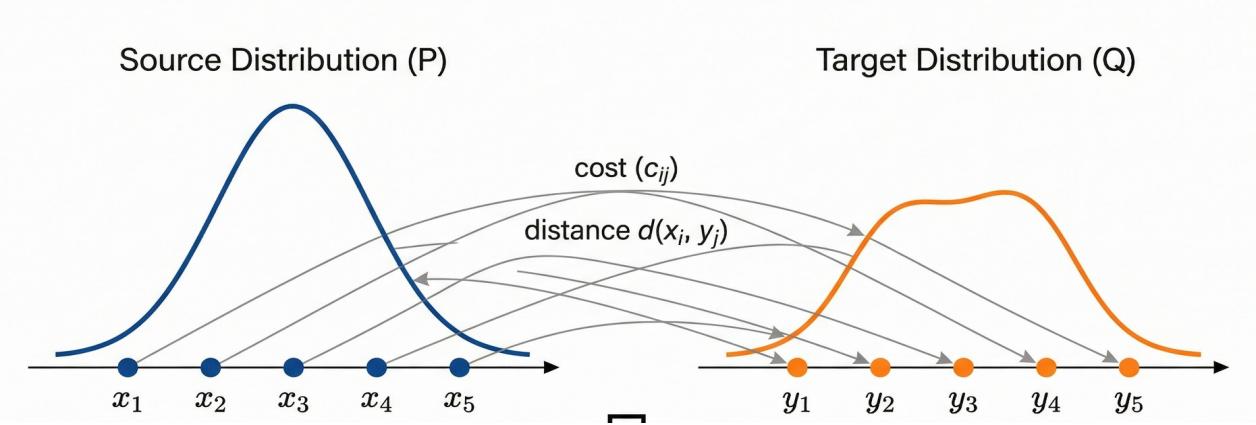
Efficient Algorithms for Incremental Metric Bipartite Matching

Ritesh Seth (IIIT Delhi) · Mrinal Garg (IIT Bombay) · Sujoy Bhore (IIT Bombay) · Sharath Raghvendra (NC State University) · Syamantak Das (IIIT Delhi)

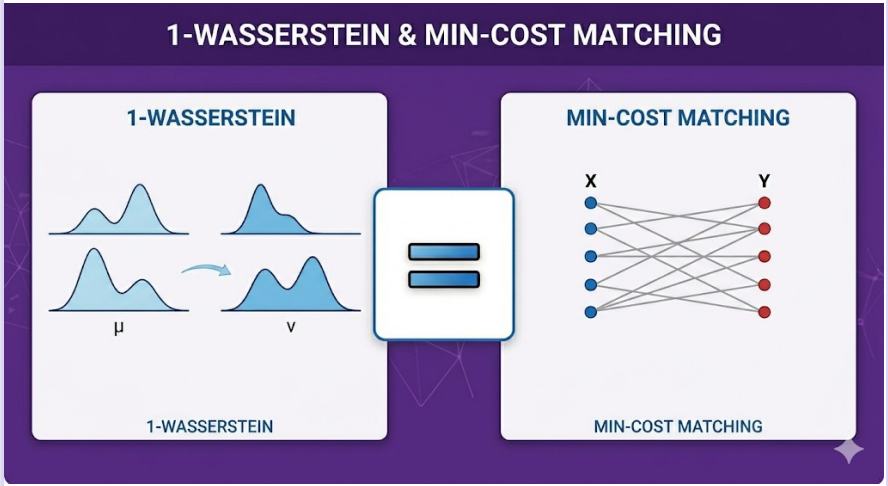
ICLR 2026

The 1-Wasserstein Distance

$W_1(P, Q) =$ $\inf|_{\pi \in \prod(P,Q)} \int_{X \times X} d(x, y) d\pi(x, y)$ where $\prod$(P, Q) denotes the set of couplings of P and Q in metric (X, d)



W_1 = Minimum Cost Matching



Discrete sets of points

Each of n points has mass 1/n

Special case of W_1 → Minimum Cost Perfect Bipartite Matching

Match → Pay Distance → Minimize Cost

Distance = ANY metric (Euclidean · Graph · Embedding)

The 1-Wasserstein Distance Is Everywhere in ML

Generative Models

WGAN, WAE use W_1 as training loss between real & generated distributions

Domain Adaptation

Align source & target distributions by minimizing W_1 between feature spaces

Fairness Auditing

Measure distributional gap between demographic groups continuously

Distributional Drift

Detect when model's input distribution shifts at inference time

Medical Imaging

Track longitudinal changes in patient data (MRI scans) over time

Time Series

Earth Mover's Distance for comparing evolving temporal datasets

In all these settings — data arrives as streams. W_1 must be recomputed continuously.

The Incremental Setting

The Real-World Setup

1 Samples $r_1, r_2, r_3 \ldots$ arrive one by one in metric space

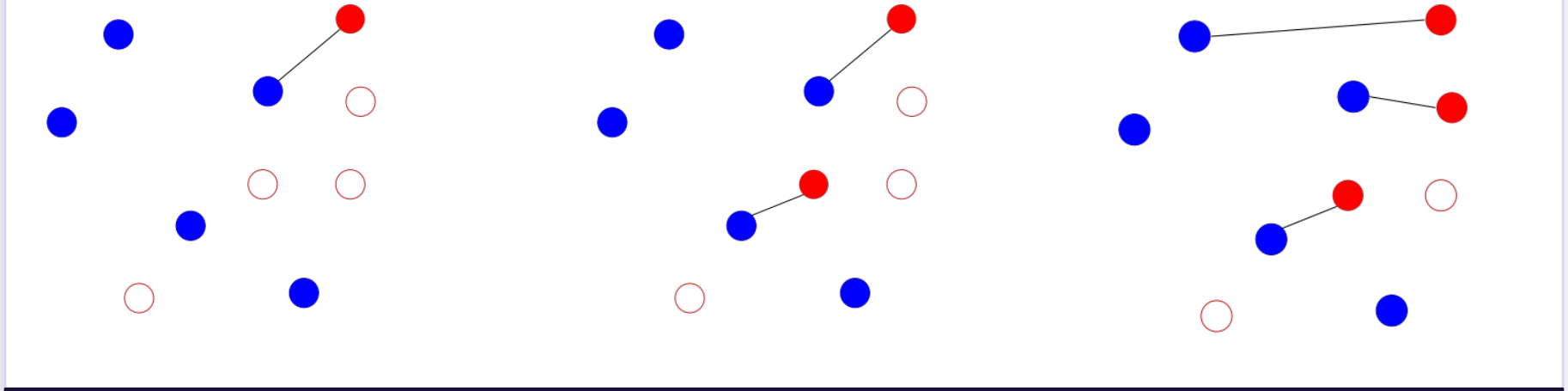
e.g. model predictions, sensor readings

2 Fixed reference set S of size n

e.g. real dataset, atlas, server fleet

3 After each r_t: assign request to server

or compute W_1 estimate



Can we maintain a near-optimal matching in ANY metric space, with fast updates after each new point?

The Streaming Bottleneck

Why Naive Approaches Fail

✗ Existing algorithm — $O(n^2)$ per update

Hungarian algorithm — optimal cost but far too slow for streaming

✗ Prior on incremental matching — Euclidean only

Goranci et al. (ICML'25) — o(n)-amortized runtime with constant cost. Breaks for graphs, manifolds, or any non-Euclidean metric

✗ Nothing better known — Even with constant approx

No algorithm better than $O(n^2)$-update time with constant approx

Can we maintain a near-optimal matching in ANY metric space, with fast updates after each new point?

What We Built

The first incremental algorithm: maintains an constant approximate matching after every new sample — fast, and in any metric space.

Theorem. Deterministic algorithm maintaining an $O(1/\delta^{0.631})$-approximate matching under insertions, with amortized update time:

$$\tilde{O}(n^{(1+\delta)}) \text{ per insertion}$$

ANY Metric Space

Graphs · Manifolds · String edit · Protein embeddings — not just Euclidean

Sublinear Updates

No recomputing from scratch. Each new sample handled in $\tilde{O}(n^{(1+\delta)})$ time

Constant Approx.

Guaranteed cost within $O(1/\delta^{0.631})$ of offline optimal at every step

GPU-Ready

Batch push-relabel runs on GPU. Handles high-throughput arrival streams

How It Works — The Key Idea

Core insight: Don't work in the original metric. Build a hierarchy of O(log 1/δ) progressively scaled metrics and apply a primal-dual approach to maintain a matching combining all levels — using push-relabel instead of augmenting paths.

Metric Hierarchy

Rescale & round distances into O(log 1/δ). Makes analysis tractable; distances are discretized integers at each level.

Prior: *Static (Agarwal & Sharathkumar '14): builds levels once*

Ours: We maintain ALL levels simultaneously under insertions

Push-Relabel (not augmenting paths)

When request r_i arrives: apply a Primal-Dual based approach to match the request in these scaled metric spaces.

Prior: *Augmenting paths: scan $O(n^2)$ edges per request*

Ours: Push-relabel: amortized $O(n^{(1+\delta)})$ via dual accounting

GPU Parallelism

Batch arrivals processed in parallel. Admissible graph construction + push + relabel all parallelize naturally onto GPU (adapted from Israeli and Itai, '86).

Prior: *Sequential: one request blocks the next*

Ours: Batch of 200 requests processed concurrently on GPU

Why Should You Care?

If you compute 1-Wasserstein distances on streaming data —

You've been paying $O(n^2)$ per step. Our algorithm maintains W_1 continuously with $\tilde{O}(n^{(1+\delta)})$ amortized updates.

→ *Drop-in replacement for batch W_1 solvers in any streaming pipeline.*

If your data lives in a non-Euclidean metric space —

Graphs, manifolds, edit distances, road networks. Existing incremental methods don't apply. Ours does.

→ *First algorithm with provable guarantees for incremental matching in general metrics.*

If you need high-throughput real-time assignment —

GPU-parallelized batch push-relabel. Tested on MNIST, NYC Taxi, Beijing Road Network.

→ *github.com/ritesh-777/Incremental-Metric-Bipartite-Matching-Algorithm*