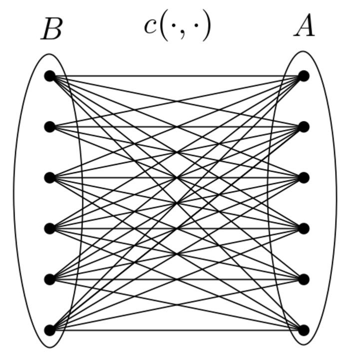
A Graph Theoretic Additive Approximation of Optimal Transport

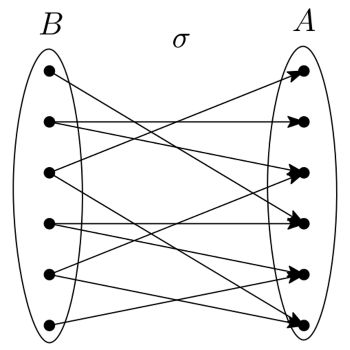
Nathaniel Lahn*, Deepika Mulchandani*†, Sharath Raghvendra*
*Virginia Tech; †Now at Walmart Labs

Optimal Transport

- A distance measure used in several machine learning applications
- Given:
  - A : Set of n vertices with 'demands' $d_a, a \in A$
  - B: Set of n vertices with 'supplies' s_b, $b \in B$
  - Costs $c(a,b)$ for every $(a,b) \in A \times B$.
  - $C = \max_{(a,b)\in A\times B} c(a,b)$
  - Assume that $\sum_{a\in A} d_a = \sum_{b \in B} s_b = 1$

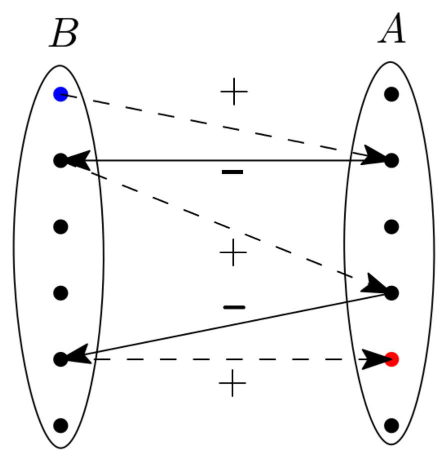


- A maximum transport plan $\sigma : B \times A \to \mathbb{R}$ has:
  - $\sum_{a\in A} \sigma(a,b) = s_b$ for every $b \in B$
  - $\sum_{b\in B} \sigma(a,b) = d_a$ for every $a \in A$
  - $c(\sigma) = \sum_{(a,b)\in A\times B} c(a,b)\sigma(a,b)$
  - Let σ^* be a minimum-cost maximum transport plan (optimal).



- Goal: Compute a maximum transport plan with: $c(\sigma) \le c(\sigma^*) + \delta$

Residual Graph



- **'Forward'** edges allow flow to be increased
- **`Backward'** edges allow flow to be decreased
- **Free vertices** have supply or demand remaining.
- **An augmenting path** P is any path that:
  - Starts and ends at a free vertex, and,
  - Alternates between forward and backward edges
- **Augmenting paths** can be used to **increase** the **flow** by:
  - Increasing $\sigma(a,b)$ for forward edges
  - Decreasing $\sigma(a,b)$ for backward edges

Dual Feasibility

- To ensure $c(\sigma)$ remains small, use **dual weights** $y(\cdot)$
- Dual weights are feasible if:
  - $y(u) + y(v) \le c(u,v) + 1$ for any forward edge (u,v)
  - $y(u) + y(v) \ge c(u,v)$ for any backward edge (u,v)
- The **slacks** $s(u,v)$ are:
  - $c(u,v) + 1 - y(u) - y(v)$ for any forward edge (u,v)
  - $y(u) + y(v) - c(u,v)$ for any backward edge (u,v)
- An augmenting path P is **admissible** if $s(P) = 0$.

Previous Results

- Exact algorithms are impractically slow for most applications.
- Instead focus has shifted towards approximation algorithms.
- Sinkhorn algorithm: $\tilde{O}(n^2(C/\delta)^2)$
  - Accounting for numerical precision requires additional C/δ factor increase.
- Multiple variants with empirical improvements
- Some results have strong $\tilde{O}(n^2C/\delta)$ theoretical bound, but are viable in practice.

- Most **prior work** uses the **Sinkhorn** projection technique
- Methods are algebraic
- Best theoretical bounds have log terms.
- Exponential matrix scaling, causes numerical precision issues.
- Many are highly practical (especially Sinkhorn based methods), but do not achieve best theoretic bounds.
- Easily parallelizable (matrix operations).

Our Result

- Provides a **new diameter-sensitive analysis** of a 30-year-old algorithm **[Gabow-Tarjan '89]** to obtain a running time of: $\boldsymbol{O(n^2C/\delta + n(C/\delta)^2)}$
  - Methods are graph-theoretic.
  - Matches best previous bounds, but with **no log terms.**
  - Does not suffer from numerical precision failures.
  - Scales well to very small values of δ.
  - Practical performance is competitive.
  - Not obvious how to parallelize.

Lower order term for $\frac{C}{\delta} = o(n)$.

Note, optimal can be computed in $\tilde{O}(n^{2.5})$ [Lee, Sidford].

Algorithm Workflow

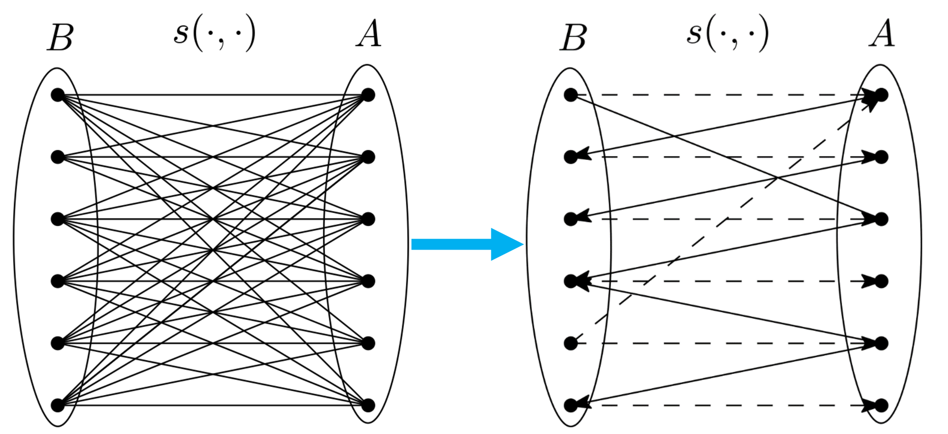
Scale Up & Round Input → **Apply One Scale of Gabow-Tarjan Algorithm** → **Scale Down Flow** → **Reduce Flow to Make Demands Feasible** → **Arbitrarily Assign Remaining Flow**

- Scale Up & Round Input: $\alpha \approx 2nC/\delta$; $\bar{s}_b \approx \lfloor \alpha \cdot s_b \rfloor$; $\bar{d}_a \approx \lceil \alpha \cdot d_a \rceil$; $\bar{c}(a,b) = \lfloor c(a,b)/\delta \rfloor$
- Apply One Scale of Gabow-Tarjan Algorithm: Computes transport plan $\bar{\sigma}$ w.r.t scaled instance with additive error $\approx \delta \cdot \sum \bar{s}_b$
- Scale Down Flow: $\sigma(a,b) = \bar{\sigma}(a,b)/\alpha$
- Reduce Flow to Make Demands Feasible: Some demands are exceeded due to rounding. An easy fix is to reduce flow slightly.
- Arbitrarily Assign Remaining Flow: Allocate remaining $\approx \delta/C$ supplies each at cost C

A Scale of the Gabow-Tarjan Algorithm

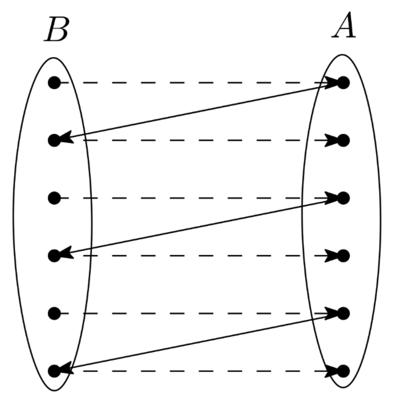
- While σ is not maximum:
  - Execute Dijkstra's Algorithm to find an admissible AP
  - Find multiple admissible augmenting paths using DFS
- Analysis:
  - $\approx 2C/\delta$ iterations (often less in practice)
  - Each iteration takes roughly n^2 time
  - Total augmenting path lengths: $\sim n\left(\frac{C}{\delta}\right)^2$ (often much less in practice)

Dijkstra's Algorithm



- Executing Dijkstra's algorithm takes roughly n^2 time
- Computes subgraph of admissible edges
- Admissible augmenting paths have small cost

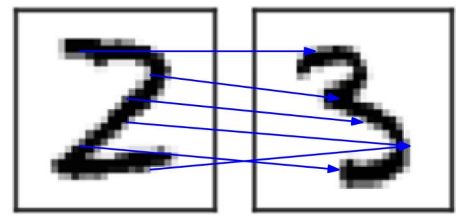
Partial DFS

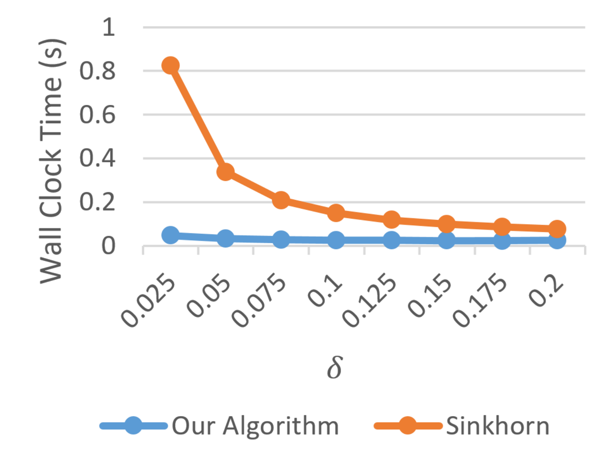


- Takes roughly n^2 time + total path length
- Each path has length at most $\sim C/\delta$
- Total supply $\sim nC/\delta$
- Total path length: $\sim n\left(\frac{C}{\delta}\right)^2$

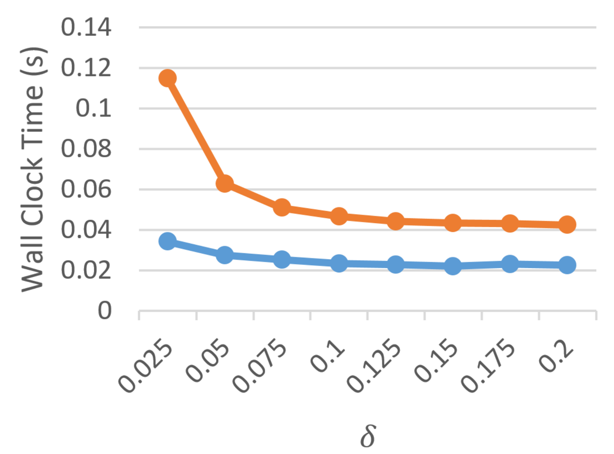
Experimental Results

- Implementation of our algorithm written in Java without any input specific optimizations.
- Compare running times with optimized MATLAB Sinkhorn code written using worst case scaling parameters for both algorithms.
- Input: MNIST Image data.
- Squared-Euclidean distance for costs
- Normalized so C = 1

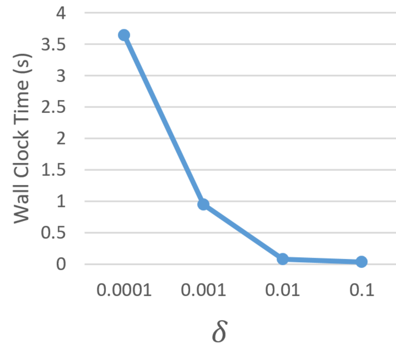




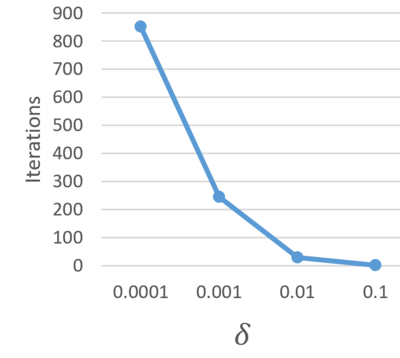
Sinkhorn receives: δ
Our algorithm receives: δ
Sinkhorn produces lower error than our algorithm.



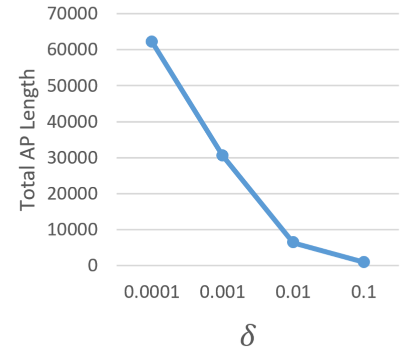
Sinkhorn receives: 5δ
Our algorithm receives: δ
Sinkhorn produces higher error than our algorithm



Our algorithm does not suffer from numerical precision issues for small δ.



Iterations are somewhat less than that suggested by worst case analysis.



Work associated with augmenting path lengths is negligible in practice.

References

Cuturi, Marco. "Sinkhorn distances: Lightspeed computation of optimal transport." In *Advances in Neural Information Processing Systems*. 2013.

Gabow, H. N., & Tarjan, R. E. "Faster scaling algorithms for network problems." *SIAM Journal on Computing*, 1989.

Lahn, Nathaniel, Deepika Mulchandani, and Sharath Raghvendra. "A graph theoretic additive approximation of optimal transport." In *Advances in Neural Information Processing Systems*. 2019.

Lee, Y. T., & Sidford, A. "Path finding methods for linear programming: Solving linear programs in O (vrank) iterations and faster algorithms for maximum flow." In Symposium on Foundations of Computer Science. 2014.