Toward Self-Adjusting k-ary Search Tree Networks

Evgeniy Feder
ITMO University
Saint-Petersburg, Russia
dzaba67@yandex.ru

Anton Paramonov
EPFL
Lausanne, Switzerland
anton.paramonov2000@gmail.com

Pavel Mavrin
Neapolis University Pafos
Pafos, Cyprus
pavel.mavrin@gmail.com

Iosif Salem
TU Berlin
Berlin, Germany
iosif.salem@inet.tu-berlin.de

Stefan Schmid
TU Berlin
Berlin, Germany
stefan.schmid@tu-berlin.de

Vitaly Aksenov
City, University of London
London, UK
aksenov.vitaly@gmail.com

***Abstract*—Datacenter networks are becoming increasingly flexible with the incorporation of new optical communication technologies, such as optical circuit switches, enabling self-adjusting topologies that can adapt to the traffic pattern in a demand-aware manner. In this paper, we take the first steps toward demand-aware and self-adjusting k-ary tree networks. These are more powerful generalizations of existing binary search tree networks (like SplayNet [14]), which have been at the core of self-adjusting network (SAN) designs. k-ary search tree networks are a natural generalization offering nodes of higher degrees, reduced route lengths, and local routing in spite of reconfigurations (due to maintaining the search property). Our main results are algorithms for static k-ary tree networks and two online heuristics for self-adjusting k-ary tree networks.**

***Index Terms*—self-adjusting networks, k-ary trees, online algorithms, dynamic programming**

I. Introduction

With more services being offloaded to the cloud, inter-datacenter traffic grows exponentially, leading to new approaches in a network design. Traditional designs are static and perform well only under certain workloads (e.g., all-to-all). However, datacenter traffic follows patterns which can be exploited in the design of more efficient networks [2].

As a result, innovative dynamic datacenter network topologies have emerged: physical networks now have the ability to *self-adjust*, e.g., [10], [12]. This flexibility raises an optimization challenge: how to optimally adjust the topology to improve network efficiency in terms of routing cost? There is a trade-off between the cost of changing the network topology (reconfiguration cost) and the benefit of reducing the distance of frequently communicating racks, henceforth called nodes (routing cost). In the static case, we aim at computing an optimal demand-aware network topology with low time complexity. In the online case, where the future communication demand is unknown, we would opt for topology changes that are likely to pay off in the future.

The developing field of Self-Adjusting Networks (SANs) aims to address these optimization challenges. SANs often assume a family of allowed topologies, e.g., trees [14], [11], [6], [9], [1], skip lists [4], bounded degree graphs [5], [3], etc., within which the network has to remain across topology adjustments. SplayNet [14], a self-adjusting binary search tree network generalizing splay trees [16], was the first proposed SAN. SplayNet has been extended to ReNet [5], a statically optimal SAN for sparse communication patterns, but also to a distributed version, DiSplayNet [11]. The search property is particularly useful for SANs, since maintaining it allows local and greedy routing despite changes in the topology.

In this work, we take the first steps to generalizing binary to k-ary search tree networks. We present offline static and online self-adjusting k-ary search tree networks. We note that designing k-ary search tree networks is different than designing k-ary search trees since in a network each node should store only one key-identifier; the assignment of identifiers to nodes is a bijection and does not change. In the case of k-ary search trees keys are used for routing (traversing from root to searched key) and serve as the elements of tree nodes. In contrast, in the network case we can use the nodes' keys for routing (hence called routing keys), but each node should have an extra fixed key that serves as its identifier (node key). Self-adjusting k-ary search trees have been studied by Sherk in [15]. However, that approach does not apply in our case due to the requirement of having one key-identifier per node.

II. Results

In this section we give an overview of our results. For more details, we refer to the technical report [8].

A. Model

We consider a network of n nodes and a communication sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$, where $\sigma_t = (u, v)$ is a communication request from source u to destination v. The network topology G must be chosen from a family of desired topologies, i.e., k-ary search tree in our case. The topology can be reconfigured between requests with a cost equal to the number of links (edges) added or removed. The total service cost of σ is the sum of routing and reconfiguration costs. Our goal is to serve the communication sequence with the minimum total cost.

We distinguish two problem variants. In the **offline static** variant, σ is represented as a demand matrix D

and no reconfiguration can occur. The total cost is just the scalar product of the demand matrix and the distances in the network: $\sum d(u,v) \cdot D[u,v]$. In the **online self-adjusting** variant, σ is not known in advance and we can change the topology after serving a request. The total cost is $\sum_{i=1}^{m} (\text{routingCost}(G_{i-1}, \sigma_i) + \text{adjustmentCost}(G_{i-1}, G_i))$, where $\text{routingCost}(G_{i-1}, \sigma_i)$ is the length of the path in edges for the request σ_i in G_{i-1} and $\text{adjustmentCost}(G_{i-1}, G_i)$ is the adjustment cost to reconfigure the network from step $i-1$. This paper considers both problem variants when the set of allowed topologies is limited to the set of k-ary search trees.

Definition 1. *(i) A* k-ary Search Tree *is a rooted tree on keys (node identifiers) where each node stores a key (node identifier) and a routing array* $r = (r_1, r_2, \ldots, r_{k-1})$ *containing routing elements (not keys), while the subtrees satisfy the search property. Note that the key does not necessarily belong to the routing array.*

(ii) A routing-based k*-ary Search Tree is a* k*-ary search tree such that in each node its identifier* ***belongs*** *to the routing array.*

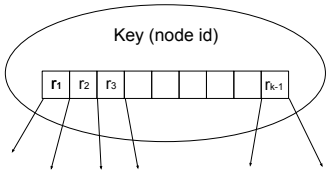

Fig. 1: Node in k-ary search tree

B. Optimal static k-ary search tree networks

Our first result is an algorithm that constructs an optimal static *routing-based* k-ary search tree network via dynamic programming in $\mathcal{O}(n^3 \cdot k)$ time. This approach is similar to the one used to build the optimal binary search tree network [14].

If we consider the special case of uniformly distributed traffic, i.e., each non-diagonal demand matrix element is one, the complexity can be reduced to $\mathcal{O}(n^2 \cdot k)$. However, the latter approach becomes impractical as the the number of nodes in the network is becomes larger, due to the quadratic complexity. For that, we present a centroid $(k+1)$-degree search tree that can be build in $\mathcal{O}(n)$ time and is almost-optimal. In this tree, the root has $k+1$ children of almost equal size which are full k-ary trees. Note that such a tree can be transformed to a k-ary search tree by rooting it by the leftmost leaf. We prove that this tree is *almost-optimal* and can be build in $\mathcal{O}(n)$ time, improving the $\mathcal{O}(n^3 \cdot k)$ dynamic profgramming algorithm. Intuitively, in k-ary search trees each node has $k+1$ neighbours except for the root meaning that we split the most loaded node only on k parts, while it is possible to have $k+1$. We experimentally calculated the cost and found that our centroid $(k+1)$-degree tree is indeed optimal for the uniform workload for all $n \leq 10^3$ and $k \leq 10$. We believe that these results are good evidence towards generalizing to the general case.

C. Online self-adjusting k-ary search tree networks

To solve the online problem, we present a novel k-ary SplayNet data structure. As the original SplayNet [14], we base our algorithm on a k-ary version of SplayTree [16]. However, we cannot use the k-ary SplayTree proposed by Sherk [15] since in our problem each node has only one key-identifier. Thus, we designed new tree rotations named k-splay. As in the usual zig-zig or zig-zag splay function, we take three nodes and rearrange them while correctly redistributing the routing keys in a new structure. Given these new rotations, to perform a request we just k-splay both argument nodes to their lowest common ancestor.

We experimentally evaluated our heuristics, which we overview below. We use real [7], [10], [13] and synthetic workloads [2]. We show that our k-ary SplayNet outperforms a simple full k-ary tree almost on all workloads, while it is competitive against the static optimal k-ary tree, built by our dynamic programming algorithm, and outperforms it on the workloads with high temporal complexity.

As we found in the static case section, the centroid tree can serve the requests faster than the simple full k-ary tree, for example, on the uniform workload. Thus, it would be interesting to apply a centroid heuristic for k-ary SplayNet. This gives us $(k+1)$-SplayNet which structure is shown on the picture below:

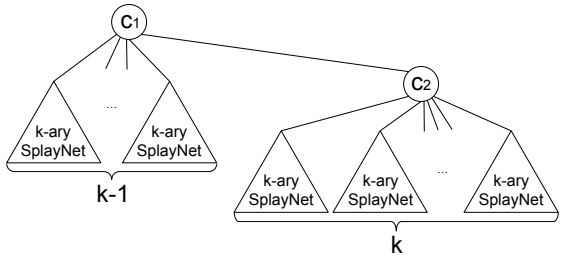

Fig. 2: $(k+1)$-SplayNet structure.

Note that centroid nodes c_1 and c_2 never move while nodes in the subtrees can. By running the experiments, we found that 3-SplayNet works better than the standard SplayNet on the workloads with low temporal locality, i.e., closer to the uniform workload, which was expected.

D. Future Work

We presented online and offline algorithms for k-ary search tree networks. However, there are some improvements that can be done. First, our static algorithm builds only the optimal routing-based k-ary search trees, while we would be interested in the optimal tree without any constraint. However, this problem may be NP-complete. Secondly, our centroid tree based $(k+1)$-SplayNet structure is too rigid — we need two centroid nodes that cannot move. The question is whether we can come up with smarter rotations. We believe that our work paves the way to new SANs for k-ary search tree networks for general and specific traffic patterns.

References

[1] Chen Avin, Marcin Bienkowski, Iosif Salem, Robert Sama, Stefan Schmid, and Paweł Schmidt. Deterministic self-adjusting tree networks using rotor walks. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 67–77. IEEE, 2022.

[2] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. On the complexity of traffic traces and implications. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(1):1–29, 2020.

[3] Chen Avin, Kaushik Mondal, and Stefan Schmid. Demand-aware network design with minimal congestion and route lengths. *IEEE/ACM Transactions on Networking*, 30(4):1838–1848, 2022.

[4] Chen Avin, Iosif Salem, and Stefan Schmid. Working set theorems for routing in self-adjusting skip list networks. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2175–2184. IEEE, 2020.

[5] Chen Avin and Stefan Schmid. Renets: Statically-optimal demand-aware networks. In *Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 25–39. SIAM, 2021.

[6] Otavio Augusto de Oliveira Souza, Olga Goussevskaia, and Stefan Schmid. Cbnet: Minimizing adjustments in concurrent demand-aware tree networks. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 382–391. IEEE, 2021.

[7] US DOE. Characterization of the doe mini-apps. https://portal.nersc.gov/project/CAL/doe-miniapps.htm, 2016.

[8] Evgenii Feder, Anton Paramonov, Iosif Salem, Stefan Schmid, and Vitaly Aksenov. Toward self-adjusting k-ary search tree networks. *arXiv preprint arXiv:2302.13113*, 2023.

[9] Evgeniy Feder, Ichha Rathod, Punit Shyamsukha, Robert Sama, Vitaly Aksenov, Iosif Salem, and Stefan Schmid. Lazy self-adjusting bounded-degree networks for the matching model. In *41th IEEE Conference on Computer Communications, INFOCOM 2020, Virtual Conference, May 2-5, 2022*. IEEE, 2022.

[10] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. Projector: Agile reconfigurable data center interconnect. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 216–229, 2016.

[11] Bruna Peres, Otavio Augusto de Oliveira Souza, Olga Goussevskaya, Chen Avin, and Stefan Schmid. Distributed self-adjusting tree networks. *IEEE Transactions on Cloud Computing*, 2021.

[12] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, et al. Jupiter evolving: Transforming google's datacenter network via optical circuit switches and software-defined networking. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 66–85, 2022.

[13] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 123–137, 2015.

[14] Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker. Splaynet: Towards locally self-adjusting networks. *IEEE/ACM Transactions on Networking*, 24(3):1421–1433, 2015.

[15] Murray Sherk. Self-adjusting k-ary search trees. *Journal of Algorithms*, 19(1):25–44, 1995.

[16] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *Journal of the ACM (JACM)*, 32(3):652–686, 1985.