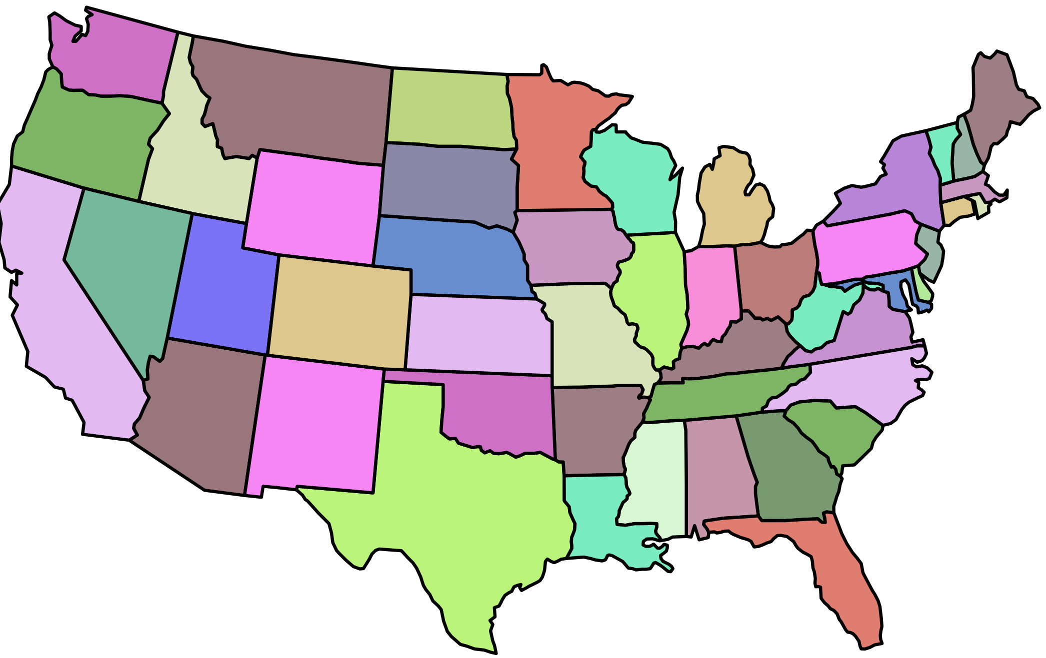
Improved Grid Map Layout by Point Set Matching

David Eppstein
Marc van Kreveld
Bettina Speckmann
Frank Staals

University of California, Irvine,
Utrecht University, TU Eindhoven

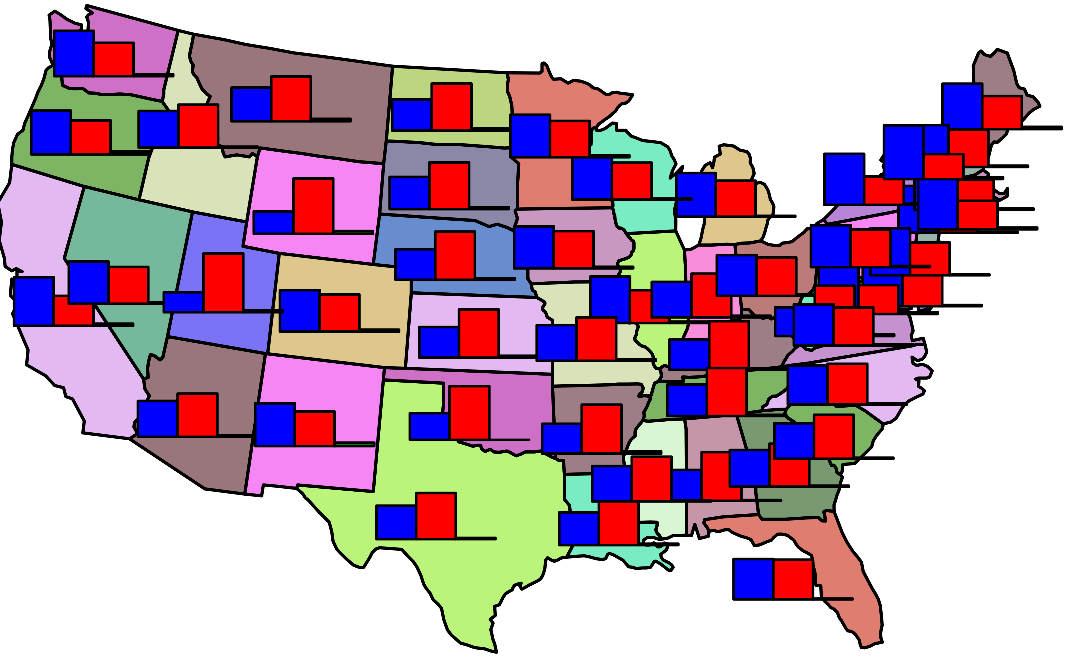
Visualising Geographic Data

Given a map with n regions we want to visualise some data for each region.



Visualising Geographic Data

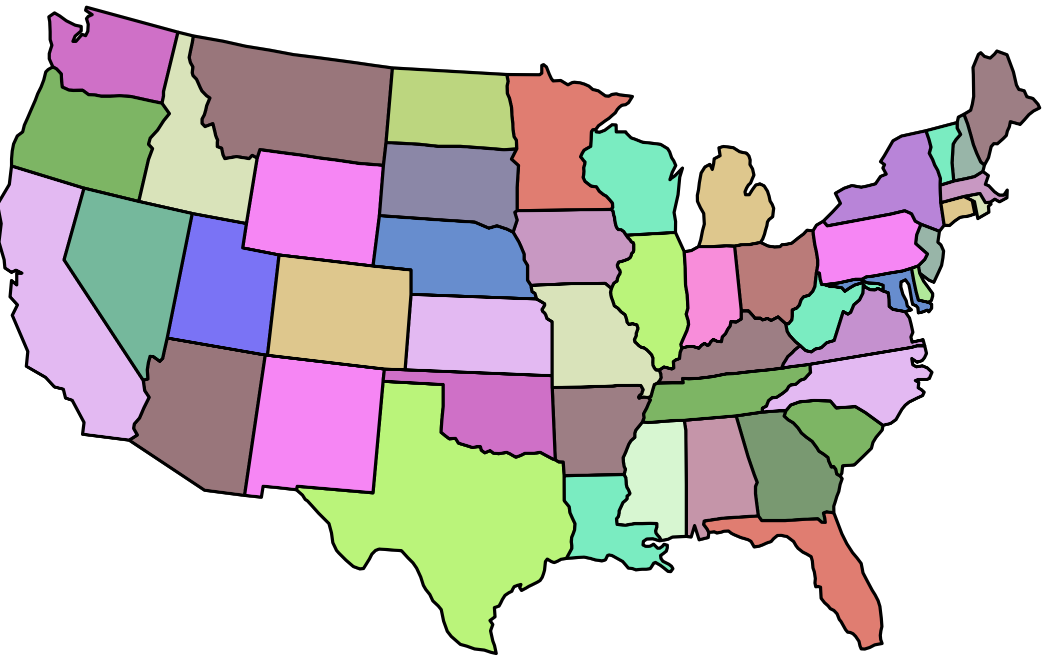
Given a map with n regions we want to visualise some data for each region. e.g. US Presidential Elections

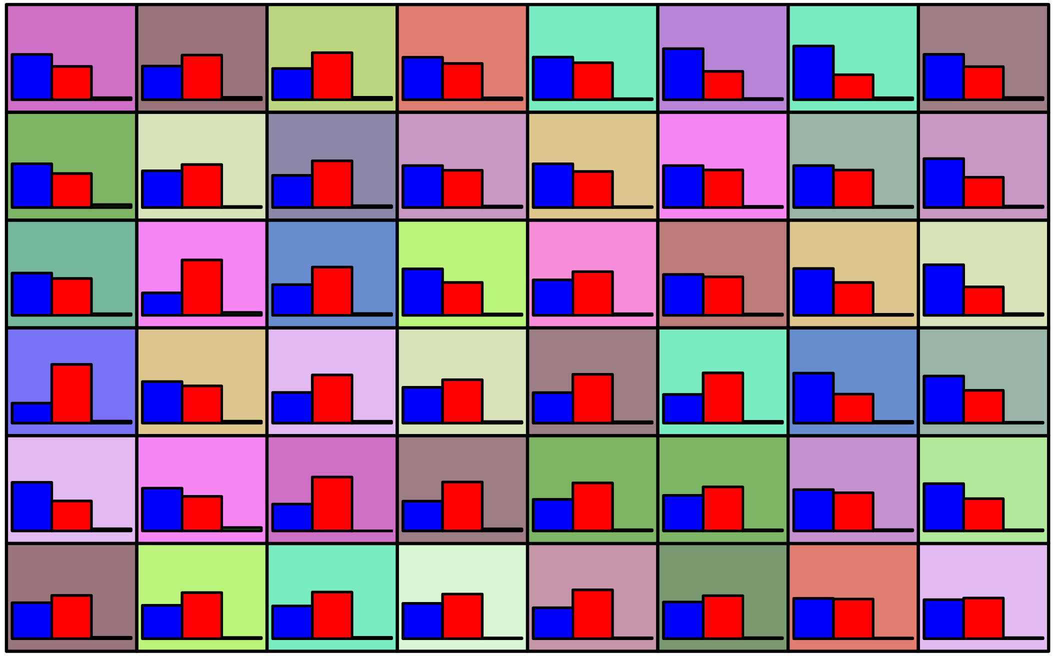


Problem: Visual Clutter

Visualising Geographic Data

Given a map with n regions we want to visualise some data for each region.

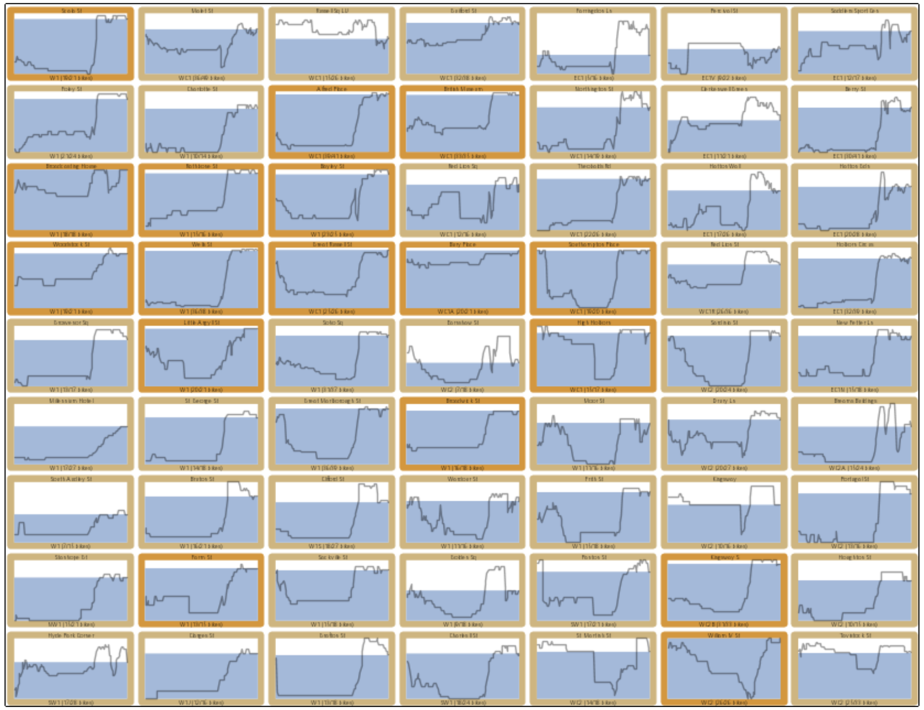




Idea: Use a Grid Map

Visualising Geographic Data

Given a map with n regions we want to visualise some data for each region.

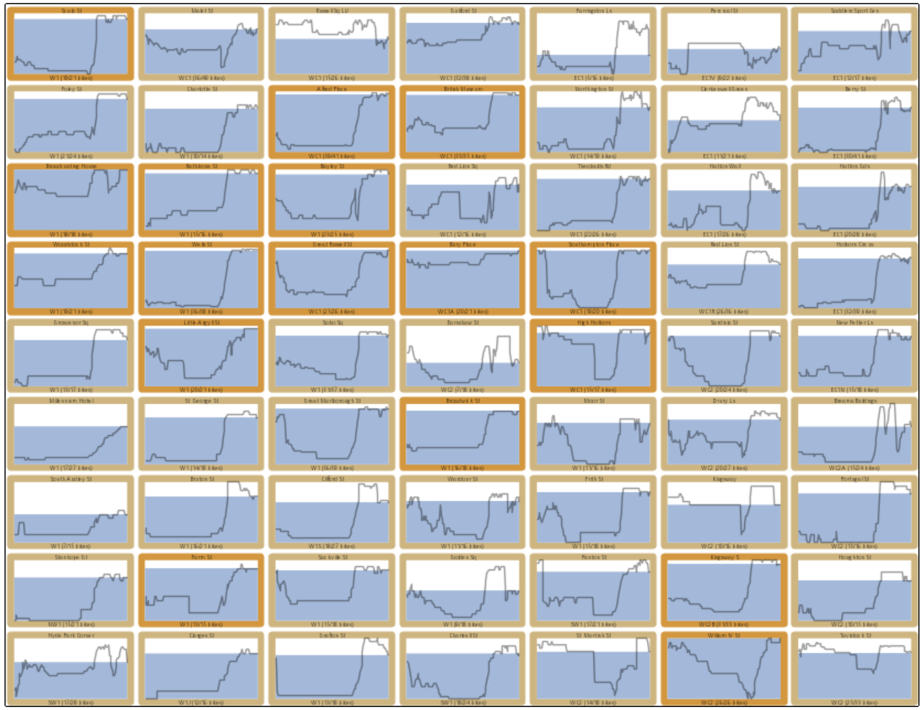


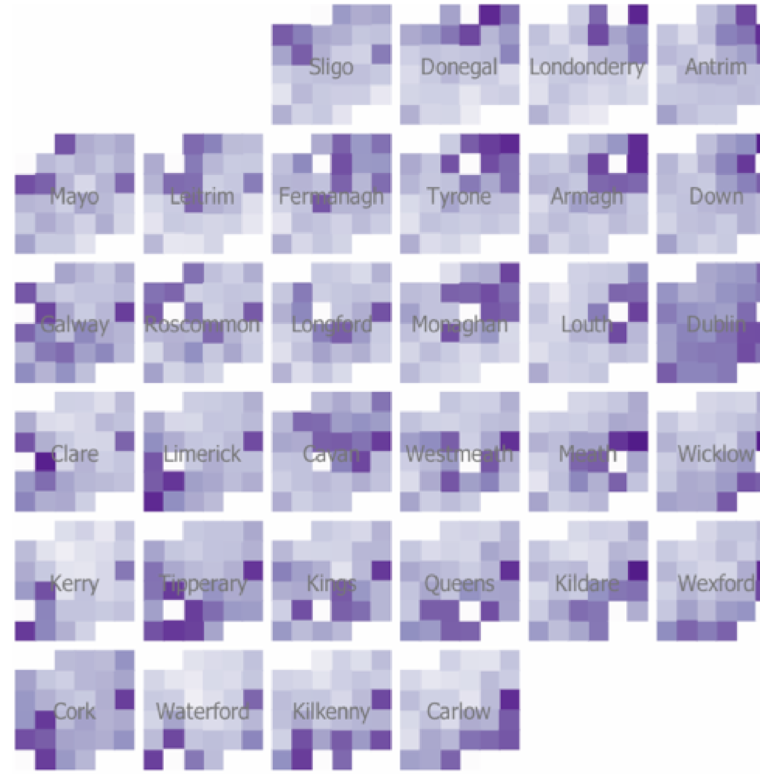
Idea: Use a Grid Map

- London BikeGrid: `gicentre.org/bikegrid`

Visualising Geographic Data

Given a map with n regions we want to visualise some data for each region.





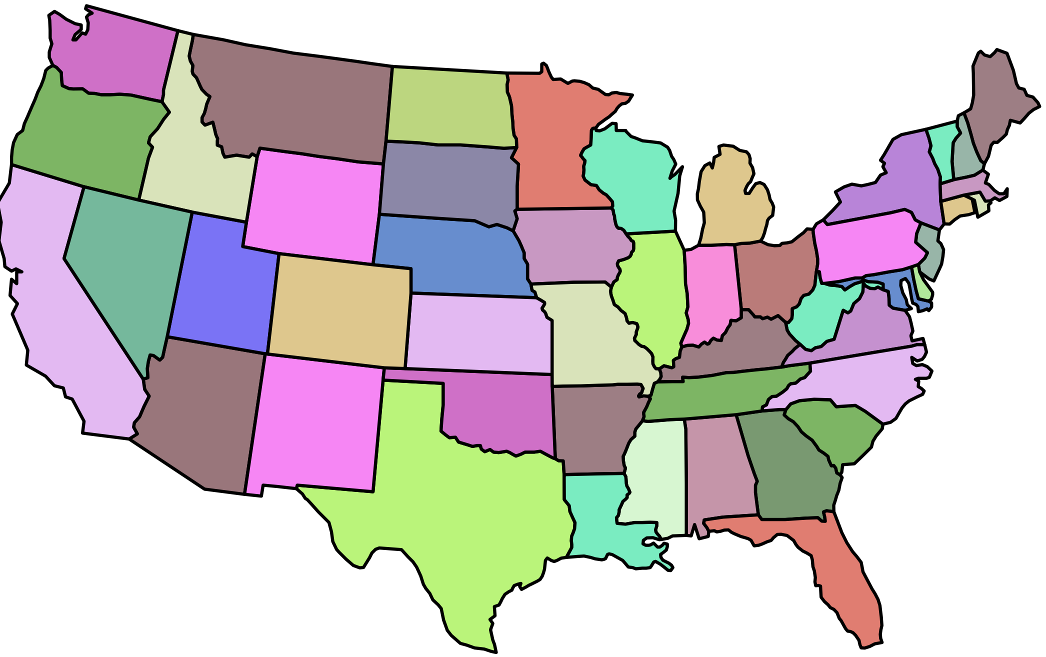
Idea: Use a Grid Map

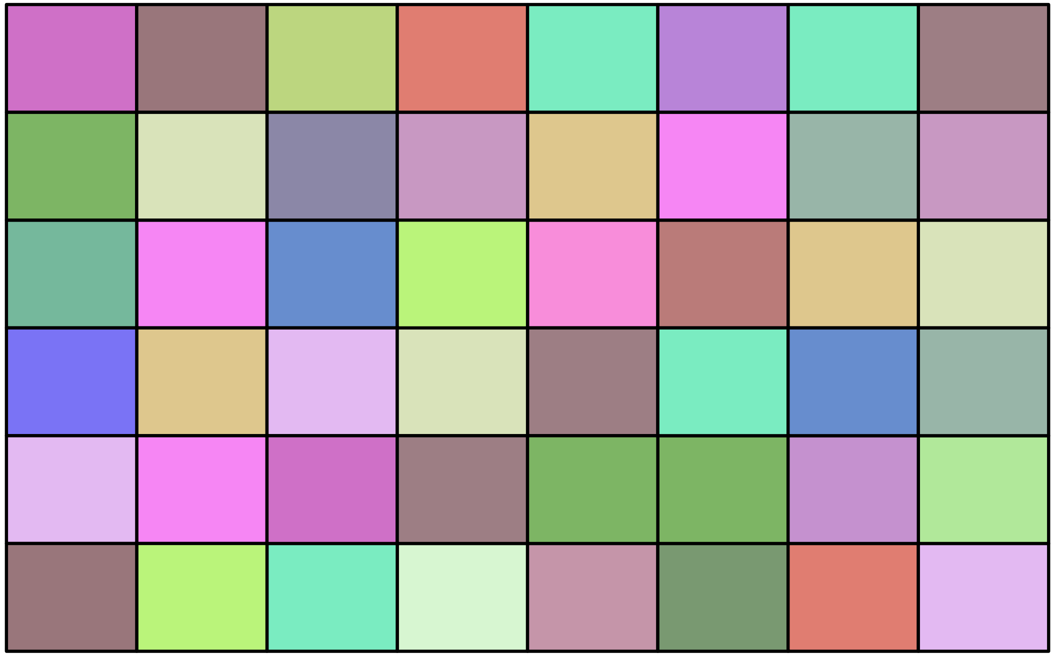
- London BikeGrid: `gicentre.org/bikegrid`
- OD Maps [Slingsby, Kelly, Dykes, Wood]

based on Spatial Tree Maps [Dykes,Wood]

Assigning Cells to Regions

How do we assign the grid cells to the regions?



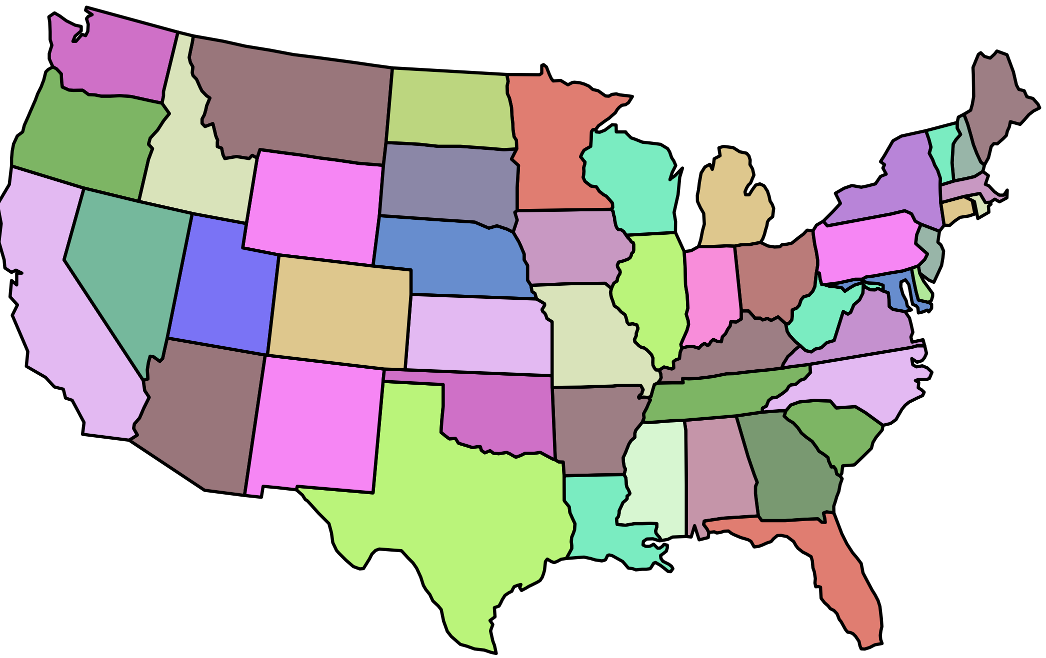


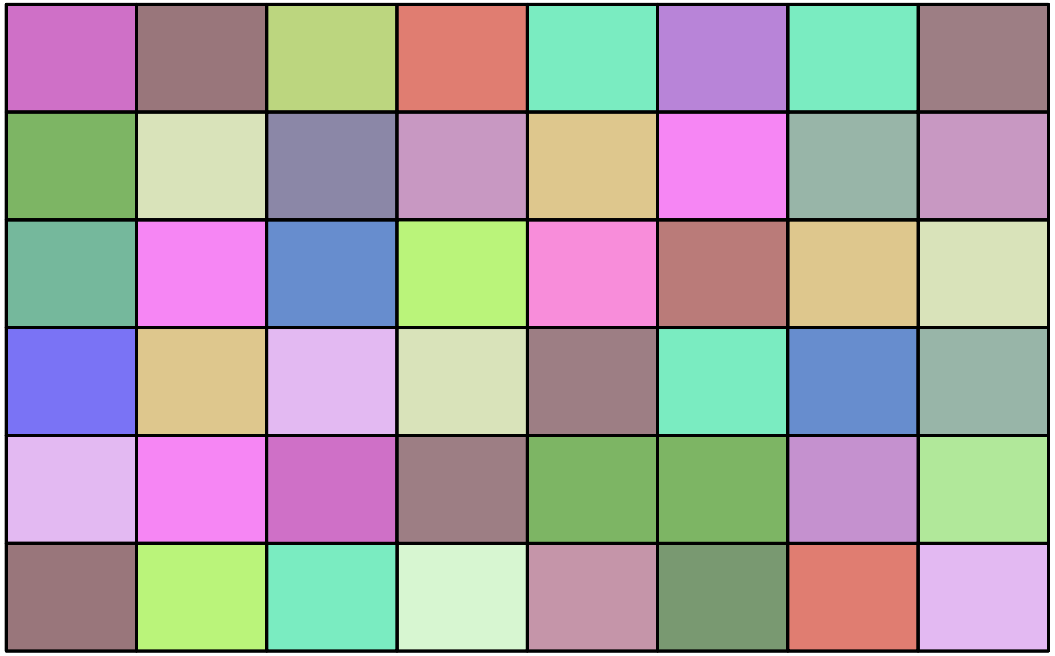
Tasks:

- Locate a cell
- Compare different cells
- Look for spatial patterns

Assigning Cells to Regions

How do we assign the grid cells to the regions?





Tasks:

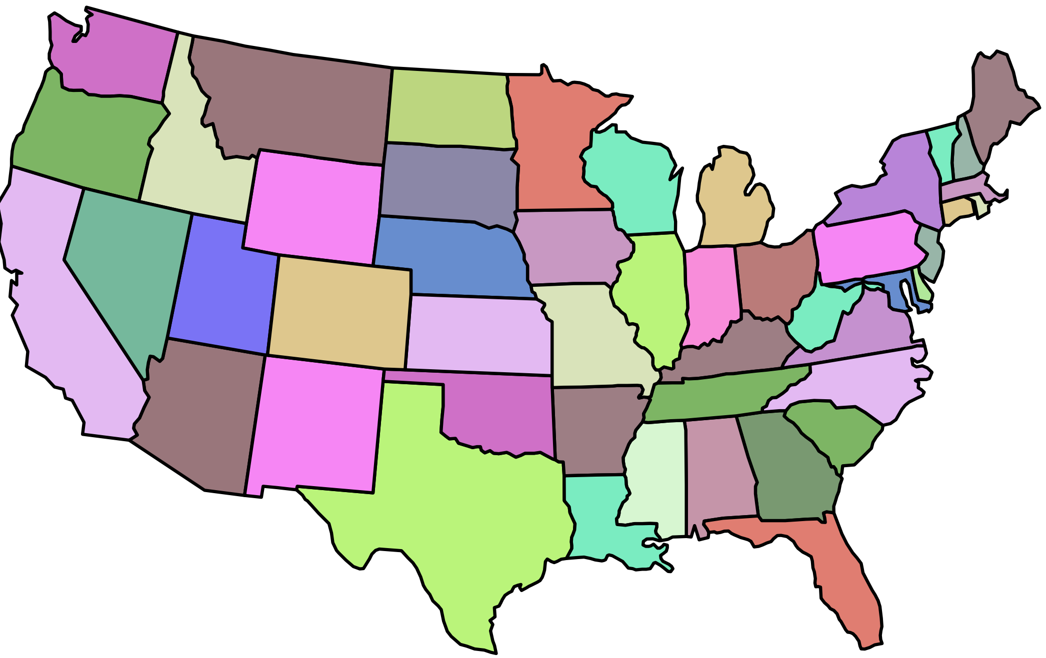
- Locate a cell
- Compare different cells
- Look for spatial patterns

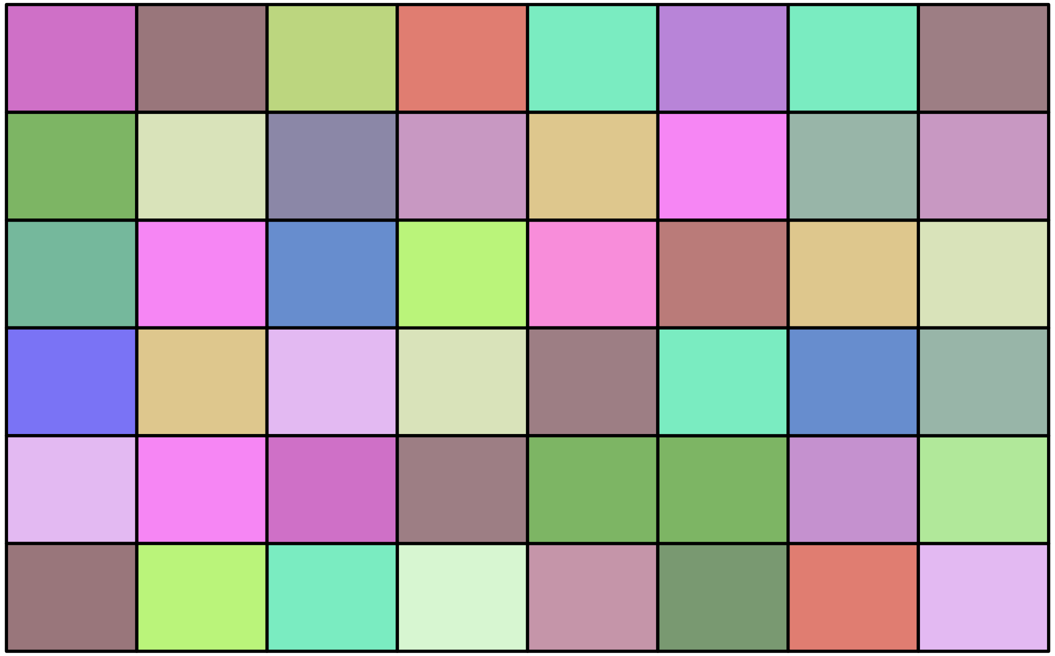
Optimisation criteria:

- Location
- Adjacency
- Relative orientation

Assigning Cells to Regions

How do we assign the grid cells to the regions?





Tasks:

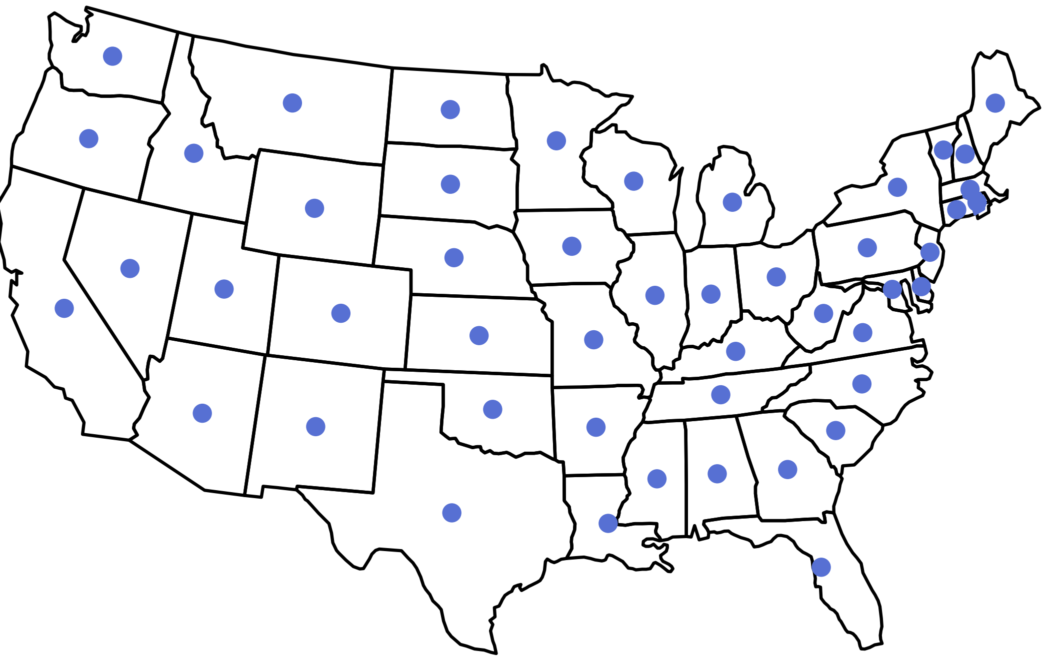
- Locate a cell
- Compare different cells
- Look for spatial patterns

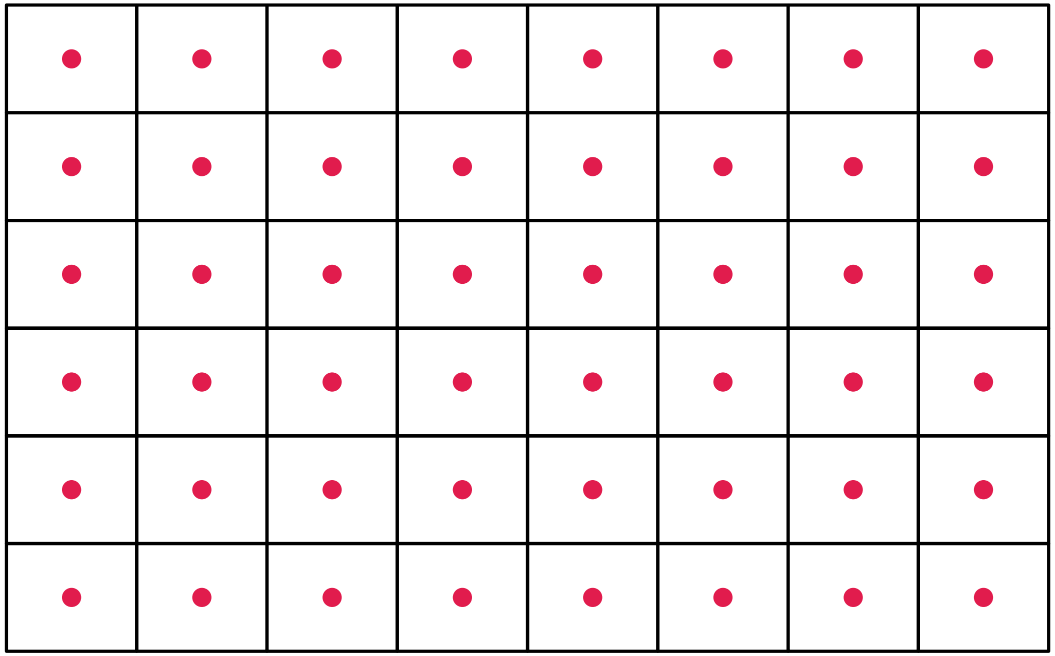
Optimisation criteria:

- Location
- ~~Adjacency~~ NP-Hard
- Relative orientation

Assigning Cells to Regions

How do we assign the grid cells to the regions?





1-to-1 Point Set Matching Problem

Regions $\rightsquigarrow$ set of blue points A.

Grid cells $\rightsquigarrow$ set of red points B.

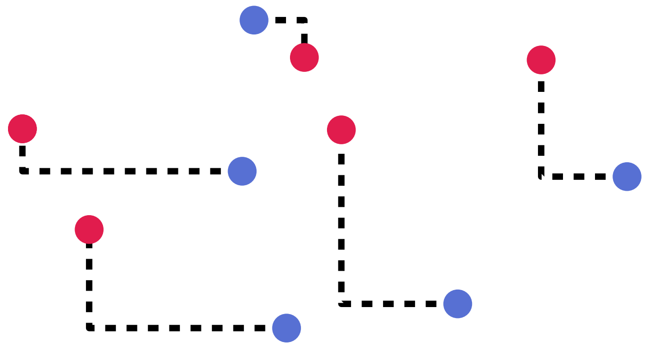
Optimisation criteria:

- Location
- ~~Adjacency~~ NP-Hard
- Relative orientation

Goal: find the best matching $\phi : A \rightarrow B$

Optimising Location

Minimize the sum of the L_1-distances between matched points



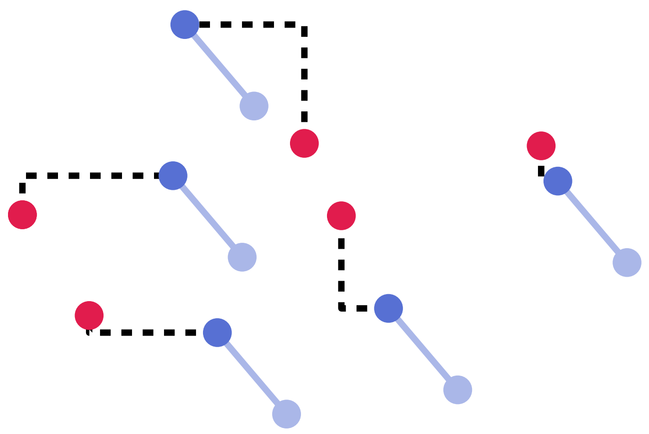
We want to find a matching ϕ^* that minimises

$$D_I(\phi) = \sum_{a \in A} d(a, \phi(a))$$

where $d(a, b) = |a_x - b_x| + |a_y - b_y|$.

Optimising Location

Minimize the sum of the L_1-distances between matched points under translation and scaling.

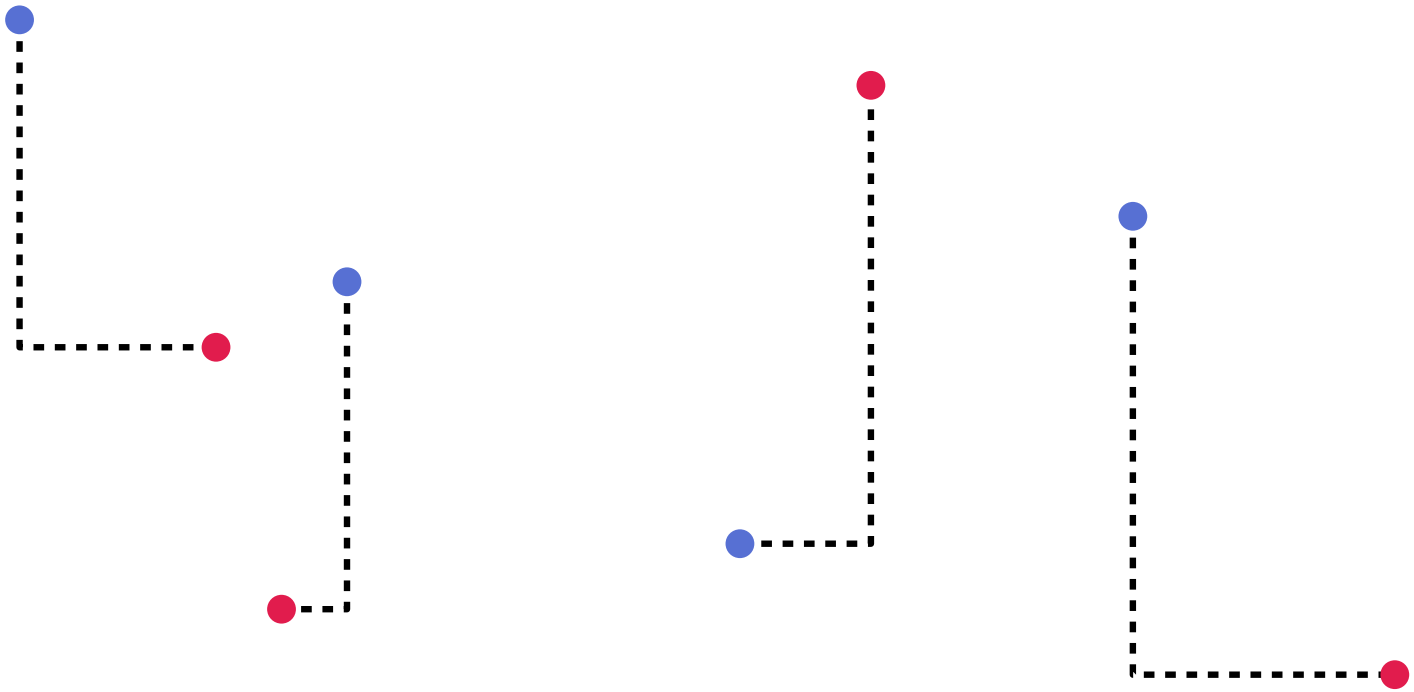


We want to find a matching ϕ^*, translation t^*, and scaling λ^* that minimise

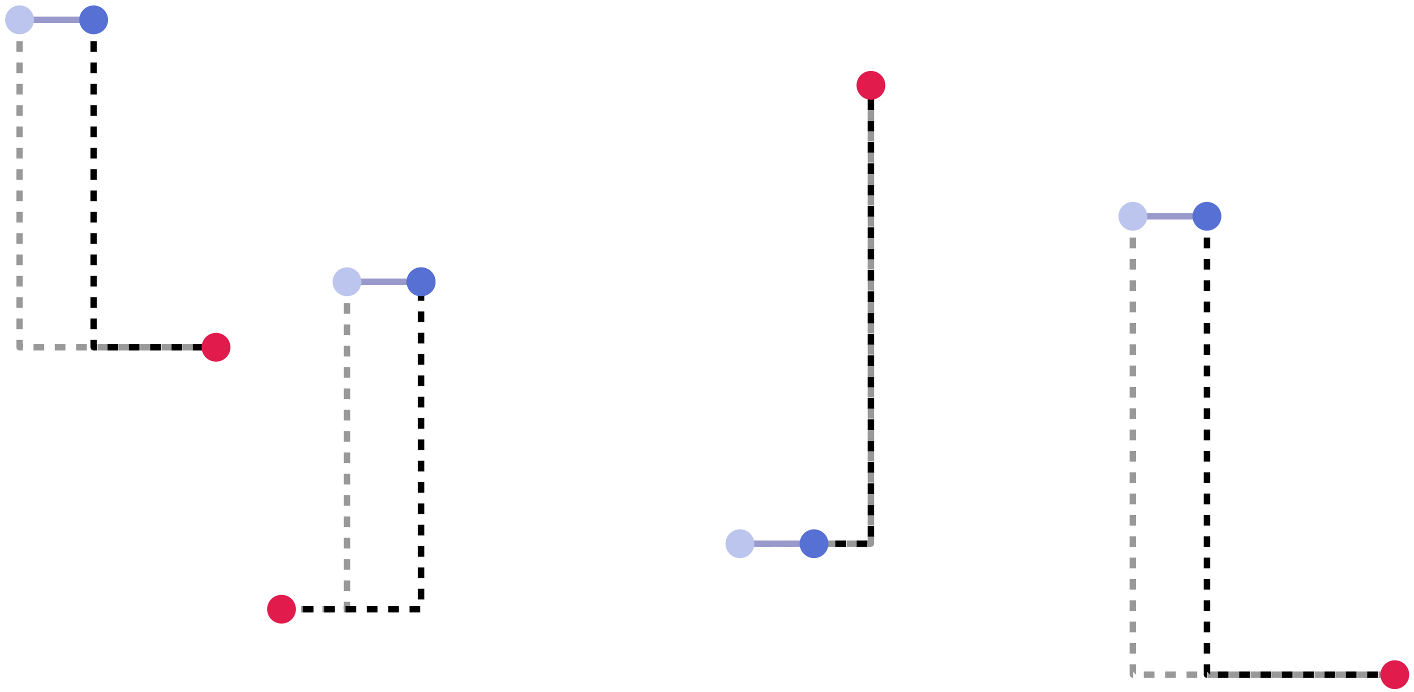
$$D(\phi, t, \lambda) = \sum_{a \in A} d(\lambda a + t, \phi(a)).$$

where $d(a, b) = |a_x - b_x| + |a_y - b_y|$.

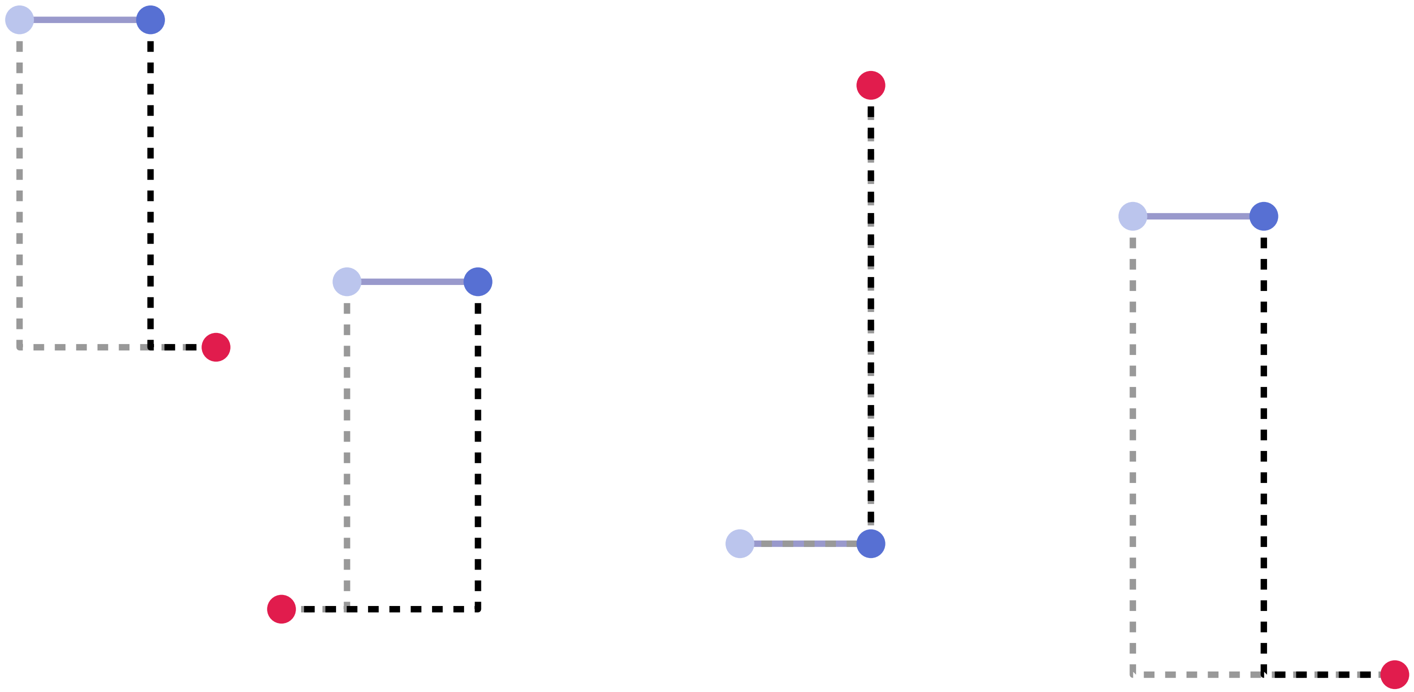
Translation only



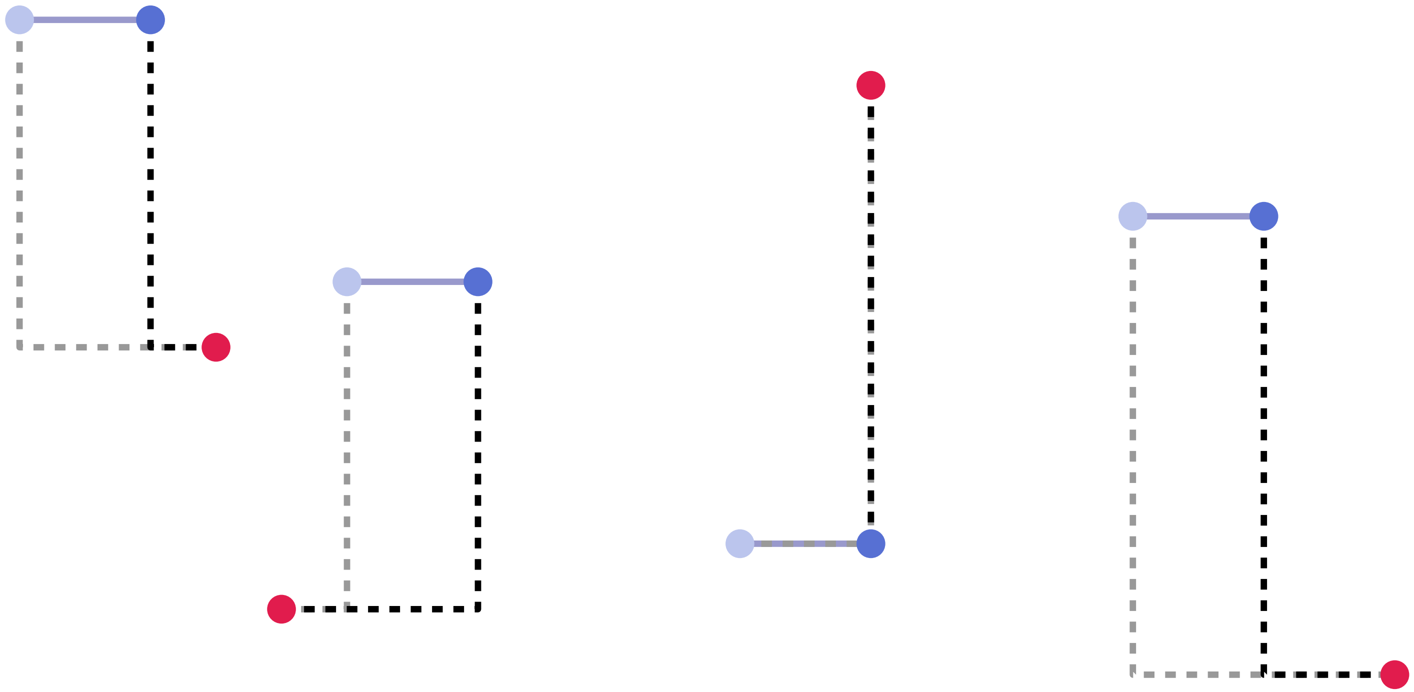
Translation only



Translation only



Translation only



Aligning A and B decreases $D_{\mathcal{T}}$

Lemma 1. *For any matching ϕ, there is a t that x-aligns A and B and minimises $D_{\mathcal{T}}(\phi, \cdot)$.*

Minimising $D_{\mathcal{T}}$

There is an optimal matching at an x-alignment.

Same trick for y-alignment.

Minimising $D_{\mathcal{T}}$

There is an optimal matching at an x-alignment.

Same trick for y-alignment.

There is an optimal matching at an x- and y-alignment.

$\Longrightarrow$ There are at most n^4 such alignments.

Minimising $D_{\mathcal{T}}$

There is an optimal matching at an x-alignment.

Same trick for y-alignment.

There is an optimal matching at an x- and y-alignment.

$\Longrightarrow$ There are at most n^4 such alignments.

Theorem 1. *A ϕ^* and t^* that minimise $D_{\mathcal{T}}$ can be computed in $O(n^4 \cdot n^2 \log^3 n) = O(n^6 \log^3 n)$ time.*

Uses the matching algorithm by Vaidya (1988)

Minimising D_Λ and D

Minimum distance matching under scaling?

Use exactly the same approach.

Minimising D_Λ and D

Minimum distance matching under scaling?

Use exactly the same approach.

Minimum distance matching under translation and scaling?

Minimising D_Λ and D

Minimum distance matching under scaling?

Use exactly the same approach.

Minimum distance matching under translation and scaling?

Same idea: x-align (y-align) two pairs of points.

Minimising D_Λ and D

Minimum distance matching under scaling?

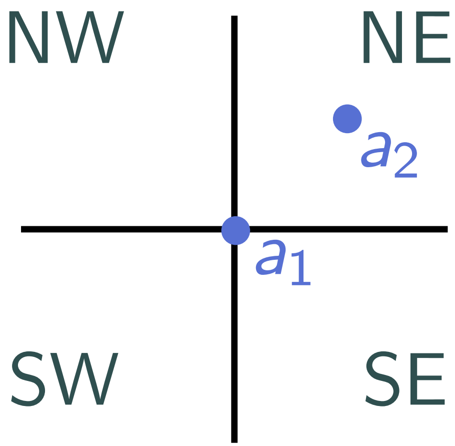
Use exactly the same approach.

Minimum distance matching under translation and scaling?

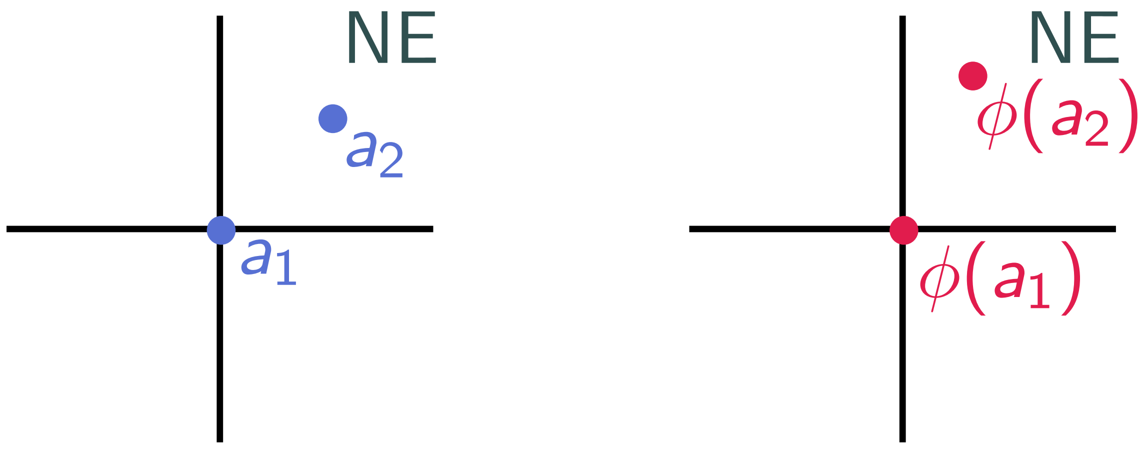
Same idea: x-align (y-align) two pairs of points.

Theorem 2. *A ϕ^*, t^*, and λ^* that minimise D can be computed in $O(n^8 \cdot n^2 \log^3 n) = O(n^{10} \log^3 n)$ time.*

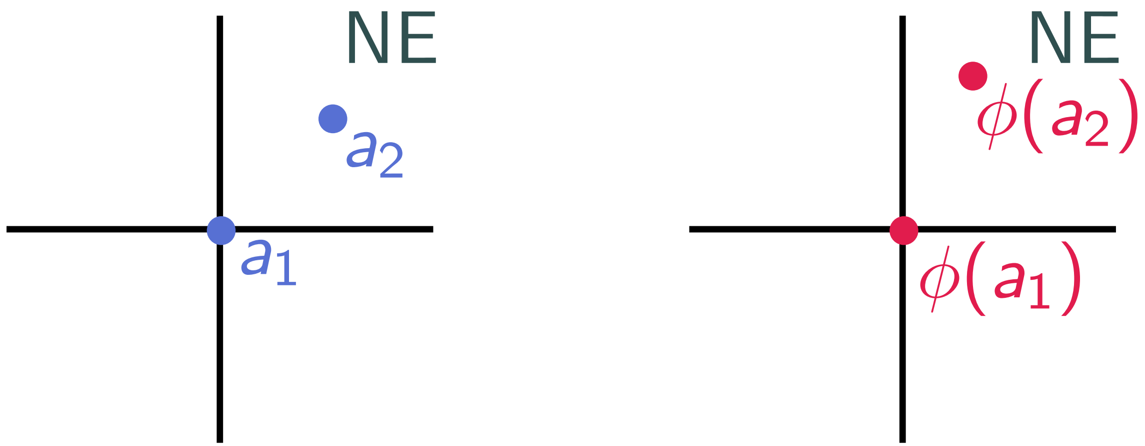
Preserving directional relations



Preserving directional relations

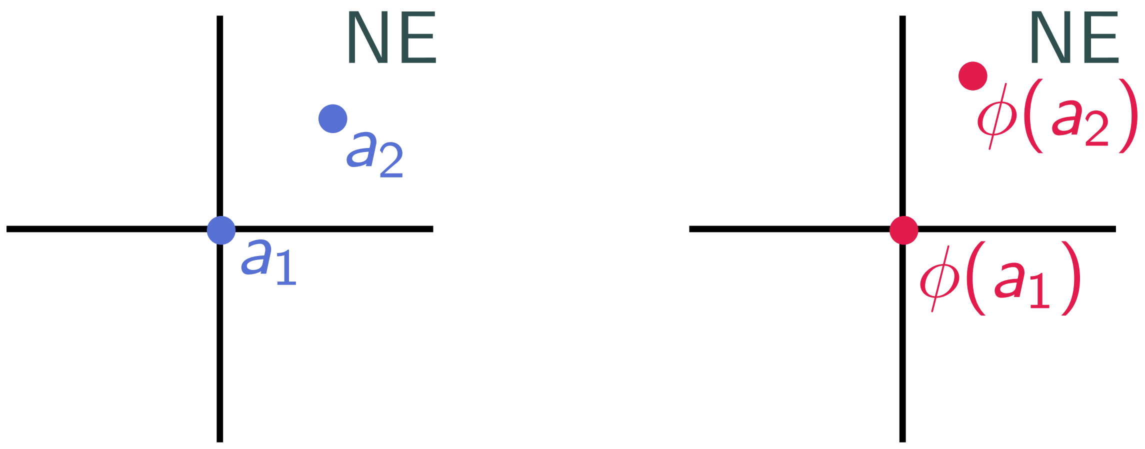


Preserving directional relations



Maximize the number of pairs with the right orientation.

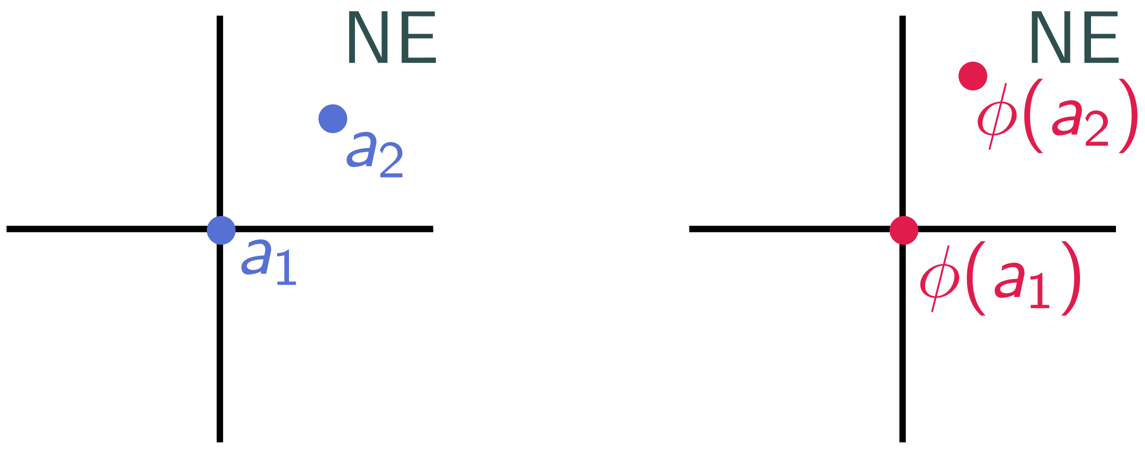
Preserving directional relations



Minimize the number of $\underbrace{\text{pairs with the wrong orientation}}_{\text{out-of-order pairs}}$.

$$W(\phi) = |\{(a_1, a_2) \mid (a_1, a_2) \in A \times A \wedge$$
$$dir(a_1, a_2) \neq dir(\phi(a_1), \phi(a_2))\}|.$$

Preserving directional relations



Minimize the number of $\underbrace{\text{pairs with the wrong orientation}}_{\text{out-of-order pairs}}$.

$$W(\phi) = |\{(a_1, a_2) \mid (a_1, a_2) \in A \times A \wedge \\ dir(a_1, a_2) \neq dir(\phi(a_1), \phi(a_2))\}|.$$

Translation and scaling do not influence W.

A 4-approximation algorithm

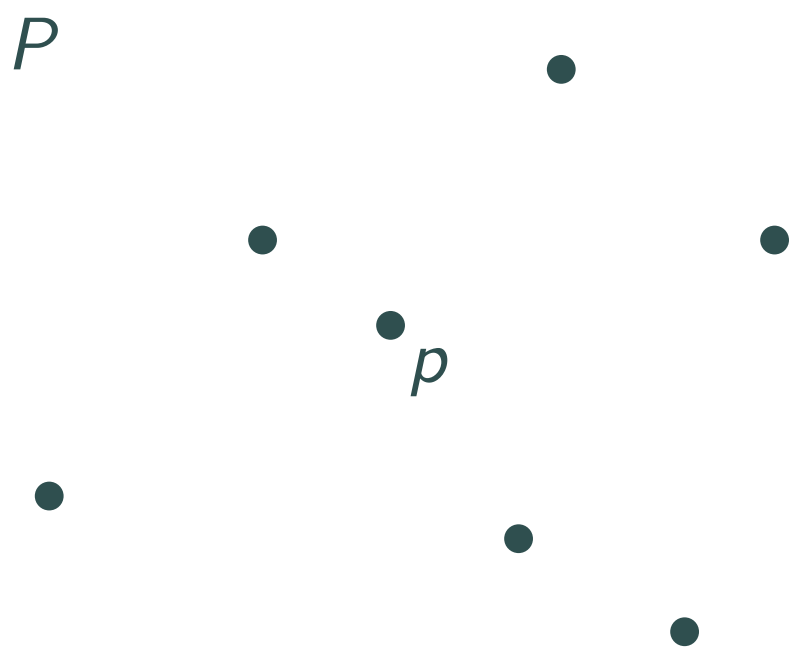
Compute a minimum distance matching with distance measure

$$\begin{aligned} w(a, b) = {} & |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + \\ & |y\text{-}rank_A(a) - y\text{-}rank_B(b)|. \end{aligned}$$

A 4-approximation algorithm

Compute a minimum distance matching with distance measure

$$w(a, b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|.$$



$$x\text{-}rank_P(p) = 3$$
$$y\text{-}rank_P(p) = 4$$

A 4-approximation algorithm

Compute a minimum distance matching with distance measure

$$w(a,b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|.$$

$w(a,b)$ is the L_1-distance in terms of ranks.

A 4-approximation algorithm

Compute a minimum distance matching with distance measure

$$w(a, b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|.$$

$w(a, b)$ is the L_1-distance in terms of ranks.

So compute an optimal matching using Vaidya's Algorithm.

Theorem 3. *A 4-approximation for minimising W can be computed in $O(n^2 \log^3 n)$.*

Implementation & Evaluation

Implementation: Easy; we only need an LP-solver.

Evaluation: We compare with spatial tree maps [Wood & Dykes], and minimizing the L_2^2 distance [Cohen & Guibas]

Implementation & Evaluation

Implementation: Easy; we only need an LP-solver.

Evaluation: We compare with spatial tree maps [Wood & Dykes], and minimizing the L_2^2 distance [Cohen & Guibas]

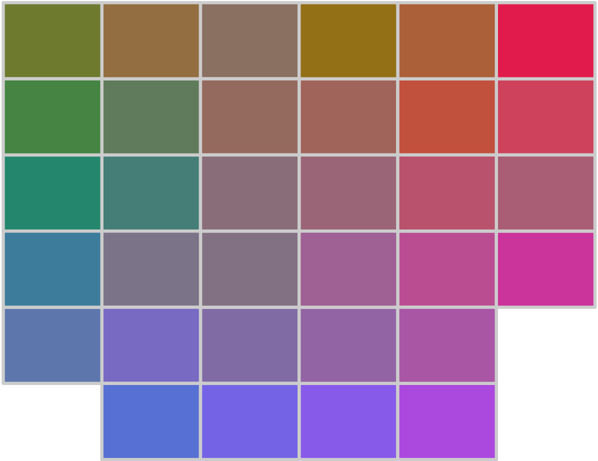
- Quantitative
  - distance
  - # and % preserved directional relations
  - # and % preserved adjacencies
- Qualitative

Implementation & Evaluation

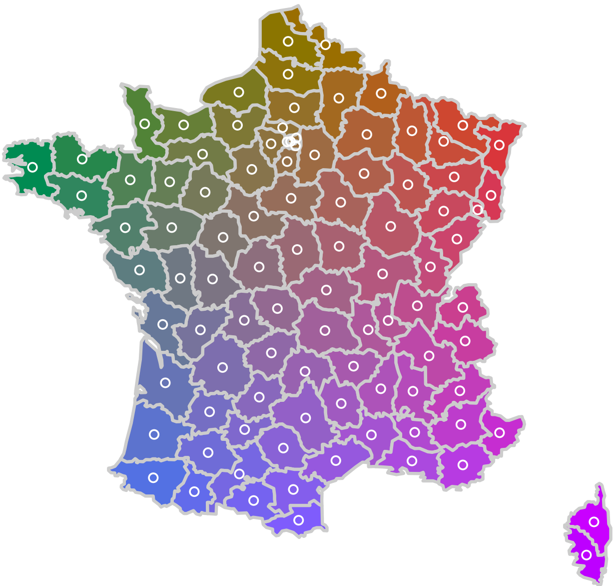
Implementation: Easy; we only need an LP-solver.

Evaluation: We compare with spatial tree maps [Wood & Dykes], and minimizing the L_2^2 distance [Cohen & Guibas]

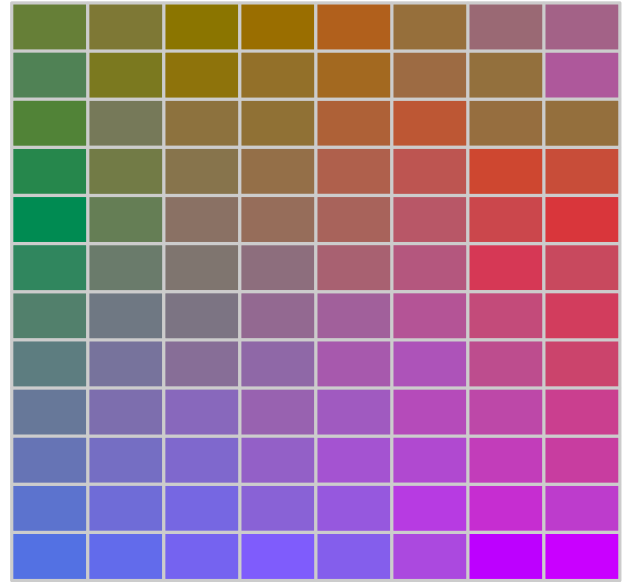
- Quantitative
  - distance
  - # and % preserved directional relations
  - # and % preserved adjacencies
- Qualitative



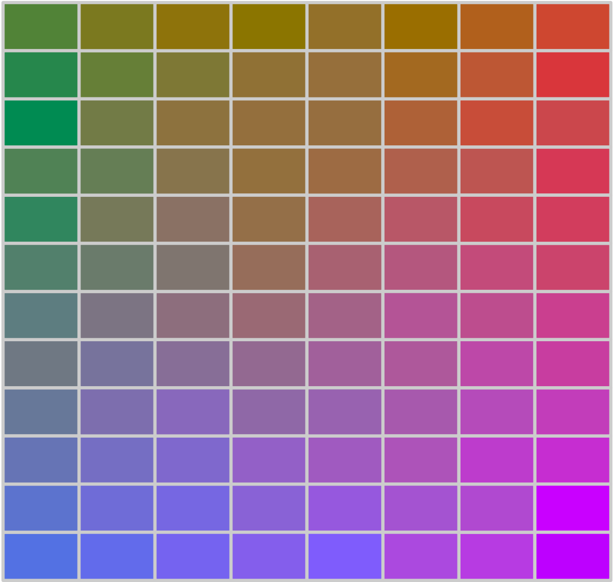
Results



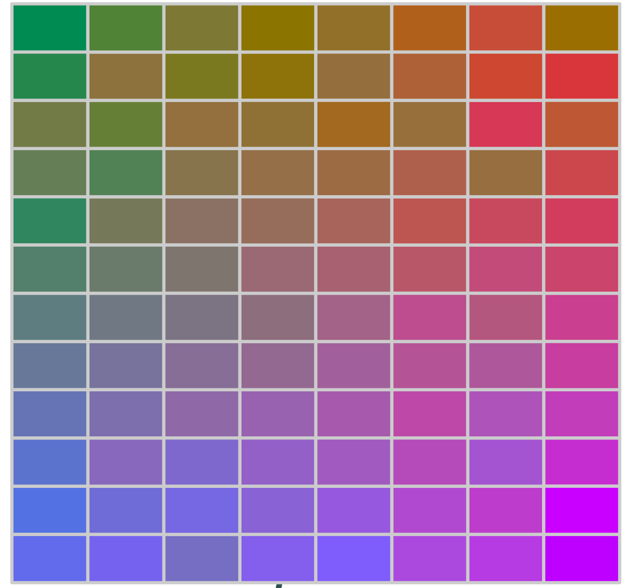
	Dir. Rel.	Adj.
SG	88%	69%
L_1	96%	76%
W	97%	82%
L_2^2	98%	81%



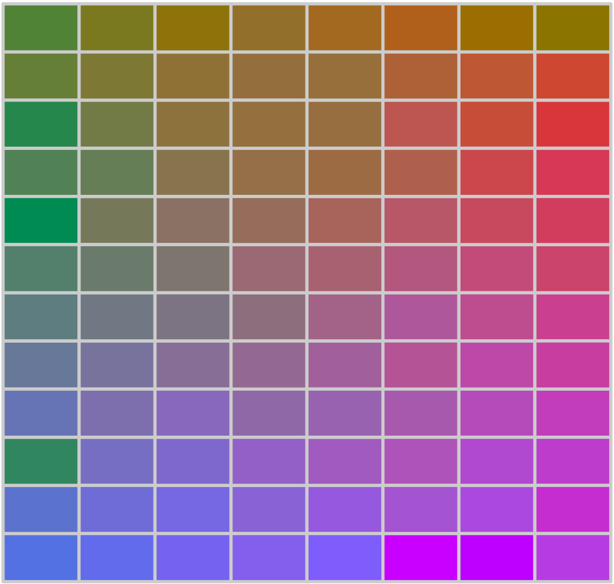
SG [Wood and Dykes]



L_2^2

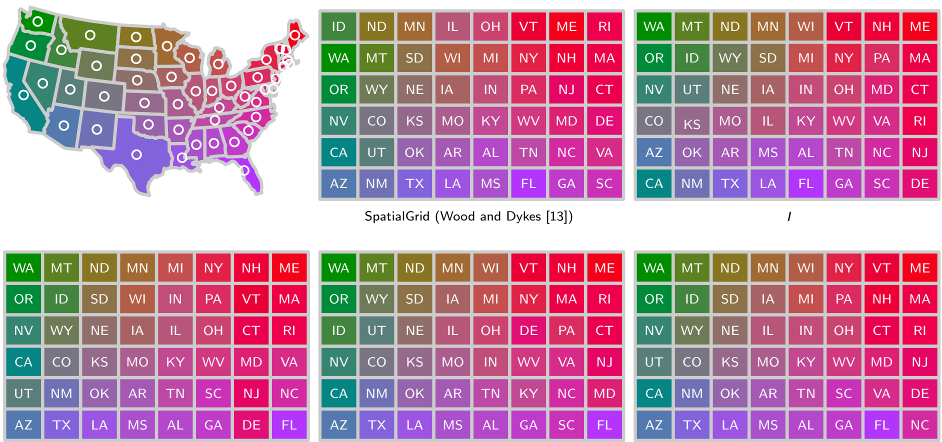


L_1

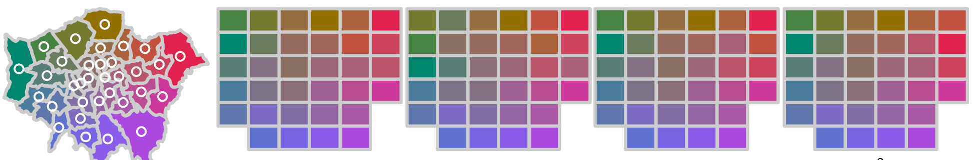


W

Results



SpatialGrid (Wood and Dykes [13]) — I — L_1 — W — L_2^2



I — L_1 — W — L_2^2

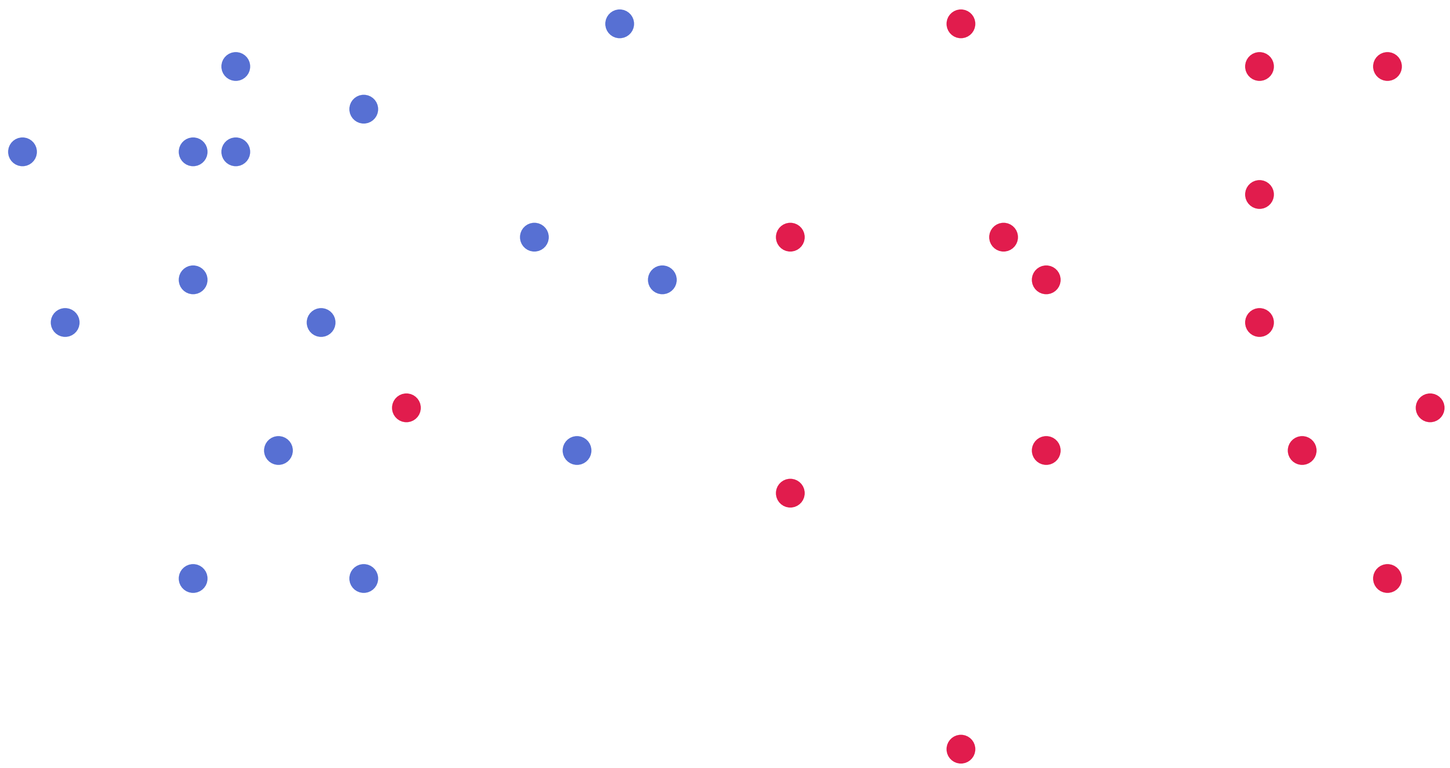
Method	Distance			Directional Rel.		Adjacencies	
	L_1	L_2	L_2^2	#	%	#	%
SpatialGrid	4545	3592	300482	2008	89.01%	77	73.33%
I	4035	3342	311327	2024	89.72%	79	75.24%
L_1	2838	2355	166060	2046	90.69%	78	74.29%
W	4221	3352	273273	2098	93.00%	79	75.24%
L_2^2	2929	2260	139110	2096	92.91%	83	79.05%

Method	Distance			Directional Rel.		Adjacencies	
	L_1	L_2	L_2^2	#	%	#	%
I	2897	2296	182257	1040	98.48%	59	72.84%
L_1	2803	2286	200593	1008	95.45%	54	66.67%
W	2936	2277	177927	1042	98.67%	61	75.31%
L_2^2	2890	2228	172089	1042	98.67%	61	75.31%

Results for the United States and the London Boroughs are in the paper.

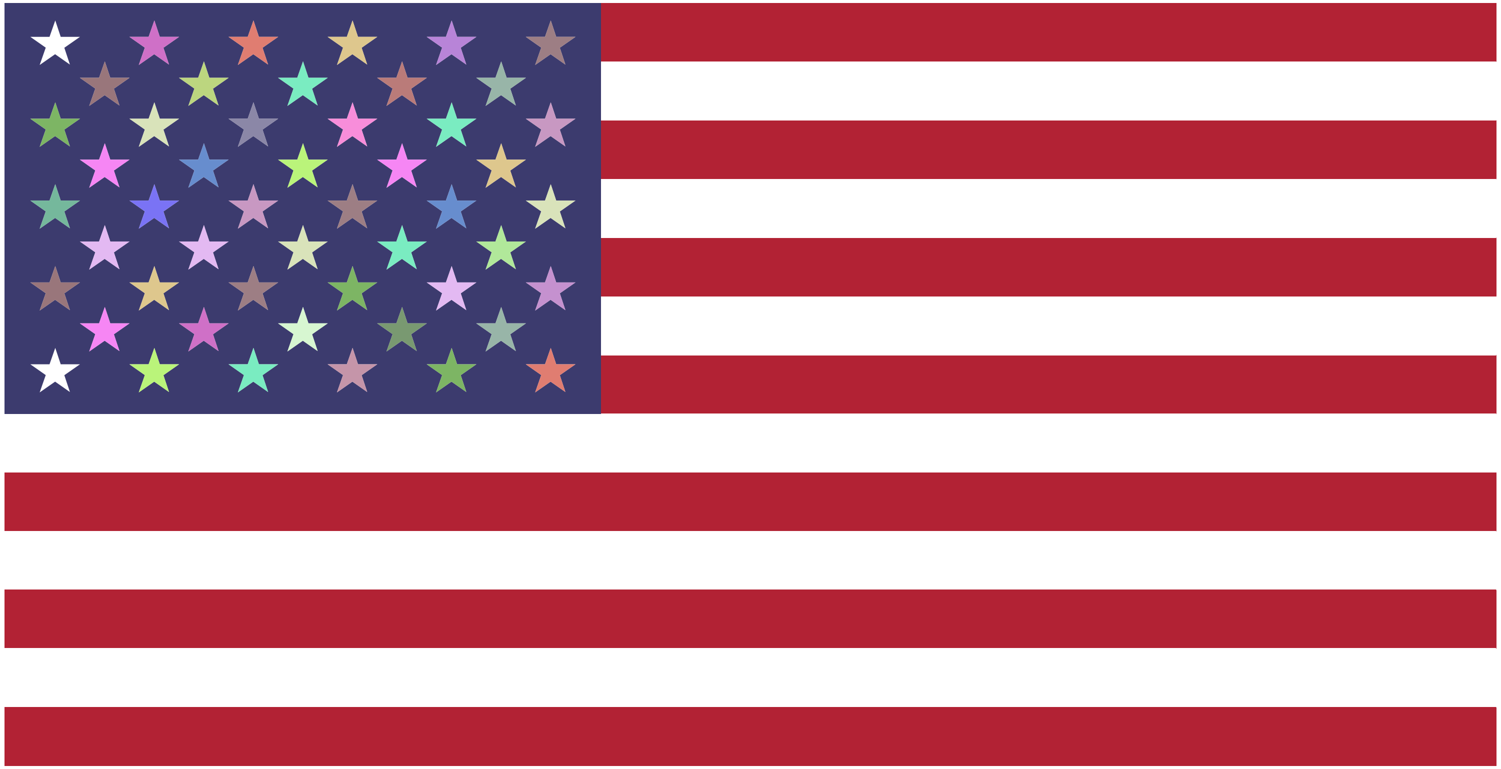
Concluding Remarks & Future Work

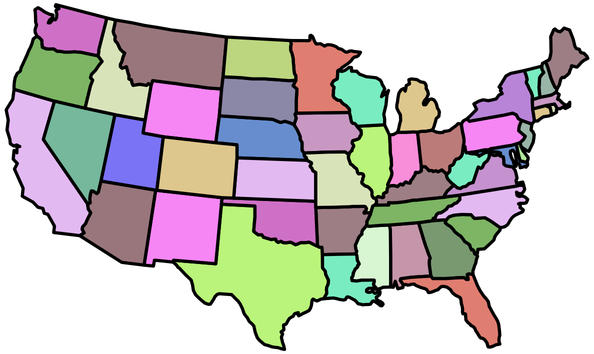
Our method works for arbitrary point sets.



Concluding Remarks & Future Work

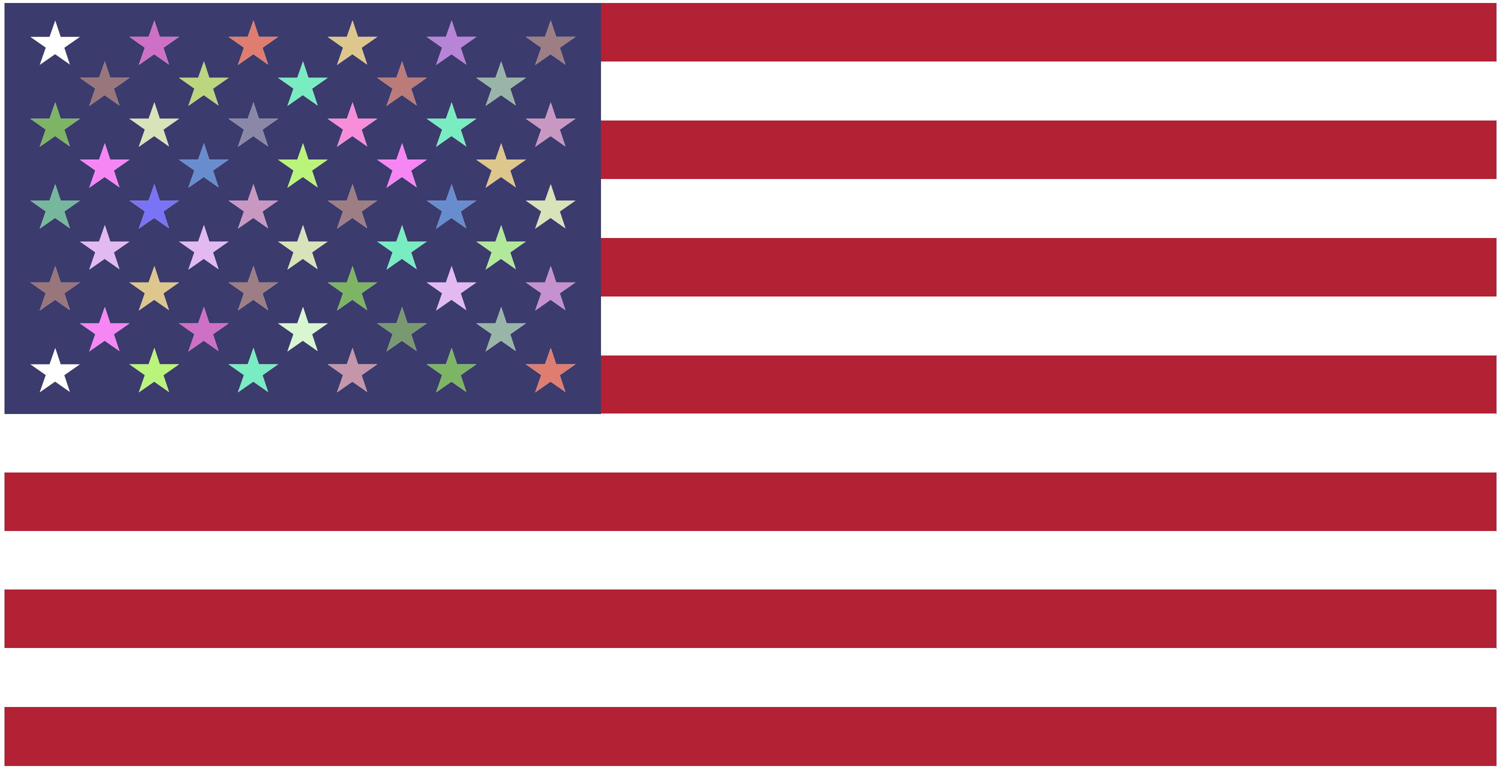
Our method works for arbitrary point sets.

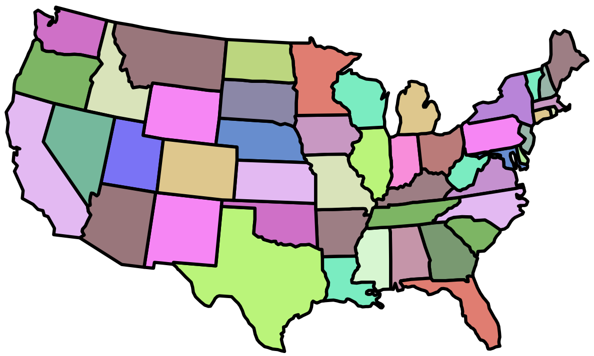




Concluding Remarks & Future Work

Our method works for arbitrary point sets.

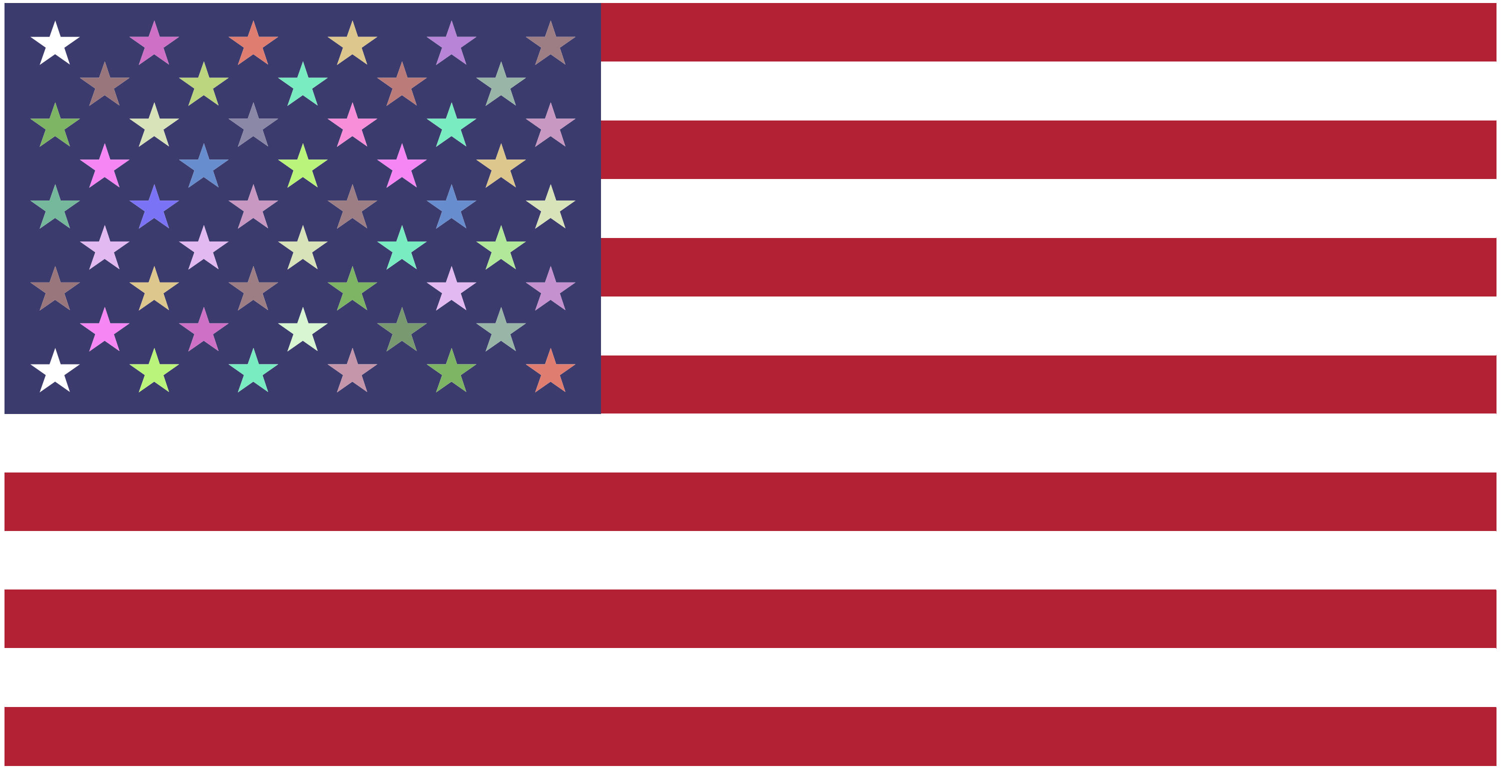


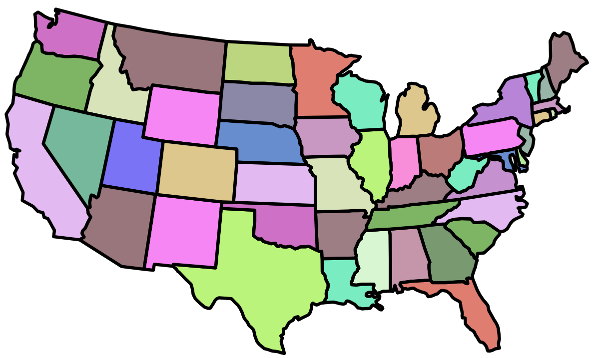


Future work: How to find cells (not) to use?

Concluding Remarks & Future Work

Our method works for arbitrary point sets.

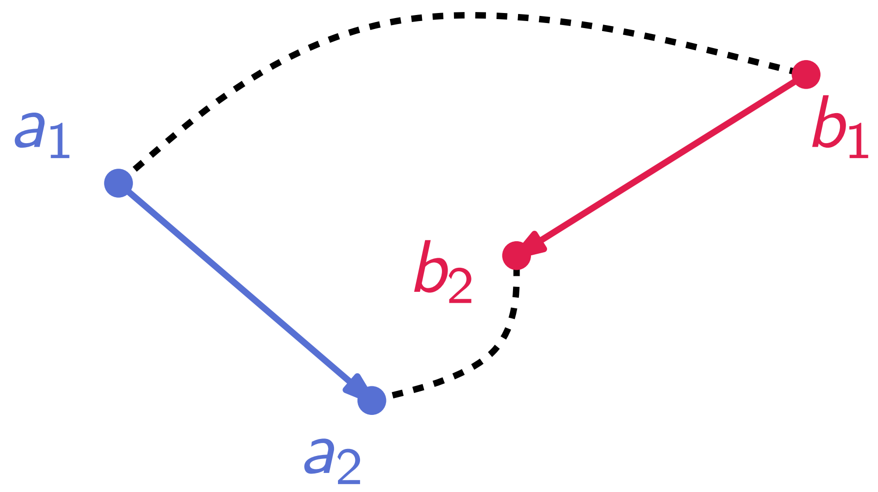




Future work: How to find cells (not) to use?

Thank you!

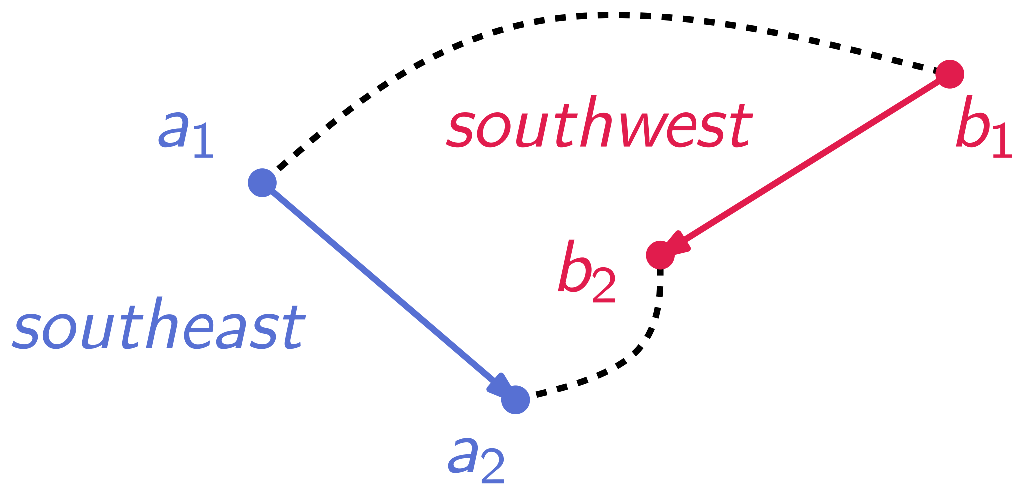
Inversions vs Directions



$x\text{-}rank_A(a_1) < x\text{-}rank_A(a_2)$ and
$x\text{-}rank_B(b_1) > x\text{-}rank_B(b_2)$

(a_1, a_2) is an inversion.

Inversions vs Directions



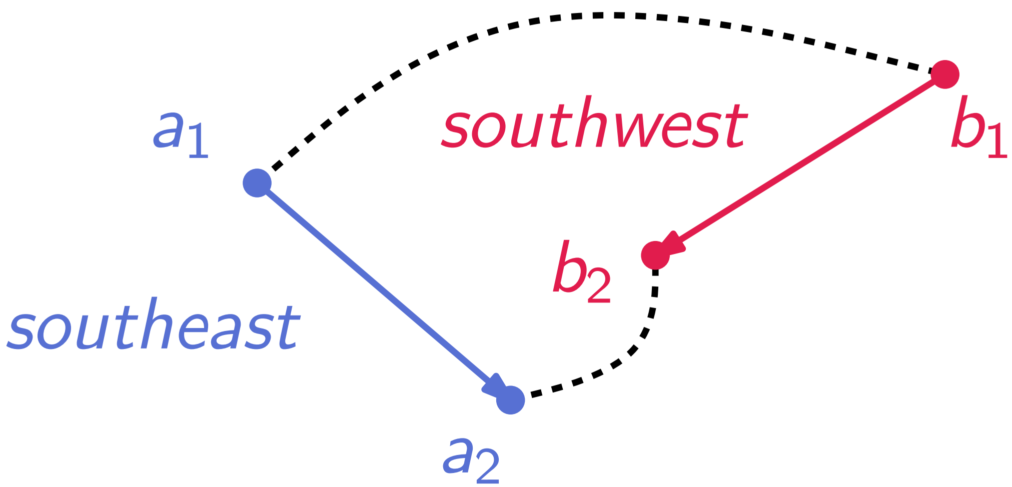
$x\text{-}rank_A(a_1) < x\text{-}rank_A(a_2)$ and
$x\text{-}rank_B(b_1) > x\text{-}rank_B(b_2)$

(a_1, a_2) is an inversion.

$\Updownarrow$

(a_1, a_2) is an out-of-order pair

Inversions vs Directions



$x\text{-}rank_A(a_1) < x\text{-}rank_A(a_2)$ and
$x\text{-}rank_B(b_1) > x\text{-}rank_B(b_2)$

(a_1, a_2) is an inversion.

$$\Updownarrow$$

(a_1, a_2) is an out-of-order pair

So $W(\phi) = \#\text{inversions} = I(\phi)$.