Algorithms for Data Traffic in Reconfigurable Optical Networks

PURDUE UNIVERSITY

Alex Pothen
S M Ferdous, B. Samineni,
M. Halappanavar, B. Krishnamoorthy

funded by U.S. DEPARTMENT OF ENERGY | Office of Science

March 17, 2026

1. Data Centers, k-Disjoint Matchings and k-Matchings
2. Computational Model for Streaming Matching
3. Streaming k-Disjoint Matching
4. Results and Conclusions

Data Centers and Matchings

Reconfigurable optical topologies can be used to design dynamic, demand-aware, data center networks. Optical switches are used to provide direct connectivity between racks, where each switch acts as a reconfigurable optical matching.

Data Centers and Matchings

Reconfigurable optical topologies can be used to design dynamic, demand-aware, data center networks. Optical switches are used to provide direct connectivity between racks, where each switch acts as a reconfigurable optical matching.

Given a traffic matrix (that determines which pairs of servers need to send and receive data from each other), and k optical switches, the problem becomes how to compute k heavy edge-disjoint matchings that carry large amounts of the data traffic for each switch.

Data Centers and Matchings

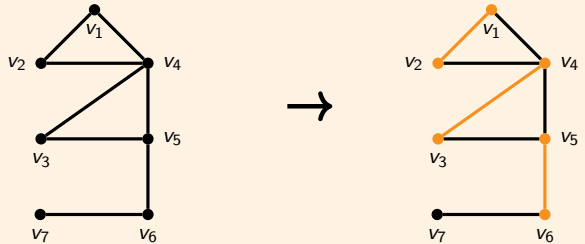
Reconfigurable optical topologies can be used to design dynamic, demand-aware, data center networks. Optical switches are used to provide direct connectivity between racks, where each switch acts as a reconfigurable optical matching.

Given a traffic matrix (that determines which pairs of servers need to send and receive data from each other), and k optical switches, the problem becomes how to compute k heavy edge-disjoint matchings that carry large amounts of the data traffic for each switch.

We formalize it as the k-edge-disjoint matching problem, where the goal is to maximize the sum of the weights of the edges in $k > 1$ edge-disjoint matchings.

Matchings

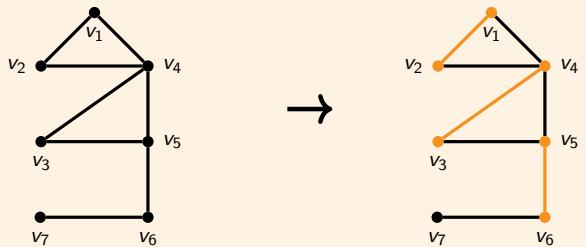
Matching



A subset of edges in a graph such that at most one edge is incident on each vertex. When edges are weighted, we seek a matching with the maximum sum of weights.

Matchings

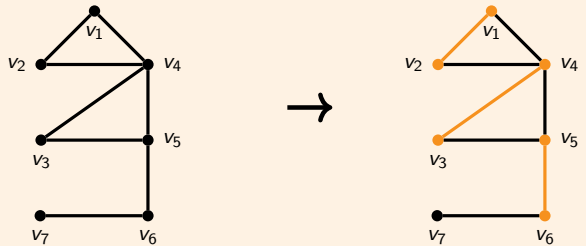
Matching



Exact algorithms are polynomial-time, but impractical for large graphs.

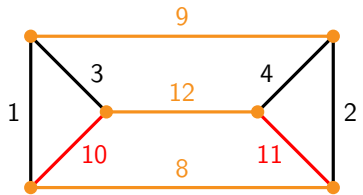
Matchings

Matching



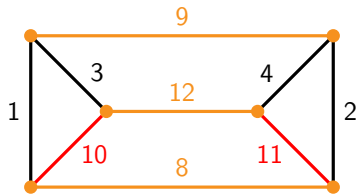
Approximation algorithms have been designed since the 1990's, but even these are impractical when edges are in the billions.
We implement two streaming matching algorithms.

k-disjoint Matching



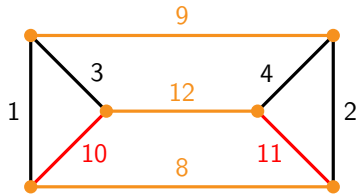
We compute k edge-disjoint matchings, with the maximum sum of weights of the matching edges. $k = 2$ disjoint matchings are shown.

k-disjoint Matching



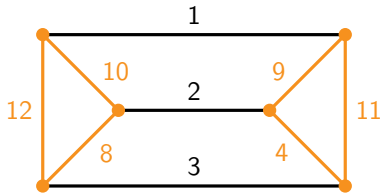
For $k \geq 2$, the problem is NP-hard, and also APX-hard (intractable to solve exactly or to approximate to high accuracy).

k-disjoint Matching



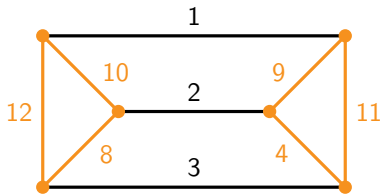
A few offline approximation algorithms have been designed recently, but even these are impractical when edges are in the billions.
We design and implement two streaming algorithms that reduce the memory required.

k-Matching (b-Matching)



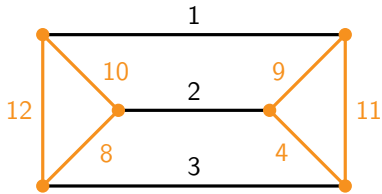
A subset of edges in a graph such that at most k edges are incident on each vertex.
With weights on edges, we seek a k-matching with the maximum sum of weights.

k-Matching (b-Matching)



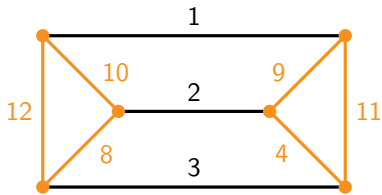
There are polynomial-time exact algorithms, but they are not practical for large graphs.

k-Matching (b-Matching)



A few offline approximation algorithms have been designed recently, but even these are impractical when edges are in the billions.

k-Matching (b-Matching)



We implement a streaming algorithm for k-matching and obtain a k-disjoint matching from it.

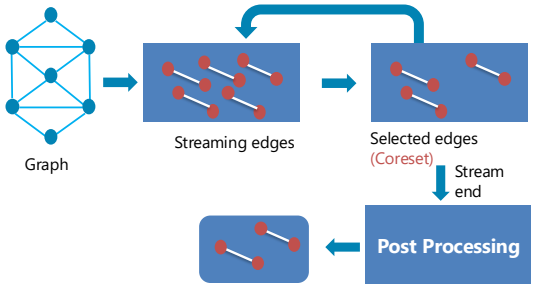
1 Data Centers, k-Disjoint Matchings and k-Matchings

2 Computational Model for Streaming Matching

3 Streaming k-Disjoint Matching

4 Results and Conclusions

Model for Streaming Matchings



Constraints

Memory $\propto$ Output

Metrics

Approximation ratio*

Runtime

Number of passes

Approx. Ratio = $\frac{Solution}{Optimal}$

Advantages of Streaming Algorithms

Problem	Output Size	Memory Required	Approx. Ratio	Time Complexity
Matching	$O(\|V\|)$	$O(\|V\| \log \|V\|)$	$1/(2+\varepsilon)$	$O(\|E\|)$
(Polystreaming)	$O(\|V\|)$	$O(P + \|V\| \log \|V\|)$	$1/(2+\varepsilon)$	$O((\|E\|/P) \log \|V\|)$
Edge Cover	$O(\|V\|)$	$O(\|V\| \log \|V\|)$	3/2, 2	$O(\|E\|)$
k-disjoint match	$O(k\|V\|)$	$O(k\|V\| \log^2 \|V\|)$	$1/(3+\varepsilon)$, $k/((k+1)(2+\varepsilon))$	$O(k\|E\|)$
Hypergraph match	$O(\|V\|)$	$(O(\|V\| \log^2 \|V\|)$	d^{-1}, $d^{-1.5}$	$O(d\|E\|)$

ε, parameter. P: number of processors. d: maximum size of a hyperedge.

Warm up: Non-streaming Weighted Matching Algorithm

Non-streaming algorithm

Iterate:

Select an edge e with positive weight;
Reduce its weight from itself and all neighboring edges;
Push e into a stack;

At the end:
Unwind the stack from the top; add edges greedily to the matching; once an edge is added to the matching, all of its neighboring edges in the stack are ineligible.

Warm up: Non-streaming Weighted Matching Algorithm

Non-streaming algorithm

Iterate:

Select an edge e with positive weight;
Reduce its weight from itself and all neighboring edges;
Push e into a stack;

At the end:
Unwind the stack from the top; add edges greedily to the matching; once an edge is added to the matching, all of its neighboring edges in the stack are ineligible.

A local ratio or a primal-dual analysis shows this algorithm is $1/2$-approximate.

1 Data Centers, k-Disjoint Matchings and k-Matchings

2 Computational Model for Streaming Matching

3 Streaming k-Disjoint Matching

4 Results and Conclusions

k-disjoint Weighted Matching

- Given an edge weighted graph, we wish to compute k edge-disjoint matchings, with the goal of maximizing the sum of the weights of the matchings.
- Hanauer, Henzinger, Schmid et al. (2022, 2023) have developed several $1/2$-approximation offline algorithms for computing kDM on static and dynamic graphs.
- We develop algorithms for this problem in a streaming computation model, generalizing a $1/(2+\varepsilon)$-approximate matching algorithm of Paz and Schwarzman (2019).

Streaming k-disjoint Weighted Matching

Semi-streaming Model

The edges of a graph arrive one by one, in arbitrary order. The algorithm must decide to store an edge or discard it, using information available when it arrives. At the end, k edge-disjoint matchings must be obtained from the edges stored. The total storage available is bounded by the size of the output, up to polylog factors.

- The goal is to compute an approximate k-disjoint weighted matching, using one pass through the data, bounding the memory needed, with a linear-time algorithm.

Streaming k-disjoint Weighted Matching

Semi-streaming Model

The edges of a graph arrive one by one, in arbitrary order. The algorithm must decide to store an edge or discard it, using information available when it arrives. At the end, k disjoint matchings must be obtained from the edges stored. The total storage available is bounded by the size of the output, up to polylog factors.

- The goal is to compute an approximate k-disjoint weighted matching, using one pass through the data, bounding the memory needed, with a linear-time algorithm.
- We design two algorithms:
  - The first algorithm relies on a primal-dual LP framework to obtain a $1/(3+\varepsilon)$-approximation, using $O(nk\log^2 n)$ memory, in $O(km)$ time.
  - The second algorithm computes a k-matching, and then uses an edge coloring algorithm to obtain a $(1/(2+\varepsilon))(1-1/(k+1))$-approximation using $O(nk\log^2 n)$ memory in $O(km+kn^2)$ time.
  - n is no. of vertices, m is no. of edges.

First Streaming k-disjoint Matching Algorithm

- Create a variable $\phi(c, v)$ for each color $c \in \{1, \ldots, k\}$ and $v \in V$, the sum of weights of edges already added to the stack c and incident on the vertex v.
- When an edge $e = (u, v)$ arrives, check if
$$w(e) > (1 + \varepsilon) * (\phi(c, u) + \phi(c, v)).$$
If c is an arbitrary color for which this holds, compute the gain
$$g(c, e) = w(e) - (\phi(c, u) + \phi(c, v));$$
add the gain $g(c, e)$ to $\phi(c, u)$ and $\phi(c, v)$; push e into the stack c;
If there is no such color c, discard the edge.
- Arbitrarily order the stacks from 1 to k. Unwind the stacks in increasing order, and construct k matchings from the k stacks. If an edge cannot be added to the matching from the current stack, check if it can be added to succeeding stacks.
- This leads to a $1/(3 + \varepsilon)$-approximate algorithm, and bounds the size of the stack.
- We can analyze the algorithm using a primal dual linear programming framework.

Primal LP Relaxation

$$
\begin{aligned}
\text{(P) max} \quad & \sum_{c\in[k]}\sum_{e\in E} w(e)x(c,e) \\
\text{s.t.} \quad & \sum_{e\in\delta(v)} x(c,e) \le 1 \;\forall v \in V, c \in [k] \\
& \sum_{c\in[k]} x(c,e) \le 1 \quad \forall e \in E \\
& x(c,e) \ge 0 \qquad\quad \forall e \in E, c \in [k].
\end{aligned}
$$

v is vertex, e is edge
$c \in [k] = \{1, \ldots, k\}$ is color
$x(c, e)$ is 1 if e in c-th matching;
0 else.

Dual LP Relaxation

(P) max $\sum_{c\in[k]}\sum_{e\in E} w(e)x(c,e)$

s.t. $\sum_{e\in\delta(v)} x(c,e) \le 1 \;\forall v \in V, c \in [k]$

$\sum_{c\in[k]} x(c,e) \le 1 \quad \forall e \in E$

$x(c,e) \ge 0 \qquad \forall e \in E, c \in [k].$

(D) min $\sum_{c\in[k]}\sum_{v\in V} y(c,v) + \sum_{e\in E} z(e)$

s.t. $y(c,e) + z(e) \ge w(e) \;\forall e = (u,v) \in E, c \in [k]$

$y(c,v) \ge 0 \qquad \forall v \in V, c \in [k]$

$z(e) \ge 0 \qquad \forall e \in E.$

$y(c,v)$, $z(e)$ are dual variables.
$y(c,e) := y(c,u) + y(c,v)$.

v is vertex, e is edge
$c \in [k] = \{1,\dots,k\}$ is color
$x(c,e)$ is one if e is in c-th matching

Second Algorithm: k-disjoint Matching from a k-matching

Theorem

Let F be a $1/(2+\varepsilon)$-approximate weighted k-matching in a graph G. If the induced subgraph $G[F]$ is edge colored using $(k+1)$ colors, then the collection of edges in the k heaviest color classes constitutes a $(1/(2+\varepsilon)\,(1-1/(k+1))$-approximate weighted k-disjoint matching.

The streaming $1/(2+\varepsilon)$-approximation algorithm for the maximum weight k-matching problem requires $O(nk\,\log^2 n)$ memory and is due to Huang and Sellier (2021).

1 Data Centers, k-Disjoint Matchings and k-Matchings

2 Computational Model for Streaming Matching

3 Streaming k-Disjoint Matching

4 Results and Conclusions

Computational Results

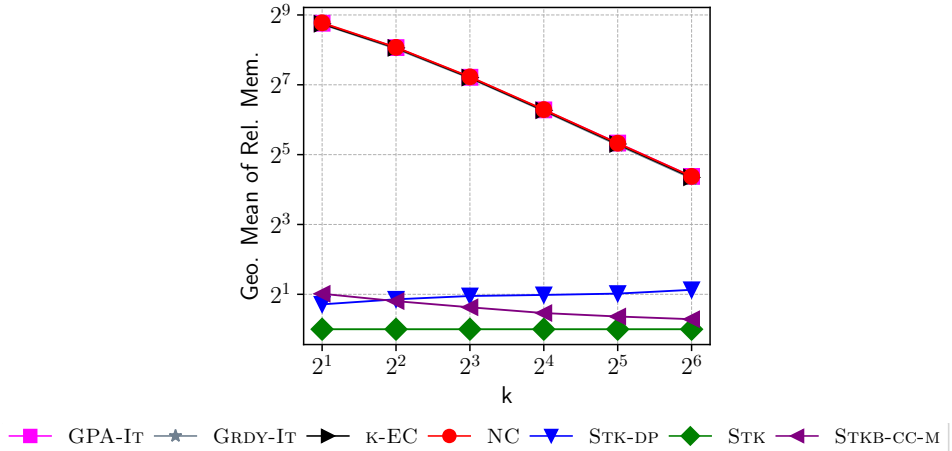
- First we report results for the Primal-Dual streaming algorithm on billion plus edge graphs from SuiteSparse collection, where offline algorithms cannot be run on a machine with 1 TB memory.
- Next we report results on graphs generated from Network traces at Data Centers, where the k-DM problem could be used to compute reconfigurable optical matchings. The offline algorithms considered here are the best performers among several 1/2-approximation matching and heuristic algorithms implemented by Hanauer, Henzinger and Schmid (2022). (Global Paths algorithm, Greedy, and an Edge-coloring algorithm).
- Exact algorithms consume much more time, and are not practical.
- Computations are done on a node of a cluster computer with 128 cores in an AMD EPYC 7662 with 1 TB of total memory over all the cores.

Streaming on Billion Edge Graphs from SuiteSparse

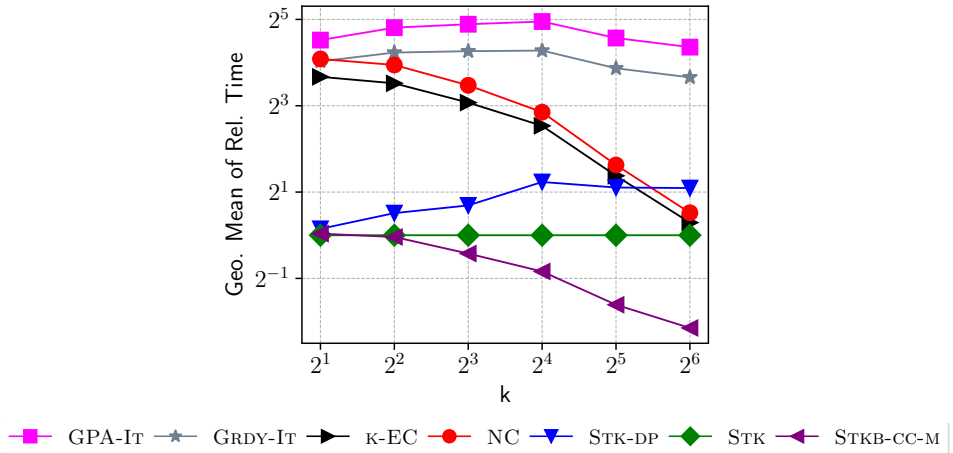
Graph	Time (sec)	Weight	Memory (GB)
Agatha_2015	1378	1.60e14	49.4
MOLIERE_2016	737	8.26e06	23.28
GAP-kron	629	1.20e10	29.66
GAP-urand	680	9.8e10	53.25
com-Friendster	475	1.02e14	22.66
mycielskian20	86	1.99e12	0.65

Results for $k = 8$, $\varepsilon = 1e - 3$. DP improves weights by up to 5%, with twice the time and memory. Off-line 1/2-approximate algorithms do not terminate on a 1 TB computer on all but the last problem!

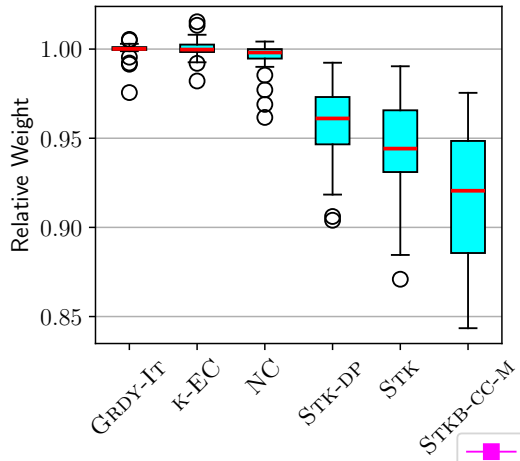
Streaming vs. Offline, Memory: Facebook Data Center



Streaming vs. Offline, Time: Facebook Data Center



Streaming vs. Offline, Weights: Facebook Data Center



Conclusions

- Matchings in graphs have a number of applications in modern data centers: efficient data traffic using reconfigurable optical switches, load balancing along network paths, scheduling traffic using crossbar switches, rack to rack communications in hierarchical networks, etc.
- There is a natural progression in algorithms for matchings as graphs increase in size: from exact algorithms to approximation algorithms, and now to streaming and dynamic algorithms.
- Constant factor approximation algorithms can be implemented in parallel, and obtain, in practice, near-optimal matchings.
- Worst-case approximation ratios do not necessarily predict the weight of an approximate matching computed by an algorithm. Relative performances of algorithms need to be evaluated empirically.
- The streaming algorithms presented here find approximate matchings and k-disjoint matchings, in one pass, in near-linear time and memory.

Our Recent Work on Matchings

S M Ferdous, Bhargav Samineni, Alex Pothen, Mahantesh Halappanavar, and Bala Krishnamoorthy, Semi-streaming algorithms for weighted k-disjoint matchings, European Symposium on Algorithms (ESA), 2024.

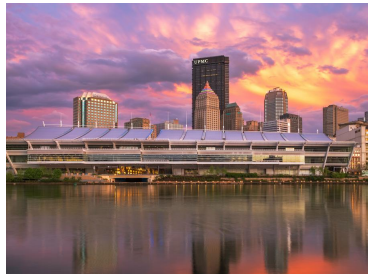
Ahammed Ullah, S M Ferdous and Alex Pothen, Weighted Matching in a Poly-streaming Model, European Symposium on Algorithms (ESA), 2025. Best Paper in the Computation Track.

Henrik Reinstadtler, S M Ferdous, Alex Pothen, Bora Ucar and Christian Schulz, Semistreaming Algorithms for Hypegraph Matching, European Symposium on Algorithms (ESA), 2025.

S M Ferdous, Alex Pothen, Arif Khan, Ajay Panyala and Mahantesh Halappanavar, A Parallel Approximation Algorithm for Maximizing Submodular b-matching, First SIAM Conference on Applied and Computational Combinatorial Algorithms (ACDA), 2021.

Alex Pothen, S M Ferdous and Fredrik Manne, Approximation algorithms in combinatorial scientific computing, Acta Numerica, 2019.

SIAM Conference on ACDA 2027



Fourth SIAM Conference on Applied and Computational Discrete Algorithms Feb. 22-24, 2027, Pittsburgh, PA
The conference features both refereed papers published in a Proceedings, and talks and posters without papers.

Call for papers will be at
www.siam.org/conferences-events/siam-conferences/acda27/