Large-Scale Generalized Matching

Baruch Awerbuch, Rainer Gemulla, Rohit Khandekar, Faraz Makari, Julián Mestre, Mauro Sozio

max planck institut informatik

Bob
wants to watch
a movie

E-Mail from
DVD rental store

Bob

wants to watch

Avatar

E-Mail from

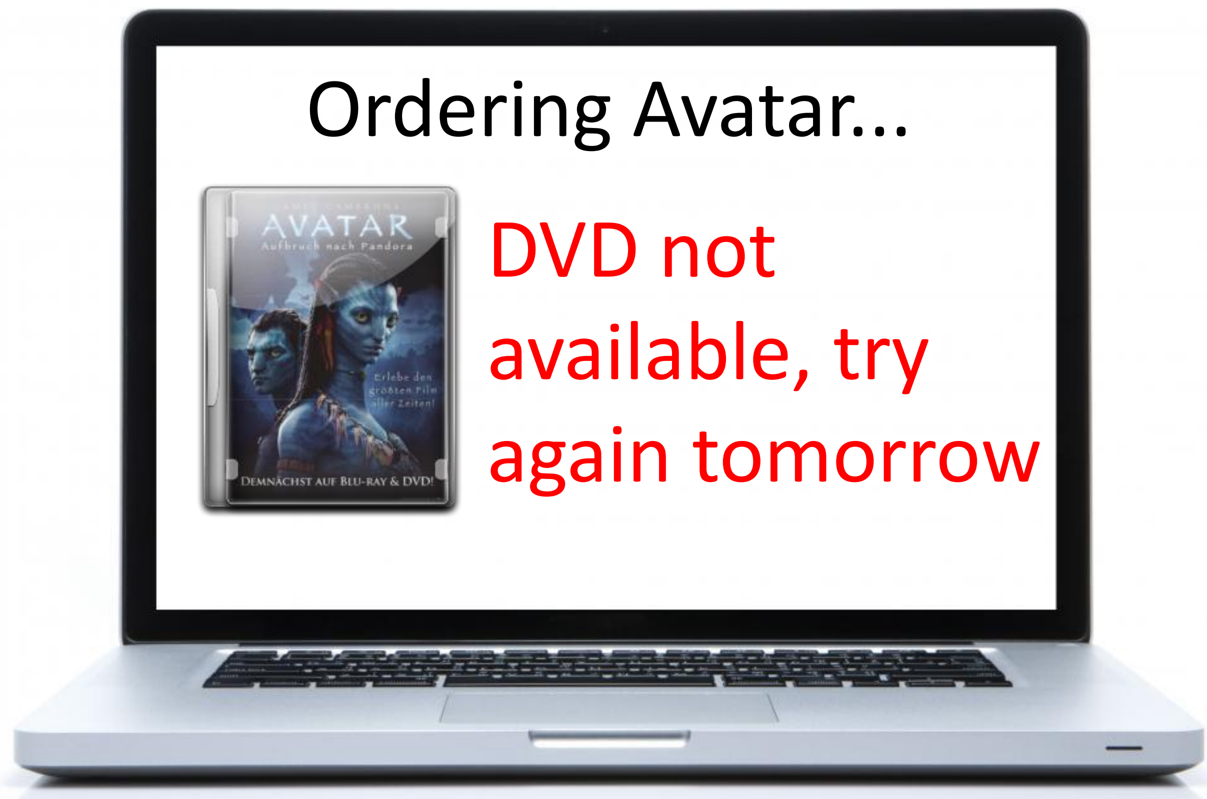
DVD rental store

Bob

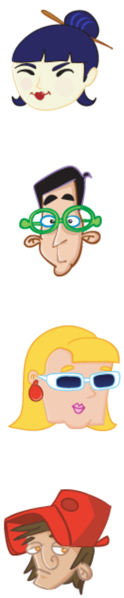
wants to watch

Avatar

DVD rental store

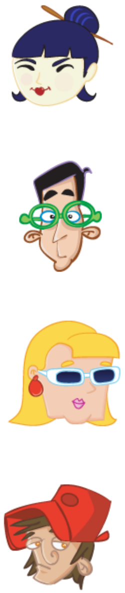


A Simple Recommender System



?	?	2
?	?	?
4	?	?
?	5	?

A Simple Recommender System

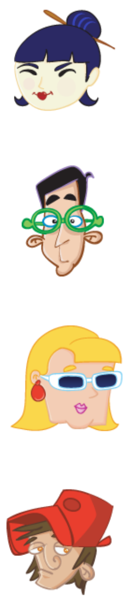


4	1	2
5	1	3
4	4	2
3	5	2

Step 1:
Predict preference

Step 2:
Recommend preferred movies

A Simple Recommender System

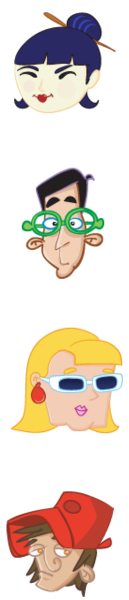


4	1	2
5	1	3
4	4	2
3	5	2

Step 1:
Predict preference

Step 2:
Recommend preferred movies

A Simple Recommender System

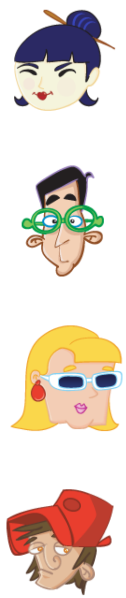


4	1	2
5	1	3
4	4	2
3	5	2

Step 1:
Predict preference

Step 2:
Recommend preferred movies *under constraints*

A Simple Recommender System

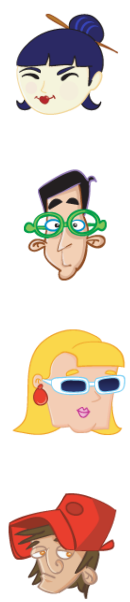


4	1	2
5	1	3
4	4	2
3	5	2

Step 1:
Predict preference

Step 2:
Recommend preferred movies *under constraints*

A Simple Recommender System



4	1	2
5	1	3
4	4	2
3	5	2

Step 1:
Predict preference

Step 2:
Recommend preferred movies *under constraints*

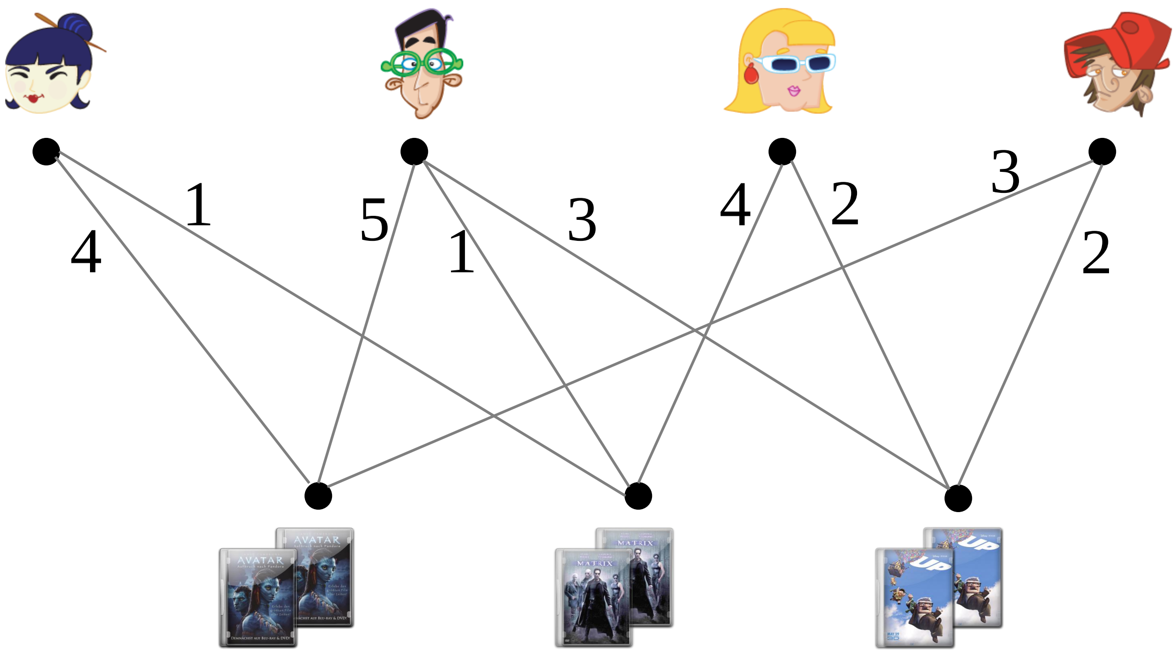
This talk

Outline

1. Introduction
2. **Generalized bipartite matching**
3. Distributed mixed packing/covering LPs
4. Distributed rounding
5. Experiments
6. Conclusion

Generalized Bipartite Matching

Given a weighted bipartite graph, degree constraints.

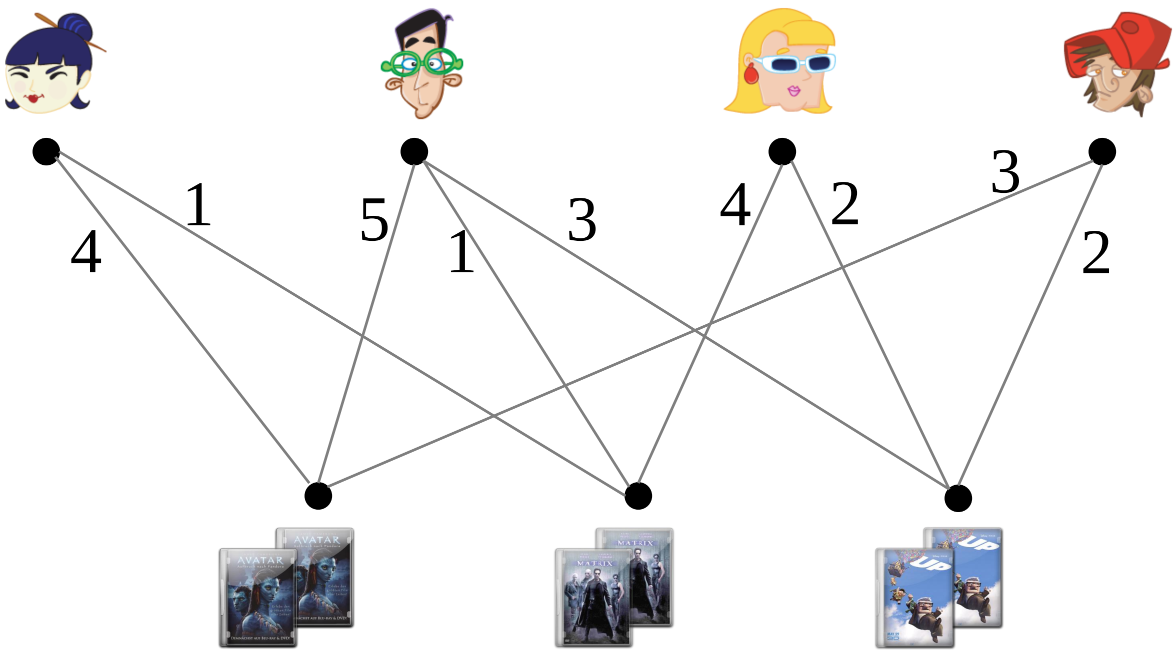


$$\mathbf{1 \leq d_u \leq 1}$$

$$\mathbf{0 \leq d_v \leq 2}$$

Generalized Bipartite Matching

Given a weighted bipartite graph, degree constraints.
Find maximum-weight subset of edges satisfying constraints.

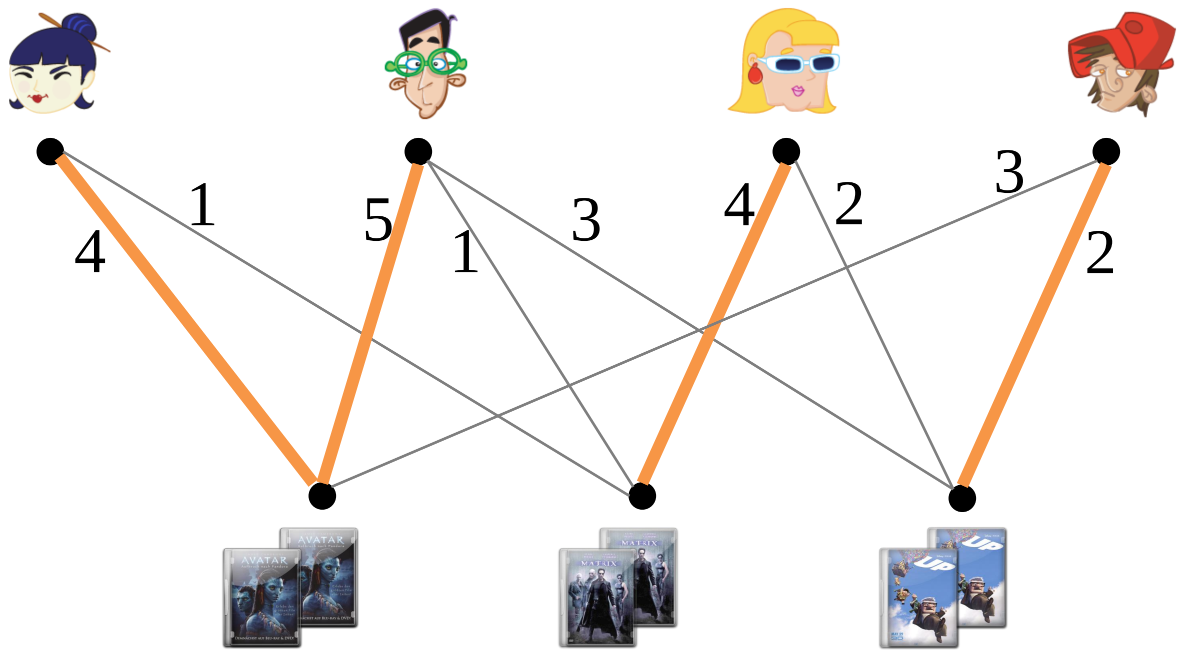


$$1 \le d_u \le 1$$

$$0 \le d_v \le 2$$

Generalized Bipartite Matching

Given a weighted bipartite graph, degree constraints.
Find maximum-weight subset of edges satisfying constraints.

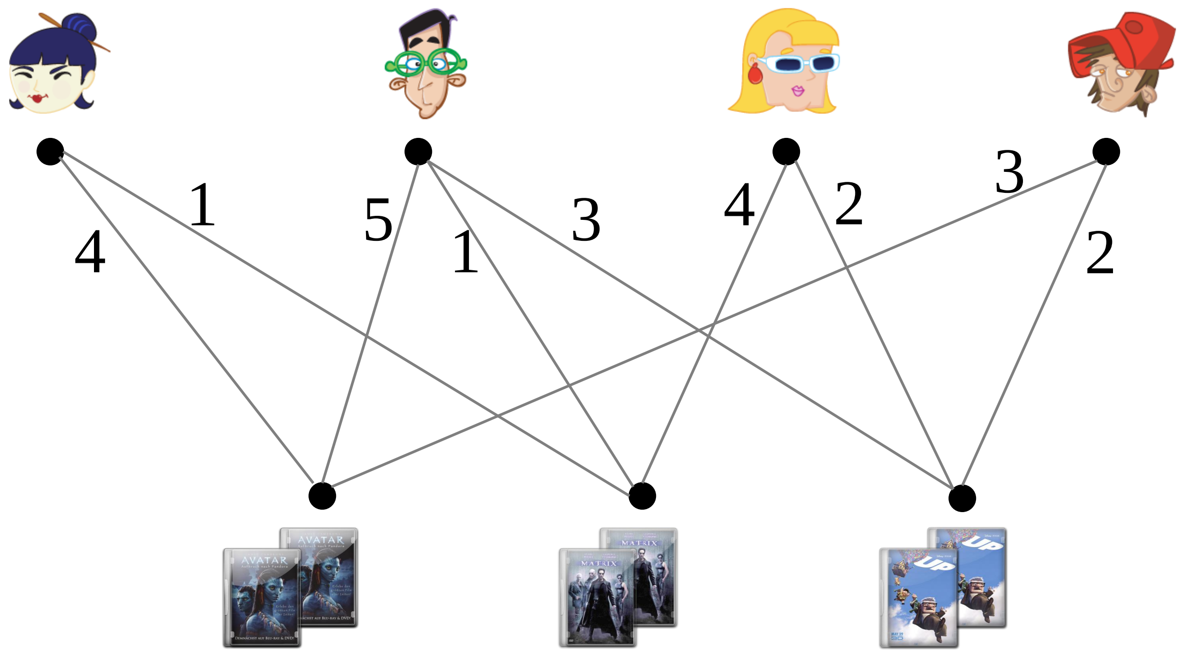


$$1 \leq d_u \leq 1$$

$$0 \leq d_v \leq 2$$

Generalized Bipartite Matching

Given a weighted bipartite graph, degree constraints.
Find maximum-weight subset of edges satisfying constraints.

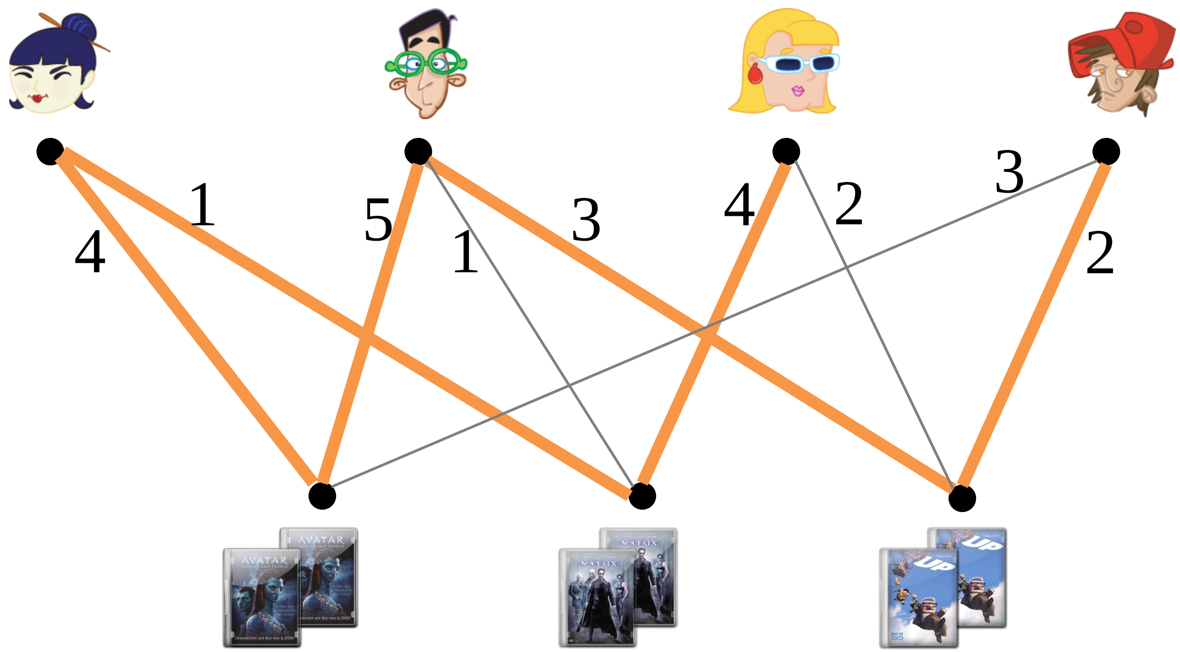


$$\mathbf{1 \leq d_u \leq 2}$$

$$\mathbf{0 \leq d_v \leq 2}$$

Generalized Bipartite Matching

Given a weighted bipartite graph, degree constraints.
Find maximum-weight subset of edges satisfying constraints.



$$\mathbf{1 \leq d_u \leq 2}$$

$$\mathbf{0 \leq d_v \leq 2}$$

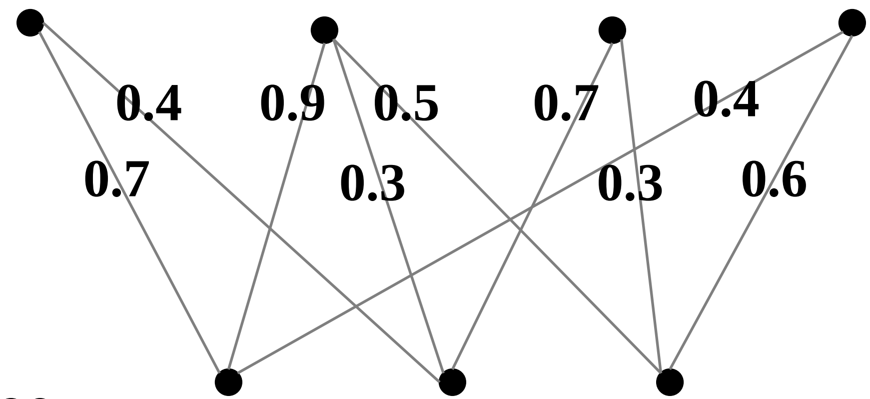
Solving GBM Problems

- Optimally solvable in PTIME
  - E.g., via a linear programming formulation
  - Small to medium-size instances well handled by out-of-the-box solvers
- Instances can get very large
  - E.g., Netflix has 20M users, 10k's of items
  - Available solvers break down

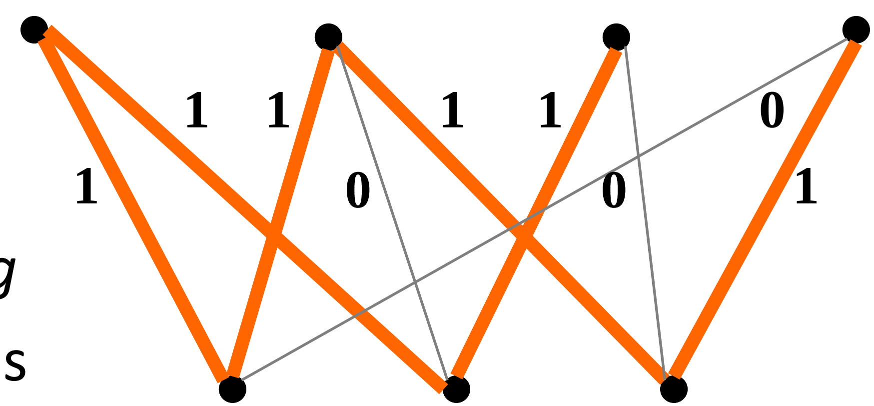
How can we solve GBM and related problems in a scalable and efficient way?

Overview

- **Phase 1:** LP relaxation
  - Outputs “edge probabilities”
  - Mixed packing-covering LP
  - *Distributed approximate solver*
  - Strong approximation guarantees



- **Phase 2:** Rounding
  - Selects actual matching
  - *Distributed dependent rounding*
  - Good approximation guarantees



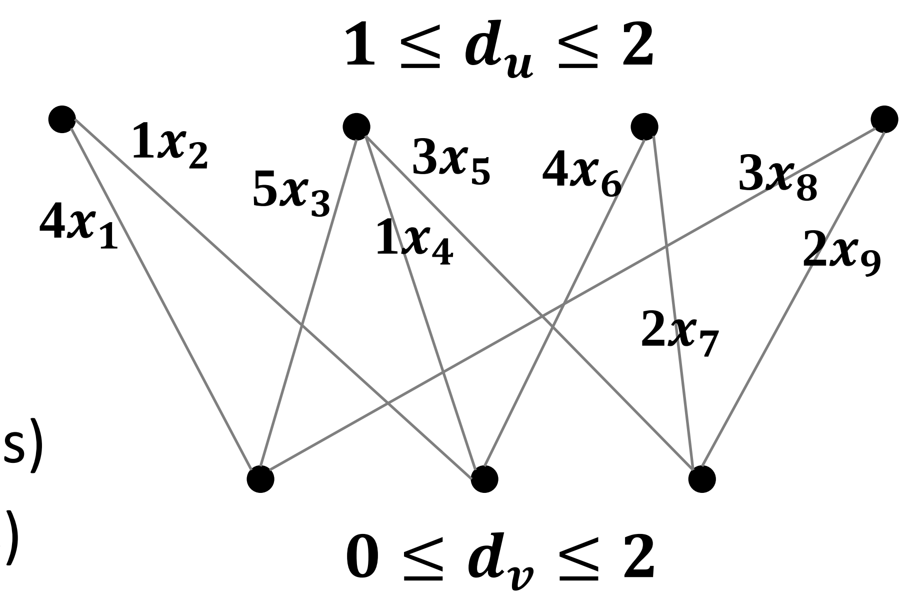
Outline

1. Introduction
2. Generalized bipartite matching
3. **Distributed mixed packing/covering LPs**
4. Distributed rounding
5. Experiments
6. Conclusion

Mixed Packing-Covering LP

maximize	$w^T x$	
subject to	$Px \leq 1$	(packing constraints)
	$Cx \geq 1$	(covering constraints)
	$x \geq 0$	

- *w*, *P*, *C* all non-negative
- For GBM
  - *w*: edge weights (Bs)
  - *x*: edge variables (Bs)
  - Covering c.: lower bounds (Ms)
  - Packing c.: upper bounds (Ms)

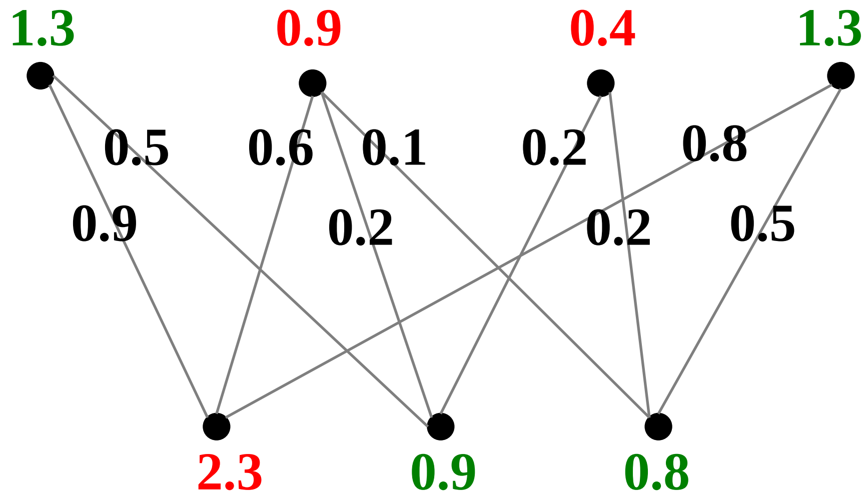


MPCSolver

- General parallel solver for MPC problems
  - *Fast convergence*: polylog rounds
  - *Almost feasible*: All constraints satisfied up to $(1 \pm \varepsilon)$
  - *Near optimal*: Objective at least $(1 - \varepsilon)$ of optimum
  - *Easy to implement*: matrix-vector operations

$\mathbf{1 \leq d_u \leq 2}$

$\mathbf{0 \leq d_v \leq 2}$

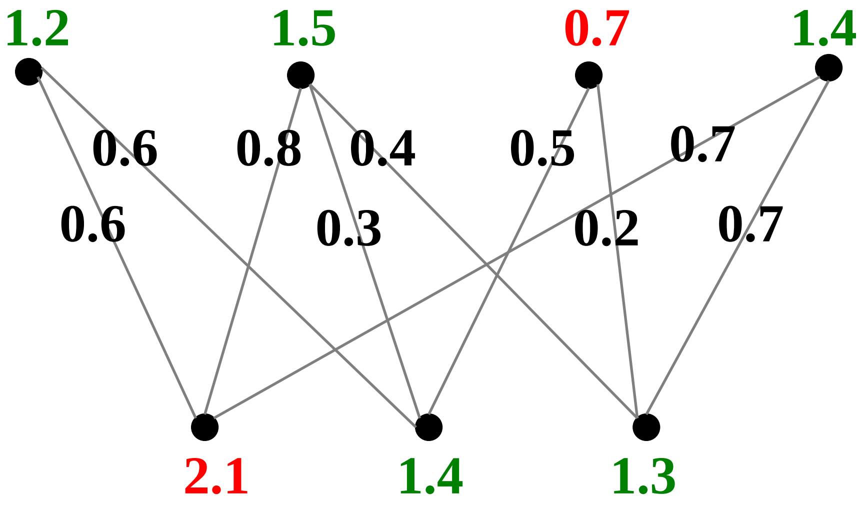


MPCSolver

- General parallel solver for MPC problems
  - *Fast convergence*: polylog rounds
  - *Almost feasible*: All constraints satisfied up to $(1 \pm \varepsilon)$
  - *Near optimal*: Objective at least $(1 - \varepsilon)$ of optimum
  - *Easy to implement*: matrix-vector operations

$\mathbf{1 \leq d_u \leq 2}$

$\mathbf{0 \leq d_v \leq 2}$

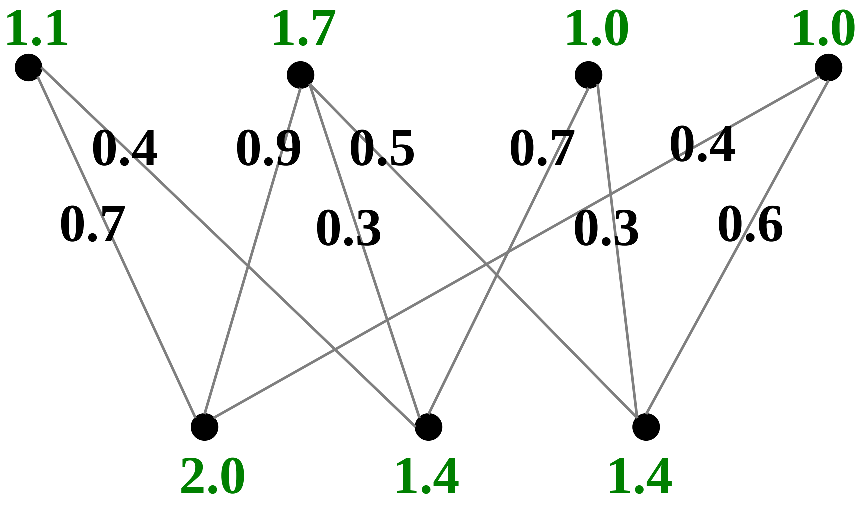


MPCSolver

- General parallel solver for MPC problems
  - *Fast convergence*: polylog rounds
  - *Almost feasible*: All constraints satisfied up to $(1 \pm \varepsilon)$
  - *Near optimal*: Objective at least $(1 - \varepsilon)$ of optimum
  - *Easy to implement*: matrix-vector operations

$\mathbf{1 \leq d_u \leq 2}$

$\mathbf{0 \leq d_v \leq 2}$



Algorithm (ε-feasibility)

repeat
 Compute $y_i(x) = \exp\left[\mu \cdot \left(\boldsymbol{P}_i x - 1\right)\right]$ for $i = 1, \ldots$
 Compute $z_i(x) = \exp\left[\mu \cdot \left(1 - \boldsymbol{C}_i x\right)\right]$ for $i = 1, \ldots$
 for $j = 1, \ldots, n$ **do**
  if $\frac{\boldsymbol{P}_j^\top y(x)}{\boldsymbol{C}_j^\top z(x)} \leq 1 - \alpha$ **then**
   $x_j \leftarrow \max\{x_j(1+\beta), \delta\}$
  if $\frac{\boldsymbol{P}_j^\top y(x)}{\boldsymbol{C}_j^\top z(x)} \geq 1 + \alpha$ **then**
   $x_j \leftarrow x_j(1-\beta)$
until convergence (Sec. 3.3)

$\tilde{O}\left(\frac{1}{\varepsilon^5}\log^3(kmMnx_{\max})\right)$ rounds.

MPCSolver in Practice

- Parallelization
  - Straightforward on shared-memory or GPU
  - Intelligent data placement and synchronization for shared-nothing
  - Also fits MapReduce framework
- From feasibility to near-optimality
  - Obtain lower bound $\lambda_{\min}$ and upper bound $\lambda_{\max}$ on objective (only covering or only packing)
  - Add constraint $w^T x \geq \lambda$
  - Binary search for λ in $\log_2 \log_{1-\varepsilon}(\lambda_{\min}/\lambda_{\max})$ steps

Outline

1. Introduction
2. Generalized bipartite matching
3. Distributed mixed packing/covering LPs
4. **Distributed rounding**
5. Experiments
6. Conclusion

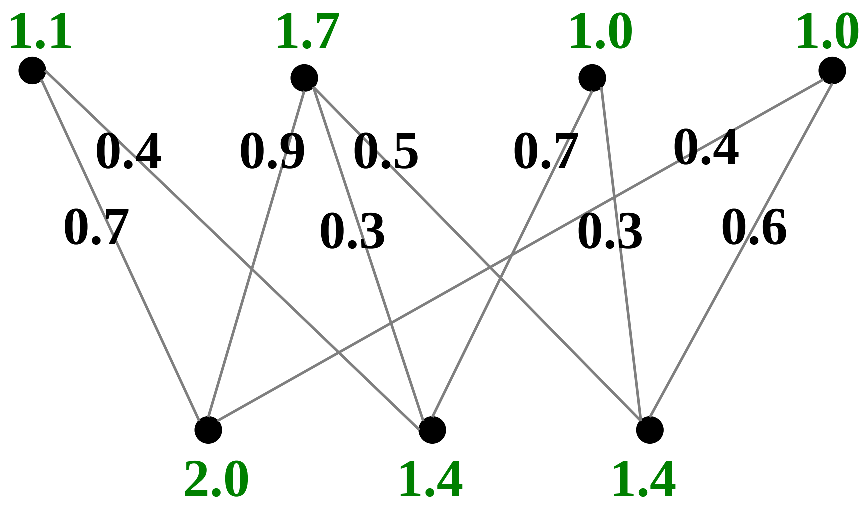
GBM Rounding

Given a near-optimal, ε-feasible fractional solution to GBM.
Find an integral solution that

1) preserves ε-feasibility (up to rounding) and
2) preserves near-optimality.

Independent rounding

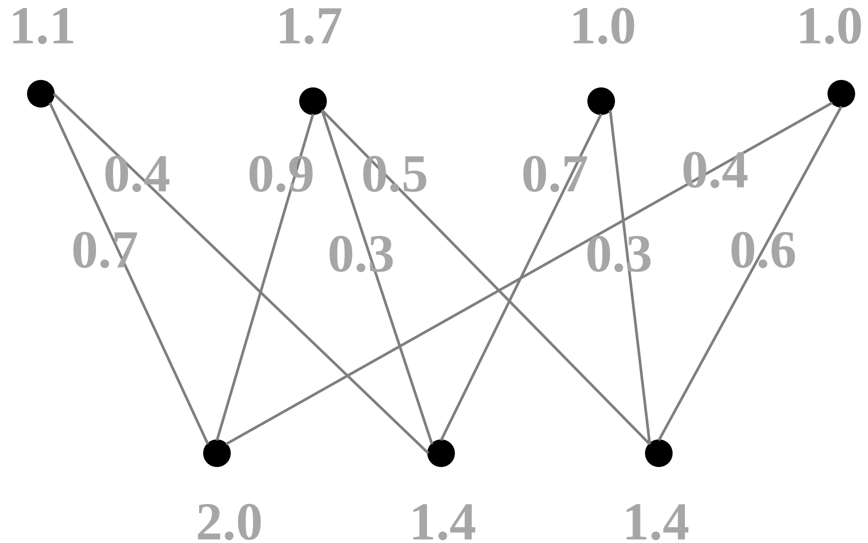
- Naive approach
- Satisfies (2) in expectation
- Violates (1)



Dependent Rounding

Sequential algorithm by Gandhi et al., 2006

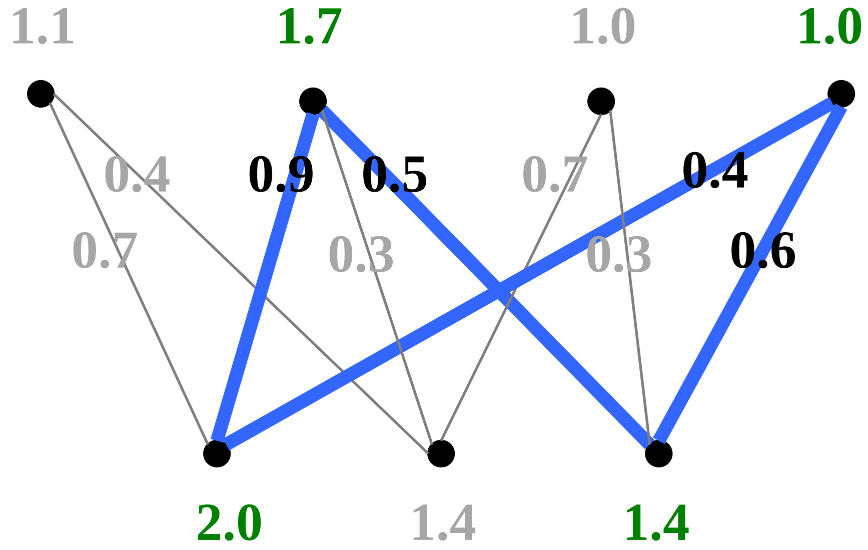
1. Find a fractional cycle or maximal path



Dependent Rounding

Sequential algorithm by Gandhi et al., 2006

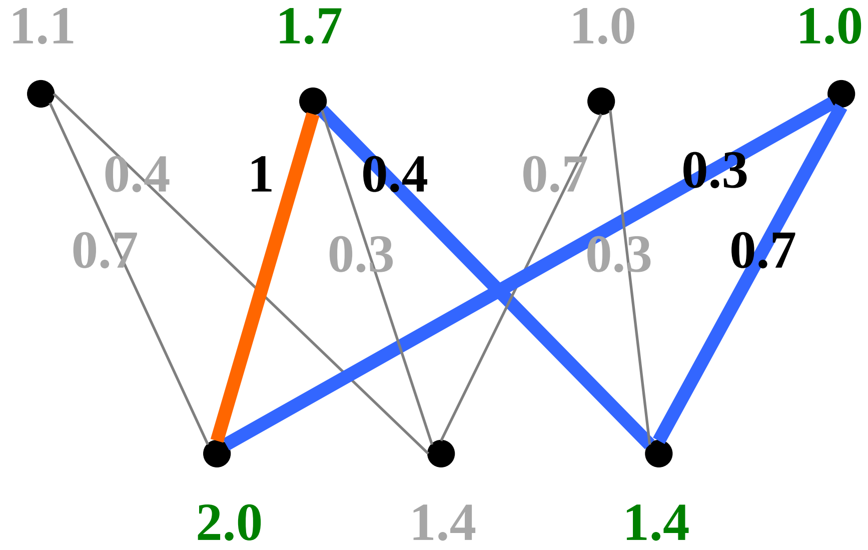
1. Find a fractional cycle or maximal path
2. Round ≥1 edge on the cycle/path



Dependent Rounding

Sequential algorithm by Gandhi et al., 2006

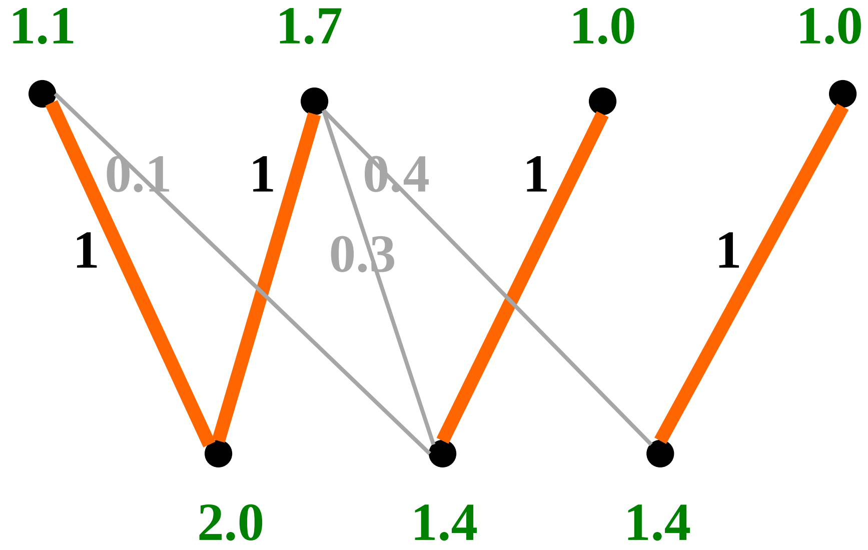
1. Find a fractional cycle or maximal path
2. Round ≥1 edge on the cycle/path
3. Repeat



Dependent Rounding

Sequential algorithm by Gandhi et al., 2006

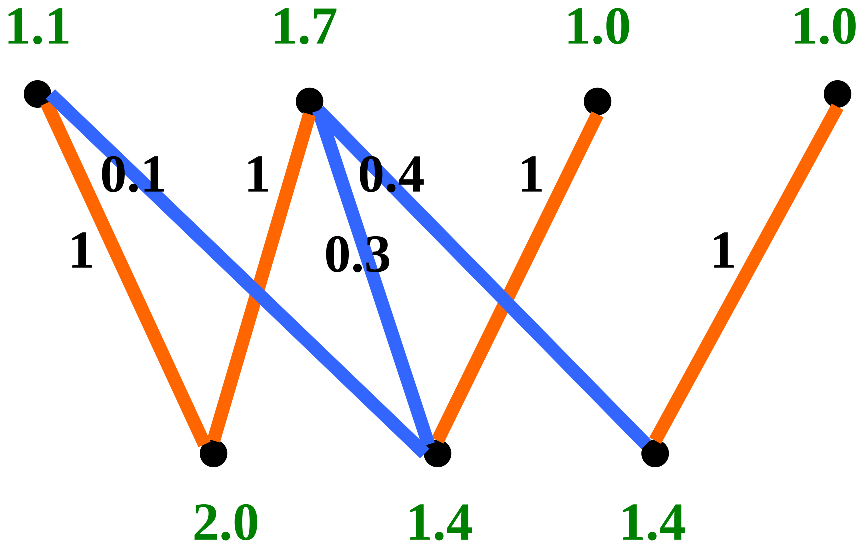
1. Find a fractional cycle or maximal path
2. Round $\geq$1 edge on the cycle/path
3. Repeat



Dependent Rounding

Sequential algorithm by Gandhi et al., 2006

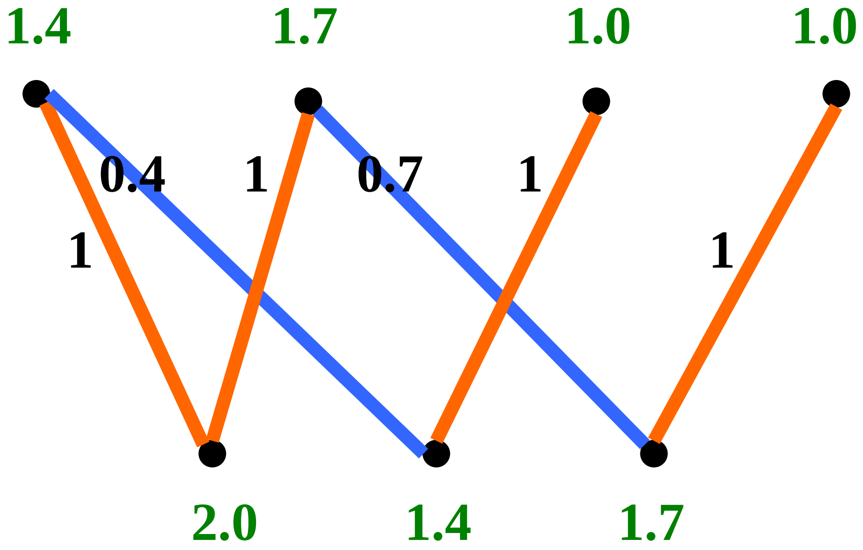
1. Find a fractional cycle or maximal path
2. Round ≥1 edge on the cycle/path
3. Repeat



Dependent Rounding

Sequential algorithm by Gandhi et al., 2006

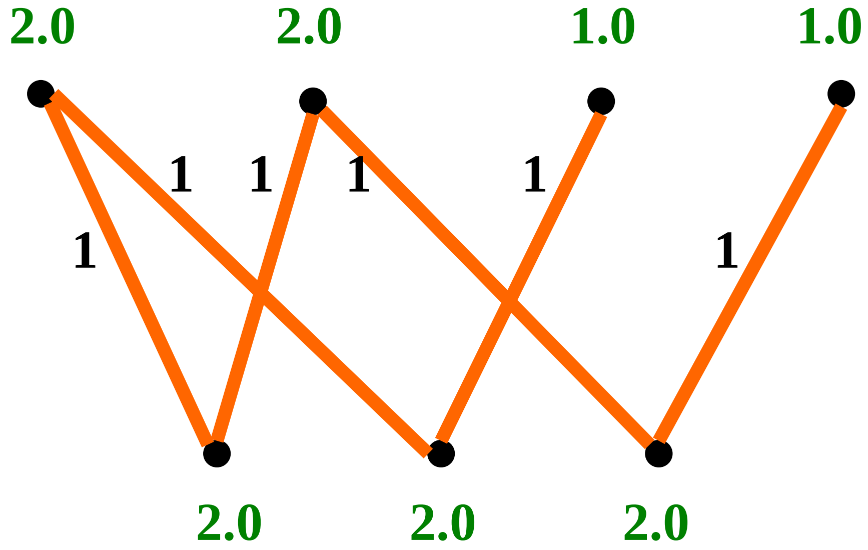
1. Find a fractional cycle or maximal path
2. Round $\geq$1 edge on the cycle/path
3. Repeat



Dependent Rounding

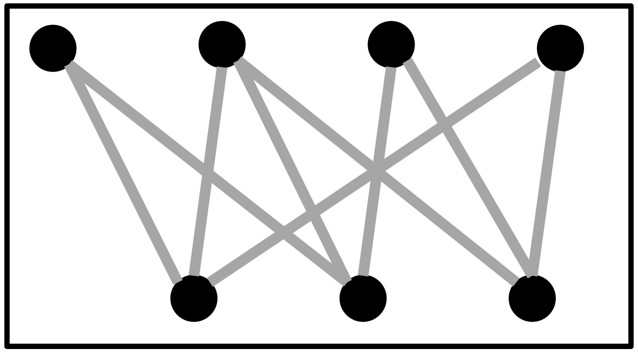
Sequential algorithm by Gandhi et al., 2006

1. Find a fractional cycle or maximal path
2. Round ≥1 edge on the cycle/path
3. Repeat



Distributed Rounding

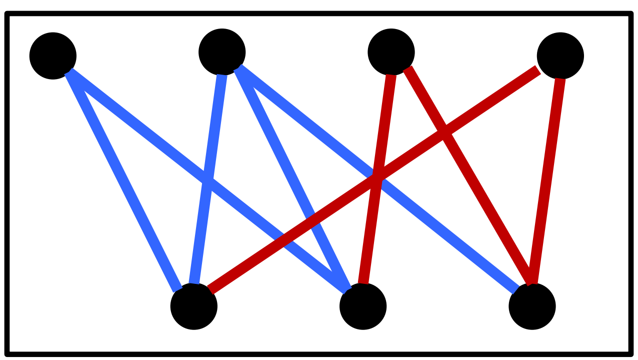
- Partitioning of edges to compute nodes



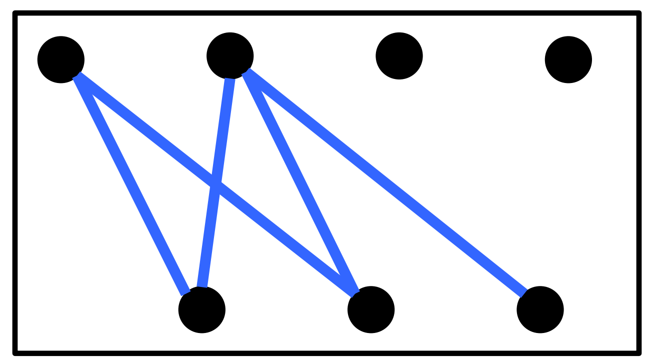
Full graph

Distributed Rounding

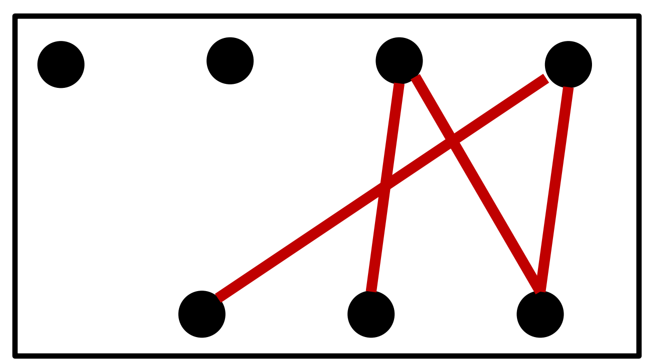
- Partitioning of edges to compute nodes
- A local cycle is a global cycle
- A local maximal path may not be globally maximal



Full graph



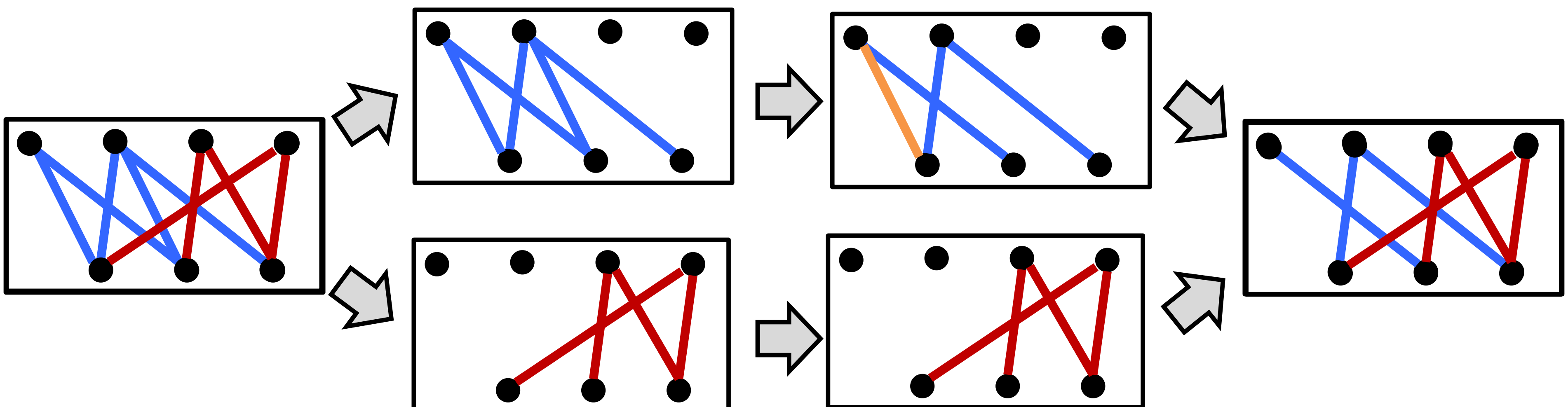
Node 1



Node 2

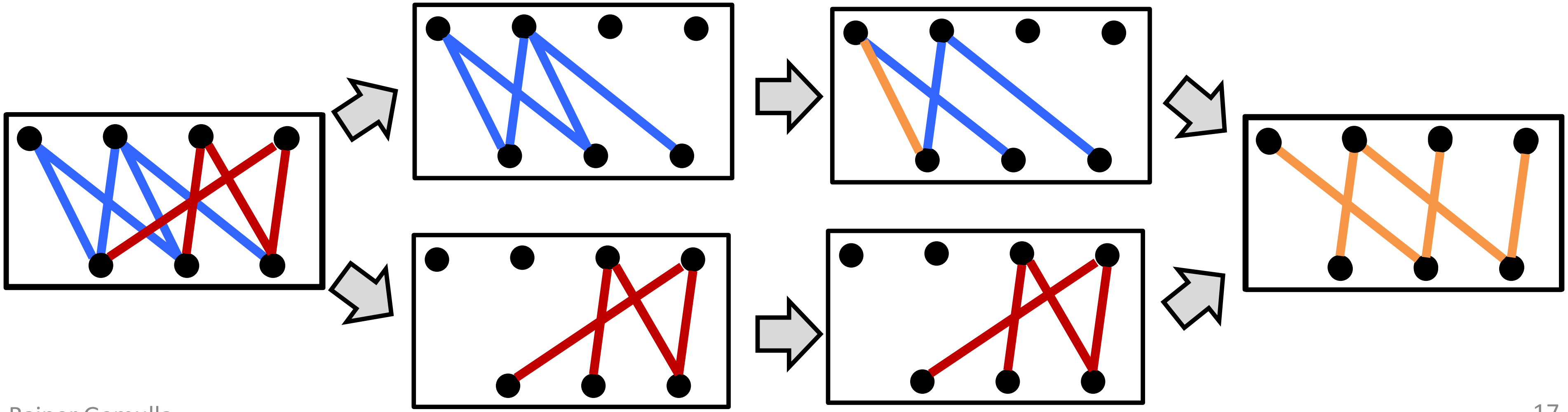
Algorithm

1. Partition edges (fractional only)
2. Process local cycles
   - *k* compute nodes, *m* vertices → *O*(*km*) edges left
3. Repeat until graph small
4. Process rest sequentially (cycles and max. paths)



Algorithm

1. Partition edges (fractional only)
2. Process local cycles
   - k compute nodes, m vertices → $O(km)$ edges left
3. Repeat until graph small
4. Process rest sequentially (cycles and max. paths)



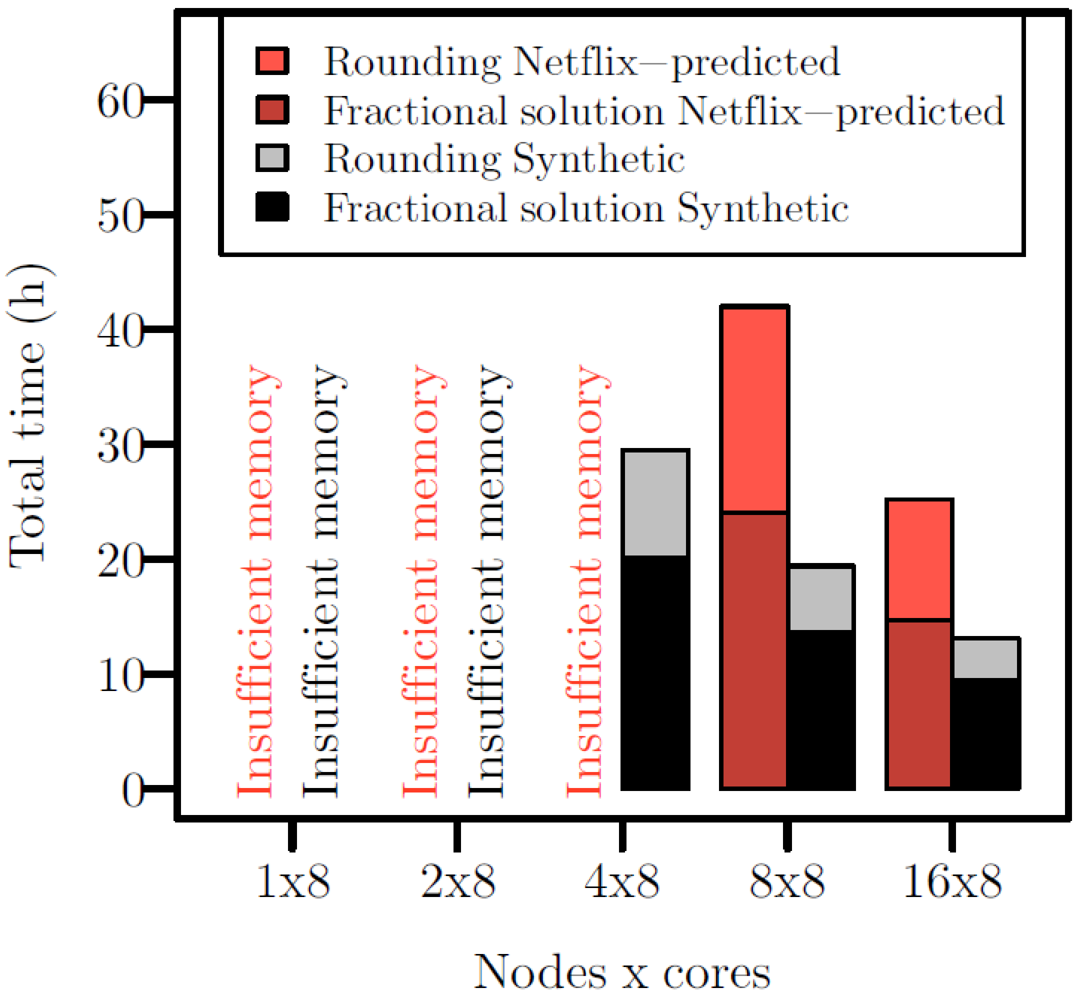
Distributed Rounding in Practice

- Edges already partitioned by MPCSolver
  → don't redistribute
- Empirical: most work done in first iteration
  → scales nicely, little communication
- Further saving in communication
  - Halving available compute nodes at each iteration
  - Even compute nodes keep their data
  - Odd compute nodes send data

Outline

1. Introduction
2. Generalized bipartite matching
3. Distributed mixed packing/covering LPs
4. Distributed rounding
5. **Experiments**
6. Conclusion

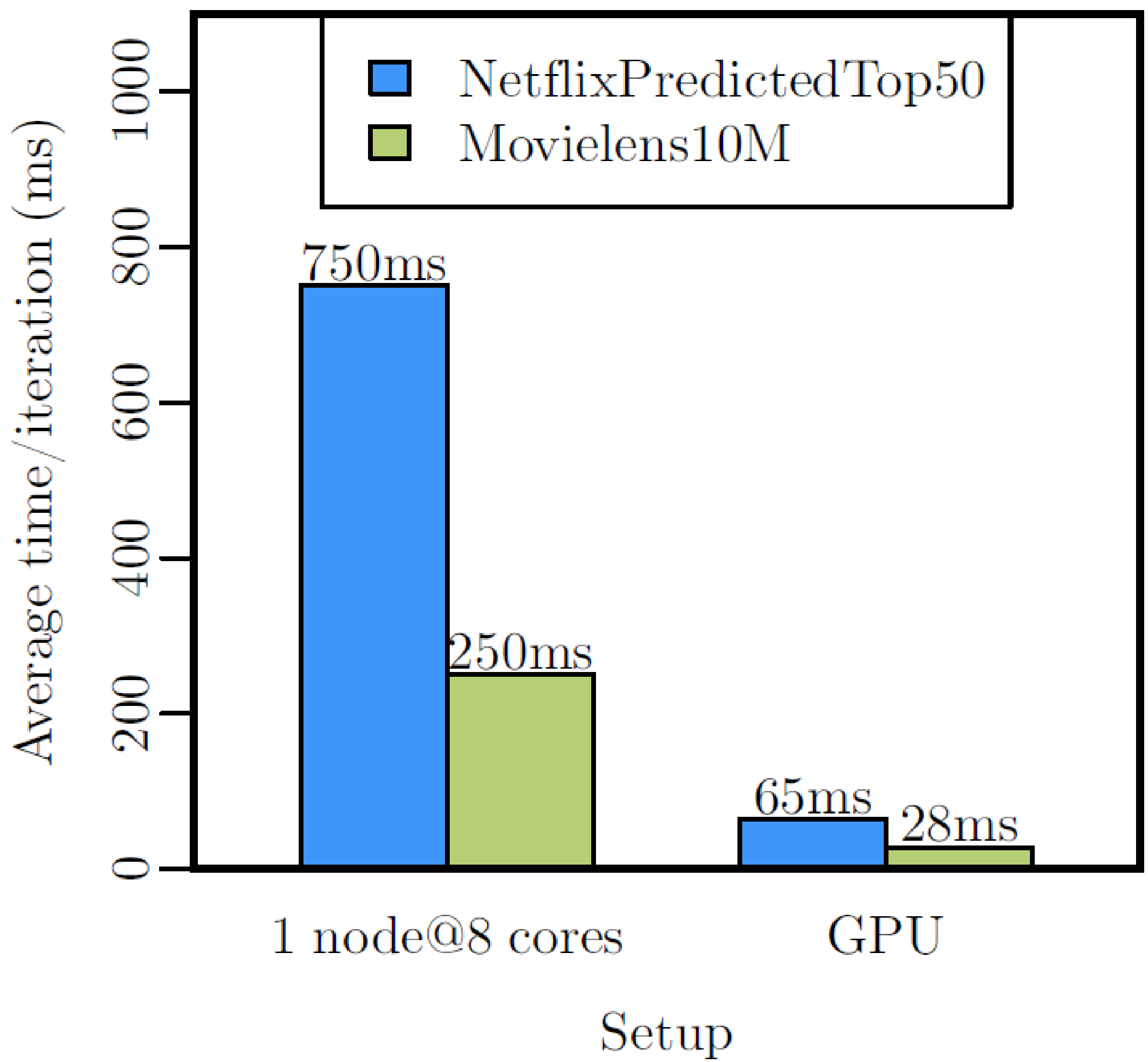
Scalability



Users	Items	Edges
490k	18k	3.2B
10M	1M	1B

(Gurobi ran out of memory on a high-memory server with 512GB RAM.)

MPCSolver on a GPU



Quality (feasibility, $\varepsilon = 0.05$)

	Netflix (pred.)	Synthetic
Users	490k	10M
Items	18k	1M
Edges	3.2B	1B
Sat. constraints	99.996%	99.993%
Max violation (fractional)	4.98%	4.99%
Max violation (integral)	2%	5%

Outline

1. Introduction
2. Generalized bipartite matching
3. Distributed mixed packing/covering LPs
4. Distributed rounding
5. Experiments
6. **Conclusion**

Summary

- Parallel approximation algorithms for
  - General mixed packing-covering linear programs
  - Rounding for generalized bipartite matching
  - Millions of vertices (users/items), billions of edges (preferences)
- Shared memory, MPI, MapReduce, GPU

A Distributed Algorithm for Large-Scale Generalized Matching
@ PVLDB, 2013