Data Center Traffic Design and Optimization with Optical and Static Topology

Özgün Barış Kır
Advisor: Prof. Oya Ekin Karaşan

Industrial Engineering Department, Bilkent University

14th January 2022

Outline

Motivation

Growing Data Load

- Digital society's expanding necessities [Foerster et al., 2019]
- Latest advancements in Machine Learning and Artificial Intelligence [Yang et al., 2019]

Recent Technologies

- Optic connections are more reliable and faster in terms of data transfer[Fenz et al., 2019]
- Utilizing optical matchings makes the model demand-aware and reconfigurable [Foerster and Schmid, 2019]
- Reconfigurable networks increase the throughput rate of the network and number of accepted requests up to 70% [Luo et al., 2019][Luo et al., 2020].
- 1% of the node pairs account for 80% percent of the total traffic[Foerster and Schmid, 2019]

Literature Review

Traditional Networks

- Static Topology (Only electrical connections) [Bienkowski et al., 2021]
- Hybrid topology (Both optical and electrical connections) without matchings [Foerster and Schmid, 2019]

Recent Developments

- Efficient routing design by using optical matchings [Fenz et al., 2019]
- Our research is an extension to their work

Complexity

- Creating matchings and finding the shortest path simultaneously increase the complexity which cannot be found in polynomial time [Foerster et al., 2019].
- [Fenz et al., 2019] proved that the problem is indeed NP-Hard.

Sets and Parameters:
N:Set of nodes
O:Set of optic arcs
A:Set of all arcs(union of static and optic arcs)
s_{ij}:represents the weight of the static link from i to j ($s_{ij} \geq o_{ij}$)
o_{ij}:represents the weight of the optic link from i to j
D_{ij}: size of the demand value from i to j
Decision Variables:
x_{ij}^{st}:is set to 1 if a link from i to j is used in the shortest path from s to t
y_{ij}^{st}:is set to 1 if an optic link from i to j is used in the shortest path from s to t and
m_{ij}:equals to 1 when there is an optic matching between node i and j, 0 otherwise
$dist_{st}$: the resulting shortest distance between s and t

[Fenz et al., 2019]'s Model

$$\text{minimize} \quad \sum_{s}\sum_{t} D_{st}\, dist_{st} \tag{1a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} m_{ij} \leq 1 \qquad \forall i \in N, \tag{1b}$$

$$\sum_{j=1}^{n} m_{ji} \leq 1 \qquad \forall i \in N, \tag{1c}$$

$$m_{ij} = m_{ji} \qquad \forall (i,j) \in O, \tag{1d}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{st} - \sum_{j:(j,i)\in A} x_{ji}^{st} = \begin{cases} 1 & if \quad i = s \\ -1 & if \quad i = t \\ 0 & else \end{cases} \qquad \forall i, s, t \in N \quad s \neq t, \tag{1e}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}(x_{ij}^{st} - y_{ij}^{st}) + o_{ij}\, y_{ij}^{st} = dist_{st} \qquad \forall s, t \in N \quad s \neq t, \tag{1f}$$

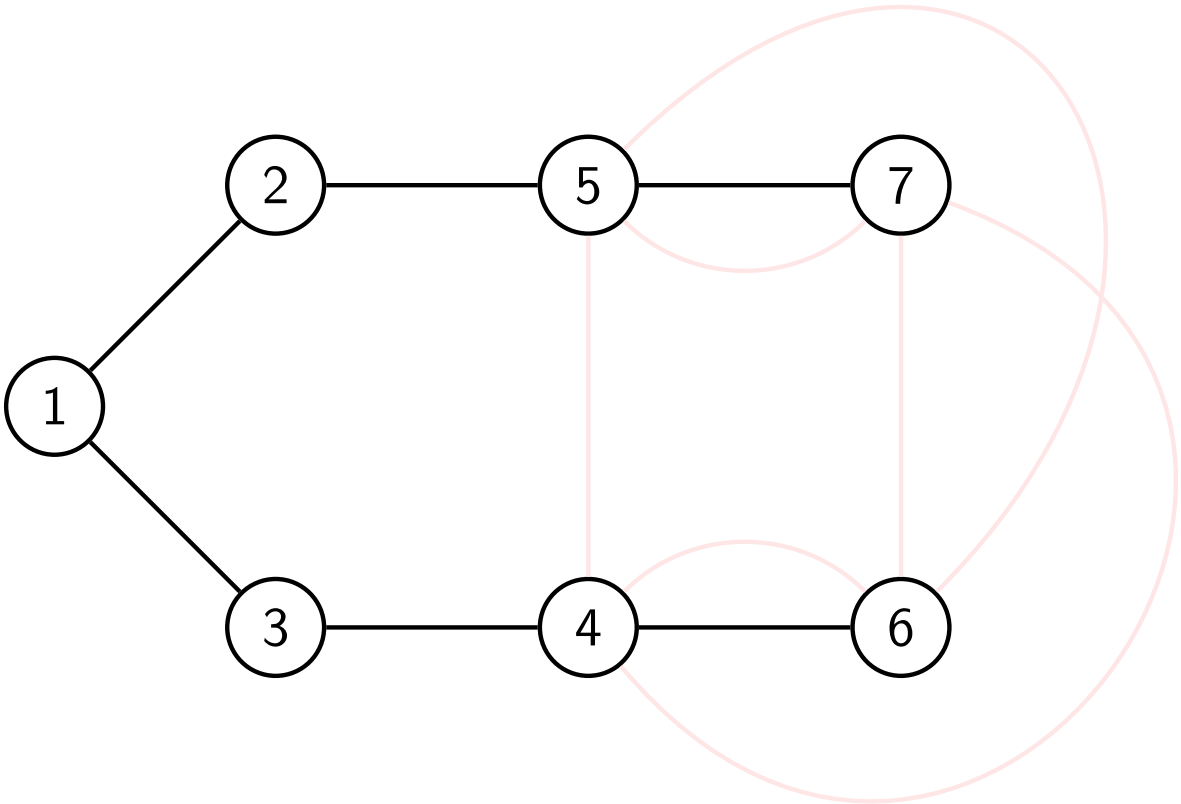
$$y_{ij}^{st} \leq m_{ij} \qquad \forall (i,j) \in O \quad \forall s, t \in N, \tag{1g}$$

$$y_{ij}^{st} \leq x_{ij}^{st} \qquad \forall (i,j) \in A \quad \forall s, t \in N, \tag{1h}$$

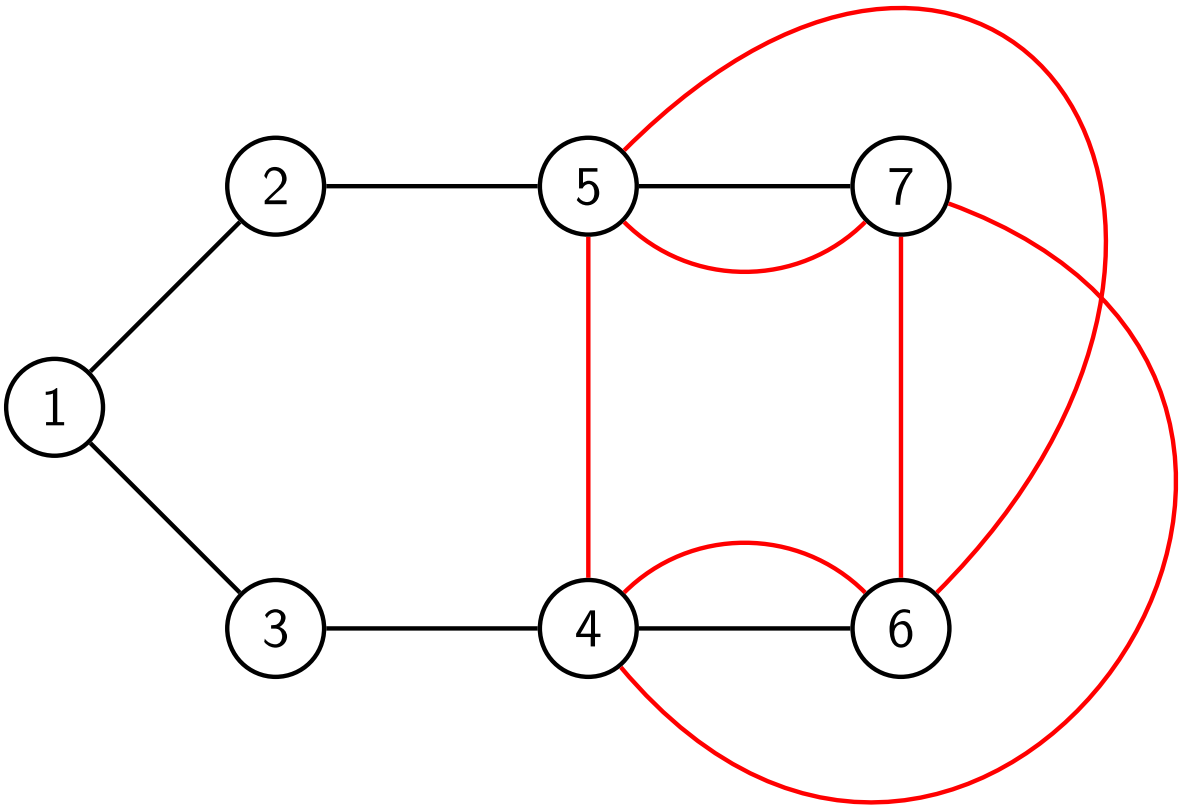
$$x_{ij}^{st}, y_{ij}^{st} \geq 0 \qquad \forall i, j, s, t \in N, \tag{1i}$$

$$m_{ij} \in \{0, 1\} \qquad \forall (i,j) \in O \tag{1j}$$

Graph for [Fenz et al., 2019]'s Model



Graph for [Fenz et al., 2019]'s Model



Parameters and Decision Variables for Our Model

New Decision Variables:

f_{ij}^{s}:static flow value from i to j with source node s

g_{ij}^{s}:optic flow value from i to j with source node s

Our Model for A Reconfigurable Switch

$$\text{minimize} \quad \sum_{s \in N} \sum_{(i,j) \in A} s_{ij} f_{ij}^s + o_{ij} g_{ij}^s \tag{2a}$$

$$\text{subject to} \quad \sum_{j:(j,i) \in A} f_{ji}^s + g_{ji}^s - \sum_{j:(i,j) \in A} f_{ij}^s + g_{ij}^s = \begin{cases} -\sum_{t:t \neq s; t \in N} D_{st} & i = s \\ D_{si} & i \neq s \end{cases} \quad \forall s \in N, \tag{2b}$$

$$\sum_{s}^{n} g_{ij}^s \leq M m_{ij} \quad \forall (i,j) \in O, \tag{2c}$$

$$\sum_{j=1}^{n} m_{ij} \leq 1 \quad \forall i \in N, \tag{2d}$$

$$\sum_{j=1}^{n} m_{ji} \leq 1 \quad \forall i \in N, \tag{2e}$$

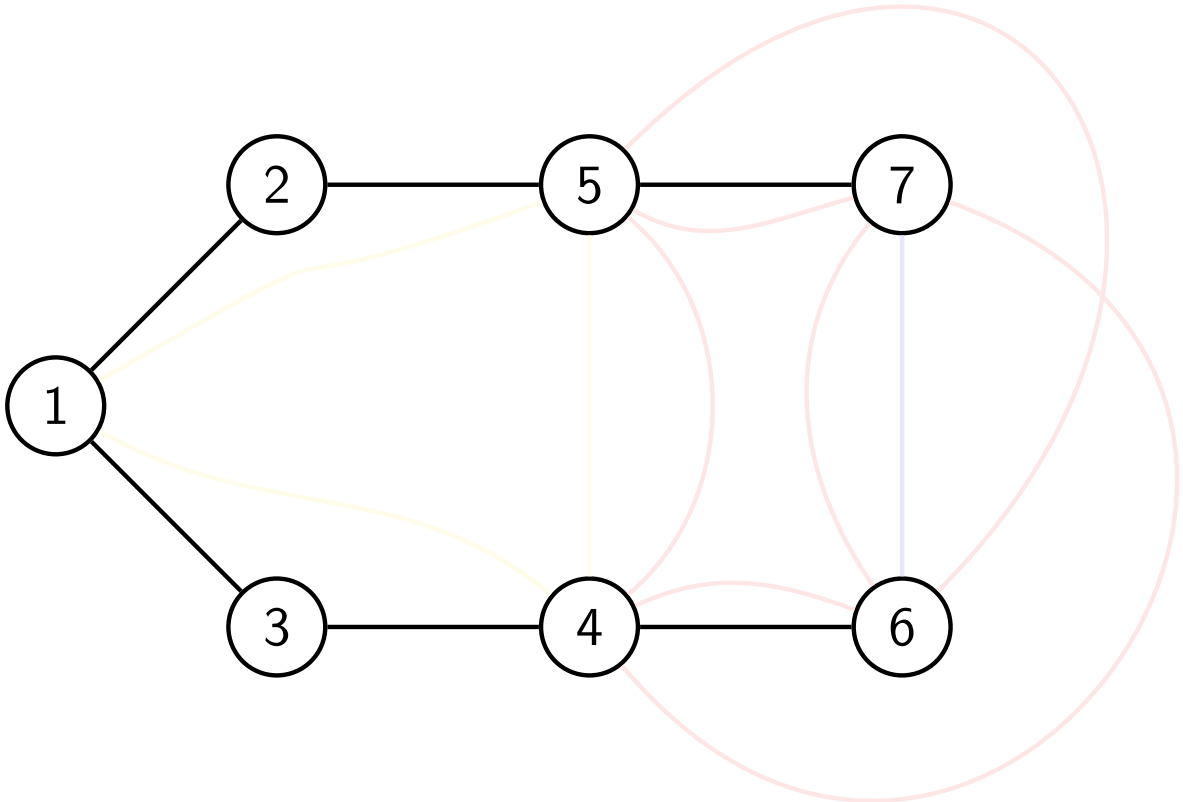
$$m_{ij} = m_{ji} \quad \forall (i,j) \in O, \tag{2f}$$

$$f_{ij}^s, g_{ij}^s \geq 0 \quad \forall i,j,s \in N, \tag{2g}$$

$$m_{ij} \in \{0,1\} \quad \forall (i,j) \in O$$

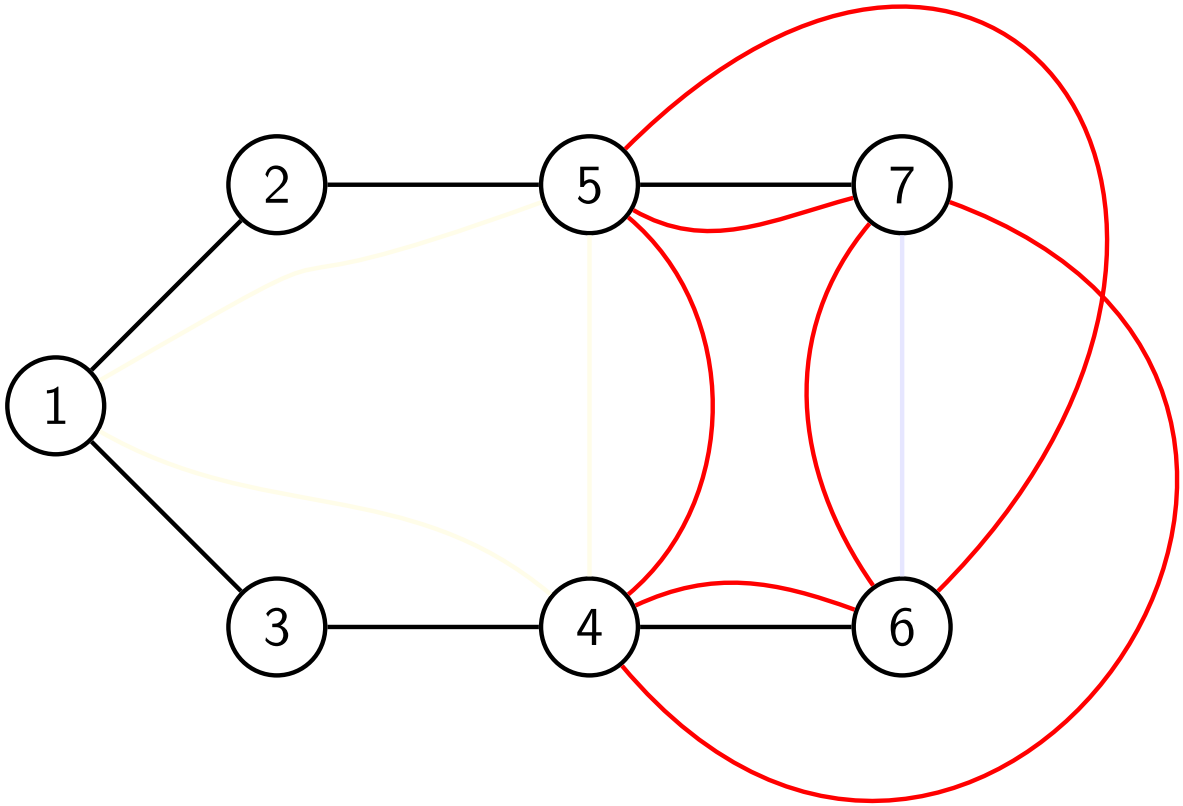
Graph for K Switch Model

S1:[1,4,5]
S2:[4,5,6,7]
S3:[6,7]



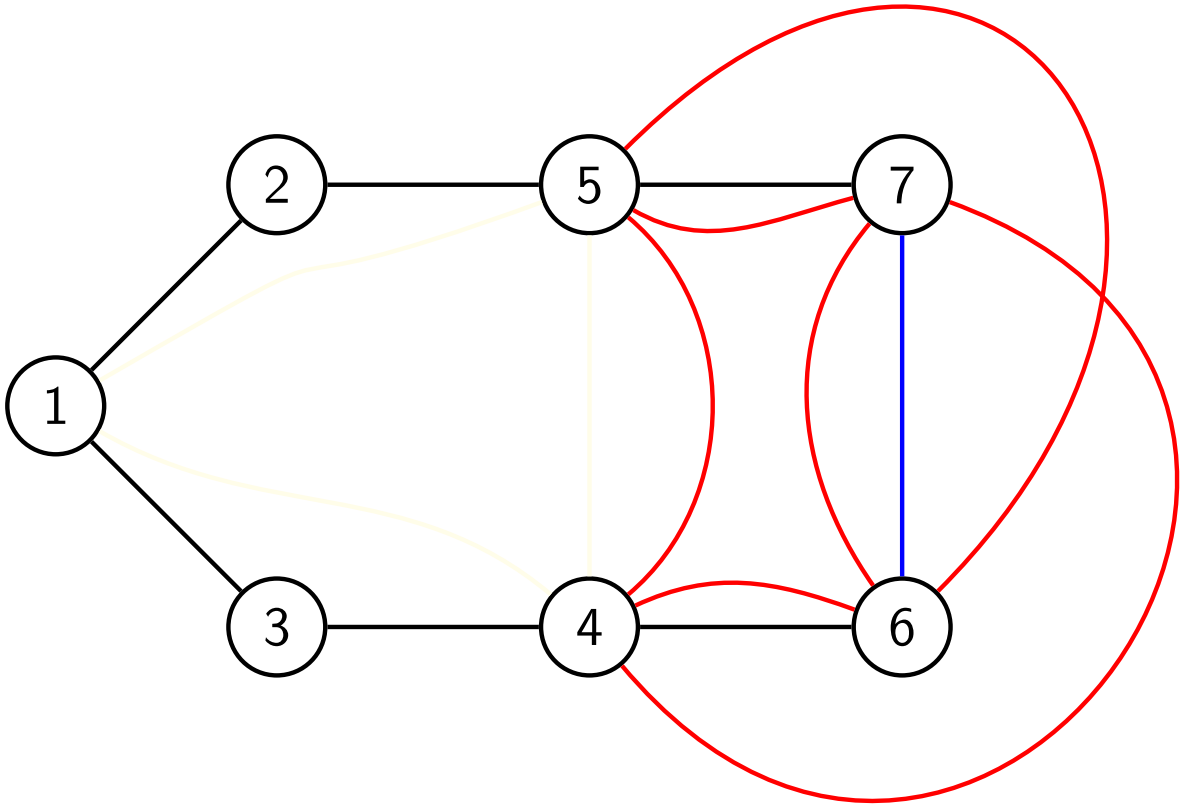
Graph for K Switch Model

S1:[1,4,5]
S2:[4,5,6,7]
S3:[6,7]



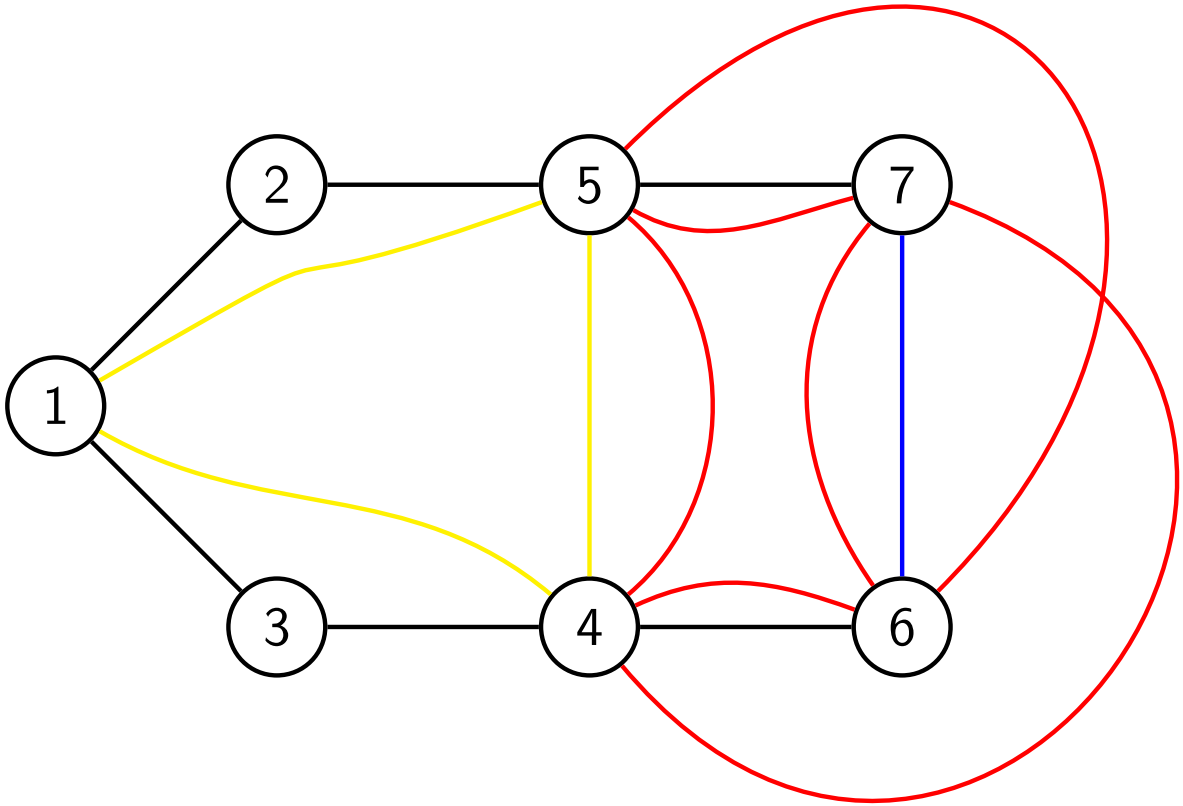
Graph for K Switch Model

S1:[1,4,5]
S2:[4,5,6,7]
S3:[6,7]



Graph for K Switch Model

S1:[1,4,5]
S2:[4,5,6,7]
S3:[6,7]



Parameters for Our Model with K Reconfigurable Switch

New Sets:

A^S: All the static links in the static topology

A^O: All optic links connected to all reconfigurable switches (Union of O_t's)

O_t: All optic links connected to reconfigurable switch t

New Decision Variable:

m_{ij}^t: equals to 1 when there is an optic matching between node i and j on the switch t, 0 otherwise

Our Model for K Reconfigurable Switch

$$\text{minimize} \quad \sum_{s \in N} \Big(\sum_{(i,j) \in A^S} s_{ij} f_{ij}^s + \sum_{(i,j) \in A^O} o_{ij} g_{ij}^s \Big) \tag{3a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} m_{ij}^t \leq 1 \qquad \forall i \in N \quad \forall t, \tag{3b}$$

$$\sum_{j=1}^{n} m_{ji}^t \leq 1 \qquad \forall i \in N \quad \forall t, \tag{3c}$$

$$m_{ij}^t = m_{ji}^t \qquad \forall (i,j) \in O_t \quad \forall t, \tag{3d}$$

$$\sum_{j:(j,i) \in A^S} f_{ji}^s + \sum_{j:(j,i) \in A^O} g_{ji}^s - \sum_{j:(i,j) \in A^S} f_{ij}^s - \sum_{j:(i,j) \in A^O} g_{ij}^s = \begin{cases} -\sum_{t:t \neq s; t \in N} D_{st} & i = s \\ D_{si} & i \neq s \end{cases} \qquad \forall s \in N, \tag{3e}$$

$$\sum_{s} g_{ij}^s \leq M \sum_{t=1}^{k} m_{ij}^t \qquad \forall (i,j) \in O_t, \tag{3f}$$

$$f_{ij}^s, g_{ij}^s \geq 0 \qquad \forall i,j,s \in N, \tag{3g}$$

$$m_{ij}^t \in \{0,1\} \qquad \forall (i,j) \in O_t \quad \forall t$$

Computational Results

We solved our model for one switch using Python's Gurobi interface and compared it with the [Fenz et al., 2019]'s model in terms of CPU Time.

Random Generator

To represent the data center traffic as realistically as possible, we created a pseudo random generator that creates static topology, optic topology, and demand matrix separately with the determined parameters for each three of them.

Connected Graph

Since it should be possible to reach every node by using only static topology, we first generated a random spanning tree and then expanded it randomly for static topology.

First Set of Results

In the first set of runs, a new set of data is created by the random generator for each run with different densities. CPU times are in seconds.

Table: Run Results

Static	Optic	Demand	# of nodes	Our CPU	CPU [Fenz et al., 2019]
.3	.3	.3	15	32.18	632.85
.9	.3	.9	15	2.37	112.74
.9	.3	.6	15	30.99	4876.51
.6	.3	.6	14	5.92	226.74
.9	.3	.3	14	5.02	241.55
.6	.3	.9	14	3.16	62.85
.6	.3	.3	12	2.54	44.02
.3	.3	.6	12	0.03	0.27
.9	.9	.3	12	3.16	62.85

Second Set of Results

In the first part of the second set of runs, the same data set is used for optic topology and demand matrix, only the static topology is changed.

Table: Second run results with only static density is varied

Static	Optic	Demand	# of nodes	Our CPU	CPU [Fenz et al., 2019]
.3	.6	.6	12	0.3	14.51
.6	.6	.6	12	0.31	3.05
.9	.6	.6	12	0.31	7.87
.3	.3	.3	12	1.93	53.29
.6	.3	.3	12	1.96	47.3
.9	.3	.3	12	1.53	43.45
.3	.9	.3	12	0.04	0.35
.6	.9	.3	12	0.03	0.38
.9	.9	.3	12	0.02	0.4

Second Set of Results

In the second part of the second set of runs, the same data set is used for static topology and demand matrix, only the optic topology is changed.

Table: Second run results with only optic density is varied

Static	Optic	Demand	# of nodes	Our CPU	CPU [Fenz et al., 2019]
.3	.3	.6	12	1.61	24.44
.3	.6	.6	12	0.19	0.71
.3	.9	.6	12	0.03	0.14
.6	.3	.9	12	0.27	2.41
.6	.6	.9	12	0.08	0.36
.6	.9	.9	12	0.03	0.1
.9	.3	.3	12	1.36	46.11
.9	.6	.3	12	0.3	6.17
.9	.9	.3	12	0.03	0.31

Second Set of Results

In the third part of the second set of runs, the same data set is used for static and optic topologies, only the demand matrix is changed.

Table: Second run results with only demand density is varied

Static	Optic	Demand	# of nodes	Our CPU	CPU [Fenz et al., 2019]
.3	.3	.3	12	2.4	56.69
.3	.3	.6	12	1.82	30.83
.3	.3	.9	12	0.22	1.94
.6	.9	.3	12	0.03	0.69
.6	.9	.6	12	0.02	0.11
.6	.9	.9	12	0.02	0.1
.9	.3	.3	12	2.25	53.88
.9	.3	.6	12	1.31	42.03
.9	.3	.9	12	0.87	5.77

Heuristics Proposed By [Fenz et al., 2019]

They first generate an algorithm that is derived by Dijkstra's Algorithm, called ReconfigDijkstra, and use it as the main procedure in heuristics algorithms.

Proposed Heuristics

- Demand First
- Gain Demand
- Gain Update
- Greedy Links

We have coded these algorithms by using Python's Gurobi so that we can look for the discrepancies between our algorithms in the future.

ReconfigDijkstra Algorithm

ReconfigDijkstra

- Single flow
- Dijkstra's Algorithm
- Does not update neighbours that are only visible by optical links when coming from an optical link.

[Foerster et al., 2018] showed that this algorithm generates optimal solutions for one flow where the triangular inequalities hold in the network.

Demand First Algorithm

Algorithm Demand First Algorithm

Input: Graph G, Demand Matrix D
Output: Resulting optical network design.
Sort the demand entries in D by size in descending order.
while *D is not empty OR all matchings have not been created*
do
(a) Pick the first entry in D.
(b) Run ReconfigDijkstra to route this demand.
(c) Update G by making used optical arcs permanent, eliminating all optical arcs that cannot be used, and delete the current demand from D.
end

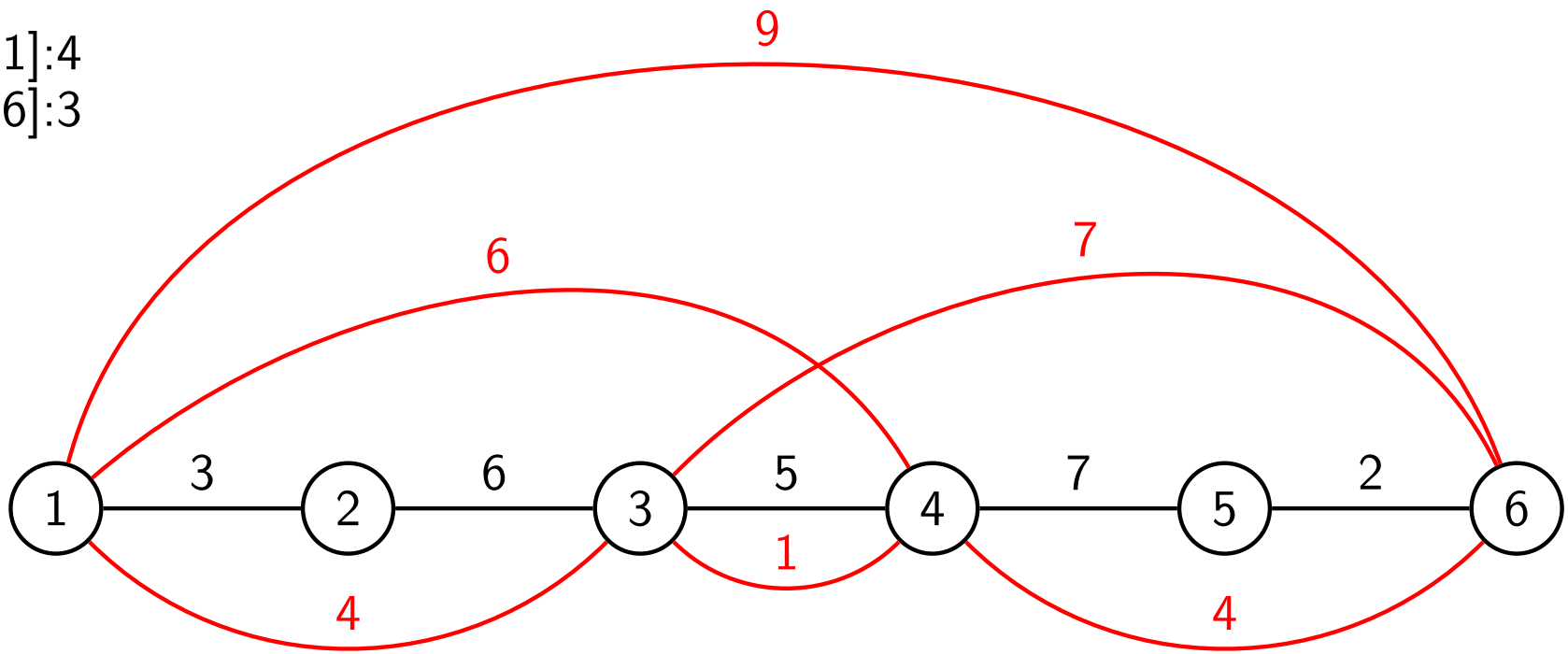
Gain Demand Algorithm

Algorithm Gain Demand Algorithm

Input: Graph G, Demand Matrix D
Output: Resulting optical network design.
1. Initialize an empty list called *gains*.
  for *Every entry in D*
  do
    (a) Run ReconfigDijkstra on G, add the value of the objective function in the updated G to value.
    (b) Reset G to its initial state.
end
2. Sort *gains* by size in increasing order.
  while *D is not empty OR all matchings have not been created*
  do
    (a) Pick the first entry in *gains*.
    (b) Run ReconfigDijkstra to route this demand.
    (c) Update G by making used matchings permanent, delete the current demand from *gains*.
end

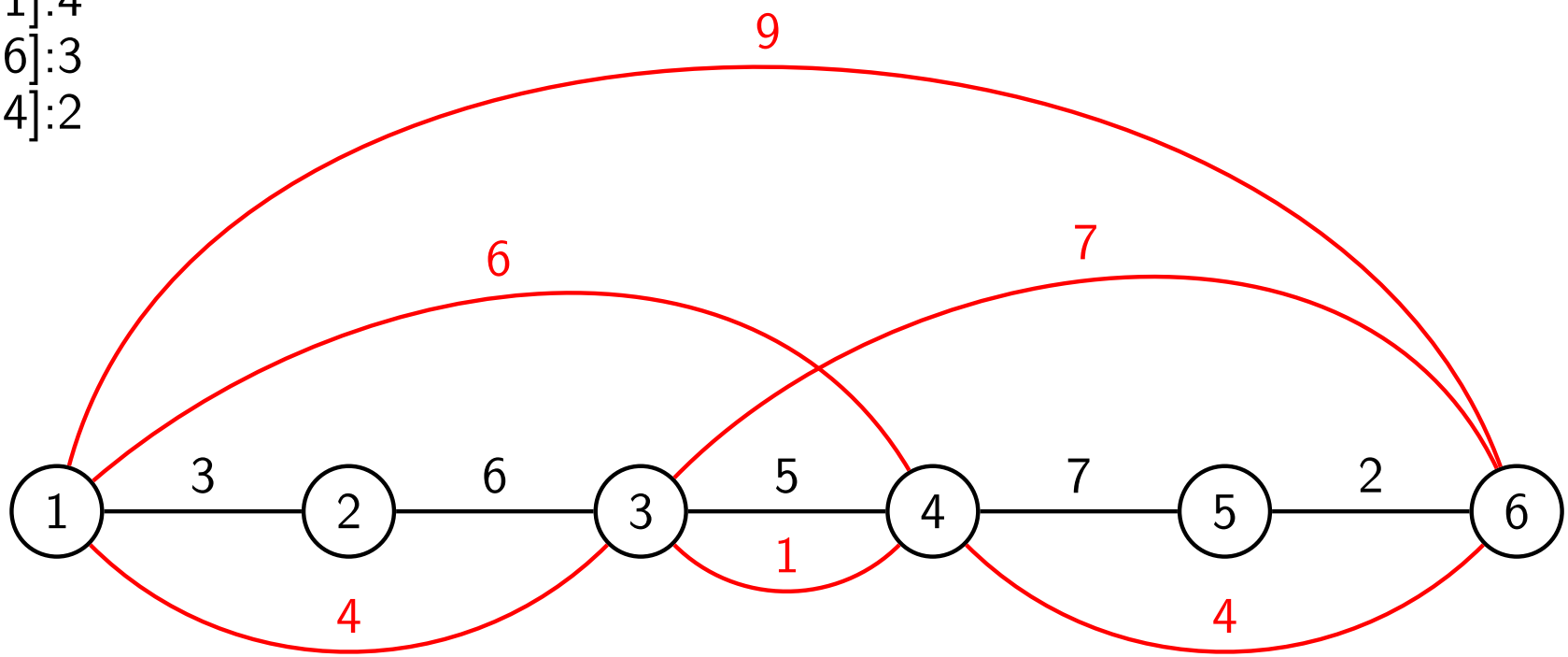
Counter Example for Demand First

D[4,1]:4
D[4,6]:3



Counter Example for Gain Demand

D[4,1]:4
D[4,6]:3
D[1,4]:2



Our Plan

Proposed Heuristics

- As we have shown above, proposed heuristics by [Fenz et al., 2019] result in inefficient solutions.
- We are planning to come up with a matheuristic approach.

Benefits of Matheuristic

- Allows to use elephants-mouse method to fix some matchings
- Reduces the computation time without deviating too much from the optimal solution

Conclusion

We showed that our model is much more faster than the [Fenz et al., 2019]'s one in every instance. Also, note that when the node size is large the difference between the CPU times of the models is even more noticeable.

In the second part of our runs, we showed that the more pairwise connections we have in the demand matrix, the complex the model becomes. Moreover, when we have relatively scarce number of connections in the optic topology, the complexity of the models decreases.

Conclusion(Cont')

We extended [Fenz et al., 2019]'s work by generalizing the problem for k reconfigurable switches whereas they proposed their model and heuristics for only one switch. We provided a mathematical model for the general case that can also be utilized when k equals to 1.

We also displayed that, by supplementing counter-examples, their heuristics end up with inefficient solutions in some cases, and pointed out that there might be a need for more efficient and accurate heuristic algorithms in the future.

Future Work

Planning to

- Come up with a new heuristic algorithms to outperform already proposed algorithms in terms of accuracy. Currently we are planning to use matheuristics.
- Try to use more powerful computers so that our exact model goes up to solve larger models without increasing the solution times.
- Transform our findings to a paper.

We also consider that this area of research is very promising for the near future and bound to generate many solutions to the problems that digital society has been facing for the last decades and will be facing in the coming ones.

References I

Bienkowski, M., Fuchssteiner, D., Marcinkowski, J., and Schmid, S. (2021).
Online dynamic b-matching: With applications to reconfigurable datacenter networks.
SIGMETRICS Perform. Eval. Rev., 48(3):99–108.

Fenz, T., Foerster, K.-T., Schmid, S., and Villedieu, A. (2019).
Efficient non-segregated routing for reconfigurable demand-aware networks.
In *2019 IFIP Networking Conference (IFIP Networking)*, pages 1–9.

Foerster, K.-T., Ghobadi, M., and Schmid, S. (2018).
Characterizing the algorithmic complexity of reconfigurable data center architectures.
In *Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems*, ANCS '18, page 89–96, New York, NY, USA. Association for Computing Machinery.

Foerster, K.-T., Pacut, M., and Schmid, S. (2019).
On the complexity of non-segregated routing in reconfigurable data center architectures.
SIGCOMM Comput. Commun. Rev., 49(2):2–8.

Foerster, K.-T. and Schmid, S. (2019).
Survey of reconfigurable data center networks: Enablers, algorithms, complexity.
volume 50, page 62–79, New York, NY, USA. Association for Computing Machinery.

Luo, L., Foerster, K.-T., Schmid, S., and Yu, H. (2019).
Dartree: Deadline-aware multicast transfers in reconfigurable wide-area networks.
In *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, pages 1–10.

Luo, L., Foerster, K.-T., Schmid, S., and Yu, H. (2020).
Splitcast: Optimizing multicast flows in reconfigurable datacenter networks.
In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 2559–2568.

References II

Yang, Z., Cui, Y., Xiao, S., Wang, X., Li, M., Li, C., and Liu, Y. (2019).
Achieving efficient routing in reconfigurable dcns.
Proc. ACM Meas. Anal. Comput. Syst., 3(3).