On Dimension Partitions in Discrete Metric Spaces

Fabien Rebatel and Édouard Thiel

Laboratoire d'Informatique Fondamentale de Marseille (LIF, UMR 7279),
Aix-Marseille Université,
163 Avenue de Luminy, Case 901, 13288 Marseille cedex 9, France
{Fabien.Rebatel,Edouard.Thiel}@univ-amu.fr

Abstract. Let (W, d) be a metric space and $S = \{s_1 \dots s_k\}$ an ordered list of subsets of W. The distance between $p \in W$ and $s_i \in S$ is $d(p, s_i) = \min\{\, d(p, q) : q \in s_i \,\}$. S is a resolving set for W if $d(x, s_i) = d(y, s_i)$ for all s_i implies $x = y$. A metric basis is a resolving set of minimal cardinality, named the metric dimension of (W, d). The metric dimension has been extensively studied in the literature when W is a graph and S is a subset of points (classical case) or when S is a partition of W ; the latter is known as the partition dimension problem. We have recently studied the case where W is the discrete space $\mathbb{Z}^n$ for a subset of points; in this paper, we tackle the partition dimension problem for classical Minkowski distances as well as polyhedral gauges and chamfer norms in $\mathbb{Z}^n$.

Keywords: dimension partition, metric dimension, distance geometry, discrete distance, norm.

1 Introduction

The partition dimension of a set is a combinatorial problem which generalizes the metric dimension of a set. These notions are classically studied in graph theory, and more generally in the field of distance geometry. Distance geometry is the characterization and study of sets of points based on the distance values between member pairs. Most of the notions of this field can be studied in the discrete space $\mathbb{Z}^n$, which adds digital and/or Euclidean geometry properties, as well as properties depending on the chosen distance. In this paper, we propose to tackle the partition dimension problem in $\mathbb{Z}^n$ for some classical norms.

Let W be a (finite or infinite) set endowed with a metric d. We first consider the metric dimension and recall some results. Let $S = (v_1, v_2, \dots, v_k)$ an ordered subset of points in W. The *representation* of $p \in W$ with respect to S is the k-tuple $r(p|S) = \{d(p, v_1), d(p, v_2), \dots, d(p, v_k)\}$, also called the coordinates of p. The set S is a *resolving set* for W if every two elements of W have distinct representation. A resolving set having minimal cardinality is called a *metric basis* for W; its cardinality is the *metric dimension* $\dim(W)$ of W.

Harary and Melter gave in [7] the intrinsic metric dimension of any path, complete graph, cycle, wheel and complete bipartite graph. They also proposed

R. Gonzalez-Diaz, M.-J. Jimenez, B. Medrano (Eds.): DGCI 2013, LNCS 7749, pp. 11–22, 2013.

an algorithm for finding a metric basis of a tree T, which gives an explicit formula for the dimension of T. Khuller *et al.* showed in [10] that all graphs having dimension 1 are paths and that there are non-planar graphs having dimension 2. The intrinsic metric dimension of trees can be efficiently solved in linear time, but finding the metric dimension of an arbitrary graph is NP-hard. The dimension of an arbitrary graph with n nodes can be approximated within a factor $O(\log n)$ in polynomial time. Collections of bounds or exact values of metric dimensions are presented in [4][9]. For other results on graphs, see [1][2].

A number of results in digital geometry have also been established. Melter and Tomescu showed in [11] that when W is the digital plane, the metric bases are sets of three non-collinear points for the Euclidean distance d_2, whereas there are no finite metric bases for the city block distance d_1 and the chessboard distance d_∞. The d_1 metric dimension of a n-dimensional axes-parallel rectangle is n [10], and the d_∞ metric dimension of a square is 3. If non axes-parallel rectangles are considered, there exists for both distances a rectangle in the digital plane such that its dimension is n, for any given $n \geqslant 3$. In [12], we have shown that the metric dimension in $\mathbb{R}^n$ is infinite for any polyhedral (or partially polyhedral) central symmetric gauge; the metric dimension in $\mathbb{Z}^n$ is also infinite for any chamfer norms. However, the metric dimension is finite in axes-parallel rectangles for both kind of spaces and distances.

The notion of partition dimension has been proposed in [5][6]. We recall the definition in a more general manner: let $S = \{s_1 \dots s_k\}$ be an ordered list of subsets of W. The distance between $p \in W$ and $s_i \in S$ is $d(p, s_i) = \min\{\, d(p, q) : q \in s_i \,\}$. The representation of p with respect to S is $r(p|S) = \{d(p, s_1), \dots, d(p, s_k)\}$. As previously, S is a resolving set for W if every two elements of W have distinct representation.

When S is a set of points, we turn back again to the metric dimension problem. If S is a partition of W, we obtain new objects: a partition (metric) base is a resolving set of minimal cardinality, called the partition dimension $\mathrm{pd}(W)$. For any nontrivial connected graph G we have $\mathrm{pd}(G) \leqslant \dim(G) + 1$ [5]. It should be noted that the partition dimension may be much smaller than the metric dimension [13]. In particular, the partition dimension of the digital plane with respect to d_1 and d_∞ is 3 and 4, respectively. For other relationships between metric and partition dimension and graph diameter, as well as results on coloring and chromatic number, see [3].

The paper is organized as follows. In section 2, we define partition classes and study the combinatorics of the partition dimension problem. Then, in section 3, we present algorithms allowing to decide if a partition is resolving, and we evaluate their complexities. Next in section 4, we present specific properties of partition dimensions for gauges in Euclidean and digital spaces. We finally conclude in section 5.

2 Preliminaries

In this part, we reformulate the problem of partition dimension to understand the relationship between the latter and the metric basis problem. Moreover, we express their differences in combinatorics.

We first introduce a simple example, in which W is the set of elements $(e_1, e_2, e_3, e_4, e_5)$. Each partition can be expressed in several manners. For instance, the partition $\{\{e_2\}, \{e_3, e_1, e_5\}, \{e_4\}\}$, is equivalent to the partition $\{\{e_1, e_3, e_5\}, \{e_2\}, \{e_4\}\}$. To avoid ambiguity, we write each of them using the canonical notation defined as follows: in each part, the elements are sorted by their index; the parts are sorted by decreasing cardinal; parts having same cardinal are sorted in lexicographic order.

2.1 Partition Class

Given a partition we define its *partition class* as the vector of cardinalities of its parts. In our example, the partition $\{\{e_1, e_3, e_5\}, \{e_2\}, \{e_4\}\}$ belongs to the class $[3, 1, 1]$.

The set of partition classes corresponds to the *integer partition*. The number of partitions of an integer n is given by the function $p(n)$ [8]. The function $p(n)$ grows extremely fast: as the first values for $n = 1 \dots 9$ are $1, 2, 3, 5, 7, 11, 15, 22, 30$, we have $p(100) = 190\,569\,292$.

Given a metric basis $M = \{m_1, \dots, m_n\}$, we can obtain a partition basis $P = \big\{W \backslash M, \{m_1\}, \dots, \{m_n\}\big\}$ written in canonical form. The partition classes for metric basis problem are those which have at most one class with more than one element (i.e. $[x, 1, \dots, 1]$, $x \geqslant 1$). It only exists one class of partition representing a metric basis for each dimension of partition. This means that for a set of 100 points, there are only 100 classes of metric basis among the 190 millions available classes.

Definition 1. *The* distance between two classes of partition *is the sum of the difference between the coordinates of their vectors:*

$$d(C_1, C_2) = \sum_i \Big| C_1[i] - C_2[i] \Big| \,.$$

If both vectors do not have the same number of part, the smallest one is completed by parts of size zero. The coordinate vectors are completed by zeros if needed. The distance between classes of partitions is a ℓ_1 distance between vector coordinates. We remark that two partition classes with a distance of 1 do not exist for a given number of points. Indeed, the only way to have a distance of 1 between two classes is to have only one part where the number of points is different, so the total number of points is not the same.

Two classes are called *adjacent* if their distance is 2, which means that there are only two parts whose cardinals differ, one gaining a point and one missing a point.

We define a distance between two partitions as follows:

Definition 2. *The* distance between two partitions *is given by the Hamming distance of their coordinate vectors.*

Once again, if partitions do not have the same size, the smallest one is padded by empty parts (Hamming distance is defined for two vectors of the same size). Two partitions will be named *adjacent* if one of the properties is fulfilled:

i) their partition distance is 1;
ii) the partition class distance is 0 and the partition distance is 2.

2.2 Counting

The number of partitions of n labeled objects into k non-empty unlabeled subsets is given by the *Stirling number of the second kind* $\left\{ {n \atop k} \right\}$ (see sequence A008277 in OEIS):

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

This number is important to evaluate the difficulty to find a resolving partition of n points for a given dimension k. If we look only at metric basis classes, the number of partitions is reduced to the binomial coefficient $\binom{n}{k}$.

Now in the case where we need to enumerate all of the possible partitions (up to size n), the cardinal will be the sum

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

known as the *Bell number* B_n (see sequence A000110 in OEIS).

The first 10 numbers of Bell are $1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147$; we have $B(20) = 5\,832\,742\,205\,057$, and $B(100)$ is given by a 116 digital number. The number of possible metric basis is given by the sum of the binomial coefficient $\sum_{i=0}^{n} \binom{n}{i} = 2^n$. As a comparison, $2^{20} = 1\,048\,576$ is much smaller than $B(20)$.

An other interesting comparison can be made between classes of the same dimension. The number of partition in each classes can be computed by this recursive function:

$$\#[\, p_i, p_{i+1}, \ldots, p_k \,] = \frac{1}{A_i} \binom{\sum_{j=i}^{k} p_j}{p_i} \#[\, p_{i+1}, \ldots, p_k \,] \ ,$$

where A_i is the number of parts in $[\, p_i, p_{i+1}, \ldots, p_k \,]$ which value equals p_i.

The number of classes of dimension n is highly unevenly distributed; the cardinal of the metric bases class is minimal and is far smallest than the cardinal of the other classes.

3 Algorithm

In this section we describe some methods to determine if a given partition is resolving. We consider $W = (e_1, e_2, \ldots, e_k)$ and we denote $D[e_i][e_j]$ its distance matrix between each pair of points.

The notation $\{p_1, p_2, \ldots, p_k\}$ formalizes a representation of a partition where e_i belongs to the part p_i. The dimension of the partition is by definition given by $dim = \max_i (p_i)$.

Algorithm 1: MinimumResolvingPartition

```
Input: D[][] : a distance matrix of n points
Output: Partition : a resolving partition
for dim ← 2 to m do
    Partition ← FirstPartitionOfSize(dim)
    if IsResolvingPartition(Partition) then
        return Partition
    while Partition ← NextPartitionOfSize(dim) do
        if IsResolvingPartition(Partition) then
            return Partition
```

The first algorithm that we give is simple and natural. Then we show several properties, helping to give an improved version with lower complexity.

3.1 Natural Method

Our aim is to determine if for a given partition, all points of W have different coordinates. We first compute all of the coordinates with the following algorithm:

Algorithm 2: GetCoordList computes the list of coordinate vectors

```
Output: Coord[][] : the list of coordinate vectors
Input: D[][] : the distance matrix ; Partition = {p_1, p_2, ..., p_m} : the
       partition
Initialization.
Coord[][] ← ∞
Computing the coordinate vectors.
foreach point_A ∈ W do
    foreach point_B ∈ W do
        part_B ← Partition[point_B]
        if Coord[point_A][part_B] > D[point_A][point_B] then
            Coord[point_A][part_B] ← D[point_A][point_B]
return Coord
```

Each coordinate of a point q is given by the minimum distance from q to points of the corresponding part. Every points belong to a part, so we need to compute the distance between each couple of points. The complexity is $O(n^2)$, where n is the number of points.

Now we need to know if all points are distinguished by coordinates. The answer is given by using a sorting algorithm that stops and returns *false* if two vectors are identical, but returns *true* if it ends sorting. Logically, the complexity of this step is in $O(dim * n \log_2(n))$ comparisons of n vectors of size *dim*.

Finally this method requires $O(n^2 + dim * n * \log_2(n))$ to determine if the partition is resolving.

3.2 Computing Coordinate Vectors in Efficient Time

Lemma 1. *Only two coordinates may change for each point from a partition to an adjacent one.*

Proof. Coordinates are given by a distance to a part (a set of points), so if this part does not change, the coordinate remains. Adjacent partitions are either in the same class with a distance of 2, which means that two points exchanged their part number, or partitions are in different classes with a distance 1, meaning that a single point has changed its part number. In both cases, two parts have changed, so two coordinates may change for any given point. □

Lemma 2. *Computing coordinate vectors for a given partition, knowing the coordinate vectors of an adjacent partition, can be made in $O(n \log_2(n))$ time.*

In order to prove this lemma, we explain the algorithm and show its complexity. We use a data structure to maintain up-to-date some information for a partition. Each point needs to know an ordered list by distance to each point of each part. The more convenient structure is an indexed binary heap.

A heap is a classical data structure in computer science. This structure is based on the *heap property* for a binary tree: if the node A is a parent of the node B then we have $key(A) < key(B)$.

A binary heap has also the property to be always compact, i.e. all the leafs of the tree have their depths different by at most 1. In an n elements heap, the depth is $\log_2 n$.

Operations like FindMin, in constant time $O(1)$, DeleteMin and Insert, in $\log_2 n$ time, are classically defined for min-heap. We need to define two new operations: DeleteRandom and ChangeKey.

The first one is provided by an index for points in a heap. We can directly find the location of a key in a heap in constant time $O(1)$. Without the index, finding a random key in a heap may take $O(n)$ operations.

ChangeKey can be decomposed in two operations: IncreaseKey or DecreaseKey. Decreasing a key in a min-heap is done by switching a node by its father while the father's key is greater than the node's one. IncreaseKey is the reverse operation: it consists in exchanging the node by its smaller son until the son's key is greater or equal to its own key.

DeleteRandom, which removes a random node r, can be easily defined knowing ChangeKey, in two steps. First, we switch the node r with the last node l of the heap, knowing that the last node is always removable. Second, we consider that the kept node l has just changed its key.

These two operations are made in the worst case in $O(\log_2(n))$ time.

For each point of W, we keep up-to-date *dim* min-heaps, one for each parts of *Partition*. The coordinate vectors are given through a series of FindMin.

There exist two kinds of adjacent partitions. The first one is two partitions of the same class with distance of 2, i.e. two points from the first partition switch their parts number to become the second partition. The second kind is constituted by partitions having distance of 1, i.e. going from the first one to the second one is simply made by picking a point from a part and moving it to an other part.

Considering adjacent partitions of the first kind, the operation can be assimilated to changing a key of a point. Indeed, removing a point from a heap, then directly adding an other point is equivalent to consider that the key of the leaving point is changed by the key of the incoming point.

This operation requires two calls of ChangeKey for each point, so its complexity can be approximated by $O(n \log_2(n))$ operations. This is an upper bound because the number of points contained in two heaps is always smaller than n since each point belongs to one and only one heap.

Considering now the second kind of adjacent partitions, the update operations are made by a call of DeleteRandom on the heap containing the leaving node, and a call of Insert into an other part.

The complexity of this update is in $O(n \log_2(n))$, which is an upper bound for the same reasons explained just before.

3.3 Comparing Coordinate Vectors in Efficient Time

We now show how to improve the step of determining if two coordinate vectors are identical.

Lemma 3. *Two points of distinct parts of a partition are always distinguished.*

Proof. Every points of a part are at zero distance from it, while outer points have positive distance from it. Thus coordinates are different. □

Thanks to the lemma 3, we could improve the second part of the sorting algorithm, described in section 3.1. Indeed, the sorting step can be split into *dim* independent subproblems.

Lemma 4. *Comparing coordinate vectors for a given partition knowing the coordinate vectors can be made in optimal time $O(n * dim)$ (the size of the coordinate vectors).*

All of the coordinates are bounded by $[0, d_{max}]$, where $d_{\max}$ is the diameter of W. Since we restrict to discrete distances, which have integer values, the maximum number of distinct distances is $d_{\max}$. This number is lower than n for rectangles in $\mathbb{Z}^k$, due to the fact that $\mathbb{Z}^k$ is a module (a discrete vector space) and the maximum number of distinct vectors (up to a symmetry) is bounded by the number of vectors from a corner. This is also true in general case (not

a rectangle) for distances d_1 and d_∞, for which d_{max} is exactly the number of different possible distance values.

Considering this, we can use a specific data structure: a *trie*. A trie (or *prefix tree*) is an ordered tree data structure commonly used to store strings. In this specific structure, unlike in a binary tree, a node is not associated to a specific key but the position of the node in the trie determines a specific prefix of the key.

In our case, we define a node as a structure containing two fields: a table of size d_{max} pointers of nodes, and an integer which indicates the current number of nodes contained in the table. This allows us to create a table containing all the coordinate vectors where the depth i in the trie corresponds to the coordinate i in a vector. Each leaf of the created trie will have a depth of *dim*, considering that the coordinates are given by the parts of a partition.

This data structure provides InsertKey, DeleteKey and FindKey in $O(m)$ where m is the size of the key. Finally, we can search in optimal time for identical coordinate vectors, because we only consider once each vector and each of its coordinates.

3.4 Complexity

We just described structures and procedures to improve the research of a resolving partition using properties of local changes; we now study the complexity in time and space.

We have previously seen that each point needs a min-heap indexed structure of size $2n$ for each of the *dim* part, and finally a trie composed by nodes of size n. A larger upper bound of the number of nodes is given by the number of points n multiplied by the depth *dim* in the trie and the size of a node. We consider $sizeof(node) \approx n$ because we are interested in discrete geometry problems and then we can use an index table for distances in order to use less memory as possible by compacting the table pointer of each node.

So the complexity in space is $O(n * dim + n^2 * dim)$.

Analyzing the complexity in time is done in two steps: analyze the initialization, then analyze the looping steps (given by the described procedures).

For recall, a looping step is composed in two stages: updating structures in order to change the coordinate vectors from a partition to an adjacent one; then updating the trie in order to search undistinguished points. For each of the n points, 2 min-heap with combined size less than n will be updated. Updating the trie is in the worst case: changing every of the n key of size *dim*, by removing a key and adding a new one.

So the complexity in time for local changes is $O(n \log_2 n + n * dim)$.

Time complexity for the initialization is bigger because it is not a local change, indeed every point will be attributed to a part and all parts will change.

In the worst case every points are added to the same part, so for each point, we will create a min-heap of size n by successively pushing each point. This is done in $O(n \log_2 n)$. Once made, the coordinate vectors are added to the trie, so time complexity for initialization is $O(n^2 \log_2 n + n * dim)$.

4 Dimension for Gauges

We have seen in previous section that searching partition dimension by enumeration is actually time consuming. Enumerating is not mandatory for simple cases; we establish in this section the resolving partitions for common distances in rectangles and convex shapes.

We briefly recall the definition of a gauge. 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 $x \in \lambda\mathcal{C}$. Formally, $\gamma_{\mathcal{C}}(x) = \inf\{\lambda \in \mathbb{R}_+ : x \in \lambda\mathcal{C}\}$. When $\mathcal{C}$ is a polyhedron, $\gamma_{\mathcal{C}}$ is called a polyhedral gauge. By definition, all norms are gauges for their unit ball. Conversely, a gauge for $\mathcal{C}$ is a norm, denoted $n_{\mathcal{C}}$, iff $\mathcal{C}$ is central-symmetric. We call *distance gauge*, denoted $d_{\mathcal{C}}$, the metric induced by a central-symmetric gauge $\gamma_{\mathcal{C}}$. We only consider central-symmetric gauges in the paper.

4.1 Dimension in Rectangles

Lemma 5. *The partition dimension of gauges in a rectangle is 3 or 4.*

We have shown in [12] that the metric dimension for gauges is always 2 in a rectangle except for the polyhedral gauges which have a vertical or horizontal facet in their ball. Hence, the partition dimension is at most $2+1$ for that kind of gauges. For the others gauges we can provide a simple pattern which always gives a resolving 4-partition (see figure 1): the first part is the bottom-left corner of the considered rectangle; the second part is the remaining points of the bottom line; the third part is the remaining points of the first column; the fourth part is constituted by all other points.

Note that the coordinates given by the fourth part are useless for the resolving problem, but are necessary to get an actual partition of the rectangle.

The lemma is still valid for non-gauge distances whose balls are all convex.

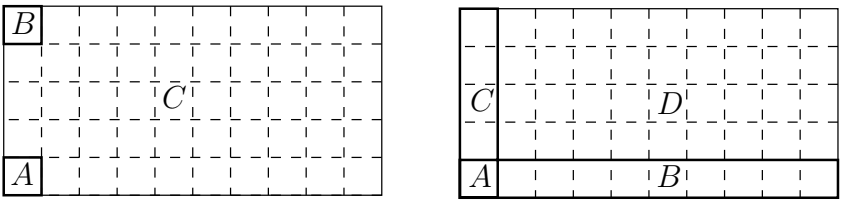

Fig. 1. Illustration of the lemma 5. On the left a resolving 3-partition of the rectangle for gauges which do not contain vertical nor horizontal facet built using metric basis pattern (see [12]). On the right, the pattern used to prove that any convex distance has at least a resolving 4-partition in a rectangle.

4.2 Dimension in Convex Sets

Tomescu studied partition dimension in space $\mathbb{Z}^2$ for d_1 and d_∞. Here we extend that study to every gauges, and to another unusual family of distances: the non-homogeneous distances whose every balls are convex.

Lemma 6. *In the digital plane $\mathbb{Z}^2$, the partition dimension for any gauge which has no vertical nor horizontal facet is* 3.

A proof of this lemma is given using the same pattern proposed by Tomescu: a perpendicular split of the plane in three areas, as follows. A is composed by points verifying $(y < 0)$; B is the set of points corresponding to $(y \geqslant 0 \text{ and } x < 0)$; finally C is the set of the remaining points $(y \geqslant 0 \text{ and } x \geqslant 0)$. Using lemma 3 we need to prove that the points inside an area are distinguished (see figure 2).

- Points in B are distinguished: considering the line $y = -1$ as a centre of a distance ball which gives the coordinate distance from the part A to the part B, the frontier of the ball consists in a vertical line moving through B. Distance from the C area are given by an horizontal line parallel to the intersection between B and C. Intersection of both of the balls line results in a single point, so each point in B has unique coordinates.
- Points in C are distinguished: for the exact same reasons explained for the B area (B and C play symmetric roles), every point in C has unique coordinates.
- Points in A are distinguished: in order to prove this, we look at the upper part of A where points are closer to the C area. In this part, the front line of the C ball consists in a vertical line, but the front line of the B ball is a strictly monotonic curve which does not have any vertical not horizontal part because we use gauges with these properties, and so intersection between both of this front line results in a single point. The same result happens in the lower part of A by symmetries of the B and C part. □

Lemma 7. *In the digital plane $\mathbb{Z}^2$, the partition dimension for any convex distance is lower than* 4.

To prove that we also use the same pattern that Tomescu used to prove the dimension of d_∞ in $\mathbb{Z}^2$. It is nearly the same pattern used for demonstrating the previous lemma using an additional area D which takes off the A area of its lower points $(y < -1)$, see figure 2.
It is clear that in this configuration, points in the areas C and D have unique coordinates for the same reasons explained to prove uniqueness properties in the area B and C in the previous lemma.

Both of the previous lemma 6 and 7 remain valid in the continuous space $\mathbb{R}^2$, but a stronger property exists: indeed we have

Lemma 8. *In the continuous space $\mathbb{R}^2$, the partition dimension for d_∞ is 3.*

Proof. Applying a 45° rotation to the perpendicular pattern, we get the exact same case presented for $dim(d_1, \mathbb{Z}^2)$ (up to the rotation). □

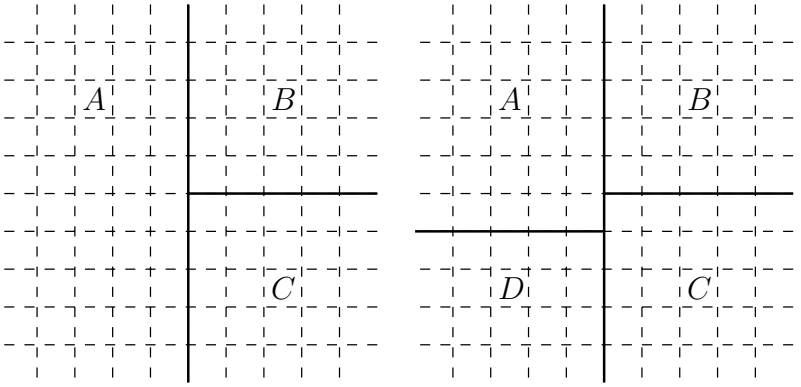

Fig. 2. On the left, illustration of the pattern used to prove the lemma 6. This always gives a resolving 3-partition of the space $\mathbb{Z}^2$ for gauges which do not contain vertical nor horizontal facet. On the right, the pattern used to prove the lemma 7. That pattern gives a resolving 4-partition for any convex distance in $\mathbb{Z}^2$.

Then it is clear that

Corollary 1. *For any distance d,* $dim(d, \mathbb{Z}^2) \geqslant dim(d, \mathbb{R}^2)$.

Considering lemma 8, we finally conjecture that

Conjecture 1. For any convex distance d, we have $dim(d, \mathbb{R}^2) = 3$.

5 Conclusion

In this paper, we have studied the partition dimension problem in the Euclidean and discrete spaces, for usual distance functions used in digital geometry. We have established combinatorial results for the problem, then we have proposed algorithms verifying the resolving property of a partition and stated their complexity. We have finally presented resolving partitions and bounds for geometrical cases.

The procedures presented in the algorithmic section are applied for computing an enumeration algorithm. They could also be applied for a randomized local search algorithm, because the adjacency notion is exactly the notion of locality required by that kind of algorithm.

Unlike for the metric basis, the partition dimension seems to be only related to the distance, while being independent from the considered convex W. The combinatorics of the number of possible partitions explodes with the size of W; notwithstanding, the partition dimension is bounded by a low number, which makes the enumeration achievable.

References

1. Buczkowski, P., Chartrand, G., Poisson, C., Zhang, P.: On k-dimensional graphs and their bases. Periodica Mathematica Hungarica 46, 9–15 (2003)
2. Cáceres, J., Hernando, C., Mora, M., Pelayo, I., Puertas, M.: On the metric dimension of infinite graphs. Electronic Notes in Discrete Math. 35, 15–20 (2009)
3. Chappell, G.G., Gimbel, J.G., Hartman, C.: Bounds on the metric and partition dimensions of a graph. Ars Comb. 88, 349–366 (2008)
4. Chartrand, G., Eroh, L., Johnson, M., Oellermann, O.: Resolvability in graphs and the metric dimension of a graph. Discrete Applied Math. 105(1-3), 99–113 (2000)
5. Chartrand, G., Salehi, E., Zhang, P.: On the partition dimension of a graph. Congressus Numerantium 131, 55–66 (1998)
6. Chartrand, G., Salehi, E., Zhang, P.: The partition dimension of a graph. Aequationes Mathematicae 59, 45–54 (2000)
7. Harary, F., Melter, R.: On the metric dimension of a graph. Ars Combinatoria 2, 191–195 (1976)
8. Hardy, G., Wright, E.: An introduction to the theory of numbers, 5th edn. Oxford University Press (October 1978)
9. Hernando, C., Mora, M., Pelayo, I., Seara, C., Cáceres, J., Puertas, M.: On the metric dimension of some families of graphs. Electronic Notes in Discrete Mathematics 22, 129–133 (2005)
10. Khuller, S., Raghavachari, B., Rosenfeld, A.: Landmarks in graphs. Discrete Applied Mathematics 70(3), 217–229 (1996)
11. Melter, R., Tomescu, I.: Metric bases in digital geometry. Computer Vision, Graphics, and Image Processing 25(1), 113–121 (1984)
12. Rebatel, F., Thiel, É.: Metric Bases for Polyhedral Gauges. In: Debled-Rennesson, I., Domenjoud, E., Kerautret, B., Even, P. (eds.) DGCI 2011. LNCS, vol. 6607, pp. 116–128. Springer, Heidelberg (2011)
13. Tomescu, I.: Discrepancies between metric dimension and partition dimension of a connected graph. Discrete Mathematics 308(22), 5026–5031 (2008)