Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures

Klaus-Tycho Foerster
University of Vienna, Austria
klaus-tycho.foerster@univie.ac.at

Manya Ghobadi
Microsoft Research
mgh@microsoft.com

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

ABSTRACT

Emerging data center architectures are becoming *reconfigurable*. While prior work has shown the practical benefits of reconfigurable topologies, the underlying algorithmic complexity is not yet well understood. In particular, most reconfigurable topologies are *hybrid*, where parts of the network are reconfigurable (consisting of optical or wireless devices) while other parts are static (consisting of electrical switches). Current proposals enforce a routing policy that routes flows on either part "exclusively" by labeling flows as mice or elephant. We show that such *artificial segregation* in routing policy results in non-optimal paths and argue for algorithms that route packets across the network seamlessly. In doing so, we present the first algorithmic study of reconfigurable network architectures and provide optimality and hardness proofs in terms of topology and routing policy. Our results show that classical matching algorithms, as used in prior work, are optimal only when the topology consists of *one* reconfigurable switch, and the routing policy is *enforced* to be segregated. In other words, if there is an option of routing flows seamlessly along reconfigurable and non-reconfigurable parts of the network, matching algorithms are not optimal. In fact, when the hybrid network is seen from a joint perspective, optimal routing is an NP-hard problem. We further show that optimally routing even two flows in a network with multiple reconfigurable switches is an NP-hard problem as well.

CCS CONCEPTS

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

ANCS '18, July 23–24, 2018, Ithaca, NY, USA

ACM ISBN 978-1-4503-5902-3/18/07…$15.00
https://doi.org/10.1145/3230718.3230722

1 INTRODUCTION

Data center networks are critical infrastructures of today's online services. In an era of increased engagement with technology on many fronts—health, business, science, and social life—the performance and reliability of data center networks play a critical role in their design [10, 11, 19, 36].

As a result, the *data center topology design problem*, that is, the problem of finding efficient data center topologies, has received much interest over the last decade [2, 7, 20–22, 28, 32, 38–40, 46]. Emerging technologies for reconfigurable data center topologies introduce an additional degree of freedom to the data center topology design problem [6, 8, 15, 18, 23, 24, 30, 41, 49]. These works show that reconfigurable topologies improve network performance, making them an attractive solution for future data centers.

One of the biggest challenges of reconfigurable topologies is *demand-awareness*, the problem of deciding how to program reconfigurable links and in turn route flows across the network. Despite much interest in the topic, there is no algorithmic modeling of these architectures. In fact, each prior work is tied to its own reconfiguration algorithm that is tailored to the technology used and to inherent assumptions about flow routing. As a result, the algorithms and the conclusions are not portable from paper to paper (and technology to technology).

In particular, most prior work assumes the network consists of two *segregated* parts: the non-reconfigurable part (interconnected by electrical switches) and the reconfigurable part (interconnected by optical or wireless devices) [8, 15, 27, 31, 45, 49]. Hence, a common way to solve the demand-awareness problem is to route the large flows along the reconfigurable part of the network and let the non-reconfigurable part handle the leftovers. In other words, the flows are partitioned into categories, *e.g.,* elephant and mice flows, and these form the foundation of routing policies and topology adaptations. As we go on to show, this segregated approach results in non-optimal network configurations. Furthermore, there is no complexity analysis for the practical case of *multiple* reconfigurable switches in the topology.

In this paper, we present a first characterization of the algorithmic complexity for emerging data center network architectures in which the static topology is enhanced by reconfigurable links provided by optical switches.

Our results are not restricted to specific communication patterns or network topologies. In particular, we show that:

- When the reconfigurable part consists of only one switch, and if flows are artificially restricted to be segregated between the non-reconfigurable and reconfigurable parts of the topology, classic matching algorithms, such as Edmond's (weighted) *Blossom* algorithm [14, 33], as used in Helios [15] and c-Through [45], are optimal (§3.1). However, if flows can be routed seamlessly along a mix of non-/reconfigurable links, finding the optimal routes becomes an NP-hard problem, even for a single optical switch (§3.2). We believe the segregated routing might be an artifact of circuit establishment time in prior work and encourage the community to consider the more practical approach of seamless routing.
- Surprisingly, NP-hardness strikes much earlier when multiple reconfigurable devices are present in the network, even for two flows (§4.2). A single flow can be optimally routed (§4.1), but the computation of such a routing requires techniques beyond simple matching algorithms. Even though having multiple reconfigurable devices is a necessary requirement for practical reasons, to the best of our knowledge, no prior work has characterized this case.
- While the main focus of our work is on the important case of optical circuit switches, some results generalize to other reconfigurable technologies, including hybrid data center architectures using *unidirectional* connections (§4.3).

We believe that our model and algorithmic study open new perspectives on an active research area, raising topics for future research, such as (*i*) the design of new heuristics for the practical case of multiple reconfigurable switches, or, (*ii*) the case of a single reconfigurable switch,when artificial segregation is removed from the routing policy.

2 MODEL

We study the problem of computing a data center topology to optimally serve a given communication pattern, where the topology combines static (fixed) and reconfigurable links.

Network model. Let $N = (V, E, S, w)$ be a weighted *hybrid* network [31, 44] connecting the nodes V (*e.g.,* top-of-the-rack switches), using 1) (usually electrical) static links E and 2) optical links implemented through reconfigurable optical circuit switches S.

An optical switch s_ℓ connects a set of nodes $V_\ell \subseteq V$ by choosing a matching M_ℓ on V_ℓ, where two matched nodes are connected by a bidirectional link; see Figure 1. For the sake of generality, we assume each link, whether electrical or optical, comes with a weight w (a cost, *e.g.,* latency).

Generality. 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 links, simply implying any appropriate technology that matches our model.

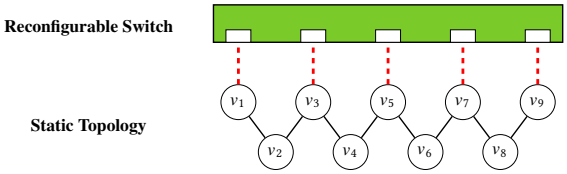

Figure 1: Line topology with nine nodes, where the five nodes, v_1, v_3, v_5, v_7, v_9, are connected to a reconfigurable switch (dashed): the choice of the matching inside the switch, *e.g.,* (v_3, v_5) and (v_1, v_9), depends on the current communication demands. If the routing policy is segregated, the flows of four nodes, v_2, v_4, v_6, v_8, are cut off from the reconfigurable switch. Furthermore, in a segregated setting, when v_1 wants to send flows to v_7 or v_9, one of the flow paths will be very long. Without segregation, reconfigurable links can be used as shortcuts.

Traffic demands. The resulting network should serve a certain communication pattern, represented as a $|V| \times |V|$ communication matrix D (the *demand matrix*) with positive real-valued entries. An entry (i, j) in D represents the communication frequency from the node v_i to the node v_j.

Reconfiguration problem. We say that the hybrid network N is *configured* by the reconfigurable switches. That is, we will refer to the set of configured links $\mathcal{M} = \cup_{\ell=1}^{n} M_\ell$, the union of the matchings provided by the reconfigurable switches, as the *configuration* of N. For ease of notation, we will simply write $N(\mathcal{M})$ to denote the concrete topology resulting from configuration $\mathcal{M}$ and define $dist_{N(\mathcal{M})}(i, j)$ to be the shortest (weighted) distance from node v_i to node v_j on the network $N(\mathcal{M})$. Given a hybrid network N and a communication demand D, our goal is to compute a network $N(\mathcal{M})$ which minimizes the (weighted) average path length for serving D in N by providing a set of matchings $\mathcal{M}$ accordingly. Succinctly stated:

$$\min \sum_{(i,j)\in D} D[i, j] \cdot dist_{N(\mathcal{M})}(i, j) \qquad (1)$$

That is, we want to minimize the sum of the path lengths, weighted by the demand (*i.e.,* flow size) and link costs: for each ordered pair of nodes $v_i, v_j \in V$, we multiply the (weighted) length of the shortest path $dist_{N(\mathcal{M})}(i, j)$ from v_i to v_j on $N(\mathcal{M})$ with their entry (i, j) in D.

Different goals. We note that other objectives, such as, *e.g.,* link load, congestion, or flow completion times, have been studied in recent work [26, 47]. While all these objectives are conceptually related, we only focus on (1) in this work. Furthermore, reconfigurations can be performed in mili- to microseconds [3, 18, 48], but are not instantaneous, which poses the problem of how to gracefully transition between two topologies. We do not study these issues, they fall into the realm of consistent network updates [16].

3 ONE RECONFIGURABLE SWITCH

A common approach in previous work on hybrid networks is to use Edmond's *Blossom* algorithms [14, 33] for the reconfigurable network components [6, 15, 45]. The underlying intuition is that large flows are prioritized, and all other flows have to share the remaining network. As both network components are not jointly optimized, the resulting matching can be inefficient.

3.1 Segregated Networks

We show in the following that the above idea is only optimal in a specific *routing policy* [44] setting, namely when the connections choose between either a direct reconfigurable (single-hop) or an arbitrary static connection.

THEOREM 1. *Let $N = (V, E, S, w)$ be a weighted hybrid network with a single reconfigurable circuit switch. If every route must either solely use* 1) *the static links E or* 2) *a single hop via the reconfigurable switch, then an optimal reconfiguration $N(\mathcal{M})$ can be computed in polynomial time for any D.*

PROOF. We start by computing the shortest path lengths for all matrix entries in D via the static network E, multiplying each one by its demand entry in D. For an ordered node pair $v_i, v_j \in V$, we denote the resulting value as $|v_i, v_j|_{E,D}$. Similarly, we denote the link weight between v_i, v_j via an appropriate matching in the reconfigurable switch as $|v_i, v_j|_M$, ($-\infty$ if neither is connected to the reconfigurable switch).

We create a complete undirected graph $G' = (V, E')$ with the n nodes, where the link weights between two nodes v_i and v_j are set as $|v_i, v_j|_{E,D} + |v_j, v_e|_{E,D} - |v_i, v_j|_M$: the net gains of choosing a link to be included in a matching of the reconfigurable switch, as a possible shortcut via the reconfigurable component of the hybrid network. We then compute a maximum weight matching on G' in polynomial time [33] and choose the corresponding matching for $N(\mathcal{M})$.

No matching can negatively impact the expected path lengths; it can only improve them. Even if no matching improves the expected path lengths, the communication pattern can simply ignore these links. Hence, optimality of $N(\mathcal{M})$ follows from the fact that there are no multi-hop dependencies when reconfigurable links are chosen via the maximum weight matching in G'; every chosen link $e = (v_i, v_j)$ only impacts the communication pattern between v_i and v_j. □

3.2 Non-Segregated Networks

We saw in the last section that reconfiguring a single switch is easy if the routing policy is artificially restricted to segregation. However, as shown in Figure 1, non-segregation can lead to network configurations with better performance.

If the network configuration can be planned well ahead of time, it is possible to perform rigorous and long-running optimizations, *e.g.,* employing mixed integer programs. Current work, however, relies on heuristics respectively sophisticated mechanisms, such as quickly rotating through a set of matchings [34]. For example, the authors of *FireFly* [6] note: "*The high-level idea behind our heuristic is to extend the traditional Blossom algorithm for computing the maximum matching to incorporate multi-hop traffic.*"

We go on to show that these heuristic choices can be substantiated because the non-segregated problem is NP-hard. More precisely, we prove NP-hardness, even for sparse unit demands and unit link weights.

THEOREM 2. *Let N be a weighted hybrid network with one reconfigurable switch, connected to all n nodes. Optimal network reconfiguration is NP-hard, even for sparse unit size communication patterns and unit weight links.*

To prove Theorem 2, we first prove a supporting lemma, using multiple switches with small port counts [48]. Note that the proof is directly extendable to higher port counts.

LEMMA 1. *Let N be a weighted hybrid network with multiple reconfigurable switches, each connecting three nodes. Computing an optimal network reconfiguration is NP-hard, even for sparse unit size entries in D and unit weight links.*

PROOF. Our reduction is from 3-SAT [17], with r variables and k clauses C_i, each containing exactly three literals. For any 3-SAT instance I, we create an instance I' of our problem as follows: all link costs will be one, as well as all communication demands greater than zero.

We create a node c_i for each clause C_i, along with a single destination node d for all communication patterns. We also create r variable gadgets, with the following idea: if we set an appropriate truth assignment in the reconfigurable switch, the resulting matching can satisfy some clause, yielding a short path. Otherwise, the path will be slightly longer via this variable. The construction details are presented in Figure 2, zooming in on a variable gadget with two clauses.

For each variable x_j contained in any clause C_i, we connect the node c_i to either x_j^{true} or x_j^{false}, depending on whether the literal is set to x_j or $\neg x_j$ in C_i. Moreover, each clause node c_i has unit communication demand to d. As those are all entries in the matrix D, we have a sparse communication pattern.

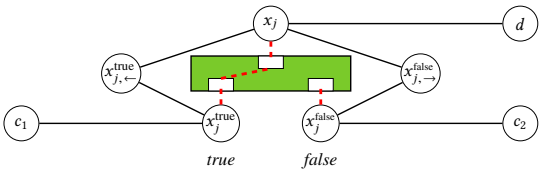

Figure 2: Variable gadget for x_j, set to true by matching x_j^{true} to x_j, with clauses $C_1 = (x_j, \dots)$ and $C_2 = (\neg x_j, \dots)$. Now, the shortest path from clause-node c_1 to d has a length of three via x_j, but of four for c_2.

It remains to study the optimal solutions for our instances I'. Observe that if I is satisfiable, we can create a set of matchings $\mathcal{M}$ corresponding to the correct truth assignment of I, where each flow has an optimal path length of three (one hop to a variable gadget, one hop through the variable gadget, one hop to d), $3k$ in total. Assume, for the sake of contradiction, that I is not satisfiable and we still obtain a path length of $3k$ or better in I'. As no flow can have a path length of two by construction, every flow must have a path length of three. However, this implies that every clause node c_i is connected to at least one variable gadget that "satisfies" a literal in C_i. □

One switch. We can now consider the case that there is only one reconfigurable switch, connected to all nodes. Our proof can easily be adapted to the case that the switch is just connected to a constant fraction of all nodes, as in, *e.g.,* [15, 45].

Proof. We follow the proof idea from Lemma 1, *i.e.,* a reduction from 3-SAT. The challenge is that we no longer have convenient 3-port reconfigurable switches, but a single switch that connects all nodes. In order to re-use the proof idea, we first allow for demand matrix entries of arbitrary size; we later show how to set them to unit size again. We again create r variable gadgets, but this time, they are slightly modified. More specifically, they contain the nodes x_j, x_j^{true}, x_j^{false}, and a further node x'_j, with the latter node connected to x_j via a static link. Next, we create communication demands with an arbitrarily high value f_∞ from both x_j^{true} and x_j^{false} to x_j. If the whole graph were to consist of those four nodes, an optimal matching would create a link between one of $x_j^{\text{true}}, x_j^{\text{false}}$ and x_j and connect the remaining node from x_j^{true}, x_j^0 to x'_j, yielding an objective function value of $f_\infty + 2 \cdot f_\infty$ (one path of length one, the other of length two via x'_j). When the graph is extended to r such gadgets, the optimal solution $N(\mathcal{M})$ again has a cost of $r \cdot f_\infty + 2 \cdot f_\infty$. Next, we add the node d again, but also with a node d', again with a demand of f_∞ from d to d', forcing an optimal solution to match d and d'. As before, we connect d to all nodes x_j with static links; similarly, with c_i and c'_i, we connect the c_i to the variable gadgets as before.

Observe that to obtain an optimal solution, all matchings are already pre-defined, with the exception of the nodes in the variable gadget, where choosing an assignment corresponding to true or false yields the same value for the objective function. At this point, we re-add the demands from the clause nodes c_i to d, and observe that the analogous proof arguments hold as for the proof of Theorem 1. This concludes the case where arbitrarily high demands of f_∞ are allowed.

We now consider sparse unit size communication pattern entries in D. Note that the previous demands of f_∞ do not need to be arbitrarily high for the construction to work; they just need to outweigh any benefit of the communication pattern between the clause nodes c_i and d. For example, $f_\infty = 1000 \cdot n^2$ clearly suffices, we could easily optimize for smaller values. For each node v with a demand of f_∞, w.l.o.g. $f_\infty \mod 2 = 0$, to some other node v', we create $2 \cdot f_\infty$ further nodes, where half are connected to v with static links, and the other half are not connected to any static link. We remove the demand of f_∞ from v, and replace it with unit demands from the first half of the new nodes to v'. Due to construction, the first half of the nodes need to create a perfect matching with the second half – imagine the outcome as a star where each of the $f_\infty/2$ arms has a length of two. This modification increases the instance size only polynomially but retains the correct NP-hardness reduction from 3-SAT; we omit the technical details. □

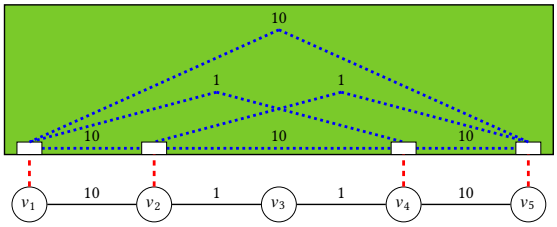

Figure 3: A 4-port reconfigurable switch with all possible links shown in dotted blue, along with link weights of 1 or 10. The static links form a path from v_1 to v_5. The shortest path from v_1 to v_5 traverses the reconfigurable switch twice, via v_1--v_4—v_3—v_2--v_5. Note that this is an extreme example, to showcase that our algorithms work even in this setting. Our NP-hardness proof in §4.2 for two flows does not require different link weights for the reconfigurable switches.

4 MANY RECONFIGURABLE SWITCHES

As seen in the last sections, restrictions to segregated reconfigurable components are tractable, and non-segregated connections with a reconfigurable switch are NP-hard. We now turn our attention to the other end of the spectrum, the study of multiple reconfigurable switches. Unlike Section 3, it is unclear how matching algorithms apply in a useful way.

Furthermore, in a multi-hop setting, a single flow might traverse a single reconfigurable switch multiple times under adversarial weight functions; see the example in Figure 3. As the direct connection from v_1 to v_5 has a cost of 10, a detour crossing the same reconfigurable switch twice is optimal.

4.1 One Flow: Easy

However, we can apply network flow algorithms by modifying the hybrid network topology for our computations.

Theorem 3. *Let $N = (V, E, S, w)$ be a weighted hybrid network. If D contains a communication pattern for only one pair of nodes, then an optimal reconfiguration $N(\mathcal{M})$ can be computed in polynomial time, if the triangle inequality holds for the link weights of every reconfigurable switch.*

Proof. Our proof will rely on the fact that an integral minimum cost flow can be computed in polynomial time,

for a single source-destination pair [1]. In our construction, we assume all links have unit capacity. We also create a fake source and destination node, which are respectively connected to the real source and destination with a static directed link of weight 0, enforcing that the maximum flow is one. As the solution is integral, we obtain a single path.

It remains to combinatorially augment the network to apply flow algorithms. The key idea we want to capture is that every node, for every reconfigurable switch, may only connect to one other node. Thus, for nodes v connected to switches s_ℓ, we create a fake node v_ℓ, connecting it to v with link weight 0. We remove the reconfigurable switches, recreating connectivity with weight-preserving links between fake nodes. Unfortunately, this construction fails to capture one important aspect. Consider a 3-port switch s_ℓ with nodes u, v, w. A flow from u to w could take the path $u - u_\ell - v_\ell - w_\ell - w$, which cannot be translated back to the original hybrid network: it would require v to match with both u and w. Nevertheless, as we require that each reconfigurable switch upholds the triangle inequality for its link weights, the path $u - u_\ell - v_\ell - w_\ell - w$ can be optimized to a valid one of equal or shorter length, namely $u - u_\ell - w_\ell - w$. We can thus compute a minimum cost flow in polynomial time, yielding an optimal solution after "post-processing" the infeasible parts. □

4.2 Two Flows and Beyond: Hard

The previous section shows that optimizing reconfiguration is easy when there is a single communication flow. From a complexity point of view, we can extend the tractability results to more flows when the number of different switch configurations is small: for example, hybrid networks containing $O(1)$ switches with $O(1)$ ports each can be (inefficiently) solved in polynomial time by a brute-force approach. However, for many switches, computing an optimal reconfiguration already becomes NP-hard for even two flows, if we want to minimize the worst-case path length [29].

THEOREM 4. *Let $N = (V, E, S, w)$ be a weighted hybrid network where all reconfigurable switches have four ports. Let D be a communication pattern with two unit size entries sharing the same destination v_j. A reconfiguration that minimizes the maximum path length for the flows is NP-hard.*

PROOF. Our reduction is from the NP-complete Partition problem [17]: given k integers $i_1, \ldots, i_k$ of total sum P, is there a partition into two disjoint sets of equal sum. Our construction sketch uses only 4-port reconfigurable switches.

For any Partition instance I, we create an instance I' of our problem, as depicted in Figure 4: the two flows have to pass through k reconfigurable switches with 4 ports each. To obtain a connected graph, only two matchings are feasible at every switch, with one flow routed along the upper path, and the other along the lower path. Depending on the choice, the respective flow is impacted by either a delay (weight) of some i_j or 0. As such, if the k integers $i_1, \ldots, i_k$ can be distributed among both flows s.t. their summed up delay is identical, then I is a yes-instance. Otherwise, one of the two flows incurs a larger delay; *i.e., I* is a no-instance. □

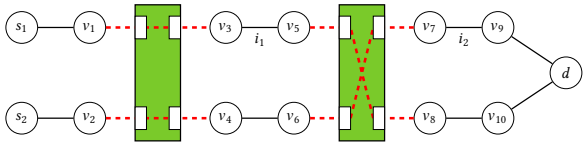

Figure 4: Topology for a partition instance with two values i_1, i_2. When $i_1 = i_2$, the shown solution yields s_1, d & s_2, d paths of identical length.

4.3 Unidirectional Links

In the last sections, we investigated hybrid networks with *bidirectional* dynamic connections. A natural model extension is to consider *unidirectional* dynamic links.

Background. Farrington *et al.* show that unidirectional circuits outperform bidirectional circuits in asymmetric demand situations, by configuring circuits unidirectionally in *Helios* [15]. Unidirectional reconfigurable links also exist in the wireless domain; *e.g.,* Zhou *et al.* [49] use 3D beamforming to let 60GHz wireless signals bounce off data center ceilings.[1] *FireFly* [6] brings the same concept to steerable free-space lasers with a reflecting ceiling mirror; *ProjecTor* [18] extends this with digital micromirror devices attached to so-called disco-balls, thus improving the number of possible connections (from ten to thousands) and the switching time (from 20ms to 7-12μs).

Directed Model. We study directed weighted hybrid networks, where we follow the naming conventions of free-space optics for ease of readability. In other words, we say that a node v may have multiple lasers and receivers. As before, our results apply to any technology adhering to the described model, *e.g.,* unidirectional optical circuits or directed point-to-point wireless connections.

Formally, a laser at a node v may match to a receiver at a node from some set $V_v \in V$, creating a directed link, but every receiver may only be matched with one laser. In other words, we investigate weighted hybrid free-space optics networks $N = (V, E, S, w)$, where S describes the laser/receiver setting, and a network reconfiguration $N(\mathcal{M})$ describes a valid directed matching of the lasers to the receivers.

One Flow. We modify the proof of Theorem 3, removing the triangle inequality requirement on the way. As before, the key idea we want to capture is that every node may have, at most, one outgoing reconfigurable link (a laser) and, at most, one incoming reconfigurable link (a receiver).

[1] We note that the implementation of Zhou *et al.* [49] uses bidirectional links by employing frequency division on two bands near 60Ghz.

Even though a node may have multiple lasers and receivers, it suffices to create a single one in our construction: if an integral unit size flow were to make use of two lasers (or two receivers) at a single node, the flow would contain a loop, which can be canceled out due to non-negative link weights.

For each node v with at least one laser, we add a node v_{out}, and if v has at least one receiver, a node v_{in}, we also add a directed link from v_{in} to v and a directed link from v to v_{out}, both of weight 0. Next, we appropriately add directed links from *out* nodes to *in* nodes, where a directed link from v_{out} to w_{in} has the same weight as the corresponding link created in the original directed weighted hybrid network. Feasible solutions can be directly translated between the augmented and the original problem setting with identical costs.

COROLLARY 1. *Let N be a directed weighted hybrid network. If D contains only one communication entry from one node v_i to another node v_j, then an optimal reconfiguration $N(\mathcal{M})$ can be computed in polynomial time.*

Outlook. While the NP-hardness results of the previous sections can be directly adapted to the directed case, transferring Theorem 1 to multiple lasers/receivers per node is non-trivial; see also the discussion by Devanour *et al.* [12, §3] on the non-hybrid case. Nor is it clear how to extend Theorem 1 to multiple reconfigurable switches. We leave the investigation of such scenarios to future work.

5 RELATED WORK

To the best of our knowledge, we are the first to study polynomial-time and *exact* network reconfiguration algorithms for data center networks enhanced with reconfigurable switches. Much previous work has not accounted for a static topology at all, *e.g.,* [4, 5], or has focused on optimizing the reconfigurable part of hybrid networks, using, *e.g.,* matching [6, 15, 30, 31, 45], edge-coloring [9], or stable-marriage algorithms [18], see [34, Table 1]. In this section, we focus on the most closely related works and refer the reader to [26, 47] for a more comprehensive overview.

So far, polynomial-time algorithms for optimal or provably approximate demand-aware networks have only been established for networks forming a single tree providing local routing [35], for constant degree networks where the demand is sparse [5] and for non-constant degree networks [4] where the degree can depend on the node popularity. All these designs assume that all links are reconfigurable, so it is unclear how to extend these results to hybrid networks combining both reconfigurable and non-configurable links. That said, the lower bounds on the achievable average path length derived in [5] also apply to our model: if the computed network N(M) is of bounded degree the expected path length is at least in the order of the conditional entropy of the demand matrix D.

Non-segregated routing in hybrid networks is used by Xia *et al.* [48] by leveraging small port-count converter switches for reconfiguration which can dynamically convert between a Clos network and approximate random graphs of different sizes. However, rather than computing optimal configurations, they locally reconfigure the topology according to the traffic patterns, where the performance depends on how well Clos networks respectively random graphs handle the traffic.

Kassing *et al.* [28] took the aforementioned concept of random graphs [40] and proposed building such a data center topology deterministically, leveraging expanders [13, 42], using only static devices. The performance of their topologies depends on using many source-destination paths, instead of restricted routing policies. In fact, the idea is conceptually similar to the non-segregated approach for hybrid networks.

Venkatakrishnan *et al.* [44] studied the reconfigurable parts of the topology and pointed out that routing policies restricted to direct or single-hop routing are inefficient: when indirect or multi-hop routing is employed in the reconfigurable domain, the reachability of nodes can be increased exponentially. The authors provide near optimal scheduling algorithms for the segregated case where network configuration is calculated separately and leave the joint problem as an open question. We advocate taking non-segregated routing policies a step further, *i.e.,* taking the static topology of the hybrid network into account for the joint optimization.

Mellette *et al.* [34] proposed an approach orthogonal to ours, where the switches rotate through a set of pre-defined matchings, also leveraging Valiant-style [43] multi-hop optical connections. Such indirect routing would also be of interest to us when considering reconfiguration latency.

Lastly, while our paper focuses on data center technology, some recent work has studied reconfigurable wide area networks, employing Reconfigurable Optical Add-Drop Multiplexers (ROADMs) [24, 25] or rate adaptive links [37]. These are relevant to our work and we believe it would be interesting to develop a model that makes algorithms and conclusions portable between the two areas.

6 CONCLUSION

We have presented a first formal characterization of the algorithmic complexity of data center network architectures based on reconfigurable switches. We understand our work as a first step and believe it opens several questions for future research. More particularly, it would be interesting to chart a more detailed landscape of the algorithmic complexity underlying such network architectures, and to study the design of approximation algorithms and thresholds.

Acknowledgements. We would like to thank Chen Avin for valuable discussions and feedback. We would also like to thank our shepherd Aurojit Panda and the anonymous reviewers for their helpful feedback on our paper.

REFERENCES

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. *Network flows - theory, algorithms and applications*. Prentice Hall.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *SIGCOMM*. ACM, 63–74.

[3] Dan Alistarh, Hitesh Ballani, Paolo Costa, Adam Funnell, Joshua Benjamin, Philip M. Watts, and Benn Thomsen. 2015. A High-Radix, Low-Latency Optical Switch for Data Centers. *Computer Communication Review* 45, 5 (2015), 367–368.

[4] Chen Avin, Alexandr Hercules, Andreas Loukas, and Stefan Schmid. 2018. *rDAN*: Toward robust demand-aware network designs. *Inf. Process. Lett.* 133 (2018), 5–9.

[5] Chen Avin, Kaushik Mondal, and Stefan Schmid. 2017. Demand-Aware Network Designs of Bounded Degree. In *DISC (LIPIcs)*, Vol. 91. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 5:1–5:16.

[6] Navid Hamed Azimi, Zafar Ayyub Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. 2014. FireFly: a reconfigurable wireless data center fabric using free-space optics. In *SIGCOMM*. ACM, 319–330.

[7] Maciej Besta and Torsten Hoefler. 2014. Slim Fly: A Cost Effective Low-Diameter Network Topology. In *SC*. IEEE Computer Society, 348–359.

[8] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. 2014. OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility. *IEEE/ACM Trans. Netw.* 22, 2 (2014), 498–511.

[9] Li Chen, Kai Chen, Zhonghua Zhu, Minlan Yu, George Porter, Chunming Qiao, and Shan Zhong. 2017. Enabling Wide-Spread Communications on Optical Fabric with MegaSwitch. In *NSDI*. USENIX Association, 577–593.

[10] Cisco. 2015. Cisco Global Cloud Index: Forecast and Methodology, 2015-2020. *White Paper* (2015).

[11] Rodrigo De Souza Couto, Stefano Secci, Miguel Elias Mitre Campista, and Luís Henrique Maciel Kosmalski Costa. 2016. Reliability and Survivability Analysis of Data Center Network Topologies. *J. Network Syst. Manage.* 24, 2 (2016), 346–392.

[12] Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Monia Ghobadi, Ratul Mahajan, and Amar Phanishayee. 2016. *Stable Matching Algorithm for an Agile Reconfigurable Data Center Interconnect*. Technical Report. Microsoft Research.

[13] Michael Dinitz, Michael Schapira, and Asaf Valadarsky. 2017. Explicit Expanding Expanders. *Algorithmica* 78, 4 (2017), 1225–1245.

[14] Jack Edmonds. 1965. Paths, Trees and Flowers. *Canad. J. Math* 17 (1965), 449–467.

[15] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *SIGCOMM*. ACM, 339–350.

[16] Klaus-Tycho Foerster, Stefan Schmid, and Stefano Vissicchio. 2018. Survey of Consistent Network Updates. *CoRR* abs/1609.02305v2 (2018).

[17] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[18] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil R. Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel C. Kilper. 2016. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *SIGCOMM*. ACM, 216–229.

[19] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. 2011. Understanding network failures in data centers: measurement, analysis, and implications. In *SIGCOMM*. ACM, 350–361.

[20] Albert G. Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In *SIGCOMM*. ACM, 51–62.

[21] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. 2009. BCube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM*. ACM, 63–74.

[22] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. 2008. Dcell: a scalable and fault-tolerant network structure for data centers. In *SIGCOMM*. ACM, 75–86.

[23] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. 2011. Augmenting data center networks with multi-gigabit wireless links. In *SIGCOMM*. ACM, 38–49.

[24] Su Jia, Xin Jin, Golnaz Ghasemiesfeh, Jiaxin Ding, and Jie Gao. 2017. Competitive analysis for online scheduling in software-defined optical WAN. In *INFOCOM*. IEEE, 1–9.

[25] Xin Jin, Yiran Li, Da Wei, Siming Li, Jie Gao, Lei Xu, Guangzhi Li, Wei Xu, and Jennifer Rexford. 2016. Optimizing Bulk Transfers with Software-Defined Optical WAN. In *SIGCOMM*. ACM, 87–100.

[26] Christoforos Kachris and Ioannis Tomkos. 2012. A Survey on Optical Interconnects for Data Centers. *IEEE Communications Surveys and Tutorials* 14, 4 (2012), 1021–1036.

[27] Srikanth Kandula, Jitendra Padhye, and Paramvir Bahl. 2009. Flyways To De-Congest Data Center Networks. In *HotNets*. ACM SIGCOMM.

[28] Simon Kassing, Asaf Valadarsky, Gal Shahaf, Michael Schapira, and Ankit Singla. 2017. Beyond fat-trees without antennae, mirrors, and disco-balls. In *SIGCOMM*. ACM, 281–294.

[29] Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. 1990. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics* 26, 1 (1990), 105–115.

[30] He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter. 2014. Circuit Switching Under the Radar with REACToR. In *NSDI*. USENIX Association, 1–15.

[31] He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. 2015. Scheduling techniques for hybrid circuit/packet networks. In *CoNEXT*. ACM, 41:1–41:13.

[32] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas E. Anderson. 2013. F10: A Fault-Tolerant Engineered Network. In *NSDI*. USENIX Association, 399–412.

[33] László Lovász and Michael D Plummer. 2009. *Matching theory*. Vol. 367. American Mathematical Soc.

[34] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. 2017. RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. In *SIGCOMM*. ACM, 267–280.

[35] Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker. 2016. SplayNet: Towards Locally Self-Adjusting Networks. *IEEE/ACM Trans. Netw.* 24, 3 (2016), 1421–1433.

[36] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. *Computer Communication Review* 45, 5 (2015), 183–197.

[37] Rachee Singh, Monia Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. 2018. RADWAN: Rate Adaptive Wide Area Network. In *SIGCOMM*. ACM.

[38] Ankit Singla. 2016. Fat-FREE Topologies. In *HotNets*. ACM, 64–70.

[39] Ankit Singla, Philip Brighten Godfrey, and Alexandra Kolla. 2014. High Throughput Data Center Topology Design. In *NSDI*. USENIX Association, 29–41.

[40] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. 2012. Jellyfish: Networking Data Centers Randomly. In *NSDI*. USENIX Association, 225–238.

[41] Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, and Yueping Zhang. 2010. Proteus: a topology malleable data center network. In *HotNets*. ACM, 8.

[42] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. 2016. Xpander: Towards Optimal-Performance Datacenters. In *CoNEXT*. ACM, 205–219.

[43] Leslie G. Valiant. 1982. A Scheme for Fast Parallel Communication. *SIAM J. Comput.* 11, 2 (1982), 350–361.

[44] Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, and Pramod Viswanath. 2016. Costly Circuits, Submodular Schedules and Approximate Carathéodory Theorems. In *SIGMETRICS*. ACM, 75–88.

[45] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T. S. Eugene Ng, Michael Kozuch, and Michael P. Ryan. 2010. c-Through: part-time optics in data centers. In *SIGCOMM*. ACM, 327–338.

[46] Haitao Wu, Guohan Lu, Dan Li, Chuanxiong Guo, and Yongguang Zhang. 2009. MDCube: a high performance network structure for modular data center interconnection. In *CoNEXT*. ACM, 25–36.

[47] 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.

[48] Yiting Xia, Xiaoye Steven Sun, Simbarashe Dzinamarira, Dingming Wu, Xin Sunny Huang, and T. S. Eugene Ng. 2017. A Tale of Two Topologies: Exploring Convertible Data Center Network Architectures with Flat-tree. In *SIGCOMM*. ACM, 295–308.

[49] Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y. Zhao, and Haitao Zheng. 2012. Mirror mirror on the ceiling: flexible wireless links for data centers. In *SIGCOMM*. ACM, 443–454.