Algorithmica (2023) 85:1850–1882
https://doi.org/10.1007/s00453-022-01073-0

Complexity and Approximation for Discriminating and Identifying Code Problems in Geometric Setups

Sanjana Dey[1,6] · Florent Foucaud[2,3,4] · Subhas C. Nandy[1] · Arunabha Sen[5]

Received: 3 June 2021 / Accepted: 15 November 2022 / Published online: 19 December 2022

Abstract
We study geometric variations of the discriminating code problem. In the *discrete version* of the problem, a finite set of points P and a finite set of objects S are given in $\mathbb{R}^d$. The objective is to choose a subset $S^* \subseteq S$ of minimum cardinality such that for each point $p_i \in P$, the subset $S_i^* \subseteq S^*$ covering p_i satisfies $S_i^* \neq \emptyset$, and each pair $p_i, p_j \in P$, $i \neq j$, we have $S_i^* \neq S_j^*$. In the *continuous version* of the problem, the solution set S^* can be chosen freely among a (potentially infinite) class of allowed geometric objects. In the 1-dimensional case ($d = 1$), the points in P are placed on a horizontal line L, and the objects in S are finite-length line segments aligned with L (called intervals). We show that the discrete version of this problem is NP-complete. This is somewhat surprising as the continuous version is known to be polynomial-time solvable. This is also in contrast with most geometric covering problems, which are usually polynomial-time solvable in one dimension. Still for the 1-dimensional discrete version, we design a polynomial-time 2-approximation algorithm. We also design a PTAS for both discrete and continuous versions in one dimension, for the restriction where the intervals are all required to have the same length. We then study the 2-dimensional case ($d = 2$) for axis-parallel unit square objects. We show that both continuous and discrete versions are NP-complete, and design polynomial-time approximation algorithms that produce $(16 \cdot OPT + 1)$-approximate and $(64 \cdot OPT + 1)$-approximate solutions respectively, using rounding of suitably defined integer linear

A preliminary version of this paper was presented at the ISAAC 2020 conference and appeared as [13].

This author was partially funded by the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25), by the ANR project GRALMECO (ANR-21-CE48-0004-01), by the ANR project HOSIGRA (ANR-17-CE40-0022) and by the IFCAM project "Applications of graph homomorphisms" (MA/IFCAM/18/39).

This work is part of the author's PhD thesis submitted to Indian Statistical Institute, Kolkata.

✉ Subhas C. Nandy
subhas.c.nandy@gmail.com

Extended author information available on the last page of the article

programming problems. Finally, we apply our techniques to a related variant of the discrete problem, where instead of points and geometric objects we just have a set S of objects. The goal is to select a small subset S^* of objects so that all objects of S are discriminated by their intersection with the objects of S^*. This problem can be viewed as a graph problem by stating it in terms of the vertices of the geometric intersection graph of S. Under this graph-theoretical form, it is known as the *identifying code problem*. We show that the identifying code problem for axis-parallel unit square intersection graphs (in $d = 2$) can be solved in the same manner as for the discrete version of the discriminating code problem for unit square objects described above, and all our positive approximation results still hold in this setting.

Keywords Discriminating code · Identifying code · Approximation algorithm · Segment stabbing · Geometric hitting set

1 Introduction

We consider geometric versions of the Discriminating Code problem, which are variations of classical geometric covering problems. Here, a set of point sites $P = \{p_1, p_2, \ldots, p_n\}$ is given in $\mathbb{R}^d$. Let S be a set of geometric objects (i.e. closed curves along with their interior) in $\mathbb{R}^d$, where each $s_i \in S$ has a unique identification. We use $S_i \subseteq S$ to denote the set of objects containing $p_i \in P$. The objective is to choose a minimum-size subset $S^* \subseteq S$ of objects such that (i) the subset $S_i^* \subset S^*$ containing p_i is not an empty set for each $i = 1, 2, \ldots, n$, and (ii) each pair of points $p_i, p_j \in P$, $i \neq j$, satisfy $S_i^* \neq S_j^*$. Thus, (i) suggests that each p_i has an identification, and (ii) suggests that p_i and p_j $(i \neq j)$ are discriminated by at least one element in S_i^* or S_j^*. From now onwards, we will use d to denote d-dimension for some integer d.

In the *discrete* version, both P and S are given as input. In the *continuous* version, only the points in P are given, and the objects can be chosen freely (among some infinite class of allowed objects).

The problem is motivated as follows. Consider a terrain that is difficult to navigate. A set of sensors, each assigned a unique identification number (id), are deployed in that terrain, all of which can communicate with a single base station. If a region of the terrain suffers from some specific problem, a subset of sensors will detect that and inform the base station. From the id's of the alerted sensors, one can uniquely identify the affected region, and a rescue team can be sent. The covering zone of each sensor can be represented by an object in S. In the *arrangement* of these objects (that is, the partition of the plane into *cells* defined by the union of boundaries of all objects, each cell corresponds to a face of this union [11]) divides the entire plane into regions. A representative point of each region may be considered as a site. The set P consists of some of those sites. We need to determine the minimum number of sensors such that no two sites in P are covered by the same set of ids. Apart from coverage problems in sensor networks, this problem has applications in fault detection, heat prone zone in VLSI circuits, disaster management, environmental monitoring, localization and contamination detection [27, 37], to name a few.

The general version of the problem has been formulated as a combinatorial problem in a bipartite graph in [7, 8] as follows.

Problem: MINIMUM DISCRIMINATING CODE (MIN- DISC- CODE)
Input: A connected bipartite graph $G = (U \cup V, E)$, where $E \subseteq \{(u, v) | u \in U, v \in V\}$.
Output: A minimum-size subset $U^* \subseteq U$ such that $U^* \cap N(v) \neq \emptyset$ for all $v \in V$, and $U^* \cap N(v) \neq U^* \cap N(v')$ for every pair $v, v' \in V$, $v \neq v'$.

In the geometric version of the MIN- DISC- CODE problem introduced in [3], which will be further referred to as G- MIN- DISC- CODE, the two sets of nodes in the bipartite graph G are $U = S$, a set of geometric objects, and $V = P$, a set of points in $\mathbb{R}^d$; an object in S is adjacent to all the points in P that it contains. (Graph G is called the *incidence graph of* (P, S).) The *id* of a point $p \in P$ with respect to the set S is the union of the id's of the subset $S' \subseteq S$ that contains p. Given an instance (P, S), two points $p_i, p_j \in P$ are called *twins* if each member in S that contains p_i also contains p_j, and vice-versa. An instance (P, S) of G- MIN- DISC- CODE is *twin-free* if no two points in P are twins. As mentioned earlier, for a twin-free instance, a subset of S that can uniquely assign id's to all the points in P is said to *discriminate* the points of P and is called a *discriminating code*. In the discrete version of the problem, the set S of objects is also given along with the set of points P as the input; the objective is to find a subset $S^* \subseteq S$ of minimum cardinality that is a discriminating code for the points in P. In the continuous version, we can freely choose the objects S^* in $\mathbb{R}^d$ such that each point gets a unique id, and the size of the set S^* is minimum. The two problems are formally stated as follows.

Problem: CONTINUOUS- G- MIN- DISC- CODE (FOR OBJECTS OF A PRESCRIBED TYPE)
Input: A point set P to be discriminated.
Output: A minimum-size set S^* of objects of the prescribed type, placed *anywhere* in the region under consideration, that discriminates the points in P.

Problem: DISCRETE- G- MIN- DISC- CODE
Input: A point set P to be discriminated, and a set of objects S to be used for the discrimination.
Output: A minimum-size subset $S^* \subseteq S$ that discriminates all points in P.

We can use [20] to show the following.

Proposition 1 *Checking whether a given instance* (P, S) *of* DISCRETE- G- MIN- DISC- CODE *with* $|P| = n$ *and* $|S| = m$ *is twin-free can be done in time* $O(m \cdot n)$.

Proof It is known that checking whether a graph has twins (vertices with the same neighbourhood) can be done in linear time (in terms of the number of vertices and edges) using the technique of *partition refinement* (see Algorithm 2 from [20, Section 3]). As mentioned before, our geometric instance (P, S) can be associated to its incidence graph $G = (U \cup V, E)$ and is twin-free if and only if there is no pair of graph-twins inside V in G. Thus, we can build the graph G in linear time by directly using the description of (P, S), and then applying the linear-time algorithm from [20].

The graph G has $m+n$ vertices and $O(m \cdot n)$ edges and thus, the running time follows □

Note that one can also use the above method to decide whether an instance P of Continuous- G- Min- Disc- Code is twin-free, by computing all feasible equivalence classes of objects that can be placed (with respect to the intersections with P). However, in general there can be as many as $2^{|P|}$ such equivalence classes, and it might not always be trivial to compute them: this would depend on the context.

A well-studied special case of Min- Disc- Code for graphs is the problem Minimum Identifying Code, initially defined in [22] as follows (where $N[v_i]$ denotes the set of vertices adjacent to v_i, together with v_i itself):

Problem: Minimum Identifying Code (Min- ID- Code)
Input: A graph $G = (V, E)$.
Output: A subset $V^* \subseteq V$ of minimum size such that V^* is a dominating set, and $V^* \cap N[v_i] \neq V^* \cap N[v_j]$ for every pair $v_i, v_j \in V, i \neq j$.

The Min- ID- Code problem is a special case of Min- Disc- Code [7, 16]: indeed, given a graph G, let $B(G)$ denote the closed neighbourhood incidence bipartite graph of G (one part consists of the vertex set of G, and the other part, of the (multi-)set of all closed neighbouroods of vertices of G; a closed neighbourhood vertex is adjacent to all the vertices it contains). Then, Min- ID- Code for G is equivalent to Min- Disc- Code for $B(G)$. Both Min- ID- Code and Min- Disc- Code are NP-complete [7–9]. A polynomial-time algorithm is available for Min- Disc- Code on trees [8]. It was shown that Min- ID- Code for graphs (and thus Min- Disc- Code) is log-APX hard [26], and this holds even for split graphs, bipartite graphs, co-bipartite graphs [16], and for bipartite graphs of girth 6 [6]. However, for line graphs and planar graphs, Min- ID- Code remains NP-complete but constant factor approximation algorithms are available (see [17] and [4], respectively).

Min- ID- Code was studied in particular for the related setting of geometric intersection graphs, for example on unit disk graphs [31] and interval graphs [6, 15, 18]. The problem remains NP-complete for these graph classes, and a 6-approximation algorithm exists for interval graphs [6]. For unit interval graphs, the computational hardness is not known, but a polynomial-time approximation scheme (PTAS) algorithm is given in the second author's PhD thesis [15]. A PTAS also exists for Min- ID- Code on planar graphs [4].

1.1 Further Related Work

In the context of the above-mentioned practical applications, the Discrete- G- Min- Disc- Code problem in 2D was defined in [3], where an integer programming formulation (ILP) of the problem was given along with an experimental study. The Continuous- G- Min- Disc- Code problem was introduced under a different name in [19], and shown to be NP-complete for disks in 2D, but polynomial-time in 1D (even when the intervals are restricted to have bounded length). These two problems are related to the class of *geometric covering problems*, for which also both the discrete and continuous version are studied extensively [25]. A related problem is the

Test Cover problem [12], which is similar to Min- Disc- Code (but defined on hypergraphs). It is equivalent to the variant of Min- Disc- Code where the covering condition "$U^* \cap N(v) \neq \emptyset$" is not required. Thus, a discriminating code is a test cover, but the converse may not be true, since there may exist a vertex uncovered by a test cover. Geometric versions of Test Cover have been studied under various names. For example, the *separation* problems in [5, 10, 21] can be seen as continuous geometric versions of Test Cover in 2D, where the objects are half-planes. Similar problems are also sometimes called *shattering* problems, see [34].

More references on several coding mechanisms on graphs based on different applications, namely locating-dominating sets, open locating dominating sets, metric dimension, etc, and their computational hardness results are available in [15, 18].

1.2 Our Results

We show that Discrete- G- Min- Disc- Code in 1D using interval objects of arbitrary length, is NP-complete by using a polynomial time reduction from a restricted version of the 3-SAT problem. Here, the challenge is to overcome the linear nature of the problem and to transmit the information across the entire construction without affecting intermediate regions. This result is in contrast with Continuous- G- Min- Disc- Code in 1D, which is polynomial-time solvable [19]. This is also in contrast with most geometric covering problems, which are often polynomial-time solvable in 1D [25].

We then design a simple polynomial-time 2-factor approximation algorithm for Discrete- G- Min- Disc- Code in 1D, that is much simpler than that published in the preliminary version of this paper [13].

We also design a PTAS for both Discrete- G- Min- Disc- Code and Continuous- G- Min- Disc- Code in 1D, when all the objects are required to have the same (unit) length. In this context, it needs to be mentioned once again that Continuous- G- Min- Disc- Code for arbitrary intervals is polynomially solvable [19].

We also study both problems in 2D for axis-parallel unit square objects, which form a natural extension of 1D intervals to the 2D setting. The continuous version is known to be NP-complete for unit disks [19], and we show that the reduction can be adapted to our setting, for both the continuous and discrete cases.

We then design polynomial-time constant-factor approximation algorithms for both problems in that setting. The approximation factors are 16 and 64 for the continuous and discrete problem respectively.[1] To obtain these algorithms, we re-formulate our problems into a problem of *stabbing* line segments in 2D, which can be reduced to a geometric Hitting Set problem.

Our results on Discriminating Code problems are summarized in Table 1.

Finally, we consider the related Min- Id- Code problem restricted to unit square graphs (geometric intersection graphs for 2D axis-parallel unit squares). We show that Min- Id- Code for unit square graphs can be solved in the same manner as the Discrete- G- Min- Disc- Code problem for axis-parallel unit square objects, and

[1] Note that, the $(4 + \epsilon)$ factor approximation algorithms presented in the conference version of this paper [13] were wrong and the algorithms have been corrected here.

Table 1 Summary of our results on G- Min- Disc- Code problems

Object type	Continuous- G- Min- Disc- Code		Discrete- G- Min- Disc- Code	
	Hardness	Algorithm	Hardness	Algorithm
1D intervals	–	Polynomial-time ([19])	NP-hard (Thm. 6)	2-approx. (Thm. 8)
1D bounded intervals	–	Polynomial-time ([19])	Open	2-approx. (Thm. 8)
1D unit intervals	Open	PTAS (Cor. 12)	Open	PTAS(Thm. 11)
2D axis-parallel unit squares	NP-hard (Thm. 14)	$(16 \cdot OPT + 1)$-approx. (Thm. 17)	NP-hard (Thm. 14)	$(64 \cdot OPT + 1)$-approx. (Thm. 18)

our approximation results for Discrete- G- Min- Disc- Code still hold for Min- Id- Code on this class of graphs.

1.3 Structure of the Paper

We start with our results about G- Min- Disc- Code in 1D in Sect. 2. We then present our results on G- Min- Disc- Code for axis-parallel unit squares in 2D in Sect. 3. Our results about Min- ID- Code for unit square intersection graph is presented in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 The G-Min-Disc-Code Problem in 1D

It has been shown that Continuous- G- Min- Disc- Code is polynomial-time solvable in 1D [19]. Thus, in this section we focus on Discrete- G- Min- Disc- Code.

An instance (P, S) of the Discrete- G- Min- Disc- Code problem is a set $P = \{p_1, \ldots, p_n\}$ of points and a set S of m intervals of arbitrary lengths placed on a real line $\mathbb{R}$. Assuming that the points are sorted with respect to their x-coordinate values, we define $n + 1$ *gaps* $\mathcal{G} = \{g_1, \ldots, g_{n+1}\}$, where $g_1 = (-\infty, p_1)$, $g_i = (p_{i-1}, p_i)$ for $2 \le i \le n$, and $g_{n+1} = (p_n, \infty)$.

Observe that (i) if both endpoints of an interval $s \in S$ lie in the same gap of $\mathcal{G}$, then it can not discriminate any pair of points; thus s is *useless*, and (ii) if more than one interval in S have both their endpoints in the same two gaps, say $g_a = (p_a, p_{a+1})$, $g_b = (p_b, p_{b+1}) \in \mathcal{G}$, then both of them discriminate exactly the same point-pairs. Thus, they are *redundant* and we need to keep only one interval among them. In a linear scan, we can first eliminate the useless and redundant intervals. From now onwards, m will denote the number of intervals, none of which are useless or redundant. Hence, m may be $O(n^2)$ in the worst case.

2.1 NP-Completeness

The Discrete- G- Min- Disc- Code problem is in NP, since given a subset $S' \subseteq S$, one can test in polynomial time whether the problem instance (P, S') is twin-free (i.e., whether the id of every point in P induced by S' is unique) by Proposition 1.

We prove the NP-hardness of Discrete- G- Min- Disc- Code using a polynomial-time reduction from the 3-SAT-2l problem (defined below).

Problem: 3-SAT-2l
Input: A collection of m clauses $C = \{c_1, c_2, \ldots, c_m\}$ where each clause contains at most three literals, over a set of n Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$, and each literal appears at most twice.
Output: A truth assignment of X such that each clause is satisfied (if it exists).

Theorem 2 *([40]) 3-SAT-2l is NP-complete.*

Proof It is stated in [40, Theorem 2.1] that "Boolean satisfiability is NP-complete when restricted to instances with 2 or 3 variables per clause and at most 3 occurrences per variable". The following simple argument can be used to derive the statement. Consider a formula with 2 or 3 variables per clause and at most 3 occurrences per variable. If the same literal appears exactly three times in the formula, it means its negation never appears, so we might as well set its variable so that this literal is true, and remove all clauses containing that variable. We then get an equivalent formula. By doing this repeatedly, we obtain an equivalent instance where each literal appears at most twice. Thus, Theorem 2.1 from [40] implies that 3-SAT-2l is NP-complete. □

Given an instance (X, C) of 3-SAT-2l, we construct in polynomial time an instance $(P, S) = \Gamma(X, C)$ of the Discrete- G- Min- Disc- Code problem on the real line $\mathbb{R}$. The main challenge of this reduction is to be able to connect variable and clause gadgets, despite the linear nature of our 1D setting. The basic idea is that we will construct an instance where some specific set of *critical* point-pairs will need to be discriminated (all other pairs being discriminated by some partial solution forced by our gadgets). Let us start by describing our basic gadgets.

Definition 3 A *covering gadget Π* consists of three intervals I, J, K and four points p_1, p_2, p_3 and p_4 satisfying $p_1 \in I$, $p_2 \in I \cap J$, $p_3 \in I \cap J \cap K$ and $p_4 \in J \cap K$ as in Fig. 1. Every other interval of the construction will either contain all four points, or none. There may exist a set of points in $K \setminus \{I \cup J\}$, depending on the need of the reduction.

Observation 1 *The points p_1, p_2, p_3, p_4 can only be discriminated by choosing all three intervals I, J, K in the solution.*

Proof Follows from the fact that none of the intervals in $\Gamma(X, C)$, that is not a member of the covering gadget Π can discriminate the four points in Π. Moreover, if we do not choose I, then p_3, p_4 are not discriminated. If we do not choose J, p_1, p_2 are not discrimnated. If we do not choose K, p_2, p_3 are not discriminated (see Fig. 1). □

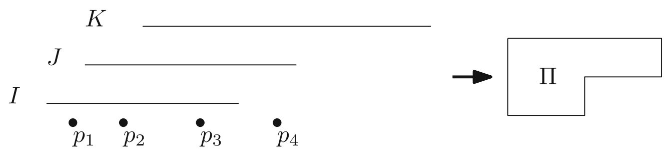

Fig. 1 A covering gadget Π, and its schematic representation

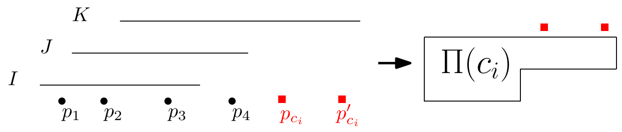

Fig. 2 A clause gadget $\Pi(c_i)$, and its schematic representation

The idea of the covering gadget is to forcefully cover the points placed in $K \setminus \{I \cup J\}$, so that they are covered by K (which needs to be in any solution), and hence discriminated from all other points of the construction.

Let us now define the gadgets modeling the clauses and variables of the 3-SAT-2l instance.

Definition 4 Let c_i be a clause of C. The *clause gadget* for c_i, denoted $G_c(c_i)$, is defined by a covering gadget $\Pi(c_i)$ along with two points p_{c_i}, p'_{c_i} placed in $K \setminus \{I \cup J\}$ (see Fig. 2).

The idea behind the clause gadget is that some interval that ends between points p_{c_i}, p'_{c_i} will have to be taken in the solution, so that this pair gets discriminated.

Definition 5 Let x_j be a variable of X. The *variable gadget* for x_j, denoted $G_v(x_j)$, is defined by a covering gadget $\Pi(x_j)$, and five points $p^1_{x_j}, \dots, p^5_{x_j}$ placed consecutively in $K \setminus \{I \cup J\}$. We place six intervals $I^0_{x_j}$, $I^1_{x_j}$, $I^2_{x_j}$, $I^0_{\overline{x_j}}$, $I^1_{\overline{x_j}}$, $I^2_{\overline{x_j}}$, as in Fig. 3.

- Interval $I^0_{x_j}$ starts between $p^1_{x_j}$ and $p^2_{x_j}$, and ends between $p^3_{x_j}$ and $p^4_{x_j}$.
- Interval $I^0_{\overline{x_j}}$ starts between $p^2_{x_j}$ and $p^3_{x_j}$, and ends between $p^4_{x_j}$ and $p^5_{x_j}$.
- Interval $I^1_{x_j}$ starts between $p^2_{x_j}$ and $p^3_{x_j}$, and ends after $p^5_{x_j}$.
- Interval $I^2_{x_j}$ starts between $p^4_{x_j}$ and $p^5_{x_j}$, and ends after $p^5_{x_j}$.
- Interval $I^1_{\overline{x_j}}$ starts between $p^1_{x_j}$ and $p^2_{x_j}$, and ends after $p^5_{x_j}$.
- Interval $I^2_{\overline{x_j}}$ starts between $p^3_{x_j}$ and $p^4_{x_j}$, and ends after $p^5_{x_j}$.

(The end-point of the four intervals, namely $I^1_{x_j}$, $I^1_{\overline{x_j}}$, $I^2_{x_j}$, $I^2_{\overline{x_j}}$, will be determined at the time of construction of the whole instance.)

In a variable gadget $G_v(x_j)$, the intervals $I^1_{x_j}$ and $I^2_{x_j}$ represent the two possible occurrences of literal x_j, while $I^1_{\overline{x_j}}$ and $I^2_{\overline{x_j}}$ represent the two possible occurrences of $\overline{x_j}$. The right end points of each of these four intervals will be in the clause gadget of the clause where the occurrence of that literal takes place. An example for the

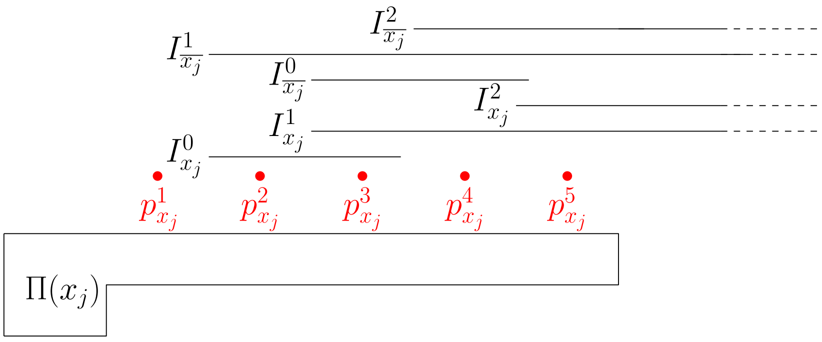

Fig. 3 Variable gadget for variable x_j

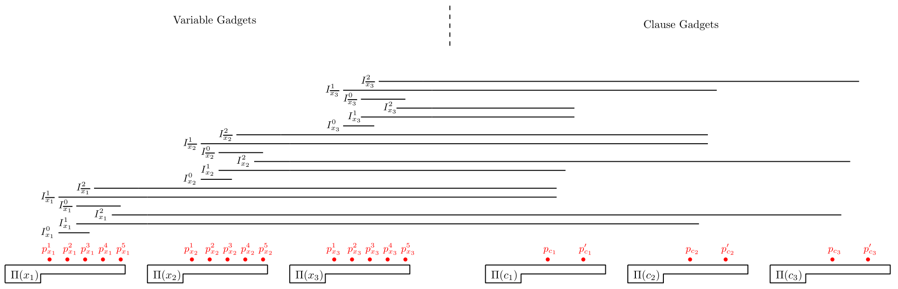

Fig. 4 The instance $\Gamma(X, C)$ for the formula $(X, C) = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_3)$

construction of $\Gamma(X, C)$ is shown in Fig. 4. We assume that every literal appears in at least one clause.[2]

- For each variable $x_i \in X$, $\Gamma(X, C)$ contains a variable gadget $G_v(x_i)$.
- The gadgets $G_v(x_1), G_v(x_2), \ldots, G_v(x_n)$ are positioned consecutively, in this order, without overlap.
- For each clause $c_j \in C$, $\Gamma(X, C)$ contains a clause gadget $G_c(c_j)$.
- The gadgets $G_c(c_1), G_c(c_2), \ldots, G_c(c_m)$ are positioned consecutively, in this order, to the right of the placement of the variable gadgets, without overlap.
- For every variable x_i, assume x_i appears in clauses c_{i_1} and c_{i_2}, and $\overline{x_i}$ appears in c_{i_3} and c_{i_4} (possibly $i_1 = i_2$ or $i_3 = i_4$). Then, we extend the intervals $I^1_{x_i}$, $I^1_{\overline{x_i}}$, $I^2_{x_i}$, $I^2_{\overline{x_i}}$ so that $I^1_{x_i}$ ends between $p_{c_{i_1}}$ and $p'_{c_{i_1}}$, $I^2_{x_i}$ ends between $p_{c_{i_2}}$ and $p'_{c_{i_2}}$, $I^1_{\overline{x_i}}$ ends between $p_{c_{i_3}}$ and $p'_{c_{i_3}}$, and $I^2_{\overline{x_i}}$ ends between $p_{c_{i_4}}$ and $p'_{c_{i_4}}$.

Let $\mathcal{C}^{\Pi}$ be the union of the discriminating codes (i.e., all intervals of type I, J, K, as mentioned in Observation 1) of all covering gadgets of type $\Pi(x_i)$ of $\Gamma(X, C)$.

Consider any covering gadget Π_1. By Observation 1, the points p_1, p_2, p_3, p_4 are discriminated among each other by the set $\mathcal{C}^{\Pi}$, and they are discriminated from all other points of $\Gamma(X, C)$, since they are the only ones to be covered by one of the

[2] If a literal does not appear in any clause, then we can assign a truth value to its variable so that all its occurrences are true, and further ignore this variable.

intervals I, J from Π_1. Moreover, all the points covered by the interval K from Π_1 are discriminated from all the points not covered by K. Thus, overall, all point-pairs of $\Gamma(X, C)$ are discriminated by $\mathcal{C}^{\Pi}$, except the following *critical* point-pairs:

- The point-pairs among the five points $p^1_{x_i}, \ldots, p^5_{x_i}$ of each variable gadget $G_v(x_i)$, and
- The point-pair $\{p_{c_j}, p'_{c_j}\}$ of each clause gadget $G_c(c_j)$.

In the proof of the following main result of this section, we will demonstrate, in particular, that if there exists a truth assignment of the variables in X such that all the clauses in C are satisfied, then the critical point-pairs are also discriminated.

Theorem 6 DISCRETE- G- MIN- DISC- CODE *in 1D is NP-complete.*

Proof We prove that (X, C) is satisfiable if and only if $\Gamma(X, C)$ has a discriminating code of size $6n + 3m$. In both parts of the proof, we will consider the set $\mathcal{C}^{\Pi}$ defined above. Each variable gadget and clause gadget contains one covering gadget. Thus, $|\mathcal{C}^{\Pi}| = 3(n + m)$.

Consider first some satisfying truth assignment of X. We build a solution set $\mathcal{C}$ as follows. First, we put all intervals of $\mathcal{C}^{\Pi}$ in $\mathcal{C}$. Then, for each variable x_i, if x_i is true, we add intervals $I^0_{x_i}$, $I^1_{x_i}$ and $I^2_{x_i}$ to $\mathcal{C}$. Otherwise, we add intervals $I^0_{\overline{x_i}}$, $I^1_{\overline{x_i}}$ and $I^2_{\overline{x_i}}$ to $\mathcal{C}$. Notice that $|\mathcal{C}| = 6n+3m$. As observed before, it suffices to show that $\mathcal{C}$ discriminates the point-pair $\{p_{c_j}, p'_{c_j}\}$ of each clause gadget $G_c(c_j)$, and the points $p^1_{x_i}, \ldots, p^5_{x_i}$ of each variable gadget $G_v(x_i)$. (All other pairs are discriminated by $\mathcal{C}^{\Pi}$.)

Since the assignment is satisfying, each clause c_j contains a true literal $l_i \in \{x_i, \overline{x_i}\}$. Then, one interval of $G_v(x_i)$ is in $\mathcal{C}$ and discriminates p_{c_j} and p'_{c_j}. Furthermore, consider a variable x_i. Point $p^1_{x_i}$ is discriminated from $p^2_{x_i}, \ldots, p^5_{x_i}$ as it is the only one not covered by any of $I^0_{x_i}$, $I^1_{x_i}$, $I^2_{x_i}$, $I^0_{\overline{x_i}}$, $I^1_{\overline{x_i}}$, and $I^2_{\overline{x_i}}$. If x_i is true, $p^2_{x_i}$ is covered by $I^0_{x_i}$; $p^3_{x_i}$ is covered by $I^0_{x_i}$ and $I^1_{x_i}$; $p^4_{x_i}$ is covered by $I^1_{x_i}$; $p^5_{x_i}$ is covered by $I^1_{x_i}$ and $I^2_{x_i}$. If x_i is false, $p^2_{x_i}$ is covered by $I^1_{\overline{x_i}}$; $p^3_{x_i}$ is covered by $I^0_{\overline{x_i}}$ and $I^1_{\overline{x_i}}$; $p^4_{x_i}$ is covered by $I^0_{\overline{x_i}}$, $I^1_{\overline{x_i}}$ and $I^2_{\overline{x_i}}$; $p^5_{x_i}$ is covered by $I^1_{\overline{x_i}}$ and $I^2_{\overline{x_i}}$. Thus, in both cases, the five points are discriminated, and $\mathcal{C}$ is discriminating, as claimed.

For the converse, assume that $\mathcal{C}$ is a discriminating code of $\Gamma(X, C)$ of size $6n+3m$. By Observation 1, $\mathcal{C}^{\Pi} \subseteq \mathcal{C}$. Thus there are $3n$ intervals of $\mathcal{C}$ that are not in $\mathcal{C}^{\Pi}$.

First, we show that $\mathcal{C} \setminus \mathcal{C}^{\Pi}$ contains exactly three intervals of each variable gadget $G_v(x_i)$. Indeed, it cannot contain less than three, otherwise we show that the points $p^1_{x_i}, \ldots, p^5_{x_i}$ cannot be discriminated. To see this, note that each consecutive pair $\{p^s_{x_i}, p^{s+1}_{x_i}\}$ $(1 \leq s \leq 4)$ must be discriminated, thus $\mathcal{C}$ must contain one interval with an endpoint between these two points. There are four such consecutive pairs in $G_v(x_i)$, thus if $\mathcal{C} \setminus \mathcal{C}^{\Pi}$ contains at most two intervals of $G_v(x_i)$, it must contain $I^0_{x_i}$ and $I^0_{\overline{x_i}}$. But now, the points $p^1_{x_i}$ and $p^5_{x_i}$ are not discriminated, a contradiction.

Let us now show how to construct a truth assignment of (X, C). Notice that at least one of $I^0_{x_i}$ and $I^0_{\overline{x_i}}$ must belong to $\mathcal{C}$, otherwise some points of $G_v(x_i)$ cannot be discriminated. If $I^0_{x_i} \in \mathcal{C}$ but $I^0_{\overline{x_i}} \notin C$, then necessarily $I^1_{x_i} \in \mathcal{C}$ to discriminate $p^2_{x_i}$ and $p^3_{x_i}$, and $I^2_{x_i} \in \mathcal{C}$ to discriminate $p^4_{x_i}$ and $p^5_{x_i}$. In this case, we set x_i to true. Similarly, if $I^0_{\overline{x_i}} \in \mathcal{C}$ but $I^0_{x_i} \notin C$, then necessarily $I^1_{\overline{x_i}} \in \mathcal{C}$ to discriminate $p^1_{x_i}$ and $p^2_{x_i}$, and $I^2_{\overline{x_i}} \in \mathcal{C}$

to discriminate $p^3_{x_i}$ and $p^4_{x_i}$. In this case, we set x_i to false. Finally, if both $I^0_{x_i}$ and $I^0_{\overline{x_i}}$ belong to $\mathcal{C}$, the third interval of $\mathcal{C}\backslash\mathcal{C}^{\Pi}$ in $G_v(x_i)$ may be any of the four intervals covering $p^5_{x_i}$. If this third interval is $I^1_{x_i}$ or $I^2_{x_i}$, we set x_i to true; otherwise, we set it to false.

Observe that when we set x_i to true, none of $I^1_{\overline{x_i}}$ and $I^2_{\overline{x_i}}$ belongs to $\mathcal{C}$; likewise, when we set x_i to false, none of $I^1_{x_i}$ and $I^2_{x_i}$ belongs to $\mathcal{C}$. Thus, our truth assignment is coherent. As for every clause c_j, the point-pair $\{p_{c_j}, p'_{c_j}\}$ is discriminated by $\mathcal{C}$, one interval correspoding to a true literal discriminates it. The obtained assignment is satisfying, completing the proof. □

2.2 A 2-Approximation Algorithm

We next design a 2-approximation algorithm for Discrete- G- Min- Disc- Code in 1D by carefully choosing at most n intervals to discriminate the n points of P. We remark that this algorithm is substantially simpler than the one from the preliminary version of this paper [13].

First, we will need the following proposition, already observed in [19].

Proposition 7 *([19]) Any solution of* Discrete- G- Min- Disc- Code *and* Continuous- G- Min- Disc- Code *in 1D for inputs of n points has size at least $\frac{n+1}{2}$.*

Proof In order to discriminate the consecutive points in P, for any feasible solution SOL for P, every gap between two consecutive points will contain an end-point of at least one interval in SOL. There exist $n - 1$ gaps for the n points in P. But we must also have intervals covering the first point and the last point, which amounts to $n + 1$ positions for the end-points of intervals of SOL. Thus, any solution has size at least $(n + 1)/2$. □

Theorem 8 *There exists a 2-factor approximation algorithm solving* Discrete- G- Min- Disc- Code *in 1D, that runs in $O(m \log m)$ time and $O(m)$ space.*

Proof We assume that the points of $P = \{p_1, \ldots, p_n\}$ of the instance (P, S) are sorted in increasing order with respect to their x-coordinate values. At Step i, our partial solution $S_i \subseteq S$ will have the property that it covers and discriminates all the points in $\{p_1, \ldots, p_i\}$ (using at most i intervals). Thus, at step n, S_n is a valid solution for (P, S) of size at most n.

For the first step, we let S_1 consist of any interval that contains p_1. For the next steps, we assume that i iterations have already been executed, and thus we have computed the set S_i that discriminates $P_i = \{p_1, \ldots, p_i\}$. We now consider the set $P_{i+1} = \{p_1, \ldots, p_{i+1}\}$. We distingish three cases as follows.

Case 1 The id of p_{i+1} using the intervals in S_i is non-null and is different from the id of every point $p \in P_i$. Here, simply let $S_{i+1} = S_i$, that is, S_i is already a feasible solution for P_{i+1}.

Case 2 The id of p_{i+1} using the intervals in S_i is null, that is, no interval of S_i covers p_{i+1}. Here, we choose any arbitrary interval $s \in S$ that covers p_{i+1}. Thus, we have $S_{i+1} = S_i \cup \{s\}$. As the members in P_i are already discriminated up to i-th iteration, they remain discriminated by S_{i+1} (even if the new interval covers a subset of P_i).

Case 3 The id of p_{i+1} using the intervals in S_i is non-null, but is the same as the id of some $p_j \in P_i$ $(j < i)$. Note that, in such a case the id of p_{i+1} can match with at most one element of P_i as the id's of P_i are all distinct with respect to S_i. Here, we choose an interval s of S that can discriminate p_j and p_{i+1}. Such an interval always exists as we have already checked that (P, S) is twin-free. Thus, $S_{i+1} = S_i \cup \{s\}$ is our valid partial solution.

As we have inserted at most one interval at each iteration, $|S_n| \leq n$. By Proposition 7, $|S_n| \leq 2OPT - 1$ and the 2-approximation factor follows.

We now analyze the time and space complexity. We will use two tree data structures for processing the points in P and the intervals in S efficiently. These are a *height-balanced binary tree* $\mathcal{T}_H$, and a *priority search tree* $\mathcal{T}_P$.

$\mathcal{T}_H$: It is a binary tree in which the depth of the two subtrees of every node does not differ by more than 1 [24]. A height-balanced binary tree with n nodes has height $\Theta(\log n)$. Each operation (lookup, insertion or deletion) takes time $\Theta(\log n)$ in the worst case.

$\mathcal{T}_P$: A priority search tree [11, 29] is a hybrid of a priority queue and a binary search tree. It stores a set of 2-dimensional points (a pair of real numbers) for the efficient answering of 1.5-dimensional queries in a one-side open query box of the form $(-\infty, a] \times [b, c]$. In other words, it can report/count the points whose x-coordinate is smaller than a, and y-coordinate lies in the range $[b, c]$. The preprocessing time and space complexities of this data structure are $O(n \log n)$ and $O(n)$ respectively; the time complexity for reporting/counting a 1.5-dimensional query is $O(s + \log n)/O(\log n)$, where s is the number of points returned by the search.

Before the start of the algorithm, we compute a $\mathcal{T}_P$ data structure with a set of pairs of reals $(\ell(s), r(s))$ corresponding to the segments S, where $\ell(s)$ and $r(s)$ denote the coordinates of the left and right end-point of the interval $s \in S$ (on the x-axis). The preprocessing time and space required for $\mathcal{T}_P$ are $O(m \log m)$ and $O(m)$ $(m = |S|)$ respectively. Identifying the intervals in S that contains a point $p \in P$ and does not contain a point $q \in P$ is equivalant to a 1.5-dimensional range query with the query box $(-\infty, x(p)] \times (x(p), x(q))$, where $x(p)$ denotes the x-coordinate of the point p.

The height-balanced binary tree $\mathcal{T}_H$ stores the *groups* generated after processing the intervals in S_i, and is updated at the end of each iteration i. A *group* is a maximal set of pairwise intersecting intervals of S_i. These groups are totally ordered in the sense that a pair of consecutive groups share only their common end-point. Each group contains at most one point of P_i. Since $|S_i| \leq i$, and the number of groups created with $|S_i|$ is $2 \times |S_i| + 1$, the size of the data structure $\mathcal{T}_H$ for storing the groups during the entire execution is $O(n)$. While processing p_{i+1}, this tree structure is used to identify an appropriate interval to cover the point p_{i+1} in $O(\log n)$ time. As a result, we also know which case to follow for discriminating p_{i+1}. The cost of maintenance of $\mathcal{T}_H$ after each iteration is also $O(\log n)$

If Case 2 happens, we need to identify an interval $s \in S$ that covers p_{i+1}, i.e., a segment with $\ell(s) < x(p_{i+1}) < r(s)$, or in other words, a pair of reals $(\ell(s), r(s))$ that lies in the range box $(-\infty, x(p_{i+1})] \times [x(p_{i+1}), \infty)$.

If Case 3 happens, then we need to choose a segment $s \in S$ satisfying

either $x(p_j) < \ell(s) < x(p_{i+1}) < r(s)$ i.e., $(\ell(s), r(s)) \in (x(p_j), x(p_{i+1})] \times [x(p_{i+1}), \infty)$

or $\ell(s) < x(p_j) < r(s) < x(p_{i+1})$ i.e., $(\ell(s), r(s)) \in (-\infty, x(p_j)] \times [x(p_j), x(p_{i+1}))$.

As mentioned earlier, in either of the cases such an element can be found in the data structure $\mathcal{T}_P$ in $O(\log m)$ time. Thus, the only task that remains is to modify the data structure $\mathcal{T}_H$ after inserting $s = [a, b] \in \mathcal{T}_H$. Let a and b lie in the groups $g_\alpha = [\theta_1, \theta_2]$ and $g_\beta = [\psi_1, \psi_2]$, respectively. Now, g_α and g_β is to be deleted from $\mathcal{T}_H$ and four intervals $[\theta_1, a]$, $[a, \theta_2]$, $[\psi_1, b]$ and $[b, \psi_2]$ need to be inserted in $\mathcal{T}_H$. If $= [a, b]$ lies entirely in the same group $[\theta_1, \theta_2]$, then $[\theta_1, \theta_2]$ is split into three groups $[\theta_1, a]$, $[a, b]$ and $[b, \theta_2]$. Surely, we need to attach p_j and p_{i+1} to the appropriate interval groups to which they belong. This needs another $O(\log n)$ time. Thus, processing p_{i+1} requires $O(\log m + \log n)$ time in the worst case.

The construction of $\mathcal{T}_P$ needs $O(m \log m)$ time and $O(m)$ space [28, 29]. Processing n points requires $O(n(\log n + \log m))$ time as explained in the previously. As $n = O(m)$, the result follows. □

2.3 A PTAS for the Unit Interval Case

We now design a PTAS for the 1D case where all intervals in S have the same length.

The following observation (which was also made in the related setting of identifying codes of unit interval graphs [15, Proposition 5.12]) plays an important role in designing our PTAS.

Observation 2 *In an instance* (P, S) *of* Discrete- G- Min- Disc- Code *in 1D, if the objects in* S *are intervals of the same length, then discriminating all the pairs of* consecutive *points in* P *is equivalant to discriminating* all *the pairs of points in* P.

Proof Assume that we have a set $S' \subseteq S$ that covers all points and discriminates all consecutive point-pairs, but two non-consecutive points p_i and p_j $(i < j)$ are not discriminated. Since p_i and p_j are covered by the same set of intervals of S' and the intervals are of same (unit) length, they must be at a distance at most 1 apart. Now, since they are not consecutive, p_{i+1} lies between p_i and p_j. Since S' discriminates p_i and p_{i+1}, there is an interval $I \in S'$ with an endpoint in the gap $g_i = [p_i, p_{i+1}]$. If it is the right endpoint, I covers p_i but not p_j, a contradiction. Thus, it must be the left endpoint. But since the distance between p_i and p_j is at most 1, I contains p_j (but not p_i), again a contradiction. □

For a given $\epsilon > 0$, we choose $\lceil \frac{n\epsilon}{4} \rceil$ points, namely $q_1, q_2, \ldots, q_{\lceil \frac{n\epsilon}{4} \rceil} \in P$, called the *reference points*, as follows: q_1 is the $\lceil \frac{2}{\epsilon} \rceil$-th point of P from the left, and for each $i = 1, 2, \ldots, \lfloor \frac{n\epsilon}{4} \rfloor$, the number of points in P between every consecutive pair (q_i, q_{i+1}) is $\lceil \frac{4}{\epsilon} \rceil$ (both inclusive). The number of points to the right of $q_{\lceil \frac{n\epsilon}{4} \rceil}$ may be less than $\lceil \frac{2}{\epsilon} \rceil$. For each *reference point* q_i, we choose two intervals $I_i^1, I_i^2 \in S$ such that both I_i^1, I_i^2 contain (span) q_i, and the left (resp. right) endpoint of I_i^1 (resp. I_i^2) have the minimum x-coordinate (resp. maximum x-coordinate) among all intervals in S that span q_i. Observe that all the points in P that lie in the range $G_i = [\ell(I_i^1), r(I_i^2)]$ are *covered*, where $\ell(I_i^1)$ and $r(I_i^2)$ are the x-coordinates of the left endpoint of I_i^1 and the right endpoint of I_i^2, respectively. These ranges will be referred to as *group-ranges*. Since the endpoints of the intervals are distinct, the span of a *group-range* is strictly greater than 1.

We now define a *block* as follows. Observe that the ranges G_i and G_{i+1} may or may not overlap. If several consecutive ranges $G_i, G_{i+1}, \ldots, G_k$ are pairwise overlapping, then the horizontal range $[\ell(I_i^1), r(I_k^2)]$ forms a block. The region between a pair of consecutive blocks will be referred to as a *free region*. We use $B_1, B_2, \ldots, B_l$ to name the blocks in order, and $F_0, F_1, \ldots, F_l$ to name the free regions (from left to right). The points in each block are covered. Here, the remaining tasks are (i) for each block, choose intervals from S such that consecutive pairs of points in that block are discriminated, and (ii) for each free region, choose intervals from S such that all its points are covered, and the pairs of consecutive points are discriminated.

Observation 3 *There exists no interval $I \in S$ that contains both a point in F_i and a point in F_{i+1}.*

Proof Note that, F_i and F_{i+1} are sepatated by the block B_{i+1}. If there exists an interval I that contains a point in F_i and a point in F_{i+1}, then I will contain the point $q_j \in B_{i+1}$ just to the right of F_i, which is the reference point of the leftmost group-range G_j of the block B_{i+1}. This contradicts the existence of $I \in S$. Also, the size of I then has to be greater than one, which is impossible. □

Thus, the discriminating code for a free region F_i is disjoint from that of its neighboring free region F_{i+1}. So, we can process the free regions independently.

2.3.1 Processing of a Free Region

Let the neighboring group-ranges of a free region F_i be G_a and G_{a+1}, respectively. There are at most $\frac{4}{\epsilon}$ points lying between the reference points of G_a and G_{a+1}. Among these, several points of P to the right (resp. left) of the reference point of G_a (resp. G_{a+1}) are inside *block* B_i (resp. B_{i+1}). Thus, there are at most $\frac{4}{\epsilon}$ points in F_i. Let $S_{F_i} \subseteq S$ be a set whose intervals cover at least one point of F_i. Note that, though we have deleted all the redundant intervals of S, there may exist several intervals in S whose one endpoint lies in a gap inside that free region, and their other endpoint lies in distinct gaps of the neighboring block. In Fig. 5, there are some blue intervals which are redundant with respect to the points $F_i \cap P$, but are non-redundant with respect to the whole point set P. However, the number of such intervals is at most $\frac{4}{\epsilon}$ due to the definition of (I_i^1, I_i^2) of the right-most group-range of the block B_i and left-most group-range of the block B_{i+1}.

Thus, we have $|S_{F_i}| = O(1/\epsilon^2)$. We consider all possible subsets of intervals of S_{F_i}, and test each of them for being a discriminating code for the points in F_i. Let $\mathcal{D}_i$

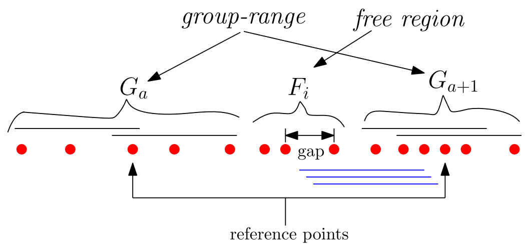

Fig. 5 Demonstration of redundant edges in a free region which are non-redundant in the problem instance (P, S)

be all possible different discriminating codes of the points in F_i, with $|\mathcal{D}_i| = 2^{O(1/\epsilon^2)}$ in the worst case.

2.3.2 Processing of a Block

Consider a block B_i; its neighboring free regions are F_i and F_{i+1}. Consider two discriminating codes $d \in \mathcal{D}_i$ and $d' \in \mathcal{D}_{i+1}$. We create a graph $G_i = (V_i, E_i)$ whose nodes V_i correspond to the gaps of B_i which are not discriminated by the intervals used in $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$. Each edge $e \in E_i$ corresponds to an interval in S that discriminates pairs of consecutive points corresponding to two different nodes (gaps) of V_i. Now, we can discriminate each non-discriminated pair of consecutive points in B_i by computing a minimum edge-cover of G_i in $O(|V_i|^2)$ time [30]. As mentioned earlier, all the points in B_i are covered. Thus, the discrimination process for the block B_i is over. We will use $\theta(d, d')$ to denote the size of a minimum edge-cover of B_i using $d \in \mathcal{D}_i$ and $d' \in \mathcal{D}_{i+1}$.

2.3.3 Computing a Discriminating Code for *P*

We now create a multipartite directed graph $H = (\mathcal{D}, \mathcal{F})$. Its i-th partite set corresponds to the discriminating codes in $\mathcal{D}_i$, and $\mathcal{D} = \cup_{i=0}^{l} \mathcal{D}_i$. Each node $d \in \mathcal{D}$ has its weight equal to the size of the discriminating code d. A directed edge $(d, d') \in \mathcal{F}$ connects two nodes d and d' of two adjacent partite sets, say $d \in \mathcal{D}_i$ and $d' \in \mathcal{D}_{i+1}$, and has its weight equal to $\theta(d, d')$. For every pair of partite sets $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$, we connect every pair of nodes (d, d'), $d \in \mathcal{D}_i$ and $d' \in \mathcal{D}_{i+1}$, where $i = 0, 1, \ldots, l-1$. Every node of $\mathcal{D}_0$ is connected to a node s with weight 0, and every node of $\mathcal{D}_l$ is connected to a node t with weight 0.

Lemma 9 *The minimum weight s-t path in H is a lower bound on the size of the optimum discriminating code for (P, S), where the weight of a path is equal to the sum of costs of all the vertices and edges on that path.*

Proof Let Π be the minimum weight s-t path in the graph H, which corresponds to a set of intervals $S' \subseteq S$. To show, $|S'| \leq |S_{opt}|$, where $S_{opt} \subseteq S$ corresponds to the minimum weight discriminating code. For a contradiction, let $|S'| > |S_{opt}|$. As S_{opt} is a discriminating code, the points of every *free region* F_i are discriminated by a subset, say $\delta_i \in S$. Since, we maintain all the discriminating codes in $\mathcal{D}_i$, surely the subset $\delta_i \in \mathcal{D}_i$. Let $b_i \subset S$ be the set of intervals that span the points of the block B_i. As S_{opt} is a discriminating code, the points in B_i are discriminated by the intervals in $b_i \cup \delta_i \cup \delta_{i+1}$. Thus, the set of intervals $\beta_i = b_i \backslash (\delta_i \cup \delta_{i+1})$ discriminate the pair of points of B_i that are not discriminated by $\delta_i \cup \delta_{i+1}$. Observe that, for every $i = 0, 1, \ldots, l$, we have $\delta_i \in \mathcal{D}_i$. Moreover, there exists a path Π_{opt} that connects $\delta_i, i = 0, 1, \ldots, l$, whose each edge (δ_i, δ_{i+1}) has cost equal to $|\beta_i|$. Thus, we have the contradiction that Π_{opt} is a path in H having cost less than that of Π. □

The set of intervals may not form a discriminating code for P, as the points in a block may not all be covered. However, the additional intervals $\{(I_i^1, I_i^2), i = 1, 2, \ldots, \lceil \frac{n\epsilon}{2} \rceil\}$ ensure that the optimum size of the discriminating code satisfies $S_{opt} \geq \lceil \frac{n+1}{2} \rceil$ due to the fact that we have $(n + 1)$ gaps, and each interval in S covers exactly 2 gaps. This fact, along with Lemma 9 implies:

Lemma 10 $|SOL| \leq (1 + \epsilon)S_{opt}$.

Proof By Lemma 9, $|S'| \leq S_{opt}$. The number of extra intervals to cover the *block*s is $\frac{n\epsilon}{2}$. Again, $\frac{n}{2} \leq MEC(P) \leq S_{opt}$, where $MEC(P)$ is the size of minimum edge-cover of the graph G created with the points in P and the intervals in S. Thus, $|SOL| \leq (1 + \epsilon)S_{opt}$. □

We now analyze the time complexity of the algorithm. Note that, the number of possible discriminating codes in a free region is $2^{O(1/\epsilon^2)}$. Thus, in the graph H, the number of edges between a pair of consecutive partite sets $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ is $|\mathcal{D}_i| \times |\mathcal{D}_{i+1}| = 2^{O(1/\epsilon^2)}$. As the computation of the cost of an edge between the sets $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ invokes the edge-cover algorithm of an undirected graph, it needs $O(|B_i|^2)$ time [30]. Thus, the total running time of the algorithm is $A + B$, where A is the total time of generating the edge costs, and B is the time for computing a shortest path of H. We have $A \leq \sum_{i=1}^{\lceil \frac{n\epsilon}{4} \rceil} 2^{O(1/\epsilon^2)} \times O(|B_i|^2)$. As the B_i's are mutually disjoint, we get $A = O(n^2 \times 2^{O(1/\epsilon^2)})$. Moreover, $B = O(|\mathcal{F}|) = O(\frac{n}{\epsilon} \times 2^{O(1/\epsilon^2)})$ [39].

In order to reduce the space requirement of the algorithm, we generate partite sets of the multipartite graph H one by one, and compute the length of the shortest path from s up to each node of that set. Initially, the length of the path up to a node $d \in \mathcal{D}_0$ is $|d|$. While generating $\mathcal{D}_{i+1}$, the nodes in $\mathcal{D}_i$ are available along with the length of the shortest path $\chi(d)$ up to each node $d \in \mathcal{D}_i$ from s. Now, we execute the following steps:

Step 1 We generate the nodes of $\mathcal{D}_{i+1}$, and initialize their cost $\chi(.)$ with ∞.
Step 2 For each pair of nodes (d, d'), $d \in \mathcal{D}_i$, $d' \in \mathcal{D}_{i+1}$, do the following:

- Compute the edge cost $\theta(d, d')$, which is the size of the edge-cover of the block B_i using the discriminating codes d of the *free region* F_i and d' of the *free region* F_{i+1}. This needs $O(|B_i|^2)$ time using the matching algorithm of an undirected graph [30].
- Compute the length of the shortest path from s to d' using the edge (d, d'), which is $\chi^* = \chi(d) + \theta(d, d') + |d'|$.
- If the computed length is less than the existing value of $\chi(d')$, then update $\chi(d')$ with χ^*.

As the number of discriminating codes in each partite set is $2^{O(1/\epsilon^2)}$ in the worst case which are computed online while considering the $(i+1)$-th partite set, and each discriminating code is of length at most $O(\frac{1}{\epsilon})$, we have the following result.

Theorem 11 *The* Discrete- G- Min- Disc- Code *problem in 1D for unit interval objects admits a PTAS: for every $\epsilon > 0$, there is a $(1+\epsilon)$-factor approximation algorithm with time complexity $2^{O(1/\epsilon^2)}n^2$ using $\frac{1}{\epsilon}2^{O(1/\epsilon^2)}n^2$ space.*

Moreover, in this unit interval setting, we easily reduce an instance of Continuous- G- Min- Disc- Code problem to an instance of Discrete- G- Min- Disc- Code problem by first computing the $O(n^2)$ possible non-redundant unit intervals. Thus:

Corollary 12 *The* Continuous- G- Min- Disc- Code *problem in 1D for unit interval objects has a PTAS with the same approximation factor, time and space complexity as those for* Discrete- G- Min- Disc- Code.

3 The G-Min-Disc-Code Problem in 2D

Here, the point set $P = \{p_1, p_2, \ldots, p_n\}$ is given in $\mathbb{R}^2$, and the shape of allowed objects used for covering and discriminating the points of P are axis-parallel squares of equal size. We will use the term *unit square* to refer to these objects.

3.1 NP-Completeness

In [19], it has been shown that Continuous- G- Min- Disc- Code for unit disks in 2D is NP-complete. They reduced the P_3-Partition- Grid problem, stated below, to Continuous- G- Min- Disc- Code for unit disks in 2D. We will modify their reduction and apply it to Continuous- G- Min- Disc- Code for axis-parallel unit squares in 2D.

A *grid graph* is a graph whose vertices are positioned in $\mathbb{Z}^2$, and a pair of vertices are adjacent if they are at Euclidean distance 1 [41].

Problem: P_3- Partition- Grid [41]
Input: A grid graph G.
Output: A partition of the vertices of G into disjoint P_3-paths, where a P_3-path is a path with three vertices.

Given an instance G of P_3-Partition- Grid, we construct an instance P_G (a point set) for Continuous- G- Min- Disc- Code as follows. For every vertex v of G with coordinates (x, y), we create a point $p(v)$ with coordinates (x, y) and add it to P_G. The construction from [19] stops here, and we will now slightly change it. For each point $p(v)$ with coordinates (x, y), we replace it by a point with coordinates $(y-x, y+x)$, that is, we rotate the whole point set by an angle of $\pi/4$ and stretch it by a factor of $\sqrt{2}$ (See Fig. 6 for an illustration).

Lemma 13 *A P_3-partition for $G = (V, E)$ exists if and only if there exists a set of $\frac{2|V|}{3}$ axis-parallel unit squares discriminating the points in P_G.*

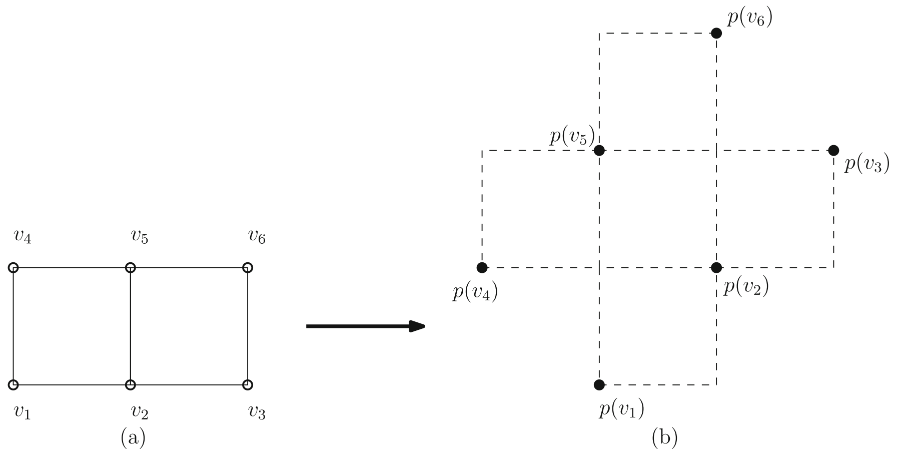

Fig. 6 **a** A grid graph G. **b** Its corresponding geometric instance P_G, where the dashed axis-parallel unit squares are those covering two points each

Proof The key idea is to notice that any axis-parallel unit square can contain at most two points of P_G, and if it contains two, then it contains two points corresponding to vertices of G joined by an edge (the center of the square is then placed at the middle-most position of the line segment joining the two points). Moreover, any two points corresponding to an edge of G can be covered by some axis-parallel unit square in that way. Three points corresponding to the three vertices of a P_3-path $v_1v_2v_3$ in G can be discriminated using two unit squares s and s', centered at the mid-points of the two segments joining $(p(v_1), p(v_2))$ and $(p(v_2), p(v_3))$, respectively. Now, $p(v_1)$ is covered by s only, $p(v_3)$ by s' only, and $p(v_2)$ by both. Thus, if a P_3-partition of G exists, we have our solution of size $\frac{2|V|}{3}$ to the Continuous- G- Min- Disc- Code problem.

Conversely, assume that we have $\frac{2|V|}{3}$ axis-parallel unit squares that discriminate all points of P_G. Recall that every square can cover at most two points. For any square s covering two points $p(v_1)$, $p_(v_2)$, we necessarily have that v_1v_2 is an edge in G. Moreover, one of $p(v_1)$, $p_(v_2)$ needs to be covered by a second square s' (so that the two points are discriminated). Thus, any solution needs at least $\frac{2|V|}{3}$ squares, and any solution of exactly this size will consist of disjoint sets of three points covered by two squares (one point covered by both squares, and the other two, by one of the squares each). These three points must correspond to three vertices of G forming a P_3. Thus, we obtain our P_3-partition of G, as claimed. □

Lemma 13 leads to the following result:

Theorem 14 Continuous- G- Min- Disc- Code *and* Discrete- G- Min- Disc- Code *for axis-parallel unit squares in 2D are NP-complete.*

Proof The statement follows directly from Lemma 13 in the case of Continuous- G- Min- Disc- Code. Let S_G contain the set of all axis-parallel unit squares that cover two points of P_G. For Discrete- G- Min- Disc- Code, we can simply modify the reduction by creating the instance (P_G, S_G) from G. □

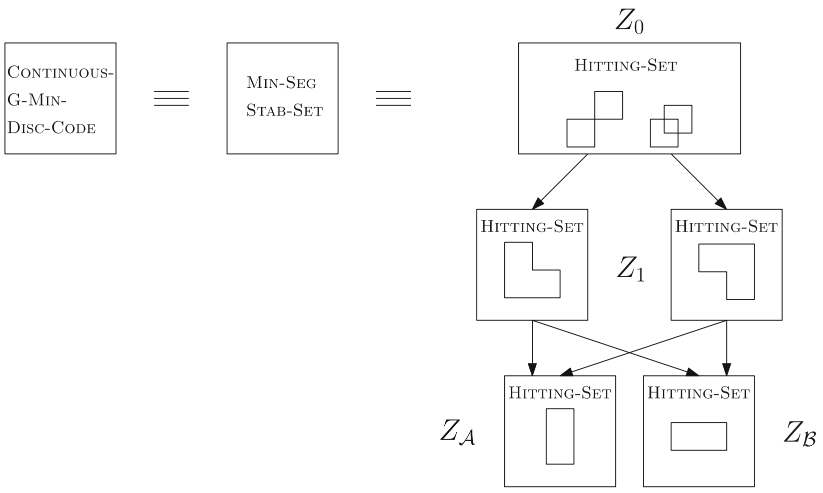

Fig. 7 Schematic of the breaking-up of the problem

3.2 Approximation Algorithms

We formulate an approximation algorithm by extending the ideas for the 1D case, described in Sect. 2.2. We will use the techniques of rounding some suitable Integer Linear Programmes (ILPs). Here, our goal is to choose a set Q of points in $\mathbb{R}^2$ of minimum cardinality such that (i) every point of P is covered by at least one axis-parallel unit square among those centered at the points in Q (*covering condition*) and (ii) for every pair of points $p_i, p_j \in P$ $(i \neq j)$, there exists at least one square in Q whose boundary intersects the interior of the line segment $[p_i, p_j]$ exactly once (*discrimination condition*).

We transform our CONTINOUS- G- MIN- DIC- CODE problem into an equivalent problem of segment stabbing (which will be defined below). The segment stabbing problem can also be seen as a hitting set problem of a pair of shapes, called L-shapes. Each of these L-shape hitting set problems is further split into two hitting set problems of unit-height rectangles (or unit-width rectangles). A schematic representation of this process is shown in Fig. 7.

We define the set of line segments $L(P) = \{[p_i, p_j]$ for all $p_i, p_j \in P, i \neq j\}$. Thus, the discrimination condition leads to the following problem (where by *stabbing* a line segment ℓ by a unit square s, we mean that exactly one end-point of ℓ lies inside s).

Problem: MINIMUM SEGMENT- STABBING SET (MIN- SEG- STAB- SET)
Input: A set L of segments in 2D.
Output: A minimum-size set S of axis-parallel unit squares in 2D such that each segment is stabbed by some square of S.

In fact, MIN- SEG- STAB- SET for the input segments $L(P)$ is equivalent to the TEST COVER problem for P using axis-parallel unit squares as tests. As in the edge-cover

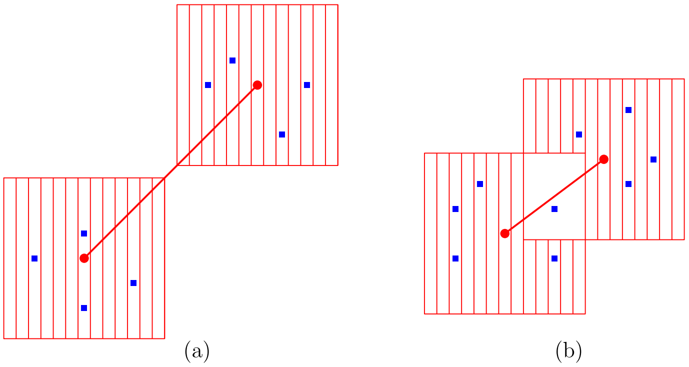

Fig. 8 Object that needs to be hit corresponding to segment $\ell = [a, b]$, where (a) $length(\ell) \geq 1$ and (b) $length(\ell) < 1$

formulation of Discrete- G- Min- Disc- Code problem in 1D (see Sect. 2.2), here also a feasible solution of Min- Seg- Stab- Set ensures the following:

Observation 4 *Every feasible solution Φ of* Min- Seg- Stab- Set *(a) discriminates every point-pair in P, and (b) at most one point is not covered by any square in Φ.*

In order to discriminate the two endpoints of a member $\ell = [a, b] \in L(P)$, we need to consider the two cases: $length(\ell) \geq 1$ and $length(\ell) < 1$, where $length(\ell)$ denotes the length of ℓ. In the former case, if a center is chosen in any one of the unit squares $D(a)$ and $D(b)$, the segment ℓ is stabbed, where $D(q)$ is the axis parallel unit square centered at a point q. However, more generally in the second case, to stab ℓ, we need to choose a center in the region $(D(a)\setminus D(b)) \cup (D(b)\setminus D(a))$. In Fig. 8 the shaded region is the feasible region for placing the center of the unit squares to stab a line segment in $L(P)$. We define a set of distinct objects $\mathcal{O}$ corresponding to the elements of $L(P)$, where each object corresponds to the feasible region of placing the center of a stabbing square of an element of $L(P)$. Thus, the Min- Seg- Stab- Set problem reduces to a Hitting- Set problem, where the objective is to choose a minimum number of points in $\mathbb{R}^2$, such that each object in $\mathcal{O}$ contains at least one of those chosen points.

3.2.1 The Hitting-Set Problem

We use the technique followed in [1] to solve this problem. Consider the arrangement $\mathcal{A}$ of the objects in $\mathcal{O}$. Create a set Q of points by choosing one point in each cell of $\mathcal{A}$. Thus, the size of the set Q is polynomial in the size of the set P. A square centered at a point q inside a cell $A \in \mathcal{A}$ will stab all the segments whose corresponding objects have common intersection region A. For each point $q_\alpha \in Q$, we use an indicator variable x_α, and can write an integer linear programming (ILP) problem as follows.

$$
\begin{aligned}
Z_0 : \min \sum_{\alpha=1}^{|Q|} x_\alpha, & \\
\text{subject to } \ \sigma_1(\ell) + \sigma_2(\ell) \geq 1 \ & \text{ for each segment } \ \ell = [a, b] \in L(P), \\
\text{where } \ \sigma_1(\ell) = & \sum_{q_\alpha \in Q \cap (D(a) \setminus D(b))} x_\alpha, \\
\sigma_2(\ell) = & \sum_{q_\alpha \in Q \cap (D(b) \setminus D(a))} x_\alpha, \\
\text{and } \ x_\alpha \in \{0, 1\} & \text{ for all points } q_\alpha \in Q.
\end{aligned}
$$

As the ILP problem is NP-hard [35], we relax the integrality condition of the variables x_α for all $q_\alpha \in Q$ from Z_0, and solve the corresponding LP problem in polynomial time.

$$
\begin{aligned}
\overline{Z}_0 : \min \sum_{\alpha=1}^{|Q|} x_\alpha & \\
\text{subject to } \sigma_1(\ell) + \sigma_2(\ell) \geq 1 \ \forall \ \ell = [a, b] \in L(P), & \\
\text{and } \quad 0 \leq x_\alpha \leq 1 \ \forall \ q_\alpha \in Q. &
\end{aligned}
$$

Observe that in the optimum solution $\overline{OPT}_0$ of $\overline{Z}_0$, for each constraint (corresponding to a segment $\ell \in L(P)$), at least one of $\sigma_1(\ell)$ or $\sigma_2(\ell)$ will be greater than or equal to $\frac{1}{2}$. We now define two sets, namely $\mathcal{O}_1$ and $\mathcal{O}_2$. If $\sigma_1(\ell) \geq \frac{1}{2}$ then we put ℓ in the set $\mathcal{O}_1$, and if $\sigma_2(\ell) \geq \frac{1}{2}$ then put ℓ in the set $\mathcal{O}_2$. In other words, we choose to hit the objects $(D(p_i) \setminus D(p_j))$ for all $\ell_{ij} = [p_i, p_j]$, $i < j$, if $\sigma_1(\ell_{ij}) \geq \frac{1}{2}$, and choose to hit the objects $(D(p_j) \setminus D(p_i))$ for all $\ell_{ij} = [p_i, p_j]$, $i < j$, if $\sigma_2(\ell_{ij}) \geq \frac{1}{2}$. It needs to be mentioned that, for a constraint corresponding to a point-pair $\ell = [a, b]$ both $\sigma_1(\ell) \geq \frac{1}{2}$ and $\sigma_2(\ell) \geq \frac{1}{2}$ may happen. In that case ℓ may be considered in any one the sets $\mathcal{O}_1$, $\mathcal{O}_2$ arbitrarily. We form a new ILP as follows:

$$
\begin{aligned}
Z_1 : \min \sum_{\alpha=1}^{|Q|} x_\alpha & \\
\text{subject to } \sigma_1(\ell) \geq 1 \ \ \forall \ \ \ell \in \mathcal{O}_1, & \\
\sigma_2(\ell) \geq 1 \ \ \forall \ \ \ell \in \mathcal{O}_2, & \\
\text{and } \quad x_\alpha \in \{0, 1\} \ \forall \ q_\alpha \in Q. &
\end{aligned}
$$

We use $\overline{Z}_1$ to denote the LP corresponding to the ILP Z_1, OPT_0 and OPT_1, the optimal solutions of Z_0 and Z_1 respectively, and $\overline{OPT}_0$ and $\overline{OPT}_1$ the optimal solutions of $\overline{Z}_0$ and $\overline{Z}_1$ respectively. Observe that $2\overline{OPT}_0$ produces a feasible solution to $\overline{Z}_1$. Thus,

$$
\overline{OPT}_1 \ \leq 2\overline{OPT}_0 \ \ (\leq 2OPT_0). \tag{1}
$$

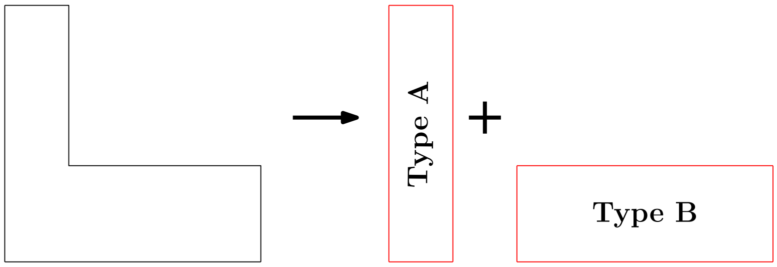

Fig. 9 An L-shaped object, which is the union of a type A and a type B object

However, as the values of the variables in $\overline{OPT}_1$ are fractional, it is not possible to generate a solution of Z_0 from $\overline{OPT}_1$. Observe the objects $\mathcal{O}_1 \cup \mathcal{O}_2$ considered in Z_1 are either a unit square, or a L-shaped object for which one of the length or the width is 1. Thus, our objective is to solve the L- HIT problem, stated below.

3.2.2 The L-HIT Problem

Here, given a set of L-shaped objects as defined above, and a set of points Q, we wish to choose a minimum-size set of points in Q to hit all the L-shaped objects in $\mathcal{O}_1 \cup \mathcal{O}_2$.

We can view an L-shaped object as the union of two rectangles of type A and B, where each type A rectangle has height 1 and width less than or *equal* to 1 and each type B rectangle has width 1 and height less than 1 (see Fig. 9). (Thus, unit squares are considered to be type A rectangles.)

While solving $\overline{Z}_1$, for each constraint (with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$) any one or both of the following cases may happen: (a) the sum of variables whose corresponding points lie in a type A rectangle is $\geq \frac{1}{2}$, and (b) the sum of variables whose corresponding points lie in a type B rectangle is $\geq \frac{1}{2}$. We accumulate all the rectangles in the set $\mathcal{A}$ (resp. $\mathcal{B}$) where condition (a) (resp. condition (b)) is satisfied. The objective is to choose a minimum number of points in Q to hit all the rectangles in $\mathcal{A}$ and $\mathcal{B}$. We formulate two ILPs' $Z_{\mathcal{A}}$ and $Z_{\mathcal{B}}$ corresponding to two MIN- UHR- HIT- SET problems with the set of rectangles $\mathcal{A}$ and $\mathcal{B}$ respectively, as stated below.

Problem: MINIMUM UNIT- HEIGHT RECTANGLE HITTING SET (MIN- UHR- HIT- SET)
Input: A set $\mathcal{R}$ of unit-height rectangles in $\mathbb{R}^2$.
Output: A set of points that hits all the members of $\mathcal{R}$.

A PTAS for the MIN- UHR- HIT- SET problem is known [32]; however it cannot be used in Eq. (1) since that does not guarantee any approximation factor for the optimum solution of the corresponding LP problem. However, the MIN- UHR- HIT- SET problem for a set of rectangles $\mathcal{R} = \mathcal{A}$ (resp. $\mathcal{B}$) can be formulated as an ILP $Z_{\mathcal{A}}$ (resp. $Z_{\mathcal{B}}$) as follows:

$Z_{\mathcal{A}}$: min $\sum_{\alpha=1}^{|Q|} x_\alpha$,
subject to $\sum_{q_\alpha \in A_i \cap Q} x_\alpha \geq 1, \forall$ rectangle $A_i \in \mathcal{A}$,
and $x_\alpha \in \{0, 1\}, \forall\, \alpha \in Q$.

$Z_{\mathcal{B}}$: min $\sum_{\alpha=1}^{|Q|} x_\alpha$,
subject to $\sum_{q_\alpha \in B_i \cap Q} x_\alpha \geq 1, \forall$ rectangle $B_i \in \mathcal{B}$,
and $x_\alpha \in \{0, 1\}, \forall\, \alpha \in Q$.

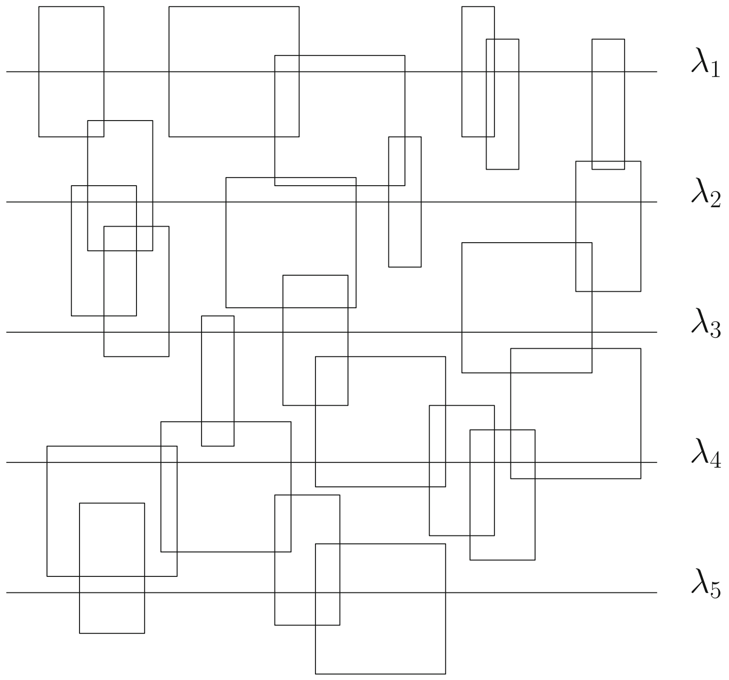

Fig. 10 Demonstration of splitting the plane into unit-width horizontal strips; λ_is' denote the horizontal lines defining the strips

Denoting by $\overline{Z}_{\mathcal{A}}$ and $\overline{Z}_{\mathcal{B}}$ the LP version of $Z_{\mathcal{A}}$ and $Z_{\mathcal{B}}$, and $\overline{OPT}_{\mathcal{A}}$ and $\overline{OPT}_{\mathcal{B}}$ the optimum solutions for $\overline{Z}_{\mathcal{A}}$ and $\overline{Z}_{\mathcal{B}}$ respectively, we observe that $2\overline{OPT}_1$ gives a feasible solution to both $\overline{Z}_{\mathcal{A}}$ and $\overline{Z}_{\mathcal{B}}$ simultaneously. Note that, as the variables participating in $\overline{OPT}_1$ and $\overline{OPT}_2$ may not be disjoint, it is not possible to write $\overline{OPT}_{\mathcal{A}} + \overline{OPT}_{\mathcal{B}} = 2\overline{OPT}_1$. However, $\overline{Z}_{\mathcal{A}} \leq 2\overline{OPT}_1$ and $\overline{Z}_{\mathcal{B}} \leq 2\overline{OPT}_1$. Thus by Eq. (1), we have

$$\overline{OPT}_{\mathcal{A}} + \overline{OPT}_{\mathcal{B}} \leq 4\overline{OPT}_1 \leq 8\overline{OPT}_0. \tag{2}$$

For solving the Min- UHR- Hit- Set problem, we use a similar approximation scheme as the one for unit-height rectangles described in [2]. Here, $\mathbb{R}^2$ is split using x-axis parallel lines from a set $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots\}$ such that λ_i and λ_{i+1} are at distance 1 of each other, and such that each rectangle is hit by one of the lines of $\mathcal{L}$. Thus, each rectangle in $\mathcal{A}$ is intersected by *exactly* one line of $\mathcal{L}$ (assuming that no rectangle in $\mathcal{A}$ is aligned with a line in $\mathcal{L}$). See Fig. 10 for a visual representation.

Let $\mathcal{A}_{even}$ (resp. $\mathcal{A}_{odd}$) denote the set of rectangles in $\mathcal{A}$ that are intersected by even (resp. odd) numbered lines of $\mathcal{L}$. Now, denoting by $Z_{\mathcal{A}}$, $Z(\mathcal{A}_{even})$ and $Z(\mathcal{A}_{odd})$ the ILP of the hitting set problems corresponding to the set of rectangles $\mathcal{A}$, $\mathcal{A}_{even}$ and $\mathcal{A}_{odd}$ respectively, $OPT_{\mathcal{A}}$, $OPT(\mathcal{A}_{even})$ and $OPT(\mathcal{A}_{odd})$ as the optimum solutions of these problems, and $\overline{OPT}_{\mathcal{A}}$, $\overline{OPT}(\mathcal{A}_{even})$ and $\overline{OPT}(\mathcal{A}_{odd})$ as the optimum solutions of the corresponding LP problems, we have

$$\overline{OPT}(\mathcal{A}_{even}) \leq \overline{OPT}_{\mathcal{A}} \text{ and } \overline{OPT}(\mathcal{A}_{odd}) \leq \overline{OPT}_{\mathcal{A}}.$$

Now, combining these two inequalities, we have

$$\overline{OPT}(\mathcal{A}_{even}) + \overline{OPT}(\mathcal{A}_{odd}) \leq 2\overline{OPT}_{\mathcal{A}}. \tag{3}$$

We will now need the following lemma.

Lemma 15 *If there exists a horizontal line λ that stabs a set of axis-parallel rectangles $\mathcal{D}$, then the optimum hitting set for the set $\mathcal{D}$ can be computed in polynomial time using the LP relaxation of the corresponding ILP problem.*

Proof Let Π be the optimal hitting set for the rectangles in $\mathcal{D}$. Consider the intersection of the elements of $\mathcal{D}$ with the line λ. As λ intersects all the members in $\mathcal{D}$, if we shift every member of Π on λ it will not miss to hit any rectangle of $\mathcal{D}$ that it was hitting in its earlier position. Thus, our problem reduces to hitting a set of intervals obtained by the intersection of λ with the rectangles in $\mathcal{D}$. Considering the arrangement of those intervals and choosing a point in each cell of the arrangement, we can formulate this hitting set problem as an ILP. Again, since the coefficient matrix corresponding to the constraints of this ILP satisfies the consecutive-1-property, we can get the optimal solution of this ILP by solving its LP relaxation [38]. □

Denoting by $Z(\mathcal{A}_i)$ the hitting set problem with the set of rectangles intersected by $\lambda_i \in \mathcal{L}$, $OPT(\mathcal{A}_i)$ and $\overline{OPT}(\mathcal{A}_i)$ as the optimum solutions for the ILP and LP versions of $Z(\mathcal{A}_i)$, and Lemma 15, we have $\overline{OPT}(\mathcal{A}_{odd}) = OPT(\mathcal{A}_{odd}) = OPT(\mathcal{A}_1) + OPT(\mathcal{A}_3) + \ldots$. The reason is that for the problem $Z(\mathcal{A}_{odd})$ none of the rectangles in $\mathcal{A}_i$ overlap with any rectangle in $\mathcal{A}_j$, for all i, j odd and $i \neq j$. Thus, the hitting set problem for those instances can be solved independently. Similarly, we have $\overline{OPT}(\mathcal{A}_{even}) = OPT(\mathcal{A}_{even}) = OPT(\mathcal{A}_2) + OPT(\mathcal{A}_4) + \cdots$.

Equations (2), (3) and the subsequent discussions lead to the following. Considering the previously computed optimal solutions $SOL(Z_{\mathcal{A}})$ and $SOL(Z_{\mathcal{B}})$ for $Z_{\mathcal{A}}$ and $Z_{\mathcal{B}}$, which, by the previous discussions, together form a solution for Z_0, we obtain the following chain of inequalities.

$$
\begin{aligned}
|SOL(Z_{\mathcal{A}})| + |SOL(Z_{\mathcal{B}})| &= |SOL(\mathcal{A}_{even})| + |SOL(\mathcal{A}_{odd})| + |SOL(\mathcal{B}_{even})| \\
&\quad + |SOL(\mathcal{B}_{odd})| \\
&= \overline{OPT}(\mathcal{A}_{even}) + \overline{OPT}(\mathcal{A}_{odd}) + \overline{OPT}(\mathcal{B}_{even}) \\
&\quad + \overline{OPT}(\mathcal{B}_{odd}) \text{ by Lemma 15} \qquad (4) \\
&\le 2 \times (\overline{OPT}(Z_{\mathcal{A}}) + \overline{OPT}(Z_{\mathcal{B}})) \text{by Equation (3)} \\
&\le 16 \times \overline{OPT}_0 \text{ by Equation (2)} \\
&\le 16 \times OPT_0.
\end{aligned}
$$

Thus, we have proved the following.

Lemma 16 *The aforesaid algorithm computes a 16-factor approximate solution for* Min- Seg- Stab- Set.

We accumulate the hitting set for type A rectangles corresponding to each horizontal line and the hitting set for type B rectangles corresponding to each vertical line in a set Q^*. By Observation 4, at most one point in P may not be covered by the squares centered at the points of Q^*. Thus, we may require at most one extra square to cover that uncovered point. Thus, we have the following.

Theorem 17 *There exists a polynomial-time algorithm for the* Continuous- G- Min- Disc- Code *problem for axis-parallel unit squares which produces a solution of size at most* $16 \cdot OPT + 1$*, where* OPT *is the size of an optimal solution.*

3.3 Approximation Algorithm for Discrete-G-Min-Disc-Code

In this section, we modify the algorithm of Sect. 3.2 for Continuous- G- Min- Disc- Code to solve Discrete- G- Min- Disc- Code. Recall that here, in addition to the set of points P (in $\mathbb{R}^2$), the set S of axis-parallel unit squares is also given in the input. As in Sect. 3.2, Discrete- G- Min- Disc- Code reduces to the *discrete* version of the Min- UHR- Hit- Set problem, whose objective is to hit a set $\mathcal{O}$ of unit width/height rectangles by choosing a minimum cardinality subset of a given set of points Q. Unlike the continuous version (Lemma 15), the discrete version of the hitting set problem for a set of unit-height rectangles intersected by a horizontal line cannot be solved in polynomial time, since the points to be used for hitting the rectangles are already specified (cannot be chosen suitably). However, if we can design an LP-based α-factor approximation algorithm for the discrete version of Min- UHR- Hit- Set when the unit-height rectangles are intersected by a single horizontal line, then we can use it to get a 16α-factor approximation algorithm for the discrete version of Discrete- G- Min- Disc- Code problem (see Eqs. (3) and (4)). (By *LP-based* we mean that we must be able to compute a solution that is at most α times the optimal solution size of the LP for Discrete- Min- UHR- Hit- Set when all rectangles are intersected by a horizontal line.)

We will describe an LP-based 4-factor approximation algorithm for the Discrete- Min- UHR- Hit- Set problem where the rectangles are hit by a horizontal line (see Lemma 22 stated at the end of the section). Thus, this will imply the following.

Theorem 18 *There exists a polynomial-time algorithm for the* DISCRETE- G- MIN-DISC- CODE *problem for axis-parallel unit squares which produces a solution of size at most* $64 \cdot OPT + 1$, *where* OPT *is the size of an optimal solution.*

3.3.1 DISCRETE-MIN-UHR-HIT-SET for Rectangles Stabbed by a Single Horizontal Line

Let us first solve a restricted version of the DISCRETE- MIN- UHR- HIT- SET problem, where the input is a set of axis-parallel unit-height rectangles R intersected by a single horizontal line λ and a set of points Q (see Fig. 11). The objective is to choose a minimum number of points from Q to hit all the rectangles in R. This problem can be formulated as the following ILP.

$$\mathcal{U}_\lambda : \min \sum_{q_\alpha \in Q} x_\alpha$$

$$\text{Subject to } \sigma_1(r) + \sigma_2(r) \geq 1, \text{ for all } r \in R,$$

where $\sigma_1(r)$ (resp. $\sigma_2(r)$) is the sum of the variables corresponding to the points in Q above (resp. below) the line λ that lie inside the rectangle r. We will use OPT_λ to denote the optimum solution of this ILP.

On the basis of the LP relaxation of this ILP, we can partition the rectangles into two groups: R^a and R^b, such that R^a (resp. R^b) contains the rectangles whose solution in the LP relaxation satisfies $\sigma_1(r) \geq \sigma_2(r)$ (resp. $\sigma_1(r) < \sigma_2(r)$). The rectangles in R^a (resp. R^b) will thus be assumed to be hit by the points in Q^a (resp. Q^b) that lie above (resp. below) the line λ; $Q^a \cup Q^b = Q$, $Q^a \cap Q^b = \emptyset$.

Let $\mathcal{U}^a_\lambda$ and $\mathcal{U}^b_\lambda$ be the ILPs for the minimum hitting set problems for the rectangles in R^a and points in Q^a (resp. R^b and Q^b). As in the relation of $\overline{OPT}_{\mathcal{A}}$, $\overline{OPT}_{\mathcal{B}}$ and

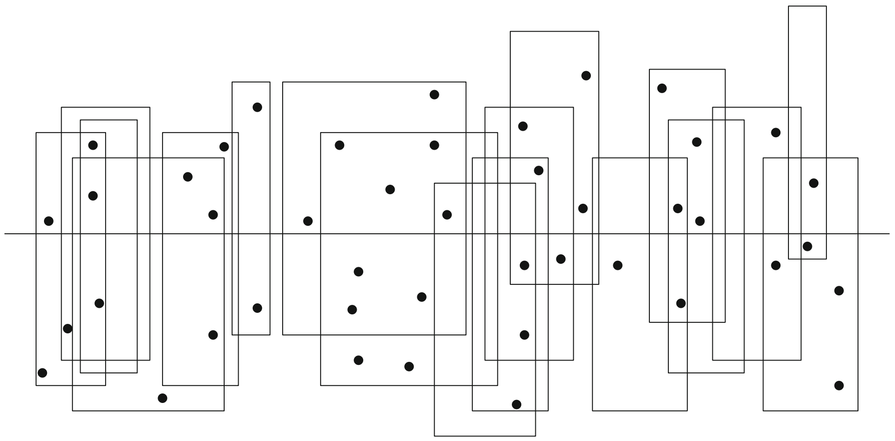

Fig. 11 An instance of discrete hitting set of unit-height rectangles stabbed by a horizontal line

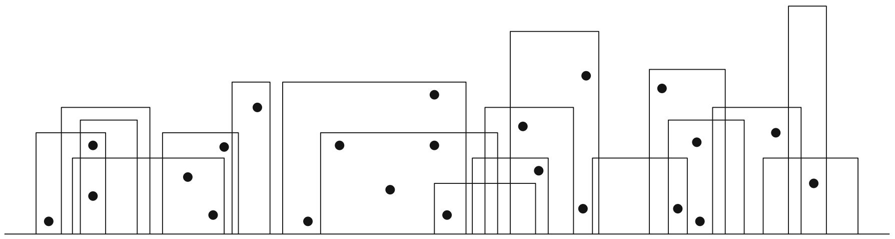

Fig. 12 The instance where the rectangles above the horizontal line are considered

$\overline{OPT}_1$ in Eq. (2), here also denoting by $\overline{OPT}^a_\lambda$ and $\overline{OPT}^b_\lambda$ as the optimum solutions of the LP relaxation of $\mathcal{U}^a_\lambda$ and $\mathcal{U}^b_\lambda$, respectively, we have

$$\overline{OPT}^a_\lambda + \overline{OPT}^b_\lambda \leq 2\overline{OPT}_\lambda \tag{5}$$

due to the fact that Q^a and Q^b are disjoint point sets and $2\overline{OPT}_\lambda$ is a feasible solution of both the problems $\overline{\mathcal{U}}^a_\lambda$ and $\overline{\mathcal{U}}^b_\lambda$. Now, we concentrate on solving the ILP $\mathcal{U}^a_\lambda$. The problem $\mathcal{U}^b_\lambda$ can be solved in a similar manner.

3.3.2 Approximation Algorithm for Solving $\mathcal{U}^a_\lambda$

We have a set of rectangles R^a, each one having its bottom boundary aligned with a horizontal line λ, and a set of points $Q^a \subseteq Q$ that lie above the line λ. The objective is to find a subset of Q^a of minimum size, say OPT^a, to hit all the members in R^a (see Fig. 12).

Note that this problem can be solved in polynomial time by a dynamic programming technique similar to the one used in [23]. However, in order to plug-in the solution of this problem into Eq. (5), we need that the solution is obtained using the LP relaxation of its corresponding ILP formulation (it may not be optimal, but we need a guaranteed approximation factor).

We will use known techniques from the literature to get an integer-valued 2-factor approximation algorithm for the ILP problem $\mathcal{U}^a_\lambda$ obtained from its LP solution (see Eq. (6)).

$$\begin{aligned} \mathcal{U}^a_\lambda : \min \sum_{q_\alpha \in Q^a} x_\alpha \\ \text{Subject to} \sum_{q_\alpha \in Q^a \cap r} x_\alpha \geq 1 \text{ for all } r \in R^a, \\ x_\alpha \in \{0, 1\} \text{ for all } q_\alpha \in Q^a \end{aligned} \tag{6}$$

Definition 19 Let $\epsilon > 0$ be fixed and consider a set system $(X, \mathcal{R})$. A set $N \subseteq X$ is an *ϵ-net of* $(X, \mathcal{R})$ if for every subset $S \in \mathcal{R}$ for which $|S| \geq \epsilon|X|$, N contains a point of S.

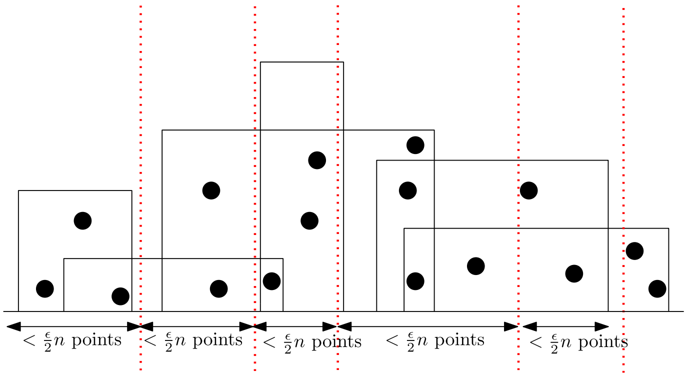
$< \frac{\epsilon}{2}n$ points
$< \frac{\epsilon}{2}n$ points
$< \frac{\epsilon}{2}n$ points
$< \frac{\epsilon}{2}n$ points
$< \frac{\epsilon}{2}n$ points

Fig. 13 Illustration of the existence of a $\frac{2}{\epsilon}$-size ϵ-net for the set system (Q^a, R^a)

In other words, an ϵ-net is a hitting set of the set system over X containing only large enough sets of $\mathcal{R}$.

Lemma 20 ([36]) *For a set system (Q^a, R^a) and a horizontal line λ such that Q^a is a set of points lying above λ and R^a is a set of rectangles each having their base on λ, for every positive $\epsilon > 0$, an ϵ-net of size $\frac{2}{\epsilon}$ can be constructed in polynomial time.*

Proof We split the half-plane above the line λ into disjoint vertical strips such that each strip contains $\lfloor \frac{\epsilon n}{2} \rfloor$ points (see Fig. 13). Now, from each strip, we choose the bottom-most points. This is an ϵ-net of size $\frac{2}{\epsilon}$ for the set system (Q^a, R^a) due to the fact that for any rectangle containing at least ϵn points, its horizontal span must contain that of a strip, and hence the corresponding rectangle is hit by the point chosen in that strip. □

The following theorem is proved in [14].

Theorem 21 ([14]) *An ϵ-net of size $\frac{d}{\epsilon}$ for a set system $(X, \mathcal{R})$ with $\epsilon = 1/\overline{OPT}$, where $\overline{OPT}$ is the optimal value of the LP for* Hitting Set *on $(X, \mathcal{R})$, is a hitting set of $(X, \mathcal{R})$ of size at most $d \cdot \overline{OPT}$.*

Finally, plugging-in Lemma 20 and Theorem 21 with $d = 2$ in Eq. (5), we have the following result needed to complete the proof of Theorem 18.

Lemma 22 *In polynomial time, one can compute a solution for* Discrete- Min-UHR- Hit- Set *when all rectangles are intersected by a horizontal line, whose size is at most 4 times the optimal value of the corresponding LP.*

4 Min-ID-Code for Geometric Intersection Graphs

In this section, we will use techniques similar to those used in the previous sections and apply them to the setting of the graph problem Min- ID- Code, for the intersection graph of axis-parallel unit squares (unit square graphs). To the best of our knowledge, Min- ID- Code was not yet studied for unit square intersection graphs in the literature.

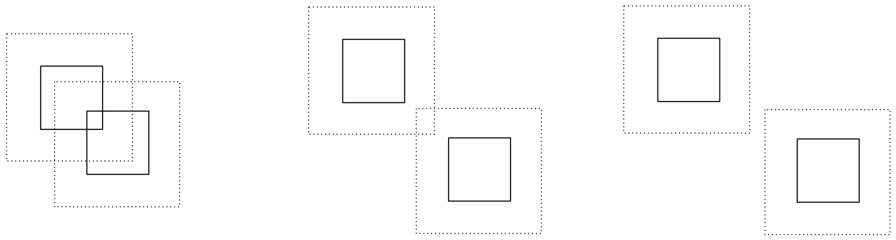
Fig. 14 Possible intersection patterns of a pair of axis-parallel unit squares (full lines): the dotted square around each square corresponds to the locations where a square centered at this point will intersect the enclosed unit square

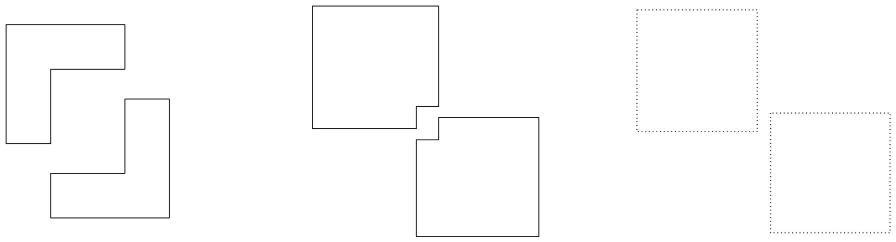
Fig. 15 Feasible regions for placing the center of the square $s \in ID$ for discriminating $s_i, s_j \in S$: three possible situations

Here, the input is a set S of axis-parallel unit squares in 2D. In the graph $G = (V, E)$, the nodes in $V = \{v_1, \ldots, v_n\}$ correspond to the squares in S; an edge $e_{ij} = \{v_i, v_j\} \in E$ if the squares corresponding to v_i, v_j intersect.

Note that it is not known in the literature whether Min- Id- Code on unit square graphs is NP-hard, however, the techniques from [31] used for unit disk graphs can most certainly be applied to prove it.

We can reformulate Min- ID- Code for unit square graphs in geometric terms: the objective is to compute a subset $S_{opt} \subseteq S$ of minimum cardinality such that each square in S intersects some square in S_{opt}, and for each pair of squares $s_i, s_j \in S$, there exists a square $\sigma \in S_{opt}$ such that ($\sigma \cap s_i \neq \emptyset$ and $\sigma \cap s_j = \emptyset$) or ($\sigma \cap s_i = \emptyset$ and $\sigma \cap s_j \neq \emptyset$). If we do only satisfy the discrimination constraint, then in order to satisfy the domination constraint, we may need to include at most one more square from S in S_{opt}.

Let $S' \subseteq S$ be an identifying code for the set of squares in S. A square $\sigma \in S'$ intersects a square $s_i \in S$ if the center of σ is placed inside the square δ centered at the center of s and the side-length of δ is twice the side-length of s (shown using dotted line around s_i in Fig. 14). In order to satisfy the discrimination constraint among $s_i, s_j \in S$, the center of a square $\sigma \in S'$ must be placed inside $\delta_i \nabla \delta_j$, where ∇ is the symmetric difference operator, i.e., $(\delta_i \backslash \delta_j) \cup (\delta_j \backslash \delta_i)$. In Fig. 14, different patterns of intersection of a pair $s_i, s_j \in S$ are depicted, along with their covering regions δ_i, δ_j. Thus, in order to satisfy the discrimination constraint (s_i, s_j), we need to place the center of a square $\sigma \in S'$ in the regions shown in Fig. 15.

Thus, as in Sect. 3.2, we can solve Min- ID- Code for unit square graphs by solving a problem of hitting the feasible regions corresponding to each pair of squares $s_i, s_j \in S$ using the centers of the squares in S. The objective will be to choose the minimum number of hitting squares from S. Thus, the same techniques as in Sect. 3.2 can be applied, and we obain the following theorem.

Theorem 23 Min- ID- Code *has a polynomial-time approximation algorithm for unit square graphs (if the unit square intersection model of the input graph is known) that produces a solution of size at most* $64 \cdot OPT + 1$*, where* OPT *is the size of an optimal solution.*

5 Conclusion

We have seen that Discrete- G- Min- Disc- Code is NP-complete, even in 1D. This is in contrast with most covering problems and to Continuous- G- Min- Disc- Code, which are polynomial-time solvable in 1D [19, 25].

We also proposed a simple 2-factor approximation algorithm for the Discrete- G- Min- Disc- Code problem in 1D, and a PTAS for a special case where each interval in the set S is of unit length. It seems challenging to determine whether Discrete- G- Min- Disc- Code problem in 1D becomes polynomial-time for unit intervals. As noted in [19], this would be related to Min- ID- Code on *unit* interval graphs, which also remains unsolved [18]. In fact, it also seems to be unknown whether Continuous- G- Min- Disc- Code problem in 1D remains polynomial-time solvable with the restriction that each interval is of unit length. However our PTAS algorithm for Discrete- G- Min- Disc- Code problem in 1D also produces a PTAS for the Continuous- G- Min- Disc- Code problem. We also do not know whether a PTAS exists for the general 1D case.

In 2D, both Continuous- G- Min- Disc- Code and Discrete- G- Min- Disc- Code problems are NP-complete even when S must be a set of axis-parallel unit square objects. We propose polynomial-time constant factor approximation algorithms for both Continuous- G- Min- Disc- Code and Discrete- G- Min- Disc- Code using the rounding of the relaxation of integer programming to linear programming. The question remains whether there exists any algorithm with better constant approximation or a PTAS.

We remark that all our results for axis-parallel unit squares also hold when the objects are axis-parallel rectangles of a specified (same) size i.e., the shape of all the rectangles match in height and width.

As we observed, all the techniques for designing approximation algorithms for solving Discrete- G- Min- Disc- Code problems in 2D work for solving the Min- ID- Code problem for an intersection graph of a set of axis-parallel rectangles of same size. It is an interesting problem whether similar approximation algorithms exist, where the objects in S are unit disks, or arbitrary axis-parallel rectangles.

In the recent literature, a problem known as Red- Blue Separation is being studied. In this problem, given a set R of red-colored points and a set B of blue-colored points in the plane, the objective is to find at most k geometric objects that separate

R from B, that is, each cell in the arrangement of these geometric objects contains points of at most one color (see [33] and the references in that paper). The problem G-Min-Disc-Code studied here can be seen as one where each point has different color. Our techniques for both the continuous and discrete versions of G-Min-Disc-Code problems will also work for the Red-Blue-Separation Problem with axis-parallel rectangles of same size in the continuous and discrete cases, respectively. Here, instead of considering segments joining every pair of points in the given point set, we join each red point in the set R with each point in the set B of blue points, and solve the segment stabbing problem with a set of rectangles (suitably positioned in the continuous case and a given set of rectangles in the discrete case). The approximation factors of both these problems will be the same as those obtained for the corresponding version of G-Min-Disc-Code.

Acknowledgements We thank the anonymous referees for their helpful comments that helped us improving the quality of the paper. We also thank Rajiv Raman for helpful discussions and for sharing his lecture notes [36] with us.

Declarations

Conflict of interest We do not have such Conflict of interest.

References

1. Acharyya, A., Nandy, S.C., Pandit, S., Roy, S.: Covering segments with unit squares. Comput. Geom. **79**, 1–13 (2019)
2. Agarwal, P.K., van Kreveld, M., Suri, S.: Label placement by maximum independent set in rectangles. Comput. Geom. **11**(3), 209–218 (1998)
3. Basu, K., Dey, S., Nandy, S.C., Sen, A.: Sensor networks for structural health monitoring of critical infrastructures using identifying codes. In: 15th International Conference on the Design of Reliable Communication Networks (DRCN), pp. 43–50 (2019)
4. Bazgan, C., Foucaud, F., Sikora, F.: Parameterized and approximation complexity of partial VC dimension. Theor. Comput. Sci. **766**, 1–15 (2019)
5. Boland, R.P., Urrutia, J.: Separating collections of points in Euclidean spaces. Inf. Process. Lett. **53**, 177–183 (1995)
6. Bousquet, N., Lagoutte, A., Li, Z., Parreau, A., Thomassé, S.: Identifying codes in hereditary classes of graphs and VC-dimension. SIAM J. Discrete Math. **29**, 2047–2064 (2015)
7. Charbit, E., Charon, I., Cohen, G.D., Hudry, O.: Discriminating codes in bipartite graphs. Electron. Notes Discrete Math. **26**, 29–35 (2006)
8. Charon, I., Cohen, G.D., Hudry, O., Lobstein, A.: Discriminating codes in (bipartite) planar graphs. Eur. J. Comb. **29**, 1353–1364 (2008)
9. Charon, I., Hudry, O., Lobstein, A.: Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard. Theor. Comput. Sci. **290**, 2109–2120 (2003)
10. Călinescu, G., Dumitrescu, A., Karloff, H., Wan, P.-J.: Separating points by axis-parallel lines. Int. J. Comput. Geom. Appl. **15**(06), 575–590 (2005)
11. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer-Verlag TELOS, USA (2008)
12. de Bontridder, K.M.J., Halldórsson, B.V., Halldórsson, M.M., Hurkens, A.J., Lenstra, J.K., Ravi, R., Stougie, L.: Approximation algorithms for the test cover problem. Math. Program. **98**, 477–491 (2003)
13. Dey, S., Foucaud, F., Nandy, S.C., Sen, A.: Discriminating Codes in Geometric Setups. In: Cao,Y., Cheng, S.-W., Li, M. (eds) 31st International Symposium on Algorithms and Computation (ISAAC

2020), vol. 181 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 24:1–24:16, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2020)
14. Even, G., Rawitz, D., Shahar, S.: Hitting sets when the VC-dimension is small. Inform. Process. Lett. **95**, 358–362 (2005)
15. Foucaud, F.: *Combinatorial and algorithmic aspects of identifying codes in graphs*. In: PhD thesis, Université Bordeaux 1, France (2012)
16. Foucaud, F.: Decision and approximation complexity for identifying codes and locating-dominating sets in restricted graph classes. J. Discrete Algorithms **31**, 48–68 (2015)
17. Foucaud, F., Gravier, S., Naserasr, R., Parreau, A., Valicov, P.: Identifying codes in line graphs. J. Graph Theory **73**, 425–448 (2013)
18. Foucaud, F., Mertzios, G.B., Naserasr, R., Parreau, A., Valicov, P.: Identification, location-domination and metric dimension on interval and permutation graphs II. Algorithms and complexity. Algorithmica **78**, 914–944 (2017)
19. Gledel, V., Parreau, A.: Identification of points using disks. Discrete Math. **342**, 256–269 (2019)
20. Habib, M., Paul, C., Viennot, L.: A synthesis on partition refinement: a useful routine for strings, graphs, boolean matrices and automata. In: Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, STACS '98, pp 25–38, Berlin, Heidelberg. Springer-Verlag (1998)
21. Har-Peled, S., Jones, M.: On separating points by lines. Discrete Comput. Geom. **63**, 705–730 (2020)
22. Karpovsky, M.G., Chakrabarty, K., Levitin, L.B.: On a new class of codes for identifying vertices in graphs. IEEE Trans. Inf. Theory **44**, 599–611 (1998)
23. Katz, M.J., Mitchell, J.S.B., Nir, Y.: Orthogonal segment stabbing. Comput. Geom. **30**(2), 197–205 (2005)
24. Knuth, D.E.: The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd edn. Mass, Addison-Wesley, Reading (1997)
25. D. Krupa R., A. Basu Roy, M. De, and S. Govindarajan. Demand hitting and covering of intervals. In: Algorithms and Discrete Applied Mathematics (CALDAM'17), pp 267–280, Cham. Springer International Publishing (2017)
26. Laifenfeld, M., Trachtenberg, A.: Identifying codes and covering problems. IEEE Trans. Inf. Theory **54**, 3929–3950 (2008)
27. Laifenfeld, M., Trachtenberg, A., Cohen, R., Starobinski, D.: Joint monitoring and routing in wireless sensor networks using robust identifying codes. Mob. Netw. Appl. **14**, 415–432 (2009)
28. Lee, D.T.: Interval, segment, range, and priority search trees. In: Mehta, D.P., Sahni, S. (eds.) Handbook of Data Structures and Applications. Chapman and Hall/CRC (2004)
29. McCreight, E.M.: Priority search trees. SIAM J. Comput. **14**(2), 257–276 (1985)
30. Micali, S., Vazirani, V.V.: An O(sqrt(|V|) |E|) algorithm for finding maximum matching in general graphs. In: Annual Symposium on Foundations of Computer Science, pp. 17–27 (1980)
31. Müller, T., Sereni, J.-S.: Identifying and locating-dominating codes in (random) geometric networks. Comb. Probab. Comput. **18**(6), 925–952 (2009)
32. Mustafa, N.H., Ray, S.: Improved results on geometric hitting set problems. Discret. Comput. Geom. **44**, 883–895 (2010)
33. Misra, H. M. N., Sethia, A.: Red-blue point separation for points on a circle. In: Proceedings of the 32nd Canadian Conference on Computational Geometry (CCCG) (2020)
34. Nandy, S.C., Asano, T., Harayama, T.: Shattering a set of objects in 2D. Discret. Appl. Math. **122**(1), 183–194 (2002)
35. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall Inc, Upper Saddle River, NJ, USA (1982)
36. Raman, R.: Private Communication(2022)
37. Ray, S., Starobinski, D., Trachtenberg, A., Ungrangsi, R.: Robust location detection with sensor networks. IEEE J. Sel. Areas Commun. **22**, 1016–1025 (2004)
38. Schrijver, A., Berlin, S.-V.: Combinatorial optimization: polyhedra and efficiency. Number vol. 1 in Algorithms and Combinatorics. Springer (2003)
39. Thorup, M.: Undirected single-source shortest paths with positive integer weights in linear time. J. ACM **46**(3), 362–394 (1999)
40. Tovey, C.A.: A simplified NP-complete satisfiability problem. Discret. Appl. Math. **8**(1), 85–89 (1984)
41. van Bevern, R., Bredereck, R., Bulteau, L., Chen, J., Froese, V., Niedermeier, R., Woeginger, G.J.: Partitioning perfect graphs into stars. J. Graph Theory **85**(2), 297–335 (2017)

Authors and Affiliations

Sanjana Dey[1,6] · Florent Foucaud[2,3,4] · Subhas C. Nandy[1] · Arunabha Sen[5]

Sanjana Dey
info4.sanjana@gmail.com

Florent Foucaud
florent.foucaud@uca.fr

Arunabha Sen
arunabha.sen@asu.edu

[1] ACM Unit, Indian Statistical Institute, Kolkata, India

[2] LIMOS, CNRS UMR 6158, Université Clermont Auvergne, Aubière, France

[3] LaBRI, UMR5800, CNRS, Bordeaux INP, University of Bordeaux, 33400 Talence, France

[4] LIFO EA 4022, INSA Centre Val de Loire, University of Orléans, 45067 Orléans, France

[5] Arizona State University, Tempe, AZ 85287, USA

[6] School of Computing, National University of Singapore, Singapore, Singapore