Brief Announcement: Minimizing the Weighted Average Shortest Path Length in Demand-Aware Networks via Matching Augmentation

Aleksander Figiel
TU Berlin
Germany

Darya Melnyk
TU Berlin
Germany

André Nichterlein
TU Berlin
Germany

Arash Pourdamghani
TU Berlin
Germany

Stefan Schmid
TU Berlin
Germany

ABSTRACT

Graph augmentation is a fundamental and well-studied problem that arises in network optimization. We consider a new variant of this model motivated by reconfigurable communication networks. In this variant, we differentiate between a given physical network and the measured communication demands between the nodes. Our goal is to minimize the weighted average shortest path length via matching augmentation, where the weights correspond to the communication frequency of any pair of nodes. We use results from demand-aware network design to provide a constant-factor approximation algorithm for adding a matching on a ring in case only a few nodes in the network cause almost all the communication. Since the problem is NP-hard, we design and evaluate a series of heuristics that can deal with arbitrary graphs as underlying network structures. We evaluate our heuristics on general real-world communication patterns and show that already with simple and efficient heuristics we are able to reach near-optimal quality.

CCS CONCEPTS

• **Theory of computation → Theory and algorithms for application domains**.

KEYWORDS

Matching augmentation; demand-awareness; network design

ACM Reference Format:
Aleksander Figiel, Darya Melnyk, André Nichterlein, Arash Pourdamghani, and Stefan Schmid. 2024. Brief Announcement: Minimizing the Weighted Average Shortest Path Length in Demand-Aware Networks via Matching Augmentation. In *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3626183.3660264

1 INTRODUCTION

Graph augmentation is a fundamental problem in the area of network design. Consider a communication network where the nodes are arranged in a graph and communicate with their peers via the links of the graph. If we add new links to the network, we can hope for faster communication and lower congestion in the new network. Graph augmentation for network design has been studied under two optimization criteria: one is to optimize the worst-case communication cost which is done by minimizing the diameter of the network [1, 18]. The other criterion is to optimize the average-case communication, where the average shortest path length is minimized between any two peers [8, 11–14]. However, previous work has mostly assumed that the communication between the nodes is evenly distributed, i. e., that any two nodes are equally likely to communicate, independent of their role in the network.

Our work is motivated by the recent studies of deploying optical circuit switches to construct next-generation datacenter technologies [6, 9, 17]. The idea is to make the network demand-aware by using reconfigurable optical switches to provide shorter paths to a small number of large data flows, on top of an existing datacenter network based on electrical switches. By offloading considerable amounts of traffic to optical switches, bandwidth consumption and congestion on the original network can be reduced. Optical switches are deployed between an output port and an input port of electrical switch pair (e.g., the top-of-rack switches in datacenters). The connections can be reconfigured quickly, thus providing the option for a network to adapt to the network demand over time.

In this paper, we study how to enhance a graph with a matching. In particular, we focus on the most fundamental question: adding a single matching. Our goal is to compute a matching that minimizes the average path length in the augmented network in a demand-aware manner, that is, weighted by the communication demand between two nodes (rather than for all nodes like in previous work). This approach can be used to reconfigure the network after some time to adjust to the new demand. For our use cases, we consider fundamental physical networks of high (super-constant) diameter, such as d-dimensional grids, and tori, as they are often used in distributed architectures.

Our main **contributions** is studying the problem of Minimizing Weighted Average Shortest Path Length via Matching Addition (*MWASP*) and analyze its complexity and approximability. As it turns out to be NP-hard, we propose a constant-factor approximation algorithm for *MWASP* on a ring as the underlying infrastructure graph. This algorithm groups small segments of nodes in the ring into super-nodes and then connects these super-nodes with a known construction from demand aware networks. The idea behind the grouping is that nodes can help their neighbors of high demand connect to other high-demand nodes. This construction limits us to highly skewed demand matrices where only a few nodes in the network cause almost all the communication.

We complement our theoretical results with heuristics for *MWASP*. Our experimental evaluation demonstrates the efficiency and effectiveness of our heuristics and our approximation algorithm. We show that there exists a fast heuristic that performs close to optimal considering various parameters, and providing insights into cases that each algorithm can perform the best.

2 MODEL

We are given a set of nodes $V = \{v_1, \ldots, v_n\}$ that communicate over an underlying *infrastructure graph* G, this graph corresponds to the physical network. The infrastructure graph is assumed to have a non-constant diameter and a large number of nodes n, where n is even (so that there always exists a perfect matching). The communication pattern is described by an $n \times n$ *demand matrix* D. In this matrix, $D_{u,v}$ indicates the frequency with which a node u communicates to a node v. Observe that the demand matrix is normalized, i.e., $\sum_{u,v \in V, u \neq v} D_{u,v} = 1$, and the demand from one node to itself is 0, i.e., $D_{v,v} = 0 \; \forall v \in V$. For simplicity, we assume that the demand matrix is symmetric. Thus, the demand matrix encodes an edge-weighted, simple, undirected graph. We use $\text{dist}_G(u, v)$ to denote the *distance* (the length of a shortest path between) u and v in G. Our objective is to minimize the weighted average shortest path length $\text{Obj}_D(G) = \sum_{u,v \in V} D_{u,v} \cdot \text{dist}_G(u, v)$ in G for the given demand matrix D.

Our goal is to add a perfect matching M to the set of edges of G that minimizes the weighted average shortest path length in this augmented graph $G + M$. We consider the case where the added matching edges behave the same as the edges of the underlying graph, i.e., they are indistinguishable from the edges of the infrastructure graph in terms of their weight and length. We now define Minimizing Weighted Average Shortest Path Length via Matching Addition (*MWASP*) as finding for a given graph G a matching M that minimizes $\text{Obj}_D(G + M)$.

3 APPROXIMATION ALGORITHM

In this section, we discuss the main algorithm of this paper, which provably retains a constant factor approximation for a class of demand matrices. The reason for turning to approximation is that *MWASP* is an algorithmically challenging problem:

Theorem 3.1. MWASP *is NP-hard, even if the underlying graph is a cycle and every row and column of D has at most two non-zero elements.*

Algorithm 1: Pseudocode for the SuperDAN algorithm.

```
Input: A ring $G = (V, E)$ and demand $D$ with average
       degree $\leq 1$, $\alpha = 12$
1 $D_S, f_S \leftarrow$ super graph after merging $\alpha$ consecutive nodes
2 $H_S \leftarrow$ DAN with $\Delta_{\text{avg}}(H_S) \leq \alpha$ for $D_S$ using [3]
3 $M \leftarrow \emptyset$; $U \leftarrow \emptyset$
4 forall $\{a, b\} \in E(H_S)$ do
5     pick any $u \in f_S^{-1}(a) \setminus U$ and $v \in f_S^{-1}(b) \setminus U$
6     $M \leftarrow M \cup \{\{u, v\}\}$
7     $U \leftarrow U \cup \{u, v\}$
8 return $M$
```

The main idea behind our algorithm, SuperDAN[*], is to design a higher degree network first and then transferring that result into matching.

Super-node creation. From our infrastructure graph G we create a *super-graph* S by iteratively turning all nodes of G into super-nodes. A super-node is a collection of nodes of an underlying graph.

With G being a ring, we turn it into a super-ring S within $\frac{n}{\alpha}$ steps. In the step i, we contract the consecutive nodes $v_{i \cdot \alpha + 1}, \ldots v_{(i+1) \cdot \alpha}$ into super-node s_i. We define $f_S \colon V(G) \mapsto V(S)$ to be the mapping between nodes and corresponding super-nodes. The demand D_S on the super-graph is

$$D_S(a, b) = \sum_{\substack{u,v \in V \\ f_S(u)=a, f_S(v)=b}} D_{u,v}$$

In our algorithm, we set $\alpha = 12$. For simplicity, we assume n to be divisible by α. Note that this assumption could be avoided with a more sophisticated creation of the super-nodes that ignores some nodes without communication demand.

DAN on super-graph. After creating the super-graph, we use the construction of [3] to build a demand-aware network[†] (DAN) with a degree at most $12\,\Delta_{\text{avg}}$ on top of the super-graph, where Δ_{avg} denotes the average degree of the super-demand-graph. A recent paper [7] improved this bound to $4\,\Delta_{\text{avg}} + 1$.

Matching assignment inside a super-node. We now transform the edges of the DAN on top of super-nodes into a matching on the original graph. Note that the constructed DAN has maximum degree $12\,\Delta_{\text{avg}} \leq 12$, by assumption. Thus, for each super-node v we can map each incident edge to a different node in $f_S^{-1}(v)$ and obtain a matching in G. See Algorithm 1 for an overview on SuperDAN.

Theorem 3.2. *Given a ring graph G with a demand graph D of average degree at most $\frac{1}{\alpha}$, then for $\alpha = 12$* SuperDAN *computes in $O(n^2 \cdot \log n)$ time a matching M for which*

$$\text{Obj}_D(G + M) \leq c \cdot \min_{\substack{\text{matching } M' \\ \text{on } V(G)}} \text{Obj}_D(G + M')$$

for some non-negative constant c that depends only on α.

[*]We choose this name beacuse our algorithm combines the ideas of super-graph creation and then building a demand-aware network on top of it.

[†]This is a graph on the super-nodes satisfying the communication demands.

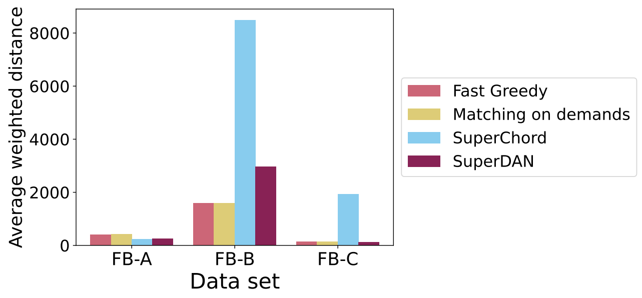

Figure 1: This considers the three Facebook data set instances [16], and shows the cost of our fastest algorithms on it. We choose these four algorithms given the sheer size of the data sets, which have $13,733$, $18,897$, $27,358$ **nodes respectively.**

We remark that Theorem 3.2 would be able to achieve $\alpha = 5$ if we replace the algorithm from Avin et al. [3] by the algorithm recently proposed in [7]; the proof is analogous.

4 HEURISTICS

In this section, we discuss some of our heuristics.

`SuperChord`. Based on the idea of creating super-graphs, we aim to benefit from the well-known Chord [19] protocol. To this end, we create a super-graph by combining $x = \frac{W(n \cdot \ln 2)}{\ln 2}$ consecutive nodes, where W represents Lambert W function, and lg the natural logarithm function[‡]. We then build the Chord on the $\frac{n}{x}$ super-nodes of the super-graph. The resulting graph has degree $\log(\frac{n}{x})$. We set $x = \frac{W(n \cdot \ln 2)}{\ln 2}$ as it ensures $x = \log(\frac{n}{x})$. This allows us to view the $\log(\frac{n}{x})$ outgoing edges of a super-node as matching edges initiated from the nodes within the super-node.

We remark that the Chord algorithm ensures a $\log n$ diameter (considering degree $\log n$) [19]. As the size of super-nodes is $O(\log n)$, we can ensure that any two nodes can reach each other via a shortest path of length $O(\log n)$, which in turn implies that `SuperChord` is an $O(\log n)$ approximation.

`Matching on demands`. This algorithm uses the *demand graph* that we introduced before, in which each edge is weighted by its demand, except the infrastructure edges. The algorithm then considers the maximum matching on the demand graph as its output. The running time of this algorithm is $O(n^3)$, given the currently best maximum weighted matching algorithm [5].

`Fast greedy`. The fast greedy algorithm starts by sorting all pairs of nodes based on their demand in descending order. Starting with the edge of highest demand, it considers the next valid edge in the sorted list as the next matching edge.

5 EXPERIMENTAL EVALUATION

To evaluate our results, we consider *real-world* instances. By doing so, we show the benefits of each algorithm in the wild.

[‡]If the number of super-nodes is not a power of two, then we consider the super-graph with closest and smaller power of two as the number of super-nodes.

Real-world demands. Our algorithms have been tested on a range of real-world datacenter traces [2] that has been the base of comparison for many previous works [4, 10, 15]. In particular, we focused on the Meta (formerly known as Facebook) dataset [16]. This dataset contains communication between racks and servers within three data center clusters (Database, Web Services, and Hadoop, sorted by their number of nodes) which we call A to C respectively. We focus on the communications between racks. Furthermore, we summarize each dataset into a list, in which the frequency of communication between each rack pair is listed.

Results. Our results indicate that our algorithms using super-graphs, in particular `SuperDAN`, can be a good option to reduce the cost in real-world instances, in conjunction with other greedy algorithms. See Figure 1 for the results.

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the Grant agreement No. 864228 (AdjustNet), 2020-2025.

REFERENCES

[1] Florian Adriaens and Aristides Gionis. 2022. Diameter Minimization by Shortcutting with Degree Constraints. In *ICDM*.
[2] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the Complexity of Traffic Traces and Implications. In *ACM SIGMETRICS*.
[3] Chen Avin, Kaushik Mondal, and Stefan Schmid. 2020. Demand-aware network designs of bounded degree. *Distributed Computing* (2020).
[4] Marcin Bienkowski, David Fuchssteiner, and Stefan Schmid. 2023. Optimizing Reconfigurable Optical Datacenters: The Power of Randomization. In *SC*. ACM.
[5] Ran Duan and Seth Pettie. 2014. Linear-Time Approximation for Maximum Weight Matching. *J. ACM* (2014).
[6] 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 CCR* (2011).
[7] Aleksander Figiel, Janne H Korhonen, Neil Olver, and Stefan Schmid. 2023. Demand-Aware Network Design with Steiner Nodes and a Connection to Virtual Network Embedding. *arXiv preprint arXiv:2308.10579* (2023).
[8] 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* (2018).
[9] Matthew Nance Hall, Klaus-Tycho Foerster, Stefan Schmid, and Ramakrishnan Durairajan. 2021. A Survey of Reconfigurable Optical Networks. In *OSN*.
[10] Kathrin Hanauer, Monika Henzinger, Stefan Schmid, and Jonathan Trummer. 2022. Fast and heavy disjoint weighted matchings for demand-aware datacenter topologies. In *IEEE INFOCOM*.
[11] Jon Kleinberg. 2000. The small-world phenomenon: an algorithmic perspective. In *STOC*.
[12] Adam Meyerson and Brian Tagiku. 2009. Minimizing Average Shortest Path Distances via Shortcut Edge Addition. In *RANDOM*.
[13] Manos Papagelis, Francesco Bonchi, and Aristides Gionis. 2011. Suggesting ghost edges for a smaller world. In *ACM CIKM*.
[14] Nikos Parotsidis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2015. Selecting shortcuts for a smaller world. In *SIAM SDM*.
[15] Arash Pourdamghani, Chen Avin, Robert Sama, and Stefan Schmid. 2023. SeedTree: A Dynamically Optimal and Local Self-Adjusting Tree. In *IEEE INFOCOM*. IEEE.
[16] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. 2015. Inside the social network's (datacenter) network. In *ACM SIGCOMM CCR*.
[17] Neta Rozen-Schiff, Klaus-Tycho Foerster, Stefan Schmid, and David Hay. 2023. Chopin: Combining Distributed and Centralized Schedulers for Self-Adjusting Datacenter Networks. In *OPODIS*.
[18] A. A. Schoone, H. L. Bodlaender, and J. Van Leeuwen. 1987. Diameter increase caused by edge deletion. *J. Graph Theory* (1987).
[19] Ion Stoica, Robert Tappan Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*.