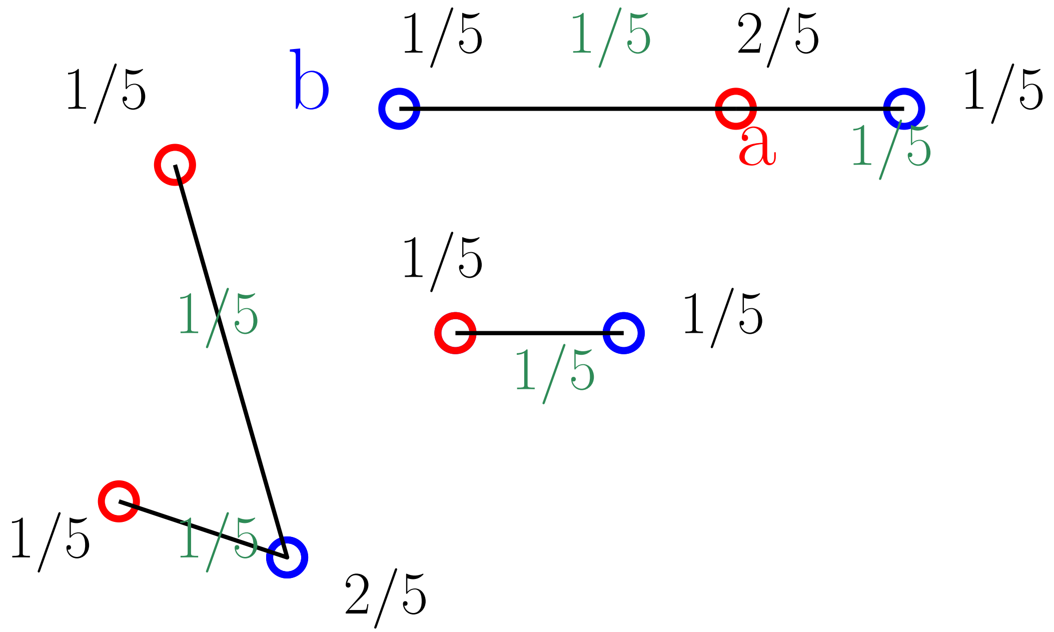
A Combinatorial Algorithm for Approximating the Optimal Transport in the Parallel and MPC Settings

Nathaniel Lahn, Sharath Raghvendra, Kaiyi Zhang

Introduction - The Optimal Transport Problem

- Distributions: μ and ν,
- Supports: A and B
- Mass on support: μ_a and ν_b
- $d(a, b)$: cost of transporting unit mass from $b \in B$ to $a \in A$.



Cost of transporting 1/5 mass from b to a is $d(a, b)/5$

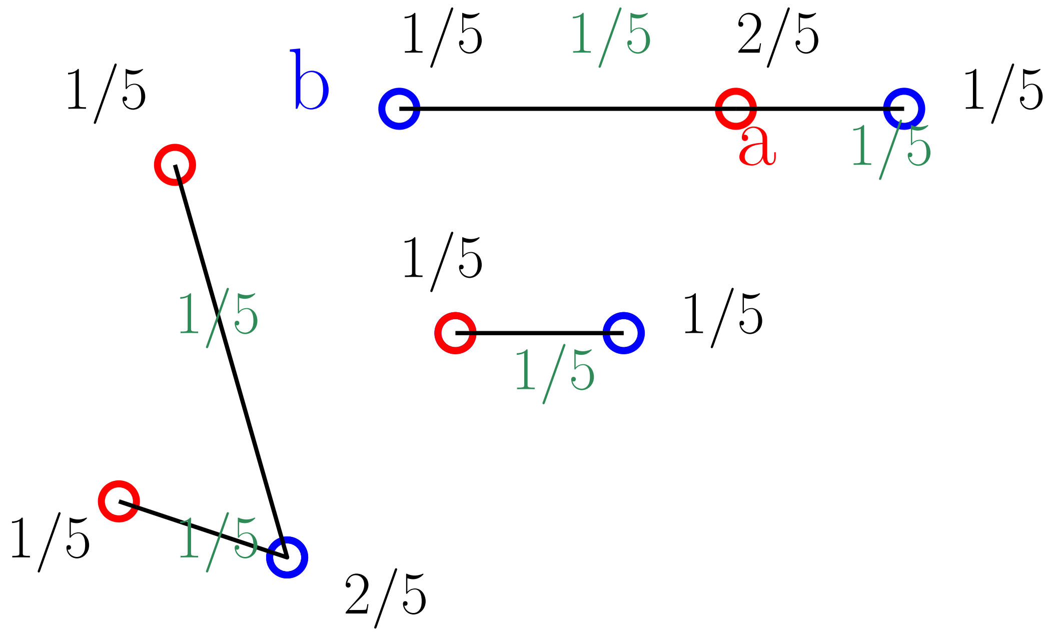
Introduction - The Optimal Transport Problem

- $\sigma(a, b)$ is the mass transported from b to a.
- Cost of the transport plan:

  $$\Sigma_{(b,a)\in B\times A}\, \sigma(a, b) d(a, b)$$

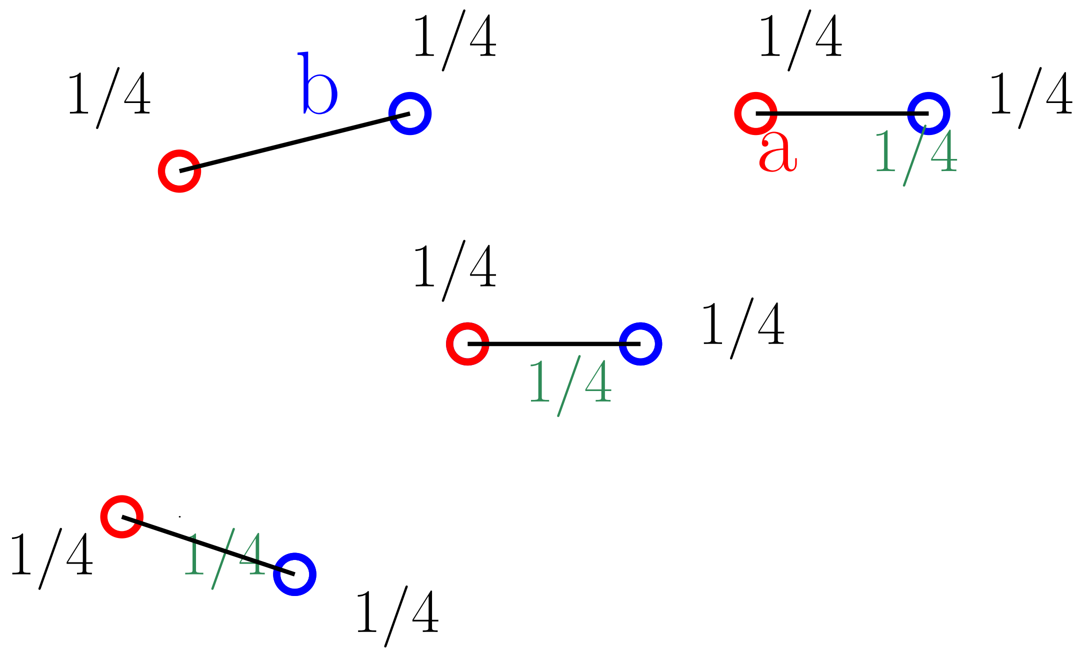
- Denote minimum-cost transport plan by σ^*, we define ε-approximate transport plan σ:

  $$c(\sigma) \le c(\sigma^*) + \varepsilon$$



Introduction - Assignment Problem

- Every Point in A and B has mass 1/n
- There are n vertex-disjoint edges with $\sigma(a,b) = 1/n$



Cost of transporting 1/4 mass from b to a is $d(a,b)/4$

Existing Work and Challenges

Sequential:

- Exact Algorithm (minimum cost flow): $O(n^3 \log n)$ (~20s, n=10k)
- ε-Approximation Algorithm: $O(n^2 \text{poly}\{1/_{\varepsilon}, \log n\})$
- Best: LMR algorithm, Lahn et al. (Neurips'19) $\tilde{O}(n^2/\varepsilon + n/\varepsilon^2)$, hard to parallelize

Parallel:

- Most successful method: Sinkhorn-Knopp, $\tilde{O}\left(\log(n)/\varepsilon^{O(1)}\right)$
  - Simple, easy implementation
  - Stability, convergence issues
- Best in theoretical: Jambulapati et al. $\tilde{O}(\log(n)/\varepsilon)$
  - Complex, hard implementation

Major Challenges: Design an efficient parallel combinatorial algorithm

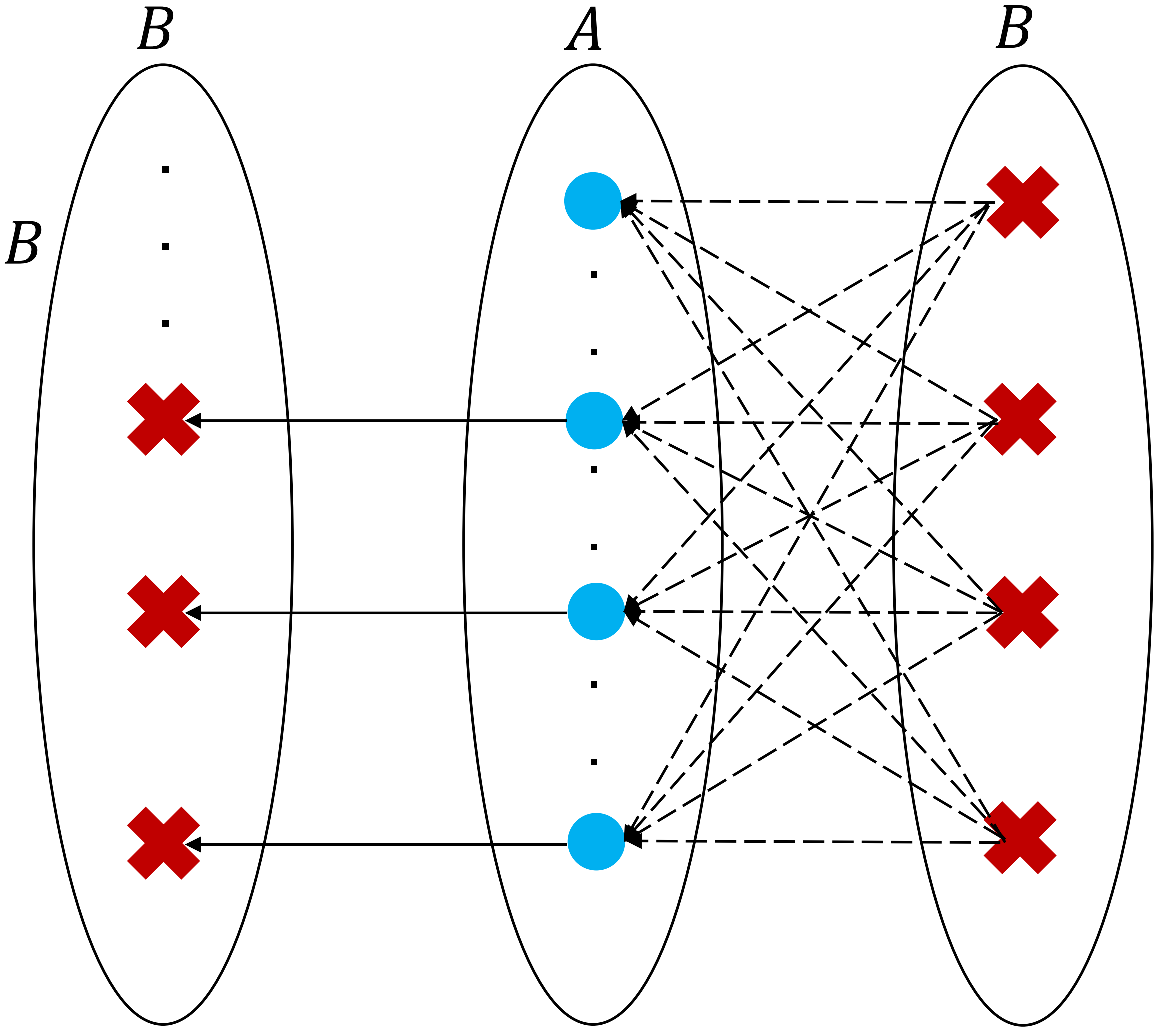
Our Approach

- Replace sequential search with push-relabel in LMR algorithm
- Benefits:
  - Easy to parallelize
  - Fully vectorized, pure matrix operations
- For each phase of our algorithm execute steps on the right
- We present the assignment problem, and OT can be reduced into assignment problem

Initialization: Extract admissible graph G'
Greedy step: Computes a maximal matching M' in the graph.
Matching Update
Dual Update

Our Approach - Initialization

Matched vertices of B

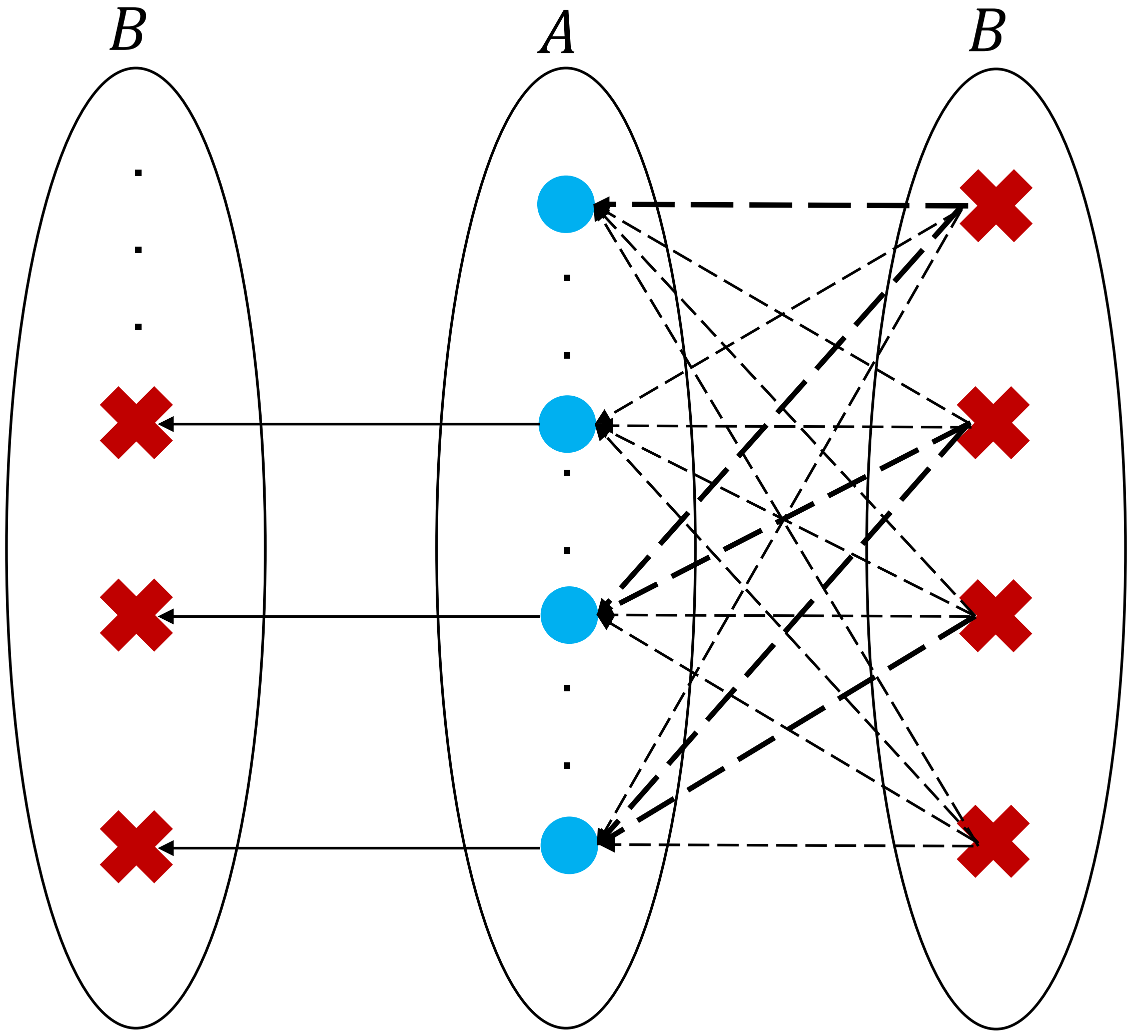


Free vertices of B

Initial Graph G

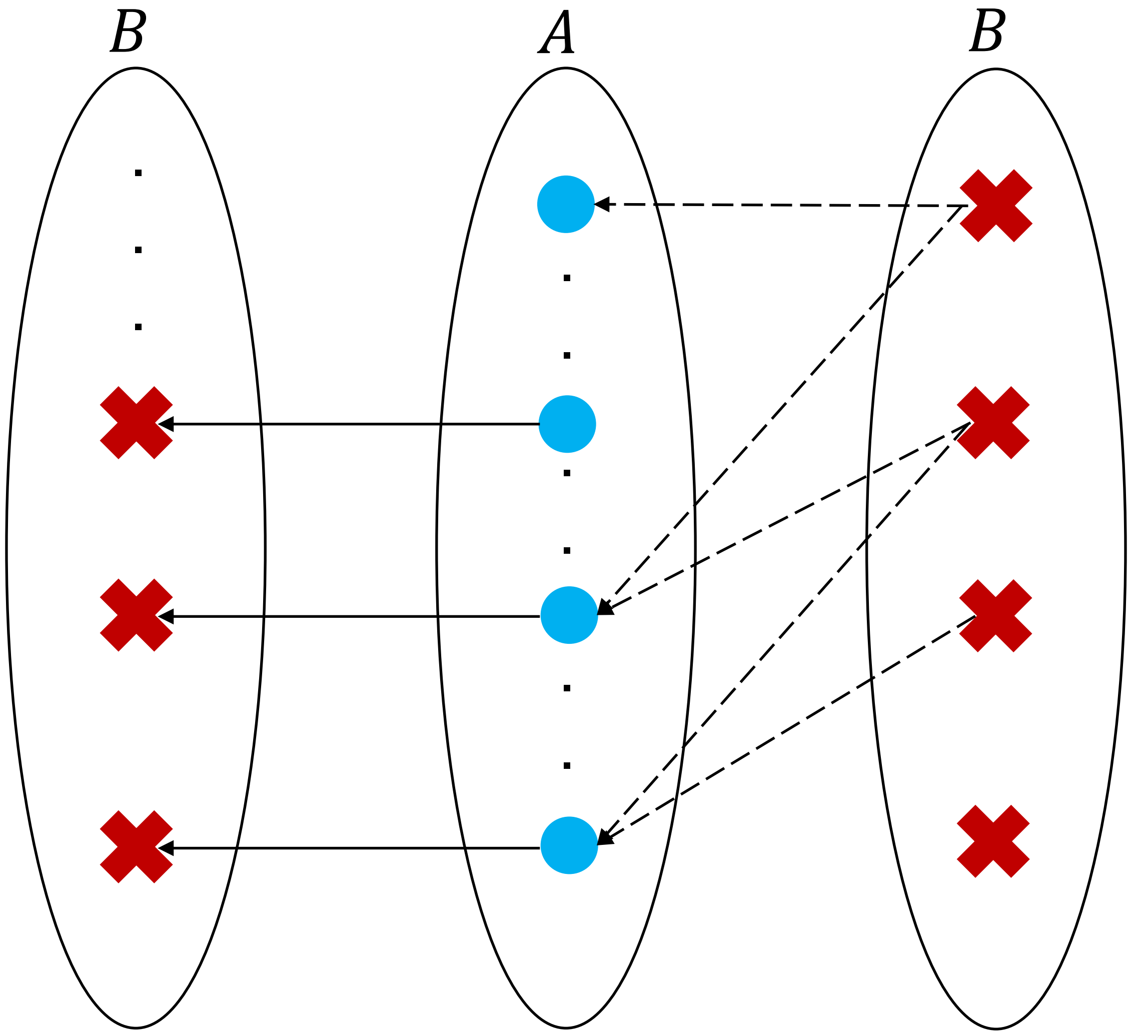
Our Approach - Initialization

Extract Zero Slack Edges From G'

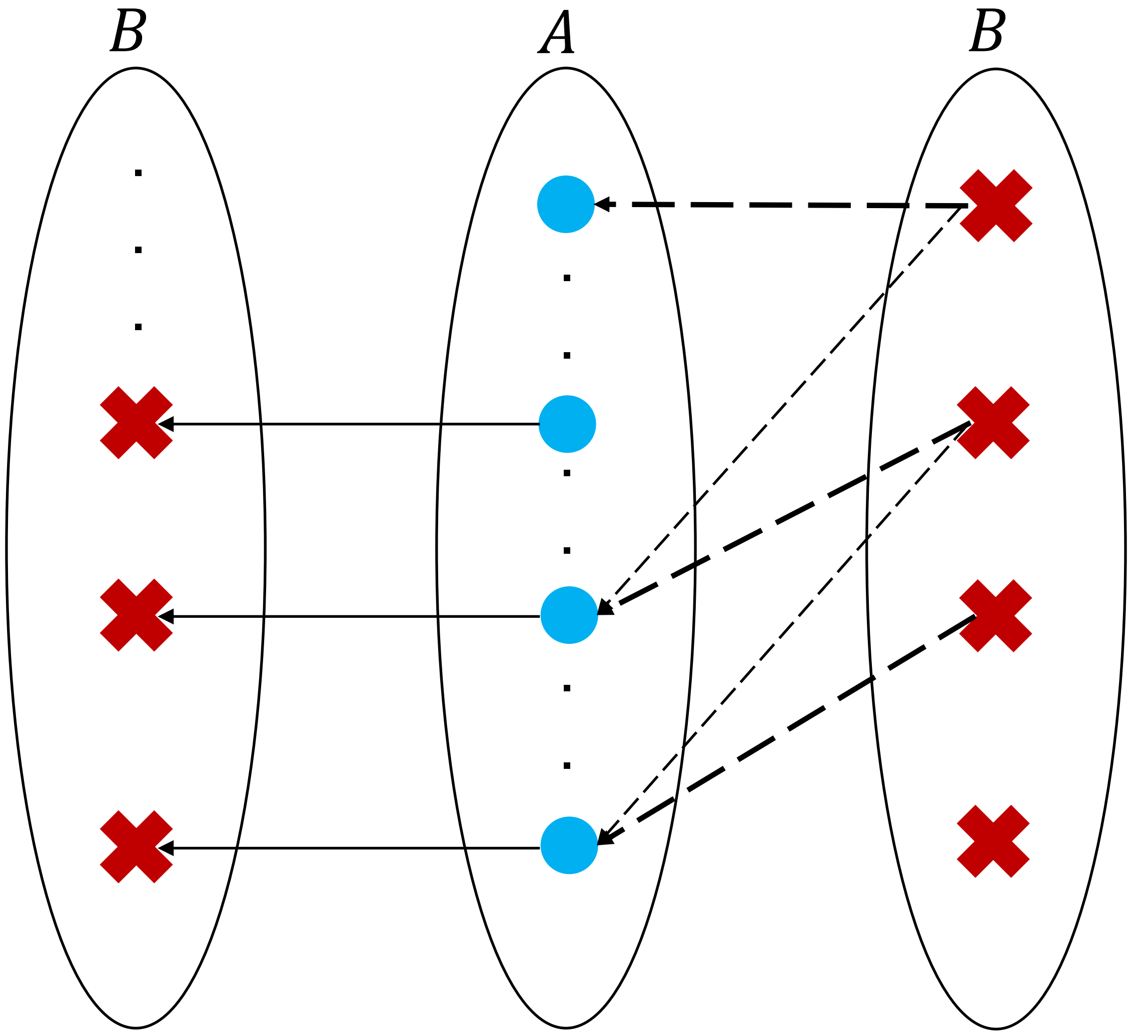


Our Approach - Initialization

Admissible Graph G'

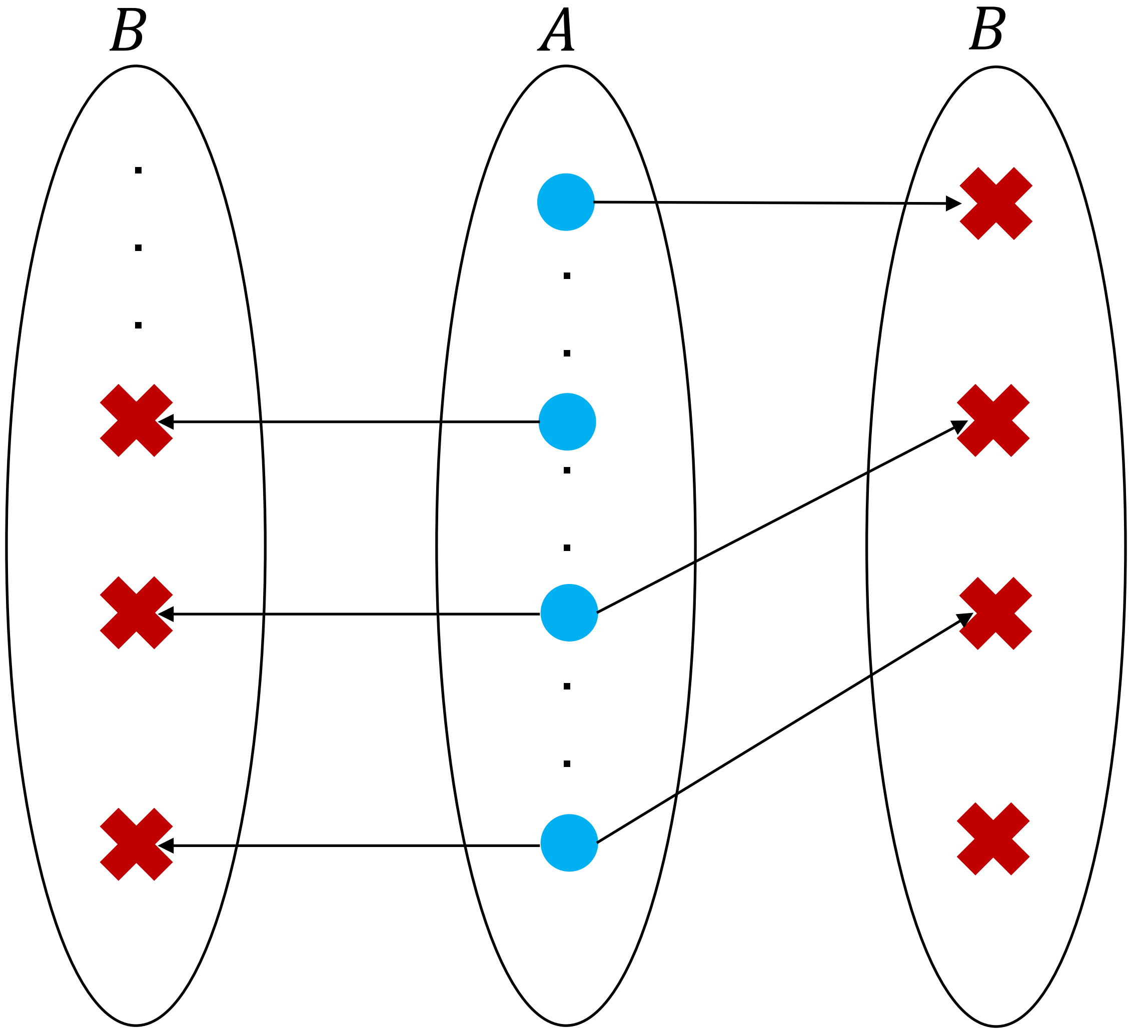


Our Approach - Greedy step



Computes a maximal matching M' in G'

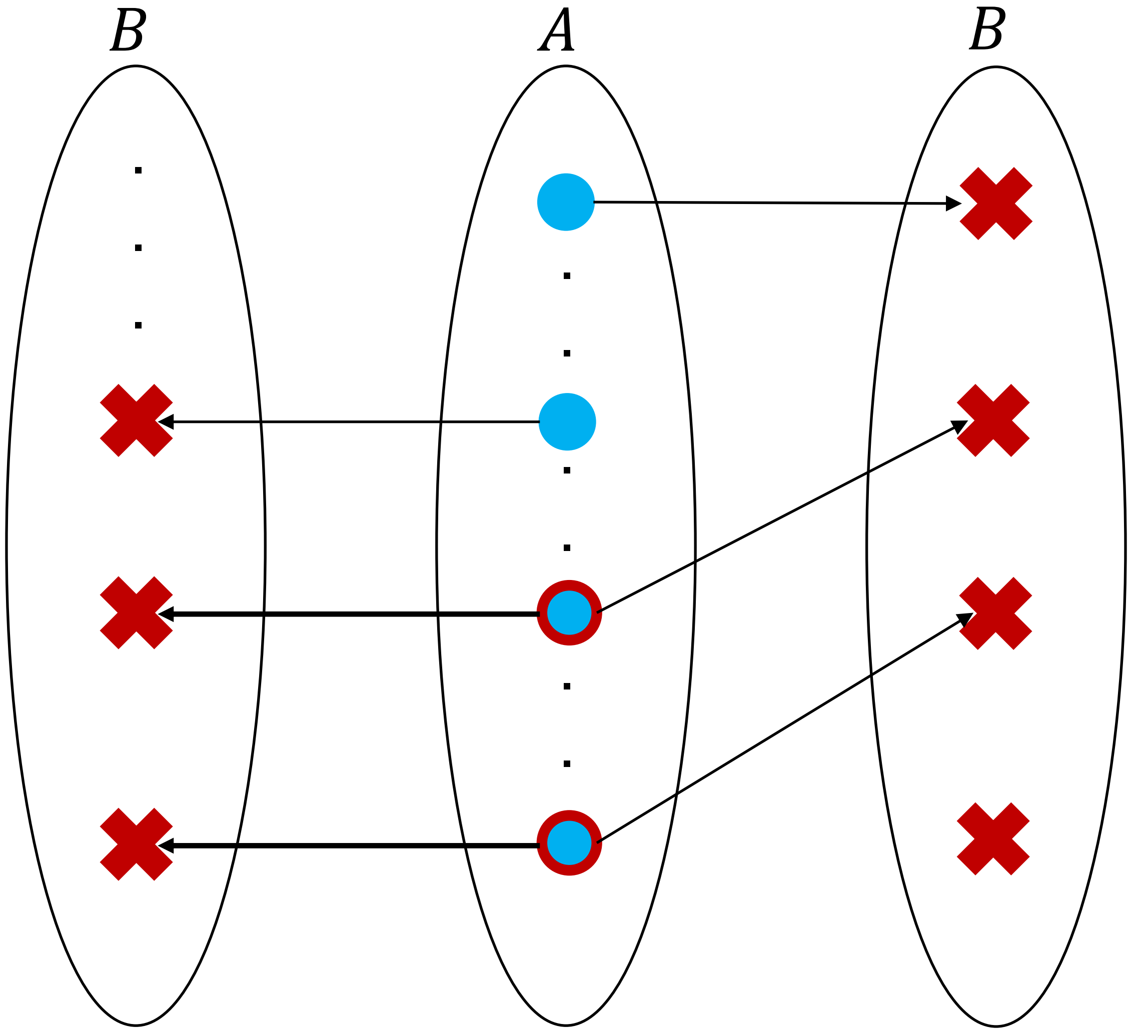
Our Approach - Greedy step



Computes a maximal matching M' in G'

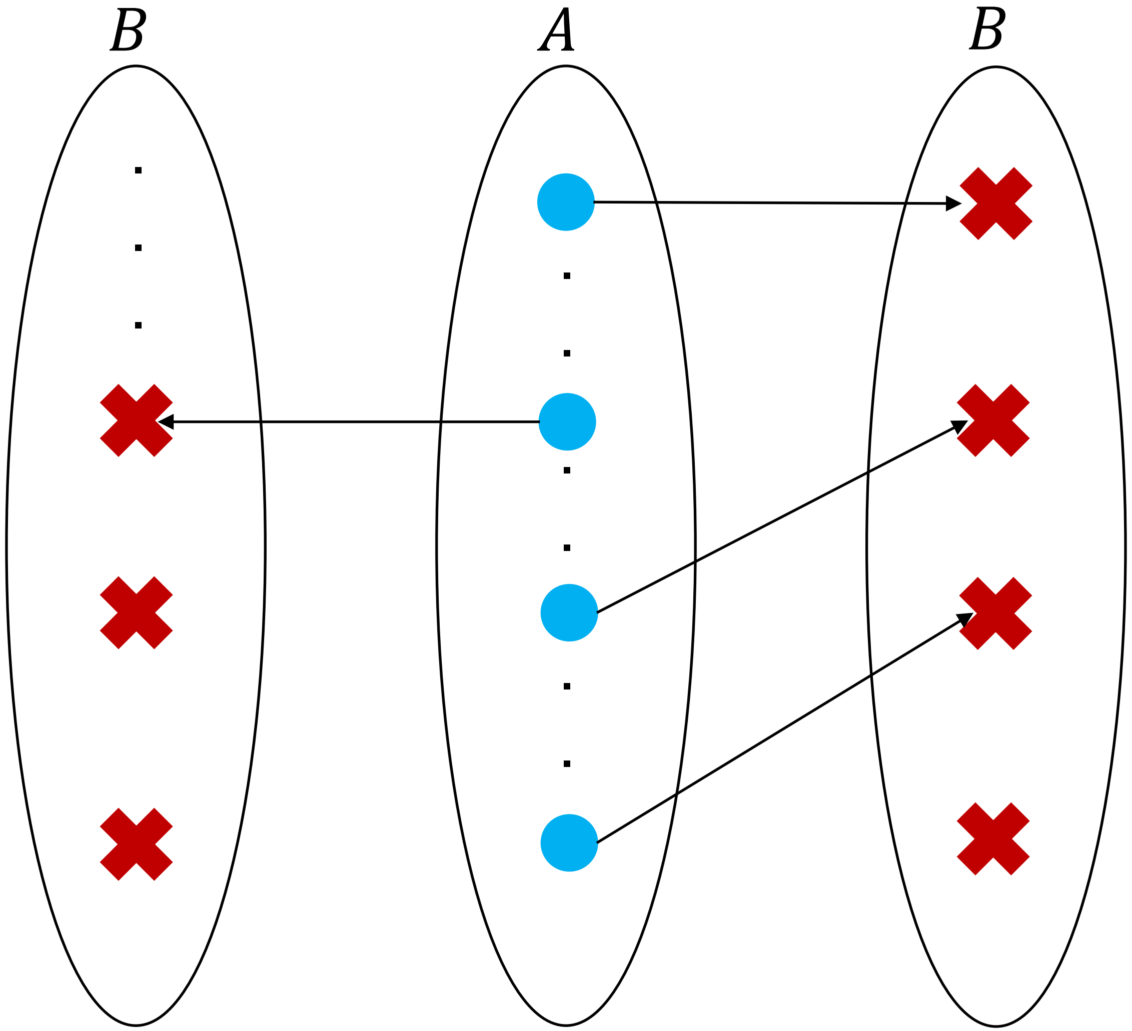
Our Approach - Matching Update

Matching
Update



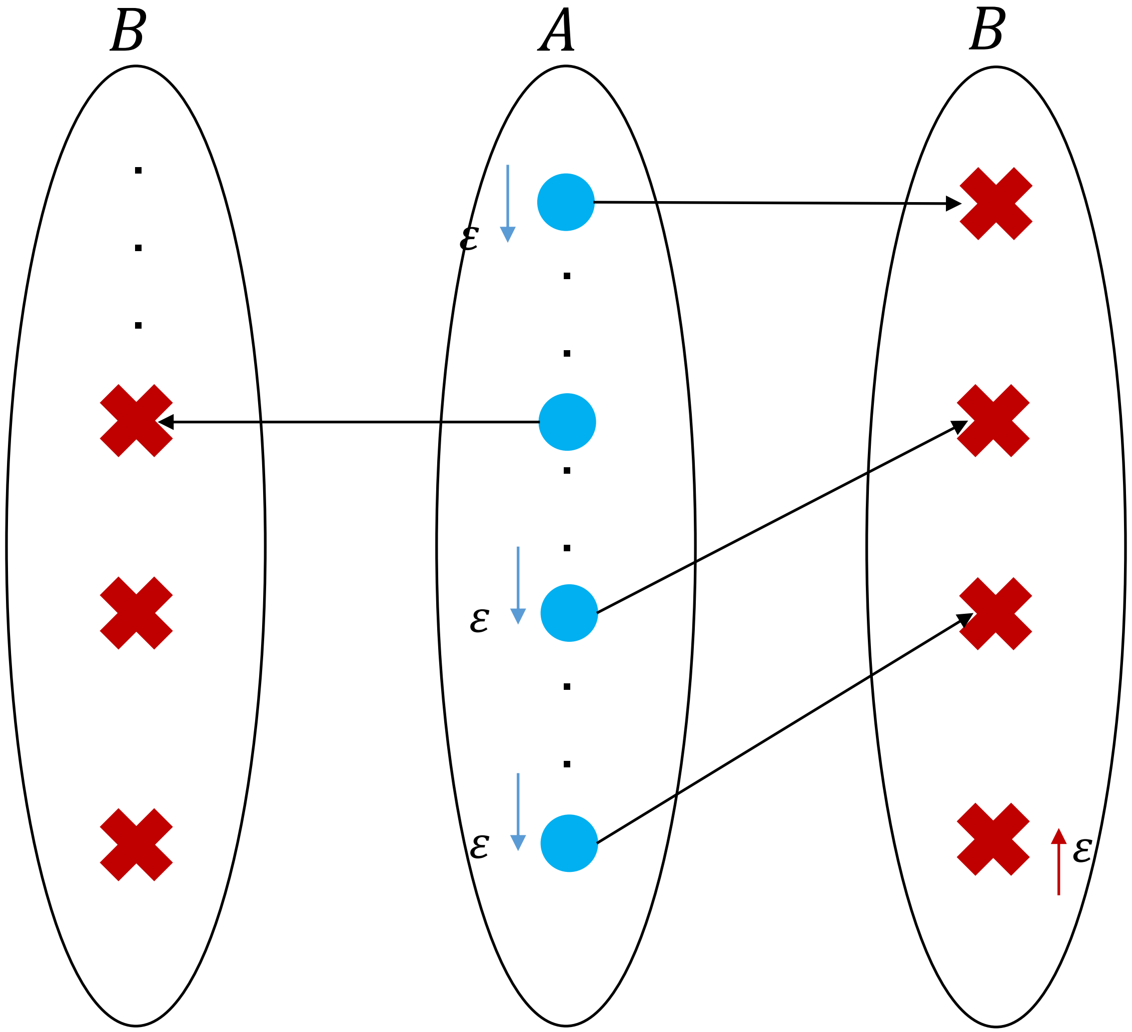
Our Approach - Matching Update

Matching Update



Our Approach - Dual Update

Dual Update



- Algorithm makes progress by increasing the dual weights
- We prove the upper bound of number of iterations based on the dual weights

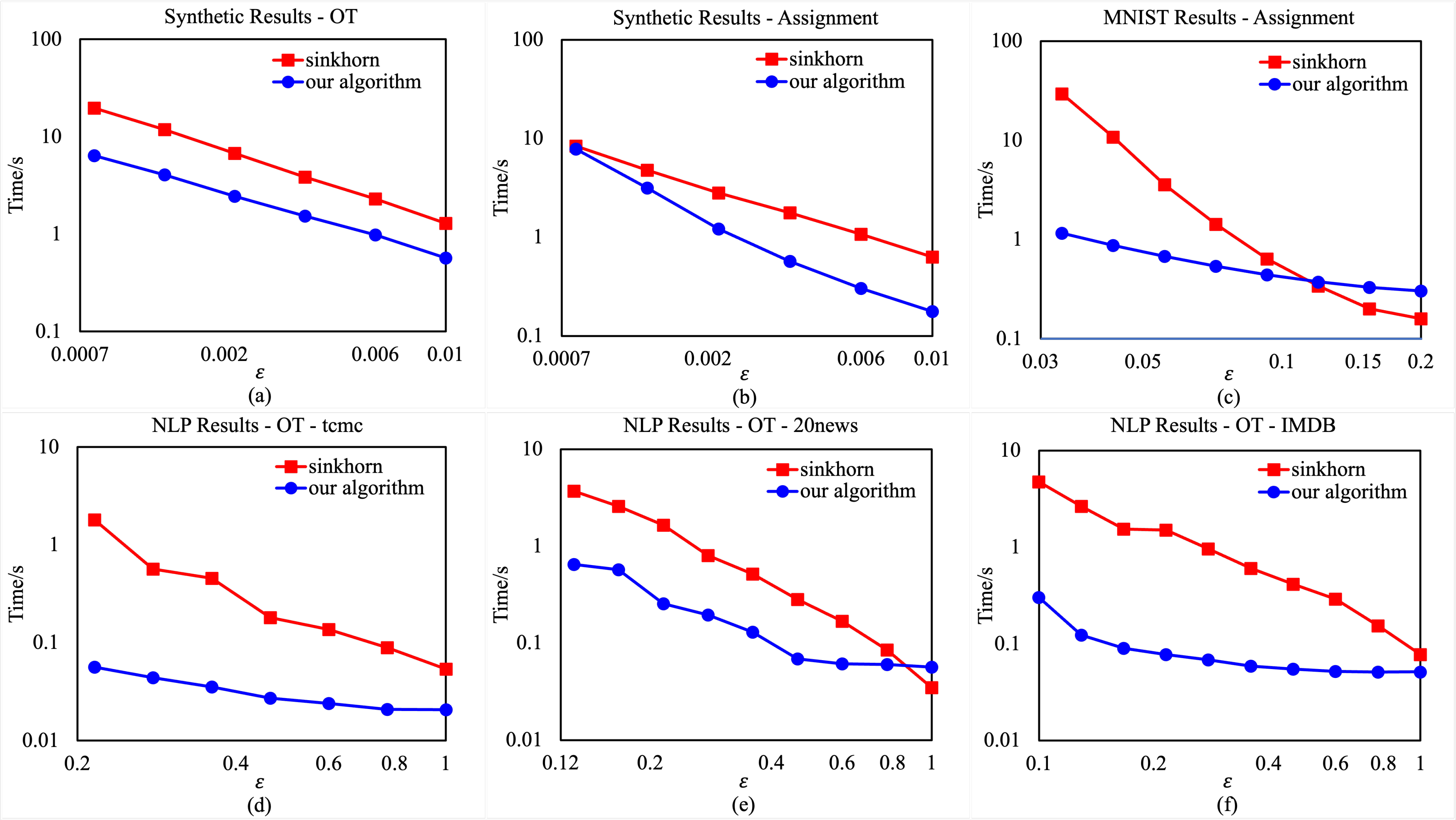
Our Approach - Running Time

- OT problem:
  - We use Israeli and Itai (1986) algorithm or MPC model to compute maximal matching M' in parallel
  - Time per phase:
    $\mathrm{O}(n^2)$ (sequential), $\mathrm{O}(\log n)$ (Israeli and Itai), $\mathrm{O}(\mathrm{loglog}\, n)$ (MPC)
  - Number of phases: $\mathrm{O}(1/\varepsilon^2)$
  - Total time:
    $\mathrm{O}(n^2/\varepsilon^2)$ (sequential), $\mathrm{O}(\log n/\varepsilon^2)$ (Israeli and Itai), $\mathrm{O}(\log\log n/\varepsilon^2)$ (MPC)
- Assignment problem:
  - $\mathrm{O}(n^2/\varepsilon)$ in sequential

Experiment Results

- Performance Comparison with Sinkhorn on GPU.
- Data Types: Synthetic data, real data (MNIST, NLP)
- Settings:
  - Assignment: synthetic data, MNIST images
  - OT: synthetic data, documents word embeddings.
- Results: our algorithm outperform Sinkhorn for most cases.

Experiment Results – Running Time



Experiment Results – Parallel Rounds