Fully Dynamic Euclidean Bi-Chromatic Matching in Sublinear Update Time

Gramoz Goranci
University of Vienna

Peter Kiss
University of Vienna

Neel Patel
University of Southern California

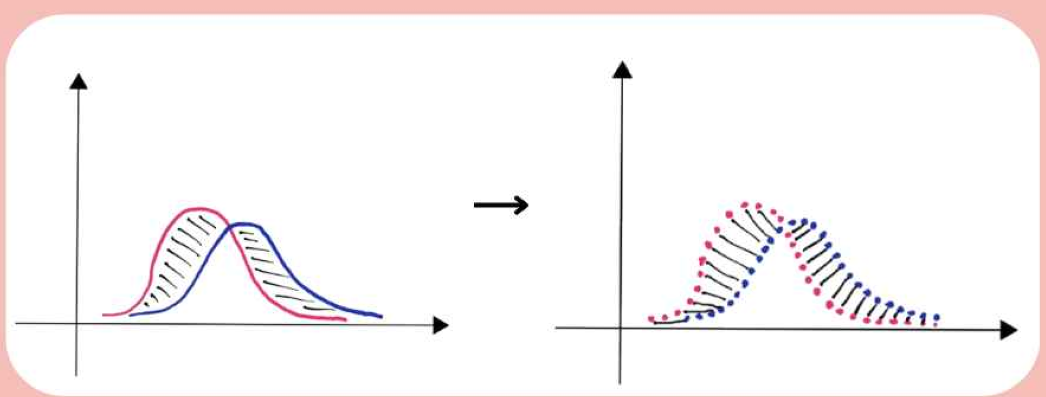
Martin P. Seybold
University of Vienna

Eva Szilagyi
UniVie Doctoral School Computer Science DoCS, University of Vienna

David Zheng
University of Illinois at Urbana-Champaign

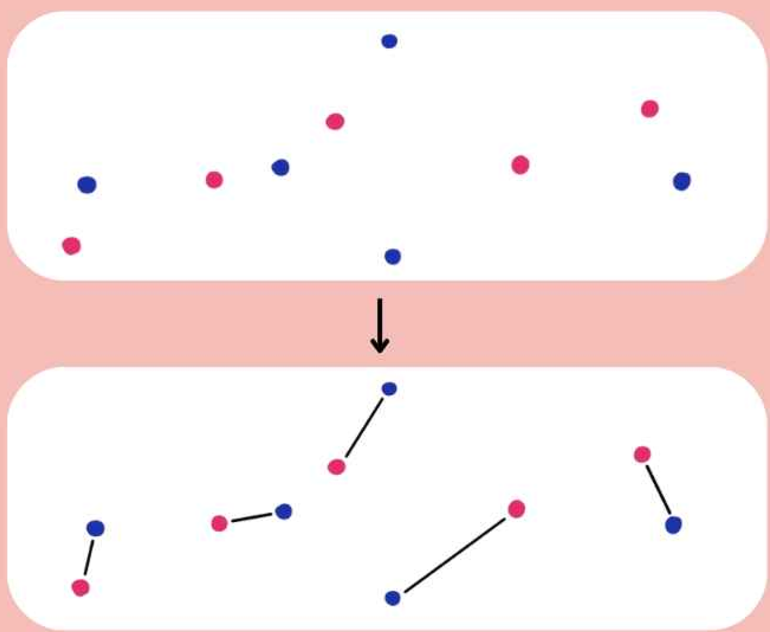
1-Wasserstein Distance

- *1-Wasserstein distance* - notion of distance between two probability distributions
- Various applications in machine learning (e.g. in model selection, model evaluation)
- Discrete 1-Wasserstein distance → Euclidean Bi-Chromatic Matching



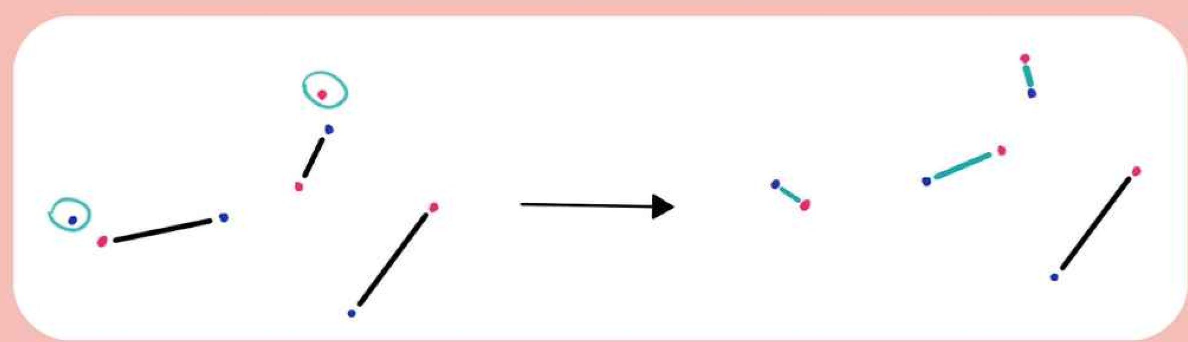
Euclidean Matching

- **Given:** set of n red and n blue points in plane
- **Goal:** Find a perfect matching between red and blue points of *minimum* weight
- *Weight* of a matching - sum of distances of all pairs in the matching



Dynamic Setting

- **Operations:** insertions and deletions of pairs of points
- **Goal:** Design a fast algorithm that maintains (a constant-approx. of) the optimal Euclidean matching



Static Algorithm

Idea

- Points in cells on lower levels should be matched first
- Remaining points should be matched at the parent node if possible

Input: set A of n red points, set B of n blue points
Output: a perfect matching on set $A \cup B$

1. Construct a p-tree T w.r.t. set $A \cup B$
2. Use the following bottom-up approach:
   2.1 Points at leaves of T are matched via *Hungarian algorithm*
   2.2 Remaining points at internal nodes are first moved, then matched via *Transportation algorithm*
3. Convert implicit matching to explicit matching at each node

Results

Theorem 1

There is a dynamic algorithm that supports updates in $O(n^{\varepsilon})$ amortized update time and maintains an $O(1/\varepsilon)$-approximate Euclidean matching. Further, reporting the change in the solution (recourse) takes $O(n^{\varepsilon})$ time.

Theorem 2

No dynamic algorithm can maintain a better than 2-approximation to Euclidean matching in sublinear update time.

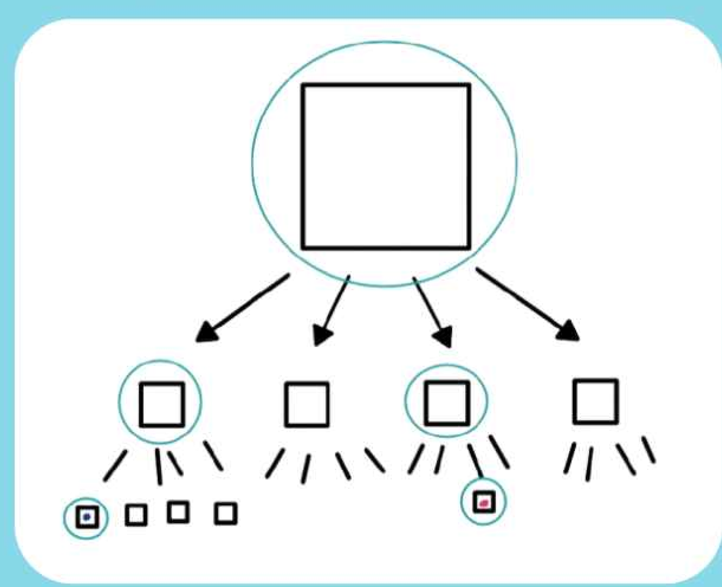
Dynamic Algorithm

Idea

- Matching only changes in *affected nodes* - nodes containing the newly inserted/deleted points
- The excess set at each affected node changes by at most one

Input: red point a and blue point b
Output: a perfect matching on set $A \cup B$, solution recourse

1. Update p-tree
2. In a bottom-up fashion, do the following in each affected node:
   2.1 Update the excess set
   2.2. If the excess set decreses, invoke the *Augment Matching procedure*



Hungarian algorithm

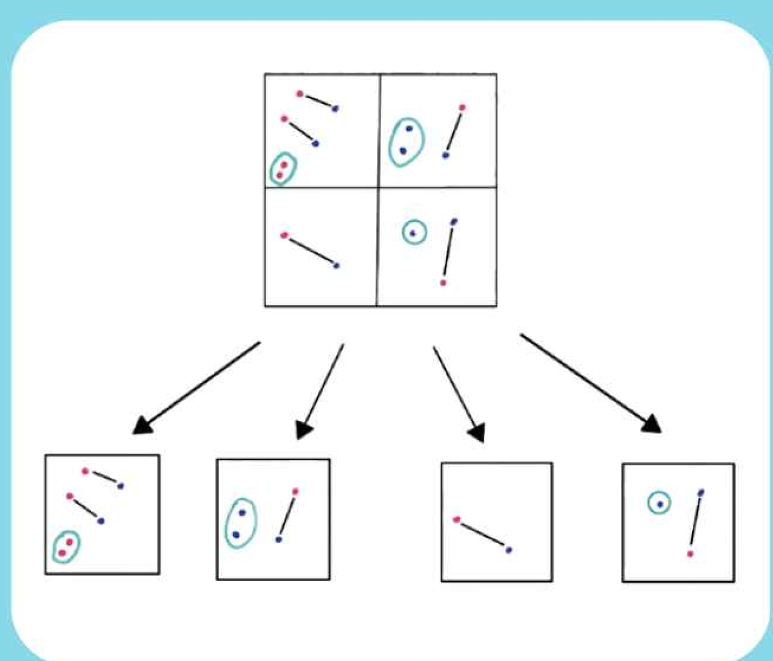
- We use the following direct corollary of the Hungairan algorithm:

Theorem

Given a set of n red and n blue points, the optimal Euclidean matching can be computed in time $O(n^3)$.

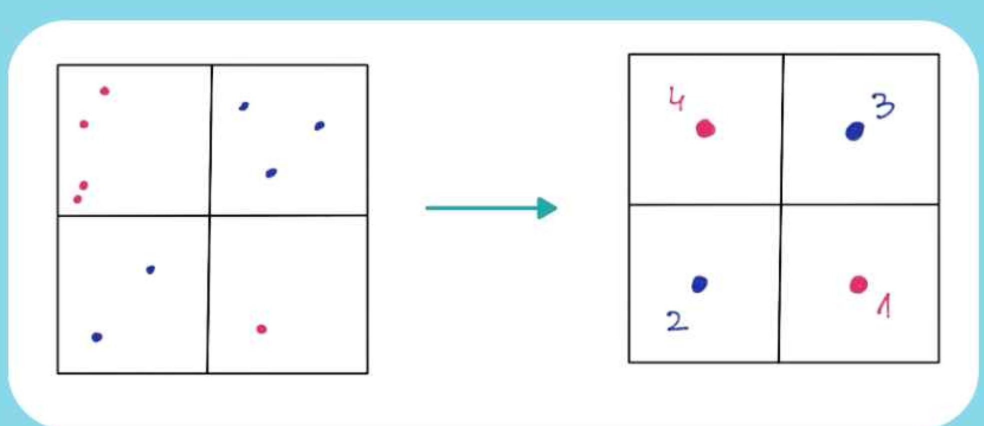
p-trees

- A tree-like structure consisting of dividing cells into p smaller subcells,each leaf contains $\leq p^2$ many points in a cell
- Can be maintained in time $O(p^2)$
- **Idea:** subset of points inside the same cell should be matched



Transportation Algorithm

- If there are too many points in a cell, Hungarian algorithm is too slow
- Instead, move all points to the middle of their subcell → solve problem on this *moved sub-instance* to obtain the *implicit matching*
- This problem is the *Euclidean transportation problem*
- Moving the points *doesn't* distort the matching a lot

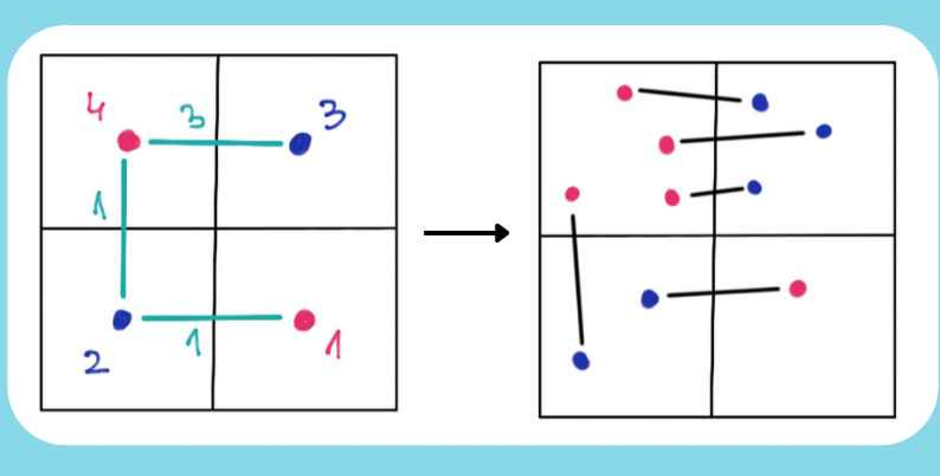


Theorem

The Euclidean transportation problem on k points with maximum demand N can be solved in time $O(k^{2.5} \log k \log N)$.

Implicit vs Explicit Matching

- Implicit matchings easier to store - they are of size $O(p^{2\lambda})$, while explicit matchings can be of size $\Omega(n)$
- Corresponding explicit matchingis only by a constant factor worse than the optimal Euclidean matching on a given set



Augment Matching Procedure

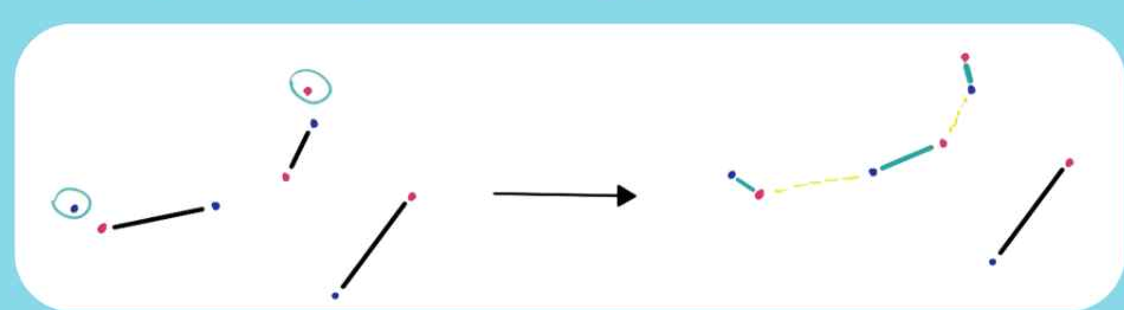
Idea

- Maintain the implicit matching at each affected node
- If the size of the excess set decreases, we can find one unmatched red point a and one blue point b in a cell

Input: Implicit matching γ on set X, red point a and blue point b
Output: optimal implicit matching on set $X \cup \{a, b\}$

1. Construct an auxiliary graph w.r.t. γ and a and b
2. Find augmenting path Π - shortest altenating path starting at a and ending at b
3. Augment γ along path Π

Claim (informal): *Augment matching* procedure returns an optimal implicit matching on set $X \cup \{a, b\}$ in $O(p^3)$ time.
Claim (informal): The change in the corresponding explicit matching can be computed in time $O(p^2)$.



References

[1] David S Atkinson and Pravin M. Vaidya. 1995. Using geometry to solve the transportation problem in the plane. Algorithmica 13, 5 (1995), 442–461.
[2] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. Naval research logistics quarterly 2, 1-2 (1955), 83–97.
[3] Pankaj K. Agarwal and Kasturi R. Varadarajan. 2004. A near-linear constant-factor approximation for euclidean bipartite matching?. In Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004, Jack Snoeyink and Jean-Daniel Boissonnat (Eds.). ACM, 247–252.