New Applications of Nearest-Neighbor Chains

Nil Mamano

Committee: David Eppstein, Michael Goodrich, Sandy Irani

Collaborators: Alon Efrat, David Eppstein, Daniel Frishberg, Michael Goodrich, Stephen Kouborov, Doruk Korkmaz, Pedro Matias, Valentin Polishchuk

University of California, Irvine

Motivating question

When is global information necessary vs when is local information sufficient?

distributed algorithms, streaming algorithms, ...

This talk: greedy algorithms

Outline

A property of (some) greedy algorithms:

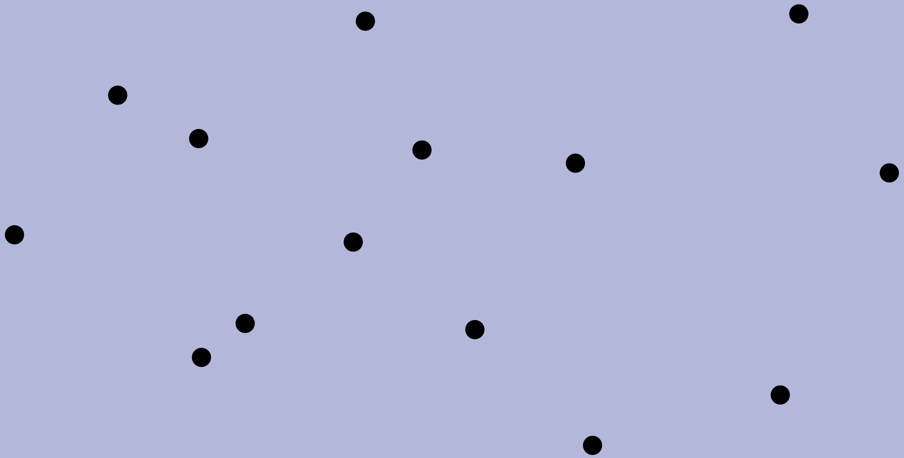
Global—Local Equivalence

An algorithm to exploit it:

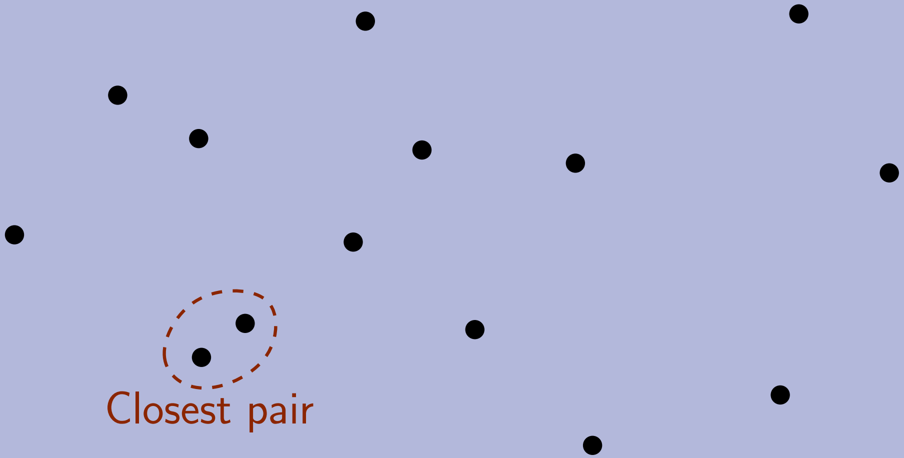
Nearest-Neighbor Chain Algorithm

Applications

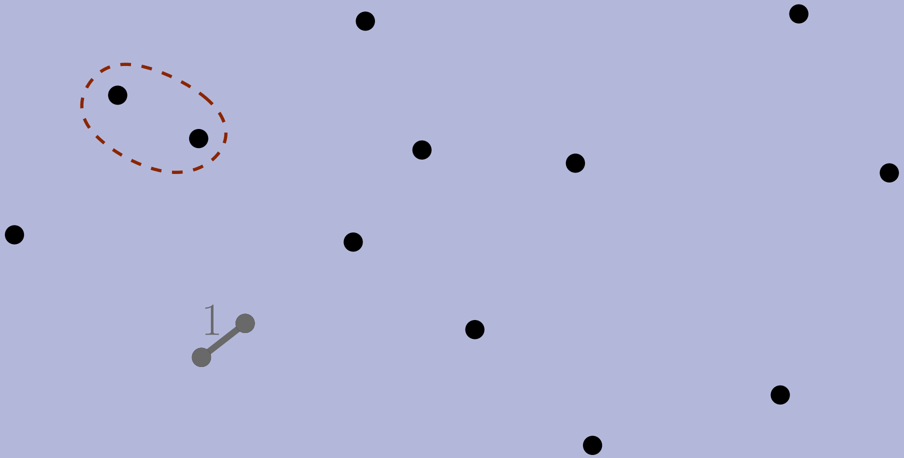
Greedy for Matching



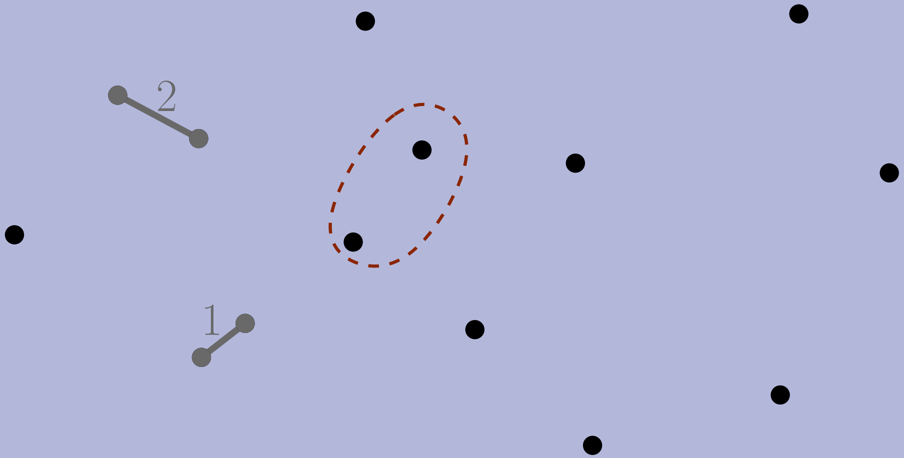
Greedy for Matching



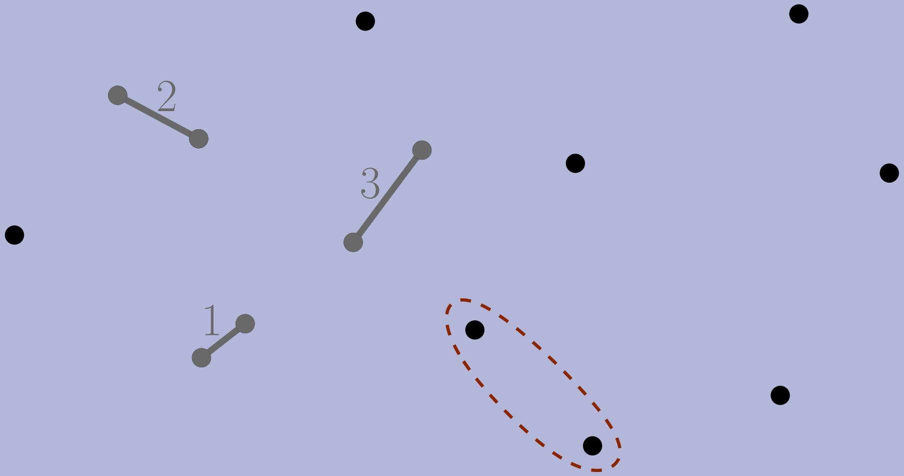
Greedy for Matching



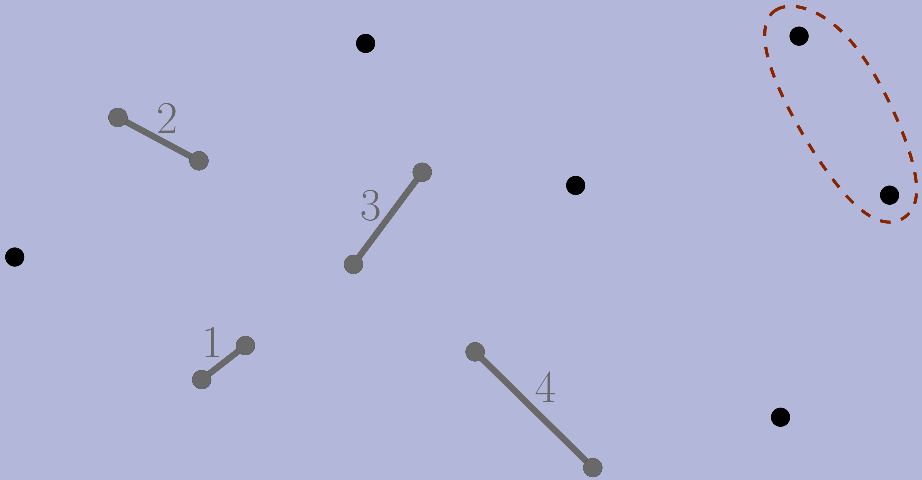
Greedy for Matching



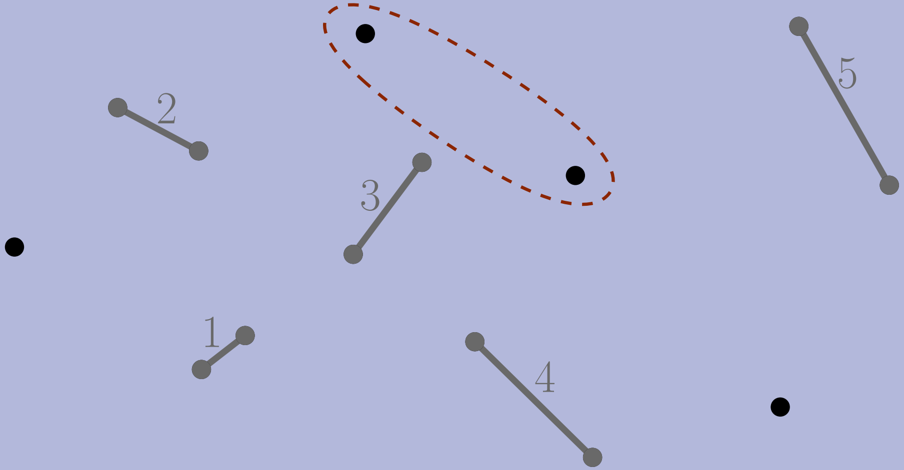
Greedy for Matching



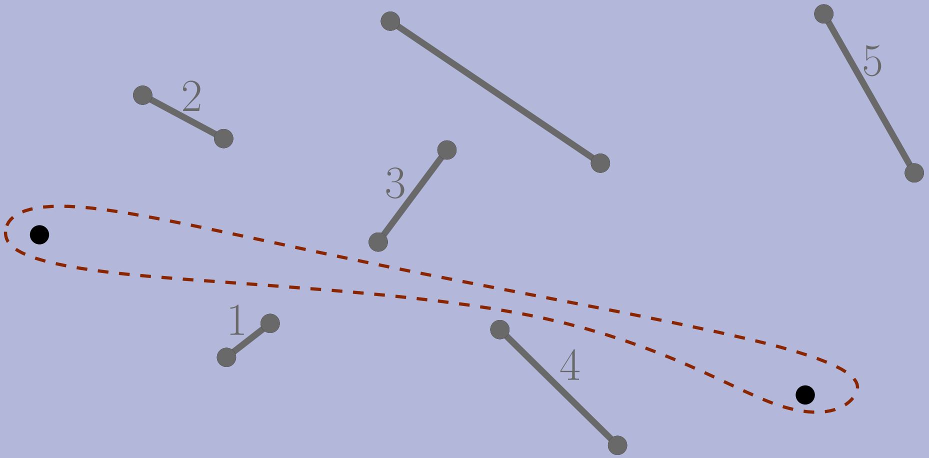
Greedy for Matching



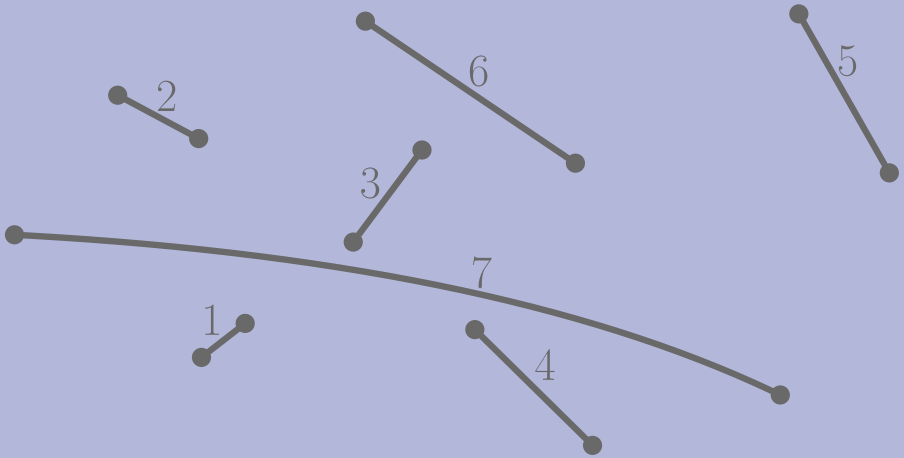
Greedy for Matching



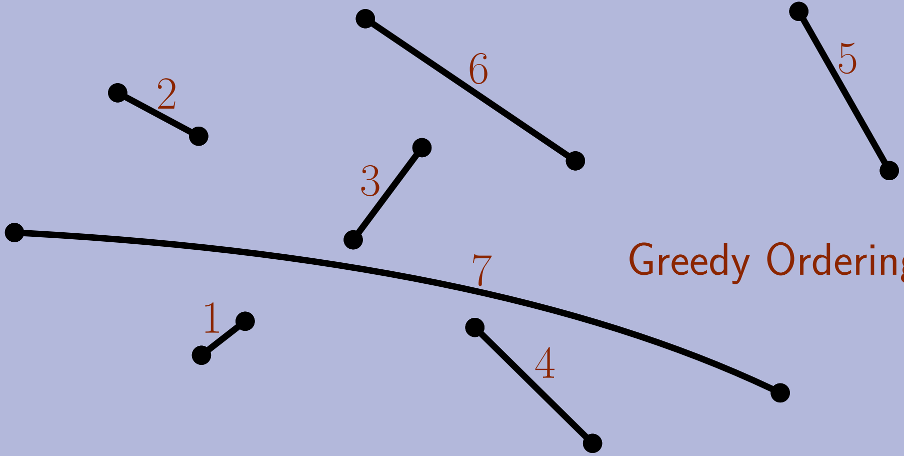
Greedy for Matching



Greedy for Matching

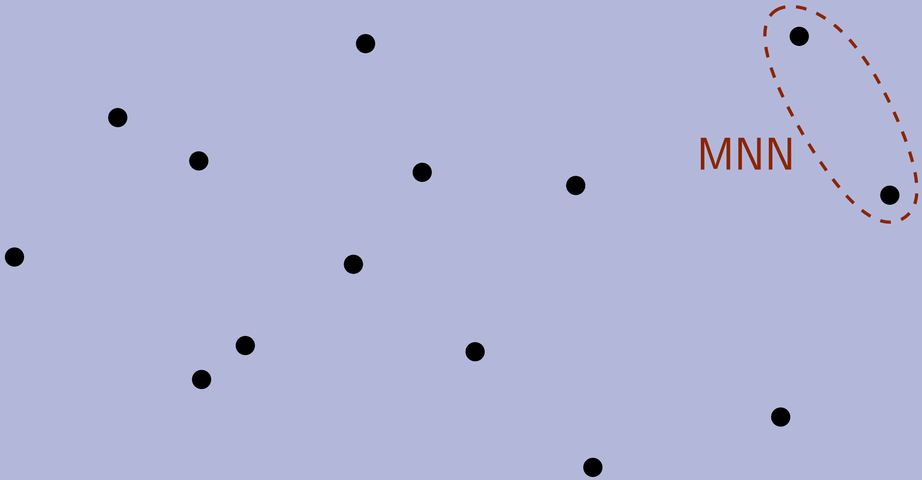


Greedy for Matching

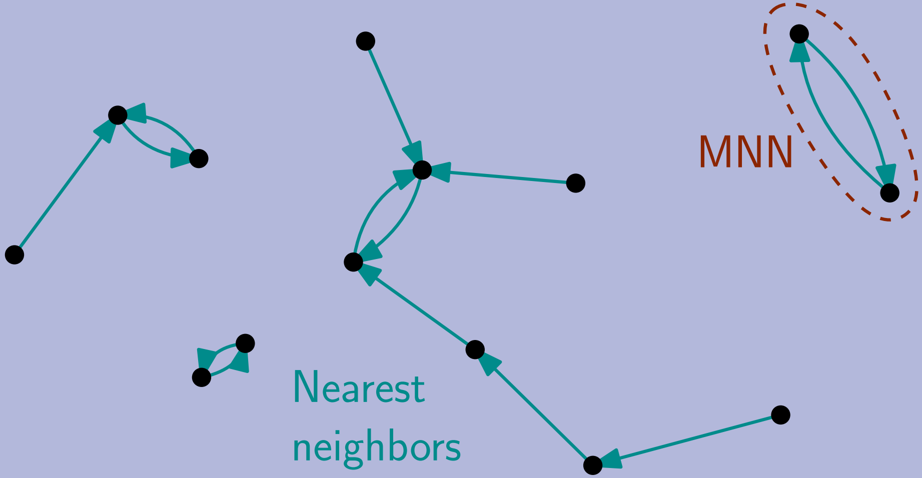


Greedy Ordering

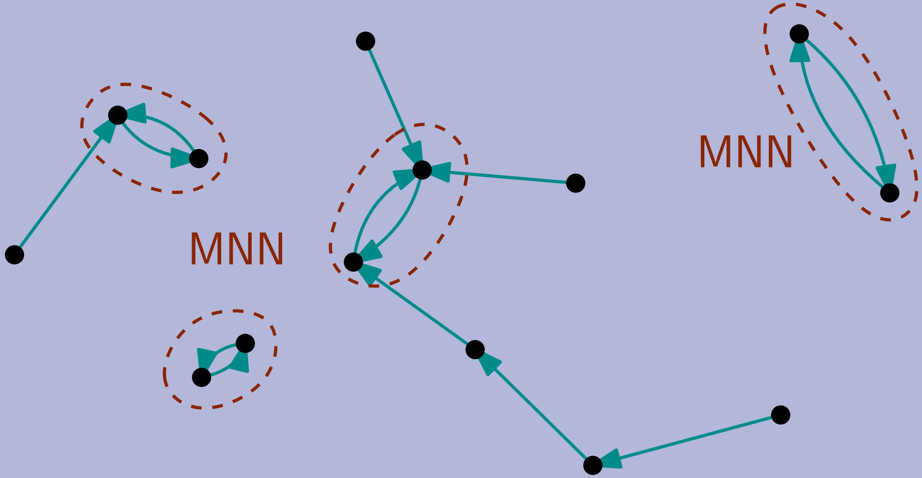
Mutual Nearest Neighbors



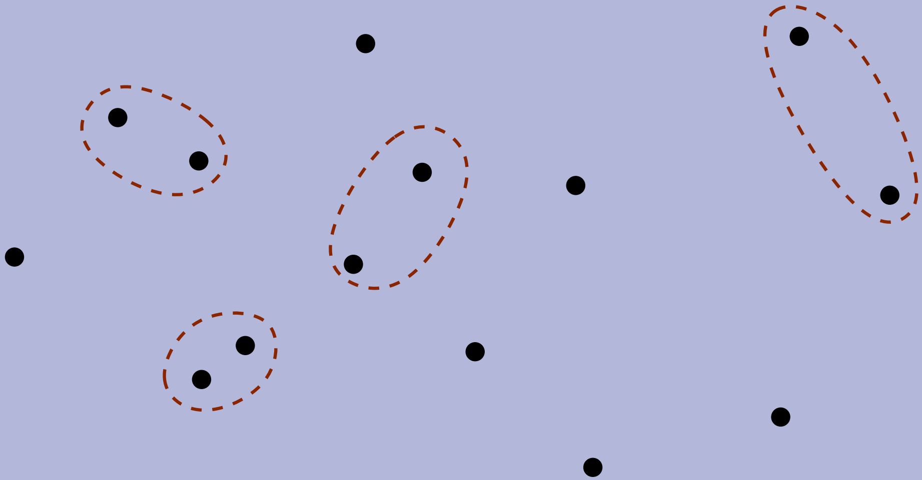
Nearest Neighbor Graph



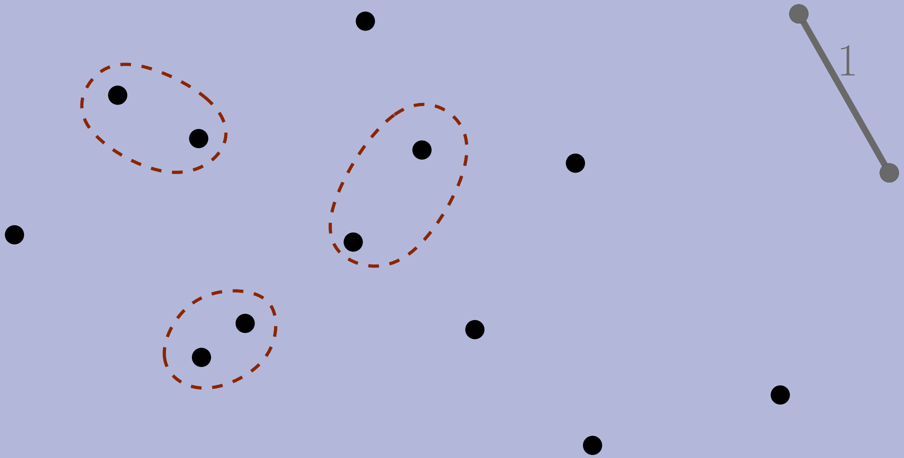
Nearest Neighbor Graph



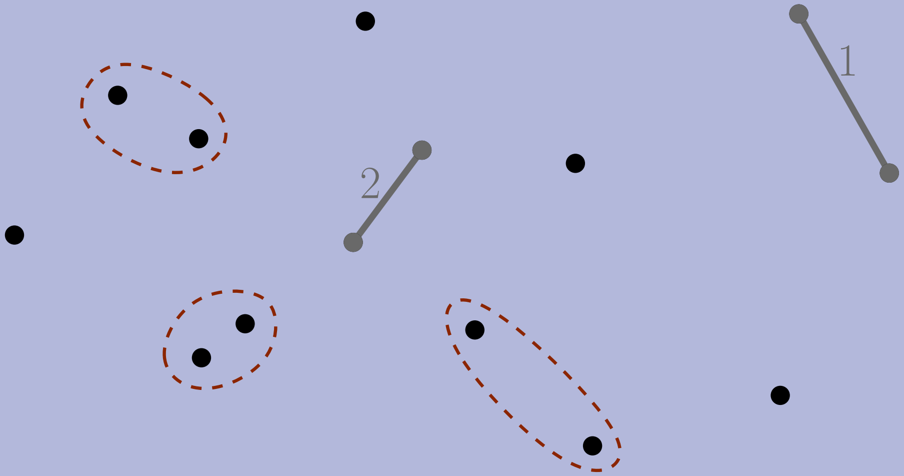
Local Greedy



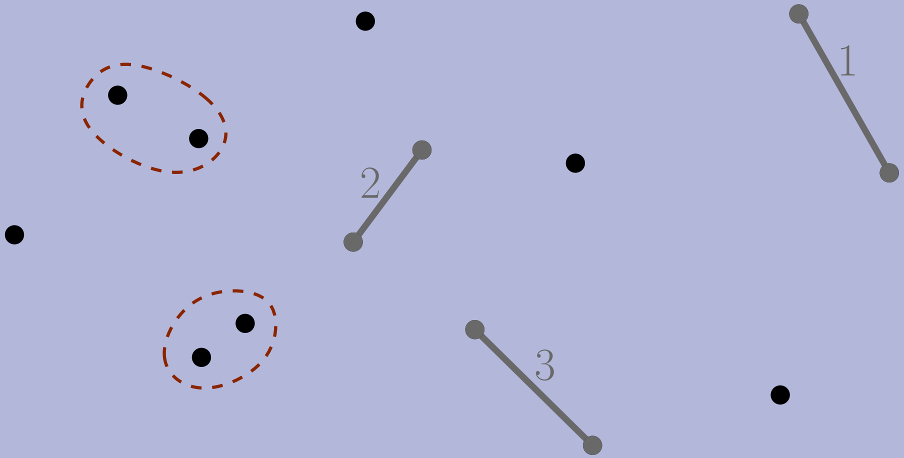
Local Greedy



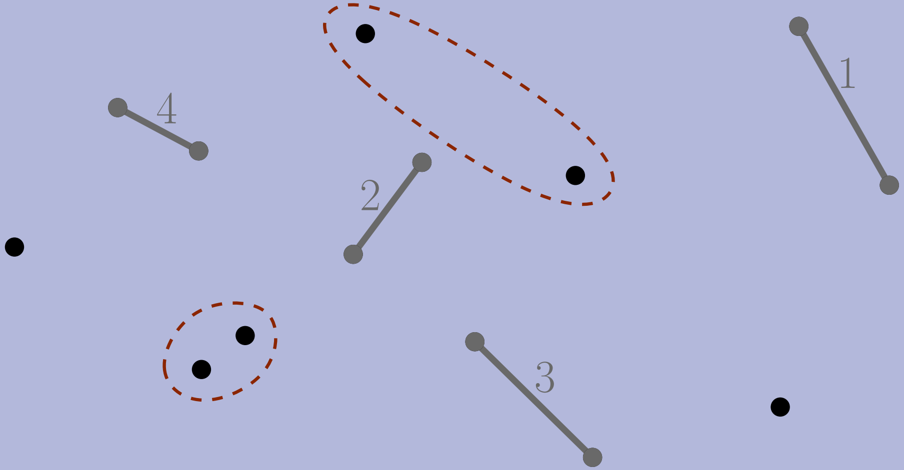
Local Greedy



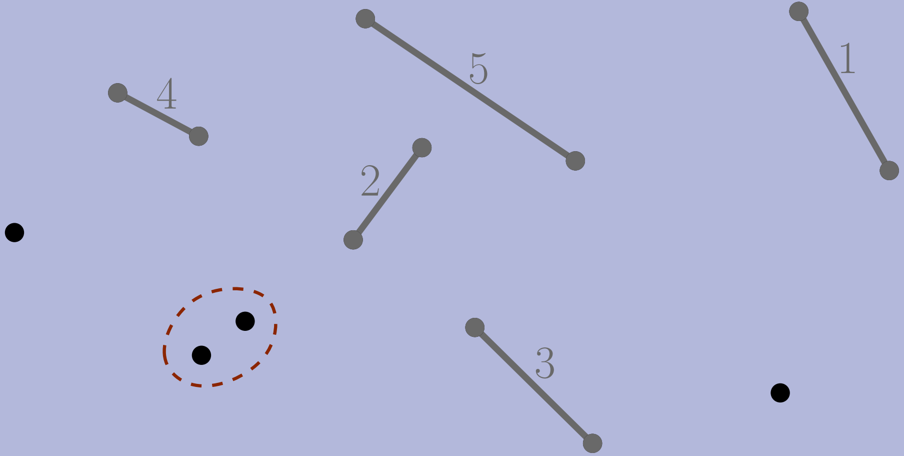
Local Greedy



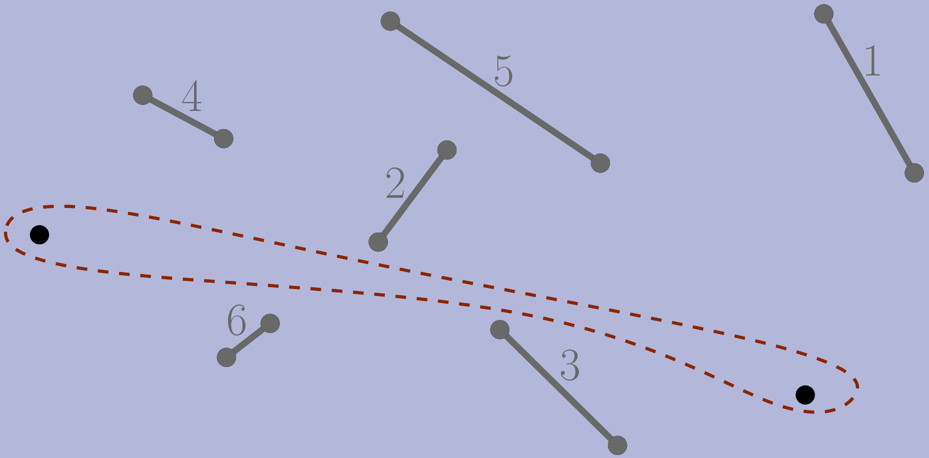
Local Greedy



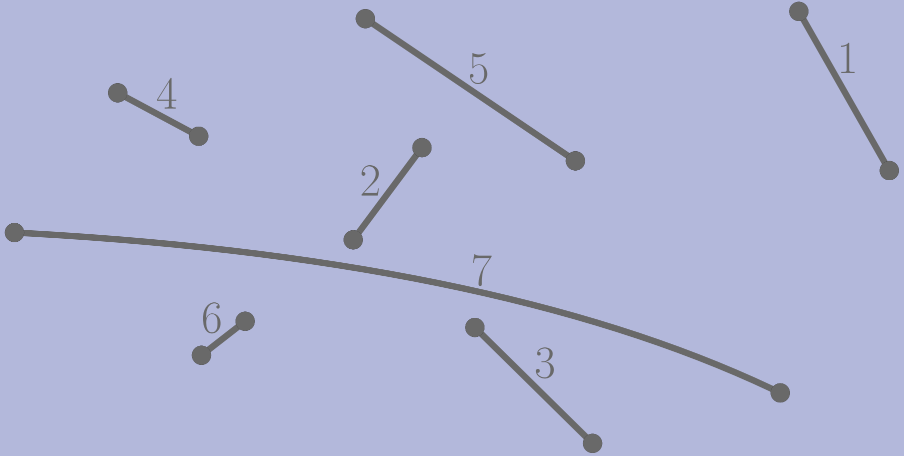
Local Greedy



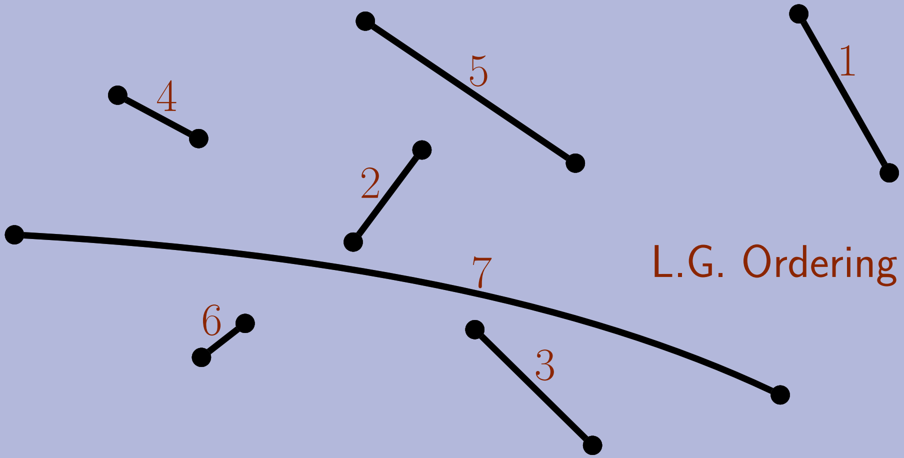
Local Greedy



Local Greedy

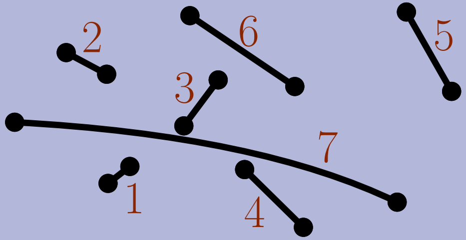


Local Greedy

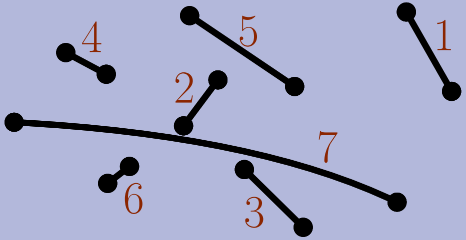


Global—Local Equivalence

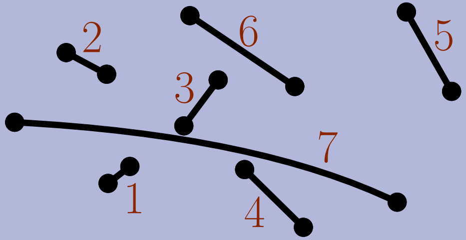
Greedy Ordering

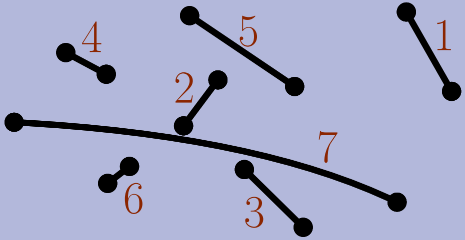


L.G. Ordering



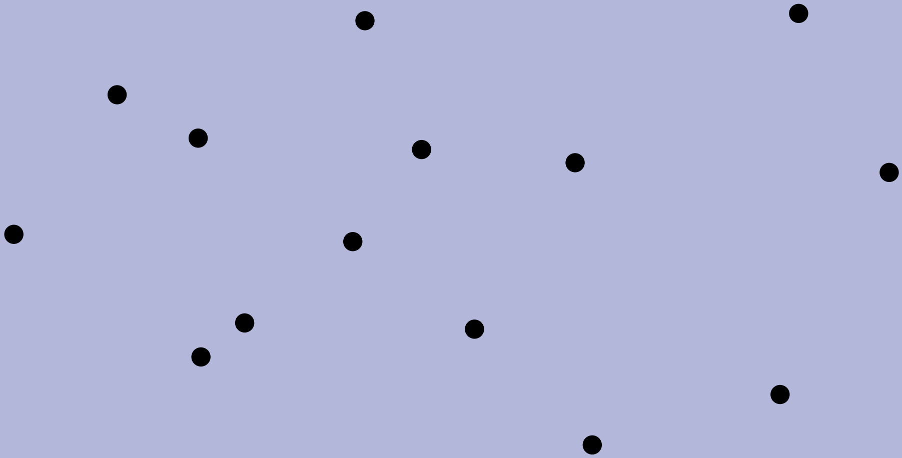
Global—Local Equivalence



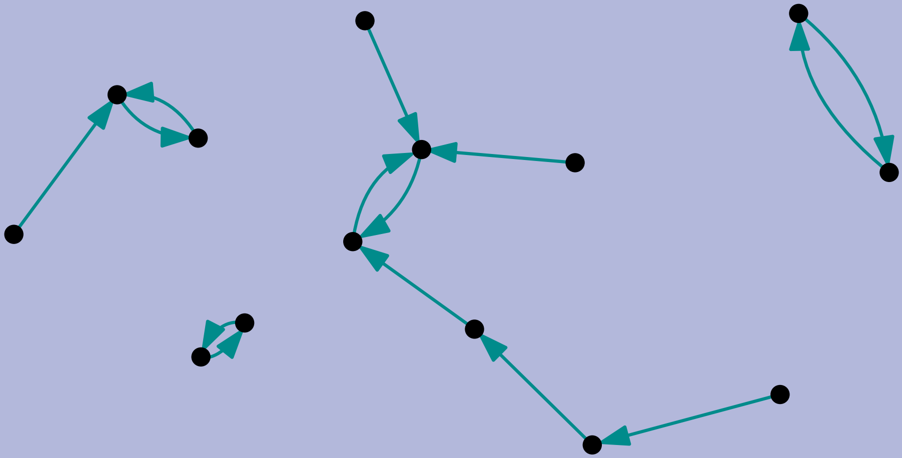


Any run of Local Greedy outputs the Greedy solution

Nearest-Neighbor Chain Algorithm

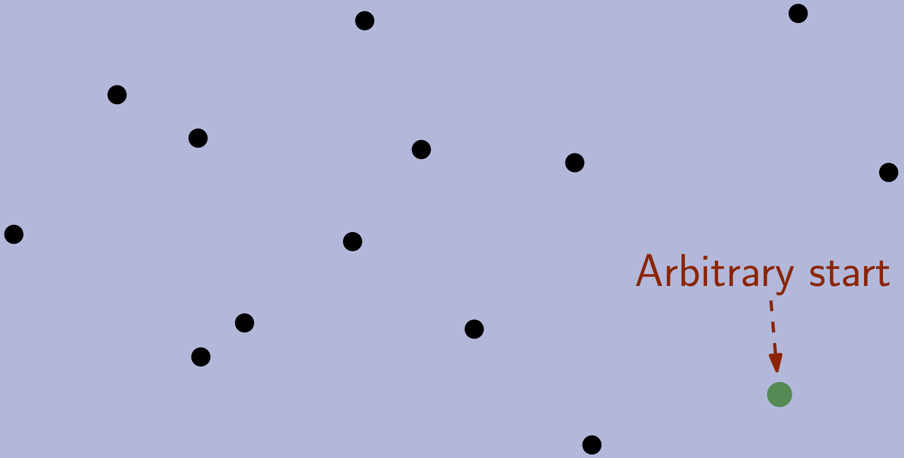


NN Data Structure

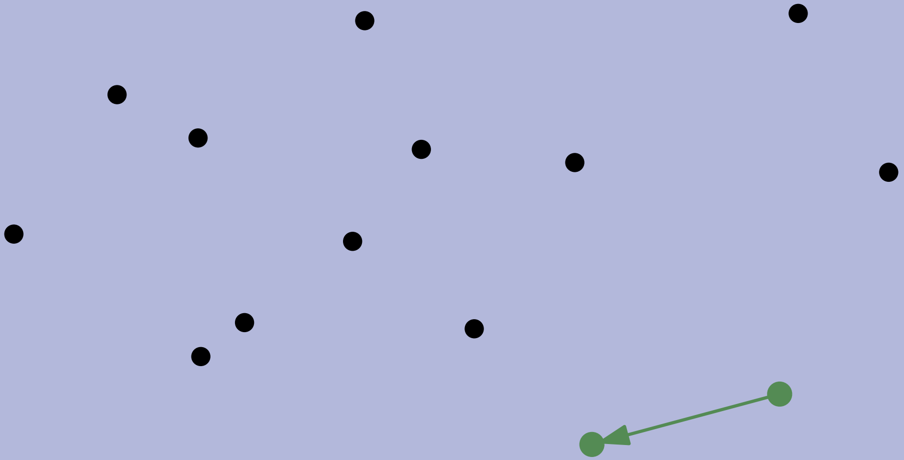


Finding one arrow = 1 NN query

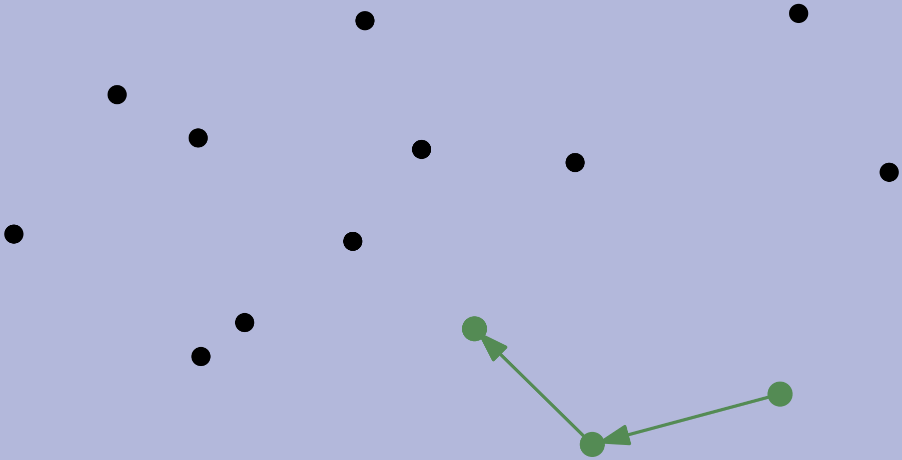
Nearest-Neighbor Chain Algorithm



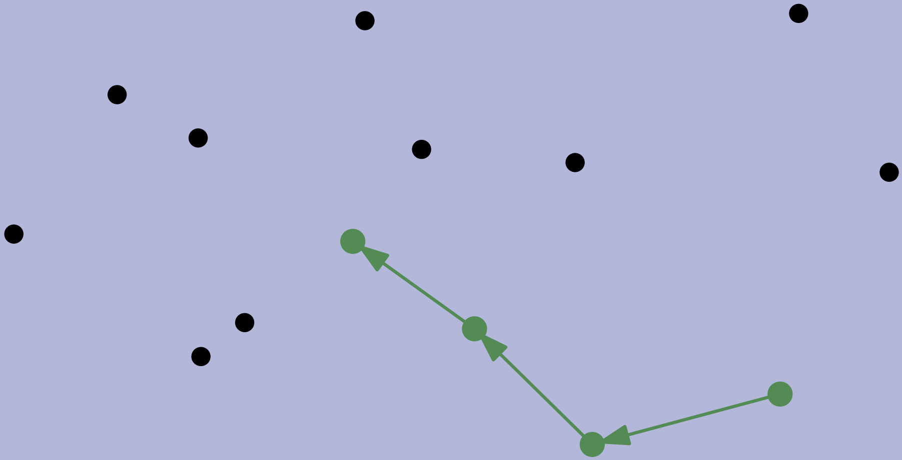
Nearest-Neighbor Chain Algorithm



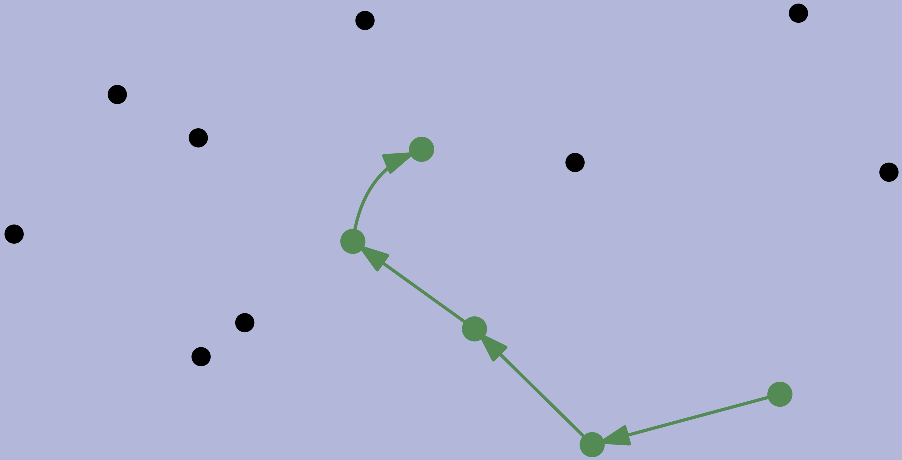
Nearest-Neighbor Chain Algorithm



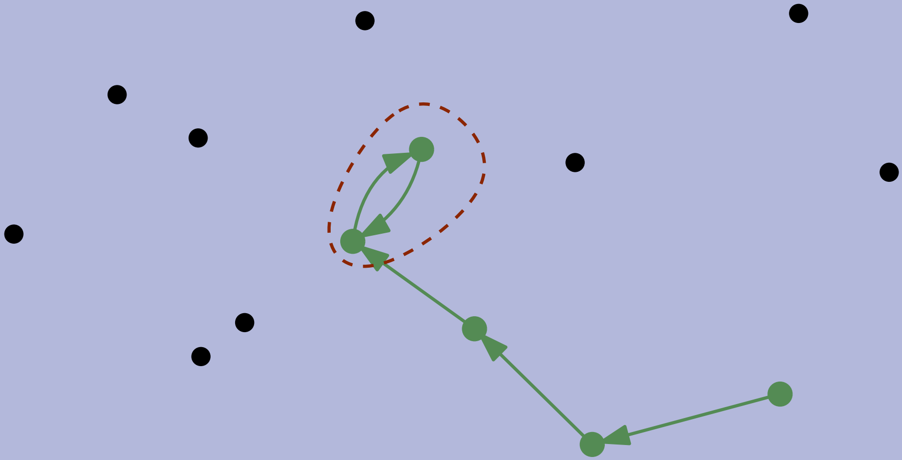
Nearest-Neighbor Chain Algorithm



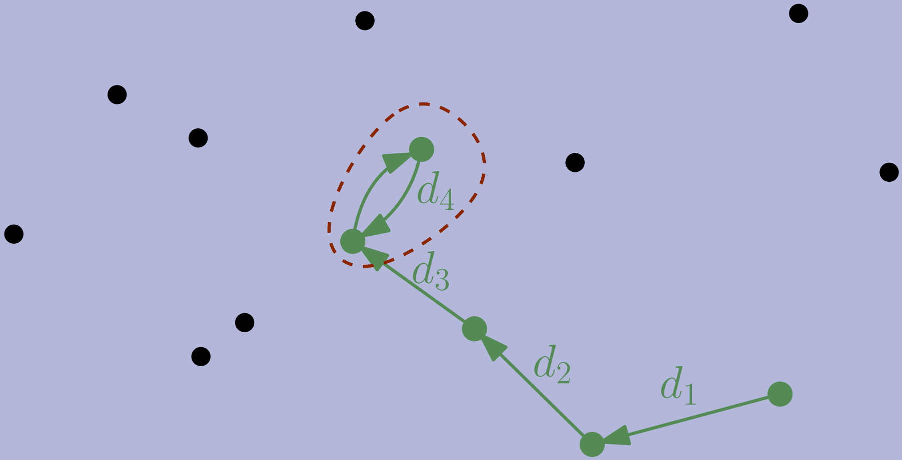
Nearest-Neighbor Chain Algorithm



Nearest-Neighbor Chain Algorithm

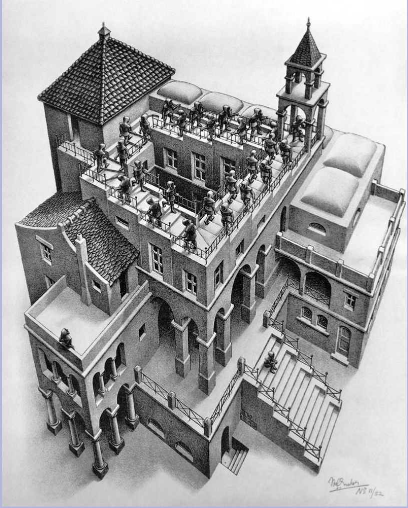


Nearest-Neighbor Chain Algorithm



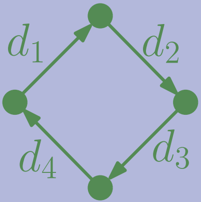
$$d_1 > d_2 > d_3 > d_4$$

Nearest-Neighbor Chain



Ascending and Descending by M. C. Escher

Cycles cannot happen:



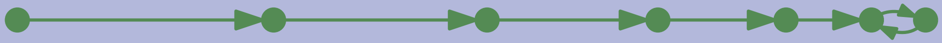
$$\xcancel{d_1 > d_2 > d_3 > d_4 > d_1}$$

Nearest-Neighbor Chain

Cost of finding MNN?

Nearest-Neighbor Chain

Cost of finding MNN?

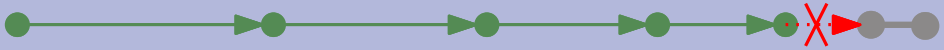


may need n NN queries to find **one** pair of MNN

Nearest-Neighbor Chain

Cost of finding MNN?

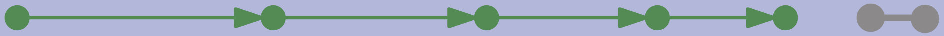
Do not throw away the entire chain!



Nearest-Neighbor Chain

Cost of finding MNN?

Do not throw away the entire chain!

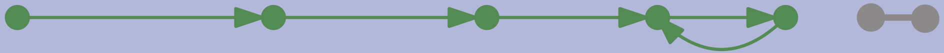


valid chain

Nearest-Neighbor Chain

Cost of finding MNN?

Do not throw away the entire chain!



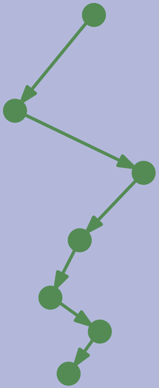
Nearest-Neighbor Chain

Cost of finding MNN?

Do not throw away the entire chain!

Nearest-Neighbor Chain

© 2019 hayneedle.com



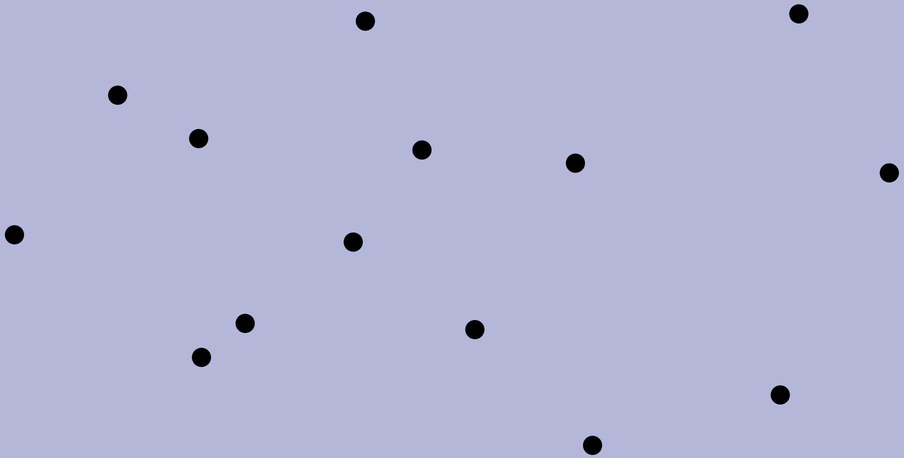
Nearest-Neighbor Chain

Pseudocode

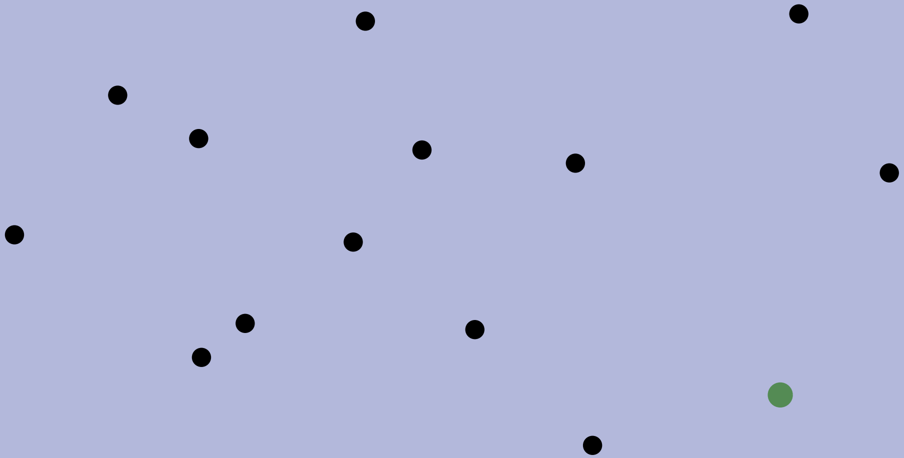
- start the chain from any point
- **until** every point is matched:
  - do a NN query from the top point
  - **if** the NN is not in the chain:
    add it
  - **else** (the NN is the previous point in the chain):
    remove both and match them

    (if the chain becomes empty, restart anywhere)

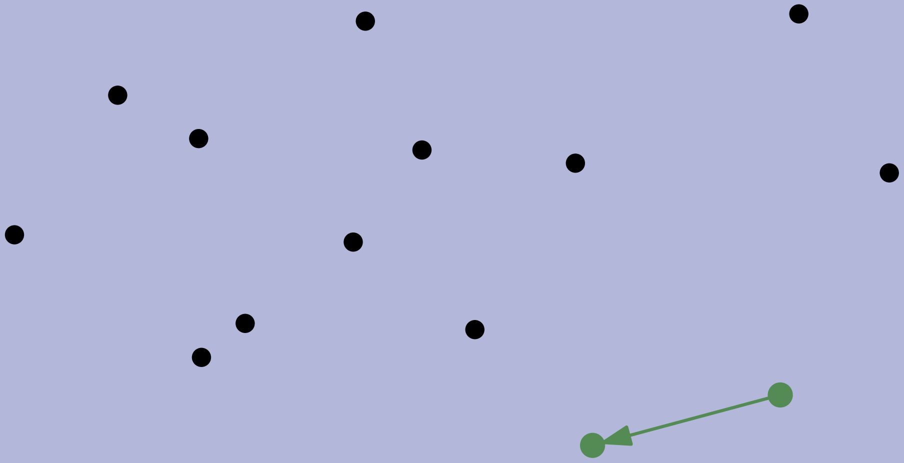
Nearest-Neighbor Chain



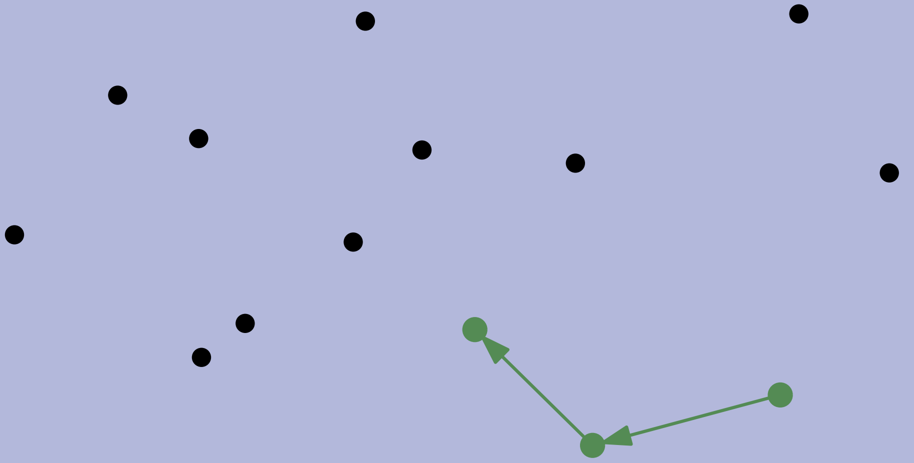
Nearest-Neighbor Chain



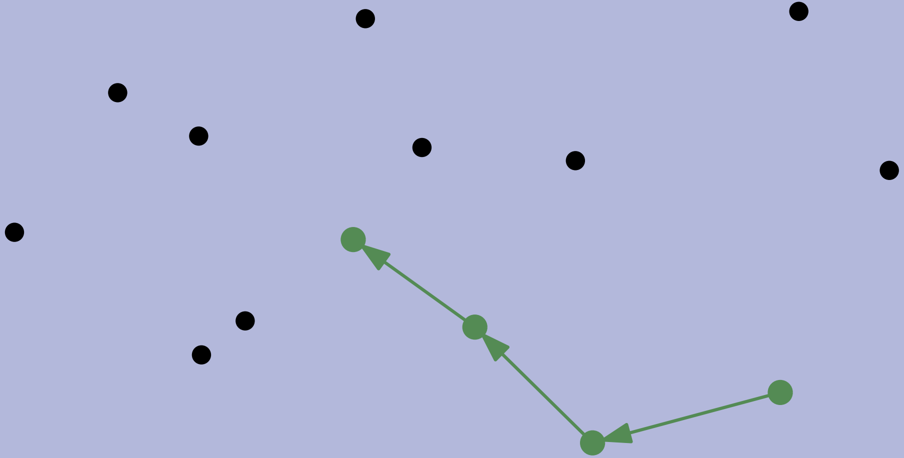
Nearest-Neighbor Chain



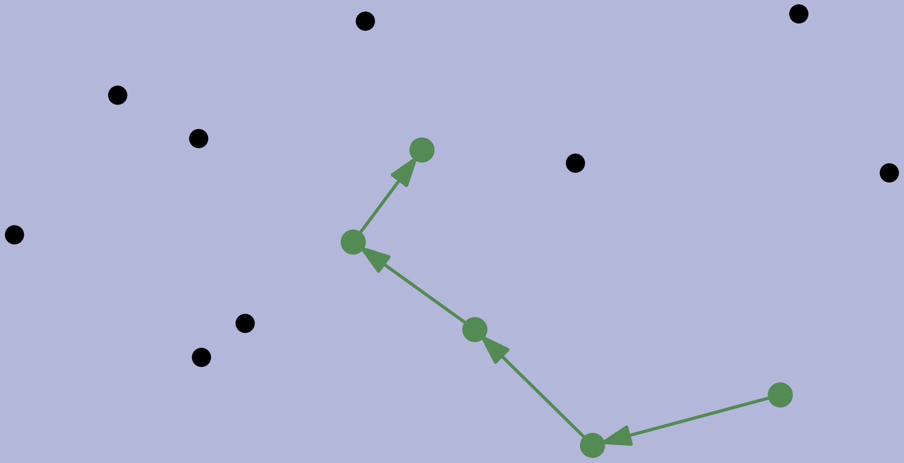
Nearest-Neighbor Chain



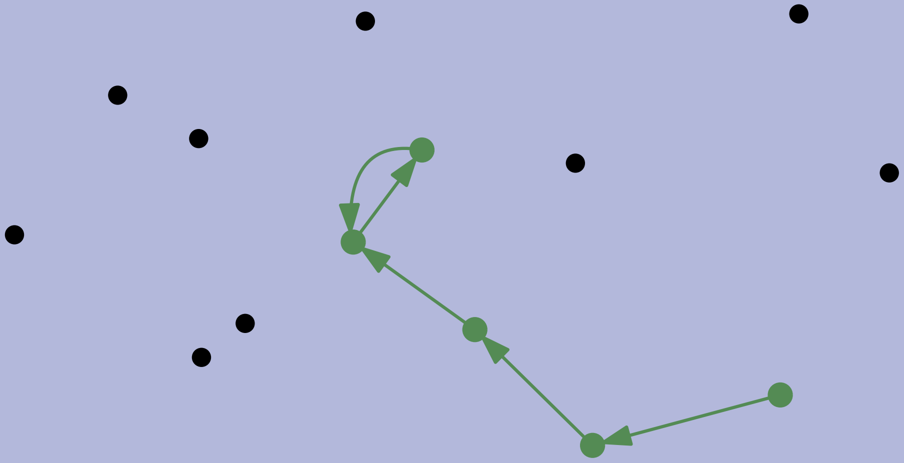
Nearest-Neighbor Chain



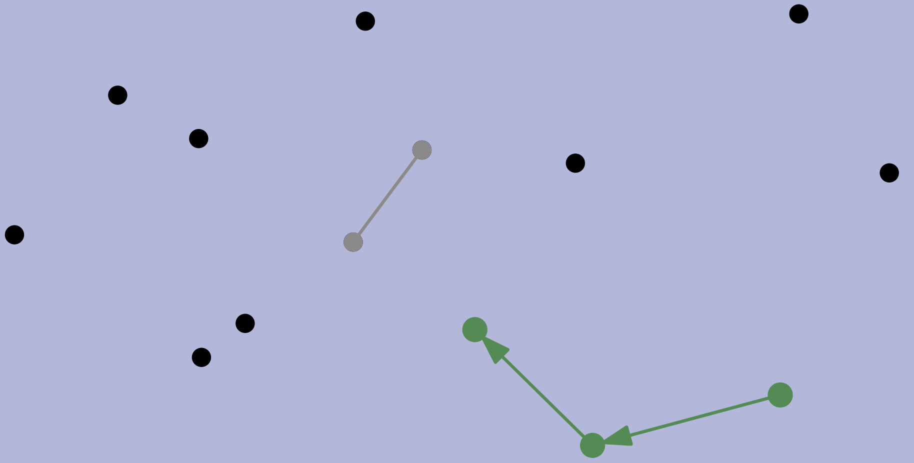
Nearest-Neighbor Chain



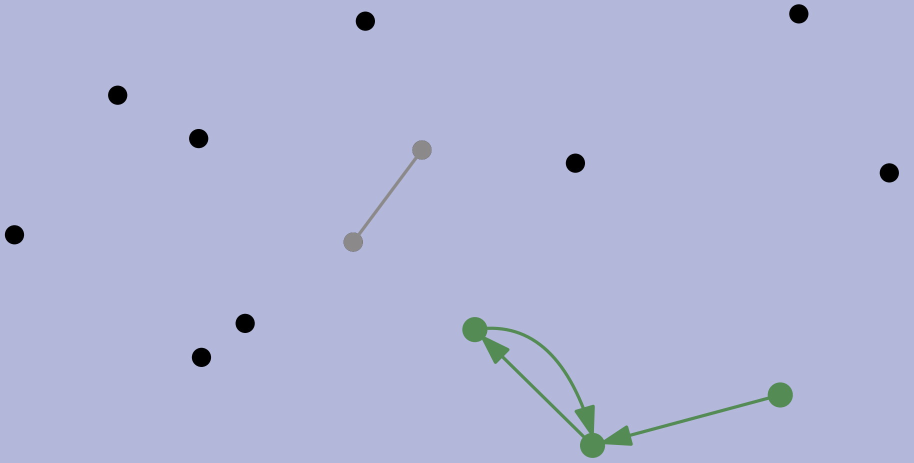
Nearest-Neighbor Chain



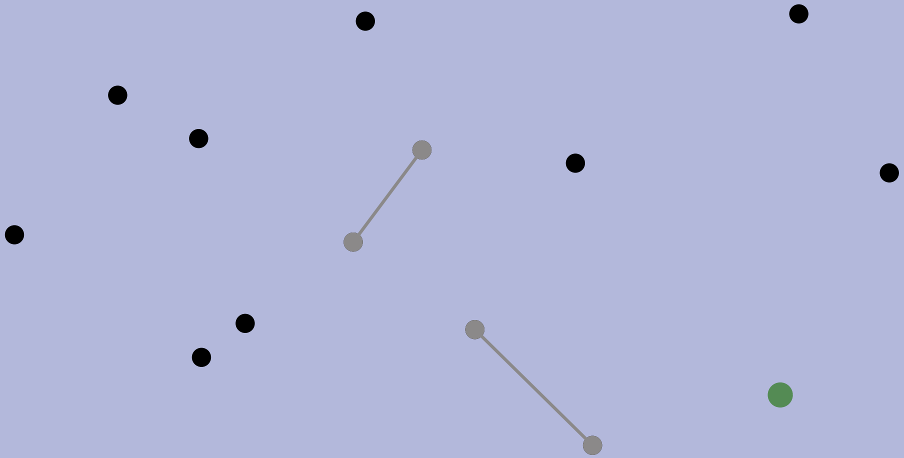
Nearest-Neighbor Chain



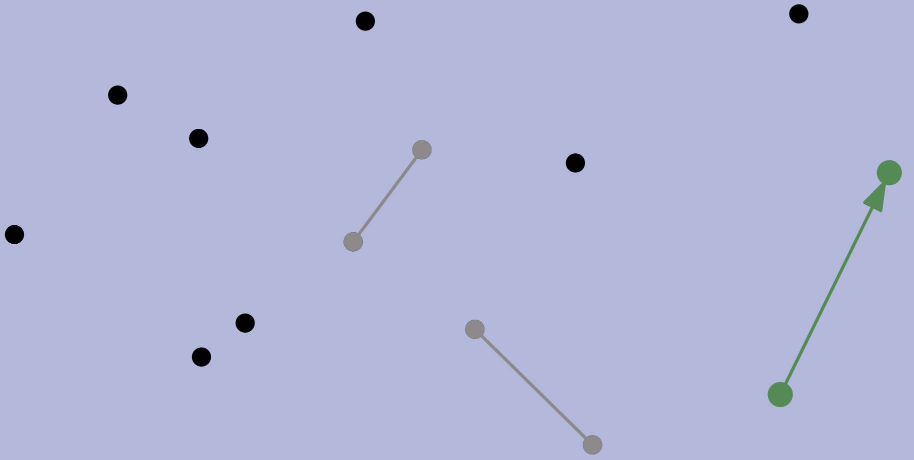
Nearest-Neighbor Chain



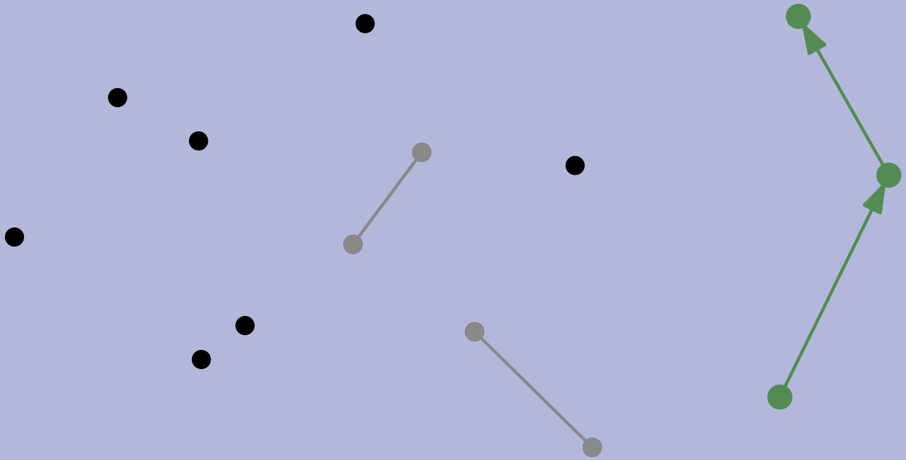
Nearest-Neighbor Chain



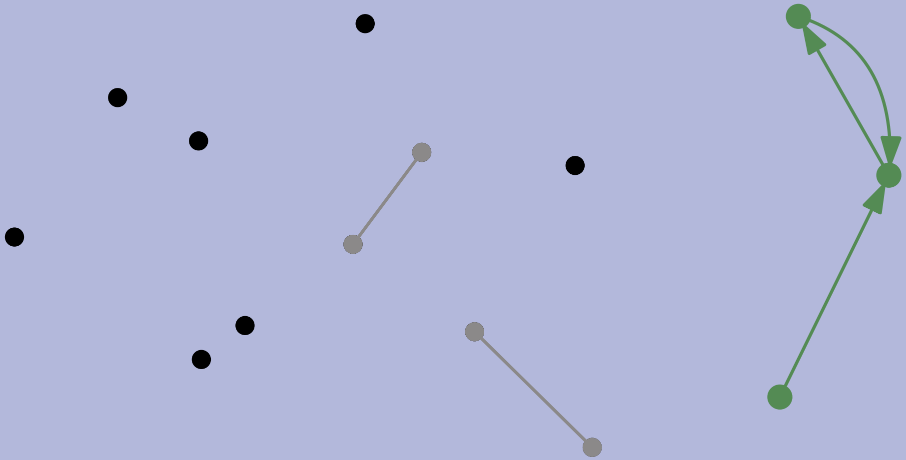
Nearest-Neighbor Chain



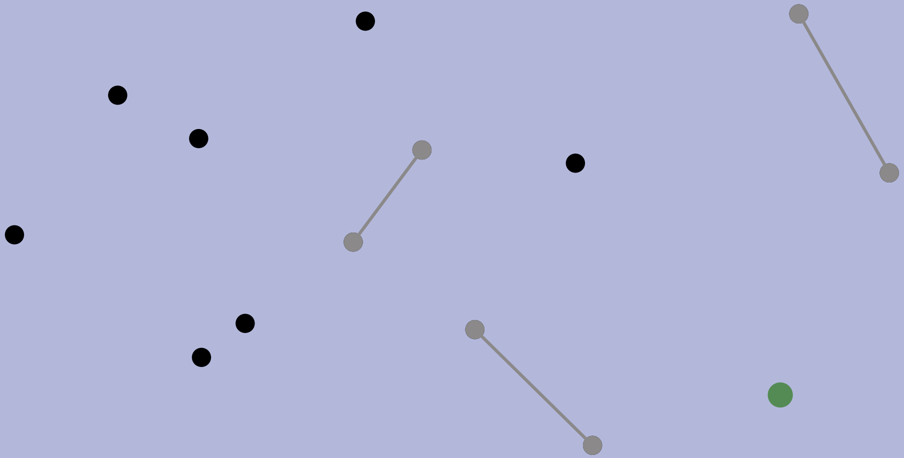
Nearest-Neighbor Chain



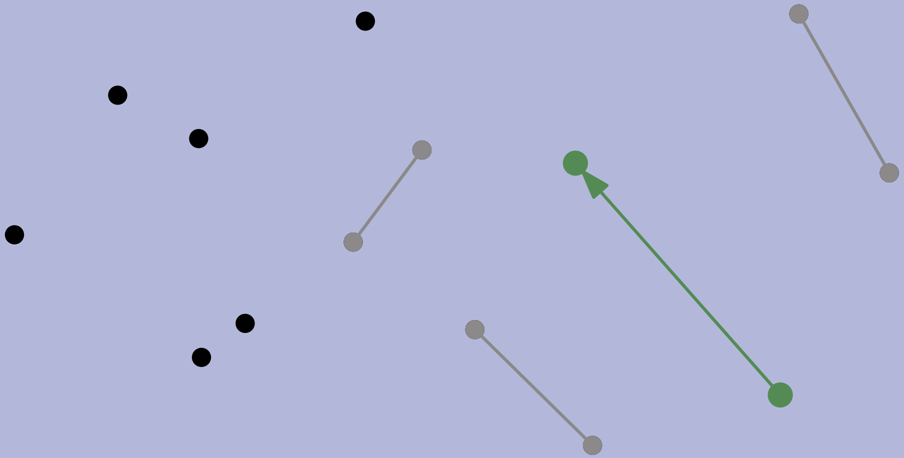
Nearest-Neighbor Chain



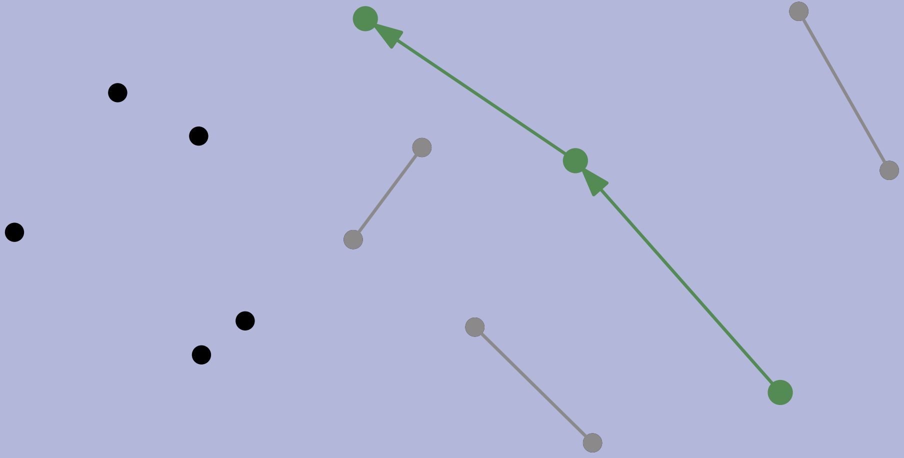
Nearest-Neighbor Chain



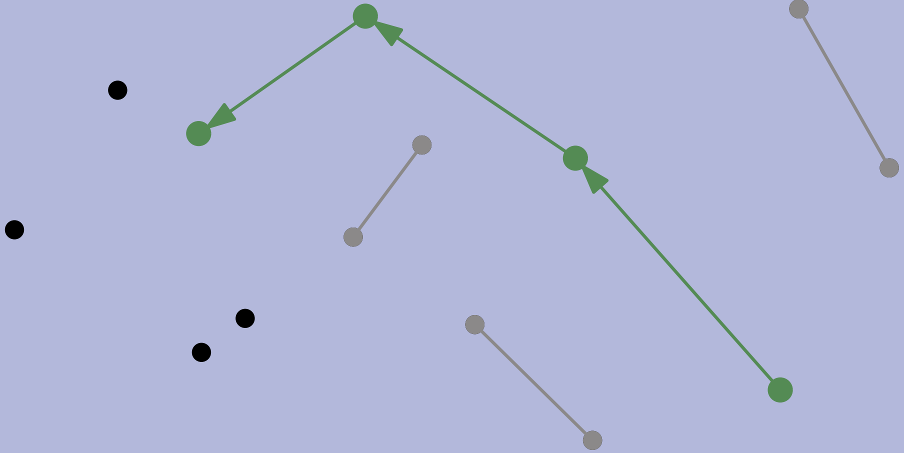
Nearest-Neighbor Chain



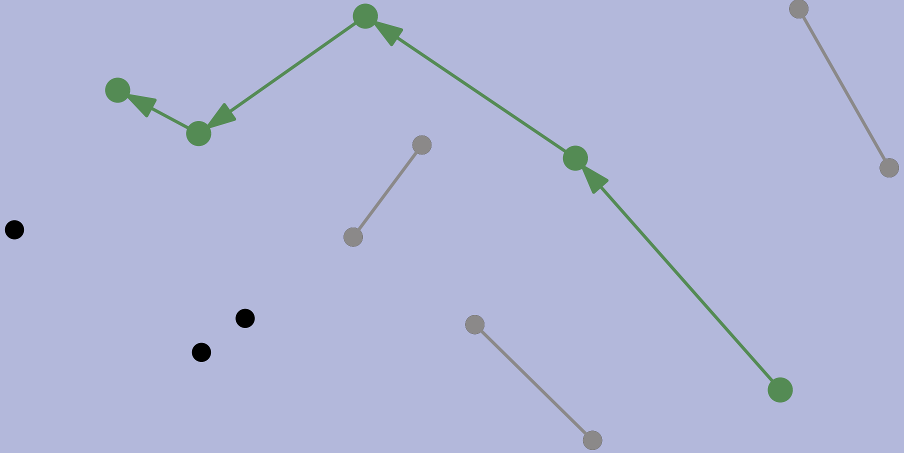
Nearest-Neighbor Chain



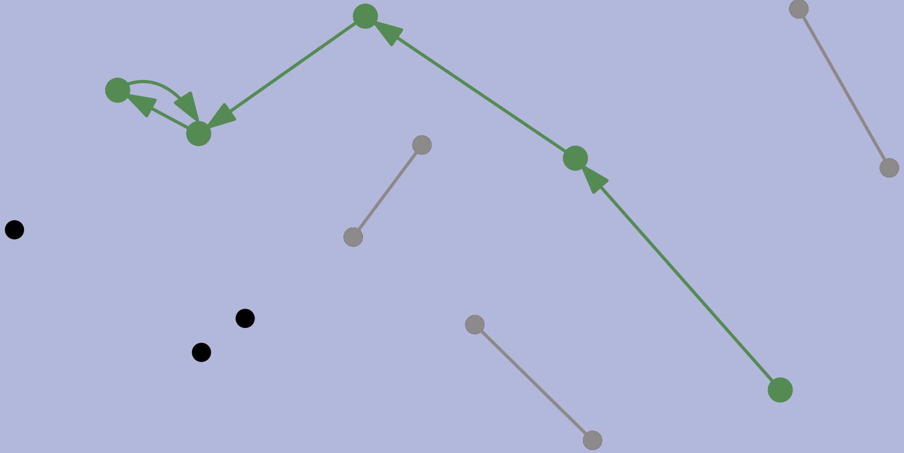
Nearest-Neighbor Chain



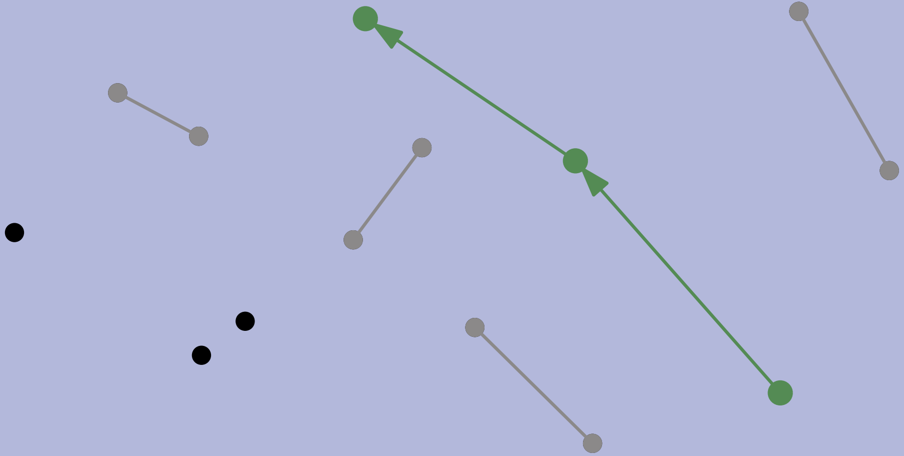
Nearest-Neighbor Chain



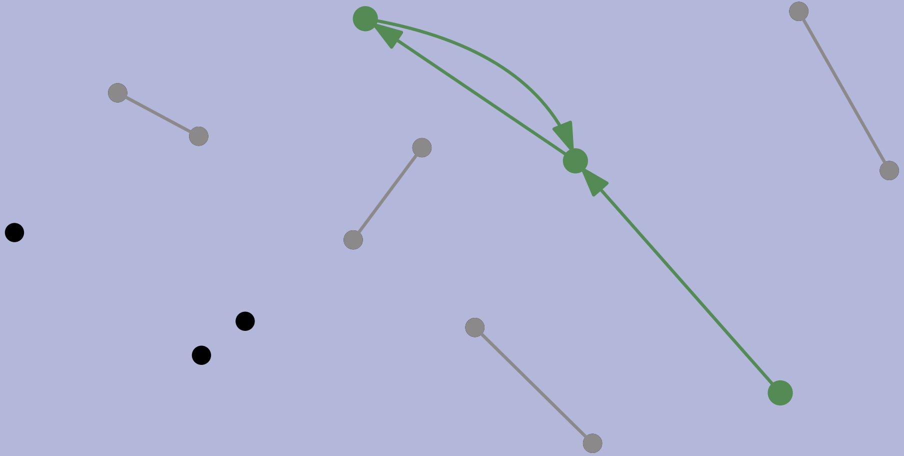
Nearest-Neighbor Chain



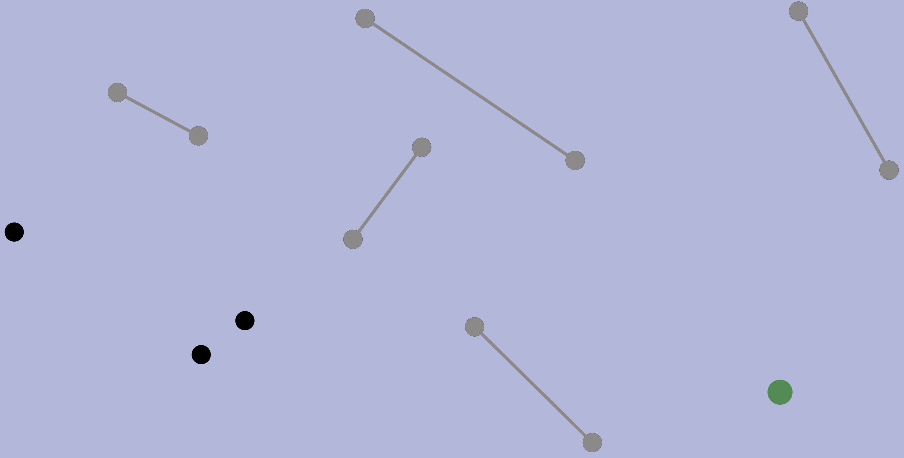
Nearest-Neighbor Chain



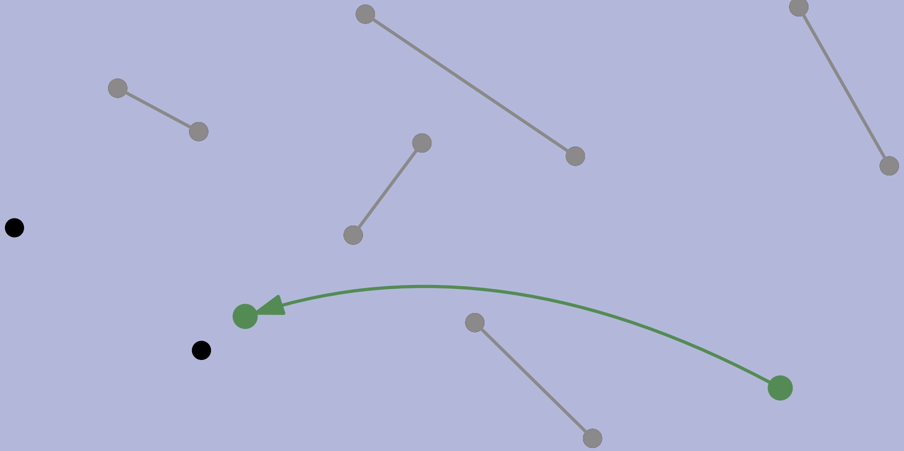
Nearest-Neighbor Chain



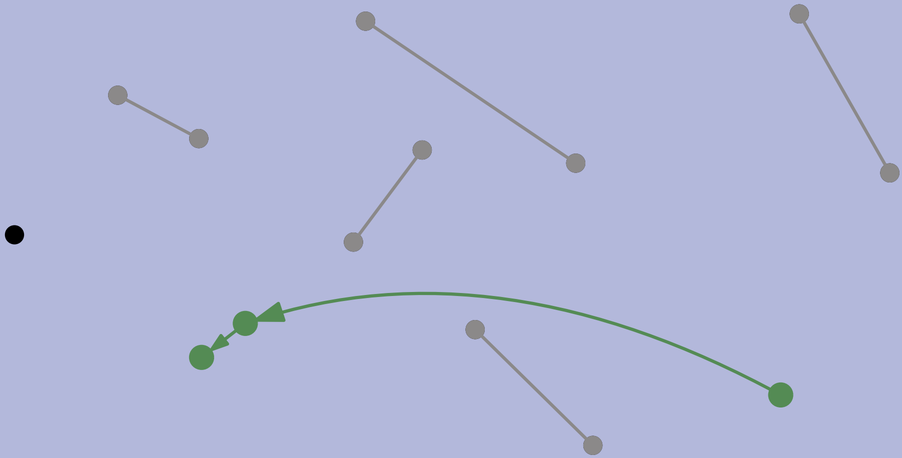
Nearest-Neighbor Chain



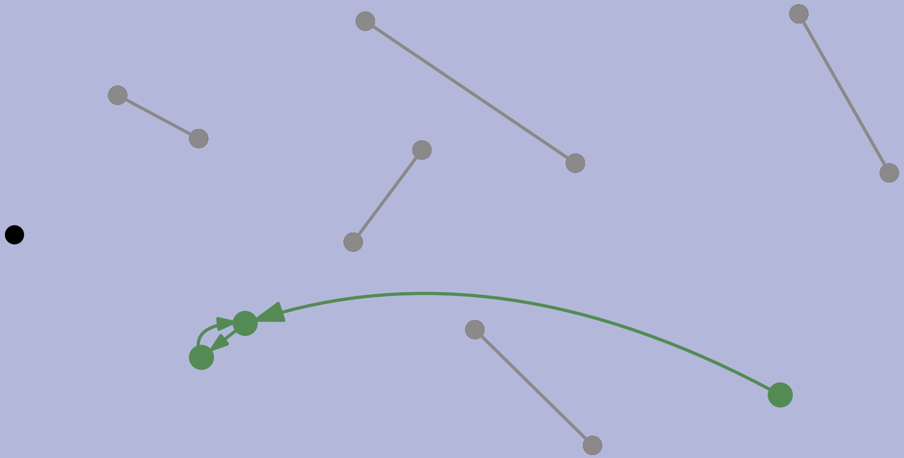
Nearest-Neighbor Chain



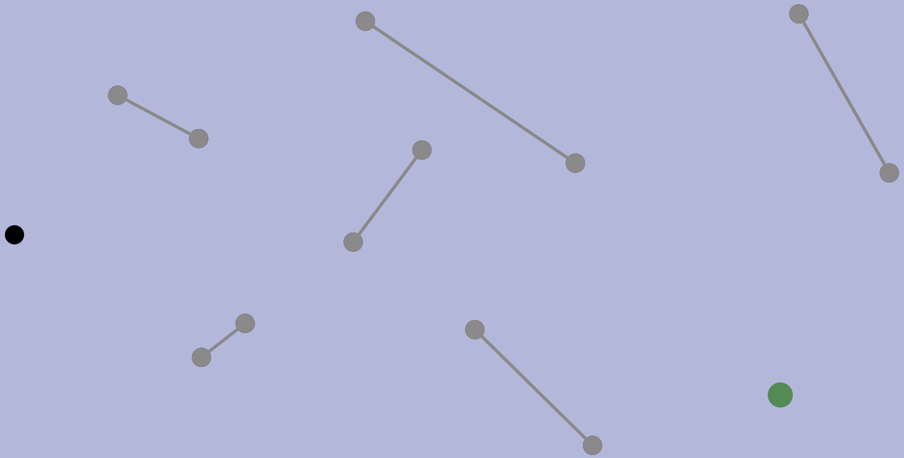
Nearest-Neighbor Chain



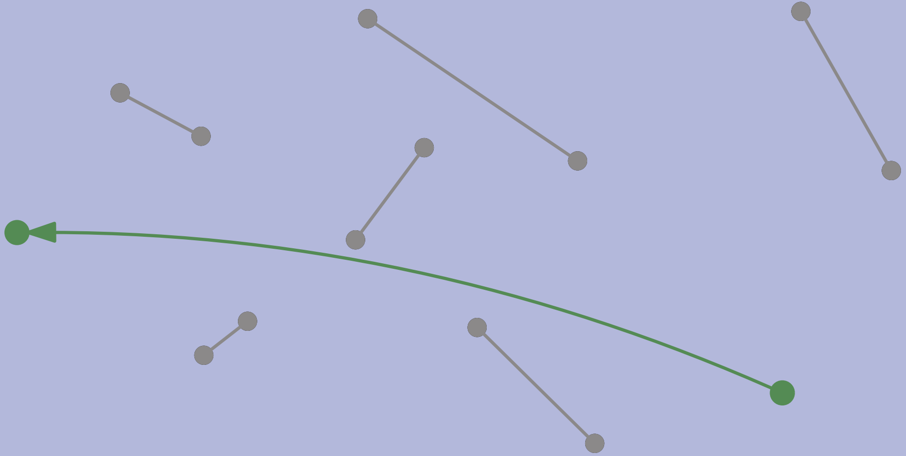
Nearest-Neighbor Chain



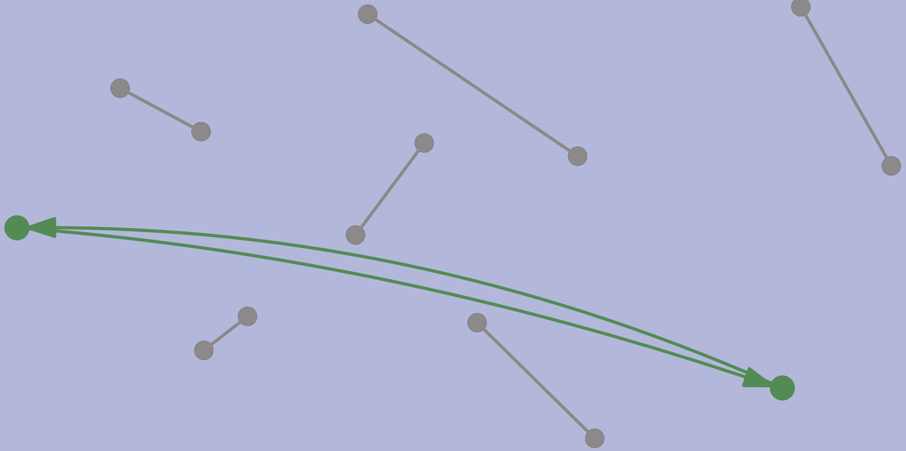
Nearest-Neighbor Chain



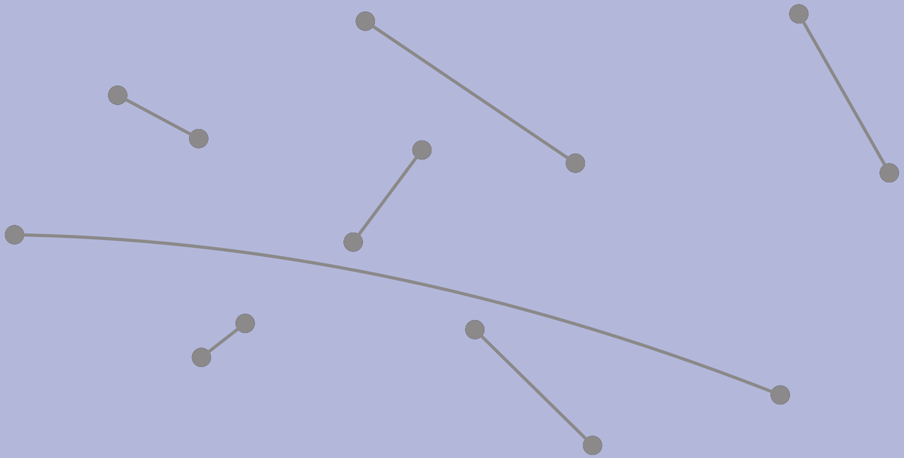
Nearest-Neighbor Chain



Nearest-Neighbor Chain



Nearest-Neighbor Chain



Analysis

Pseudocode

- start the chain from any point
- **until** every point is matched:
  - do a NN query from the top point
  - **if** the NN is not in the chain:
    add it
  - **else** (the NN is the previous point in the chain):
    remove both and match them

  (if the chain becomes empty, restart anywhere)

Analysis

Analysis

Pseudocode

- start the chain from any point
- **until** every point is matched:
  - do a NN query from the top point
  - **if** the NN is not in the chain:
    add it
  - **else** (the NN is the previous point in the chain):
    remove both and match them
    (if the chain becomes empty, restart anywhere)

Analysis

- Each point added and removed to the chain *only once*

Analysis

Pseudocode

- start the chain from any point
- **until** every point is matched:
  - do a NN query from the top point
  - **if** the NN is not in the chain:
    add it
  - **else** (the NN is the previous point in the chain):
    remove both and match them
    (if the chain becomes empty, restart anywhere)

Analysis

- Each point added and removed to the chain *only once*
- after each NN query: 1 point added or 2 removed

Analysis

Pseudocode

- start the chain from any point
- **until** every point is matched:
  - do a NN query from the top point
  - **if** the NN is not in the chain:
    add it
  - **else** (the NN is the previous point in the chain):
    remove both and match them

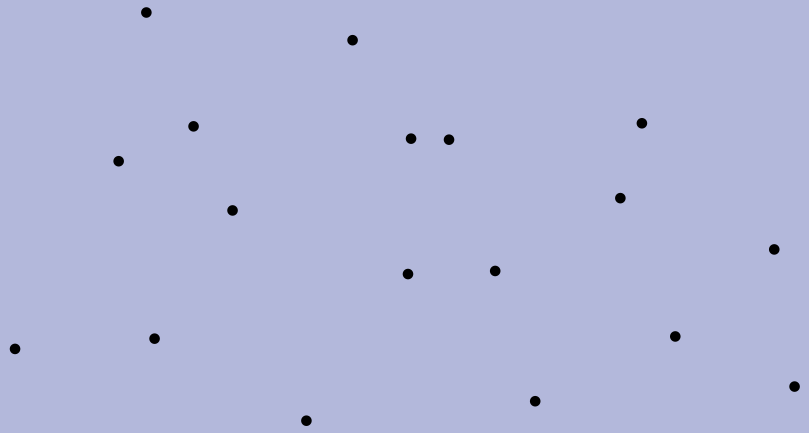
  (if the chain becomes empty, restart anywhere)

Analysis

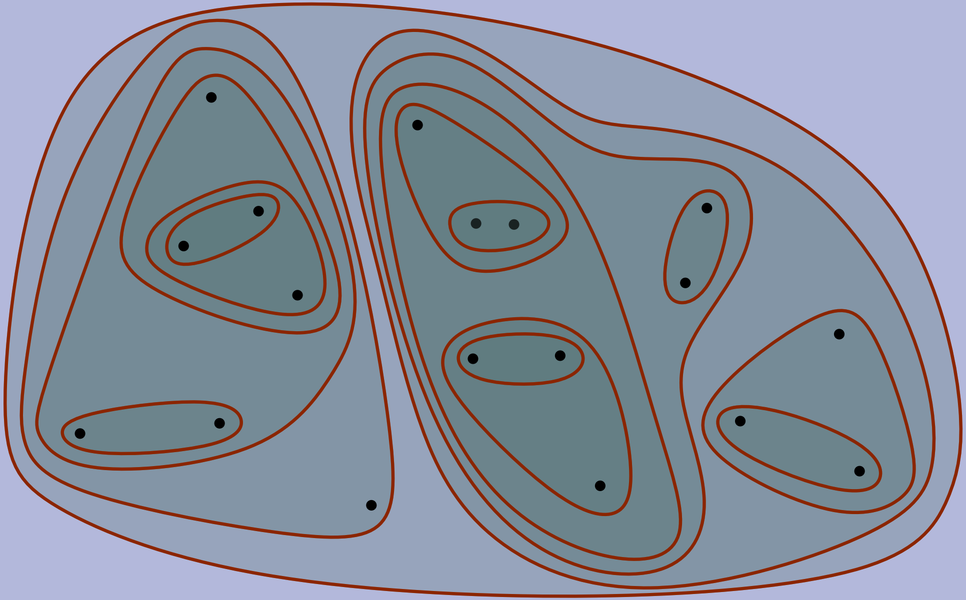
- Each point added and removed to the chain *only once*
- after each NN query: 1 point added or 2 removed

Linear number of NN queries in total: $O(nT(n))$

Hierarchical Clustering

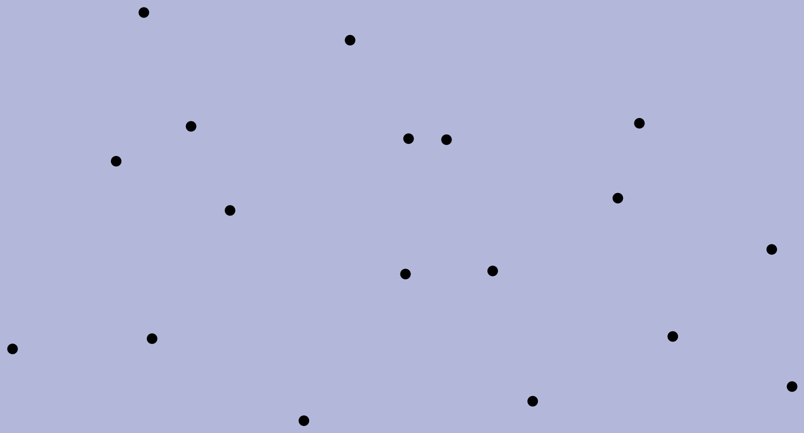


Hierarchical Clustering



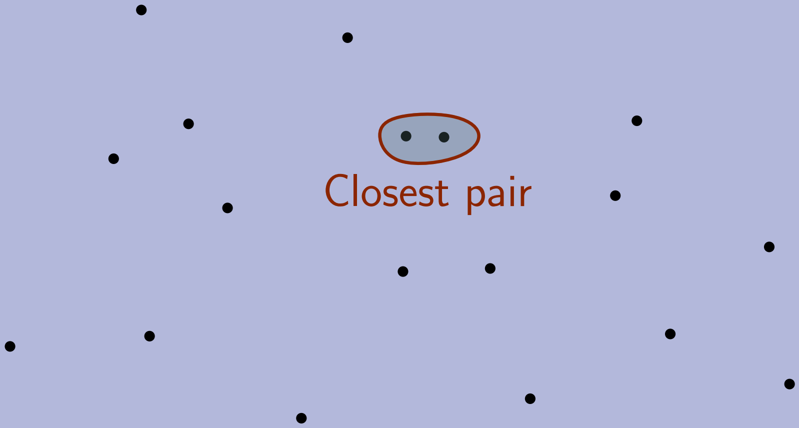
Hierarchical Clustering

Each point starts in its own cluster



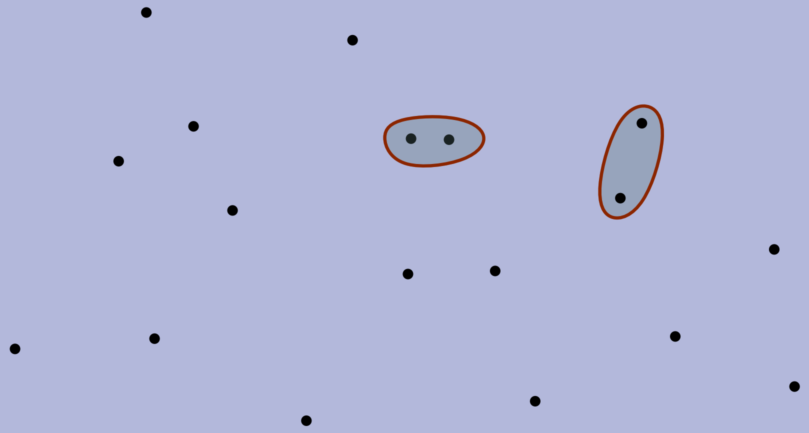
Hierarchical Clustering

Repeatedly merge two closest clusters

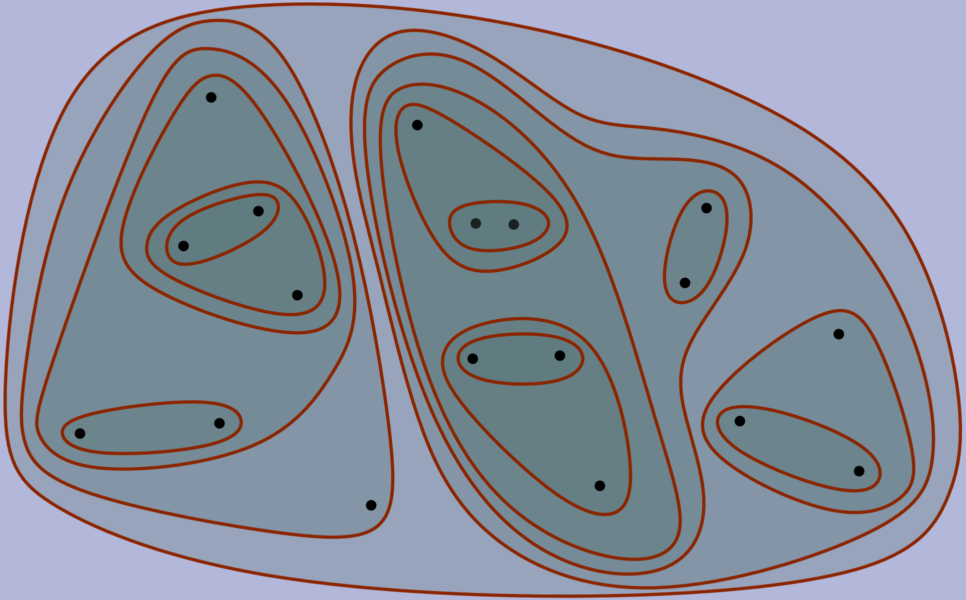


Hierarchical Clustering

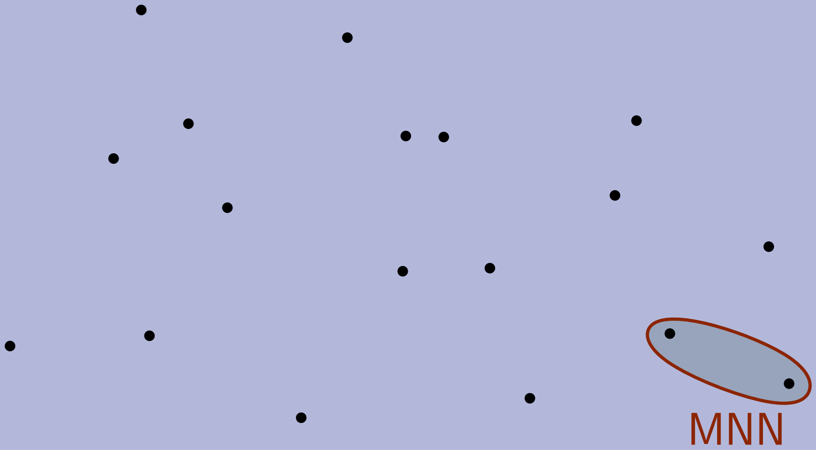
Repeatedly merge two closest clusters



Hierarchical Clustering



Hierarchical Clustering



Hierarchical Clustering

Global—Local Equivalence

Cluster distance:

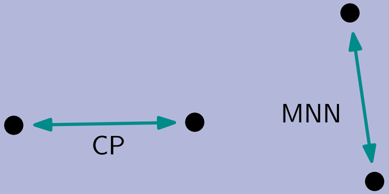
- Min. distance ✓
- Avg. distance ✓
- Max. distance ✓
- Ward's distance ✓
- Centroid distance ✗

Benzécri, J.-P. (1982), "Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques"

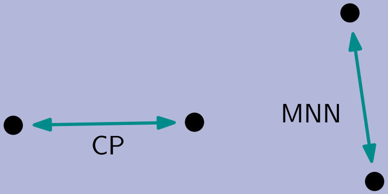
Juan, J. (1982), "Programme de classification hiérarchique par l'algorithme de la recherche en chaîne des voisins réciproques"

Centroid Distance (no GLE)

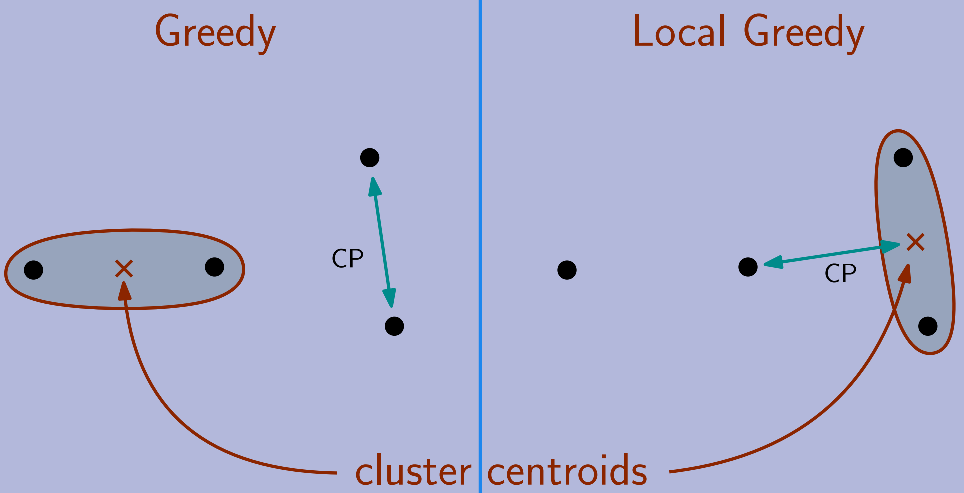
Greedy



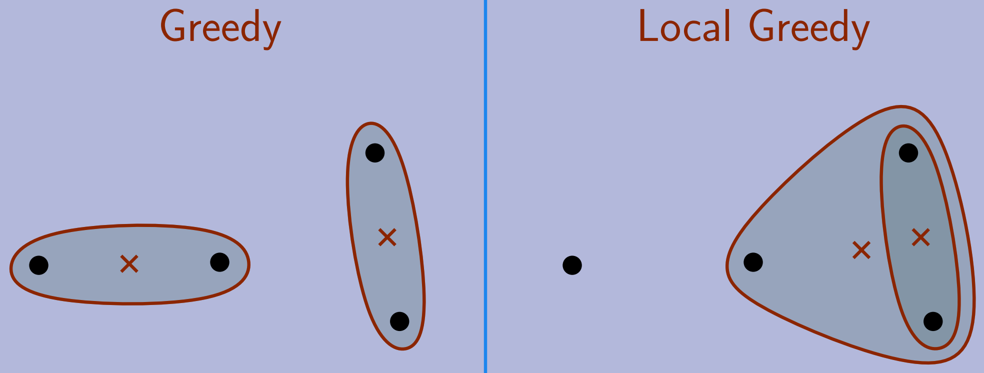
Local Greedy



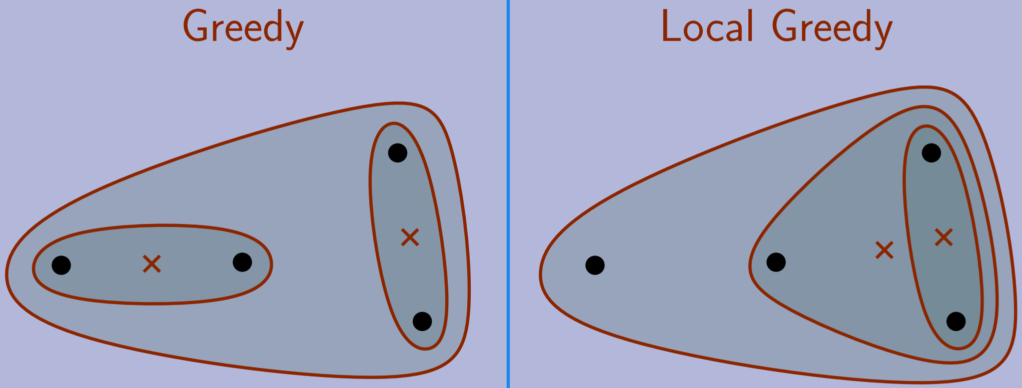
Centroid Distance (no GLE)



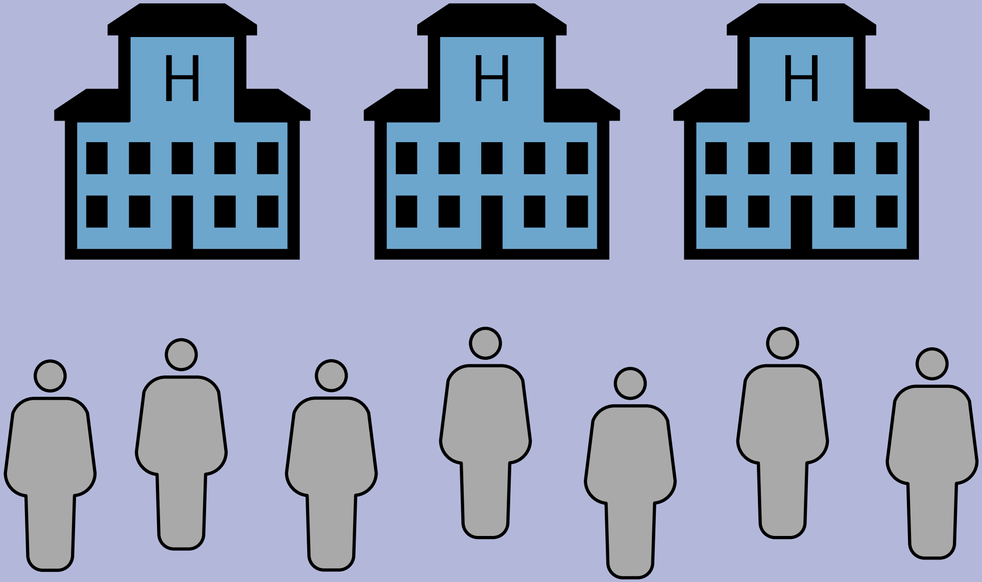
Centroid Distance (no GLE)



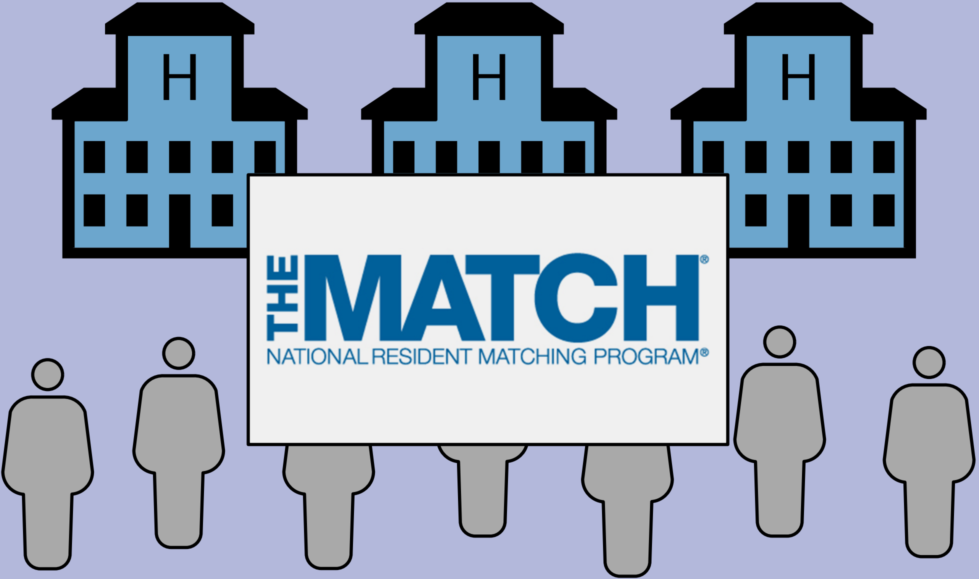
Centroid Distance (no GLE)



Stable Matching



Stable Matching



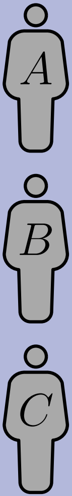
Stable Matching

Preferences

$1 > 3 > 2$

$3 > 1 > 2$

$3 > 2 > 1$



Preferences

$B > C > A$

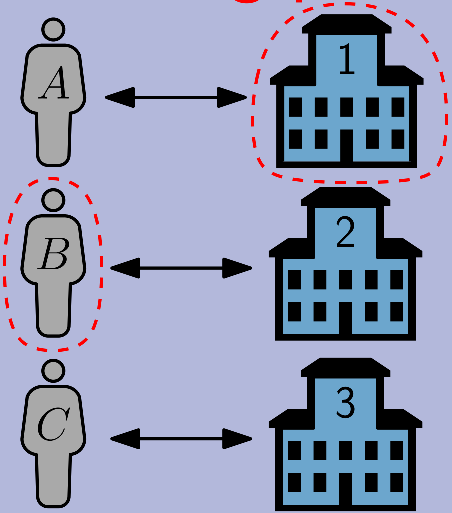
$C > B > A$

$C > A > B$

Stable Matching

Blocking pair

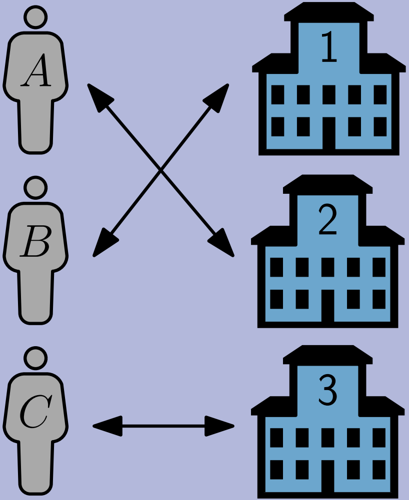
Preferences		Preferences
$1 > 3 > 2$	A ↔ 1	$B > C > A$
$3 > 1 > 2$	B ↔ 2	$C > B > A$
$3 > 2 > 1$	C ↔ 3	$C > A > B$



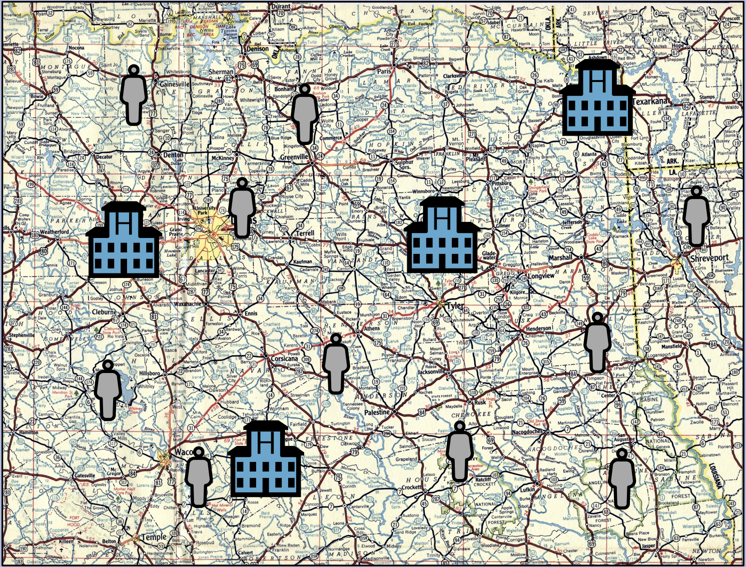
Stable Matching

Stable

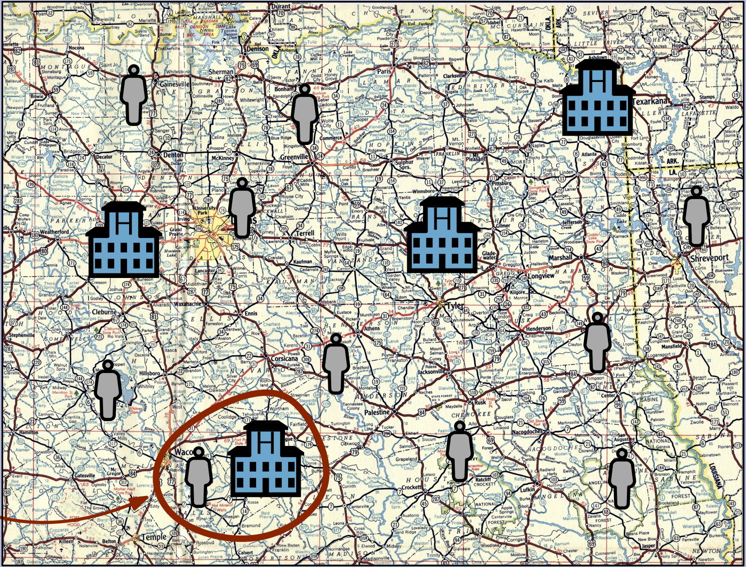
Preferences		Preferences
$1 > 3 > 2$	A	$B > C > A$
$3 > 1 > 2$	B	$C > B > A$
$3 > 2 > 1$	C	$C > A > B$



Geometric Stable Matching

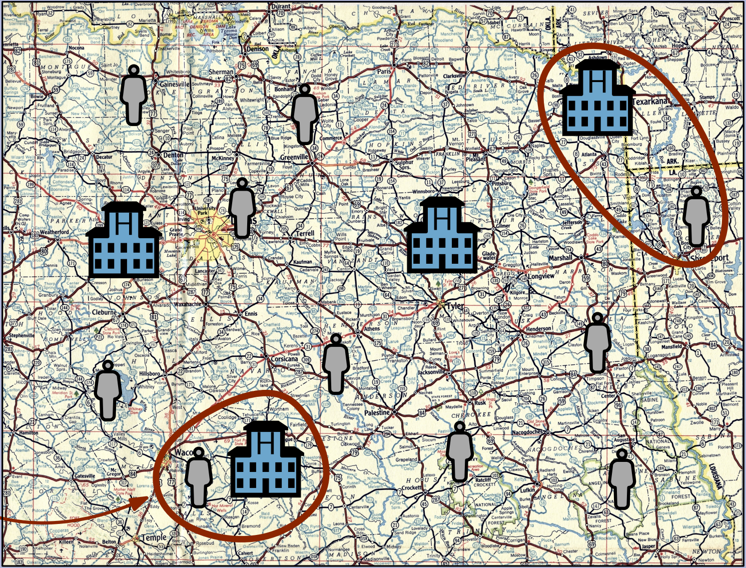


Geometric Stable Matching



Closest pair

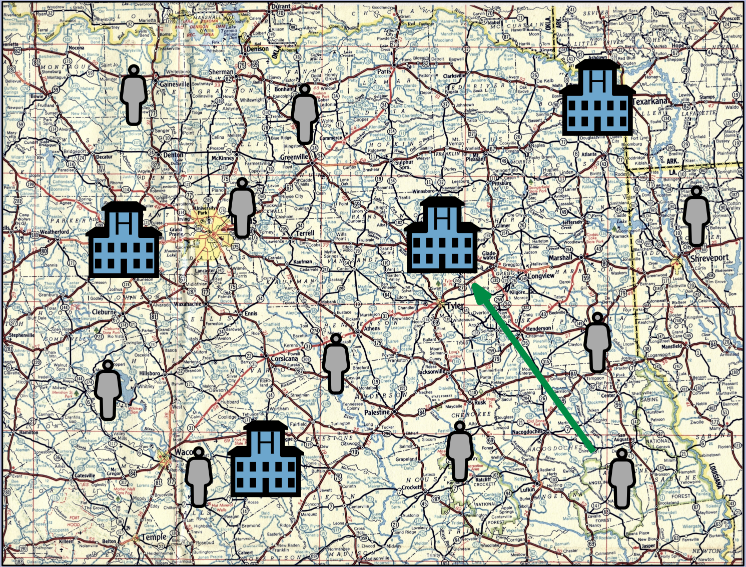
Geometric Stable Matching



MNN

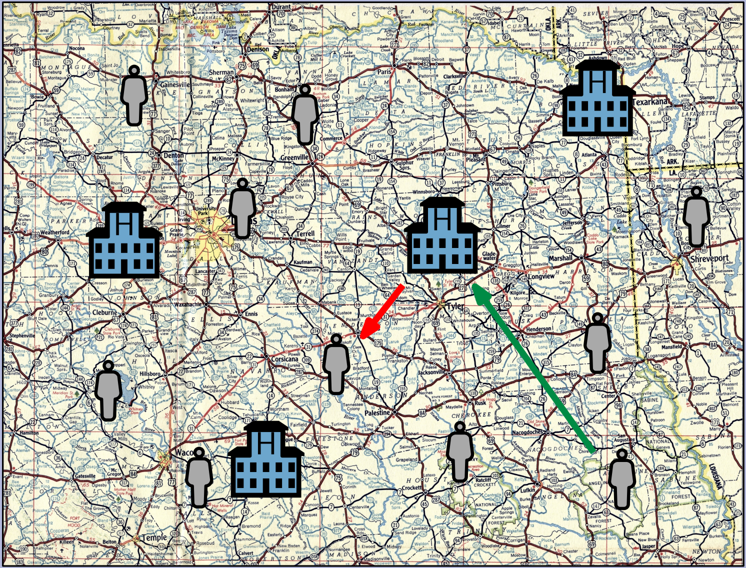
Closest pair

Geometric Stable Matching



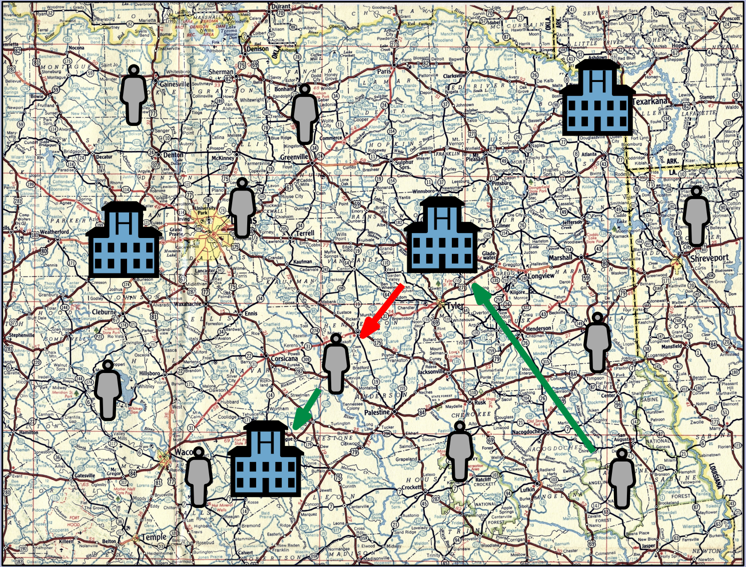
NNC

Geometric Stable Matching



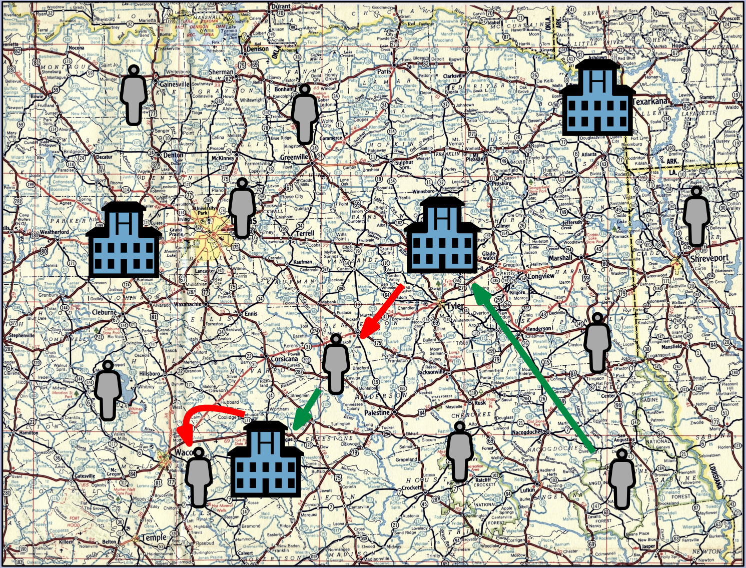
NNC

Geometric Stable Matching



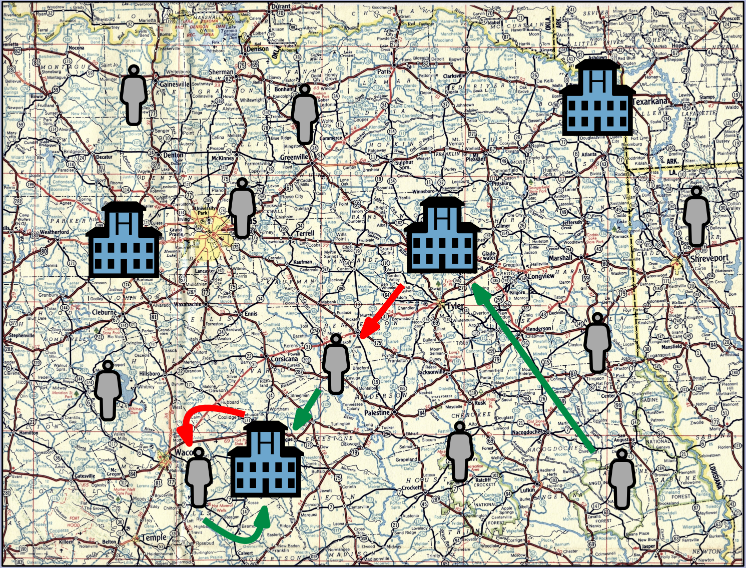
NNC

Geometric Stable Matching



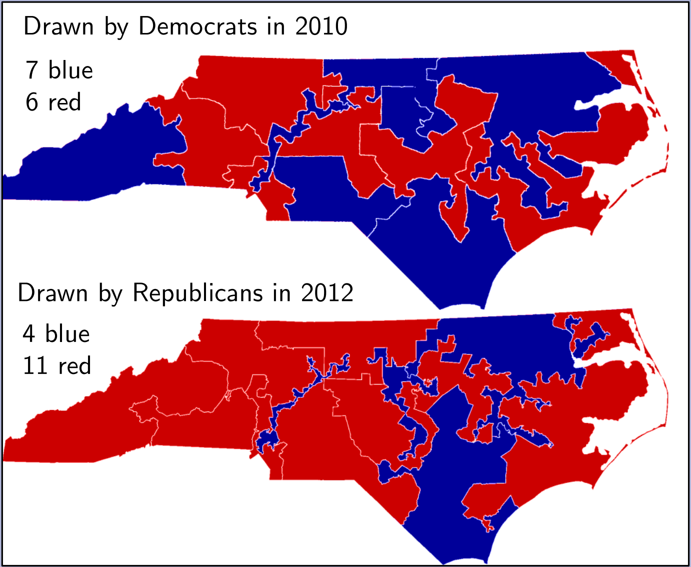
NNC

Geometric Stable Matching



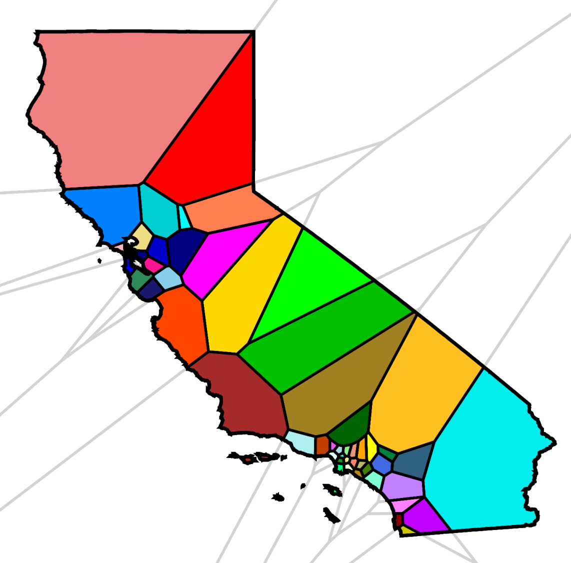
NNC

Gerrymandering



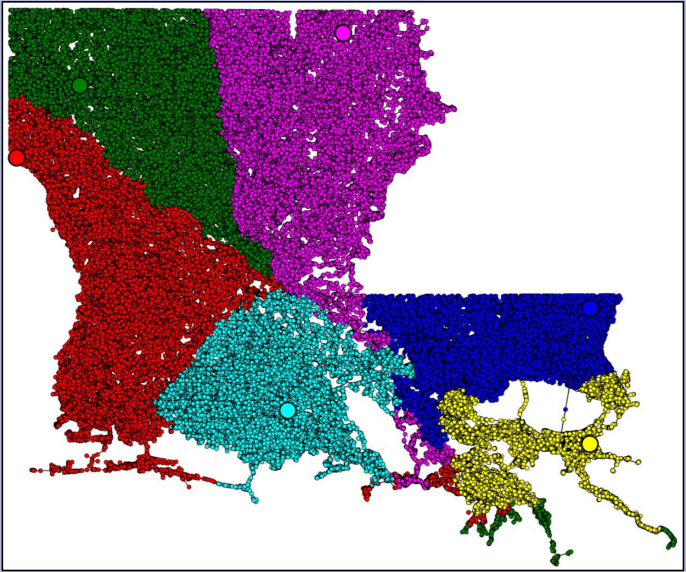
reclaimtheamericandream.org

Algorithmic Districting



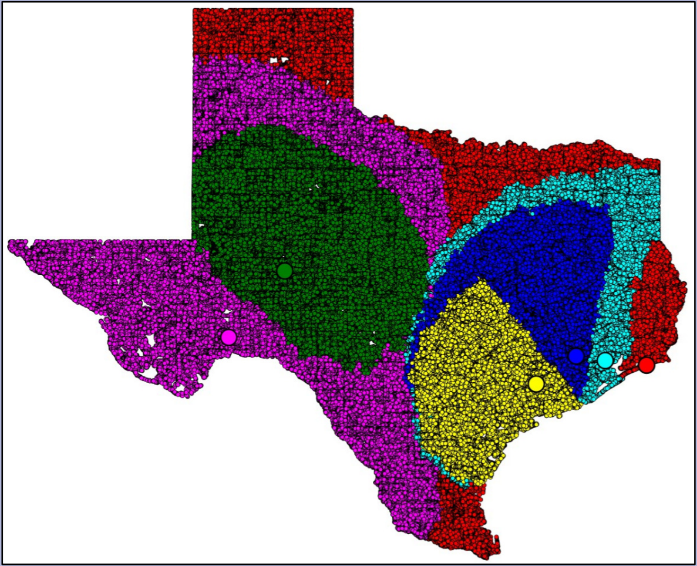
Klein et al. Balanced power diagrams for redistricting

Geometric Stable Matching



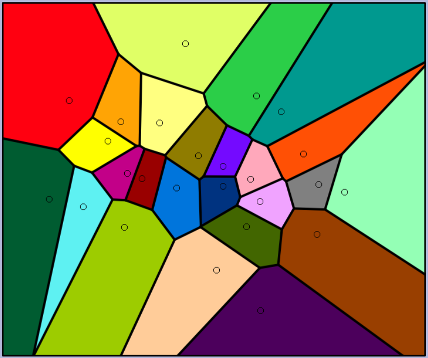
Louisiana road network

Geometric Stable Matching

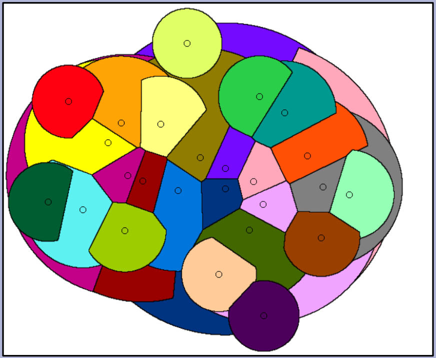


Texas road network

Stable-matching Voronoi Diagram

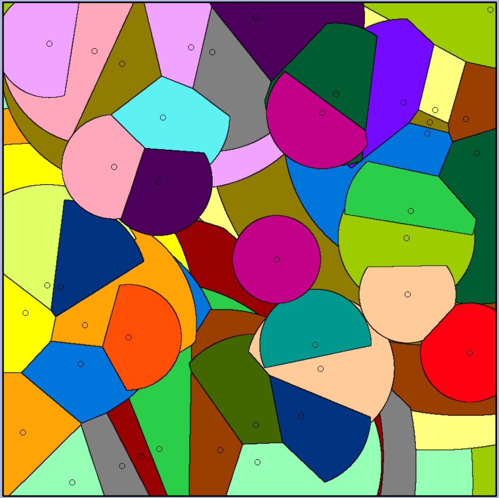


Voronoi diagram

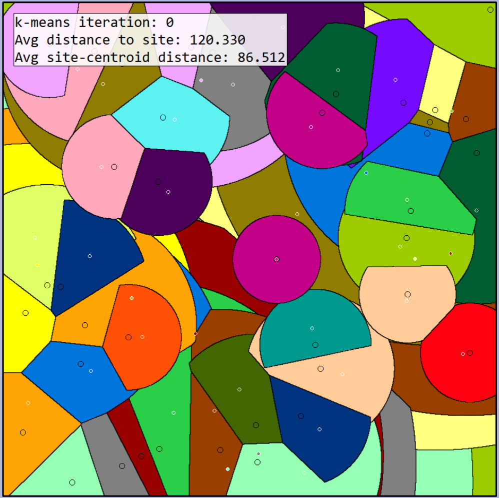


Stable-matching Voronoi diagram

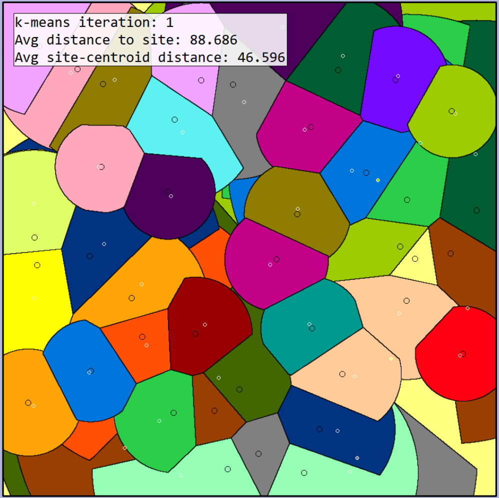
Stable k-means



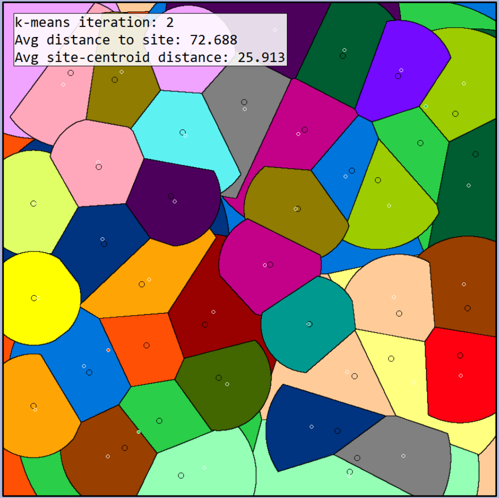
Stable k-means



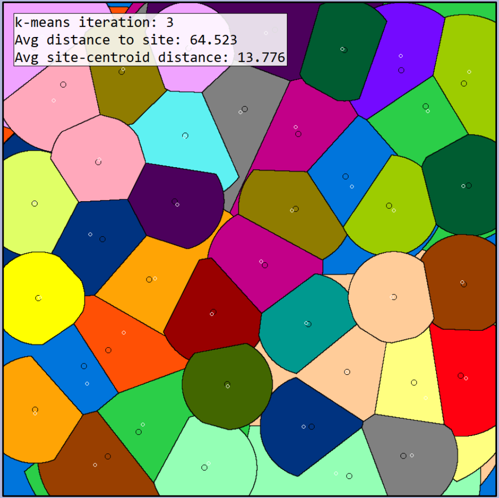
Stable k-means



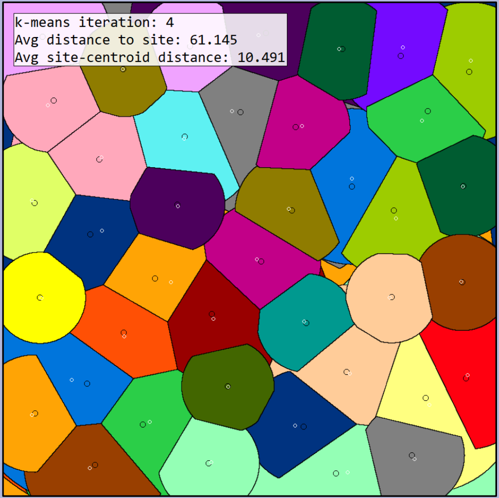
Stable k-means



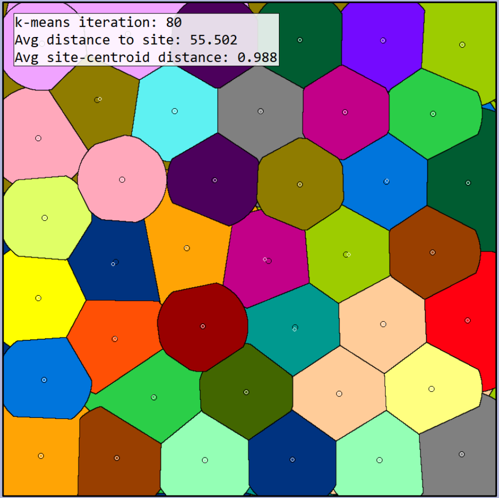
Stable k-means



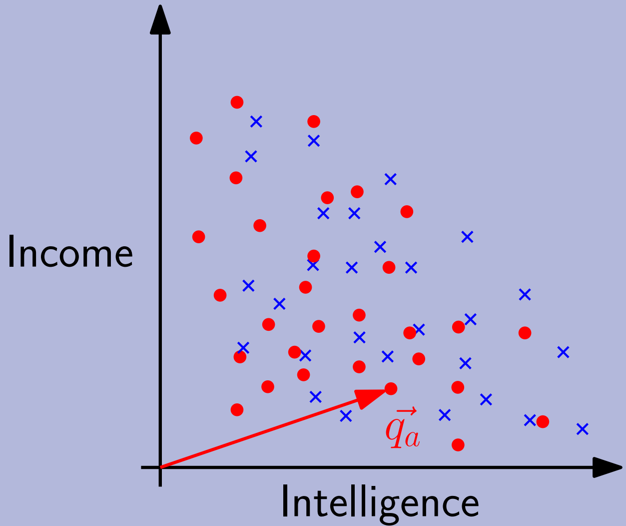
Stable k-means



Stable k-means



k-Attribute Stable Matching



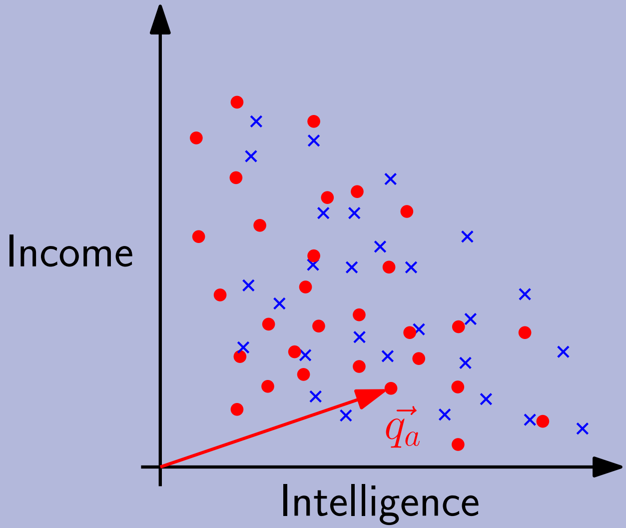
For each person q:

- Attributes: $\vec{q_a}$
- Inclinations: $\vec{q_i}$

score of q for p:

$$\vec{q_i} \cdot \vec{p_a}$$

k-Attribute Stable Matching



For each person q:

- Attributes: $\vec{q_a}$
- Inclinations: $\vec{q_i}$

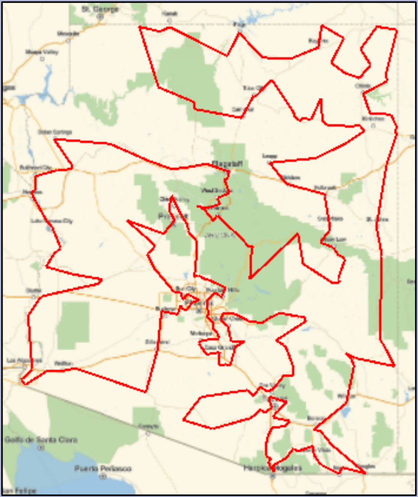
score of q for p:

$$\vec{q_i} \cdot \vec{p_a}$$

Narcissistic case:

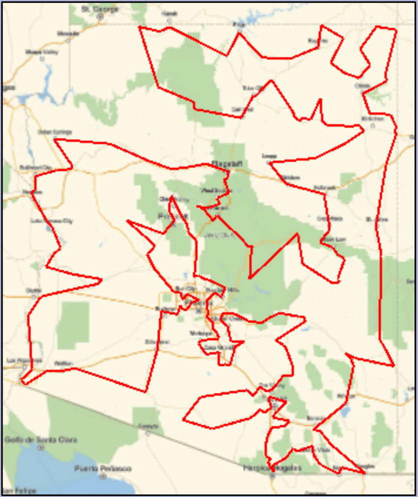
$$\vec{q_a} = \vec{q_i}$$

Euclidean TSP

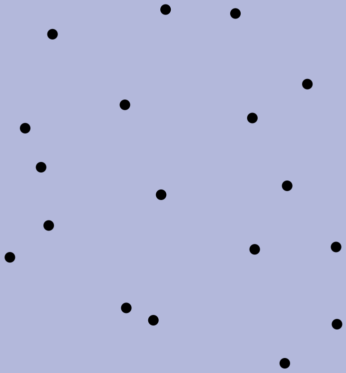


Tour through Arizona's major cities

Euclidean TSP: Multi-fragment greedy

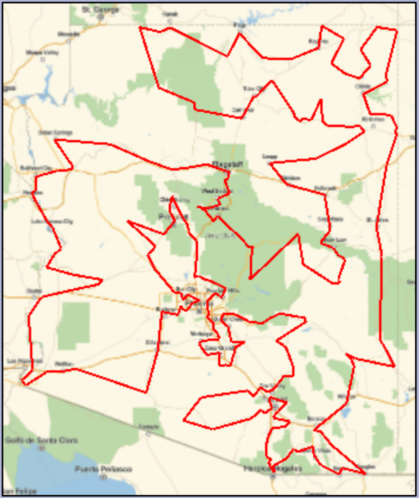


Tour through Arizona's major cities

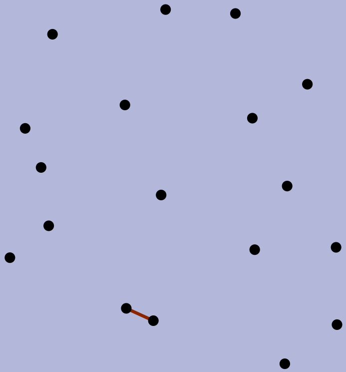


Start with one "path" per point

Euclidean TSP: Multi-fragment greedy

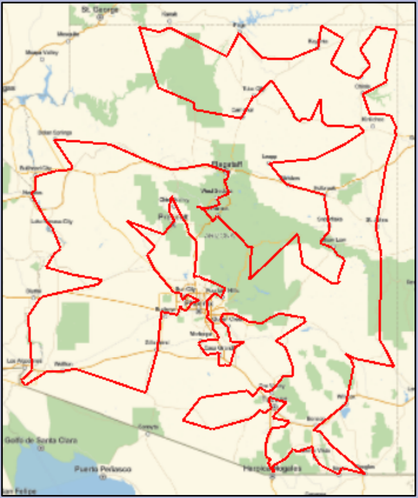


Tour through Arizona's major cities

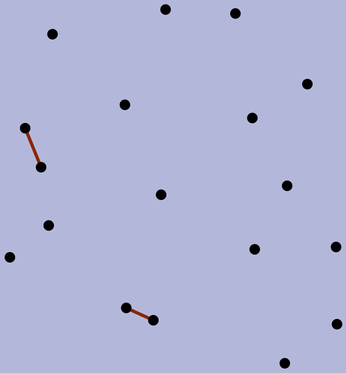


Repeatedly connect the two closest paths

Euclidean TSP: Multi-fragment greedy

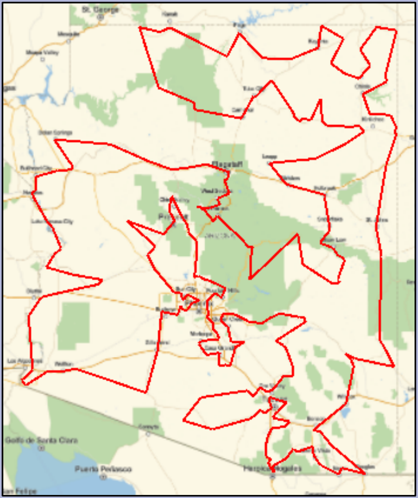


Tour through Arizona's major cities

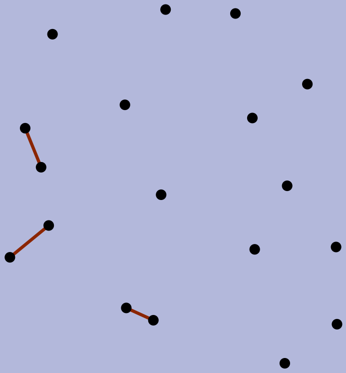


Repeatedly connect the two closest paths

Euclidean TSP: Multi-fragment greedy

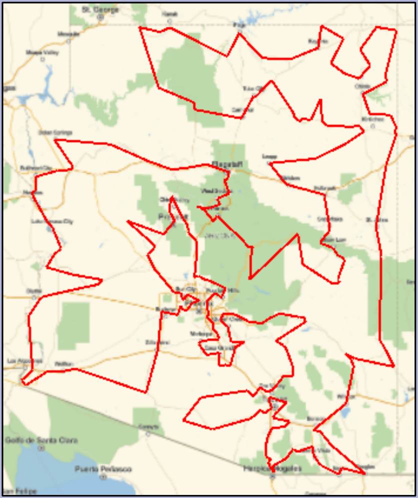


Tour through Arizona's major cities

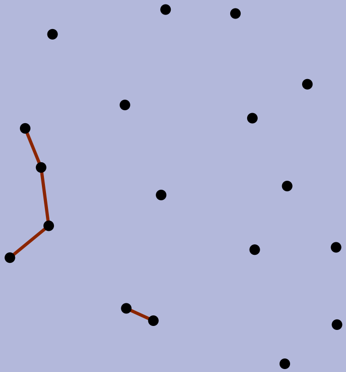


Repeatedly connect the two closest paths

Euclidean TSP: Multi-fragment greedy

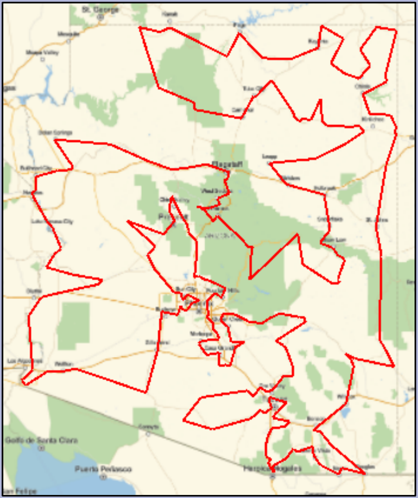


Tour through Arizona's major cities

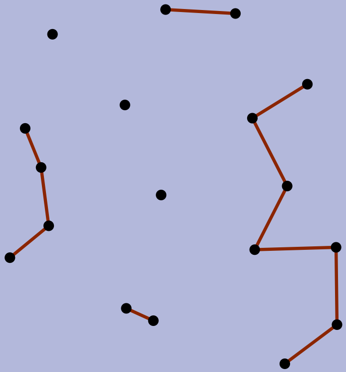


Repeatedly connect the two closest paths

Euclidean TSP: Multi-fragment greedy

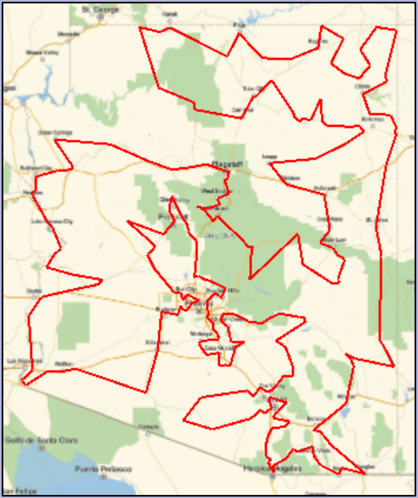


Tour through Arizona's major cities

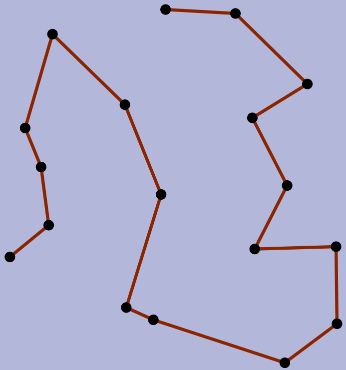


Repeatedly connect the two closest paths

Euclidean TSP: Multi-fragment greedy

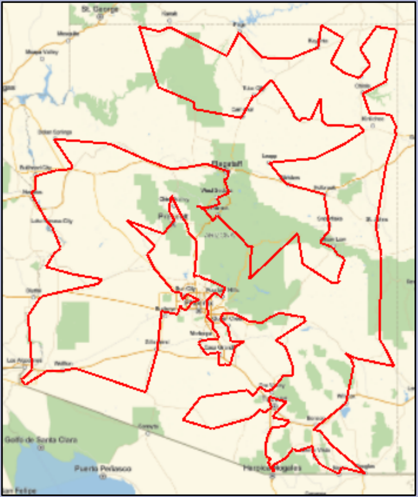


Tour through Arizona's major cities

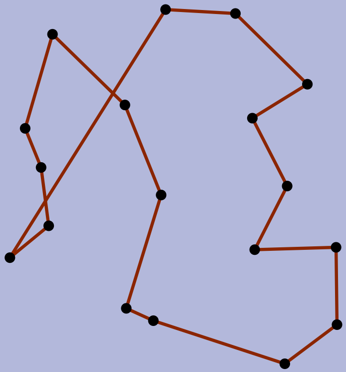


Repeatedly connect the two closest paths

Euclidean TSP: Multi-fragment greedy



Tour through Arizona's major cities

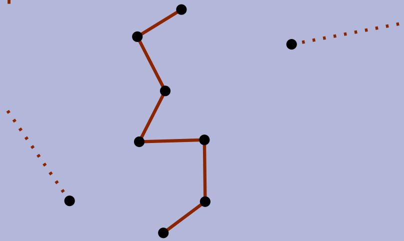


Connect last two endpoints

Euclidean TSP: Multi-fragment greedy

Global–local equivalence: instead of connecting the closest pair of paths, we can connect any pair of MNN

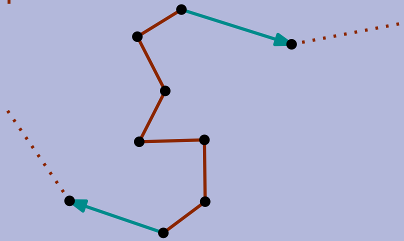
Nearest-neighbor chain: to find the NN of a path, do a NN query from each endpoint



Euclidean TSP: Multi-fragment greedy

Global–local equivalence: instead of connecting the closest pair of paths, we can connect any pair of MNN

Nearest-neighbor chain: to find the NN of a path, do a NN query from each endpoint



Shortest Superstring

`TGTATCGCAGACTGGATAAAACATCAAAAAGGAGGACACATGCTCCTCGA`

Shortest Superstring

TGTATCGCAGACTGGATAAAAC|ATCAAAAAGG|AGGACACATGCTCCTCGA

Sampling process:

ATCAAAAAGG

Shortest Superstring

TGTATCGCAGACTGGATAAAACATCAAAAAGGAGGACACATGCTCCTCGA

ATCAAAAAGG

GCTC

ATAAAACATC

ACATCAAAAA

CATGCT

CAAAAAGGAGG

TGTATCGCAGAC

ATCGCAGACT

GAGGA

CTGGATAA

TGGATAAA

GGATAAAACA

CAGACTGGA

ACTGGATA

CGCAGACTG

AGGACACATGC

AACATCA

Shortest Superstring

TGTATCGCAGACTGGATAAAACATCAAAAAGGAGGACACATGCTCCTCGA

"closest" pair

ATCAAAAAGG

GCTC

ATAAAACATC

ACATCAAAAA

CATGCT

CAAAAAGGAGG

TGTATCGCAGAC

ATCGCAGACT

GAGGA

CTGGATAA

TGGATAAA

GGATAAAACA

CAGACTGGA

ACTGGATA

CGCAGACTG

AGGACACATGC

AACATCA

Shortest Superstring

TGTATCGCAGACTGGATAAAACATCAAAAAGGAGGACACATGCTCCTCGA

mutual "nearest neighbors"

ATCAAAAAGG

GCTC

ATAAAACATC

ACATCAAAAA

CATGCT

CAAAAAGGAGG

TGTATCGCAGAC

ATCGCAGACT

GAGGA

CTGGATAA

TGGATAAA

GGATAAAACA

CAGACTGGA

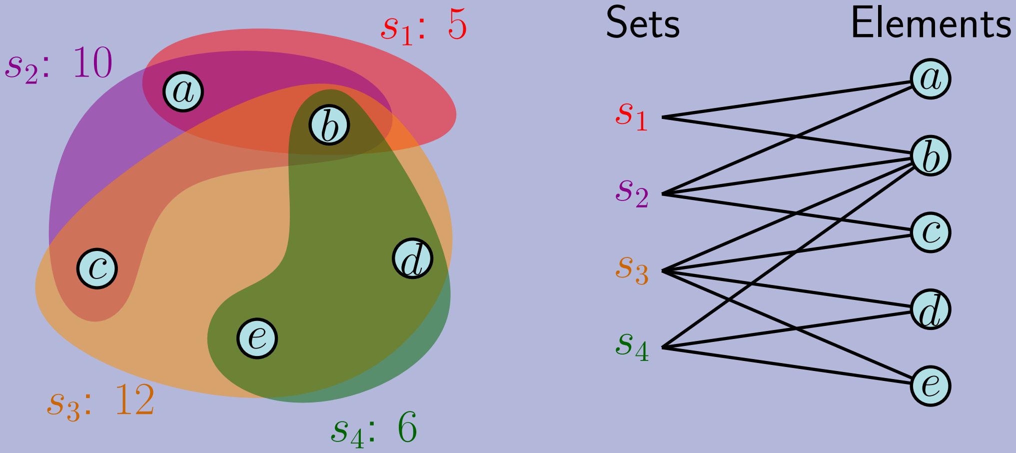
ACTGGATA

CGCAGACTG

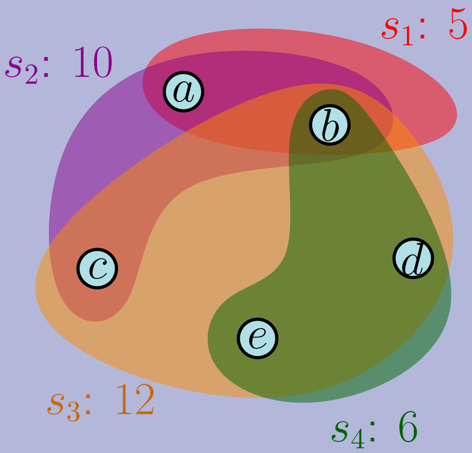
AGGACACATGC

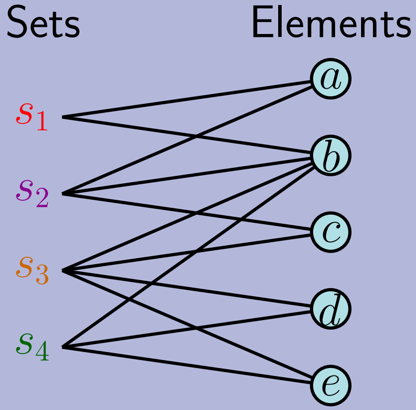
AACATCA

Set Cover



Set Cover

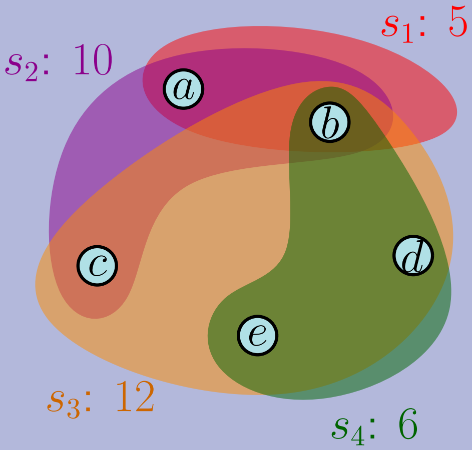


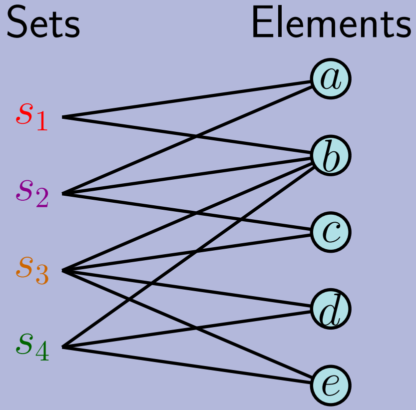


$$\text{cost-per-elem}(\mathsf{s}_i) = \frac{\text{weight}(\mathsf{s}_i)}{\#\text{uncovered in } \mathsf{s}_i}$$

Greedy: always pick set with smallest cost-per-elem

Set Cover





$$\text{cost-per-elem}(s_i) = \frac{\text{weight}(s_i)}{\#\text{uncovered in } s_i}$$

Local Greedy: pick any set with a smaller cost-per-elem than any set with a common element

Combinatorial Problems

Global—Local Equivalence

- Set cover ✓
  - Vertex cover ✓
- Dominating set ✓
- Matching ✓
- Independent set ✗

G–L Equivalence Proof

Hybrid$_i$:	l_0	$\cdots$	l_{i-1}	g_i	$\cdots$	g_n
		Local Greedy			Greedy	

Hybrid$_n$: Local Greedy

Hybrid$_0$: Greedy

We show:

$$\text{Hybrid}_i = \text{Hybrid}_{i+1}$$

D. Müllner, "Modern hierarchical, agglomerative clustering algorithms,"

G–L Equivalence Proof

MNN

Hybrid$_{i+1}$

l_i

$g_{i+1} \quad \cdots \quad g_n$

$l_0 \quad \cdots \quad l_{i-1}$

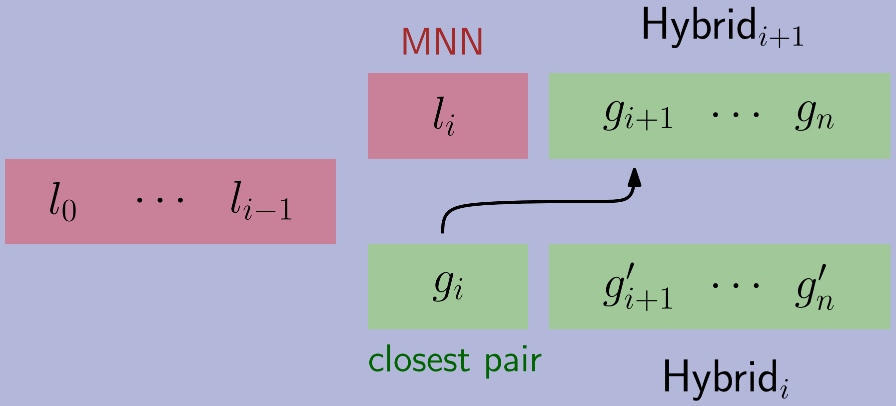
g_i

$g'_{i+1} \quad \cdots \quad g'_n$

closest pair

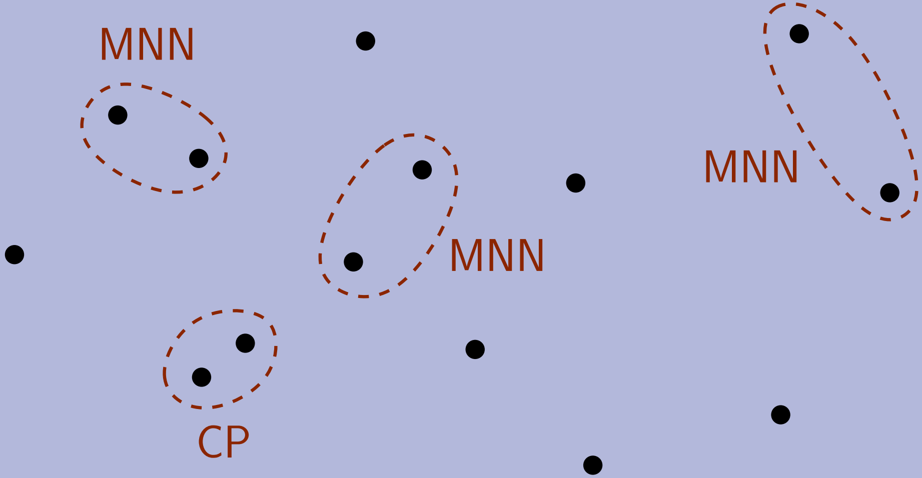
Hybrid$_i$

G–L Equivalence Proof



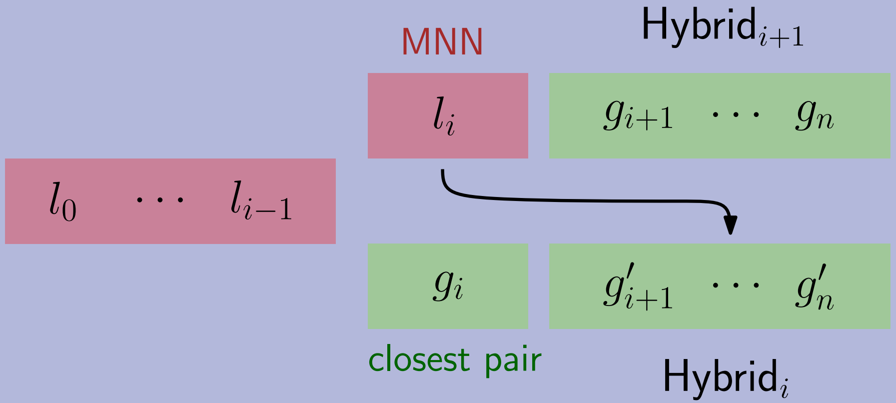
1. The CP *remains* CP even if other MNN are picked

G–L Equivalence Proof



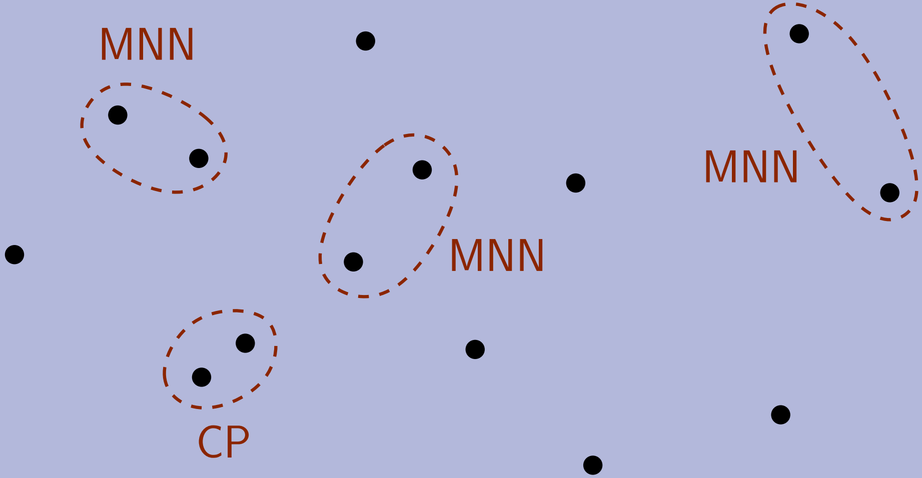
1. The CP *remains* CP even if other MNN are picked

G–L Equivalence Proof



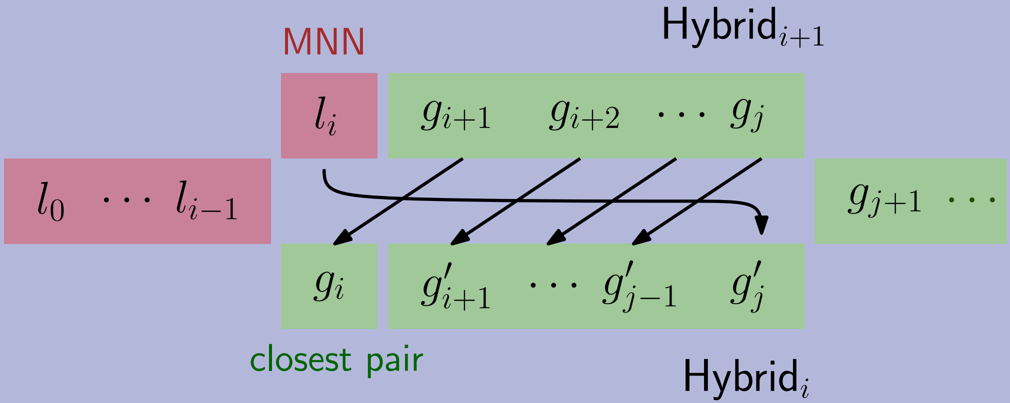
2. MNN *stay* MNN until picked

G–L Equivalence Proof

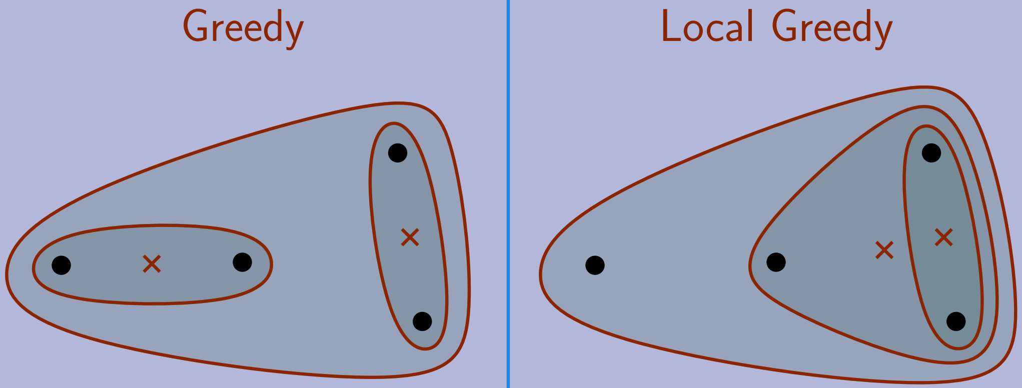


2. MNN *stay* MNN until picked

G–L Equivalence Proof

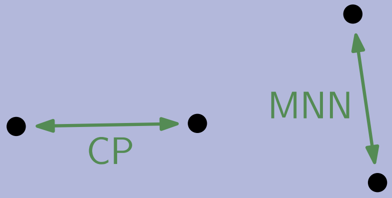


Clustering with Centroid Distance



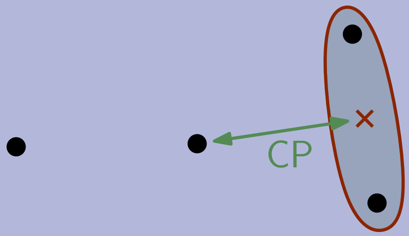
Clustering with Centroid Distance

GLE fails:



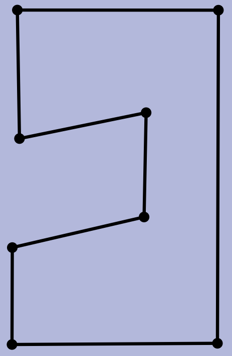
Clustering with Centroid Distance

GLE fails:

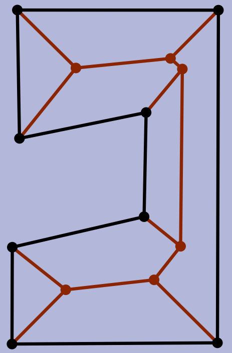


CP did not remain CP

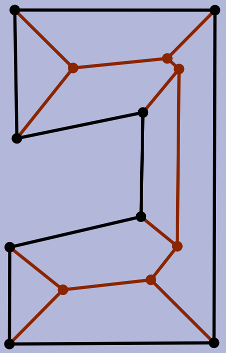
Straight Skeletons

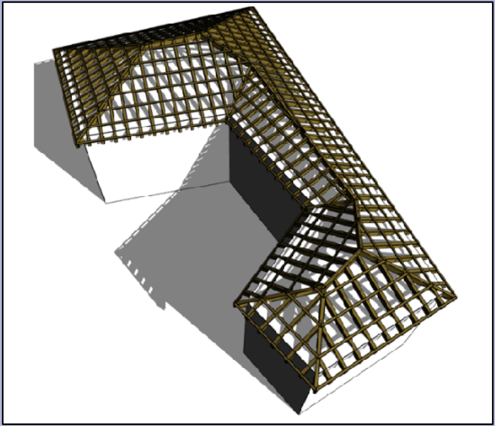


Straight Skeletons



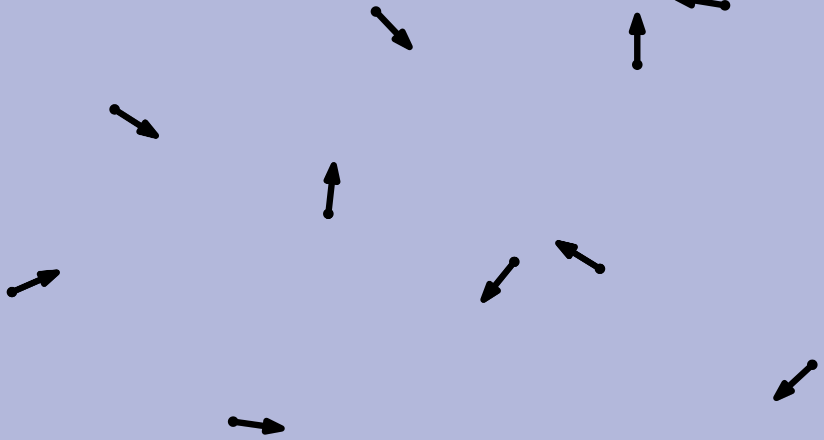
Straight Skeletons





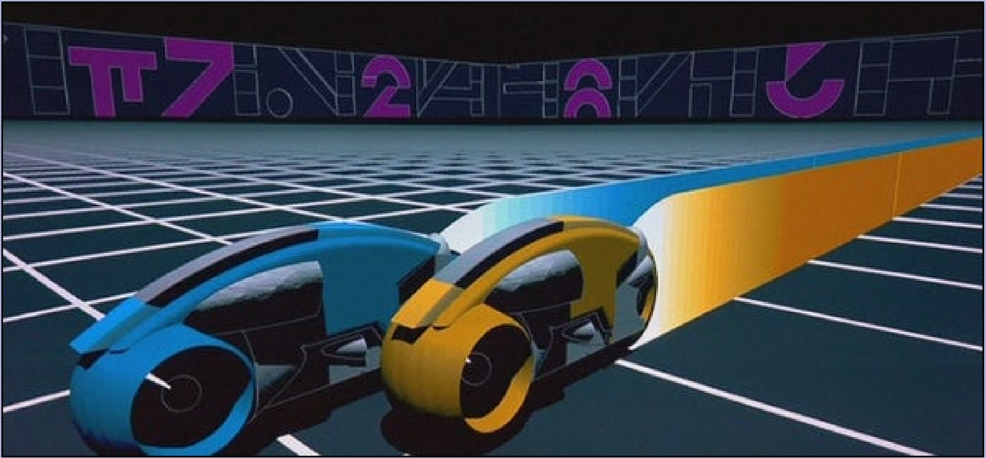
© 2019 GeometryFactory.com

Motorcycle Graphs

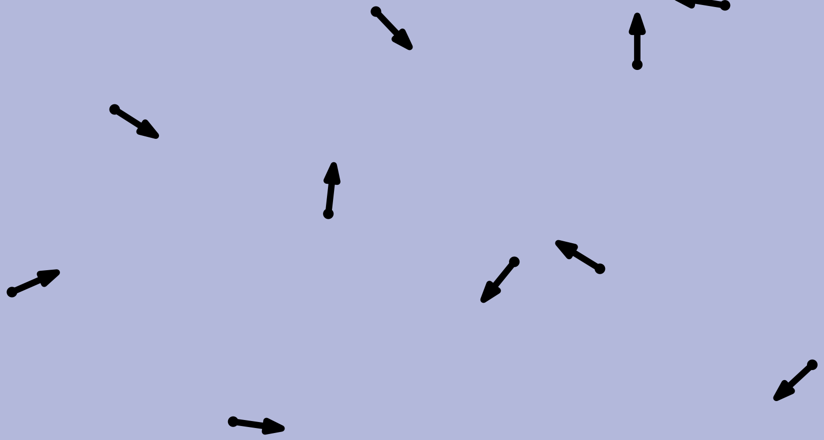


D. Eppstein, J. Erickson, “Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions,” 1998

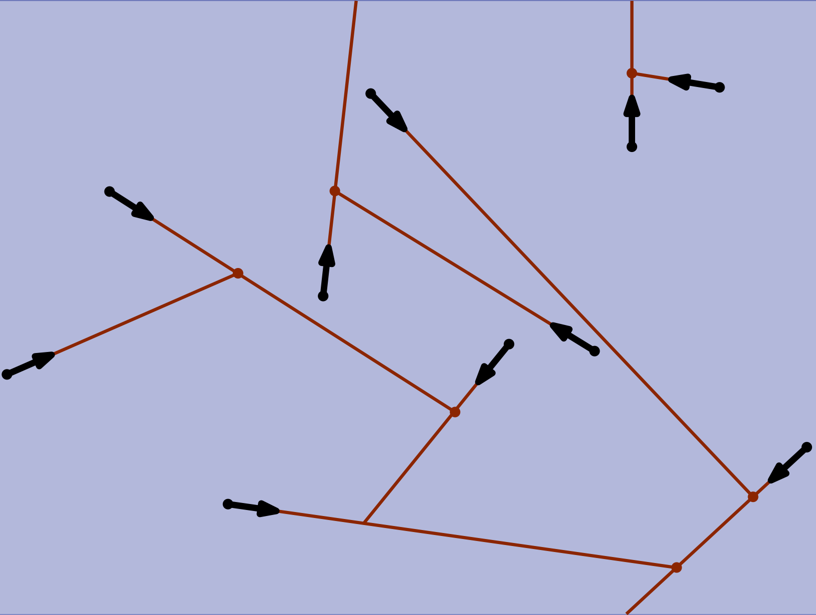
Motorcycle Graphs



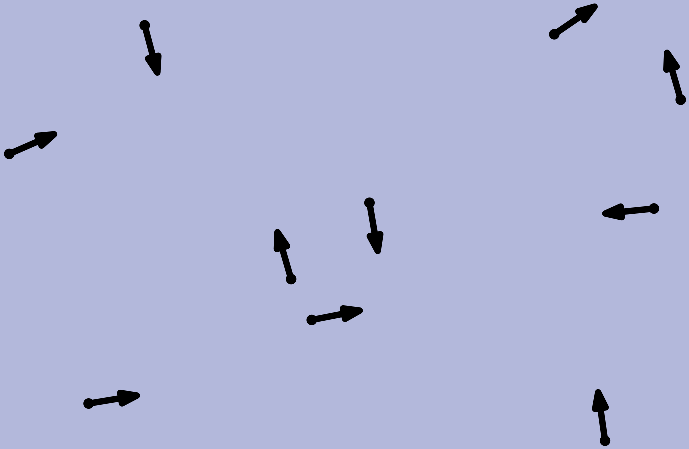
Motorcycle Graphs



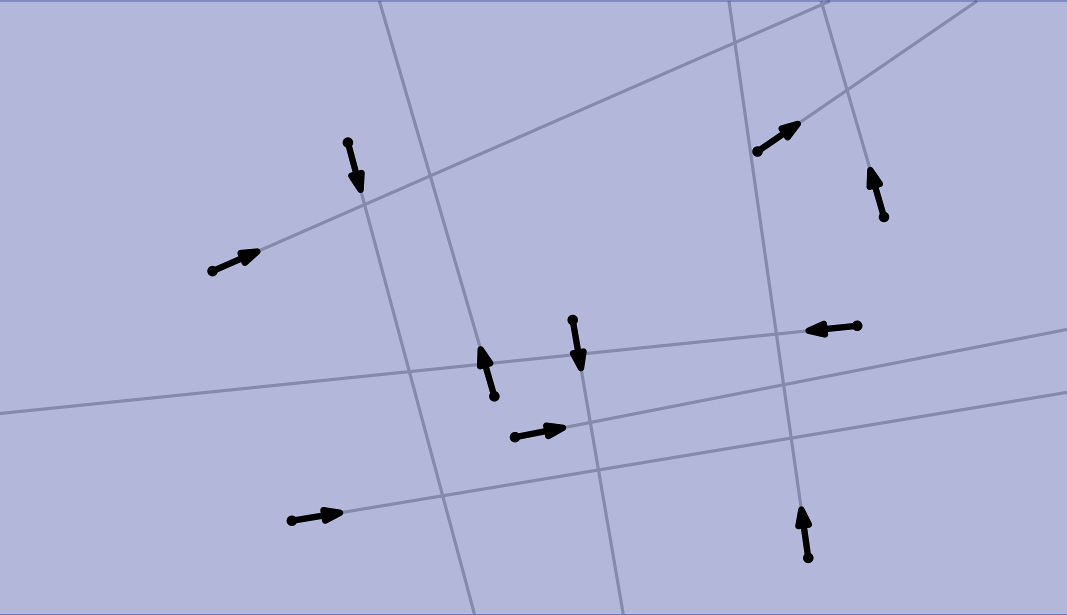
Motorcycle Graphs



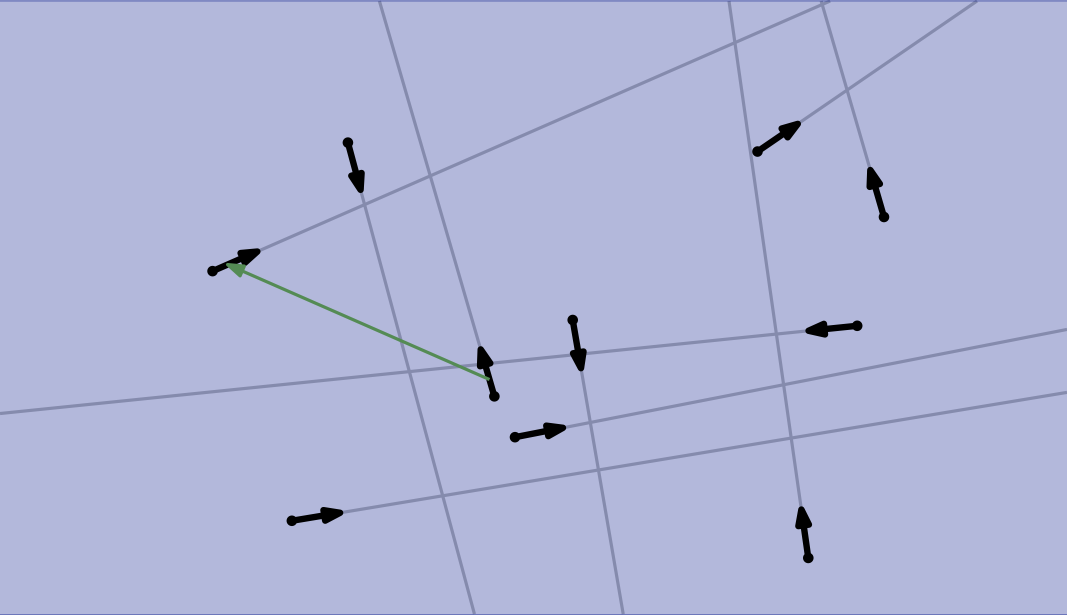
Motorcycle Graphs



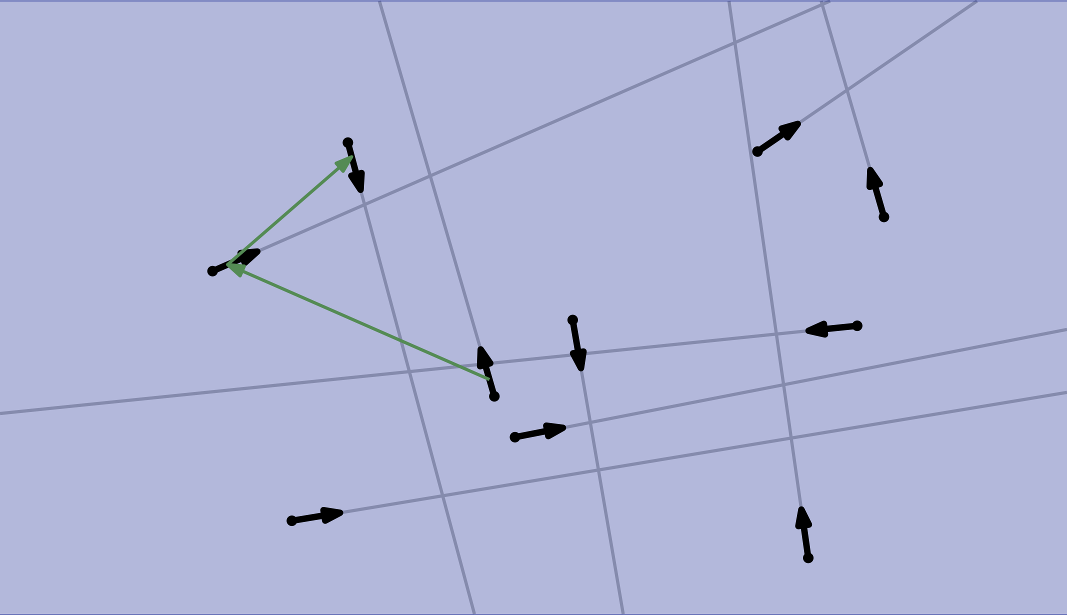
Motorcycle Graphs



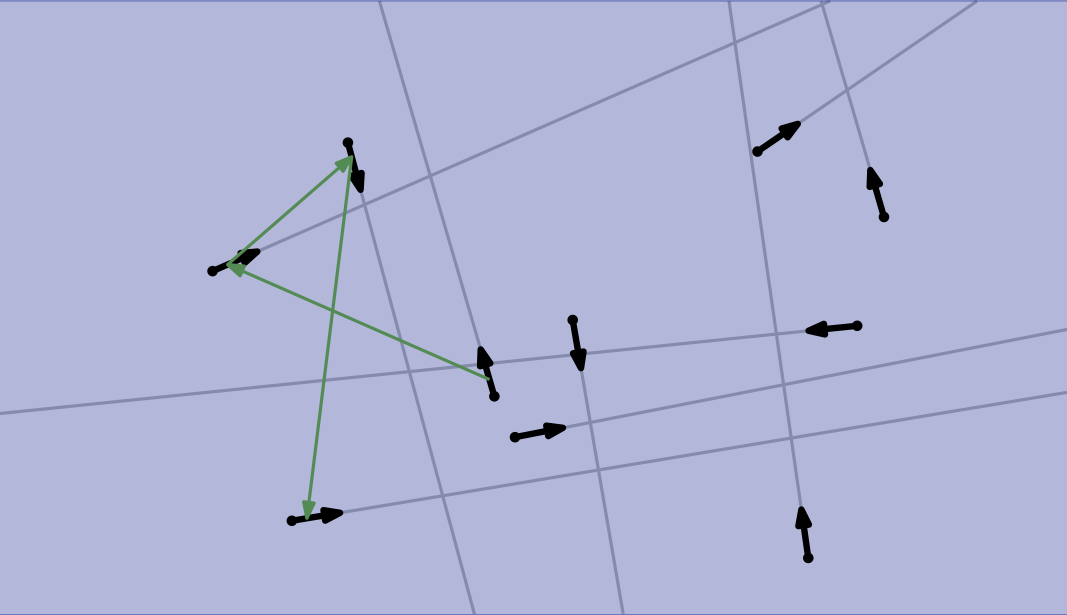
Motorcycle Graphs



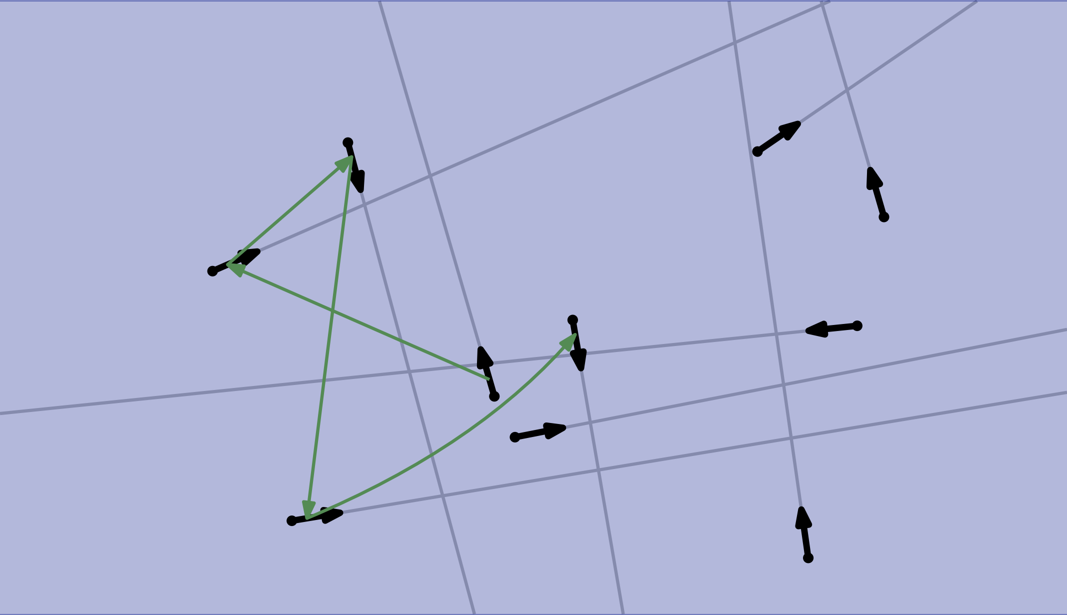
Motorcycle Graphs



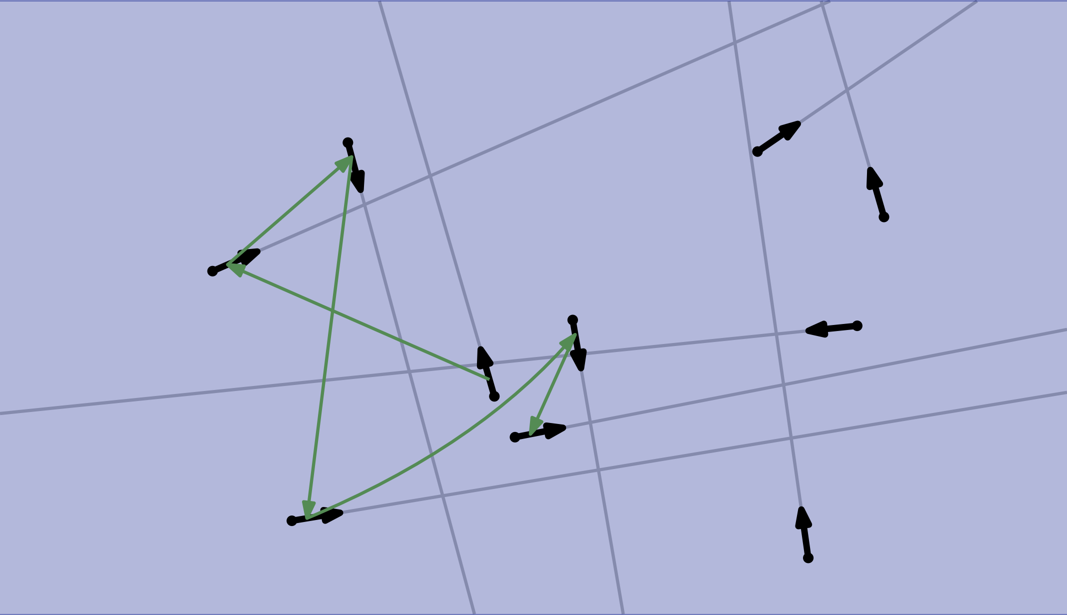
Motorcycle Graphs



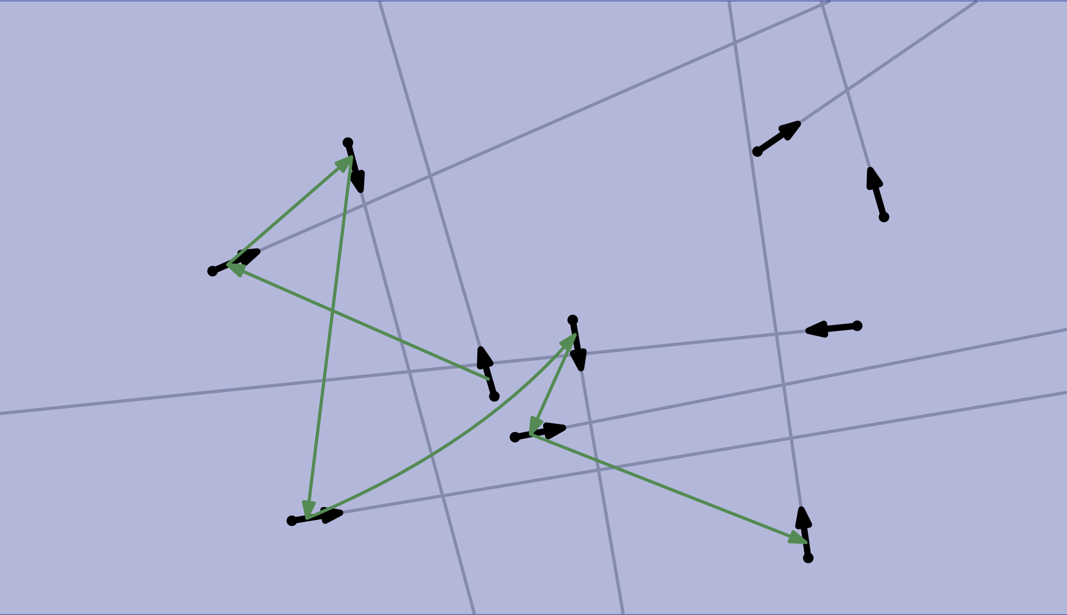
Motorcycle Graphs



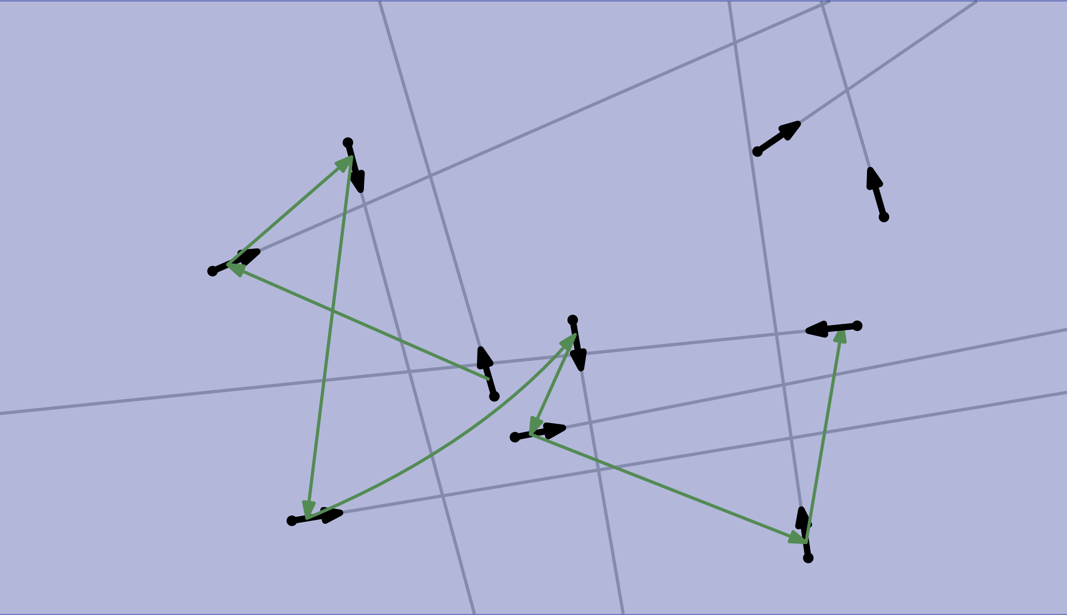
Motorcycle Graphs



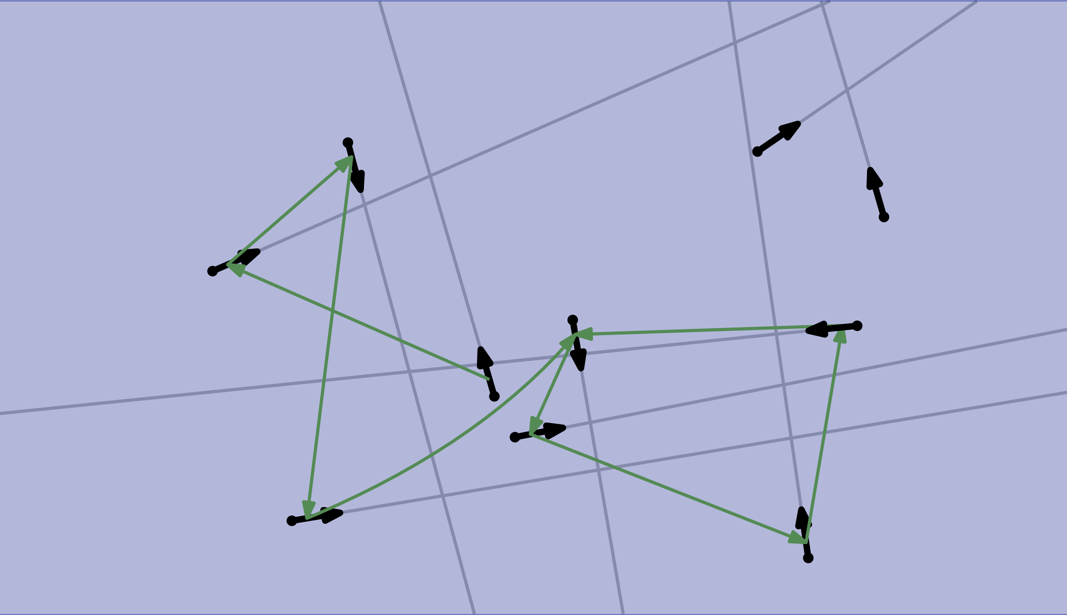
Motorcycle Graphs



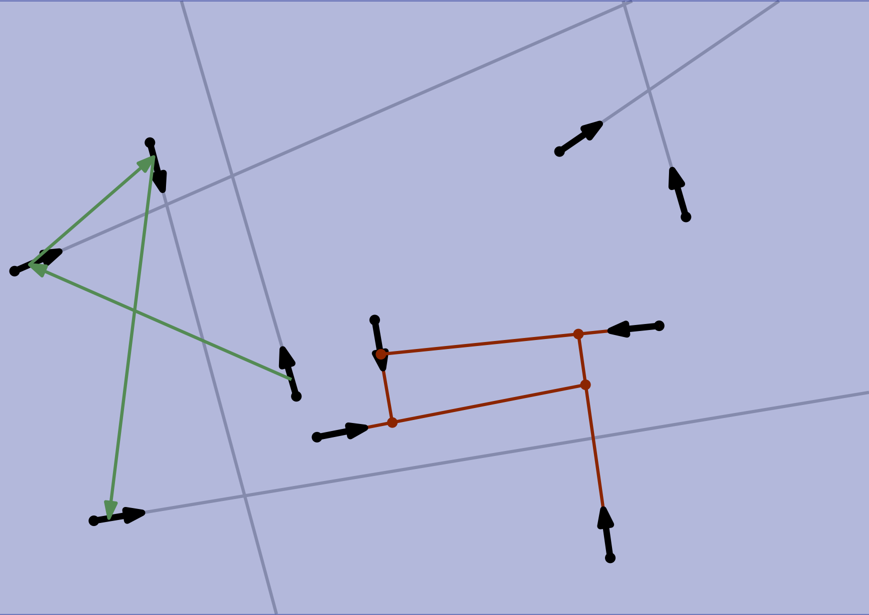
Motorcycle Graphs



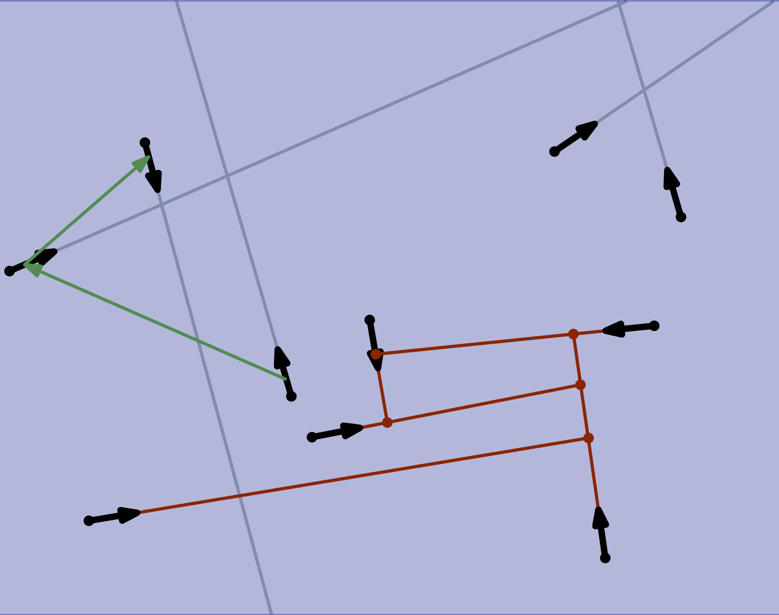
Motorcycle Graphs



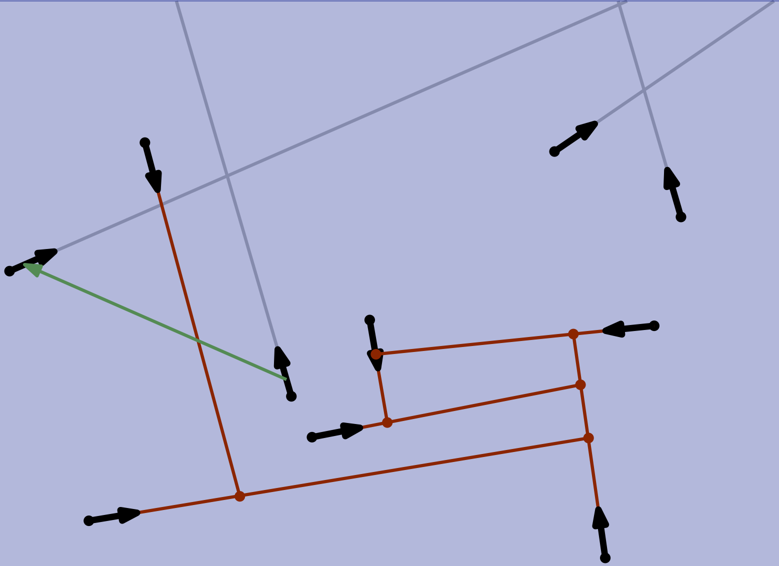
Motorcycle Graphs



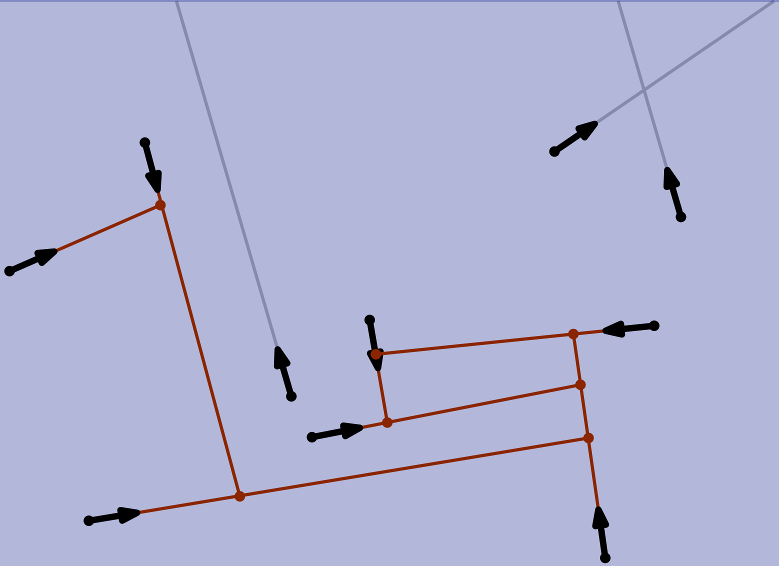
Motorcycle Graphs



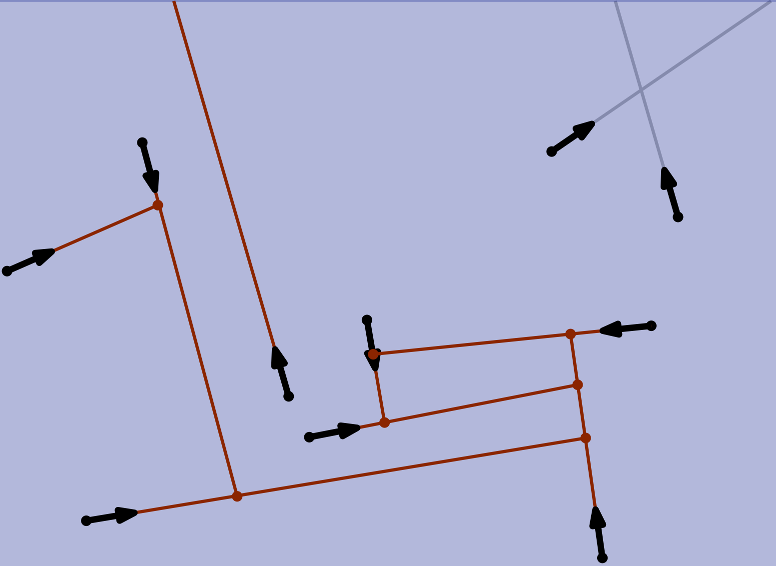
Motorcycle Graphs



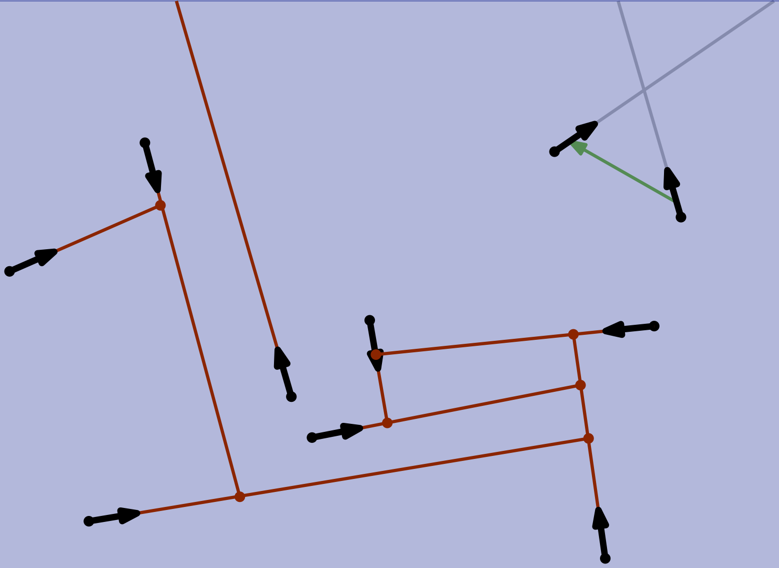
Motorcycle Graphs



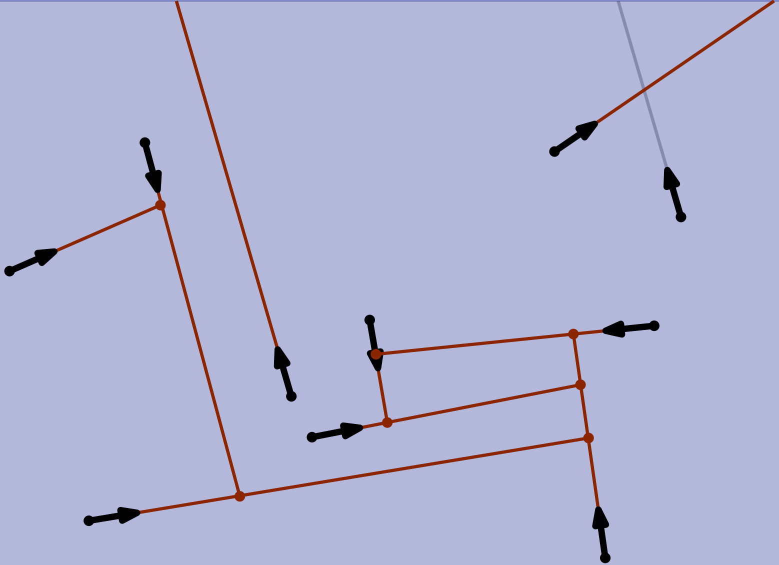
Motorcycle Graphs



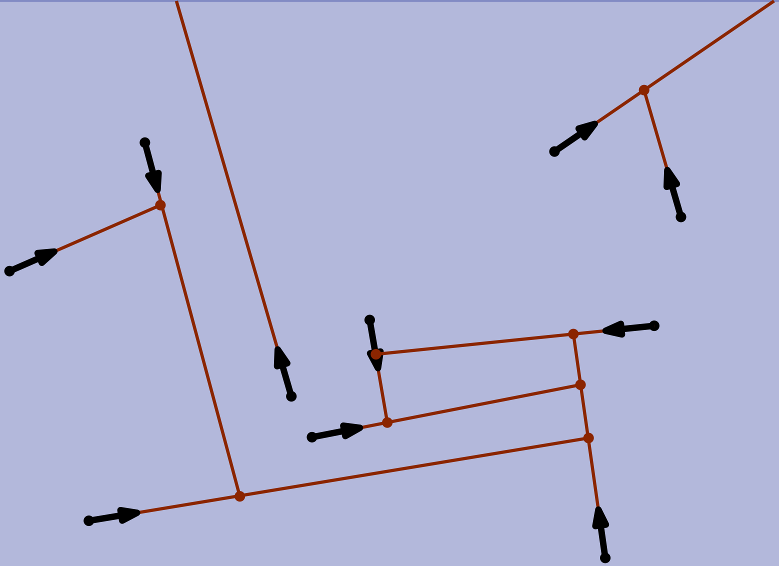
Motorcycle Graphs



Motorcycle Graphs



Motorcycle Graphs



New Results

	Prior Greedy	NNC
Euclidean TSP:	$O(n^2)$	$O(n \log n)$
Steiner TSP in planar graphs:	$O(nk)$	$O(n \ \min(k, \sqrt{n} \log n))$
Motorcycle graphs:	$O(n^{4/3+\varepsilon} \log n)$	$O(n^{4/3+\varepsilon})$
1D radio tower coverage:	$O(n \log n)$	$O(n)$
Narcissistic k-attribute SM:	$O(n^2)$	$O(n^{2-2/(1.1+k/2)})$
Geometric Stable Matching:	$\cancel{O(n \log^7 n)}$ $O(n \log^4 n)$	$\cancel{O(n \log^5 n)}$ $O(n \log^4 n)$

Research directions

Find more greedy algorithms with global–local equivalence

Find other ways to exploit GLE (besides NN chains)

Improve the "nearest neighbor" data structures

(improve the $T(n)$ in $O(nT(n))$)

Papers

NM, A. Efrat, D. Eppstein, D. Frishberg, M. Goodrich, S. Kobourov, P. Matias, V. Polishchuk, “New Applications of Nearest-Neighbor Chains: Euclidean TSP and Motorcycle Graphs” ISAAC’19

G. Barequet, D. Eppstein, M.T. Goodrich, and NM, “Stable-Matching Voronoi Diagrams: Combinatorial Complexity and Algorithms,” ICALP’18

D. Eppstein, M.T. Goodrich, and NM, “Reactive Proximity Data Structures for Graphs,” LATIN’18

D. Eppstein, M.T. Goodrich, D. Korkmaz, and NM, “Defining Equitable Geographic Districts in Road Networks via Stable Matching,” SIGSPATIAL’17

D. Eppstein, M.T. Goodrich, and NM, “Algorithms for Stable Matching and Clustering in a Grid,” IWCIA’17

Papers

NM, A. Efrat, D. Eppstein, D. Frishberg, M. Goodrich, S. Kobourov, P. Matias, V. Polishchuk, "New Applications of Nearest-Neighbor Chains: Euclidean TSP and Motorcycle Graphs" ISAAC'19

G. Barequet, D. Eppstein, M.T. Goodrich, and NM, "Stable-Matching Voronoi Diagrams: Combinatorial Complexity and Algorithms," ICALP'18

D. Eppstein, M.T. Goodrich, and NM, "Reactive Proximity Data Structures for Graphs," LATIN'18

D. Eppstein, M.T. Goodrich, D. Korkmaz, and NM, "Defining Equitable Geographic Districts in Road Networks via Stable Matching," SIGSPATIAL'17

D. Eppstein, M.T. Goodrich, and NM, "Algorithms for Stable Matching and Clustering in a Grid," IWCIA'17

Thank you!