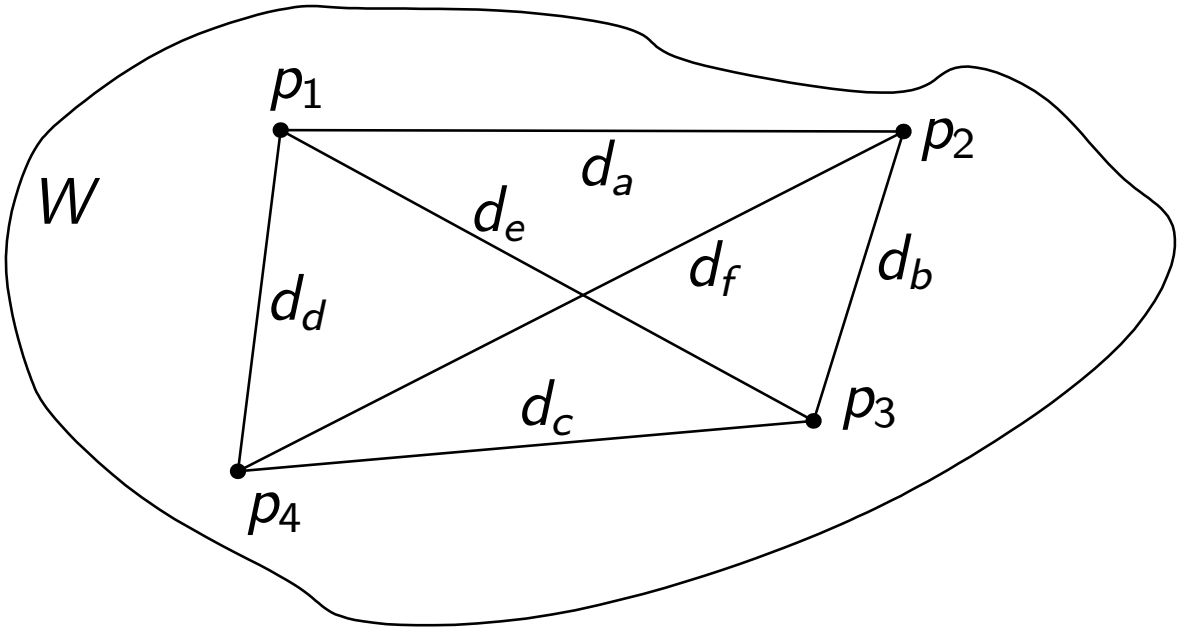
On dimension partitions in discrete metric spaces

Fabien Rebatel, Édouard Thiel

Laboratoire d'Informatique Fondamentale de Marseille
Aix-Marseille Université

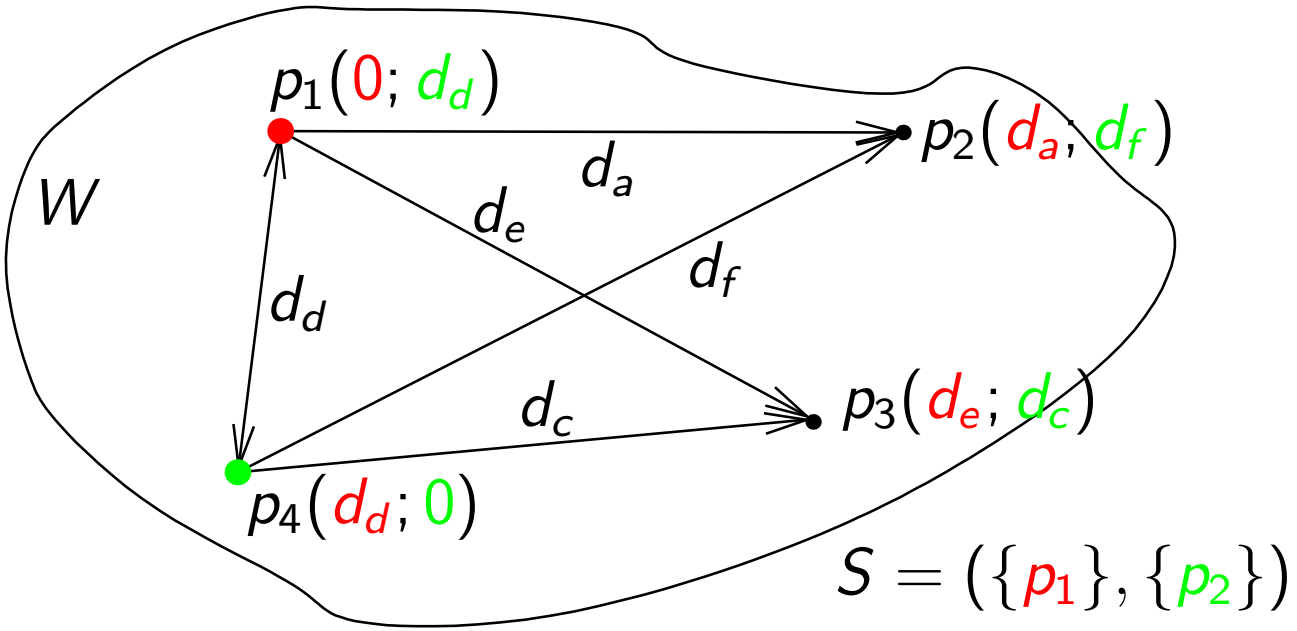
DGCI 2013, March 20-22, 2013, Sevilla, Spain

Metric basis



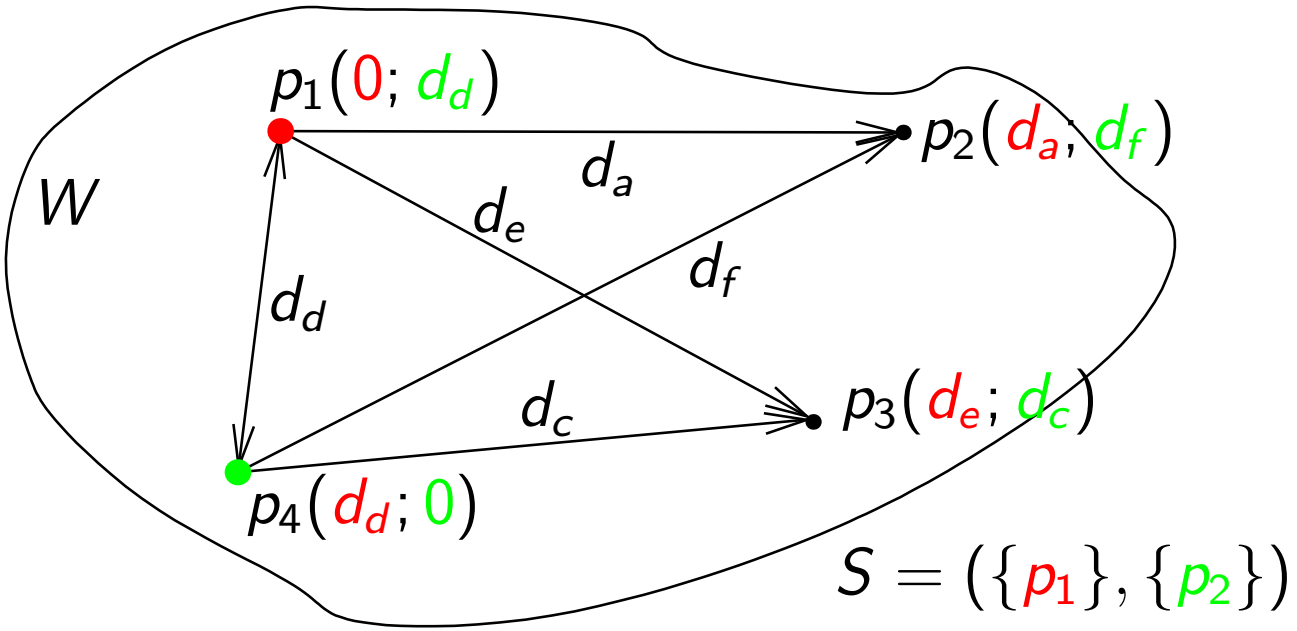
Let (W, d) be a metric space.

Metric basis



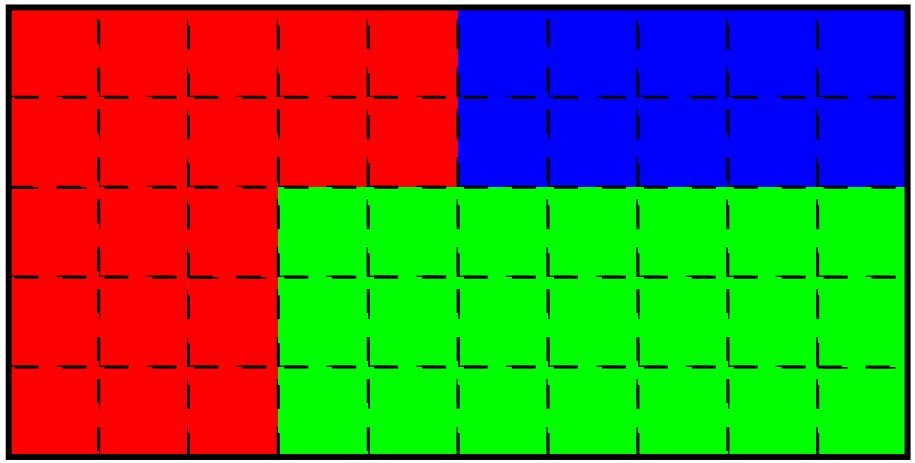
Let (W, d) be a metric space.
A subset $S \subseteq W$ is a **resolving set** for W if $d(x, p) = d(y, p)$ for all $p \in S$ implies $x = y$.

Metric basis



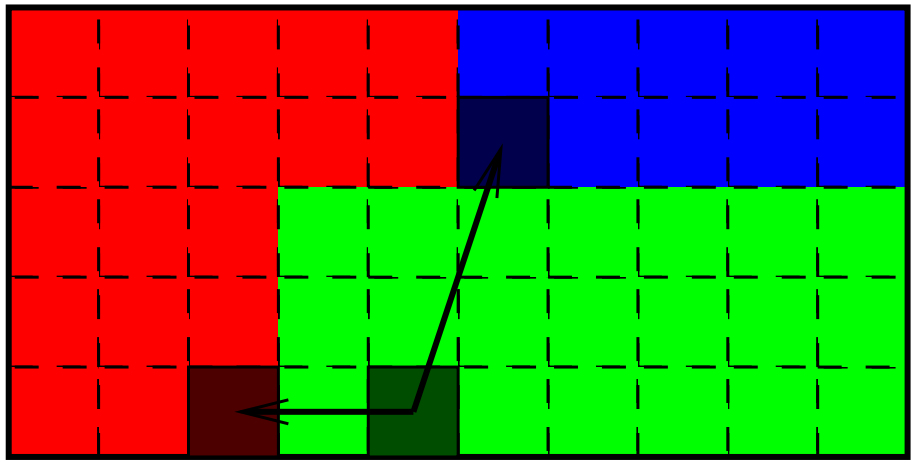
Let (W, d) be a metric space.
A subset $S \subseteq W$ is a **resolving set** for W if $d(x, p) = d(y, p)$ for all $p \in S$ implies $x = y$.
A **metric basis** is a resolving set of minimal cardinality, named the **metric dimension** of (W, d).

Partition Dimension



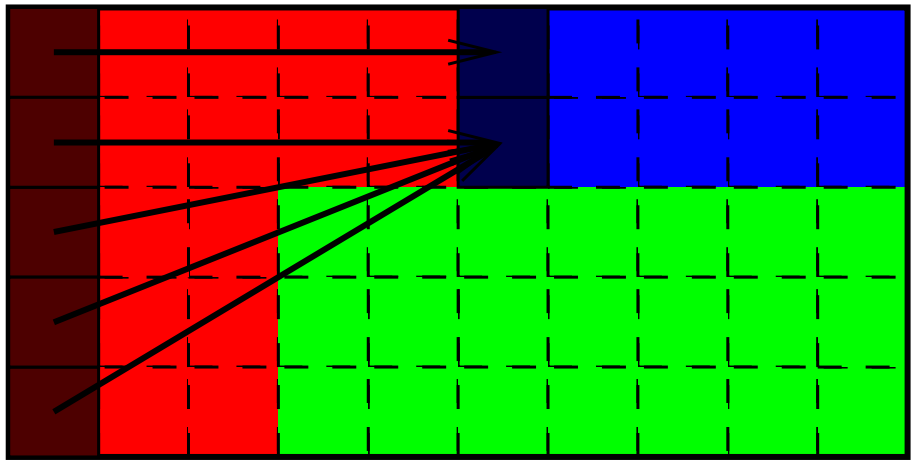
Let (W, d) be a metric space, and P a partition of that space.

Partition Dimension



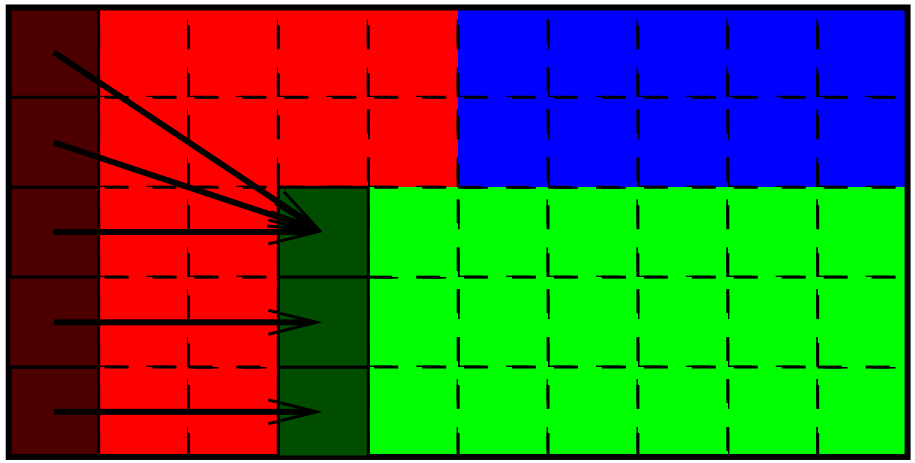
Let (W, d) be a metric space, and P a partition of that space.
The distance between a point and a part is the distance between that point and the nearest point of that part.

Partition Dimension



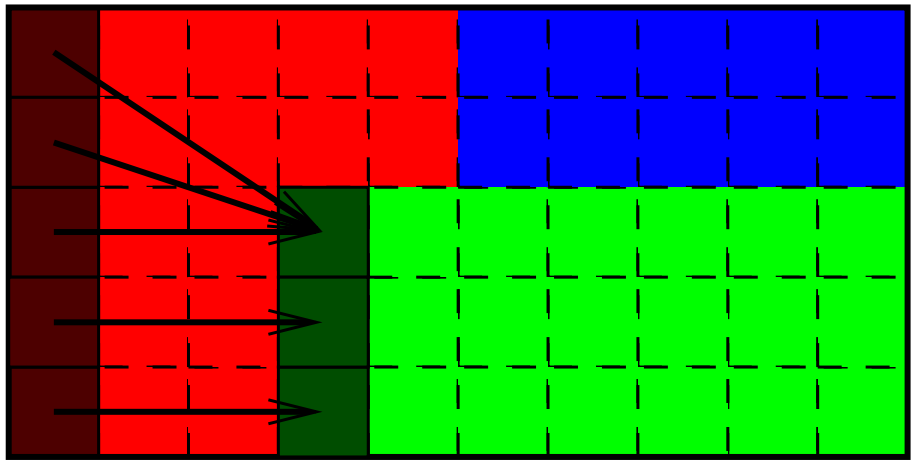
Let (W, d) be a metric space, and P a partition of that space.
The distance between a point and a part is the distance between that point and the nearest point of that part.
A partition P is a **resolving partition** for W if $d(x, p) = d(y, p)$ for all $p \in P$ implies $x = y$.

Partition Dimension



Let (W, d) be a metric space, and P a partition of that space.
The distance between a point and a part is the distance between that point and the nearest point of that part.
A partition P is a **resolving partition** for W if $d(x, p) = d(y, p)$ for all $p \in P$ implies $x = y$.
(Every red highlighted points have $(0, 3, 5)$ as coordinates for the ℓ_∞ norm.)

Partition Dimension



Let (W, d) be a metric space, and P a partition of that space.
The distance between a point and a part is the distance between that point and the nearest point of that part.
A partition P is a **resolving partition** for W if $d(x, p) = d(y, p)$ for all $p \in P$ implies $x = y$.
(Every red highlighted points have $(0, 3, 5)$ as coordinates for the ℓ_∞ norm.)
A **partition basis** is a resolving partition of minimal cardinality, named the **partition dimension** of (W, d).

Notations

Given a set $W = (e_1, e_2, e_3, e_4, e_5)$, both partitions $(\{e_2\}, \{e_3, e_1, e_5\}, \{e_4\})$ and $(\{e_1, e_3, e_5\}, \{e_2\}, \{e_4\})$ are equivalent.

Notations

Given a set $W = (e_1, e_2, e_3, e_4, e_5)$, both partitions $(\{e_2\}, \{e_3, e_1, e_5\}, \{e_4\})$ and $(\{e_1, e_3, e_5\}, \{e_2\}, \{e_4\})$ are equivalent.

We only denote a partition as follow:

- in each part, the points are sorted by their index,
- part are sorted by decreasing cardinal and
- part with same cardinal are sorted in lexicographic order.

Notations

Given a set $W = (e_1, e_2, e_3, e_4, e_5)$, both partitions $(\{e_2\}, \{e_3, e_1, e_5\}, \{e_4\})$ and $(\{e_1, e_3, e_5\}, \{e_2\}, \{e_4\})$ are equivalent.

We only denote a partition as follow:

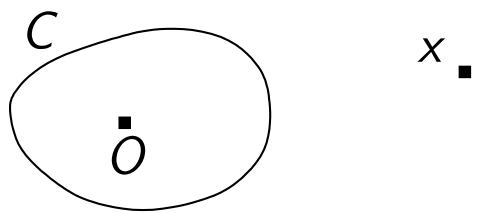
- in each part, the points are sorted by their index,
- part are sorted by decreasing cardinal and
- part with same cardinal are sorted in lexicographic order.

We denote the **partition class** as the vector of cardinality of its parts.
$Class(\{e_1, e_3, e_5\}, \{e_2\}, \{e_4\}) = [3, 1, 1]$.

Gauges and chamfer norms

Definition

Given a convex $\mathcal{C}$ containing the origin O in its interior, a **gauge** for $\mathcal{C}$ is the function $\gamma_{\mathcal{C}}(x)$ defined by the minimum positive scale factor λ, necessary for having $x \in \lambda\mathcal{C}$.

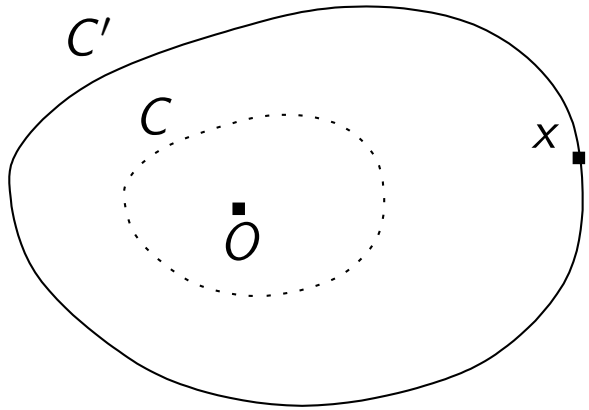


Polyhedral gauges are the $\mathbb{R}^n$ generalization of chamfer norms.

Gauges and chamfer norms

Definition

Given a convex $\mathcal{C}$ containing the origin O in its interior, a **gauge** for $\mathcal{C}$ is the function $\gamma_{\mathcal{C}}(x)$ defined by the minimum positive scale factor λ, necessary for having $x \in \lambda\mathcal{C}$.

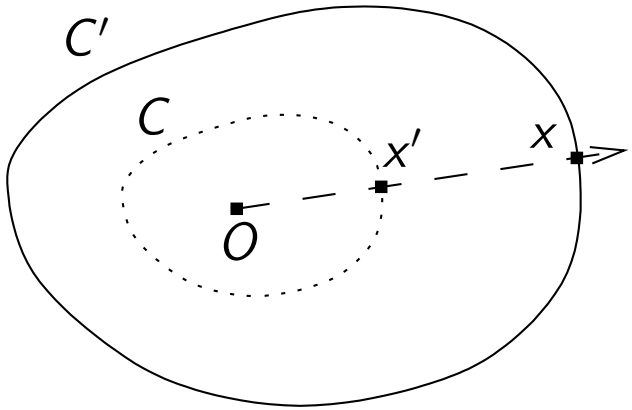


Polyhedral gauges are the $\mathbb{R}^n$ generalization of chamfer norms.

Gauges and chamfer norms

Definition

Given a convex $\mathcal{C}$ containing the origin O in its interior, a **gauge** for $\mathcal{C}$ is the function $\gamma_{\mathcal{C}}(x)$ defined by the minimum positive scale factor λ, necessary for having $x \in \lambda\mathcal{C}$.

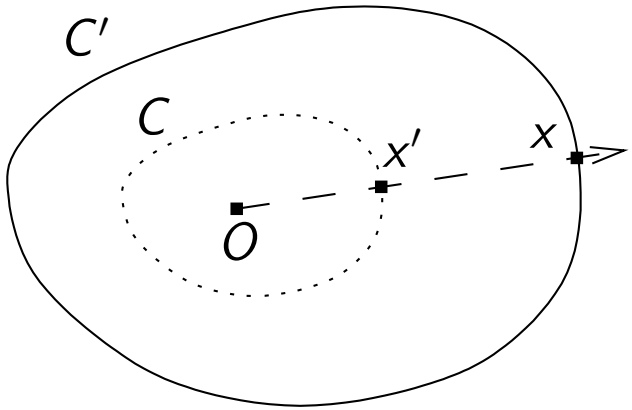


Polyhedral gauges are the $\mathbb{R}^n$ generalization of chamfer norms.

Gauges and chamfer norms

Definition

Given a convex $\mathcal{C}$ containing the origin O in its interior, a **gauge** for $\mathcal{C}$ is the function $\gamma_{\mathcal{C}}(x)$ defined by the minimum positive scale factor λ, necessary for having $x \in \lambda\mathcal{C}$.



Polyhedral gauges are the $\mathbb{R}^n$ generalization of chamfer norms.

State of art

G.Chartrand, E.Salehi, P.Zhang in 2000:

- Characterization of graphs having partition 2, n and $n-1$.

G.G.Chappell, F.G.Gimbel, C.Hartman in 2005:

- Relation between metric basis, dimension partition and diameter of a graph

J.Diaz, O.Pottonen, M.Serna, E.Jan van Leeuwen in 2012:

- Metric basis for planar graph is NP-Complete.

I.Tomescu in 2008:

- Dimension partition in graph $(\mathbb{Z}^2, \ell_1)$ and $(\mathbb{Z}^2, \ell_\infty)$

F.Rebatel, É.Thiel in 2011:

- Characterization of gauges which have 2 for metric dimension in rectangles,
- Polyhedral gauges do not have finites metric basis in $\mathbb{R}^n$ nor in $\mathbb{Z}^n$.

Numbers in metric basis problem

A metric basis is a subset of k unlabelled objects in n.

Number of basis of size k

$$\binom{n}{k}$$

The number of possible metric basis

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

Numbers in metric basis problem

A metric basis is a subset of k unlabelled objects in n.

Number of basis of size k

$$\binom{n}{k}$$

The number of possible metric basis

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

Numbers in partition dimension problem

The number of partitions of n labelled objects into k non-empty unlabelled subsets is given by:

Stirling number of second kind

$$\left\{ n \atop k \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \ .$$

Numbers in partition dimension problem

The number of partitions of n labelled objects into k non-empty unlabelled subsets is given by:

Stirling number of second kind

$$\left\{ n \atop k \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \, .$$

The number of all partitions is given by:

Bell number

$$B_n = \sum_{k=0}^{n} \left\{ n \atop k \right\} .$$

Comparison with metric basis

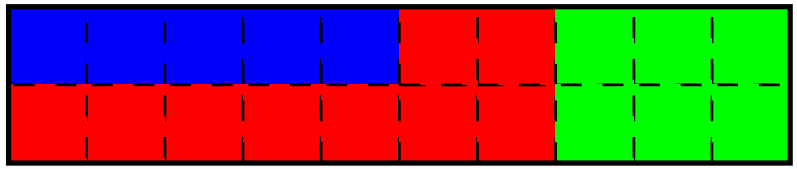
Some Bell's numbers

$B(1) = 1$
$B(5) = 15$
$B(10) = 21\,147$
$B(20) = 5\,832\,742\,205\,057$

Comparison with metric basis

Some Bell's numbers

$B(1) = 1$
$B(5) = 15$
$B(10) = 21\,147$
$B(20) = 5\,832\,742\,205\,057$

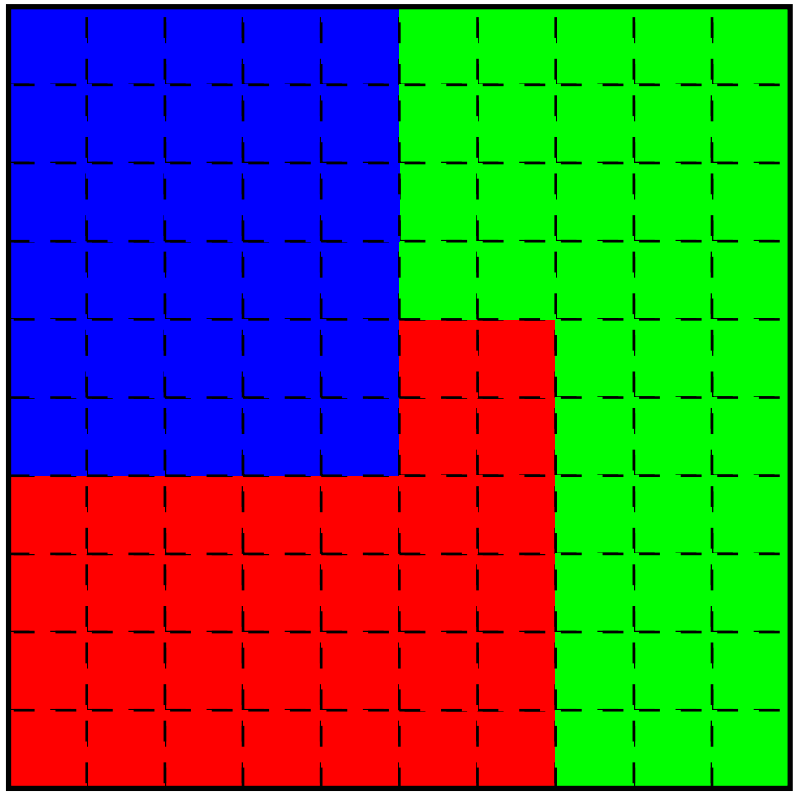


In order to compare, the number of metric basis for a set of 20 points is only $2^{20} = 1\,048\,576$.

Comparison with metric basis

Some Bell's numbers

$B(1) = 1$
$B(5) = 15$
$B(10) = 21\,147$
$B(20) = 5\,832\,742\,205\,057$
$B(100) \approx 10^{116}$



In order to compare, the number of metric basis for a set of 20 points is only $2^{20} = 1\,048\,576$.

Searching resolving partitions

Algorithm 1: MinimumResolvingPartition

Input: $D[][]$: a distance matrix of n points
Output: *Partition* : a resolving partition

```
1 for dim ← 2 to m do
2     Partition ← FirstPartitionOfSize(dim)
3     if IsResolvingPartition(Partition) then
4         return Partition
5     while Partition ← NextPartitionOfSize(dim) do
6         if IsResolvingPartition(Partition) then
7             return Partition
```

Our goal is to improve the IsResolvingPartition(*Partition*) method.

Algorithm 2: GetCoordList compute the list of coordinate vectors

Output: $Coord[\,][\,]$: the list of coordinate vectors
Input: $D[\,][\,]$: the distance matrix ; $Partition = \{p_1, p_2, \ldots, p_m\}$: the partition

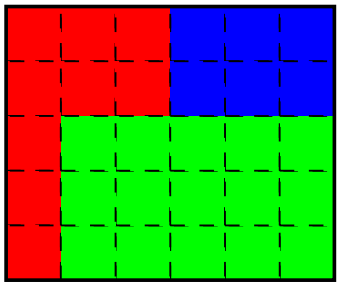
```
1 Initialization.
2 Coord[ ][ ] ← ∞
3 Computing the coordinate vectors.
4 foreach point_A ∈ W do
5     foreach point_B ∈ W do
6         part_B ← Partition[point_B]
7         if Coord[point_A][part_B] > D[point_A][point_B] then
8             Coord[point_A][part_B] ← D[point_A][point_B]
9 return Coord
```

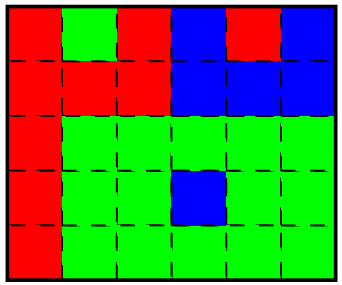
Each point need to compare its distance from each other. Complexity: $O(n^2)$

Distances

Partition class distance

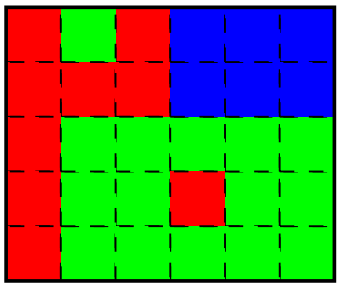
The distance between two classes of partitions is the sum of the difference between their vectors.

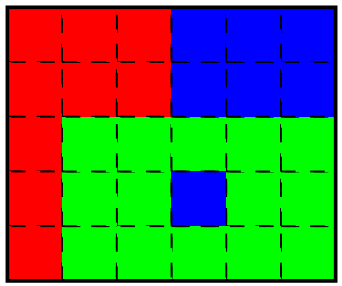




Partition distance

The distance between two partitions is given by the Hamming distance between their vectorial representation.





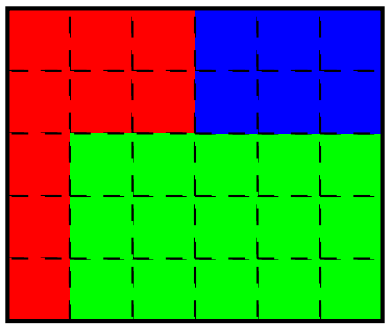
The first three examples belongs to the same class $[15, 9, 6]$, and their partition class distance is 0 to each others. The 4^{th} one belongs to $[14, 9, 7]$. The partition class distance between the top left and bottom right is 2 ; and their partition distance is 1.

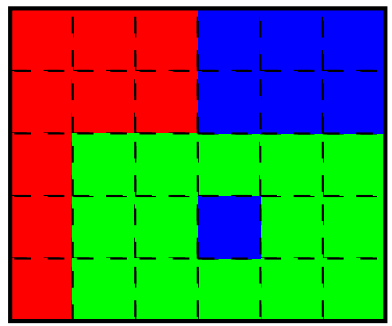
Adjacency

Partition adjacency

Two partitions are called adjacent if :

- their partition distance is 1 (adjacency of the 1^{st} kind)
- they have 0 as partition class distance and 2 as partition distance (2^{nd} kind).





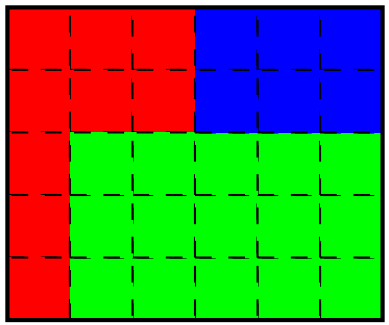
We obtain a partition P' from a partition P with 1 as partition distance by simply changing a point from a part to an other.

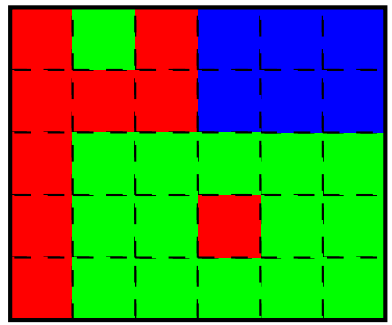
Adjacency

Partition adjacency

Two partitions are called adjacent if :

- their partition distance is 1 (adjacency of the 1^{st} kind)
- they have 0 as partition class distance and 2 as partition distance (2^{nd} kind).





Partitions of the same class with 2 as distance are obtained by simply swapping part of two points.

Improving coordinate vectors computation

We need to know the distance from each point to each part.

Lemma

A point has at most two coordinates which change from a partition to an adjacent one.

Improving coordinate vectors computation

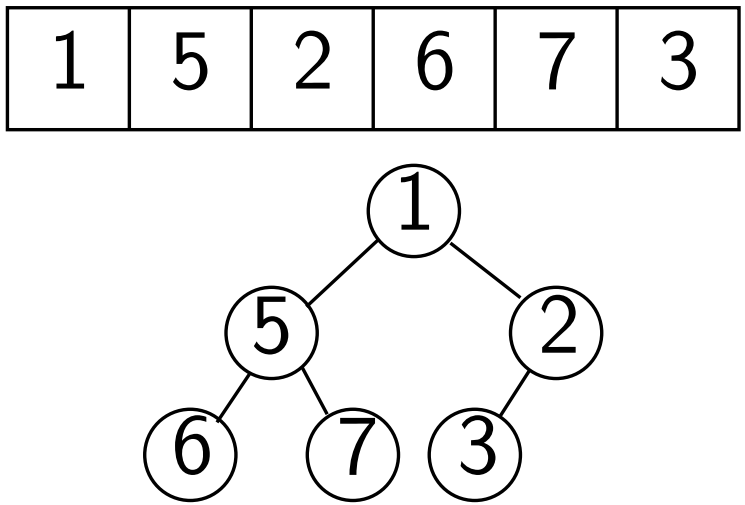
We need to know the distance from each point to each part.

Lemma

A point has at most two coordinates which change from a partition to an adjacent one.

Min-heap properties

- A none must have a lower value than its sons.
- A heap is always compact.
- FindMin is in $O(1)$.
- Insertion and DeleteMin are in $O(\log n)$.



Improving coordinate vectors computation

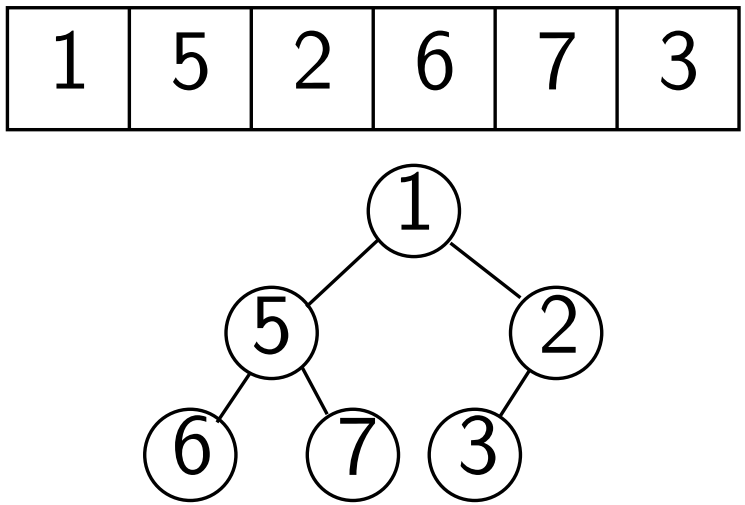
We need to know the distance from each point to each part.

Lemma

A point has at most two coordinates which change from a partition to an adjacent one.

Min-heap properties

- A none must have a lower value than its sons.
- A heap is always compact.
- FindMin is in $O(1)$.
- Insertion and DeleteMin are in $O(\log n)$.



Improving coordinate vectors computation

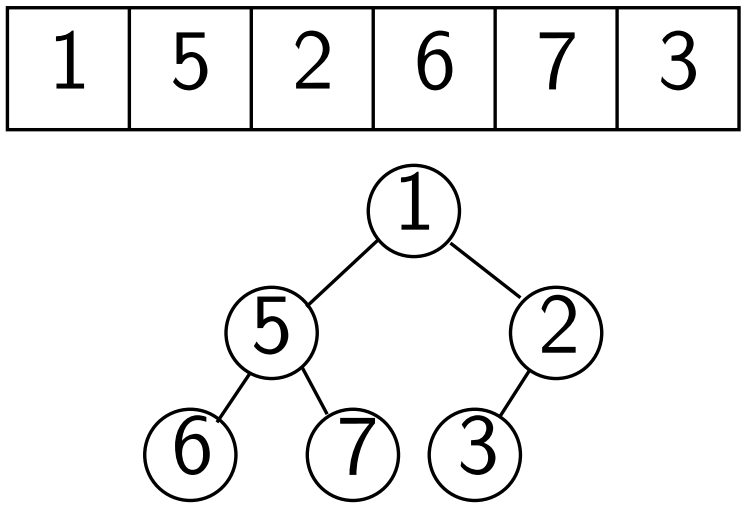
We need to know the distance from each point to each part.

Lemma

A point has at most two coordinates which change from a partition to an adjacent one.

Min-heap properties

- A none must have a lower value than its sons.
- A heap is always compact.
- FindMin is in $O(1)$.
- Insertion and DeleteMin are in $O(\log n)$.



Random deletion with indexed min-heap

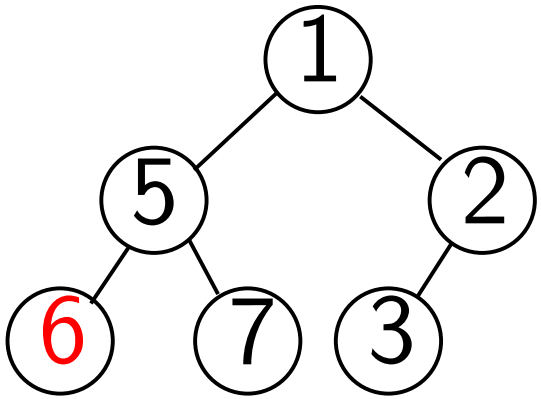
Table view

1	5	2	6	7	3	

Index

1	3	6		2	4	5

Tree view



- Choose a value to delete (ex: 6 at pos. 4).
- Swap the selected node with the last one.
- The node needs to be moved to conserve heap property.
- Finally remove the last node.

Random deletion with indexed min-heap

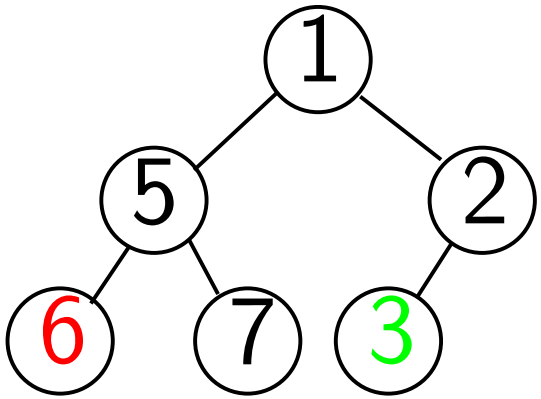
Table view

1	5	2	6	7	3	

Index

1	3	6		2	4	5

Tree view



- Choose a value to delete (ex: 6 at pos. 4).
- Swap the selected node with the last one.
- The node needs to be moved to conserve heap property.
- Finally remove the last node.

Random deletion with indexed min-heap

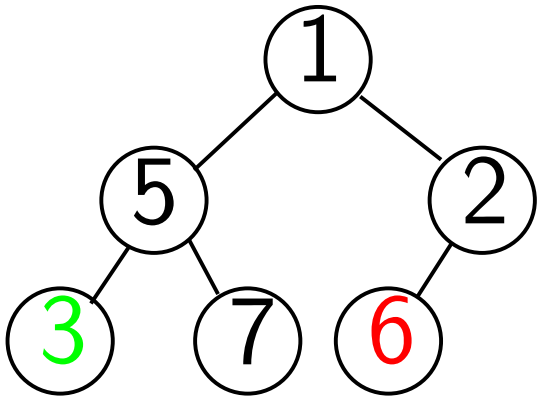
Table view

1	5	2	3	7	6	

Index

1	3	4		2	6	5

Tree view



- Choose a value to delete (ex: 6 at pos. 4).
- Swap the selected node with the last one.
- The node needs to be moved to conserve heap property.
- Finally remove the last node.

Random deletion with indexed min-heap

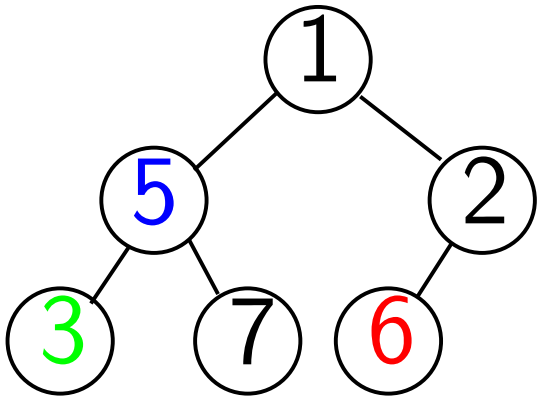
Table view

1	5	2	3	7	6	

Index

1	3	4		2	6	5

Tree view



- Choose a value to delete (ex: 6 at pos. 4).
- Swap the selected node with the last one.
- The node needs to be moved to conserve heap property.
- Finally remove the last node.

Random deletion with indexed min-heap

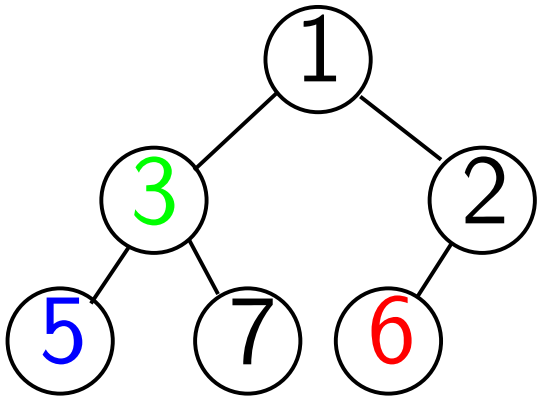
Table view

1	3	2	5	7	6	

Index

1	3	2		4	6	5

Tree view



- Choose a value to delete (ex: 6 at pos. 4).
- Swap the selected node with the last one.
- The node needs to be moved to conserve heap property.
- Finally remove the last node.

Random deletion with indexed min-heap

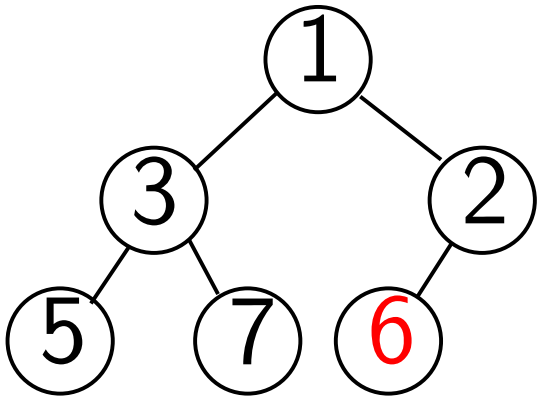
Table view

1	3	2	5	7	6	

Index

1	3	2		4	6	5

Tree view



- Choose a value to delete (ex: 6 at pos. 4).
- Swap the selected node with the last one.
- The node needs to be moved to conserve heap property.
- Finally remove the last node.

Random deletion with indexed min-heap

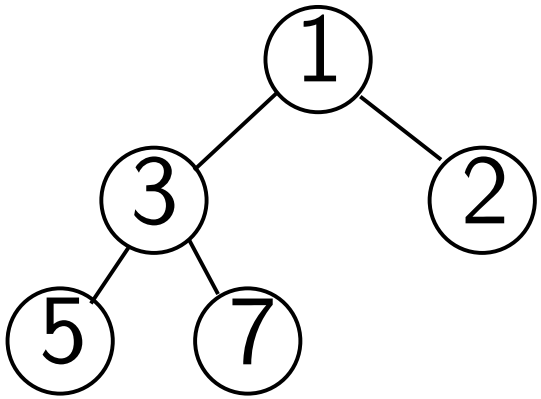
Table view

1	3	2	5	7		

Index

1	3	2		4		5

Tree view



- Choose a value to delete (ex: 6 at pos. 4).
- Swap the selected node with the last one.
- The node needs to be moved to conserve heap property.
- Finally remove the last node.

Using indexed min-heap

- Each min-heap allows FindMin in $O(1)$; Insert and DeleteMin in $O(\log_2 n)$.
- Thanks to the index we define two more operations: DeleteRandom and ChangeKey also in $O(\log_2 n)$.

Lemma

Keeping up-to-date coordinate vectors when switching from a partition to an adjacent one, can be made in $O(n \log_2 n)$.

- Adjacency of 1^{st} kind is computed by an Insert and a DeleteRandom.
- Adjacency of 2^{nd} kind is computed by two ChangeKey.

We can note that points has exactly *dim* min-heap. So when dimension gets higher, complexity of switching between two adjacent partitions is lower.

Using indexed min-heap

- Each min-heap allows FindMin in $O(1)$; Insert and DeleteMin in $O(\log_2 n)$.
- Thanks to the index we define two more operations: DeleteRandom and ChangeKey also in $O(\log_2 n)$.

Lemma

Keeping up-to-date coordinate vectors when switching from a partition to an adjacent one, can be made in $O(n \log_2 n)$.

- Adjacency of 1^{st} kind is computed by an Insert and a DeleteRandom.
- Adjacency of 2^{nd} kind is computed by two ChangeKey.

We can note that points has exactly *dim* min-heap. So when dimension gets higher, complexity of switching between two adjacent partitions is lower.

Using indexed min-heap

- Each min-heap allows FindMin in $O(1)$; Insert and DeleteMin in $O(\log_2 n)$.
- Thanks to the index we define two more operations: DeleteRandom and ChangeKey also in $O(\log_2 n)$.

Lemma

Keeping up-to-date coordinate vectors when switching from a partition to an adjacent one, can be made in $O(n \log_2 n)$.

- Adjacency of 1^{st} kind is computed by an Insert and a DeleteRandom.
- Adjacency of 2^{nd} kind is computed by two ChangeKey.

We can note that points has exactly *dim* min-heap. So when dimension gets higher, complexity of switching between two adjacent partitions is lower.

Using indexed min-heap

- Each min-heap allows FindMin in $O(1)$; Insert and DeleteMin in $O(\log_2 n)$.
- Thanks to the index we define two more operations: DeleteRandom and ChangeKey also in $O(\log_2 n)$.

Lemma

Keeping up-to-date coordinate vectors when switching from a partition to an adjacent one, can be made in $O(n \log_2 n)$.

- Adjacency of 1^{st} kind is computed by an Insert and a DeleteRandom.
- Adjacency of 2^{nd} kind is computed by two ChangeKey.

We can note that points has exactly *dim* min-heap. So when dimension gets higher, complexity of switching between two adjacent partitions is lower.

Using indexed min-heap

- Each min-heap allows FindMin in $O(1)$; Insert and DeleteMin in $O(\log_2 n)$.
- Thanks to the index we define two more operations: DeleteRandom and ChangeKey also in $O(\log_2 n)$.

Lemma

Keeping up-to-date coordinate vectors when switching from a partition to an adjacent one, can be made in $O(n \log_2 n)$.

- Adjacency of 1^{st} kind is computed by an Insert and a DeleteRandom.
- Adjacency of 2^{nd} kind is computed by two ChangeKey.

We can note that points has exactly *dim* min-heap. So when dimension gets higher, complexity of switching between two adjacent partitions is lower.

Comparing vectors

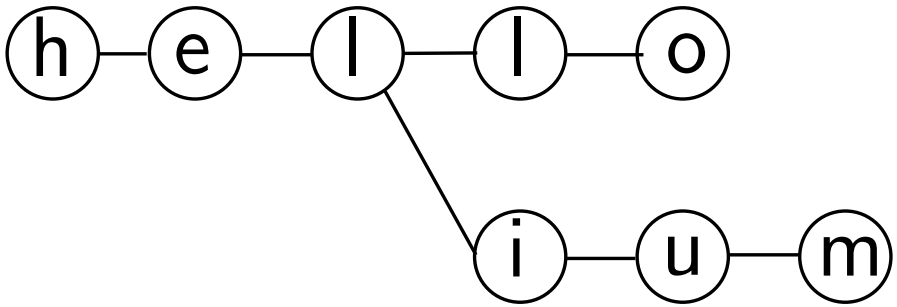
Optimal time

Comparing coordinates can be made in optimal time $O(n \times dim)$ using a *trie*.

Comparing vectors

Optimal time

Comparing coordinates can be made in optimal time $O(n \times dim)$ using a *trie*.



A trie is an ordered tree data structure commonly used to store strings.
Unlike in a binary tree, a node does not represent a key, but a specific prefix for any number of keys.

Using trie

- The depth of the trie is *dim*.
- INSERT and DELETE are linear in the size of the key.
- We know at every moment if they are duplicated coordinates with no comparisons.

Using trie

- The depth of the trie is *dim*.
- INSERT and DELETE are linear in the size of the key.
- We know at every moment if they are duplicated coordinates with no comparisons.

Using trie

- The depth of the trie is *dim*.
- INSERT and DELETE are linear in the size of the key.
- We know at every moment if they are duplicated coordinates with no comparisons.

Using trie

- The depth of the trie is *dim*.
- INSERT and DELETE are linear in the size of the key.
- We know at every moment if they are duplicated coordinates with no comparisons.

Update complexity

In the worst case, every points have some changes in coordinates, then we have to remove all of the vectors of coordinates from the trie and add the new ones. $O(n \times \text{dim})$ operations are needed.

Space complexity

- Each points need *dim* heap + only one index. $O(\dim \times n)$
- An overestimated size for the number of nodes in the trie is that each vector is distinct from others, implying we need $O(n \times \dim)$.
- The size of a node can be approximated by n.

Complexity

Space complexity is $O(n \times \dim + n^2 \times \dim)$

Time complexity

- For any changes we need to apply 1 operation on 2 different heap on each points with cumulated sizes $< n$.
- In the worst case every point have changing coordinates. Need to reconstruct the trie in $O(n \log_2 n)$.

Complexity

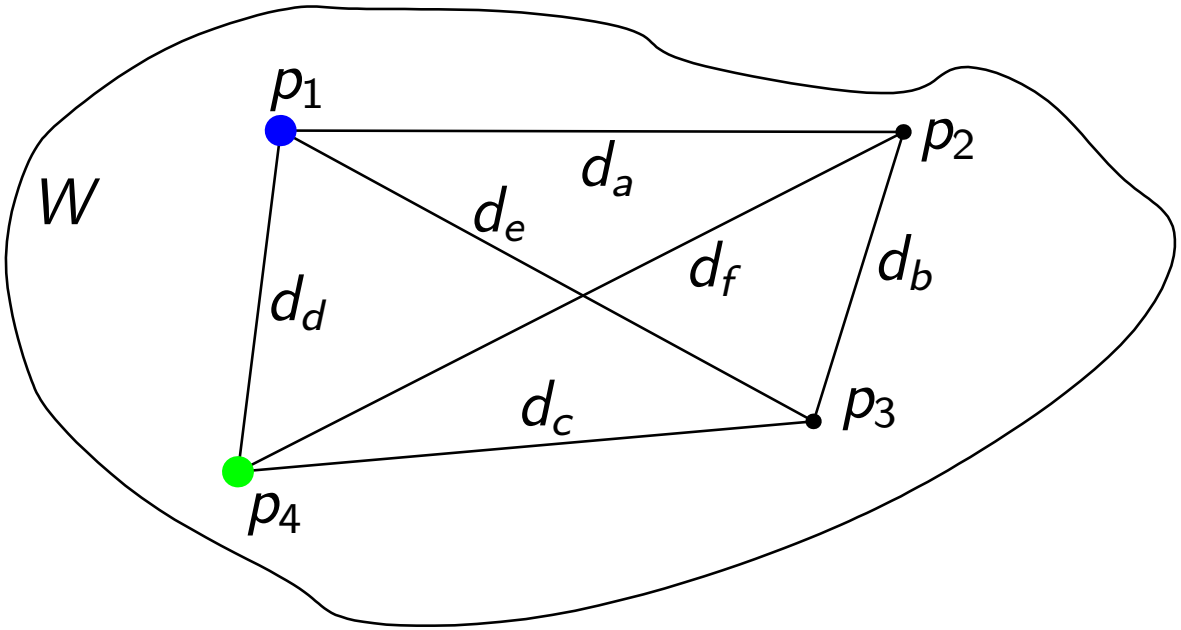
Time complexity is $O(n \times \dim + n \log_2 n)$

Initialisation complexity

$O(n \times \dim + n^2 \log_2 n)$

Tomescu

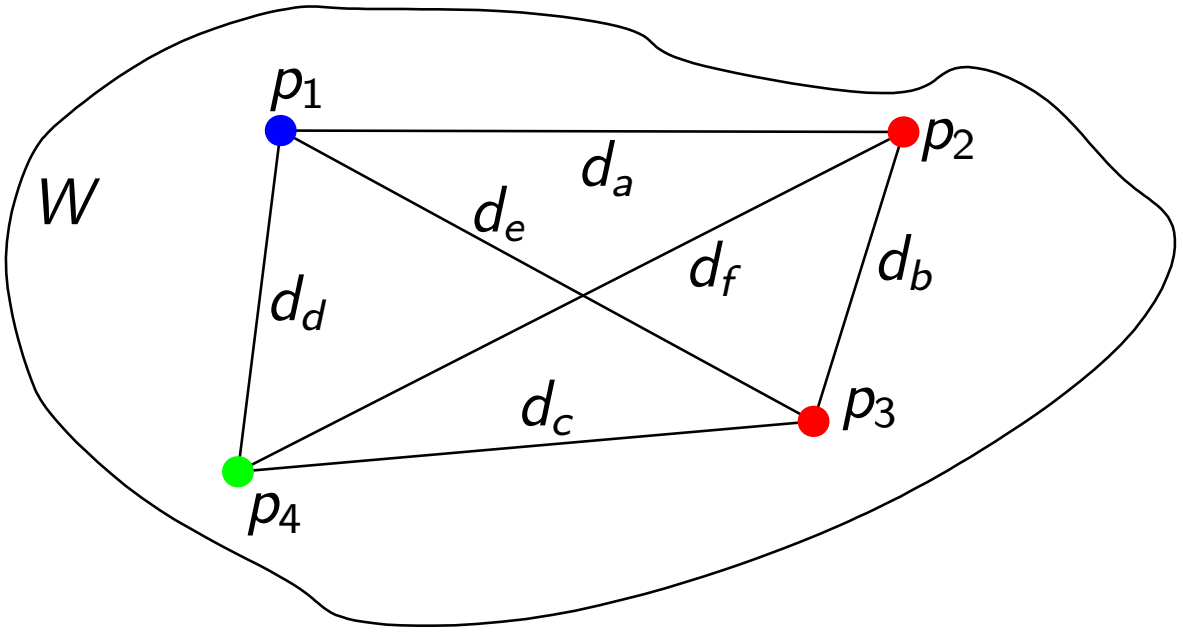
Partition dimension $\leqslant$ Metric dimension $+ 1$



Let (p_1, p_4) be a metric basis for (W, d).

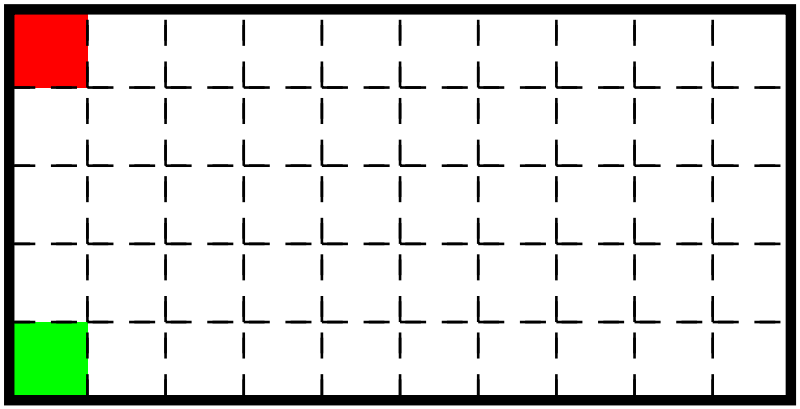
Tomescu

Partition dimension $\leqslant$ Metric dimension $+ 1$



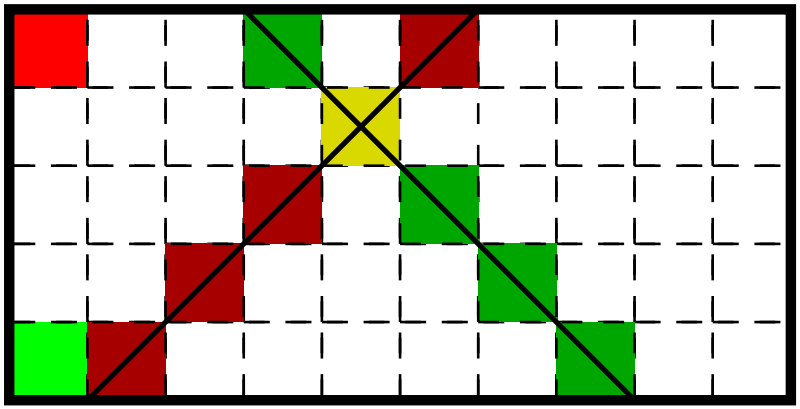
Let (p_1, p_4) be a metric basis for (W, d).
Put all remaning points in a garbage part to obtain a resolving partition $P = (\{p_2, p_3\}, \{p_1\}, \{p_4\})$.

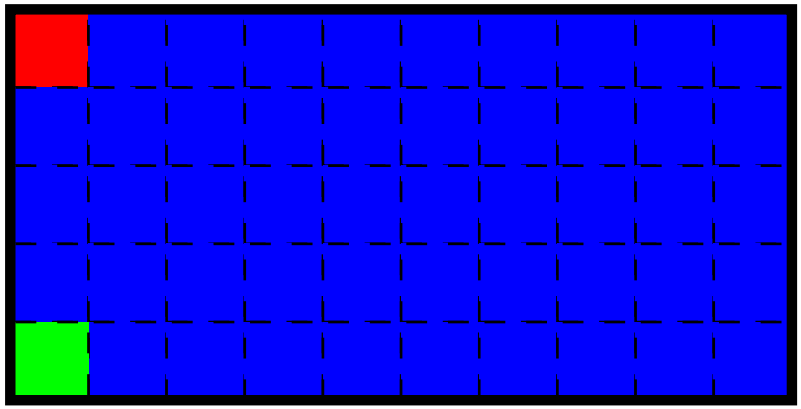
Basis in rectangles for non vertical/horizontal facet gauges



We shown in DGCI 2011 that metric dimension for any gauges which do not have vertical or horizontal facet in their ball (like ℓ_∞) is **2**.

Basis in rectangles for non vertical/horizontal facet gauges



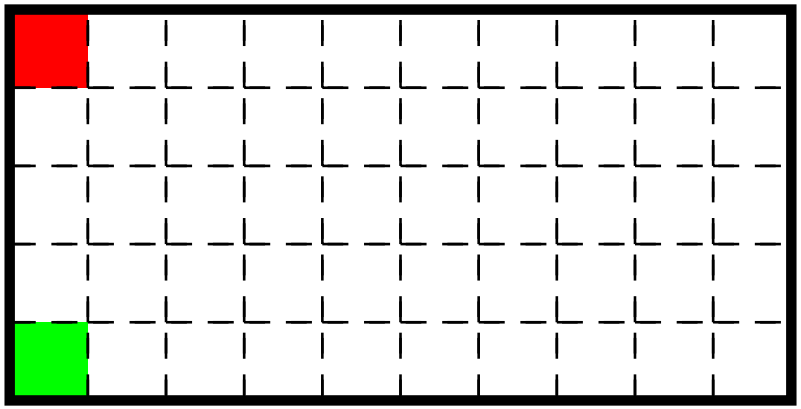


We shown in DGCI 2011 that metric dimension for any gauges which do not have vertical or horizontal facet in their ball (like ℓ_∞) is **2**.

We have by the previous lemma a partition basis for those gauges.

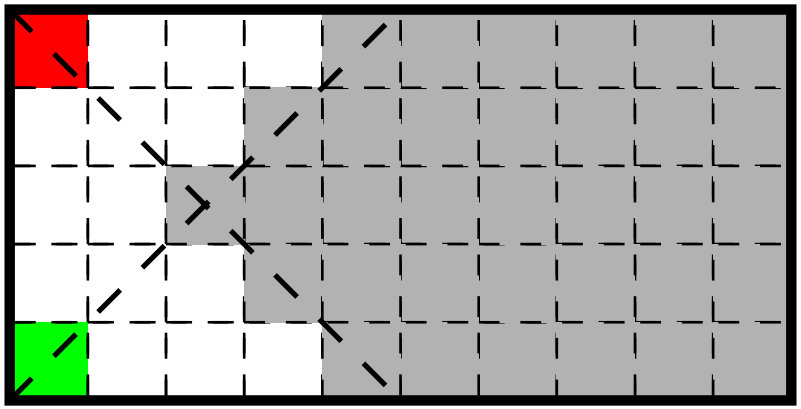
On the example, We have a metric basis/partition basis for ℓ_1.
we show that coordinate vectors are unique for any yellow point.

Partition dimension for others gauges (ℓ_∞)



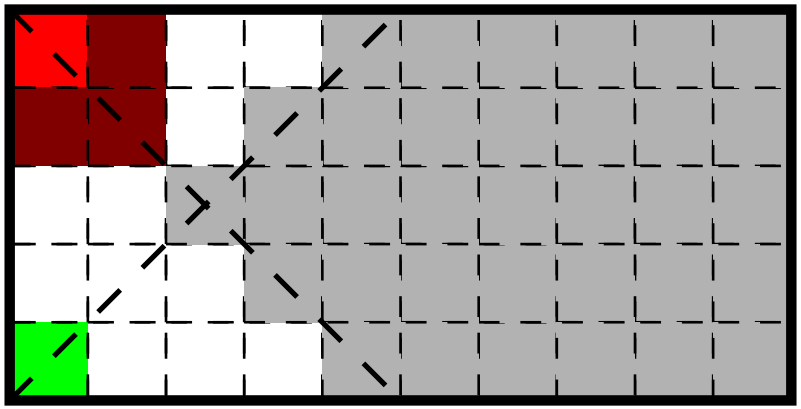
Two points are not a metric basis for ℓ_∞.

Partition dimension for others gauges (ℓ_∞)



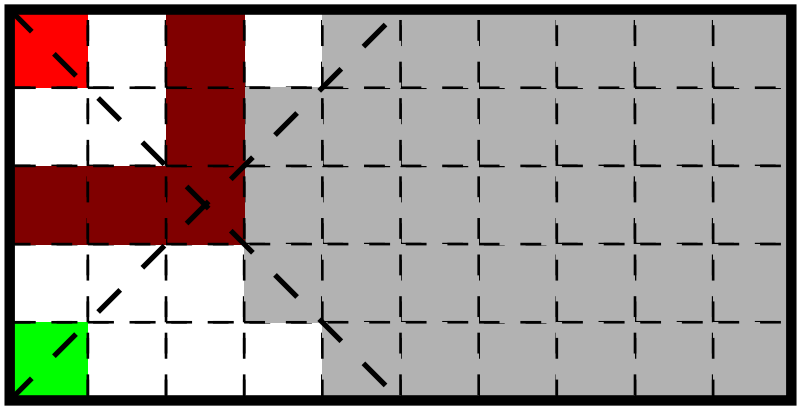
If we draw the axes of its unit ball, we obtain a grey undefined area.

Partition dimension for others gauges (ℓ_∞)



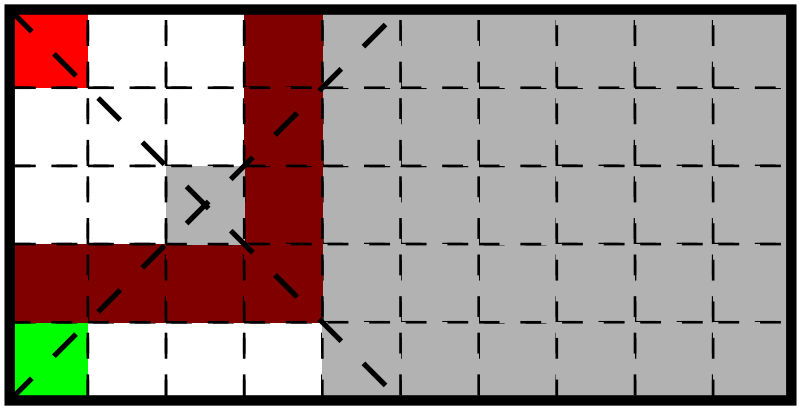
If we grows up the distance from the red and green point until the grey area,

Partition dimension for others gauges (ℓ_∞)



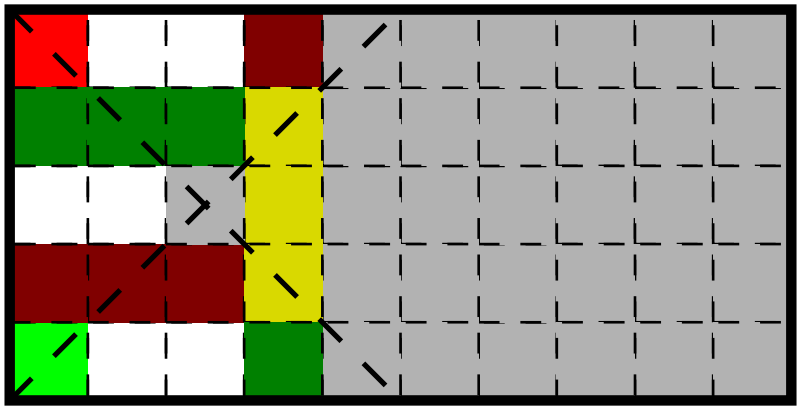
If we grows up the distance from the red and green point until the grey area,

Partition dimension for others gauges (ℓ_∞)



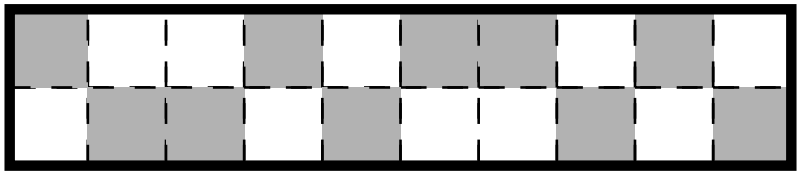
If we grows up the distance from the red and green point until the grey area,

Partition dimension for others gauges (ℓ_∞)

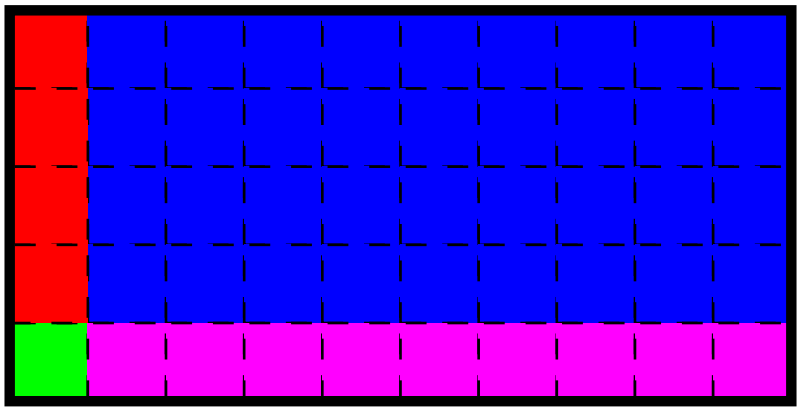


If we grows up the distance from the red and green point until the grey area, yellows points have same coordinates.

Partition dimension for others gauges (ℓ_∞)



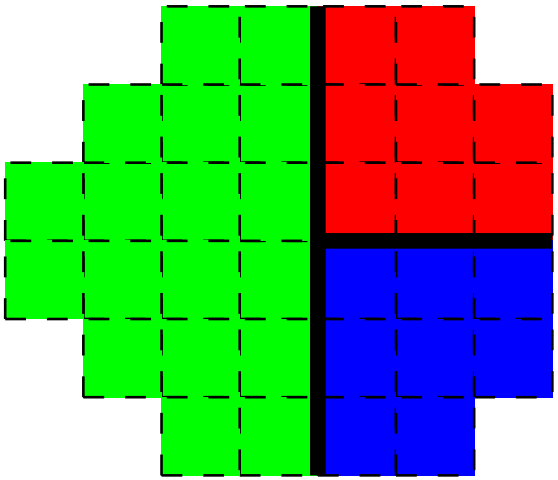
Metric dimension for ℓ_∞ is not a fixed number and can be half of the points of a rectangle in the worst case (rectangle of height 2).

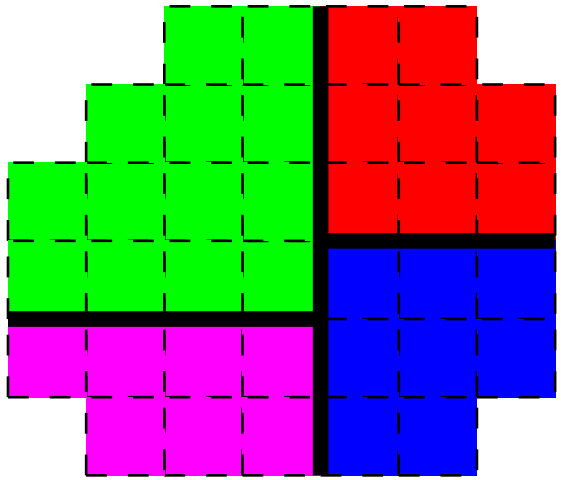


But partition dimension for polyhedral gauges with horizontal/vertical facet in their unit balls (like ℓ_∞) is 4.

Dimension in space $\mathbb{Z}^2$

We proved in DGCI 2011 that polyhedral gauges do not have finite metric basis in $\mathbb{R}^n$ nor $\mathbb{Z}^n$.





Partition Dimension in $\mathbb{Z}^2$

$3 \leqslant \text{PartitionDimension(gauges)} \leqslant 4$

Partition dimension for $(\ell_\infty, \mathbb{R}^2)$ is 3.

Conclusion

Our contributions:

- Analyse of the difficulty of the problem.
- Answering if a partition is resolving more efficiently.
- Partition dimension for gauges in rectangles.
- Partition dimension in unbounded space $\mathbb{Z}^2$ and $\mathbb{R}^2$.

Future prospects:

- Improving enumerating method NextPartition()
- Looking for higher dimensions.
- Studying partition in convex or non-convex shape.