Matching Augmentation in Demand-Aware Networks

Arash Pourdamghani

Joint work with Aleksander Figiel, Darya Melnyk, André Nichterlein and Stefan Schmid

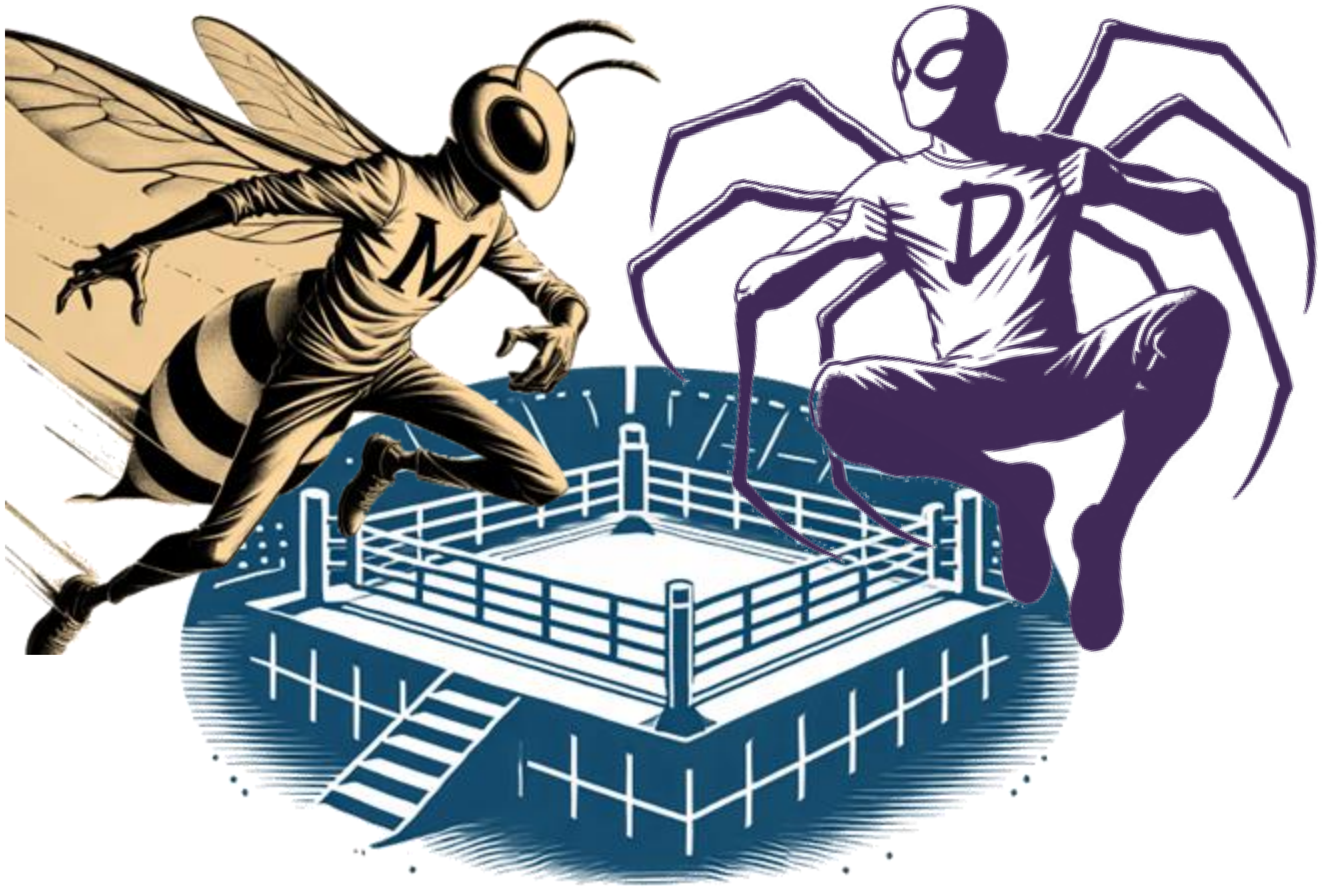
DISCOGA’25

(Based on ALENEX’25 presentations)

Or … The Tale of

MWASP

SpiderDAN

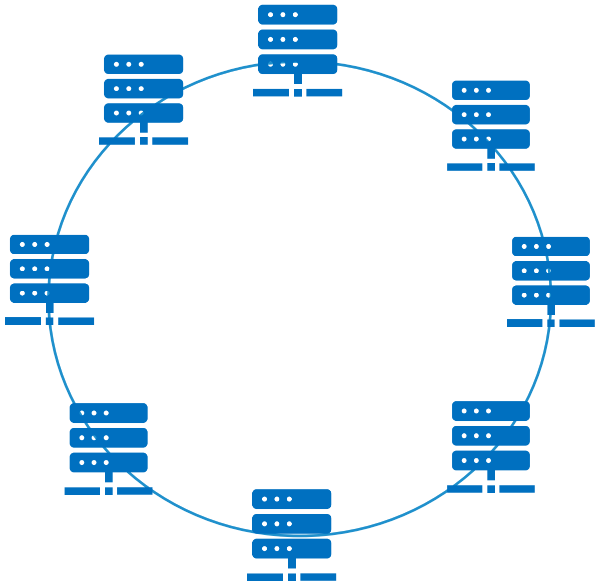


Or … The Tale of

MWASP SpiderDAN

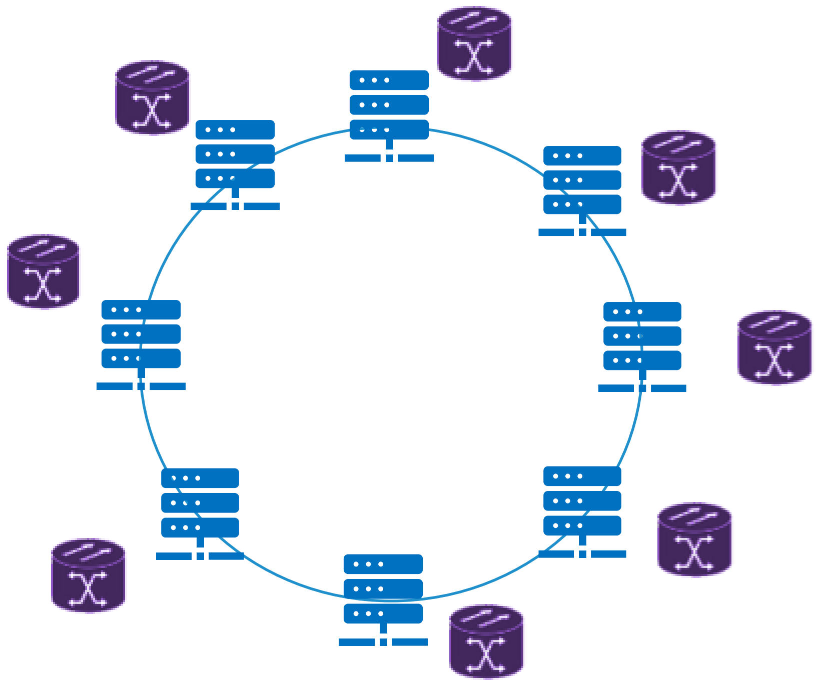
Hybrid Demand-Aware Network Design

Hybrid Demand-Aware Network Design



Nodes (e.g., servers in datacenters) want to communicate (e.g., along shortest paths).

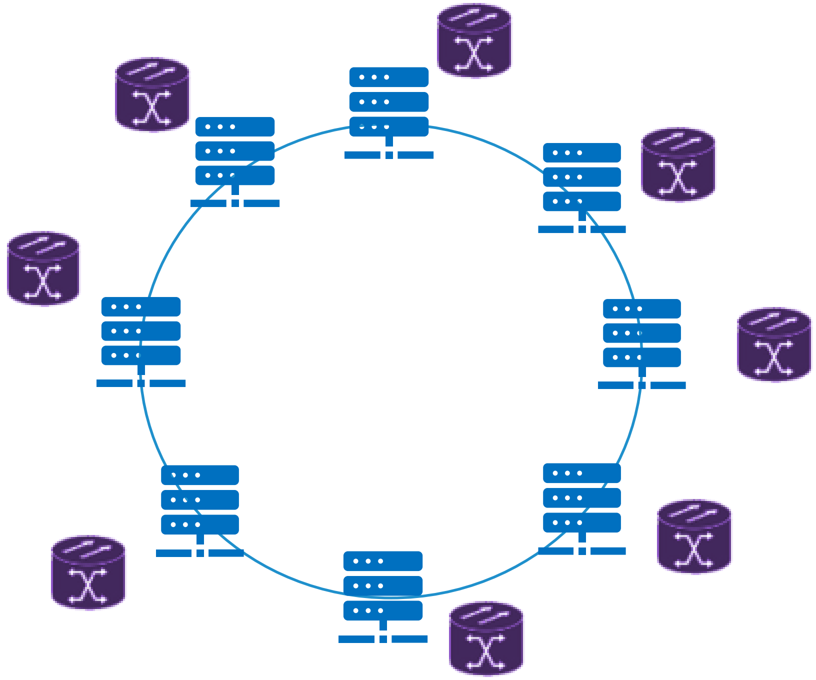
Hybrid Demand-Aware Network Design

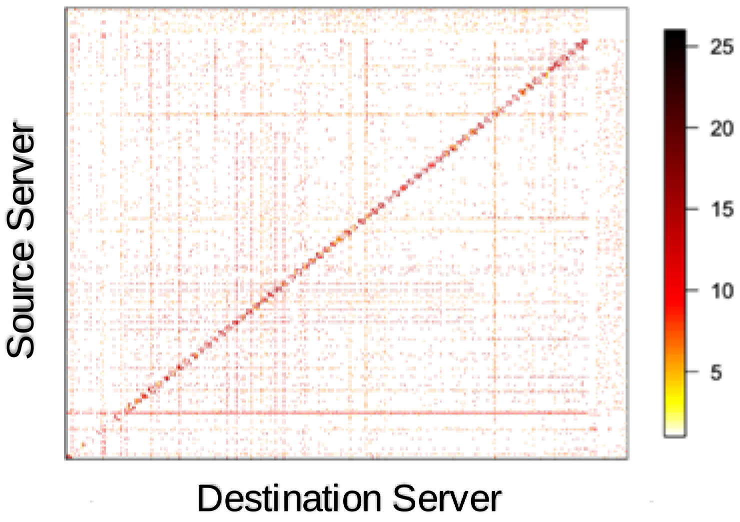


Nodes (e.g., servers in datacenters) want to communicate (e.g., along shortest paths).

Hybrid topology: a given fixed topology (here: ring) can be enhanced with additional edges (e.g., realized with an optical switch).

Hybrid Demand-Aware Network Design





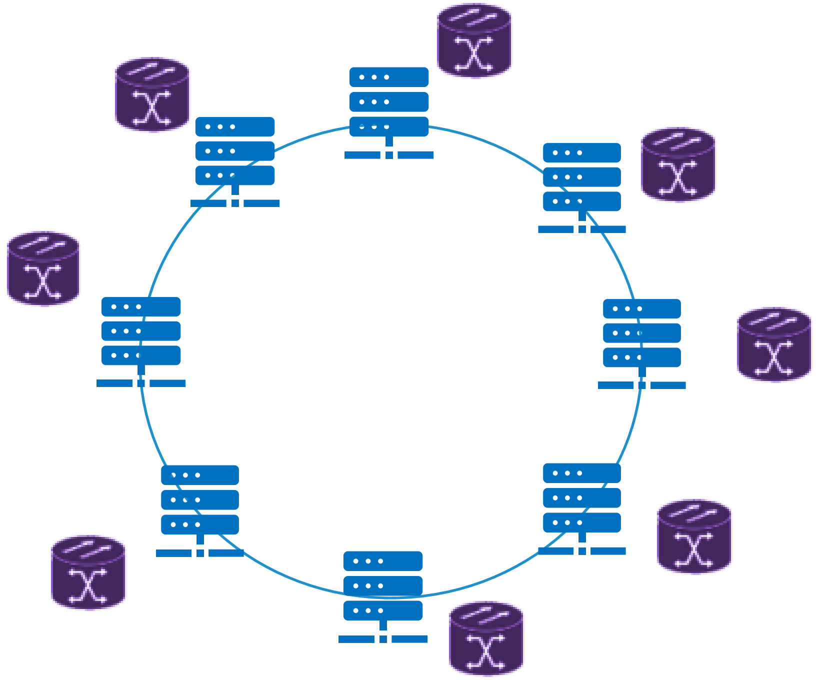
Nodes (e.g., servers in datacenters) want to communicate (e.g., along shortest paths).

Hybrid topology: a given fixed topology (here: ring) can be enhanced with additional edges (e.g., realized with an optical switch).

Some of nodes need to communicate to other nodes more frequently.

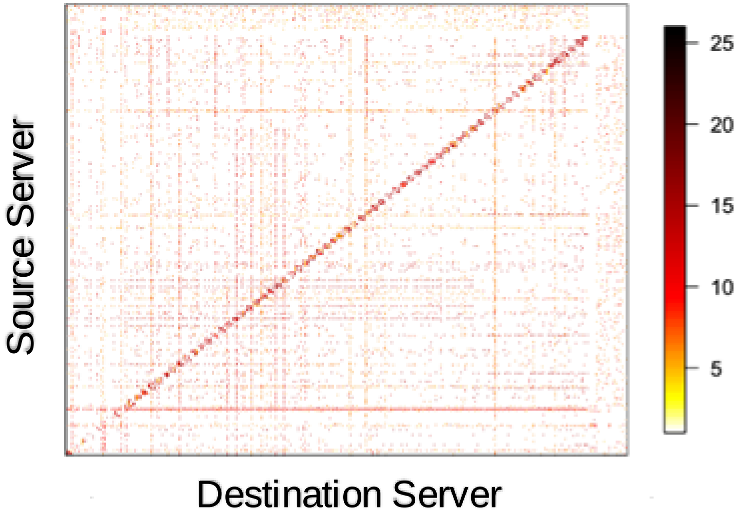
The Nature of Datacenter Traffic: Measurements & Analysis
Microsoft Research

Hybrid Demand-Aware Network Design



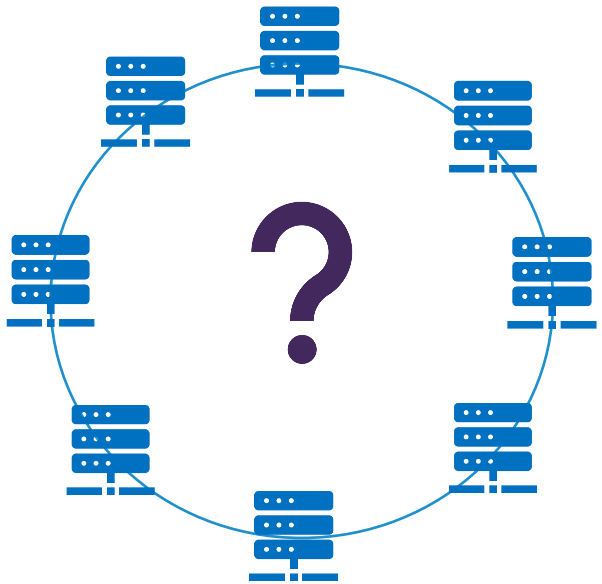
Nodes (e.g., servers in datacenters) want to communicate (e.g., along shortest paths).

Hybrid topology: a given fixed topology (here: ring) can be enhanced with additional edges (e.g., realized with an optical switch).



Some of nodes need to communicate to other nodes more frequently.

The Nature of Datacenter Traffic: Measurements & Analysis
Microsoft Research

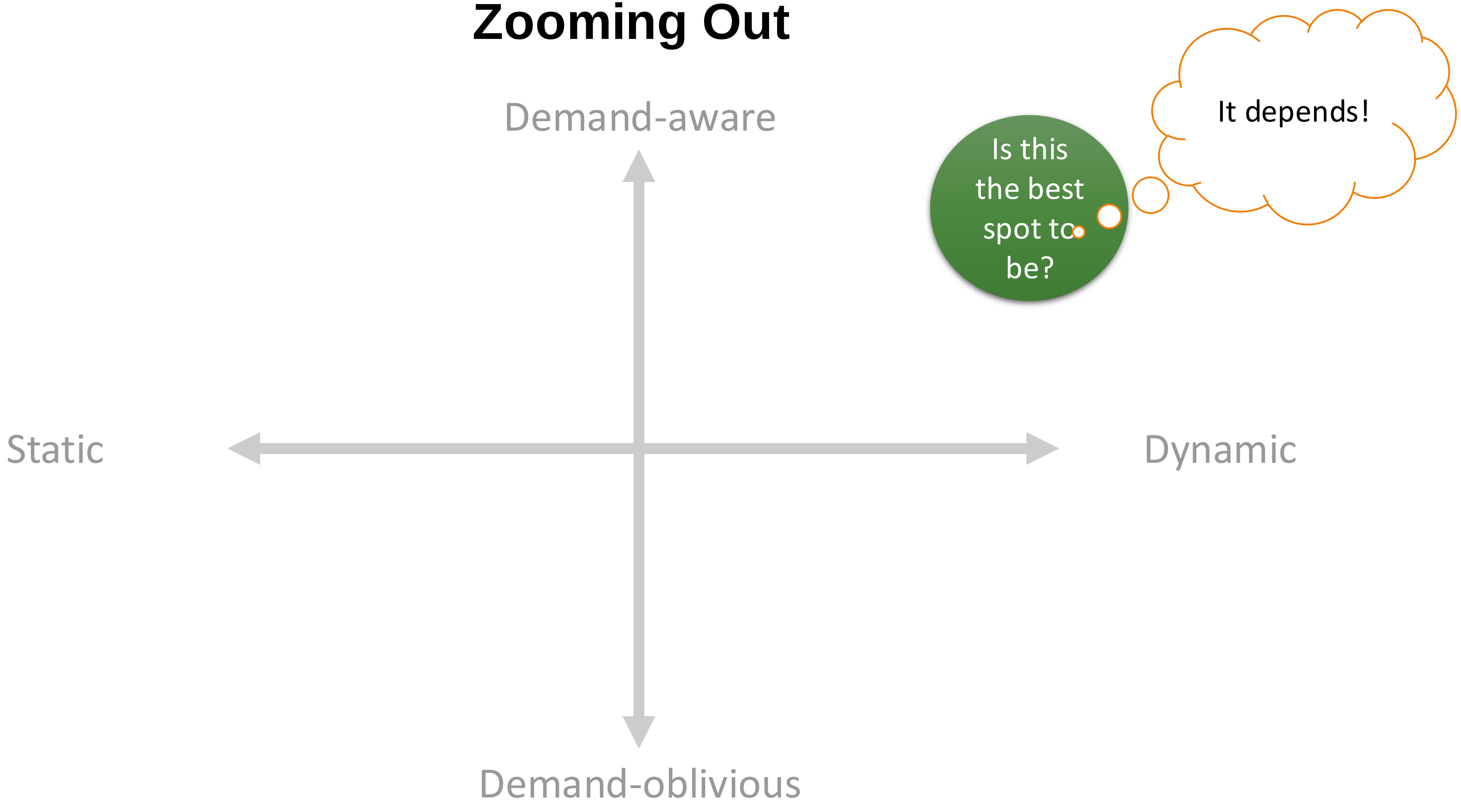


Finding the best static hybrid topology that minimizes delay/congestion/…

Zooming Out

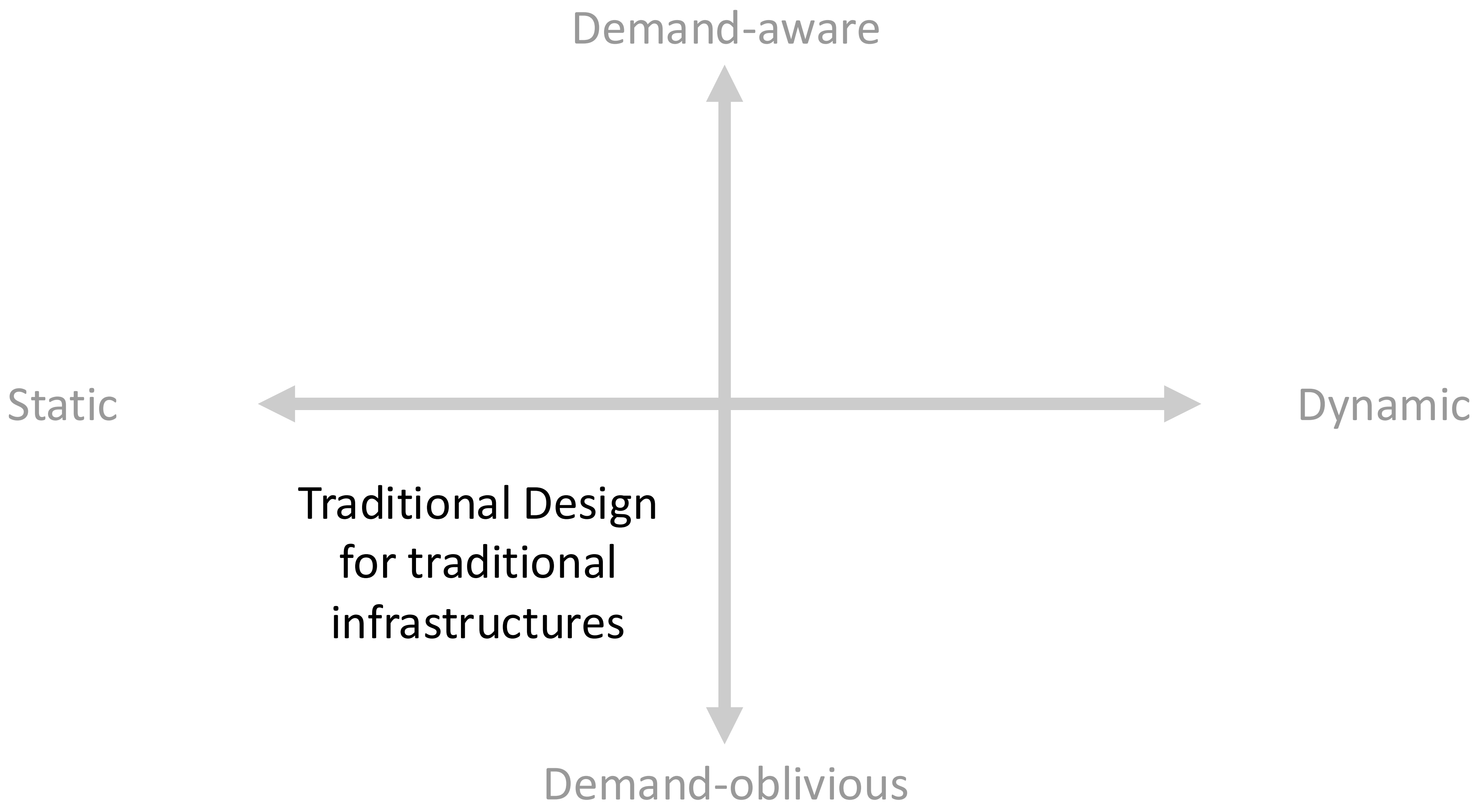
Static ⟷ Dynamic

Zooming Out



Demand-aware
Is this the best spot to be?
It depends!
Static
Dynamic
Demand-oblivious

Zooming Out


Demand-aware
Static
Dynamic
Traditional Design
for traditional
infrastructures
Demand-oblivious

Zooming Out

Demand-aware

Static

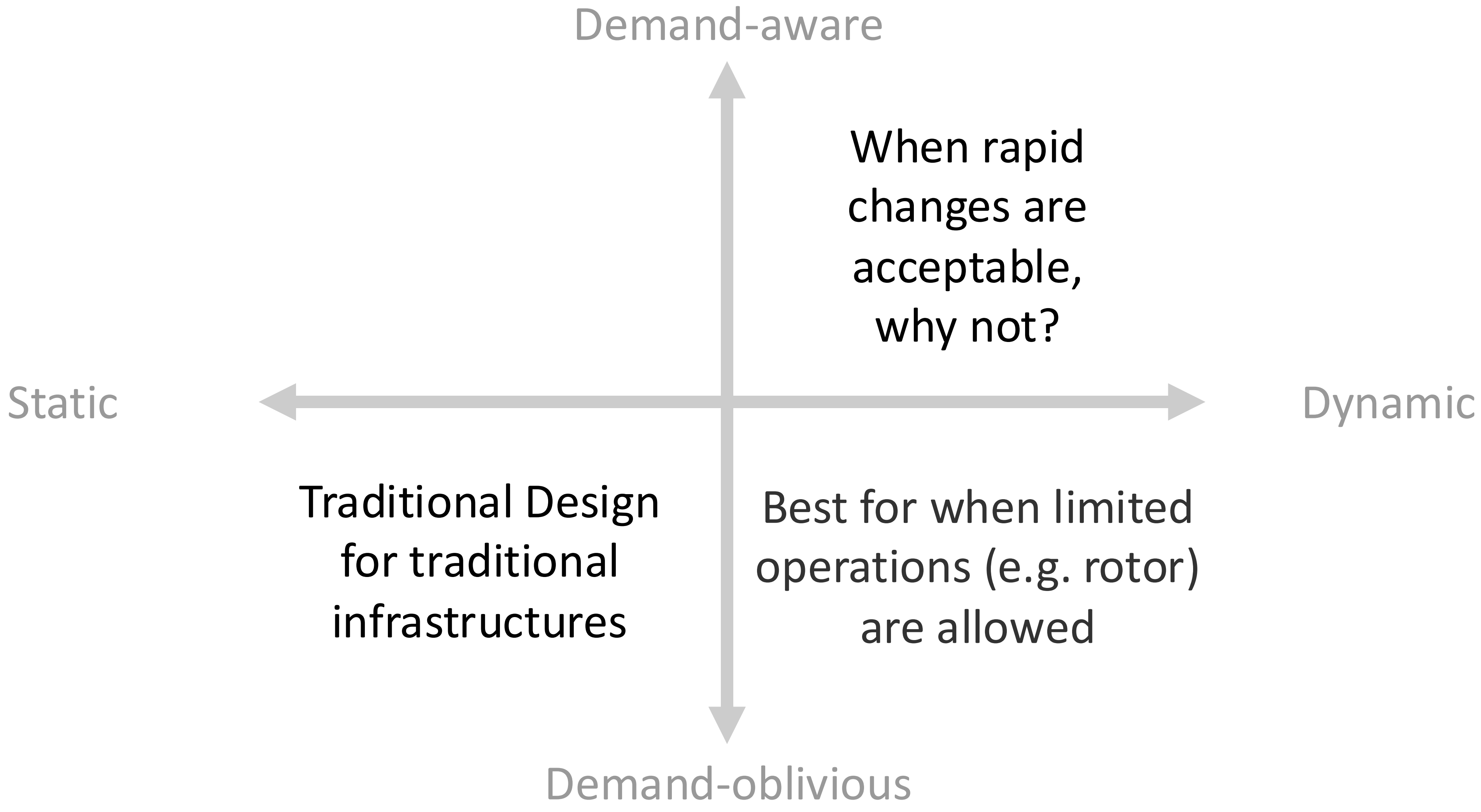
Dynamic

Traditional Design for traditional infrastructures

Best for when limited operations (e.g. rotor) are allowed

Demand-oblivious

Zooming Out


Demand-aware
When rapid changes are acceptable, why not?
Static
Dynamic
Traditional Design for traditional infrastructures
Best for when limited operations (e.g. rotor) are allowed
Demand-oblivious

Zooming Out

Demand-aware

Ideal for when changing network is not possible/costly

When rapid changes are acceptable, why not?

Static

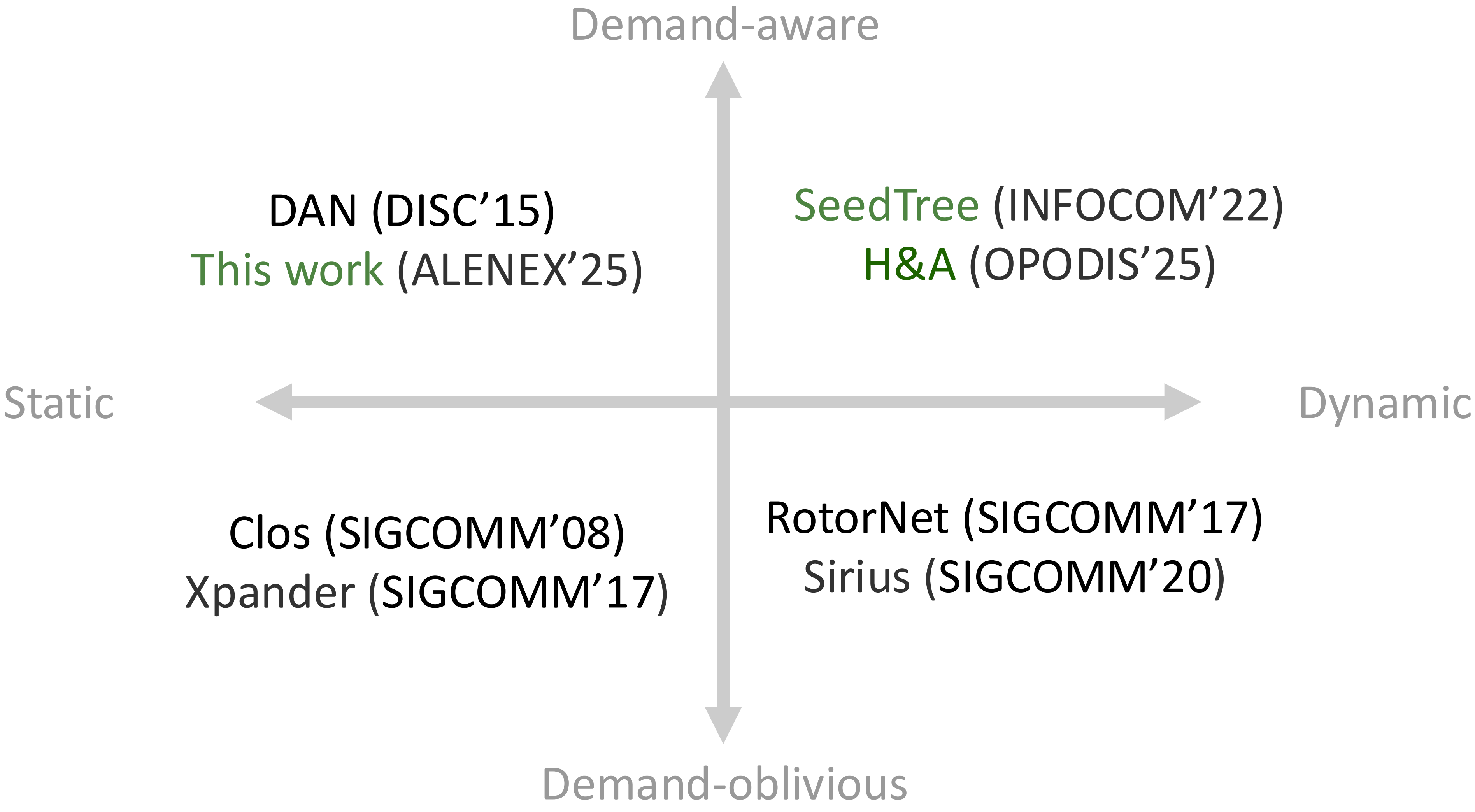
Dynamic

Traditional Design for traditional infrastructures

Best for when limited operations (e.g. rotor) are allowed

Demand-oblivious

Zooming Out: Some Prior Works in Networking



Theoretical Related Work

- Network augmentation for diameter minimization (worst case communication cost):
  - Finding number of edges needed to reduce diameter to d is NP-complete [Schoone et al., J. Graph Theory 1987]
  - Lower and upper bounds for cycles [Grigorescu, J. Graph Theory 2003]
  - with degree constraints [Adriaens and Gionis, ICDM 2022]

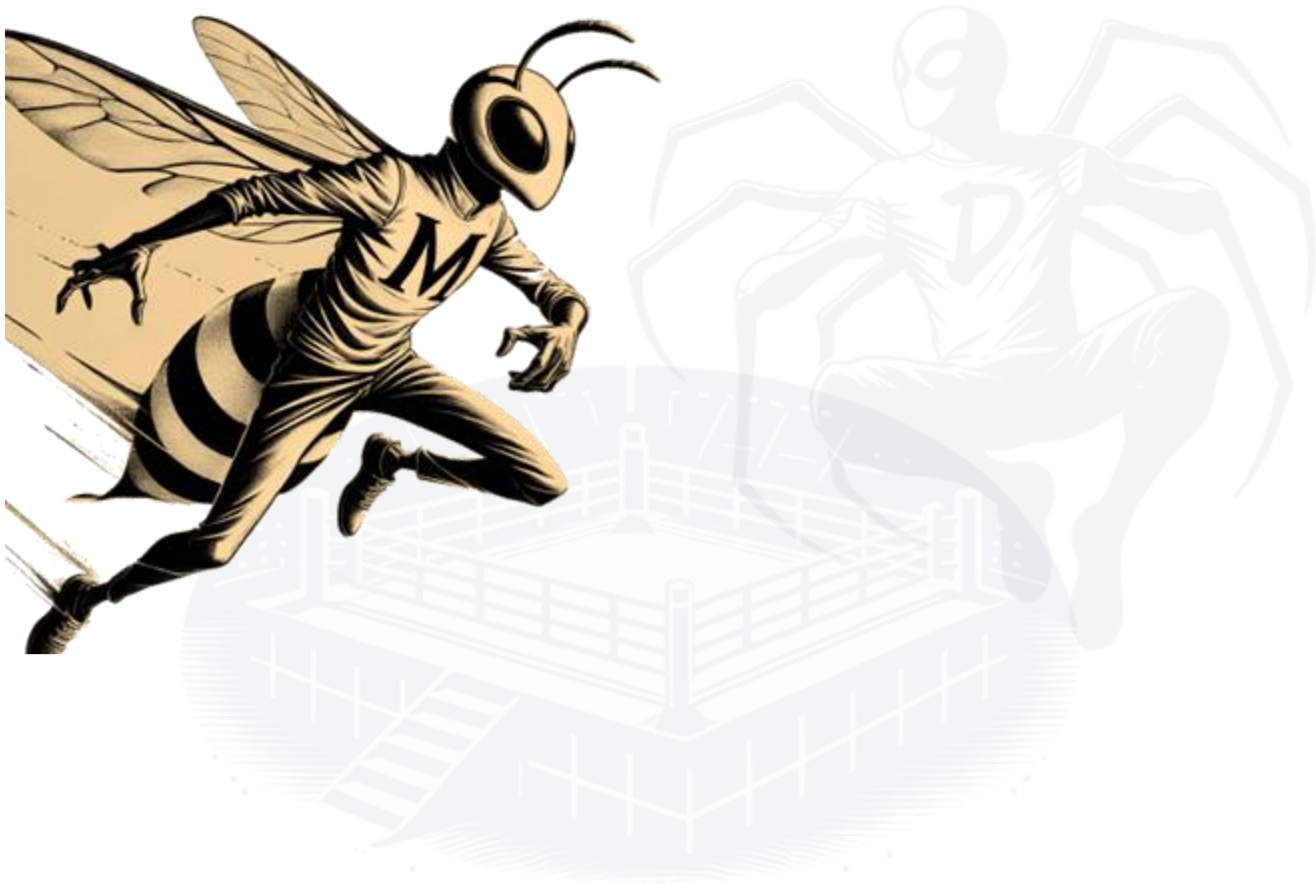
Theoretical Related Work

- Network augmentation for diameter minimization (worst case communication cost):
  - Finding number of edges needed to reduce diameter to d is NP-complete [Schoone et al., J. Graph Theory 1987]
  - Lower and upper bounds for cycles [Grigorescu, J. Graph Theory 2003]
  - with degree constraints [Adriaens and Gionis, ICDM 2022]

- Network augmentation for minimizing average shortest path length:
  - Small world phenomenon [Kleinberg, STOC 2000] and [Watts and Strogatz, Nature 1998]
  - NP-hardness and approximation for adding fixed number of edges [Meyerson and Tagiku, RANDOM 2009]

Or … The Tale of

MWASP SpiderDAN



Formal Model For This Talk

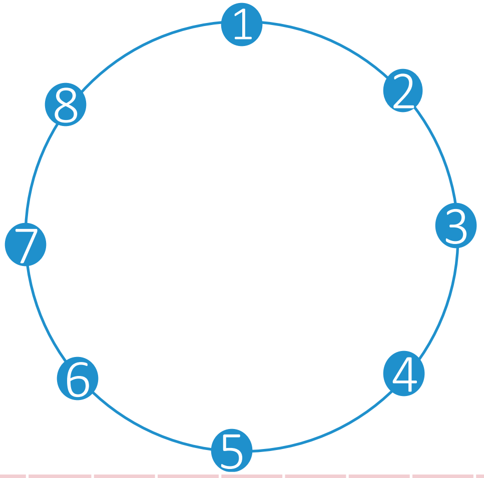
Minimizing Weighted Average Shortest Path length via matching addition (*MWASP*)

Formal Model For This Talk

Minimizing Weighted Average Shortest Path length via matching addition (*MWASP*)

Input:

An **infrastructure graph** G and a **demand matrix** D.



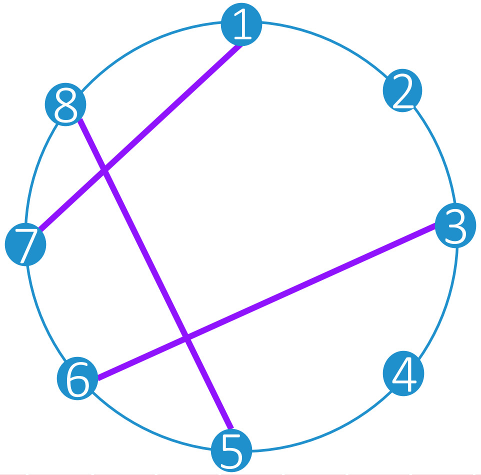
	1	2	3	4	5	6	7	8
1	0	1	0	2	0	0	23	3
2	1	0	1	2	1	0	0	0
3	0	1	0	0	0	30	0	1
4	2	2	0	0	0	1	0	0
5	0	1	0	0	0	0	0	25
6	0	0	30	1	0	0	1	0
7	23	0	0	0	0	1	0	0
8	3	0	1	0	25	0	0	0

Formal Model For This Talk

Minimizing Weighted Average Shortest Path length via matching addition (*MWASP*)

Input:

An **infrastructure graph** G and a **demand matrix** D.



	1	2	3	4	5	6	7	8
1	0	1	0	2	0	0	23	3
2	1	0	1	2	1	0	0	0
3	0	1	0	0	0	30	0	1
4	2	2	0	0	0	1	0	0
5	0	1	0	0	0	0	0	25
6	0	0	30	1	0	0	1	0
7	23	0	0	0	0	1	0	0
8	3	0	1	0	25	0	0	0

Output:

Find a **static matching** $M \subseteq \binom{V}{2}$ that minimizes:

$$\Sigma_{u,v\in V} D_{u,v} * \ dist_{G+M}(u,v)$$

in $G + M$.

NP-Hardness

Theorem 1. *MWASP* is NP-hard!

(Even if the infrastructure graph is a cycle and every row and column of the demand-matrix has at most two non-zero elements)

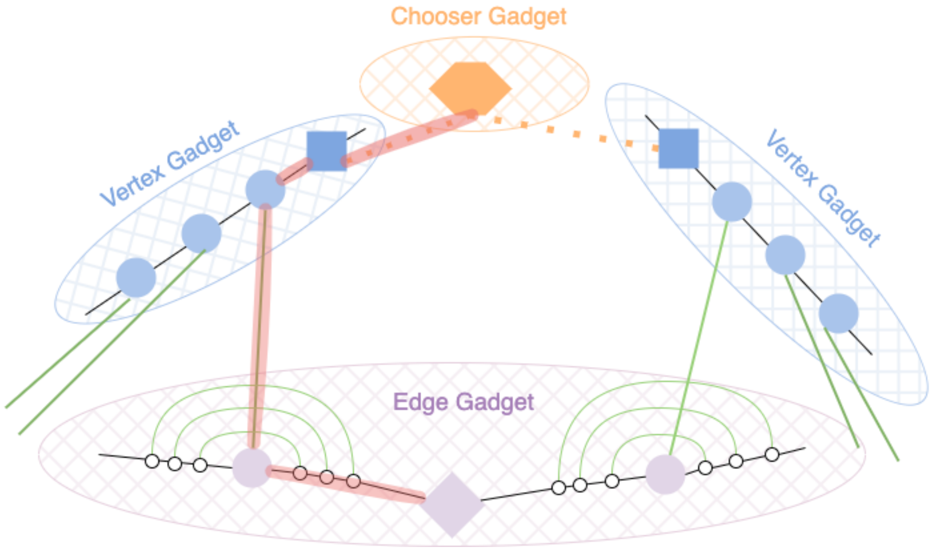
NP-Hardness

Theorem 1. *MWASP* is NP-hard!

(Even if the infrastructure graph is a cycle and every row and column of the demand-matrix has at most two non-zero elements)

- **Proof idea:**

Reduction from Vertex Cover on graphs with maximum degree three.



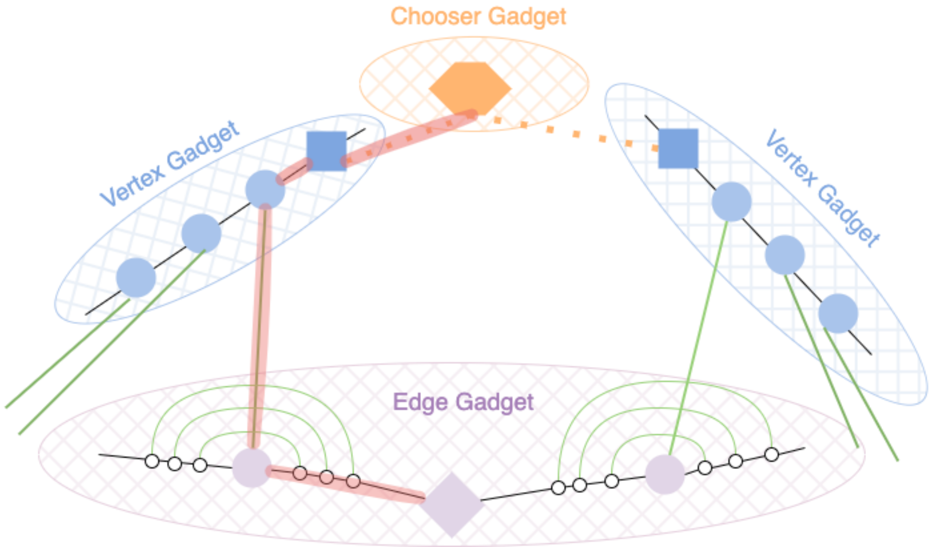
NP-Hardness

Theorem 1. *MWASP* is NP-hard!

(Even if the infrastructure graph is a cycle and every row and column of the demand-matrix has at most two non-zero elements)

- **Proof idea:**

Reduction from Vertex Cover on graphs with maximum degree three.



Open Question 1: In our construction, some edges are forced (green edges) to find the optimal results. Can we relax it towards a hardness of approximation?

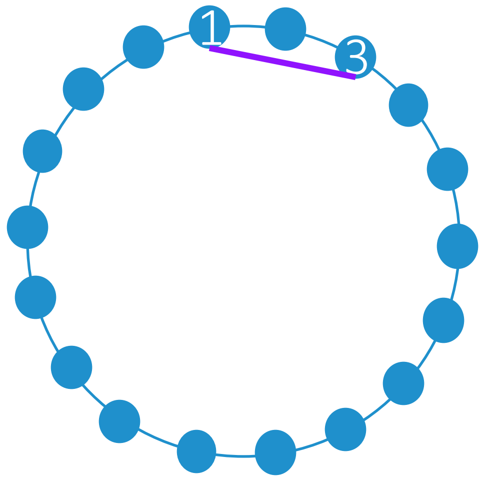
Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

(same for a maximum matching algorithm)



	1	2	3	…	$\frac{n}{2}$	…
1	0	0	50	0	49	0
2	0	0	0	0	0	0
3	50	0	0	0	0	0
…	0	0	0	0	0	0
$\frac{n}{2}$	49	0	0	0	0	0
…	0	0	0	0	0	0

Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

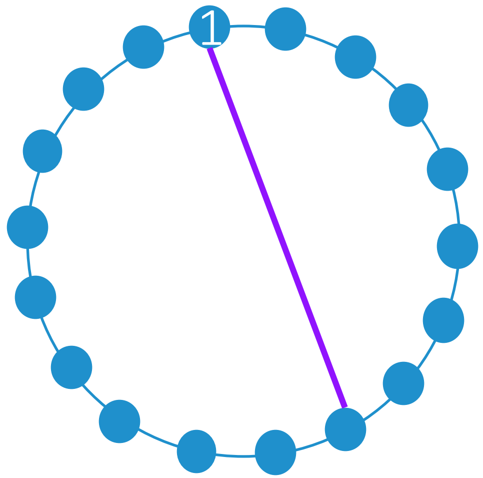
GA2: Connects nodes which are furthest away!?

Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

GA2: Connects nodes which are furthest away!?

(This idea is common in practice, i.e. a special case of Chord)



	1	2	3	…	$\frac{n}{2}$	…
1	0	0	50	0	0	0
2	0	0	0	0	0	0
3	50	0	0	0	0	0
…	0	0	0	0	0	0
$\frac{n}{2}$	0	0	0	0	0	0
…	0	0	0	0	0	0

Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

GA2: Connects nodes which are furthest away!?

GA3: Merged (somehow!) single-source solutions?!

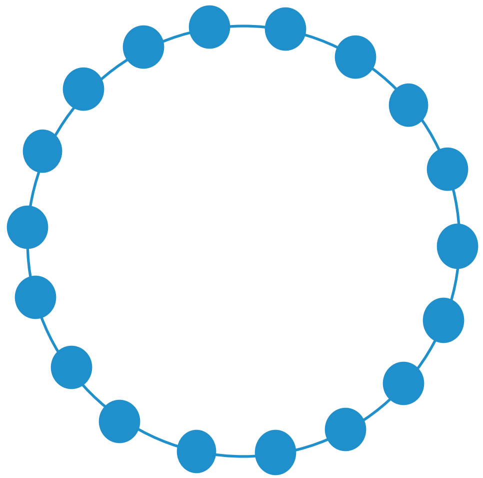
Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

GA2: Connects nodes which are furthest away!?

GA3: Merged (somehow!) single-source solutions?!

(no, some pairs can be bridges for others)



	1	2	3	…	$\frac{n}{2}$	…
1	0	0	99	0	0	0
2	0	0	0	0	10	0
3	99	0	0	0	0	0
…	0	1	0	0	0	0
$\frac{n}{2}$	0	10	0	0	0	0
…	0	1	0	0	0	0

Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

GA2: Connects nodes which are furthest away!?

GA3: Merged (somehow!) single-source solutions?!

GA4: Algorithms based on submodularity?!

Then… What About Greedy Algorithm X?

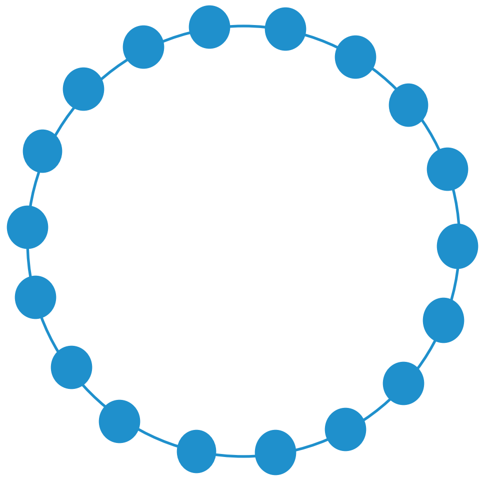
GA1: Connects nodes with the highest demand!?

GA2: Connects nodes which are furthest away!?

GA3: Merged (somehow!) single-source solutions?!

GA4: Algorithms based on submodularity?!

(no, some pairs can be bridges for others)



	1	2	3	…	$\frac{n}{2}$	…
1	0	0	99	0	0	0
2	0	0	0	0	10	0
3	99	0	0	0	0	0
…	0	1	0	0	0	0
$\frac{n}{2}$	0	10	0	0	0	0
…	0	1	0	0	0	0

Then… What About Greedy Algorithm X?

GA1: Connects nodes with the highest demand!?

GA2: Connects nodes which are furthest away!?

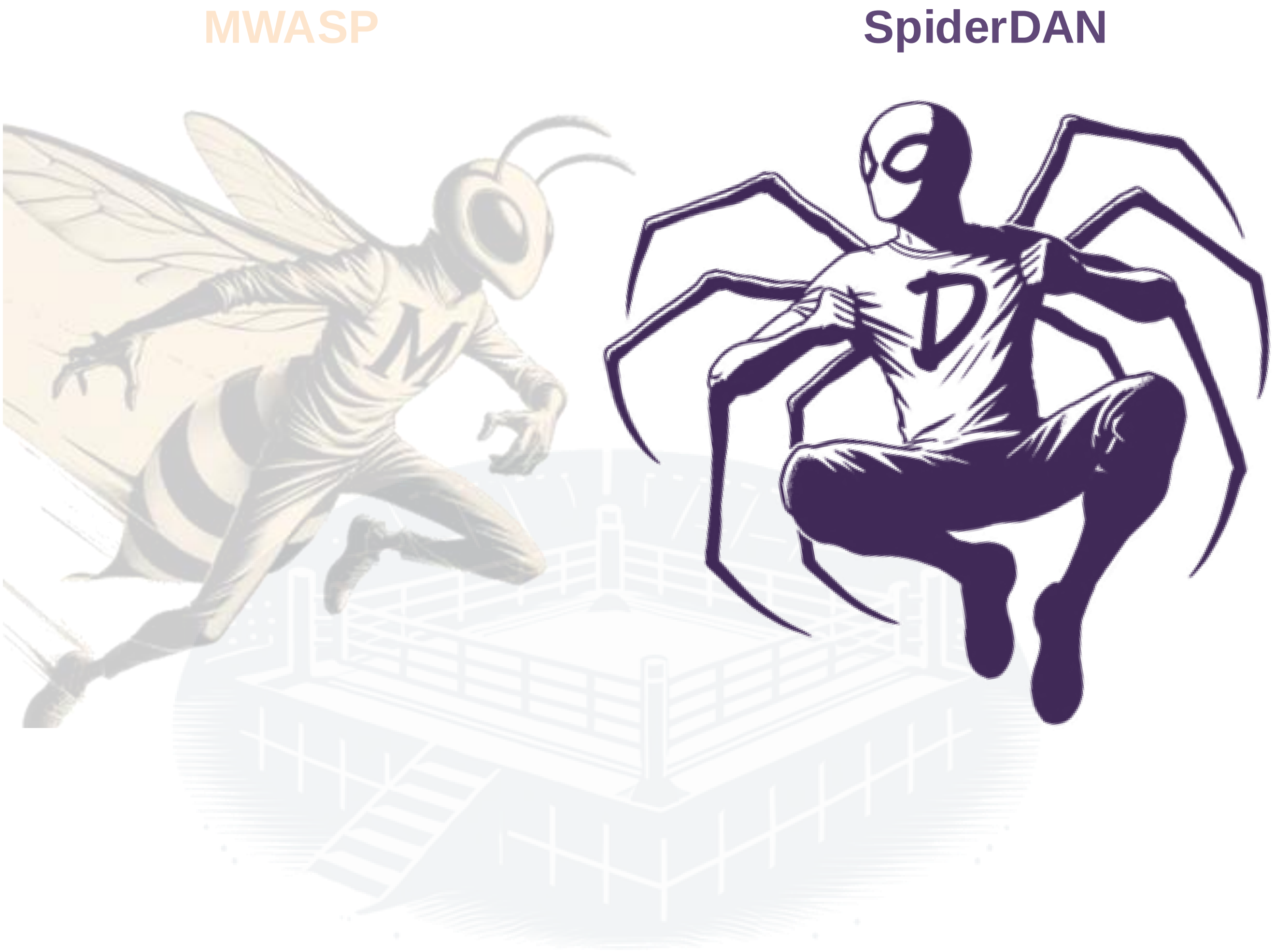
GA3: Merged (somehow!) single-source solutions?!

GA4: Algorithms based on submodularity?!

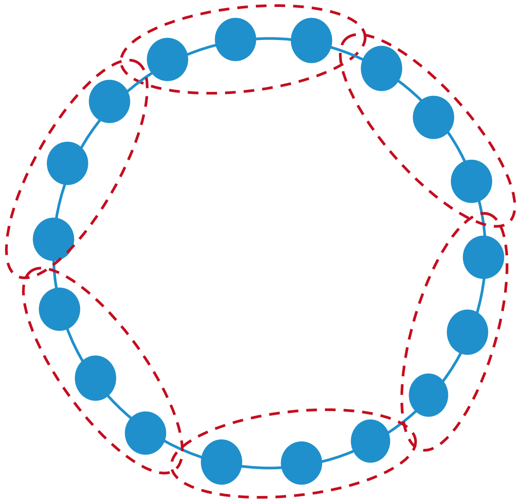
GA5: For a certain set of inputs?

Open Question 2: Is there an approximation algorithm for the general inputs?

Or … The Tale of

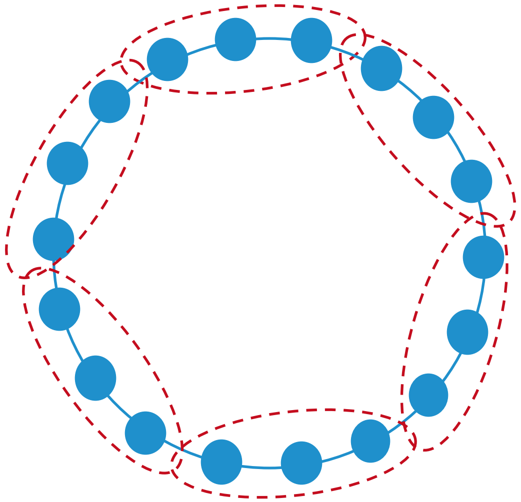


Overview of SpiderDAN Algorithm

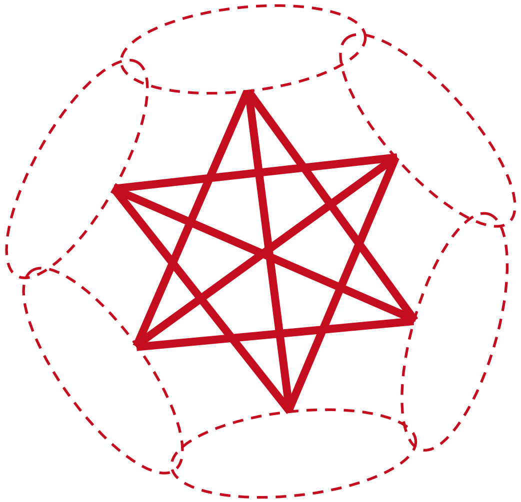


Step 1: Decomposing into super nodes of constant size

Overview of SpiderDAN Algorithm



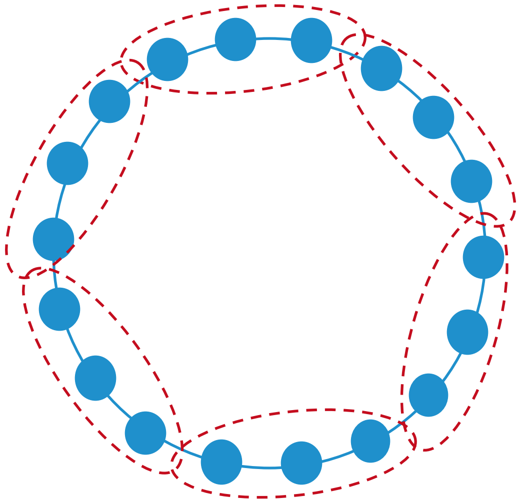
Step 1: Decomposing into super nodes of constant size



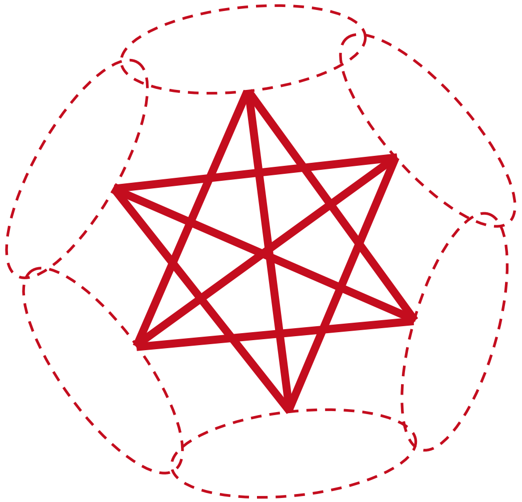
Step 2: Creating a demand-aware constant degree network#
on top of super nodes

#Chen Avin, Kaushik Mondal, and Stefan Schmid. 2020. Demand-aware network designs of bounded degree. Distributed Computing (2020).

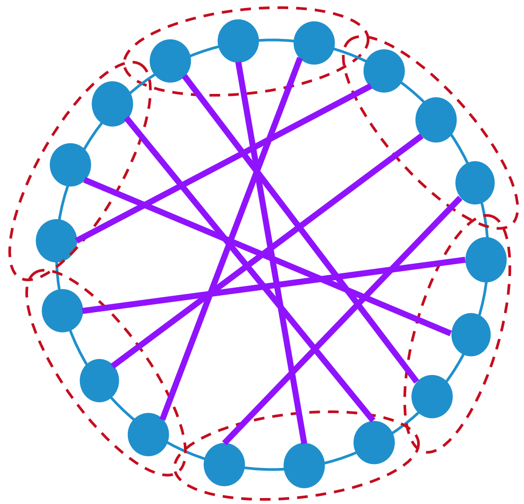
Overview of SpiderDAN Algorithm



Step 1: Decomposing into super nodes of constant size



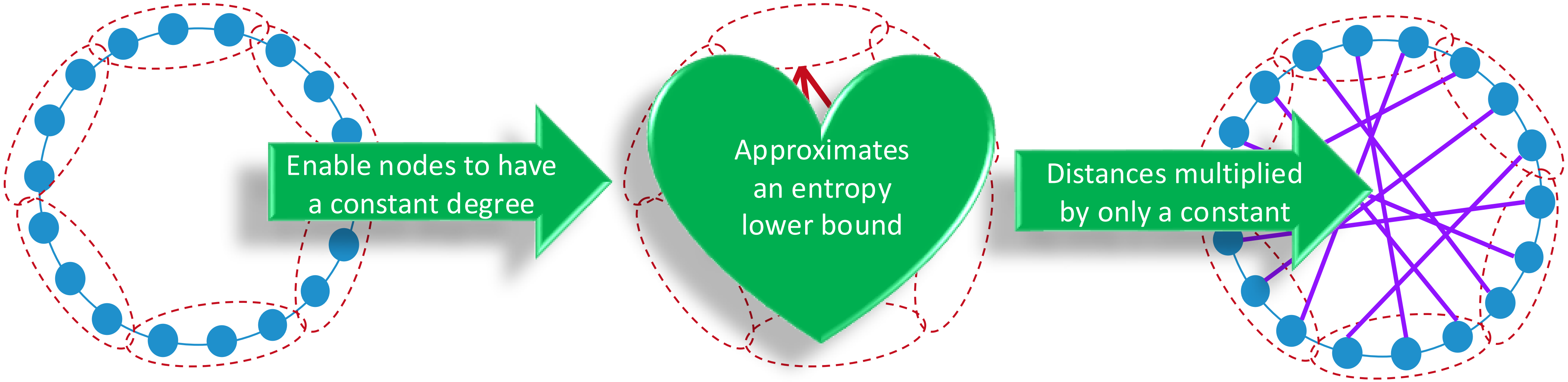
Step 2: Creating a demand-aware constant degree network# on top of super nodes



Step 3: Transforming the constant degree network to a matching

#Chen Avin, Kaushik Mondal, and Stefan Schmid. 2020. Demand-aware network designs of bounded degree. Distributed Computing (2020).

Constant Approximation Algorithm for Sparse* Demands and Infrastructure Graph with High Diameter



Step 1: Decomposing into super nodes of constant size

Step 2: Creating a demand-aware constant degree network# on top of super nodes

Step 3: Transforming the constant degree network to a matching

*Low average demand. Sparse demand are motivated by practical use cases. Formalize it shortly.

#Chen Avin, Kaushik Mondal, and Stefan Schmid. 2020. Demand-aware network designs of bounded degree. Distributed Computing (2020).

Theoretical Guarantee

Theorem 2. Given any connected infrastructure graph with non-constant diameter, and a demand graph of average degree at most $\frac{1}{\alpha}$ (for a constant α) we can compute a matching that is a constant factor approximation for **MWASP**.

Theoretical Guarantee

Theorem 2. Given any connected infrastructure graph with non-constant diameter, and a demand graph of average degree at most $\frac{1}{\alpha}$ (for a constant α) we can compute a matching that is a constant factor approximation for **MWASP**.

Lemma 1 [sketch] It is possible to decompose G into super nodes of size α, such that nodes corresponding to the same super nodes retain $O(1)$ distance of each other in the infrastructure graph.

Supernode Creation On General Graphs

Task: Merge nodes into super nodes of size α, while nodes in the same super node have pairwise distance of $O(1)$.

Supernode Creation On General Graphs

Task: Merge nodes into super nodes of size α, while nodes in the same super node have pairwise distance of $O(1)$.

1. Build spanning tree T on the infrastructure graph

Supernode Creation On General Graphs

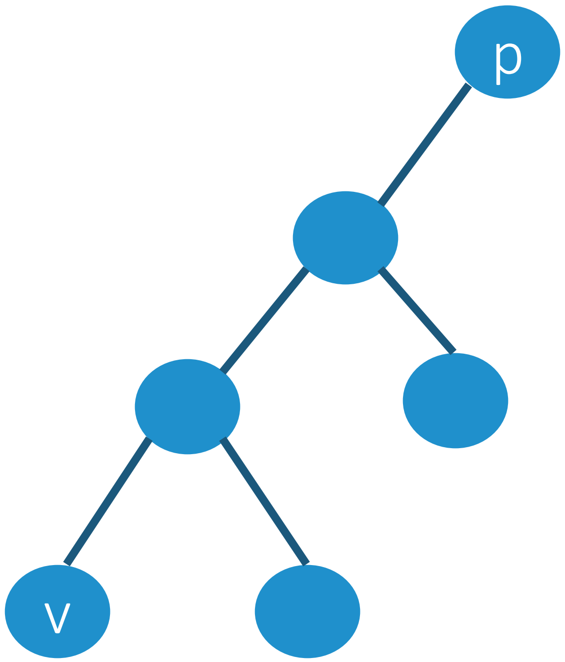
Task: Merge nodes into super nodes of size α, while nodes in the same super node have pairwise distance of $O(1)$.

1. Build spanning tree T on the infrastructure graph
2. Repeat $\frac{n}{\alpha}$ times:

    Pick deepest leaf v in T

    Consider grandparent p at distance α

    Consider T_p to be subtree rooted at p



Supernode Creation On General Graphs

Task: Merge nodes into super nodes of size α, while nodes in the same super node have pairwise distance of $O(1)$.

1. Build spanning tree T on the infrastructure graph
2. Repeat $\frac{n}{\alpha}$ times:

   Pick deepest leaf v in T

   Consider grandparent p at distance α

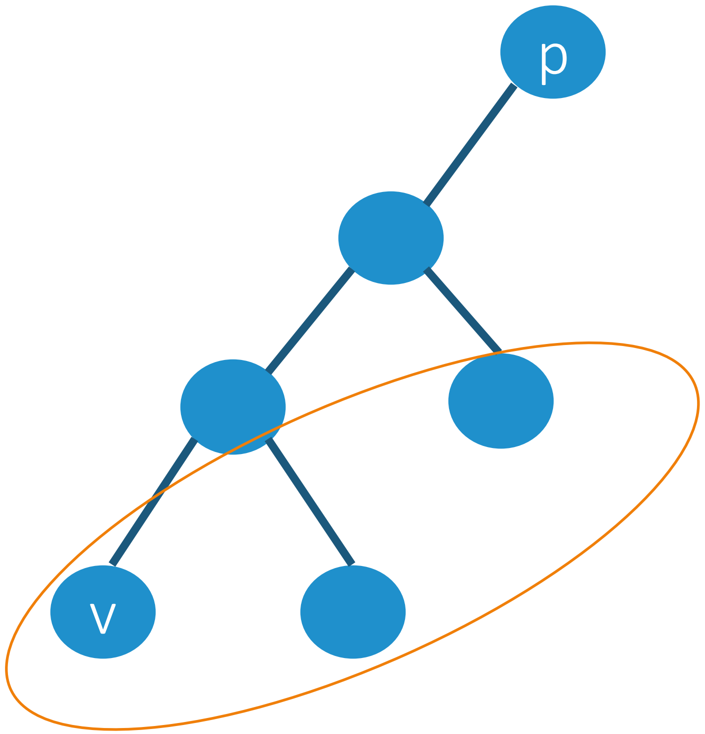
   Consider T_p to be subtree rooted at p

3. Repeat α times:

   Pick any leaf u in T_p

   Add u to to the supernode

   Remove u



Theoretical Guarantee

Theorem 2. Given any connected infrastructure graph with non-constant diameter, and a demand graph of average degree at most $\frac{1}{\alpha}$ (for a constant α) we can compute a matching that is a constant factor approximation for **MWASP**.

Lemma 1 [sketch] It is possible to decompose G into super nodes of size α, such that nodes corresponding to the same super nodes retain $O(1)$ distance of each other in the infrastructure graph.

Lemma 2 [sketch] On the resulting graph and demand matrix, it is possible to build a DAN of degree at most α , which is a constant factor approximation of an optimal solution [Chen Avin, Kaushik Mondal, and Stefan Schmid, Distributed computing 2020]

Heuristics

- **Greedy**: greedily take highest weight demand edge, if possible

Heuristics

- **Greedy**: greedily take highest weight demand edge, if possible
- **Matching on Demand**: compute maximum weight matching

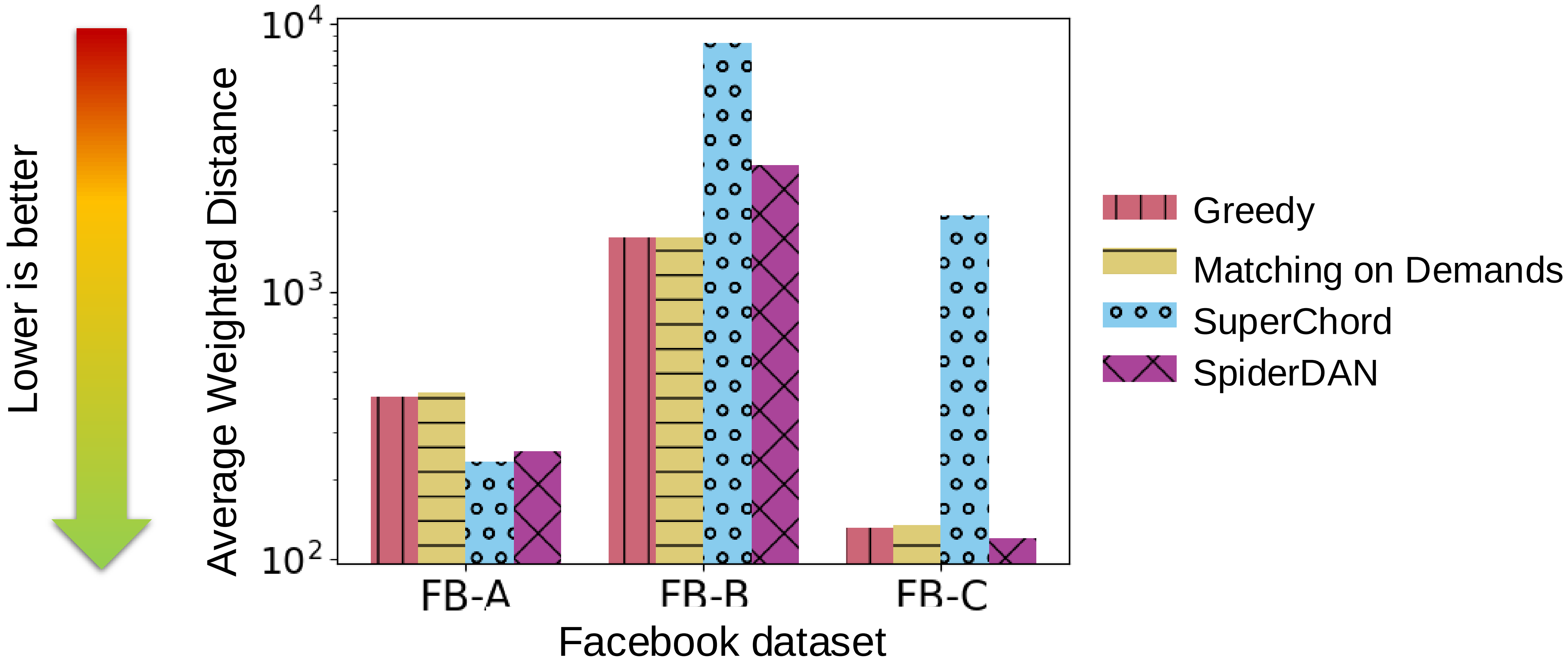
Heuristics

- **Greedy**: greedily take highest weight demand edge, if possible
- **Matching on Demand**: compute maximum weight matching
- **SuperChord**: On a cycle merge x consecutive nodes such that $x = \log(n/x)$. Run Chord protocol on the resulting super graph with degree $\log(n/x)$. Recover matching like in SpiderDAN.

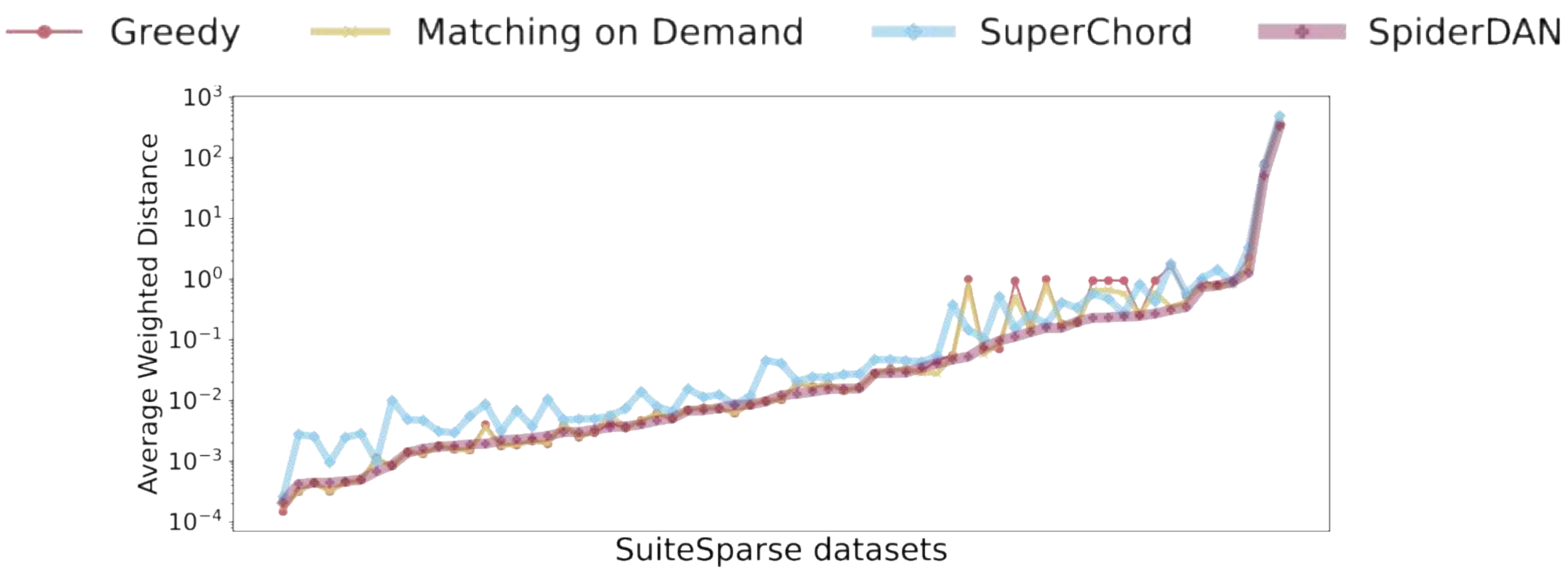
Heuristics

- **Greedy**: greedily take highest weight demand edge, if possible
- **Matching on Demand**: compute maximum weight matching
- **SuperChord**: On a cycle merge x consecutive nodes such that $x = \log(n/x)$. Run Chord protocol on the resulting super graph with degree $\log(n/x)$. Recover matching like in SpiderDAN.
- **MIP**: $O(n^3)$ binary variables and $O(n^2)$ constraints With Gurobi solves instances up to $n = 20$ in around an hour.

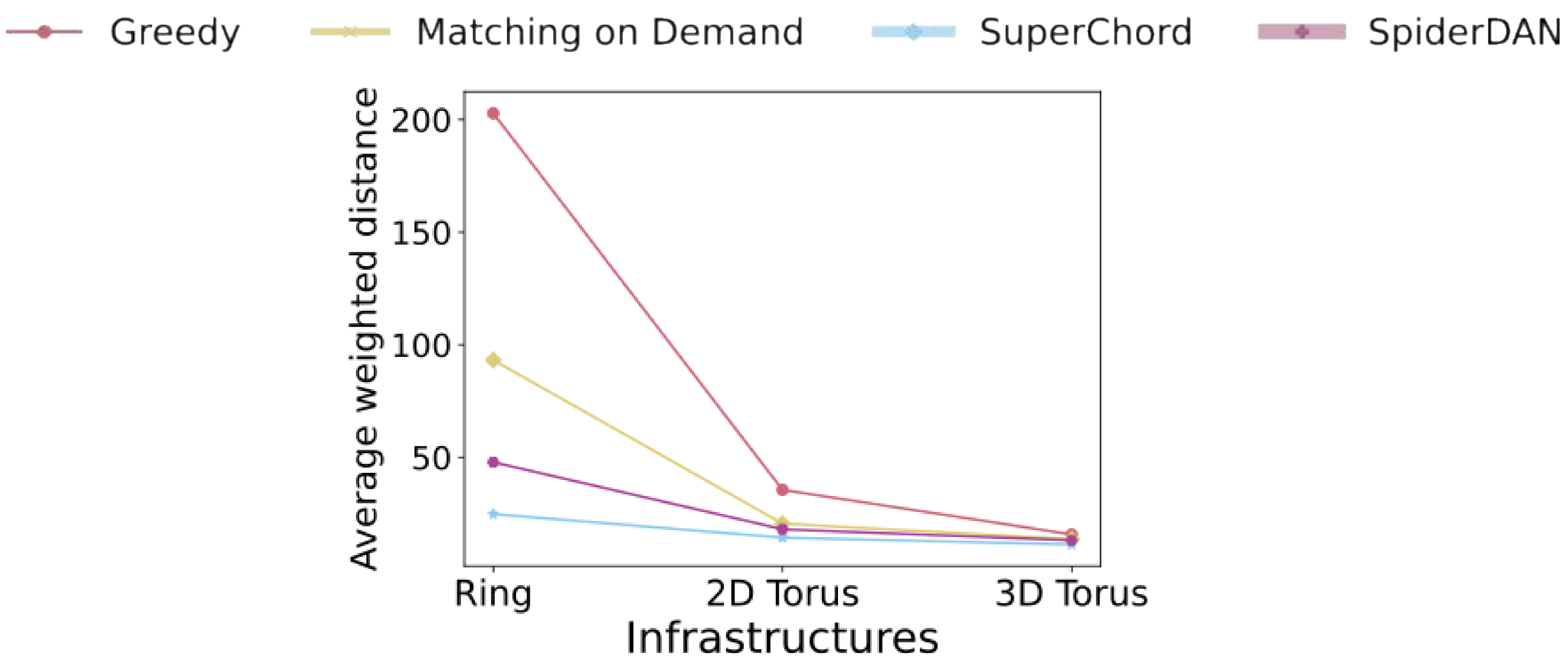
Empirical Results on Facebook Dataset



Empirical Results on SuiteSparse Matrix Collection



Empirical Results on Different Infrastructure Graphs



Open Questions ☺

1. Hardness of approximation in the general case?

2. Approximation algorithms in the general case?

Read more:
https://t.ly/wTbeq

Send me an email:
pourdamghani@tu-berlin.de

Follow our group
https://www.tu.berlin/en/eninet