, for each demand, its flow can be constructed in constant time in Algorithm 2. Thus, the running time to construct f_{final} relies on the number of demands D, which is $O(n^2)$. □

4.4 Solving Hybrid Switch Networks Optimally

We now combine our previous results to optimally solve the reconfiguration problem on hybrid switch networks.[5]

Theorem 4.6. *If each reconfigurable link in $\mathcal{E}$ is only between two leaf nodes, then the τ-reconfiguration problem on hybrid switch networks is solved optimally by Algorithm 2 when $\tau \in \{US,SS,SN\}$.*

Proof. For a hybrid switch network N, let M be an optimal reconfiguration for N , and let $\mu_{\min} = L_{\text{min-max}}(N(M))$ denote the minimized maximum load of the reconfigured network $N(M)$. Note that such an optimal reconfiguration M always exists for hybrid switch networks N, when N has at least two leaf nodes. Thus, we prove that Algorithm 2 can find such an optimal solution M.

For a load-optimization flow f_{opt} of $N(M)$, there must be at least a directed link e^* in $N(M)$ such that $L(f_{\text{opt}}(e^*)) = \mu_{\min}$. If e^* is a static link and no configured link in M is incident with e^* on a leaf node, then $\mu_{\min}$ must be stored in T in Algorithm 2; Otherwise, e^* must be contained in a triangle $\{v_i,c,v_j\}$, where $\{v_i,v_j\} \in M$ and c is the center. In the triangle $\{v_i,c,v_j\}$, by Lemma 4.3, Algorithm 1 can find a load-optimization flow f_{ij}^{Δ} serving D' s.t. $\mu_{ij}^{\Delta} = L_{\max}\left(f_{ij}^{\Delta}\right)$. By Lemma 4.4, in the triangle $\{v_i,c,v_j\}$, we know $L\left(f_{\text{opt}}(e^*)\right) \geq \mu_{ij}^{\Delta}$. Since f_{opt} is a load-optimization flow for $N(M)$, then we know $\mu_{\min} = \mu_{ij}^{\Delta}$, otherwise $L\left(f_{\text{opt}}(e^*)\right)$ can be further decreased. Thus, $\mu_{\min}$ can be computed by Algorithm 1 and finally be stored in the set of thresholds T. Since the binary search goes through T exhaustively, then $\mu_{\min}$ can be always detected and used as a threshold for Algorithm 3 to search for a matching.

[5] Recall that the UN model is NP-hard on hybrid switch networks (Section 3.2).

Algorithm 2: Reconfiguration for Hybrid Switch Networks

```
Input  :A hybrid switch network $N = (V,E,\mathcal{E},C)$,
        with nodes $V = \{v_1,...,v_n,c\}$, where $c$ is the center node
        having leaves $\{v_i,...,v_n\}$, static links $E = \{\{v_i,c\} : v_i \in V \setminus \{c\}\}$,
        demands $D$, and a routing model $\tau \in \{US,SS,SN\}$;
Output :A reconfiguration
        $M$ and a load-optimization flow $f_{\text{final}}$ of $N(M)$ or "null";
1  define a set of thresholds $T := \emptyset$;
2  find the original (unique) flow $f_{\text{old}} : \overrightarrow{E} \mapsto \mathbb{R}^+$
     serving all demands $D$ on $N$ before reconfiguration;
3  for each static (bidirected) link $\{v_i,c\} \in E$ do
4      $T = T \cup \{U(f_{\text{old}}(v_i,c)), U(f_{\text{old}}(c,v_i))\}$;
5  run Algorithm 1 on $(N,D,\tau)$ to obtain a set of 2-tuples $S^{\Delta}$;
6  for each pair $(\mu_{ij}^{\Delta}, f_{ij}^{\Delta}) \in S^{\Delta}$ do
7      put the value $\mu_{ij}^{\Delta}$ into $T$;
8  run Algorithm 3 on the input $\left(N,D,f_{\text{old}},T,S^{\Delta}\right)$;
9  if Algorithm 3 returns "null" then
10     return "null";
11 else
12     let $M$ be a RTM matching returned
         by Algorithm 3 under the minimum value $\mu_{\min} \in T$;
13     construct the configured network $N(M)$;
14     compute a load-optimization flow $f_{\text{final}}$
         for $N(M)$, $D$ and $\tau$, based on $S^{\Delta}$ and $f_{\text{old}}$ (Lemma 4.5);
15     return a reconfiguration $M$ and $f_{\text{final}}$;
```

Now, we prove that, when each reconfigurable link in $\mathcal{E}$ is between two leaf nodes in V, given a threshold $\mu_{\min}$, Algorithm 2 can find an optimal reconfiguration M for a hybrid switch network N and a flow f serving D in $N(M)$ such that $L_{\max}(f) \leq \mu_{\min}$.

Before reconfiguration, on the original flow f_{old}, for each static (bidirected) link $\{v_i,c\} \in E$, where $v_i \in V \setminus \{c\}$, if it has $L(f(v_i,c)) > \mu_{min}$ or $L(f(c,v_i)) > \mu_{min}$, then its leaf node v_i must be contained in a configured link in M, otherwise, the loads of (v_i,c) and (c,v_i) are unchanged after reconfiguration. Thus, in Algorithm 3, we color such nodes by "red" and try to find a matching to cover all "red" nodes. For each (bidirected) link $\{v_i,v_j\} \in \mathcal{E}$, where v_i and v_j are leaf nodes, it implies a triangle $\{v_i,c,v_j\}$; while the link $\{v_i,v_j\} \in \mathcal{E}$ is preserved in the auxiliary graph G if only if the minimized maximum load μ_{ij}^{Δ} in $\{v_i,c,v_j\}$ serving D' no larger than $\mu_{\min}$. Lemma 4.1 ensures that a matching M covering all "red" nodes in G must be detected if it exists. Due to the way of constructing G, for each $\{v_i,v_j\} \in M$, Lemma 4.3 implies that the local load-optimization flow f_{ij}^{Δ} in $\{v_i,v_j,c\}$ serving D' has the maximum load $L_{\max}\left(f_{ij}^{\Delta}\right) \leq \mu_{\min}$. Lemma 4.5 guarantees that a load-optimization flow f_{final} serving D in the $N(M)$ can be constructed such that $L_{\max}(f_{\text{final}}) \leq \mu_{\min}$. Note that for any static link $\{v_k,c\} \in E$, if v_k is not contained in any configured link of M, then $L(f_{\text{final}}(v_k,c)) = L(f_{\text{old}}(v_k,c)) \leq \mu_{\min}$ and $L(f_{\text{final}}(c,v_k)) = L(f_{\text{old}}(c,v_k)) \leq \mu_{\min}$. □

Algorithm 3: Determine Reconfiguration

```
Input  : A hybrid switch network $N$, demands $D$, the original flow $f_{\text{old}}$, a set of thresholds $T$ and a set of 2-tuples $S^{\Delta}$;
Output : Configured links (a RTM matching) $M$ or "null";
1  let $V' := V \setminus \{c\}$ be all leaf nodes in $N$;
2  sort thresholds in $T$ in ascending order;
3  for each threshold value $\mu \in T$ do
4      create an extra graph $G$ on leaf nodes $V'$ without edges;
5      for each static (bidirected) link $\{v_i,c\} \in E$ do
6          if its $L(f_{\text{old}}(v_i,c)) > \mu \vee L(f_{\text{old}}(c,v_i)) > \mu$ then
7              color the corresponding node $v_i \in V'$ by "red";
8          else
9              color the corresponding node $v_i \in V'$ by "blue";
10     for each pair $(\mu^{\Delta}_{ij}, f^{\Delta}_{ij}) \in S^{\Delta}$ do
11         if $\mu^{\Delta}_{ij} \le \mu$ then
12             add an edge $\{v_i,v_j\}$ into $G$;
13     solve RTM problem
         on $G$ to find a matching $M$ to cover all "red" nodes in $G$;
14     if a RTM matching $M$ is detected then
15         return $M$;
16     remove the graph $G$;
17 return "null";
```

We now briefly show that our algorithms also extend to the case where we can create a reconfigurable link to the central packet switch and also bound the runtime:

Theorem 4.7. *The τ-reconfiguration problem on hybrid switch networks is solved optimally by Algorithm 2 in a polynomial time $O(n^4 \log n)$, where n is the number of nodes, when $\tau \in \{US, SS, SN\}$.*

Proof. Theorem 4.6 has shown the correctness under the restriction, where each reconfigurable link in $\mathcal{E}$ must be between two leaf nodes. Now, we will show that a τ-reconfiguration problem can still be solved by Algorithm 2 without the restriction.

If $\mathcal{E}$ contains a reconfigurable link $\{v_i,c\}$, where v_i is a leaf node and c is the center, we could create an additional leaf node v_{n+1} in V. To introduce additional demands, for each $u \in V \setminus \{v_{n+1}\}$, we define $D(v_{n+1},u) := 0$ and $D(u,v_{n+1}) := 0$. Then for each reconfigurable link $\{v_i,c\} \in \mathcal{E}$, we remove it from $\mathcal{E}$ and add a new reconfigurable link $\{v_i,v_{n+1}\}$ into $\mathcal{E}$. For the matching M and a load-optimization flow f_{final} returned by Algorithm 2, if $\{v_i,v_{n+1}\} \in M$, then remove it and add $\{v_i,c\}$ into M, and update f_{final} by moving flow on the directed path (v_i,v_{n+1},c) (resp. (c,v_{n+1},v_i)) to the existing configured link (v_i,c) (resp. (c,v_i)). If no link in $\mathcal{E}$ contains the center c, then Theorem 4.6 has proved the correctness.

Now, we assume that there must be at least one reconfigurable link in $\mathcal{E}$ containing c. Note that the original $\mu_{\min}$ can be still stored in T after the above processing. Thus, the τ-reconfiguration problem on hybrid switch networks is solved optimally by Algorithm 2.

Runtime analysis. Careful analysis reveals that the running time of Algorithm 2 is dominated by the binary search over values of T, and invoking the polynomial-time Blossom algorithm [21] to find a maximum matching for each value of T. Hence, the overhead of binary search is logarithmic in $|T|$, due to to $|T| = |E| + |\mathcal{E}|$. Since the Blossom algorithm [21] can compute a maximum matching in $O\left(|E||V|^2\right)$, Algorithm 2 can be completed in $O(n^4 \log n)$. □

4.5 Bounds and Extensions

Given that we provided optimal algorithms for hybrid switch networks above, we now investigate theoretical performance bounds and extensions. As such, we provide bounds on the improvement of the load, prove that maximum matching algorithms do not perform well in terms of competitive analysis, and show how our algorithms can be extended to multiple small reconfigurable switches.

Improvement bounds. As we will show next, by leveraging e.g. an OCS, the maximum load can at most be decreased by a factor of two in hybrid switch networks. This result also holds under any routing extensions, e.g., flows can take multiple paths and use several reconfigurable links.

Lemma 4.8. *Given a hybrid hybrid switch network N, demands D and a routing model $\tau \in \{US, UN, SS, SN\}$, for any reconfiguration M of N, we have $L_{\text{min-max}}(N(M)) \ge L_{\text{min-max}}(N(\emptyset))/2$.*

Proof. Given a hybrid hybrid switch network $N = (V,E,\mathcal{E},C)$, demands D and a routing mode $\tau \in \{\text{US, UN, SS, SN}\}$, let $M^* \subseteq \mathcal{E}$ be an optimal reconfiguration of N and let $f^{M^*}_{\text{opt}}$ be an arbitrary load-optimization flow for the reconfigured network $N(M^*)$. Let f_{old} be the original flow serving D in N. There must be a leaf node $v_i \in V$ such that a static (directed) link, W.L.O.G., denoted by (v_i,c), has $L(f_{\text{old}}(v_i,c)) = L_{\text{min-max}}(N(\emptyset))$. We know that v_i must be included in a configured link, denoted by $\{v_i,v_j\} \in M^*$, otherwise we still have that $L_{\text{min-max}}(N(M^*)) = L_{\text{min-max}}(N(\emptyset))$ holds. In the triangle $\{v_i,v_j,c\}$, there are at most two link-disjoint directed paths from another node in $\{v_j,c\}$ to the node v_i. We know the size of the flow on the static link (v_i,c) can be at most decreased by half, which implies

$$L_{\text{min-max}}\left(N(M^*)\right) \ge L\left(f^{M^*}_{\text{opt}}(v_i,c)\right) \ge L(f_{\text{old}}(v_i,c))/2 \,.$$

Thus, for any reconfiguration M of N, we know $L_{\text{min-max}}(N(M)) \ge L_{\text{min-max}}(N(\emptyset))/2$. □

Competitivity of matching algorithms. We next investigate the theoretical performance of a maximum matching algorithm, as e.g. utilized in [65]. The heuristic idea based on a maximum matching is that for each reconfigurable link $\{u,v\} \in \mathcal{E}$, we send all flows of demands $D(u,v)$ and $D(v,u)$ on links (u,v) and (v,u) respectively, then to find a maximum matching to maximize total size of flows on a set of configured links M. As it turns out, such an optimization might yield nearly no benefit, even though an optimal algorithm could hit the theoretical lower bound provided in Lemma 4.8.

Lemma 4.9. *For a τ-reconfiguration problem on a hybrid switch network N, where $\tau \in \{US, SS\}$, a maximum matching algorithm can find a reconfiguration M of N s.t. $L_{\text{min-max}}(N(\emptyset)) - L_{\text{min-max}}(N(M)) = \epsilon$ for an arbitrarily small $\epsilon \ge 0$, but an optimal reconfiguration M^* implies $L_{\text{min-max}}(N(M^*)) = L_{\text{min-max}}(N(\emptyset))/2$.*

Proof. Recall the definition of segregated routing. Given a small value $\epsilon \ge 0$, we construct a hybrid switch network $N = (V,E,\mathcal{E},C)$, where $V = \{v_1,\ldots,v_n,a,b,c,d\}$, c is the center, and other nodes are leaves. For any two nodes $u,v \in V \setminus \{c\}$, we construct a reconfigurable link $\{u,v\} \in \mathcal{E}$. Here, $\forall e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} : C(e) = 1$. Regarding demands D, for each $v_i \in \{v_1,...,v_n\}$, we define $D(v_i,a) = \epsilon$ and $D(b,a) = n\epsilon$, $D(b,d) = n\epsilon$. Clearly, $L_{\text{min-max}}(N(\emptyset)) = 2n\epsilon$. For the maximum matching algorithm, two reconfigurable links $\{b,d\} \in \mathcal{E}$ and $\{v_i,a\} \in \mathcal{E}$, where $i \in \{1,...,n\}$, must be included in a reconfiguration M, which gives $L_{\text{min-max}}(N(M)) = (2n-1)\epsilon$. However, by selecting $\{a,b\}$ into M^*, we can have $L_{\text{min-max}}(N(M^*)) = n\epsilon = L_{\text{min-max}}(N(\emptyset))/2$. Please note that for the above example, the splittable and unsplitable models show the same results. □

Extension to smaller reconfigurable circuit switches. In case the number of ports of a single reconfigurable switch does not suffice for all nodes in the network, our algorithms also extend to the case of multiple smaller reconfigurable switches. We can connect subsets of the nodes to a reconfigurable switch each, e.g., grouped by historical data w.r.t. the traffic demands. Our hybrid switch algorithms in §4 then take this subset of possible reconfigurable links to work with and proceed as usual.

5 EVALUATIONS

In order to study the performance of our algorithms under realistic workloads, we conducted extensive experiments with a simulator, which we will release together with this paper (as open source code). In particular, we benchmark our hybrid switch algorithms against several state of the art maximum matching and greedy baselines, considering a spectrum of packet traces on hybrid switch topologies as in Fig. 2. We first describe our methodology in §5.1 and then discuss our results in §5.2.

5.1 Methodology

Baselines. We consider the following baselines and implemented the corresponding algorithms for comparison. First, we compare our *h*ybrid *s*witch *n*etwork algorithms (denoted by *HSN-US/SN*) with a *Maximum Weight Matching* algorithm as a baseline, where routing occurs either on direct reconfigurable links or via the central packet switch. The matching algorithm is employed, e.g., by [24, 65] and is also optimal with respect to the average weighted path length [27] in such a routing model.[6]

Second, we also compare to a *Greedy* approach used by, e.g., Halperin et al. [33] and Zheng et al. [73]. For the link e that currently has the highest load, we check for the largest flow that can be rerouted on a direct connection, and offload it from the electrically switched network parts. This process is iterated until the load cannot be reduced further, where different links e can be chosen in each iteration.

Lastly, we additionally plot the maximum load on the network before any reconfiguration was applied (labelled as *Oblivious*).

Traffic Workloads. It is known that traffic traces in different networks and running different applications can differ significantly [7, 13, 30, 39, 54]. Thus, we collected a number of real-world and synthetic datasets from which we generate traffic matrices to evaluate and compare the performance of our algorithms. In particular:

[6]Note that a maximum matching algorithm is not optimal regarding path lengths in all topologies. However, when the distances between all nodes are identical in the static network part, a standard maximum matching approach is optimal in hybrid switch networks w.r.t. weighted path length [27].

- **Datacenter traces:** We consider two datacenter workloads, based on traces made available by Facebook [23, 54, 72], namely from the Hadoop and the Database clusters.
- **HPC traces:** We further consider a high performance computing workload, obtained from the CESAR backbone [2].
- **Synthetic traces:** The synthetic pFabric traces are frequently considered as benchmarks in scientific evaluations [6]. In a nutshell, workloads arrive according to a Poisson process, are embedded in a datacenter context, and follow a random communication pattern between subsets of nodes. In order to generate traffic traces and produce demand matrices, we use the NS2 simulation script we obtained from the authors of the paper, using the parameter $p = 0.5$.

Experimental Setup. All considered topologies, ranging from 40 to 3000 nodes[7], employ hybrid switch networks as in Fig. 2b. We repeat each setting by running it 5 times and display the averaged results, normalizing the workload in the static topology. For the runtime, we plot the average values in seconds, but also display minimum and maximum values by shading.

Our simulations were run on a machine with two Intel Xeons E5-2697V3 SR1XF with 2.6 GHz, 14 cores[8] each and a total of 128 GB RAM. The host machine was running Ubuntu 18.04.3 LTS.

We implemented the algorithms in Python (3.7.3) leveraging the NetworkX library (2.3). For the implementation of the maximum matching algorithm we used the algorithm provided by NetworkX.

5.2 Results and Discussion

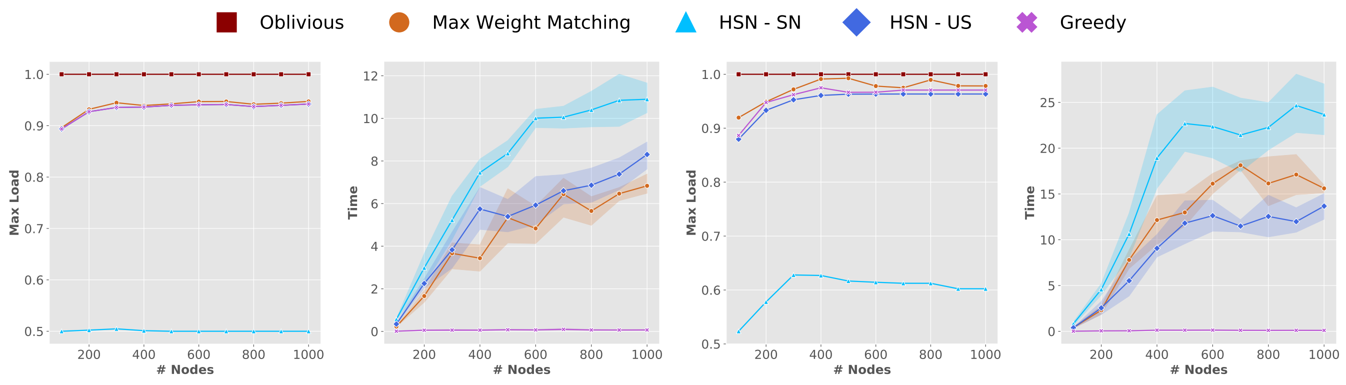
We report on the main results obtained in our simulations based on the different data sets. Figs. 3 and 5 summarize our evaluation results in terms of load and runtime for the Facebook traces; Fig. 4 shows the corresponding results for the HPC and pFabric traces.

Potential for Load Optimization. All algorithms significantly improve the load over the Oblivious baseline and provide relatively stable benefits throughout all scenarios investigated. Among these algorithms, the HSN algorithms typically clearly outperform the others.

More specifically, for the database (Figs. 3a, 5a) clusters, the reduction in the maximum load provided by the HSN-SN algorithm is almost a factor of two throughout the spectrum. For the Hadoop clusters (Fig. 3c, 5b), the performance of HSN-SN slightly decreases, but still achieves $\approx$ 60% of the original Oblivious load up until a network size of 1000 and then stays stable at $\approx$ 70% beyond. The three remaining algorithms (Greedy, Max. Weight Matching, and our HSN-US) achieve nearly identical values, with Greedy and HSN-US being slightly better. Above 1000 nodes, we can observe that their capability to further reduce the load seems to be quite restricted. Notwithstanding, they always perform significantly worse than HSN-SN, resulting in a comparatively load-increase of $\approx$ 60%.

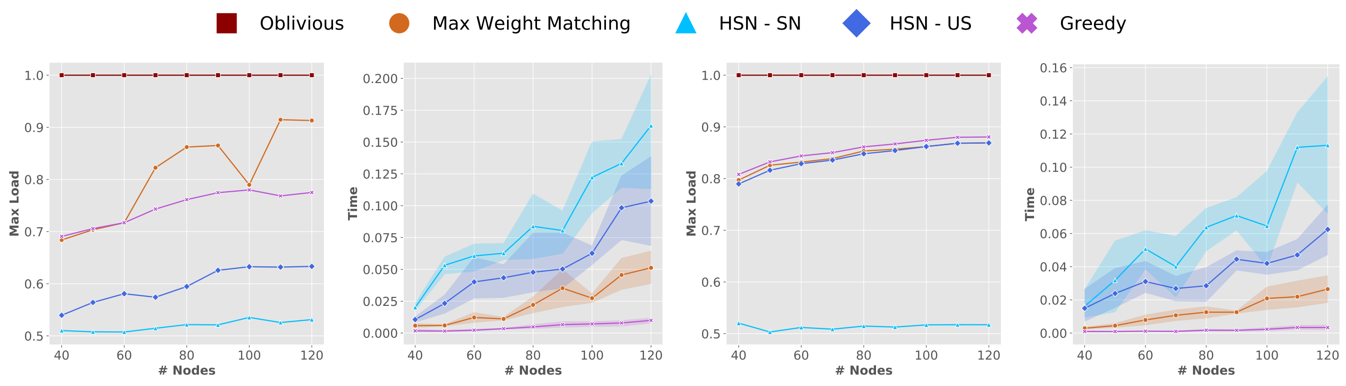
Regarding the HPC traces, we can observe similar results as in the Database Cluster, in terms of maximum load reduction. Also for the pFabric traces, our HSN-US algorithm achieves a lower maximum load compared to the Greedy or Max. Weight Matching. Here, the variance is slightly higher than in the other experiments; this matches empirical observations on the complexity of the traces produced by these synthetic traces [7].

[7]See Alistarh et al. [5] w.r.t. the feasibility of 1000 port optical switches in data centers.
[8]However, each algorithm only utilized a single core.



(a) Database Cluster: Max Load (b) Database Cluster: Runtime (c) Hadoop Cluster: Max Load (d) Hadoop Cluster: Runtime

Figure 3: Algorithmic comparison of the maximum load and runtime for different Facebook clusters



(a) pFabric: Max Load (b) pFabric: Runtime (c) HPC: Max Load (d) HPC: Runtime

Figure 4: Algorithmic comparison of the maximum load and runtime for pFabric and high-performance computing traces

In regard to the maximum load reduction, we conclude that our HSN-SN algorithm is quite stable w.r.t. to the number of nodes in the network. In contrast to that, Max. Weight Matching and the Greedy algorithm asymptotically approach the maximum load of the unconfigured network.

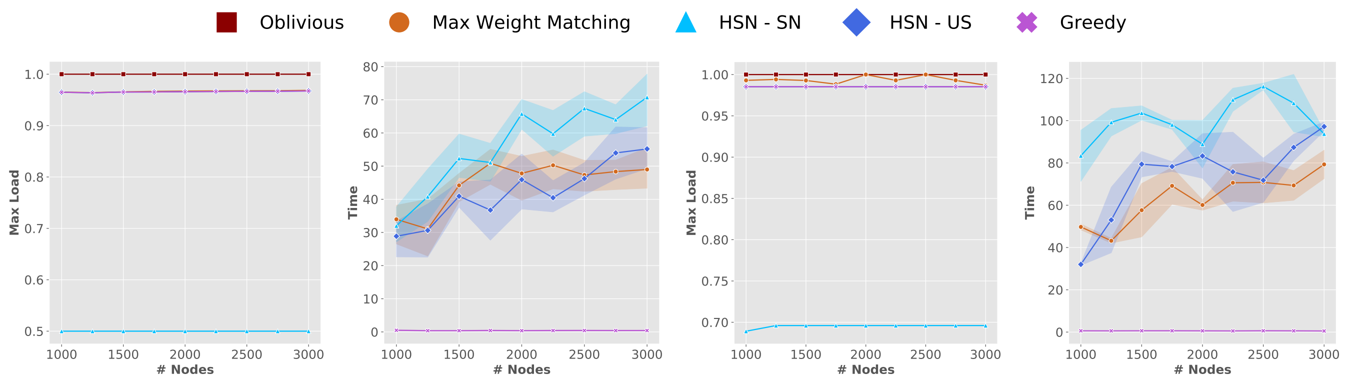
Runtime Performance. The best runtime is generally achieved by the Greedy algorithm, due to its early termination when no link can be added anymore. Our experiments show that in the case of the Greedy algorithm, this is unfortunately happening very early on. Regarding the runtime of the the Max. Weight Matching, we want to emphasize that the algorithm is unaware of the underlying problem of reducing the maximum link load. Therefore, a lot of runtime is actually wasted without achieving any further load reduction. In comparison to Max. Weight Matching, our HSN-US has a similar runtime, while spending all of it searching for the best load reduction matching.

HSN-US is consistently faster than HSN-SN, and the latter features quite a high variance in runtime. Notwithstanding, HSN-US has the benefit of only routing along single paths, which can be beneficial for performance metrics beyond load [55, 70]. On the other hand, such issues can also be alleviated with specialised multipath procotols [20, 53, 66].

Summary. While all algorithms provide load reductions, the extent of these optimizations and the required runtime differ significantly. Our results suggest that the load optimizations provided by HSN-US might prove beneficial over other segregated routing strategies, particularly because of its low runtime which is comparable to that of the Max. Weight Matching. We conclude that when considering both potential load reduction and runtime, HSN-SN provides a better tradeoff than HSN-US.

6 RELATED WORK

Most related work on flow routing in data center networks focuses on non-reconfigurable topologies [50]. That said, many recent works design and evaluate reconfigurable topologies e.g., [16, 24, 30, 33, 42, 43, 46, 47, 52, 63–65, 68, 69], often showing significant performance gains over static topologies and proving real-world viability. However, the algorithmic complexity of reconfigurable data center



(a) Database Cluster: Max Load (b) Database Cluster: Runtime (c) Hadoop Cluster: Max Load (d) Hadoop Cluster: Runtime

Figure 5: Algorithmic comparison of the maximum load and runtime for different larger Facebook clusters.

networks is mostly unstudied [28], and many fundamental questions remain open [11].

Scheduling traffic matrices with specific skew were investigated in [43, 44, 52, 63], but performance guarantees were only obtained by Venkatakrishnan et al. [63] due to leveraging submodularity, a condition that does not hold in our setting. Similarly, Avin et al. [8–10] investigate traffic matrices with low entropy, but they require scalable constant reconfigurable degrees and are oblivious to hybrid networks, and thus do not translate to the herein considered model.

The idea of leveraging good connectivity in data center contexts arose from utilizing random graphs [58], later extended into deterministic versions [19, 40, 60]. Xia et al. [68] used this idea to heuristically switch between random graphs and Clos topologies, depending on the traffic pattern, whereas Mellette et al. [47] incorporate it to improve their Rotornet [46] approach: If a flow cannot be delayed respectively be buffered, it gets sent along a short route. Both works of Mellette et al. also have the benefit that their reconfigurations are oblivious to the current traffic pattern, but hence also depend on the same for the resulting performance.

One of the notable works that does not rely on centralized computation is ProjecToR by Ghobadi et al. [30], which instead performs a distributed matching protocol reminiscent to the idea of stable matchings [1]. In their setting, they obtain a $(2+\varepsilon)$ approximation for the weighted latency objective, but do not consider load.

The algorithmic complexity of weighted latency was also considered in [26, 27], where already basic topologies and settings turned out to be intractable. On the other hand, finding a single shortest paths in partially reconfigured network can be done efficiently, and hence yields well performing heuristics [25]. Moreover some routing models can even be solved optimally. Notwithstanding, it is unclear how to transfer these results to a load-optimization setting: in topologies with unfavorable betweenness centrality, shortest path routing can overload popular links with high load.

Load-optimization in reconfigurable data centers was recently studied by Yang et al. [70], who investigated the impact of wireless interference on cross-layer optimization. Different wireless links are modeled as a conflict graph, where the task is to find sufficiently good independent (link) sets, in order to provide an interference-free reconfiguration. We see our work as orthogonal, as we only consider inherently interference-free technologies, and as thus it would be interesting to leverage their results in future work.

Another interesting line of work is by Zheng et al. [73], who study how to enhance the design of Diamond, BCube and VL2 network topologies with small reconfigurable switches, inspired by Flat-Tree [68]. They target maximum link load as well, and present intractability results on general graphs, although these results do not transfer to specific data center topologies or trees, respectively. Different routing models are not analyzed. Moreover, they propose to reconfigure the network with a greedy algorithm, which however does not come with formal performance guarantees. In evaluations of small network sizes, their combination of greedy algorithm and enhanced network design reduces the maximum load by 12% in average. We see similar greedy algorithm behavior in our evaluations, where however the greedy algorithm performance decreases to just a few percent of load improvement as network size grows.

That being said, even though our work is mostly motivated by technologies emerging in data center networks, it also applies to other reconfigurable technologies, as long as they fulfill our model properties. Fundamentally different however are reconfigurable optical wide-area networks, as therein the fiber connectivity is fixed. Hence capacities can be adjusted and alternative failover paths provided, leading to improvements for the scheduling of bulk-transfers [18, 36, 37, 45] and reliability concerns [32, 57].

7 CONCLUSION

We investigated load minimization in reconfigurable hybrid networks, leveraging the flexibility of emerging programmable physical layers. To this end we investigated the underlying problem complexity, unveiling that already tree topologies of small height induce intractability for a multitude of routing models, and that one cannot hope for general approximability via submodularity techniques. Notwithstanding, we showed that hybrid switch networks, and in turn non-blocking data center interconnects, can be optimized efficiently. Trace-driven simulations show that our hybrid switch algorithms significantly outperform a state of the art maximum matching baseline, but also greedy algorithms.

REFERENCES

[1] 2012. https://www.nobelprize.org/prizes/economic-sciences/2012/summary/.
[2] 2016. Characterization of the DOE Mini-apps. portal.nersc.gov/project/CAL/doe-miniapps.htm.
[3] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. *Network flows - theory, algorithms and applications.* Prentice Hall.
[4] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *SIGCOMM*.
[5] Dan Alistarh et al. 2015. A High-Radix, Low-Latency Optical Switch for Data Centers. *Comput. Commun. Rev.* 45, 5 (2015), 367–368.
[6] Mohammad Alizadeh et al. 2013. pFabric: minimal near-optimal datacenter transport. In *SIGCOMM*.
[7] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the Complexity of Traffic Traces and Implications. In *Proc. ACM SIGMETRICS*.
[8] Chen Avin, Alexandr Hercules, Andreas Loukas, and Stefan Schmid. 2018. *rDAN*: Toward robust demand-aware network designs. *Inf. Process. Lett.* 133 (2018).
[9] Chen Avin, Kaushik Mondal, and Stefan Schmid. 2017. Demand-Aware Network Designs of Bounded Degree. In *DISC*.
[10] Chen Avin, Kaushik Mondal, and Stefan Schmid. 2019. Demand-Aware Network Design with Minimal Congestion and Route Lengths. In *INFOCOM*. IEEE.
[11] Chen Avin and Stefan Schmid. 2018. Toward demand-aware networking: a theory for self-adjusting networks. *Comput. Commun. Rev.* 48, 5 (2018), 31–40.
[12] Navid Hamed Azimi et al. 2014. FireFly: a reconfigurable wireless data center fabric using free-space optics. In *SIGCOMM*.
[13] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proc. IMC*.
[14] Calient. 2018. Edge 640 Optical Circuit Switch. https://www.calient.net/products/edge640-optical-circuit-switch/.
[15] Jiaxin Cao et al. 2013. Per-packet load-balanced, low-latency routing for clos-based data center networks. In *CoNEXT*. ACM.
[16] Kai Chen et al. 2014. OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility. *Trans. Netw.* 22, 2 (2014), 498–511.
[17] Charles Clos. 1953. A Study of Non-Blocking Switching Networks. *Bell System Technical Journal* 32, 2 (1953), 406–424.
[18] Michael Dinitz and Benjamin Moseley. 2020. Scheduling for Weighted Flow and Completion Times in Reconfigurable Networks. In *INFOCOM*.
[19] Michael Dinitz, Michael Schapira, and Asaf Valadarsky. 2017. Explicit Expanding Expanders. *Algorithmica* 78, 4 (2017), 1225–1245.
[20] Advait Abhay Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. 2013. On the impact of packet spraying in data center networks. In *INFOCOM*.
[21] Jack Edmonds. 1965. Paths, Trees and Flowers. *Canad. J. Math* 17 (1965), 449–467.
[22] Tom Empson and Scarlet Schwiderski-Grosche. 2019. Optics for the Cloud Research Alliance establishes collaborative research approach to improving cloud technology. https://www.microsoft.com/en-us/research/blog/optics-for-the-cloud-research-alliance-establishes-collaborative-research-approach-to-improving-cloud-technology/.
[23] facebook. 2018. Facebook Network Analytics Data Sharing. https://www.facebook.com/groups/1144031739005495/.
[24] Nathan Farrington et al. 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *SIGCOMM*.
[25] Thomas Fenz, Klaus-T. Foerster, Stefan Schmid, and Anaïs Villedieu. 2019. Efficient Non-Segregated Routing for Reconfigurable Demand-Aware Networks. In *Networking*. IEEE.
[26] Klaus-T. Foerster et al. 2019. On the Complexity of Non-Segregated Routing in Reconfigurable Data Center Architectures. *Comput. Comm. Rev.* 49, 2 (2019), 2–8.
[27] Klaus-T. Foerster, Manya Ghobadi, and Stefan Schmid. 2018. Characterizing the algorithmic complexity of reconfigurable data center architectures. In *ANCS*.
[28] Klaus-T. Foerster and Stefan Schmid. 2019. Survey of Reconfigurable Data Center Networks: Enablers, Algorithms, Complexity. *SIGACT News* 50, 2 (2019), 62–79.
[29] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, USA.
[30] Monia Ghobadi et al. 2016. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *SIGCOMM*.
[31] Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab Mirrokni. 2009. Approximating Submodular Functions Everywhere. In *Proc. SODA*.
[32] Jennifer Gossels, Gagan Choudhury, and Jennifer Rexford. 2019. Robust network design for IP/optical backbones. *J. Opt. Commun. Netw.* 11, 8 (Aug 2019), 478–490.
[33] Daniel Halperin et al. 2011. Augmenting data center networks with multi-gigabit wireless links. In *SIGCOMM*.
[34] Abdelbaset S. Hamza et al. 2016. Wireless Communication in Data Centers: A Survey. *IEEE Commun. Surv. Tut.* 18, 3 (2016), 1572–1595.
[35] Kai Han, Zhiming Hu, Jun Luo, and Liu Xiang. 2015. RUSH: Routing and scheduling for hybrid data center networks. In *INFOCOM*. IEEE, 415–423.
[36] Su Jia et al. 2017. Competitive Analysis for Online Scheduling in Software-Defined Optical WAN. In *Proc. IEEE INFOCOM*.
[37] Xin Jin et al. 2016. Optimizing Bulk Transfers with Software-Defined Optical WAN. In *SIGCOMM*.
[38] Christoforos Kachris and Ioannis Tomkos. 2012. A Survey on Optical Interconnects for Data Centers. *IEEE Comm. Surv. Tut.* 14, 4 (2012), 1021–1036.
[39] Srikanth Kandula et al. 2009. The nature of data center traffic: measurements & analysis. In *IMC*.
[40] Simon Kassing et al. 2017. Beyond fat-trees without antennae, mirrors, and disco-balls. In *SIGCOMM*.
[41] Charles E. Leiserson. 1985. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Trans. Computers* 34, 10 (1985), 892–901.
[42] He Liu et al. 2014. Circuit Switching Under the Radar with REACToR.. In *NSDI*. USENIX.
[43] He Liu et al. 2015. Scheduling techniques for hybrid circuit/packet networks. In *CoNEXT*. ACM, 41:1–41:13.
[44] Ariel Livshits and Shay Vargaftik. 2018. LUMOS: A Fast and Efficient Optical Circuit Switch Scheduling Technique. *IEEE Comm. Lett.* 22, 10 (2018), 2028–2031.
[45] Long Luo et al. 2019. DaRTree: Deadline-aware Multicast Transfers in Reconfigurable Wide-Area Networks. In *IWQoS 2019*.
[46] William M. Mellette et al. 2017. RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. In *SIGCOMM*. ACM.
[47] William M. Mellette et al. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *NSDI*.
[48] William M. Mellette, Alex C. Snoeren, and George Porter. 2016. P-FatTree: A multi-channel datacenter network topology. In *HotNets*. ACM, 78–84.
[49] Cisco Visual Networking. 2016. Cisco global cloud index: Forecast and methodology, 2015-2020. white paper. *Cisco Public, San Jose* (2016).
[50] Mohammad Noormohammadpour and Cauligi S. Raghavendra. 2018. Datacenter Traffic Control: Understanding Techniques and Tradeoffs. *IEEE Communications Surveys and Tutorials* 20, 2 (2018), 1492–1525.
[51] Polatis. 2019. Series 6000n Network Optical Matrix Switch. https://www.hubersuhner.com/en/documents-repository/technologies/pdf/data-sheets-optical-switches/polatis-series-6000n.
[52] George Porter et al. 2013. Integrating Microsecond Circuit Switching into the Data Center. (2013).
[53] Costin Raiciu et al. 2011. Improving datacenter performance and robustness with multipath TCP. In *SIGCOMM*.
[54] Arjun Roy et al. 2015. Inside the social network's (datacenter) network. In *Comput. Commun. Rev.*, Vol. 45.
[55] Siddhartha Sen, David Shue, Sunghwan Ihm, and Michael J. Freedman. 2013. Scalable, optimal flow routing in datacenters via local link balancing. In *CoNEXT*.
[56] Arjun Singh et al. 2016. Jupiter rising: a decade of clos topologies and centralized control in Google's datacenter network. *Commun. ACM* 59, 9 (2016), 88–97.
[57] Rachee Singh, Manya Ghobadi, Klaus-T. Foerster, Mark Filer, and Phillipa Gill. 2018. RADWAN: Rate Adaptive Wide Area Network. In *SIGCOMM*. ACM.
[58] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. 2012. Jellyfish: Networking Data Centers Randomly. In *NSDI*. USENIX.
[59] Ashwin Sridharan, Roch Guérin, and Christophe Diot. 2005. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Trans. Netw.* 13, 2 (2005), 234–247.
[60] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. 2016. Xpander: Towards Optimal-Performance Datacenters. In *CoNEXT*.
[61] Vijay V. Vazirani. 2001. *Approximation Algorithms.* Springer, Berlin, Heidelberg.
[62] Shaileshh Bojja Venkatakrishnan et al. 2016. Costly Circuits, Submodular Schedules and Approximate Carathéodory Theorems. In *SIGMETRICS*.
[63] Shaileshh Bojja Venkatakrishnan et al. 2018. Costly circuits, submodular schedules and approximate Carathéodory Theorems. *Queueing Syst.* 88, 3-4 (2018), 311–347.
[64] Guohui Wang et al. 2009. Your Data Center Is a Router: The Case for Reconfigurable Optical Circuit Switched Paths. In *HotNets*.
[65] Guohui Wang et al. 2010. c-Through: part-time optics in data centers. In *SIGCOMM*.
[66] Damon Wischik et al. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *NSDI*. USENIX Association.
[67] Wenfeng Xia, Peng Zhao, Yonggang Wen, and Haiyong Xie. 2017. A Survey on Data Center Networking (DCN): Infrastructure and Operations. *IEEE Communications Surveys and Tutorials* 19, 1 (2017), 640–656.
[68] Yiting Xia et al. 2017. A Tale of Two Topologies: Exploring Convertible Data Center Network Architectures with Flat-tree. In *SIGCOMM*.
[69] Yiting Xia, T. S. Eugene Ng, and Xiaoye Steven Sun. 2015. Blast: Accelerating high-performance data analytics applications by optical multicast. In *INFOCOM*.
[70] Zhenjie Yang et al. 2019. Achieving Efficient Routing in Reconfigurable DCNs. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 3, Article Article 47 (Dec. 2019).
[71] Xin Yuan. 2011. On Nonblocking Folded-Clos Networks in Computer Communication Environments. In *IPDPS*.
[72] James Hongyi Zeng. 2017. Data Sharing on traffic pattern inside Facebook's datacenter network. https://research.fb.com/data-sharing-on-traffic-pattern-inside-facebooks-datacenter-network/.
[73] Jiaqi Zheng et al. 2019. Dynamic Load Balancing in Hybrid Switching Data Center Networks with Converters. In *ICPP*.

A DEFERRED PROOFS AND ALGORITHMS

A.1 Unsplittable Non-Segregated Routing

We start with the weakly NP-hard case of $h=1$ in Theorem A.2, followed by the strongly NP-hard case of $h=2$ in Theorem A.3.

The following proof will make use of the weakly NP-hard 2-Partition problem, which is defined as follows:

Definition A.1 (2-Partition[29]). Given a set of n integers $S=\{s_1,...,s_n\}$ where $B=\sum_{s_i\in S}s_i$, can we divide S into two disjoint subsets S_1 and S_2 such that $\sum_{s_i\in S_1}s_i=\sum_{s_j\in S_2}s_j$?

Theorem A.2. *The UN-load-optimization reconfiguration problem is weakly NP-hard when the given hybrid network is a hybrid switch network, i.e., a tree of height $h=1$.*

Proof. We give a reduction from the 2-partition problem. Our proof is conceptually similar to the one by Yang et al. [70, Theorem 1], but also applies to the hybrid switch model. For the UN-reconfiguration problem, we consider a tree of the height one that has the root node c; for each $s_i\in S$, there is a node $a_i\in A$ connected to c by a static (bidirected) link in E, while c has two additional adjacent nodes r and b. This construction constitutes a hybrid switch network since all nodes only connect to c, where we only allow reconfigurable links between leaf nodes (but not with c).

For each $a_i\in A$, we set $D(r,a_i)=s_i$. Without loss of generality, let n be an even number. For each odd number i, where $1\le i\le n$, we define $D(a_i,a_{i+1})=B/2$. For capacity, it has $\forall e\in\overrightarrow{E}\cup\overrightarrow{\mathcal{E}}:C(e)=1$. Now, the question is how to make links having a load value no more than $B/2$. According to our demands, only one direction in each bidirected link needs to carry traffics.

For each odd number i, we need to configure $\{a_i,a_{i+1}\}\in\mathcal{E}$, which gives a matching of $n/2$ bidirected links. Now, we have to reconfigure the (bidirected) link $\{r,b\}\in\mathcal{E}$ and decide which nodes in A have their flows going through the (directed) link $(r,b)\in\overrightarrow{\mathcal{E}}$ s.t. no (directed) link has load more than $B/2$ in the UN model, which implies a solution to the 2-Partition problem and vice versa. □

Theorem A.3. *The UN-load-optimization reconfiguration problem is strongly NP-hard when the given hybrid network, before reconfiguration, is a tree of the height $h\ge 2$.*

Proof. We give a reduction from the 3-partition problem, which is strongly NP-hard. Our construction extends the construction in the proof of Theorem 3.2 by defining more demands without changing the topology. The tree structure and demands in the proof of Theorem 3.2 are directly copied. Now, we define some additional demands as follows: for each node $s_i\in S$, which is a direct child of the root r connected by $\{r,s_i\}\in E$, we define a demand $D(r,s_i)=(m-1)B-(m-1)*s(a_i)$. Regarding reconfigurable links $\mathcal{E}$, if two nodes u and v are not connected by a static link in T, then there is a reconfigurable (bidirected) link $\{u,v\}\in\mathcal{E}$. Each directed link in $\overrightarrow{E}\cup\overrightarrow{\mathcal{E}}$ has the capacity of one. We claim that no (directed) link in $\overrightarrow{E}\cup\overrightarrow{\mathcal{E}}$ has load more than $(m-1)B$ if and only if there exists a valid 3-Partition for the set A.

Note that, in our setting, only one direction (directed link) of each bidirected link need to carry flow.

Assume A has a valid 3-Partition $A_1,...,A_m$. For each $A_i=\{a_j,a_k,a_f\}$, where $1\le i\le m$, we connect $s_j,s_k,s_f\in S$ to the corresponding nodes f^i_j,f^i_k,f^i_f in the subtree T_i respectively by adding configured links. Thus, for each r_i, where $1\le i\le m$, the flow size conveyed by the static link (r_i,r) is decreased by B, which is $(m-1)B$. For each static (directed) link (r,s_j), where $1\le j\le 3m$, it has the load value exactly $(m-1)B$.

On the other hand, we assume that a set of configured links M exists such that no static link, e.g., (r,s_i) for $1\le i\le m$, can have a load more than $(m-1)B$. Clearly, each node s_i must be included in a configured link s.t., at least a flow of size $s(a_i)$ arrives at s_i via a configured link, otherwise (r,s_i) is overflowed. Each configured (directed) link $\left(f^k_i,s_i\right)$, where $1\le i\le 3m$ and $1\le k\le m$ can only carry flow for the demand $D\left(f^k_i,s_i\right)$; otherwise there must be some flows using static links (r,s_i) to arrive at destination $s_q\in S$, where $i\ne q$ and $1\le q\le 3m$; this causes the load on (r,s_i) more than $(m-1)B$. Thus, a similar argument like the segregated model can be given, and it implies a valid 3-Partition $A_1,...,A_m$. □

A.2 Splittable Non-Segregated Routing

It remains to cover intractability for the last remaining routing model:

Theorem A.4. *The SN-reconfiguration problem is strongly NP-hard when the given hybrid network N, before reconfiguration, is a tree of height $h\ge 2$.*

Proof. Given an instance of 3-Partition (A,B,s), we construct an instance of the SN-reconfiguration problem as follows: the constructed tree T has the node r as its root, and r has a directed child node r_0. For each element $a_i\in A$, r_0 has a direct child $s_i\in S$. The root r also has m subtrees T_i for $1\le i\le m$. For each subtree T_i, its root is the node r_i, which is the direct child of r; and the root r_i has 3 child nodes $Q^i=\left\{q^i_1,q^i_2,q^i_3\right\}$. The tree T constitutes the static (bidirected) links E and nodes V of the hybrid network N. To construct the set of all reconfigurable (bidirected) links $\mathcal{E}$, for each $1\le i\le m$, there is a reconfigurable (bidirected) link $\left\{q^i_k,s_j\right\}\in\mathcal{E}$, where $k\in\{1,2,3\}$ and $s_j\in S$. Without loss of generality, we set $\forall e\in\overrightarrow{E}\cup\overrightarrow{\mathcal{E}}:C(e)=1$. Regarding demands D, for each $0\le i\le m$, we have $D(r_i,r)=B$, and for each $a_i\in A$, we have $D(s_i,r_0)=B-s(a_i)$. For each $1\le i\le m$, $D(r_i,r_0)=B$ and $D\left(r_i,q^i_k\right)=B/2+\epsilon$, where $k\in\{1,2,3\}$ and $\epsilon<B/2-\max\{a:a\in A\}$. Clearly, the constructed tree only has a height of two. We claim that after N being reconfigured, there is no (directed) link that has the load more than B if and only if there exists a valid 3-Partition for the set A.

We note that, by our setting, each bidirected link in $E\cup\mathcal{E}$ only need to carry flow in one direction.

If A has a valid 3-Partition $A_1,...,A_m$, for each $A_i=\{a_j,a_k,a_f\}$, where $1\le i\le m$, we connect $s_j,s_k,s_f\in S$ to the corresponding nodes q^i_1,q^i_2,q^i_3 in the subtree T_i respectively by adding three configured (bidirected) links, and then we send three flows of sizes $s(a_j)$, $s(a_k)$ and $s(a_f)$ from r_i to r_0 through the configured (directed) links $\left(q^i_1,s_j\right)$, $\left(q^i_2,a_k\right)$ and $\left(q^i_3,s_f\right)$ respectively. For other demands, we send them on their own static links respectively. Clearly, all demands are served but no link in $\overrightarrow{E}\cup\overrightarrow{M}$ has a load more than B.

Conversely, assume that we have an optimal reconfiguration $M\subseteq\mathcal{E}$ for the SN-reconfiguration problem and a load-optimization flow f for $N(M)$. Without loss of generality, for each $D(r_i,r)$, where $i\in$

$\{0,...,m\}$, we assume that the corresponding flow is only sent on the static (directed) link (r_i,r) in f. If not, some flows for $D(r_i,r)=B$ must also go through (r_j,r), where $j\in\{0,...,m\}$ and $r_j\neq r_i$. Since $D(r_j,r)=B$, to make $L(f(r_j,r))\leq B$, we know flows serving $D(r_j,r)=B$ must go through (r_i,r) too. Therefore, we can cancel the alternative path for each $D(r_i,r)$, where $i\in\{0,...,m\}$, to force each demand $D(r_i,r)$ only being sent on (r_i,r) without increasing the load value of any (directed) link. For each subtree T_i, where $i\in\{1,...,m\}$, we know flows serving $D\left(r_i,q_j^i\right)$, where $j\in\{1,2,3\}$, must be only sent on $\left(r_i,q_j^i\right)$ due to our assumption. Thus, for each T_i, there must be three configured (directed) links from its three leaf nodes to three nodes in S, otherwise, one static (directed) link $\left(r_i,q_j^i\right)$, where $j\in\{1,2,3\}$, must overflow after serving $D(r_i,r_0)=B$. To serve each $D(s_i,r_0)=B-s(a_i)$, where $s_i\in S$ and $a_i\in A$, the link (s_i,r_0) already carries a flow of size $B-s(a_i)$, and then each (s_i,r_0) can only convey a flow of size $s(a_i)$ for some demands $D(r_j,r_0)$, where $j\in\{1,...,m\}$. Therefore, for each subtree T_i, we need to make three configured links to generate three (directed) paths: $\left(r_i,q_1^i,s_j,r_0\right)$, $\left(r_i,q_2^i,s_k,r_0\right)$ and $\left(r_i,q_3^i,s_f,r_0\right)$, where $s_k,s_j,s_f\in S$, to convey B flows for $D(r_i,r_0)$, which implies $s(a_j)+s(a_k)+s\left(a_f\right)=B$. Finally, each subtree T_i has three configured (bidirected) links connecting three nodes in S, which indicates a valid 3-Partition $A_1,...,A_m$. □

A.3 Proof of Lemma 4.3 for $\tau=$SN

Lemma A.5. *Given a reconfigured network $N(M)$, which is a triangle on nodes $V=\{a,b,c\}$ with the only configured link $\{a,b\}\in\mathcal{E}$, then for demands D, a load-optimization flow f_{opt} in $N(M)$ can be computed in constant time by Algorithm 5 when $\tau=SN$.*

Proof. When $\tau=$SN, any two distinct demands in D in the triangle $\{a,b,c\}$ are called related if they share the same source or sink. Let f be an arbitrary flow serving D under $\tau=$SN. For any two related demands, e.g., $D(a,b)$ and $D(a,c)$, W.L.O.G., we assume $D(a,b)$ sending a flow of size $\beta>0$ along (a,c,b) and $D(a,c)$ sending a flow of size $\alpha>0$ along (a,b,c) in f; and remaining of $D(a,b)$ and $D(a,c)$ are only sent on directed links (a,b) and (a,c) respectively in f. We call such a routing as interfering for these two related demands. W.L.O.G, we also assume $\beta\geq\alpha$. The interfering between $D(a,b)$ and $D(a,c)$ in f can be canceled by redirecting a flow of size α of $D(a,c)$ from its indirect path (a,b,c) to its shortcut (a,c), while forcing $D(a,b)$ only sending a flow of size $\beta-\alpha$ along (a,c,b). Clearly, the cancellation would not increase the maximum load of f. Thus, there must be a load-optimization flow f^* serving D such that no interfering occurs between any two related demands, otherwise we can do the interfering cancellation in f^*.

Now, we need to find the load-optimization flow f^*. Given a triangle N and demands, we will prove Algorithm 5 can find f^* in constant time. Clearly, Algorithm 5 terminates in constant time since the number of demands is at most 6. It is clear that the returned flow f_{opt} is an interfering-free flow since when a demand $D(u,v)$ is marked split, all its related demands are rejected for being further splitted. Given an upper-bound μ, our algorithm guarantees that all directed links have loads no more than μ. Now, we just need to prove that μ found in Algorithm 4 is minimum. We assume that $\mu'<\mu$ is actually the minimized maximum load. Each demand marked as split in Algorithm 5: $\forall D(u,v)\in D_S$ must send a flow of size $D(u,v)-\mu'C(u,v)$ to its indirect path, where $D(u,v)-\mu'C(u,v)>D(u,v)-\mu C(u,v)$, otherwise some links would have loads more than μ'. Due to the interfering-free requirement, each demand in $D\setminus D_S$ cannot send its flow to its indirect path. W.L.O.G, let $D(p,q)$ be the unsplit demand in $D\setminus D_S$, which has the maximum load μ in S_μ in Algorithm 4. Since the related demands of $D(p,q)$, which are marked as split, need to send more flows to their indirect paths, where (p,q) is included. Then the load on the link (p,q) will be larger than μ, which contradicts the assumption. □

Algorithm 4: Determine Optimal Load

```
Input  : A Triangle $N$ on three nodes $V$, demands $D$,
         a set of split demands $D_S$;
Output : a load-otimization flow $f_{\text{opt}}$ for $N$;
1  Define a set $S_\mu=\emptyset$ and a variable $\mu\in\mathbb{R}^+$;
2  for each unsplit demand $D(u,v)\in D\setminus D_S$ do
3      Let $D'\subseteq D_s$ be "split" demads related to $D(u,v)$;
4      $D(u,v)$ is only sent on $(u,v)$ in $f_{\text{opt}}$;
5      If $|D'|=2$, let $D'=\{D(u,d),D(d,v)\}$;
6      compute $\mu=\frac{D(u,v)+D(u,d)+D(d,v)}{C(u,v)+C(u,d)+C(d,v)}$ and $S_\mu=S_\mu\cup\{\mu\}$;
7      If $|D'|\leq 1$, similar computation of $\mu$;
8  $\mu=\max\{\mu':\mu'\in S_\mu\}$;
9  for each split demand $D(p,q)\in D_S$ do
10     $D(p,q)$ send a flow of size $\mu\cdot C(p,q)$on $(p,q)$ and remaning
        flow on $(p,k,q)$ in $f_{\text{opt}}$, where $k\in V$ and $k\notin\{q,p\}$;
11 return $f_{\text{opt}}$;
```

Algorithm 5: Triangle Optimization When $\tau=SN$

```
Input  : A Triangle $N=(V,E,\mathcal{E},C)$ with nodes $V=\{a,b,c\}$,
         demands $D$, and the configured link (bidirected) $\{a,b\}\in\mathcal{E}$;
Output : a load-otimization flow $f_{\text{opt}}$ for $N$;
1  Define a set $D_S=\emptyset$ and values $\forall D(u,v)\in D:\alpha_{uv}=\frac{D(u,v)}{C(u,v)}$;
2  Let any two demands $D(u,v)$ and $D(p,q)$, where
   $u,v,p,q\in V$, $u\neq v$, and $p\neq q$, be related if either $u=p$ or $v=q$;
3  Let $D(u,v)$ be the demand in $D$ with the highest value $\alpha_{uv}$;
4  if a demand $D(p,q)$ is related to $D(u,v)$ and $\alpha_{pq}=\alpha_{uv}$ then
5      return $f_{\text{opt}}=$Algorithm 4$(N,D,D_S)$;
6  Mark the demand $D(u,v)$ "split" and $D_S=D_S\cup\{D(u,v)\}$;
7  Set $D_1=\{D(u,v),D(u,d),D(d,v)\}$, where $d=V\setminus\{u,v\}$;
8  Let $D(p,q)$ be a demand in $D_2:=D\setminus D_1$ with the highest $\alpha_{pq}$;
9  if $\exists D(i,j)\in D_1$ is unsplit and has $\alpha_{ij}\geq\alpha_{pq}$ for $D(p,q)$ then
10     return $f_{\text{opt}}=$Algorithm 4$(N,D,D_S)$;
11 if $\exists D(i,j)\in D_2$ is related to $D(p,q)$ and has $\alpha_{ij}\geq\alpha_{pq}$ then
12     return $f_{\text{opt}}=$Algorithm 4$(N,D,D_S)$;
13 Mark the demand $D(p,q)$ "split" and $D_S=D_S\cup\{D(p,q)\}$;
14 Set $D_3=\{D(p,q),D(p,d),D(d,q)\}$, where $d=V\setminus\{q,p\}$;
15 if $D(f,g):=D_2\setminus D_3\neq\emptyset$, where $f,g\in V$ and $f\neq g$ then
16     if $\nexists D(i,j)\in D\setminus D(f,g)$ is unsplit and has $\alpha_{ij}\geq\alpha_{fg}$ then
17         Mark $D(f,g)$ "split" and $D_S=D_S\cup\{D(f,g)\}$;
18 return $f_{\text{opt}}=$Algorithm 4$(N,D,D_S)$;
```