**A Graph Theoretic Additive Approximation of Optimal Transport**

**Nathaniel Lahe**, Deepika Mulchandani†, Sharath Raghvendra*

---

**Optimal Transport**

- A distance measure used in several machine learning applications.
- Given:
  - \( A \) : Set of vertices with demands \( d_a = \delta \)
  - \( B \) : Set of vertices with supplies \( c_b = \delta \)
  - Costs \( c(a,b) \) for every \( (a,b) \)
- \( C = \max c(a,b) \)
- \( \sum c(a,b) = 1 \)
- Set \( \delta = \min c(a,b) \)
- \( \sum c(\delta) = \delta \)

**Residual Graph**

- \( G \) is a residual graph.
- \( \delta \) is the minimum flow.
- \( x^* \) is the optimal flow.
- \( G^\delta \) is a residual graph.

**Dual Feasibility**

- To ensure \( (\xi,\eta) \) remains small, use dual weights \( \eta(x) \).

- Dual weights are feasible if:
  - \( \eta(x) \geq 0 \) for any forward edge \( x \)
  - \( \eta(x) \leq 0 \) for any backward edge \( x \)
  - \( \sum \eta(x) = 0 \)

**Dijkstra’s Algorithm**

- \( \sigma \) is not maximum:
  - Execute Dijkstra’s Algorithm to find an admissible AP.
  - Find multiple admissible augmenting paths using DFS.

**Algorithm Workflow**

- Scale Up & Round
  - Apply One Scale of Gabow-Tarjan Algorithm
  - Reduce Flow to Make Demands Remaining
  - Arbitrarily Assign Remaining Flow

**A Scale of the Gabow-Tarjan Algorithm**

- While \( \sigma \) is not maximum:
  - Execute Dijkstra’s Algorithm to find an admissible AP.
  - Find multiple admissible augmenting paths using DFS.

**Previous Results**

- Exact algorithms are exponentially slow for most applications.
- Instead focus has shifted towards approximative algorithms.

**Our Result**

- Provides a new diameter-sensitive analysis of a 30-year-old algorithm (Gabow-Tarjan ‘89) to obtain a running time of \( O(n^2\log(1/\epsilon)/\delta + (n^2/\delta)^{3/2}) \).

**Exponential matrix scaling, causes numerical precision issues.**

**References**