Optimizing Reconfigurable Optical Datacenters: The Power of Randomization

Marcin Bienkowski[1] David Fuchssteiner[2] Stefan Schmid[3]

[1] Wroclaw University, Poland [2] University of Vienna, Austria [3] TU Berlin, Germany

ABSTRACT

Reconfigurable optical topologies are a promising new technology to improve datacenter network performance and cope with the explosive growth of traffic. In particular, these networks allow to directly and adaptively connect racks between which there is currently much traffic, hence making an optimal use of the bandwidth capacity by avoiding multi-hop forwarding.

This paper studies the dynamic optimization of such reconfigurable topologies, by adapting the network to the traffic in an online manner. The underlying algorithmic problem can be described as an online maximum weight b-matching problem, a generalization of maximum weight matching where each node has at most $b \geq 1$ incident matching edges.

We make the case for a randomized approach for matching optimization. Our main contribution is a $O(\log b)$-competitive algorithm and we show that it is asymptotically optimal. This algorithm is hence exponentially better than the best possible deterministic online algorithm.

We complement our theoretical results with extensive trace-driven simulations, based on real-world datacenter workloads.

CCS CONCEPTS

• **Networks** → **Network architectures**; **Network algorithms**.

KEYWORDS

optical datacenter networks, demand-aware networks, optimization, online algorithms, randomization

1 INTRODUCTION

Datacenter networks have become a critical infrastructure of our digital society, and the performance requirements on these networks are increasingly stringent. Indeed, especially inside datacenters, traffic is currently growing explosively, e.g., due to the popularity of data-centric applications related to AI/ML [65], but also due to the trend of resource disaggregation (e.g., requiring fast access to remote resources such as GPUs), as well as hardware-driven and distributed training [46]. Accordingly, over the last years, great efforts have been made to improve datacenter networks [44].

A particularly innovative approach to improve datacenter network performance, is to dynamically adjust the datacenter topologies towards the workload they serve, in a *dynamic* and *demand-aware* manner. Such adjustments are enabled by emerging reconfigurable optical communication technologies, such as optical circuit switches, which provide dynamic *matchings* between racks [6, 75, 41, 55, 54, 27, 35, 17, 48, 16, 30, 70, 62, 69, 64, 67, 15, 36, 24, 34, 32]. Indeed, empirical studies have shown that datacenter traffic features much spatial and temporal structure [3, 8, 61], which may be exploited for optimization.

This paper studies the optimization problem underlying such reconfigurable datacenter networks. In particular, we consider a typical leaf-spine datacenter network where a set of racks are interconnected by b optical circuit switches, each of which provides one matching between top-of-rack switches, so b matchings in total. In a nutshell, the goal is to optimize these matchings so that the link resources are used optimally, i.e., the number of hops taken per communicated bit is minimized (more details will follow).

1.1 The Model

Our problem optimizing an optical reconfigurable datacenter network can be modeled as an online dynamic version of the classic b-matching problem [2]. In this problem, each node can be connected with at most b other nodes (using optical links), which results in a b-matching.

Input. We are given an arbitrary (undirected) static weighted and connected network on the set of nodes V (i.e., the top-of-rack switches) connected by a set of non-configurable links F: the fixed network (based, e.g., on a Clos or fat-tree topology). Let V^2 be the set of all possible unordered pairs of nodes from V. For any node pair $e = \{s, t\} \in V^2$, we call s and t the *endpoints* of e, and we let ℓ_e denote the length of a shortest path between nodes u and v in graph $G = (V, F)$. Note that u and v are not necessarily directly connected in F.

The fixed network can be enhanced with reconfigurable links, providing a matching of degree b: Any node pair from V^2 may become a *matching edge* (such an edge corresponds to a reconfigurable optical link), but the number of matching edges incident to any node has to be at most b, for a given integer $b \geq 1$.

The demand is modeled as a sequence of communication requests[1] $\sigma = \{s_1, t_1\}, \{s_2, t_2\}, \ldots$ revealed over time, where $\{s_i, t_i\} \in V^2$.

Output and objective. The task is to schedule the reconfigurable links over time, that is, to maintain a dynamically changing b-matching $M \subseteq V^2$. This means that each node pair from M is called a *matching edge*, and we require that each node has at most b incident matching edges. We aim to jointly minimize routing and reconfiguration costs, defined below.

Costs. The cost of serving (i.e., routing cost) a request $e = \{s, t\}$ depends on whether s and t are connected by a matching edge. In our model, a given request can either only take the fixed network or a direct matching edge (i.e., routing is segregated [30]). If $e \notin M$, the requests are routed exclusively on the fixed network, and the corresponding cost is ℓ_e (shorter paths imply smaller resource costs, i.e., lower "bandwidth tax" [55]). If $e \in M$, the request is served by the matching edge, and the routing costs 0.

Once the request is served, an algorithm may modify the set of matching edges: reconfiguration costs α per each node pair added or removed from the matching M. (The reconfiguration cost and time can be assumed to be independent of the specific edge.)

Online algorithms. A (randomized) algorithm Onl is *online* if it has to take decisions without knowing the future requests (in our case, e.g., which edge to include next in the matching and which to evict). Such an algorithm is said to be ρ-competitive [12] if there exists β such that for any input instance σ, it holds that

$$\mathbf{E}[\textsc{Onl}(\sigma)] \leq \rho \cdot \textsc{Opt}(\sigma) + \beta,$$

where $\textsc{Opt}(\sigma)$ is the cost of the optimal (offline) solution for σ and $\mathbf{E}[\textsc{Onl}(\sigma)]$ is the expected cost of algorithm Onl on σ. The expectation is taken over all random choices of Onl, i.e., the input itself is worst-possible, created adversarially. It is worth noting that β can depend on the parameters of the network, such as the number of nodes, but has to be independent of the actual sequence of requests. Hence, in the long run, this additive term β becomes negligible in comparison to the actual cost of online algorithm Onl.

Generalization to (b,a)-matching problem. The comparison of an online algorithm to the fully clairvoyant offline solution may seem unfair. Therefore, we consider our b-matching problem also in a more generalized setting. We denote this generalization as (b, a)*-matching*, where the restrictions imposed on an online algorithm remain unchanged: the number of matching edges incident to any node has to be at most b. However, the optimal solution is more constrained: the number of matching edges has to be at most $a \leq b$ for any node. Similar settings are studied frequently in the literature, as *resource augmentation* models, e.g., in the context of paging and caching [19, 72, 73, 7] or scheduling problems, see, e.g., [14, 39].

[1] A request could either be an individual packet or a certain amount of bytes transferred. This model of a request sequence is often considered in the literature and is more fine-grained than, e.g., a sequence of traffic matrices.

1.2 Our Contributions

Motivated by the problem of how to establish topological shortcuts in datacenter networks supported by b optical switches, we present a randomized online algorithm for the (b, a)-matching problem.

Our proposed solution achieves a competitive ratio of $O((1+\max_e\{\ell_e\}/\alpha)\cdot\log(b/(b-a+1))$, which almost optimal: we will also show a lower bound of $\Omega(\log(b/(b-a+1))$. We note that in all practical applications α is by several orders of magnitude greater than $\ell_{\max}$, and thus the factor $1 + \max_e\{\ell_e\}/\alpha$ is close to 1.

When our algorithm is compared to an optimal algorithm that has to maintain b-matching as well (i.e., when $a = b$), the competitive ratio of our algorithm becomes $O((1 + \max_e\{\ell_e\}/\alpha) \cdot \log b)$ and the lower bound becomes $\Omega(\log b)$. It is worth noting that the best deterministic online algorithm for this problem is only $\Theta(b)$-competitive.

Our analysis relies on a reduction to a uniform case (where $\alpha = 1$ and all path lengths are equal to 1), which allows us to avoid delicate charging arguments and enables a simplified analytical treatment.

We show that our randomized algorithm is not only better with respect to the worst-case competitive ratio than the deterministic online algorithm in theory, but also attractive in practice. Our empirical evaluation, based on various real-world datacenter traces, shows that our algorithm is significantly faster while achieving roughly the same matching quality.

As a contribution to the research community, to facilitate follow-up work, and to ensure reproducibility, we will open-source our implementation and all experimental artifacts together with this paper.

1.3 Organization

The remainder of this paper is organized as follows. We present our optimization framework together with a formal analysis in §2. §3 reports on our empirical results. After reviewing related work in §4, we conclude our contribution and discuss future work in §5.

2 ALGORITHM AND ANALYSIS

Recall that our goal is to adapt the reconfigurable datacenter links such that bandwidth resources are used optimally (i.e.,

bits travel the least number of hops) while accounting for reconfiguration costs, in an online manner. To this end, we aim to compute heavy matchings between nodes, e.g., the racks resp. top-of-rack switches in the reconfigurable datacenter.

We first define the *uniform* variant of the (b, a)-matching. There, the distance between each pair of nodes is 1 (i.e., $\ell_e = 1$ for any $e \in V^2$) and $\alpha = 1$ (recall that α is the cost of adding or removing a matching edge to/from M).

In the following, let

$$\ell_{\max} = \max_e\{\ell_e\} \qquad \text{and} \qquad \gamma = 1 + \ell_{\max}/\alpha.$$

We first show that it suffices to solve the uniform variant of the problem: once we do this, we can get an algorithm for the general variant, losing an additional factor of $O(\gamma)$.

Afterwards, we show a simple algorithm that uses known randomized algorithms for the paging problem to solve the uniform variant of the (b, a)-matching. This will yield a randomized $O(\gamma \cdot \log(b/(b-a+1))$-competitive algorithm for the (b, a)-matching problem.

2.1 Reduction to uniform case

We now show a reduction to the uniform case.

Theorem 1. *Assume there exists a (deterministic or randomized) c-competitive algorithm $\textsc{Alg}_1$ for the variant of the (b, a)-matching problem for the uniform case. Then there exists a (respectively, deterministic or randomized) $4\gamma \cdot c$-competitive algorithm Alg for the (b, a)-matching problem.*

Proof. Let I be the input for algorithm Alg. Alg creates (in an online manner) another input instance I_1. For any node pair e let

$$k_e = \lceil \alpha/\ell_e \rceil.$$

For any node pair e independently, we call each k_e-th request to e in I *special*; the remaining ones are called *ordinary*. Input I_1 contains only special requests from I. Furthermore, for I_1, we assume that $\alpha = 1$ and the distance between each pair of nodes is 1. Alg simply runs $\textsc{Alg}_1$ on I_1 and repeats its reconfiguration choices (modifications of the matching M). In particular, this means that Alg may performs changes to M only upon the k_e-th occurrence of a given node pair e.

To prove the theorem, we will show the following inequalities for some value β independent of the input I.

(1) $\textsc{Alg}(I) \le 2\gamma \cdot \alpha \cdot \textsc{Alg}_1(I_1) + |V^2| \cdot \gamma \cdot \alpha$
(2) $\textsc{Alg}_1(I_1) \le c \cdot \textsc{Opt}(I_1) + \beta$
(3) $\textsc{Opt}(I_1) \le (2/\alpha) \cdot \textsc{Opt}(I)$

Showing the first inequality. We fix any node pair e and we analyze the cost pertaining to handling e both by Alg and $\textsc{Alg}_1$. We partition I into disjoint intervals, so that the first interval starts at the beginning of the input sequence, and each interval except the last one ends at the special request to e inclusively. (We note that the partition differs for different choices of e.) We look at any non-last interval S and its counterpart S_1 in the input I_1. Note that except k_e requests to e and one request to e in S_1, these intervals may contain also requests to other node pairs. We now compare the costs pertaining to e in S incurred on Alg (denoted $\textsc{Alg}(S, e)$), to the costs pertaining to e in S_1 incurred on $\textsc{Alg}_1$ (denoted $\textsc{Alg}_1(S_1, e)$). We consider two cases.

- If $\textsc{Alg}_1(S_1, e) = 0$, then $\textsc{Alg}_1$ must have e in M when it is requested at the end of S_1, and moreover, it cannot perform any reconfigurations that touch e. This means that it must have e in M right after the special request to e preceding S_1, and keep e in M throughout the whole considered interval S_1. As Alg is mimicking the choices of $\textsc{Alg}_1$, it has e in M during S, and thus all k_e requests to e in S are free for Alg and $\textsc{Alg}(S, e) = 0$ as well.
- Otherwise, let $q = \textsc{Alg}_1(S_1, e) \ge 1$. As $\textsc{Alg}_1$ performed at most q reconfigurations concerning e in S_1, so does Alg in S. Furthermore, Alg pays at most for k_e requests to e (ℓ_e for each). We note that

$$\begin{aligned} k_e \cdot \ell_e &< (\alpha/\ell_e + 1) \cdot \ell_e = \alpha + \ell_e \\ &\le \alpha + \ell_{\max} = \gamma \cdot \alpha, \end{aligned}$$

  and thus $\textsc{Alg}(S, e) \le q \cdot \alpha + k_e \cdot \ell_e \le (q + \gamma) \cdot \alpha < 2q\gamma \cdot \alpha = 2\gamma \cdot \alpha \cdot \textsc{Alg}_1(S_1, e)$.

The first inequality follows by summing over all (non-last) intervals and all possible choices of e. The term $|V^2| \cdot \gamma \cdot \alpha$ upper-bounds the total cost in the last intervals: there are $|V^2|$ of them, and in each the cost is at most $k_e \cdot \ell_e < \gamma \cdot \alpha$.

Showing the second inequality. This one follows immediately as $\textsc{Alg}_1$ is c-competitive on input I_1.

Showing the third inequality. We again fix any edge e and consider the same partitioning into intervals as above. This time, however, on the basis of solution $\textsc{Opt}(I)$, we construct an offline solution Off for the input I_1 by mimicking all reconfiguration choices of Opt on input I_1. Note that I_1 contains only a subset of requests from I and thus, in response to a single request in I_1, Off may react with a sequence of reconfigurations that are redundant (i.e., remove some edge from M and then fetch it back). However, as the matching M that Off maintains is the same as that of Opt, it is feasible.

We now argue that for any interval S in I and the corresponding interval S_1 in I_1, it holds that $\textsc{Off}(S_1, e) \le (2/\alpha) \cdot \textsc{Opt}(S, e)$.

- If $\textsc{Opt}(S, e) < \alpha$, then Opt performs no reconfigurations concerning e. In this case Opt has to pay for all requests to e within S or none of them. The first case would imply that its cost is at least $k_e \cdot \ell_e \ge \alpha$, and thus the only possibility is that it does not pay for any request. In this case, Off does not perform any reorganizations pertaining to e either and does not pay

for the only request to e in S_1. Hence, $\textsc{Off}(S_1, e) = 0 = \textsc{Opt}(S, e)$.
- Otherwise, let $q = \textsc{Opt}(S, e) \geq \alpha$. Opt performs at most $\lfloor q/\alpha \rfloor$ reconfigurations pertaining to e. Off executes the same reconfigurations, and hence it pays at most $\lfloor q/\alpha \rfloor$ for reconfigurations in S_1 pertaining to e and at most 1 for the only request to e in S_1. Thus, $\textsc{Off}(S_1, e) \leq \lfloor q/\alpha \rfloor + 1 \leq 2 \cdot (q/\alpha) = (2/\alpha) \cdot \textsc{Opt}(S, e)$.

The third inequality now follows by summing the relation $\textsc{Off}(S_1, e) \leq (2/\alpha) \cdot \textsc{Opt}(S, e)$ over all intervals (including the last ones) and node pairs e, and observing that the optimal cost for I_1 can be only smaller than the cost of Off on I_1.

Combining all three inequalities. By combining all three inequalities, we obtain that

$$\begin{aligned}\textsc{Alg}(I) &\leq 2\gamma \cdot \alpha \cdot \textsc{Alg}_1(I_1) + |V^2| \cdot \gamma \cdot \alpha \\ &\leq 2\gamma \cdot \alpha \cdot c \cdot \textsc{Opt}(I_1) + 2\gamma \cdot \alpha \cdot \beta + |V^2| \cdot \gamma \cdot \alpha \\ &\leq 4\gamma \cdot c \cdot \textsc{Opt}(I) + (|V^2| + 2\beta) \cdot \gamma \cdot \alpha,\end{aligned}$$

which concludes the proof. □

2.2 Algorithm for uniform case

Below we present a randomized algorithm $\textsc{Alg}_1$, which solves the uniform variant of the problem (where $\ell_e = 1$ for all $e \in V^2$ and $\alpha = 1$).

Let the (b, a)-paging problem be a resource-augmented variant of the paging problem [19] where an online algorithm has cache of size b and optimal algorithm has a cache of size a.

Theorem 2. *Assume there exists a (deterministic or randomized) $f(b, a)$-competitive algorithm for the (b, a)-paging problem for some function $f : \mathbb{N}_{\geq 0} \times \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq} 0$. Then, there exists an (also deterministic or randomized) $O(f(b, a))$-competitive algorithm for the uniform variant of the (b, a)-matching problem.*

We note that the cost model in the paging problem as defined in many papers (see, e.g., [19, 28, 72]) differs slightly from ours in two aspects:

- In the paging problem, whenever a page is requested, it must be fetched to the cache if it is not yet there (bypassing is not allowed). In contrast, in the (b, a)-matching problem an algorithm does not have to include the requested node pair in the matching.
- In the paging problem, an algorithm pays only for including page in the cache (there is no cost for eviction as in our model). Because bypassing is not allowed, the paging problem does not include the cost of serving a request either.

We will handle these differences in our proof.

Algorithm definition. Our algorithm $\textsc{Alg}_1$ for the (b, a)-matching problem runs separate (b, a)-paging algorithms for each node; initially the caches corresponding to all vertices are empty. $\textsc{Alg}_1$ dynamically creates input sequences for the paging algorithms running at particular vertices. An input sequence for the algorithm running at node w is a sequence of node pairs having w as one of their endpoints. At all times, the paging algorithm keeps a subset of at most b such node pairs in its cache. On the basis of the (possibly random) decisions at each node, $\textsc{Alg}_1$ constructs its own solution, choosing which node pairs are kept as matching edges in M.

More precisely, whenever $\textsc{Alg}_1$ handles a request $e = (u, v)$, it passes query e to the paging algorithms running at its endpoints: separately to node u and to node v. By the definition of the paging problem, both these vertices may reorganize their caches, removing an arbitrary number of elements (node pairs) from their caches, and afterwards they need to fetch e to their caches (if it is not already there).

On this basis, $\textsc{Alg}_1$ reorganizes the matching maintaining the following invariant:

> *Any node pair q is kept in the matching if and only if q is in the caches of both its endpoints.*

Therefore, when $e = (u, v)$ is requested, the actions of $\textsc{Alg}_1$ are limited to the following: (i) some node pairs having one of u or v as endpoints may be removed from M, (ii) e becomes a matching edge in M (if it was not already there).[2] We note that our reduction itself is purely deterministic, but results in a randomized algorithm if we use randomized algorithms for solving paging sub-problems at vertices.

Bounding competitive ratio. Now we proceed with showing the desired competitive ratio of $\textsc{Alg}_1$.

Proof of Theorem 2. Fix any input instance I for the (b, a)-matching problem. It induces $|V|$ paging instances for each node: we denote the instance at node v by I_v. Let A_v be the instance of online paging algorithm run at v. By the theorem assumptions, for any node v it holds that

$$A_v(I_v) \leq f(b, a) \cdot \textsc{Opt}(I_v) + \beta_{b,a} \tag{1}$$

where $\beta_{b,a}$ is a constant independent of the input sequence and $\textsc{Opt}(I_v)$ denotes the optimal a-paging solution for input I_v.

We will show the following relations.

- $\textsc{Alg}_1(I) \leq 4 \cdot \sum_{v \in V} A_v(I_v)$.
- $\sum_{v \in V} \textsc{Opt}(I_v) \leq 14 \cdot \textsc{Opt}(I)$

We note that we do not aim at optimizing the constants here, but rather at the simplicity of the description. Clearly,

[2] Note that there are cases where algorithm $\textsc{Alg}_1$ has less matching edges than allowed by the threshold b. While this does not hinder theoretical analysis, it is worth noting that having an edge in the matching can only help us. Thus, in our experiments the removals are lazy, i.e., an edge is marked for removal and then some marked edges are pruned whenever their number incident to a node exceeds b.

combining these two inequalities with (1) immediately yields the theorem.

We start with proving the second inequality. On the basis of the optimal solution for the a-matching problem $\textsc{Opt}(I)$, we first create an online solution $\textsc{Off}(I)$ for the same problem, but having the property that right after a node pair is requested it is included in the matching. We call this *forcing property*. This can be achieved by changing $\textsc{Opt}$ decisions in the following way: whenever we have a request $e = (u, v)$ such that e is not in the matching neither directly before or after serving the request, we modify the solution in the following way: after executing $\textsc{Opt}$ reorganizations (if any), we include e in the matching. If such a modification causes the degree of matching at u to exceed b, we evict an arbitrary edge $e_u \neq e$ incident to u. We perform an analogous action for v, evicting edge e_v if necessary. Afterwards, right after serving the next request, but before implementing reorganizations of $\textsc{Opt}$, we revert these changes: remove e from M and include e_u and e_v back into M. Note that $\textsc{Off}(I) \leq 7 \cdot \textsc{Opt}(I)$ as for a request to e for which $\textsc{Opt}$ paid 1, the solution of $\textsc{Off}$ adds at most $3+3$ updates of the matching M at the total cost of $6 \cdot \alpha = 6$.

Now, we observe that for any node v, the solution $\textsc{Off}(I)$ naturally induces feasible solutions $\textsc{Off}_v(I_v)$ for paging problem inputs I_v: these solutions simply repeat all actions of $\textsc{Off}$ pertaining to edges whose one endpoint is equal to v. By the forcing property of $\textsc{Off}$, the solutions $\textsc{Off}_v$ are feasible: upon request to node pair $e = (u, v)$, it is fetched to the cache of v. Furthermore, $\sum_{v \in V} \textsc{Off}_v(I_v) \leq 2 \cdot \textsc{Off}(I)$, as inclusion of $e = (u, v)$ to M in the solution of $\textsc{Off}$ leads to two fetches in the solutions of $\textsc{Off}_u$ and $\textsc{Off}_v$. (A more careful analysis including the evictions costs that are not present in the paging problem would show that $\sum_{v \in V} \textsc{Off}_v(I_v)$ and $\textsc{Off}(I)$ can differ at most by an additive constant independent of the input).

Finally, we observe that an optimal solution for I_v can only be cheaper than $\textsc{Off}(I)$, and therefore $\sum_{v \in V} \textsc{Opt}(I_v) \leq \sum_{v \in V} \textsc{Off}(I_v) \leq 2 \cdot \textsc{Off}(I) \leq 2 \cdot 7 \cdot \textsc{Opt}(I)$.

We now proceed to prove the first inequality. When a requested node pair $e = (u, v)$ is in matching M, then $\textsc{Alg}_1$ does nothing and no cost is incurred on $\textsc{Alg}_1$ or on any of algorithms A_v; hence, we may assume that $e \notin M$. In such a case, either e is not in the cache of u, or e is not in the cache of v, or e is in neither of these two caches. That is, either u or v, or both must fetch e to their caches, which causes $A_u(I_u) + A_v(I_v)$ to grow by 1 or 2. Moreover, paging algorithms running at u and v may evict some number of node pairs from their caches (these actions are free in the paging problem). In effect, $\textsc{Alg}_1$ pays for the request e (at cost 1), includes e in the matching M (at cost 1) and removes some number of edges from M.

Let $\textsc{Alg}_1^+(I)$ be the cost of $\textsc{Alg}_1$ on I neglecting the cost of removals of edges from M. The analysis above showed that increases of $\textsc{Alg}_1^+(I)$ by 2 can be charged to the increases of $\sum_{v \in V} A_v(I_v)$ by at least 1, and thus $\textsc{Alg}_1^+(I) \leq 2 \cdot \sum_{v \in V} A_v(I_v)$. The proof of the first inequality follows by noting that the total number of removals from M is at most the total number of inclusions to M, and thus $\textsc{Alg}_1(I) \leq 2 \cdot \textsc{Alg}_1^+(I) \leq 4 \cdot \sum_{v \in V} A_v(I_v)$. □

2.3 Competitive ratio

Corollary 3. *There exists an $O(\gamma \cdot \log(b/(b-a+1)))$-competitive randomized algorithm R-BMA for the (b, a)-matching problem, for an arbitrary $\alpha \geq 1$ and arbitrary path lengths ℓ_e.*

Proof. Applying a $2 \cdot \ln(b/(b-a+1))$-competitive randomized algorithm for the paging problem [72] (better constant factors were achieved when $a = b$ [28, 50, 1]) to Theorem 2 yields an $O(\log(b/(b-a+1)))$-competitive randomized algorithm for the uniform variant of the (b, a)-matching problem.

Next, Theorem 1 shows how to create an $O(\gamma \cdot \log(b/(b-a+1)))$-competitive randomized algorithm R-BMA for arbitrary $\alpha \geq 1$ and arbitrary path lengths ℓ_e. □

2.4 Lower bound

We now show that our algorithm is asymptotically optimal by proving that the (b, a)-matching problem contains the (b, a)-paging problem with bypassing as a special case. In the bypassing variant of the (b, a)-paging problem, an algorithm does not have to fetch the requested item to the cache.

Lemma 1. *Assume that there exists a (deterministic or randomized) $f(b, a)$-competitive algorithm for the (b, a)-matching problem for some function $f : \mathbb{N}_{\geq 0} \times \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq} 0$. Then, there exists an (also deterministic or randomized) $4 \cdot f(b, a)$-competitive algorithm for the (b, a)-paging with bypassing.*

Proof. Let $\textsc{Alg}^{\text{M}}$ be an algorithm for the (b, a)-matching problem. Assume that an algorithm for (b, a)-paging with bypassing has to operate in a universe of $n > b$ items. To construct an algorithm for the paging problem, we thus create a star graph of $n + 1$ nodes $\{v_0, v_1, \ldots, v_n\}$ and set of n non-configurable links F connecting v_0 with all remaining nodes, each of length 1. Nodes $v_1, \ldots, v_n$ correspond to the universe of n items. For any paging request to an item v_i, $\textsc{Alg}^{\text{P}}$ generates a block of α requests to node pair $\{v_0, v_i\}$. $\textsc{Alg}^{\text{P}}$ internally runs $\textsc{Alg}^{\text{M}}$ on the star graph and repeats its choices: $\textsc{Alg}^{\text{P}}$ always caches the items that are connected by the matching edges to v_0 in the solution of $\textsc{Alg}^{\text{M}}$.

For now, we ignore the costs of removing edges from the matching. It is easy to observe that the cost of $\textsc{Alg}^{\text{P}}$ is at most $2/\alpha$ times larger than that of $\textsc{Alg}^{\text{M}}$. Furthermore, without loss of generality, for a block of requests to node pair $\{v_0, v_i\}$ an optimal algorithm for the (b, a)-matching either includes

$\{v_0, v_i\}$ in the matching right before the block or it does not change its matching at all. Thus the optimal solutions for the (b, a)-matching problem and (b, a)-paging problem coincide and their costs differ exactly by the factor of $1/\alpha$. Putting these bounds together, we obtain that $\textsc{Alg}^{\mathrm{P}}$ is $2 \cdot f(b, a)$-competitive for the (b, a)-paging problem with bypassing, ignoring the costs of matching removals, and thus at most $4 \cdot f(b, a)$-competitive when matching removals are taken into account. □

As noted by Epstein et al. [25] the bypassing variant is asymptotically equivalent to the non-bypassing one. Using the known lower bound for the (b, a)-paging problem [72] along with Lemma 1, we immediately obtain the following corollary.

Theorem 4. *The competitive ratio of any randomized algorithm for the (b, a)-matching problem is at least $\Omega(\log(b/(b - a + 1)))$. The results hold for an arbitrary α.*

3 EMPIRICAL EVALUATION

To complement our formal evaluation, and in particular the worst-case analysis above, we also performed an empirically study of the performance of our algorithm under real-world workloads from various datacenter operators. In particular, we benchmark our randomized algorithm against a state-of-the-art (deterministic) online b-matching algorithm [9] and against a maximum weight matching algorithm [29].

3.1 Methodology

Setup. We implemented all algorithms in Python leveraging the NetworkX library. For the implementation of the *Maximum Weight Matching* algorithm we used the algorithm provided by NetworkX, which is based on Edmond's *Blossom* algorithm [29]. Our experiments were run on a machine with two Intel Xeons E5-2697V3 SR1XF with 2.6 GHz, 14 cores each and 128 GB RAM. The host machine was running Ubuntu 20.04 LTS.

Simulation Workloads. Real-world datacenter traffic can vary significantly with respect to the spatial and temporal structure they feature, which depends on the application running [3]. Hence, our simulations are based on the following real-world datacenter traffic workloads from Facebook and Microsoft, which cover a spectrum of application domains.

- **Facebook** [61]: We use three different workloads, each from a different Facebook cluster. We use a batch processing trace from one of Facebook's *Hadoop* clusters, as well as traces from one of Facebook's database clusters, which serves SQL requests. Furthermore, we use traces from one of Facebook's *Web-Service* cluster.
- **Microsoft** [30]: This data set is simply a probability distribution, describing the rack-to-rack communication (a traffic matrix). In order to generate a trace, we sample from this distribution *i.i.d.* Hence, this trace does not contain any temporal structure by design (e.g., is not bursty) [3]. However, it is known that it contains significant spatial structure (i.e., is skewed).

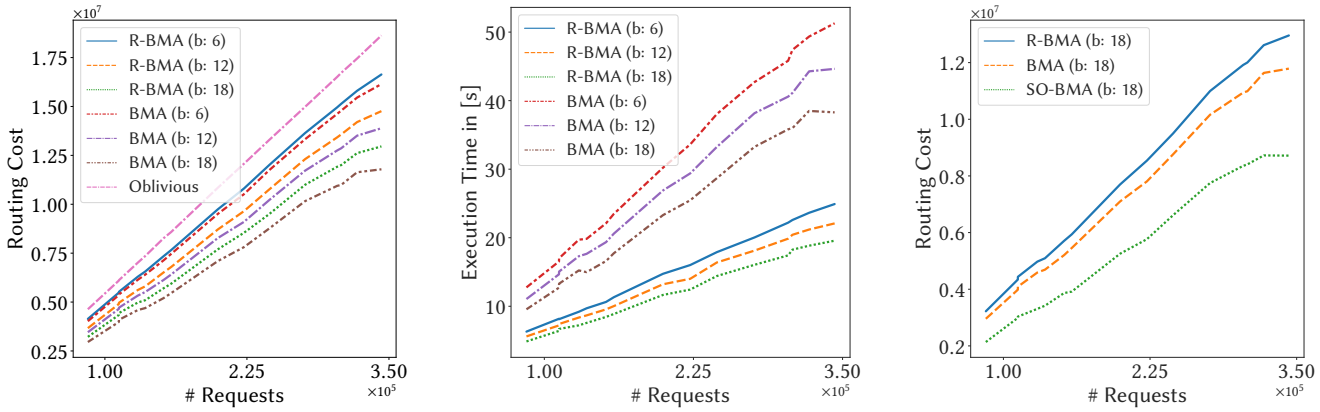
In all our simulations, we consider a typical fat-tree based datacenter topology, with 100 nodes in the case of the Facebook clusters, and with 50 nodes in the case of the Microsoft cluster. The cost of each request is calculated as the shortest path length between source and destination. Hence, if source and destination are connected by a reconfigurable link the cost equals 1. Otherwise, the routing cost is computed as the number of hops from source to destination. In general, each simulation is repeated five times and then the results are averaged.

3.2 Results and discussion

We discuss the main results of our simulations based on the traces introduced above. For each traffic trace we evaluate the routing cost and execution performance of our randomized b-matching algorithm. In particular, we evaluate the impact of different b (henceforth called cache size due to how these links are managed in our algorithm) and compare our randomized algorithm (R-BMA) to the performance of BMA [9] (BMA) and a *Maximum Weight Matching* algorithm (SO-BMA).

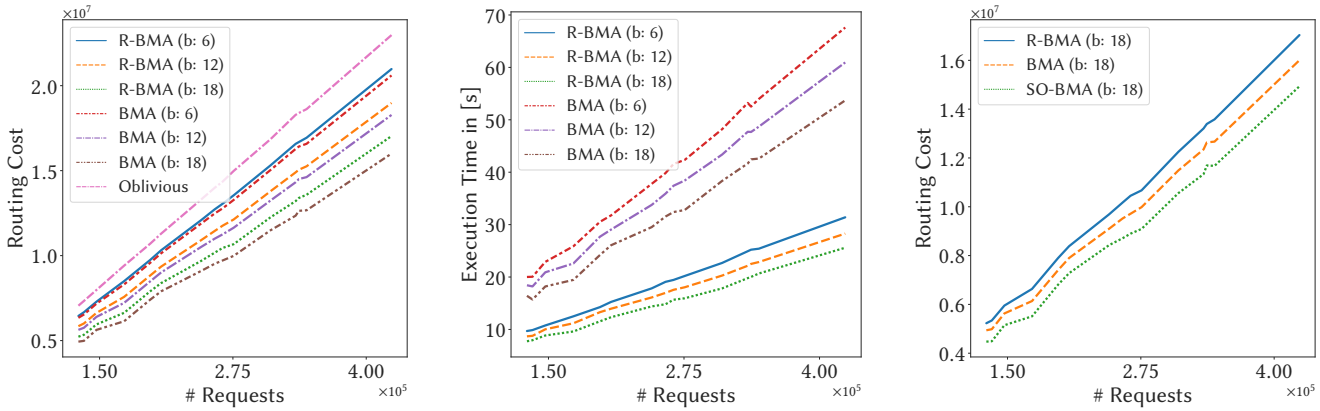
Routing Cost. We first discuss the observed routing cost. Fig. 1a shows the results of the Facebook database cluster. The violet line denotes the oblivious case, where each request is solely routed over the static network and no matching edges are present, e.g., a network without any reconfigurable switch. The results show that R-BMA achieves a significant routing cost reduction of up to 35% with a cache size of 18. In comparison to BMA, R-BMA performs almost identical to BMA on smaller cache sizes of 6. Fig. 1c shows that R-BMA routing cost reduction is not more than 5% higher on smaller numbers of requests, e.g., up to 200,000 requests. Still, in comparison to the SO-BMA, the gap with respect to the routing cost reduction widens as the number of requests grows. In contrast to the results achieved on Facebook's database cluster, Figs. 2c and 3c show that R-BMA achieves similar routing cost reductions compared to BMA and SO-BMA.

Regarding the Microsoft traces, we can observe that R-BMA achieves a similar routing cost reduction compared to BMA. Furthermore, as the cache size grows, R-BMA achieves the same routing cost reduction as BMA, as shown in Fig. 4a. Fig. 4c shows that SO-BMA performs significantly better than R-BMA. However, Microsoft's traces do not feature



(a) Routing costs. **(b) Execution time.** **(c) Best of comparison.**

Figure 1: Facebook Database cluster.



(a) Routing costs. **(b) Execution time.** **(c) Best of comparison.**

Figure 2: Facebook Web Service cluster.

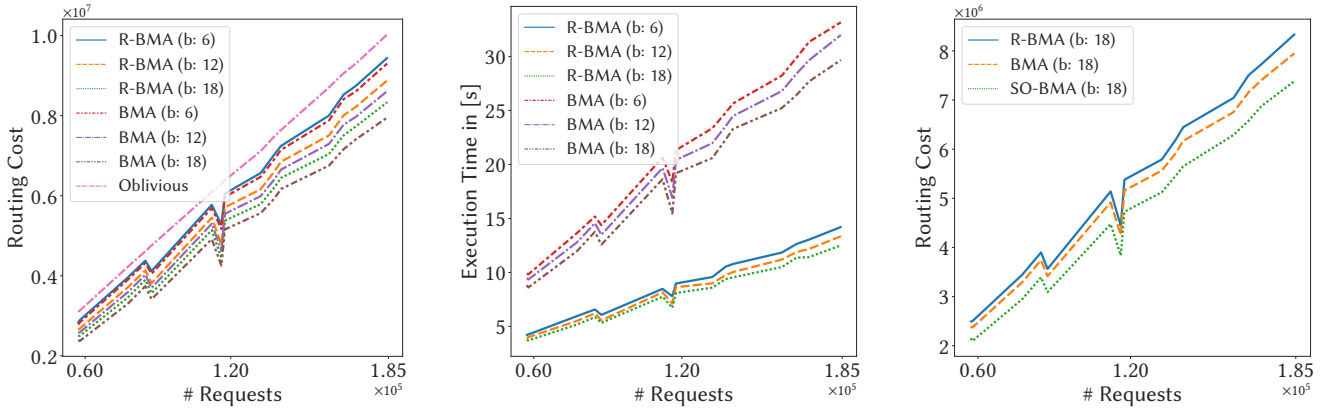
temporal structure and therefore offline algorithms such as SO-BMA have a significant advantage.

Execution Time. All our simulations on all different traces show that our R-BMA algorithm outperforms BMA [9] with respect to run-time efficiency. Furthermore, the size of the cache has a smaller impact on the execution time as for BMA. In particular, Figs. 1b, 2b, 3b show that a larger cache size can lead to a decrease in execution performance of up to 20% in the case of the BMA algorithm, whereas our randomized algorithm is comparatively more robust to a change in the size of the cache. The results of the Microsoft cluster in Fig. 4b also show that the run-time of our randomized algorithm grows slower than for BMA.

Summary. Our R-BMA algorithm achieves almost the same routing cost reduction as BMA, while also achieving competitive cost reductions compared to an optimal offline algorithm. With respect to the run-time efficiency of our randomized algorithm, we find that our algorithm significantly outperforms BMA on all workloads. We conclude that our algorithm, R-BMA, provides an attractive trade-off between routing cost reduction and run-time efficiency.

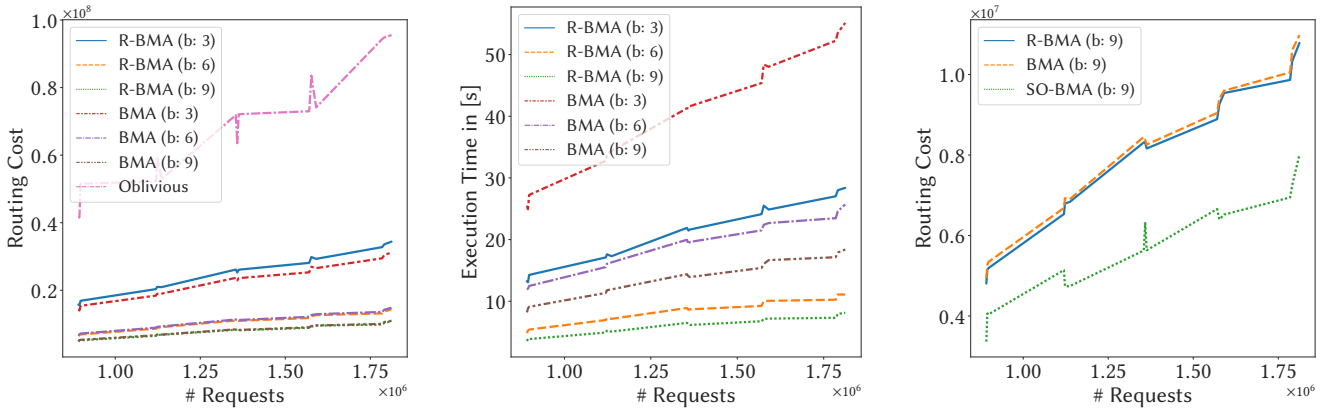
4 RELATED WORK

The design of datacenter topologies has received much attention in the networking community already. The most widely deployed networks are based on Clos topologies and multi-rooted fat-trees [26, 65, 47], and there are also interesting designs based on hypercubes [33, 71] and expander graphs [43, 66].



(a) Routing costs. **(b) Execution time.** **(c) Best of comparison.**

Figure 3: Facebook Hadoop cluster.



(a) Routing costs. **(b) Execution time.** **(c) Best of comparison.**

Figure 4: Microsoft cluster.

Existing dynamic and demand-aware datacenter networks can be classified according to the granularity of reconfigurations. Solutions such as Proteus [67], OSA [15], Cerberus [32], or DANs [4], among other, are more coarse-granular and e.g., rely on a (predicted) traffic matrix. Solutions such as ProjecToR [30, 45], MegaSwitch [17], Eclipse [69], Helios [27], Mordia [60], C-Through [70], ReNet [5] or SplayNets [62] are more fine-granular and support per-flow reconfiguration and decentralized reconfigurations. Reconfigurable demand-aware networks may also rely on expander graphs, e.g., Flexspander [68] and Kevin [74], and are currently also considered as a promising solution to speed up data transfers in supercomputers [36, 24]. The notion of demand-aware networks raise novel optimization problems related to switch scheduling [51], and recently interesting first insights have been obtained both for offline [69] and for online scheduling [64, 23, 45, 9]. Due to the increased reconfiguration time experienced in demand-aware networks, many existing demand-aware architectures additionally rely on a fixed network. For example, ProjecToR always maintains a "base mesh" of connected links that can handle low-latency traffic while it opportunistically reconfigures free-space links in response to changes in traffic patterns.

This paper primarily focuses on the *algorithmic* problems of demand-aware datacenter architectures. Our optimization problem is related to *graph augmentation* models, which consider the problem of adding edges to a given graph, so that path lengths are reduced. For example, Meyerson and Tagiku [56] study how to add "shortcut edges" to minimize the average shortest path distances, Bilò et al. [10] and Demaine and Zadimoghaddam [20] study how to augment a

network to reduce its diameter, and there are several interesting results on how to add "ghost edges" to a graph such that it becomes (more) "small world" [58, 59, 31]. However, these edge additions can be optimized globally and in a biased manner, and hence do not form a matching; we are also not aware of any online versions of this problem. The dynamic setting is related to classic switch scheduling problems [51, 18].

Regarding the specific b-matching problem considered in this paper, a polynomial-time algorithm for the static version of this problem is known for several decades [63, 2]. Recently, Hanauer et al. [38, 37] presented several efficient algorithms for the static [38] and dynamic (but offline) [37] problem variant for application in the context of reconfigurable datacenters. Bienkowski et al. [9] initiated the study of an online version of this problem and presented a $O(b)$-competitive deterministic algorithm and showed that this is asymptotically optimal. In this paper, we have shown that a randomized approach can provide a significantly lower competitive ratio as well as faster runtimes.

Finally, we note that there is a line of papers studying (bipartite) online matching variants [42, 11, 13, 21, 22, 49, 53, 57]. This problem attracted significant attention in the last decade because of its connection to online auctions and the AdWords problem [52]. Despite similarity in names (e.g., the bipartite (static) b-matching variant was considered in [40]), this model is fundamentally different from ours.

5 CONCLUSION

We revisited the problem of how to schedule reconfigurable links in a datacenter (based on optical circuit switches or similar technologies), in order to maximize network utilization. To this end, we presented a randomized online algorithm which computes heavy matchings between, e.g., datacenter racks, guaranteeing a significantly lower competitive ratio and faster running time compared to the state-of-the-art (and asymptotically optimal) deterministic algorithm. Our algorithm and its analysis are simple, and easy to implement (and teach).

That said, our work leaves open several interesting avenues for future research. In particular, we still lack a non-asymptotic and tight bound on the achievable competitive ratio both in the deterministic and in the randomized case. Furthermore, we so far assumed a conservative online perspective, where the algorithm does not have any information about future requests. In practice, traffic often features temporal structure, and it would be interesting to explore algorithms which can leverage certain predictions about future demands, without losing the worst-case guarantees.

ACKNOWLEDGMENTS

Research supported by the European Research Council (ERC), consolidator project Self-Adjusting Networks (AdjustNet), grant agreement No. 864228.

REFERENCES

[1] Dimitris Achlioptas, Marek Chrobak, and John Noga. 2000. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234, 1–2, 203–218.

[2] Richard P Anstee. 1987. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24, 3, 153–157.

[3] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the complexity of traffic traces and implications. In *Proc. ACM SIGMETRICS*.

[4] Chen Avin, Kaushik Mondal, and Stefan Schmid. 2017. Demand-aware network designs of bounded degree. In *Proc. International Symposium on Distributed Computing (DISC)*.

[5] Chen Avin and Stefan Schmid. 2021. Renets: statically-optimal demand-aware networks. In *Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*.

[6] Hitesh Ballani et al. 2020. Sirius: a flat datacenter network with nanosecond optical switching. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 782–797.

[7] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. 2012. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59, 4, 19:1–19:24. DOI: 10.1145/2339123.2339126.

[8] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 267–280.

[9] Marcin Bienkowski, David Fuchssteiner, Jan Marcinkowski, and Stefan Schmid. 2020. Online dynamic b-matching with applications to reconfigurable datacenter networks. In *Proc. 38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE)*.

[10] Davide Bilò, Luciano Gualà, and Guido Proietti. 2012. Improved approximability and non-approximability results for graph diameter decreasing problems. *Theoretical Computer Science*, 417, 12–22.

[11] Benjamin Birnbaum and Claire Mathieu. 2008. On-line bipartite matching made simple. *SIGACT News*, 39, 1, 80–87.

[12] Allan Borodin and Ran El-Yaniv. 1998. *Online Computation and Competitive Analysis*. Cambridge University Press.

[13] Niv Buchbinder, Kamal Jain, and Joseph Naor. 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proc. 15th European Symp. on Algorithms (ESA)*, 253–264.

[14] Jivitej S. Chadha, Naveen Garg, Amit Kumar, and V. N. Muralidhara. 2009. A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, 679–684. DOI: 10.1145/1536414.1536506.

[15] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen. 2014. Osa: an optical switching architecture for data center networks with unprecedented flexibility. *IEEE/ACM Transactions on Networking*, 22, 2, (Apr. 2014), 498–511. DOI: 10.1109/TNET.2013.2253120.

[16] 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 Transactions on Networking*, 22, 2, 498–511.

[17] 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 *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation* (NSDI'17). USENIX Association, Boston, MA, USA, 577–593. ISBN: 9781931971379.

[18] Shang-Tse Chuang, Ashish Goel, Nick McKeown, and Balaji Prabhakar. 1999. Matching output queueing with a combined input/output-queued switch. *IEEE Journal on Selected Areas in Communications*, 17, 6, 1030–1039.

[19] Daniel D. Sleator and Robert E. Tarjan. 1985. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28, 2, 202–208.

[20] Erik D Demaine and Morteza Zadimoghaddam. 2010. Minimizing the diameter of a network using shortcut edges. In *Proc. Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, 420–431.

[21] Nikhil R. Devanur and Kamal Jain. 2012. Online matching with concave returns. In *Proc. 44th ACM Symp. on Theory of Computing (STOC)*, 137–144.

[22] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. 2013. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proc. 24th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 101–107.

[23] M. Dinitz and B. Moseley. 2020. Scheduling for weighted flow and completion times in reconfigurable networks. In *IEEE Conference on Computer Communications (INFOCOM)*, 1043–1052.

[24] Fred Douglis, Seth Robertson, Eric Van den Berg, Josephine Micallef, Marc Pucci, Alex Aiken, Maarten Hattink, Mingoo Seok, and Keren Bergman. 2021. Fleet—fast lanes for expedited execution at 10 terabits: program overview. *IEEE Internet Computing*.

[25] Leah Epstein, Csanád Imreh, Asaf Levin, and Judit Nagy-György. 2011. On variants of file caching. In *Proc. 38th Int. Colloq. on Automata, Languages and Programming (ICALP)*. Luca Aceto, Monika Henzinger, and Jiri Sgall, (Eds.) Springer, 195–206. DOI: 10.1007/978-3-642-22006-7_17.

[26] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review* number 4. Vol. 38. ACM, 63–74.

[27] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2011. Helios: a hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 41, 4, 339–350.

[28] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel D. Sleator, and Neal E. Young. 1991. Competitive paging algorithms. *Journal of Algorithms*, 12, 4, 685–699.

[29] Zvi Galil. 1986. Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.*, 18, 1, (Mar. 1986), 23–38.

[30] Monia Ghobadi et al. 2016. Projector: agile reconfigurable data center interconnect. In *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 216–229.

[31] Andrew Gozzard, Max Ward, and Amitava Datta. 2018. Converting a network into a small-world network: fast algorithms for minimizing average path length through link addition. *Information Sciences*, 422, 282–289.

[32] Chen Griner, Johannes Zerwas, Andreas Blenk, Manya Ghobadi, Stefan Schmid, and Chen Avin. 2022. Cerberus: the power of choices in datacenter topology design (a throughput perspective). In *Proc. ACM SIGMETRICS*.

[33] 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. *ACM SIGCOMM Computer Communication Review*, 39, 4, 63–74.

[34] Matthew Nance Hall, Klaus-Tycho Foerster, Stefan Schmid, and Ramakrishnan Durairajan. 2021. A survey of reconfigurable optical networks. In *Optical Switching and Networking (OSN), Elsevier*.

[35] Navid Hamedazimi, Zafar 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 *ACM SIGCOMM Computer Communication Review* number 4. Vol. 44. ACM, 319–330.

[36] Michelle Hampson. 2021. Reconfigurable optical networks will move supercomputerdata 100x faster. In *IEEE Spectrum*.

[37] Kathrin Hanauer, Monika Henzinger, Lara Ost, and Stefan Schmid. 2023. Dynamic demand-aware link scheduling for reconfigurable datacenters. In *IEEE INFOCOM*.

[38] Kathrin Hanauer, Monika Henzinger, Stefan Schmid, and Jonathan Trummer. 2022. Fast and heavy disjoint weighted matchings for demand-aware datacenter topologies. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*.

[39] Bala Kalyanasundaram and Kirk Pruhs. 2000. Speed is as powerful as clairvoyance. *J. ACM*, 47, 4, 617–643. DOI: 10.1145/347476.347479.

[40] Bala Kalyanasundaram and Kirk R Pruhs. 2000. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233, 1-2, 319–325.

[41] Srikanth Kandula, Jitendra Padhye, and Paramvir Bahl. 2009. Flyways to de-congest data center networks. In *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*.

[42] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. 1990. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd ACM Symp. on Theory of Computing (STOC)*, 352–358.

[43] Simon Kassing, Asaf Valadarsky, Gal Shahaf, Michael Schapira, and Ankit Singla. 2017. Beyond fat-trees without antennae, mirrors, and disco-balls. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 281–294.

[44] Wolfgang Kellerer, Patrick Kalmbach, Andreas Blenk, Arsany Basta, Martin Reisslein, and Stefan Schmid. 2019. Adaptable and data-driven softwarized networks: review, opportunities, and challenges. *Proceedings of the IEEE*, 107, 4, 711–731.

[45] Janardhan Kulkarni, Stefan Schmid, and Pawel Schmidt. 2021. Scheduling opportunistic links in two-tiered reconfigurable datacenters. In *33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*.

[46] Yuliang Li et al. 2019. Hpcc: high precision congestion control. In *Proceedings of the ACM Special Interest Group on Data Communication*, 44–58.

[47] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas E. Anderson. 2013. F10: A fault-tolerant engineered network. In *NSDI*, 399–412.

[48] Yunpeng James Liu, Peter Xiang Gao, Bernard Wong, and Srinivasan Keshav. 2014. Quartz: a new design element for low-latency dcns. *SIGCOMM Comput. Commun. Rev.*, 44, 4, (Aug. 2014), 283–294. DOI: 10.1145/2740070.2626332.

[49] Mohammad Mahdian and Qiqi Yan. 2011. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, 597–606.

[50] Lyle A. McGeoch and Daniel D. Sleator. 1991. A strongly competitive randomized paging algorithm. *Algorithmica*, 6, 6, 816–825.

[51] Nick McKeown. 1999. The islip scheduling algorithm for input-queued switches. *IEEE/ACM transactions on networking*, 7, 2, 188–201.

[52] Aranyak Mehta. 2013. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science*, 8, 4, 265–368.

[53] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. 2007. Adwords and generalized online matching. *Journal of the ACM*, 54, 5.

[54] William M Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C Snoeren, and George Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, 1–18.

[55] 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 *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 267–280.

[56] Adam Meyerson and Brian Tagiku. 2009. Minimizing average shortest path distances via shortcut edge addition. In *Proc. Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, 272–285.

[57] Joseph Naor and David Wajc. 2015. Near-optimum online ad allocation for targeted advertising. In *Proc. 16th ACM Conf. on Economics and Computation (EC)*, 131–148.

[58] Manos Papagelis, Francesco Bonchi, and Aristides Gionis. 2011. Suggesting ghost edges for a smaller world. In *Proc. 20th ACM International Conference on Information and Knowledge Management (CIKM)*, 2305–2308.

[59] Nikos Parotsidis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2015. Selecting shortcuts for a smaller world. In *Proc. SIAM International Conference on Data Mining*, 28–36.

[60] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. Integrating microsecond circuit switching into the data center. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (SIGCOMM '13). Association for Computing Machinery, Hong Kong, China, 447–458. ISBN: 9781450320566. DOI: 10.1145/2486001.2486007.

[61] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. 2015. Inside the social network's (datacenter) network. In *ACM SIGCOMM Computer Communication Review*. Vol. 45, 123–137.

[62] Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker. 2016. Splaynet: towards locally self-adjusting networks. *IEEE/ACM Transactions on Networking (ToN)*, 24, 3, 1421–1433.

[63] Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency. Algorithms and combinatorics*. Springer.

[64] Roy Schwartz, Mohit Singh, and Sina Yazdanbod. 2019. Online and offline greedy algorithms for routing with switching costs. *arXiv preprint arXiv:1905.02800*.

[65] Arjun Singh et al. 2015. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *Computer Communication Review*, 45, 5, 183–197.

[66] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. 2012. Jellyfish: networking data centers randomly. In *NSDI*, 225–238.

[67] Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, and Yueping Zhang. 2010. Proteus: a topology malleable data center network. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 8.

[68] Min Yee Teh, Zhenguo Wu, and Keren Bergman. 2020. Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering. *IEEE/OSA Journal of Optical Communications and Networking*, 12, 4, B44–B54.

[69] Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, and Pramod Viswanath. 2018. Costly circuits, submodular schedules and approximate carathéodory theorems. *Queueing Systems*, 88, 3-4, 311–347.

[70] Guohui Wang, David G Andersen, Michael Kaminsky, Konstantina Papagiannaki, TS Ng, Michael Kozuch, and Michael Ryan. 2011. C-through: part-time optics in data centers. *ACM SIGCOMM Computer Communication Review*, 41, 4, 327–338.

[71] Haitao Wu, Guohan Lu, Dan Li, Chuanxiong Guo, and Yongguang Zhang. 2009. Mdcube: a high performance network structure for modular data center interconnection. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 25–36.

[72] Neal E. Young. 1991. On-line caching as cache size varies. In *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California, USA*. Alok Aggarwal, (Ed.), 241–250.

[73] Neal E. Young. 1994. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11, 6, 525–541. DOI: 10.1007/BF01189992.

[74] Johannes Zerwas, Csaba Györgyi, Andreas Blenk, Stefan Schmid, and Chen Avin. 2022. Kevin: de brujin-based topology with demand-aware links and greedy routing. *arXiv preprint arXiv:2202.05487*.

[75] 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. *ACM SIGCOMM Computer Communication Review*, 42, 4, 443–454.