Topology and context-based pattern extraction using line-segment Voronoi diagram

Sandeep Kumar Dey[a], Panagiotis Cheilaris[a], Nathalie Casati[b], Maria Gabrani[b], Evanthia Papadopoulou[a]

[a]Università della Svizzera italiana, Faculty of Informatics, Via Giuseppe Buffi 13, Lugano, Switzerland, CH-6904
[b]IBM-Research Zurich, Säumerstrasse 4, 8803, Rüschlikon, Switzerland

Abstract. Early identification of problematic patterns in real designs is of great value as the lithographic simulation tools face significant timing challenges. To reduce the processing time such a tool selects only a fraction of possible patterns which have a probable area of failure, with the risk of missing some problematic patterns. In this paper, we introduce a fast method to automatically extract patterns based on their structure and context, using the Voronoi diagram of line-segments derived from the edges of VLSI design shapes. We first use the line-segment Voronoi diagram to derive possible problematic locations and then we use the derived locations for extracting the problematic windows or patterns from the design layout. The problematic locations are prioritized by the shape and proximity information of design polygons. We did experiments for pattern selection in a portion of a 22nm random logic design layout. The design layout had 38584 design polygons (made of 199946 line-segments), Mx layer, and 7079 ORC (*Optical Rule Checker*) generated markers. We verify our approach by comparing the coverage of our extracted patterns to the ORC generated markers.

Keywords: Voronoi, line-segment, photolithography, pattern selection, ORC, OPC.

1 Introduction

With the increase in miniaturization of current VLSI patterns, there is a significant rise in printability problems of such patterns, during the photolithography process. The analysis of patterns to find faults or error-prone locations, is of prime importance to the manufacturing process. There are mainly two kinds of faults that can occur during printing: a *pinch* and a *bridge*. A pinch corresponds to an open fault and it occurs due to incomplete printing of a shape or due to discontinuity in the printing of a shape. A bridge corresponds to a short fault and occurs when two printed shapes are touching each other.

The analysis of a complete layout for finding faults or error-prone locations is difficult and very time-consuming. The printability of a layout is related to the patterns it contains. Therefore, pattern selection should be done in such a way so that the analysis of the selected patterns is sufficient to assess the quality of the whole layout. In other words, it is important to identify patterns, known as *patterns of interest* (POI), which are prone to faults. A relatively small number of POIs that cover sufficiently the whole layout allow for a faster yet effective analysis of the layout. Obtaining an optimal set of POIs is a big challenge in this domain. The success of printing a POI is verified by taking several measurements, such as critical distance, on potentially critical areas of *scanning electron microscope* (SEM) images of the printed pattern. Therefore, the location of measurement is very important for proper evaluation of a POI.

In each pattern the measurement location is called a *gauge*.[1] The gauge is typically represented by a line in the VLSI pattern, around which a critical distance is measured. Gauge locations must

be meaningful, that is, the critical distance measurement around the gauge location should be the correct measurement for the pattern. Current gauge suggestion techniques are rule-based or they are done manually by VLSI designers. The suggested gauges very often miss the location of the critical distance or the location of faults. The actual location of faults within a POI is known as a *hotspot*. Gauges are the indicators that help to categorize a pattern as POI and locate hotspots within the pattern. Good gauge suggestions improve the evaluation of a pattern, and contribute to the goal of achieving an optimal set of POIs.

In the literature, there are many variants of hotspot identification, like, machine learning algorithms,[2,3] image recognition techniques,[1,4,5] design based approach,[6] pattern matching techniques,[7,8] and topology oriented techniques.[9,22] For machine learning techniques, the learning time is high and there is a need of already available hotspots for training set. Image recognition techniques need to go over the whole layout for hotspot detection which is very time consuming. Pattern matching techniques work on a predefined set of hotspot patterns and has a limitation of detecting unseen hotspots. The design based approach also been observed to be very time consuming as it needs to analyze the entire design. In general, the random nature of layout patterns are difficult to predict for all the above mentioned methods. Topology oriented pattern extraction techniques[9] can be useful to handle random patterns, although it do not claim to be general method to detect all hotsots in a layout. In this paper, we introduce a new topology oriented technique for pattern extraction, based on the geometric information of layout shapes.

The set of POI and hotspots are of prime importance to address printability problems. After obtaining the set of POI and hotspot, designers modify the mask designs accordingly, with a goal to improve the level of yield. This needs a feedback-model for verification. VLSI designers have developed several models based of POI, including *model-based optical proximity correction*.[1,5,10] Hotspots are feedback to the OPC process for better yield. An optimized MB-OPC demands an optimal set of problematic patterns, but identifying such a set in a time efficient manner is very hard. Currently reliable OPC models are mainly based on image parameters of the test patterns.[1,5,10] The techniques involving image parameters are computationally very expensive. Another problem is the automation of identification of problematic patterns. In many cases, expertise of lithographers and design engineers provides the problematic patterns by manual inspection of the layout. Some new automatic approaches[11,12] use a combination of parameters. These techniques sample the full spectrum of patterns and thus tend to be computationally expensive. We focus on the automatic and fast identification of problematic patterns.

For the feedback to OPC models, VLSI designers have developed tools to identify optical rule violation in the simulated lithographic patterns. These tools are used for checking the feature size that can hinder to achieve an acceptable level of yield of the chip. An Optical rule checker (ORC)[13] is a program that encodes and verifies the rules for ideal simulated lithographic patterns. Given the lithographic processing conditions, an ORC run generates markers on the violation of an ORC rule. Therefore, the problematic patterns in the layout are around the ORC markers. These set of patterns can be feedback to OPC for better yield in manufacturing.

In this paper we provide a fast automatic approach to derive a near optimal set of problematic patterns for a layout; which lay a foundation for an approach to obtain sets of patterns that can be potentially used for calibration and verification of MB-OPC. In particular, we find gauge locations using the line-segment Voronoi diagram of layout shapes and give priority to the gauges depending upon the shape and proximity information of the design polygons. The gauge locations are then used to extract windows from the design layout. We extract one window per gauge location. The

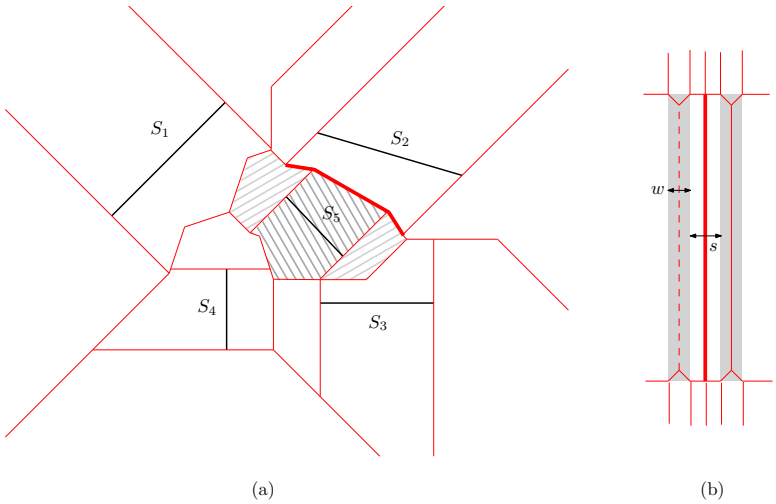

Fig 1 (a) Segment Voronoi diagram under L_∞ metric, with five distinct sites S_1, S_2, S_3, S_4, S_5, for interiors of segments and their endpoints. The lightly shaded portion is the Voronoi region of the endpoints of the segment S_5, dark shaded portion is the Voronoi region of the interior of the segment S_5. The thick red line is the Voronoi edge separating the faces defined by segments S_5 and S_2, (b) Voronoi diagram (in red) of two design polygons (in gray), w is the width parameter and is encoded by a Voronoi edge shown in red dashed line, and s is the space parameter encoded by a Voronoi edge shown in thick red line.

windows contain patterns which are potentially problematic. Finally, we verify the usefulness of the extracted windows by comparing the coverage of the problematic patterns with respect to the ORC generated markers. We observe that the set of patterns extracted by our tool covers all the ORC generated markers for the given layout.

The rest of this work is organized as follows. In section 2, we introduce the line-segment Voronoi diagram in the L_∞ metric as used in the VLSI pattern analysis. We describe the gauges and their scoring method in section 3. We describe our method to detect potentially critical locations in the VLSI design layout using the segment Voronoi diagram and then describe our pattern selection procedure in section 4. We discuss our experimental results on the design layout in section 5.

2 The L_∞ line-segment Voronoi diagram

Let S be a set of sites (simple geometric objects such as points, line-segments, or simple polygons) in the plane. The *nearest-neighbor Voronoi diagram* (Voronoi diagram)[14,15] of S is a subdivision of the plane into regions such that the region of a site $s \in S$ is the locus of points closer to s than to any other site in S. The *distance* of a site s from a point q in the plane is $d(s,q) = \min_{p \in s} d(p,q)$, where the the interpoint distance $d(p,q)$ can be the Euclidean (L_2) distance, the Manhattan (L_1, L_∞) distance, or any other metric. In this paper, we use the L_∞ metric (or maximum norm), where the distance between two points p and q is given by : $d(p,q) = d_\infty(p,q) = \max(|p_x - q_x|, |p_y - q_y|)$.

The Voronoi diagram of S is a planar graph of linear complexity on the total complexity of the input sites. Each *face* corresponds to the Voronoi region of a site $s \in S$: $reg(s) = \{q \in \mathbb{R}^2 \mid d(s,q) < d(s',q), \forall s' \in S \setminus \{s\}$. Each region contains its defining site. Figure 1(a) illustrates the Voronoi diagram of a set of line-segments $\{S_1, S_2, S_3, S_4, S_5\}$. The shaded region is the Voronoi region of line-segment S_5 and it contains S_5. The boundary between two neighboring faces is

an *edge* of the diagram (see e.g. the thick red line in Figure 1(a)). A Voronoi edge is the locus of points equidistant from the respective defining sites of the two neighboring faces. Edges meet at *vertices* of the diagram. At least three Voronoi edges meet at a Voronoi vertex. The Voronoi diagram encodes proximity information of the input sites. The time complexity of construction algorithms for Voronoi diagram of S is $O(n \log n)$,[14,15] where n is the total complexity of the input sites. For more information on Voronoi diagrams see the book of Aurenhammer et al.[15]

A VLSI layout is composed of design polygons which are mostly rectilinear. There are two important parameters, *width* and *space*, that describe patterns in a VLSI layout. The width parameter is defined as the distance between two parallel edges of a design polygon. A Voronoi edge internal to a design polygon encodes this width parameter (see e.g. the thin dashed line in Fig. 1(b)). The space parameter is defined as the distance of separation of two design polygons. A Voronoi edge induced by edges of two different design polygons encodes the space parameter (see e.g. the thick red line in Fig. 1(b)).

For point sites in a plane, the Voronoi diagram is available through MATLAB[16] (matrix laboratory), which is used in several engineering principles. In a VLSI layout, the Voronoi diagram of polygons or their edges is required. This is available through the CGAL library,[17] currently in the standard Euclidean metric.[18] CGAL is an open source C++ library that provides easy access to efficient and reliable geometric algorithms. We have developed the L_∞ line-segment Voronoi diagram in the CGAL environment,[22,23] which is submitted for inclusion in the library and is currently under review.

Voronoi diagrams of line segments in the L_∞ metric have nice properties, especially for the VLSI environment where shapes are predominantly rectilinear. Unlike its Euclidean counterpart, the L_∞ line-segment Voronoi diagram consists exclusively of straight line-segments, and Voronoi vertices are on rational coordinates. That is, for rectilinear layouts, the Voronoi diagram consists of rectlinear edges and edges of slope ± 1 and thus Voronoi vertices appear on a grid only slightly finer than the input coordinate grid. In this environment, use of the L_∞ metric is desirable because of its simplicity and its already proven utility in the area of VLSI CAD, in particular, the critical area problem.[19–21]

3 Gauge suggestion using the line-segment Voronoi diagram

For a given layout, we identify gauge locations. We use the L_∞ line-segment Voronoi diagram to suggest good gauge locations based on the proximity information of the shapes of a pattern, e.g., the spacing between shapes or the extent of interaction between neighboring shapes. We suggest five types of gauges, *internal, external, sandwich, comb* and *T*, as illustrated in Fig. 2. Figure 3 illustrates different gauges in a portion of a design layout.

Following are the description of the above mentioned type of gauges:

1. *Internal gauge* (inside a shape), G_i: This gauge lies on the center of a Voronoi edge in the interior of the polygonal shape of minimum width in the pattern (see G_i in Fig. 2). The position of G_i is a probable location for a pinch, when printing the pattern.

2. *External gauge* (between different neighboring shapes), G_e: This gauge lies on the center of the Voronoi edge between the two shapes that are closest in the pattern (see G_e in Fig. 2). The position of G_e is a probable location for the formation of a bridge between the two corresponding shapes, when printing the pattern.

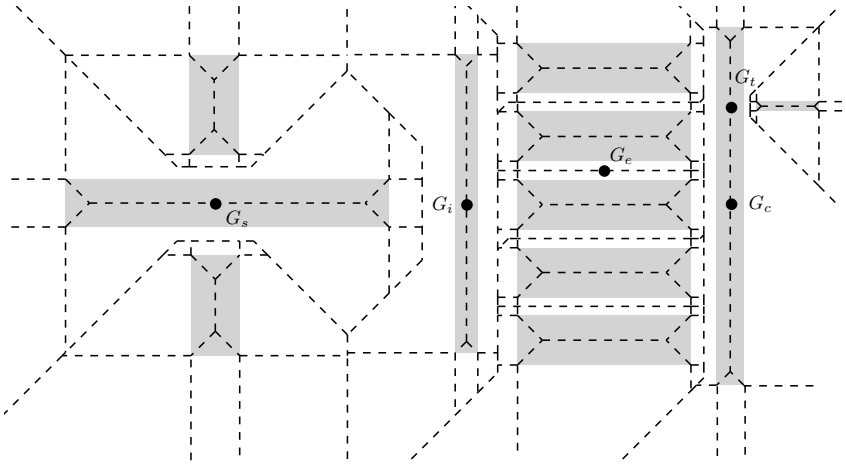

Fig 2 Different gauge suggestions.

3. *Sandwich gauge*, G_s: This gauge lies on the center of the Voronoi edge inside a polygonal shape P_1 that is "sandwiched" between two other shapes P_2 and P_3 for which the distance between P_2 and P_3 is the minimum in the pattern (see G_s in Fig. 2). There is a probability of a pinch happening at P_1 around G_s because of the influence of P_2 and P_3.

4. *Comb gauge*, G_c: This gauge lies on the center of the Voronoi edge inside a long polygonal shape, which is the base of the comb and it has close to it and on one side of it a number of polygonal shapes which are the teeth of the comb. We report the gauge for the configuration where the base of the comb shape is closer to the teeth in the pattern (see G_c in Fig. 2). The position of G_c is dangerous for a pinch, when printing the pattern.

5. *T gauge*, G_t: A comb gauge at a minimum must contain one tooth and a base. We call such a minimal comb gauge a *T gauge* (see G_t in Fig. 2). The position of G_t is a probable location for a pinch, when printing the pattern.

We have validated the usefulness of our gauges by experiments on small size patterns in a recent paper.[22] For each small size pattern we suggested gauges. We then measured the distance in pixels in the corresponding SEM image for each suggested gauge and we took the minimum. In most of the patterns we improved the critical distance measurement and in some cases we were able to locate hotspot missed by suggested gauges of conventional method. This ability may find additional uses in the area, in particular, potentially these gauges can be used in the model generation of MB-OPC.

We introduce a scoring method for gauges, which is used for prioritizing the gauges to determine problematic patterns efficiently. For each gauge we provide a score. The score associated with a gauge determines its affinity towards the printing problem. The score for a gauge indicates the potential of failure around the location of the gauge, when printing the related pattern.

3.1 Scoring method for gauges

For each type of gauge we defined scores as follows:

- Score of an internal gauge: Let P_i be a shape in the design layout. The score of an internal gauge is the minimum width value in P_i (w in Fig. 4 (a)). In case there are shapes with same width, we break ties using *extension* parameter. The extension parameter is the length of the Voronoi edge associated with an internal gauge. The extension parameter does not change the value of the score, it is just used to change the order of the gauge in the priority list in case of ties. The gauge associated with the longer shape that is having a greater value of associated extension parameter gets more priority and will be ranked higher in the ordered list of internal gauges. Lower width implies a thinner and a greater extension implies a longer shape; a thin long shape has more probability to give rise to a pinch.

- Score of an external gauge: Let P_a and P_b be two neighboring shapes in the design layout. The score of an external gauge associated with P_a and P_b is the separating distance (s in the Figure 4 (b)) between P_a and P_b. This is encoded by the associated Voronoi edge between P_a and P_b. When there are gauges with the same score, we break ties by extension parameter. The gauge whose asscociated Voronoi edge is longer gets more priority, as it implies more interaction between the shapes and higher probability of a bridge.

- Score of a sandwich gauge: Let P_x, P_y, and P_z be three shapes in a design layout such that P_y is sandwiched between P_x and P_z. We define the score for the sandwich gauge equal to the $0.5\times$ d (see Fig. 4 (c)), where d is the distance between the Voronoi edges E_{xy} and E_{yz} (see Figure 4 (c)). If there are (P_x, P_y, P_z) triples with the same score, we use *overlap* parameter to break ties; a sandwich gauge with a greater overlap parameter gets more priority. The overlap parameter is the measure of the length of the overlapping portion of Voronoi edges E_{xy}, E_y, and E_{yz} associated with the sandwich configuration (see Fig. 4 (c)).

- Score of a comb gauge: For the score of comb gauges, we first define the distance between the tooth (P_t) and the base (P_b) as the distance between the Voronoi edges E_{tb} and E_b (as shown in Figure 4 (d)), let it be d_{tb}. The Voronoi edge E_{tb} is associated with an edge of the

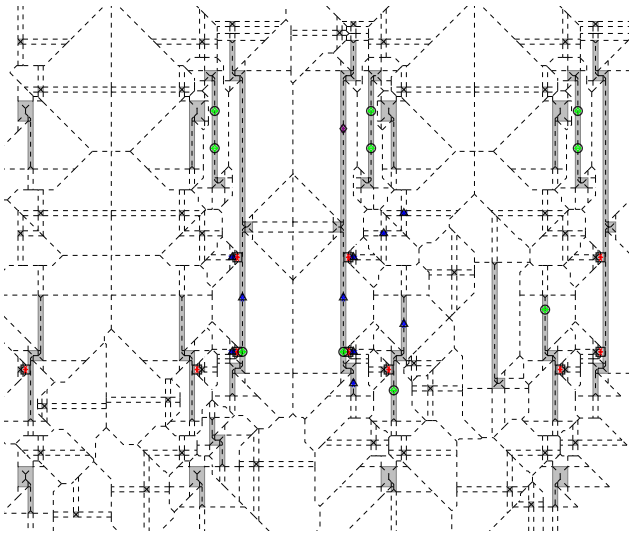

Fig 3 Showing different gauges in a portion of design layout: shapes in grey are design polygons, Voronoi diagram of design polygons is shown by dashed lines, and different gauges are color coded (blue-internal, red-external, green-sandwich).

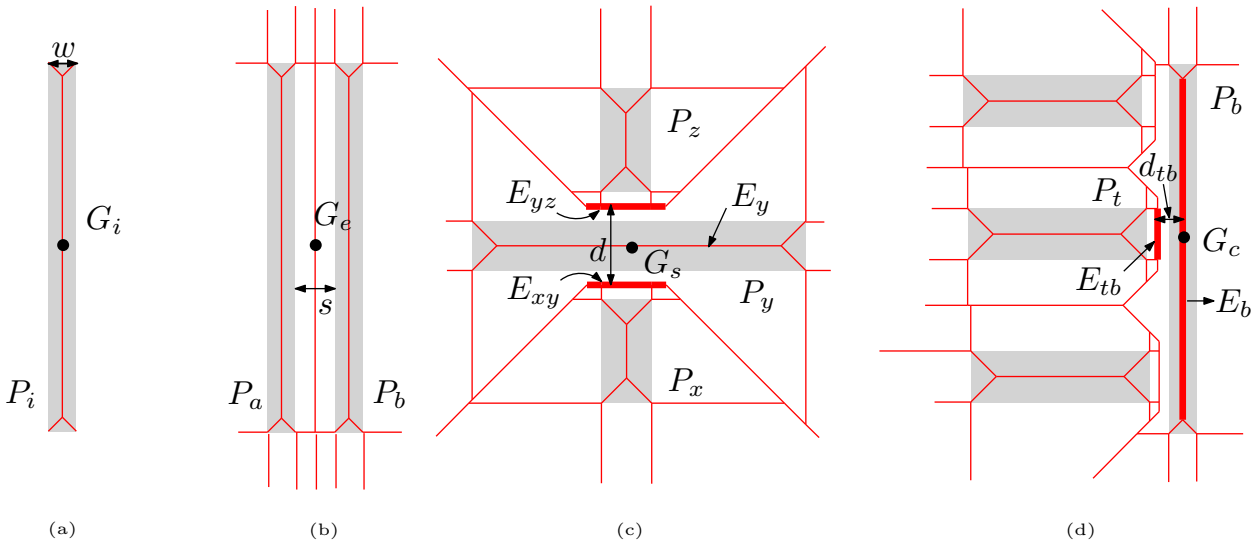

Fig 4 An example showing different gauges with their scoring formula: (a) internal gauge with score = w, (b) external gauge with score = s, (c) sandwich gauge with score = $0.5 \times d$, and (d) comb gauge with score = $0.75 \times d_{tb}$, the score of a T gauge is computed similarly with score = $0.80 \times d_{tb}$.

base and an edge of the tooth, and the Voronoi edge E_b is an edge associated with the base of the comb. The score of a comb gauge is then defined as $0.75 \times d_{tb}$. In case, we have comb gauges with same score, we break ties by the measure of overlap between E_{tb} and E_b. We give priority to the comb gauge where tooth has more overlap with the base.

- Score of a T gauge: For T gauges the score is defined as $0.80 \times d_{tb}$. The ties for T gauges are broken in the same way as for comb gauges.

The lower the score of a gauge, the higher is the probability of getting a problematic pattern around that gauge. In case gauges get same score by the our mentioned scoring method, we break the ties by extension and overlap parameters. Higher value of these parameters gives higher priority to the gauges as described in the scoring method of gauges. We also compare different type of gauges, in case, two different type of gauges have same score, we use *context* parameter to break ties. The context parameter is the measure of complexity of the pattern associated with the gauge. Let C_i, C_e, C_s, C_c, and C_t be the context parameter of internal, external, sandwich, comb and T gauges respectively. We have simply considered a hard coded order of context parameter for different gauges: $C_s > C_c > C_t > C_e > C_i$. For example, a sandwich gauge g_s and an internal gauge g_i have same score, then g_s will have more priority than g_i, because $C_s > C_i$. The scoring method along with the parameters, provide us a priority based ordered list of gauges. Figure 4 illustrates each type of gauge with its scoring formula.

4 Pattern selection based on the scoring method

We use the gauge locations as described in Section 3 to extract windows or patterns from the design layout . We feed our pattern selection tool with two inputs (see Figure: 5): (1) A design layout, (2) A set of *markers* for the design layout. A marker is a region in the design that indicates a problematic area that has high probability of faults and is generally given in form of rectangles. For a given design layout we first obtain the priority based ordered list of gauge locations, and then

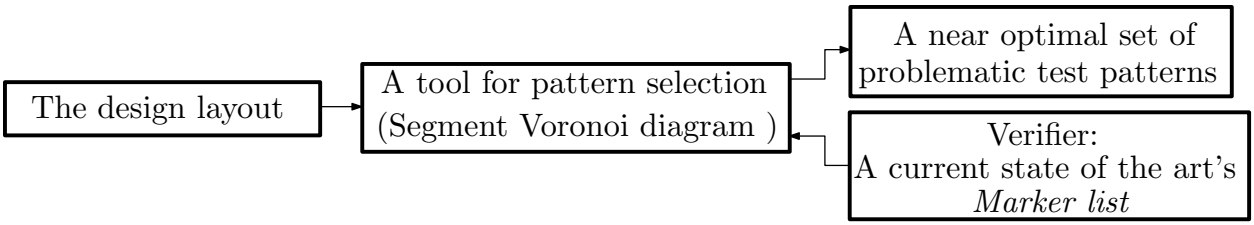

Fig 5 Block diagram of flow for the pattern selection using Segment Voronoi diagram.

we traverse the ordered list to extract patterns; one pattern per gauge. Following are the steps of our Voronoi based pattern selection tool:

1. Compute all the possible gauge locations in the given layout.
2. Sort all the gauges according to the scoring function.
3. For each gauge, consider a window of $5 - 8$ pitches with the gauge location as the center of the window.
4. For each window check if it covers any marker. Select a window only if it covers some marker, that has not already been covered by a previous window, othersiwe discard it.

As we do not desire overlapping windows, we prune the duplicate windows, which covers already covered markers. and thus we obtain a set of unique windows for a layout. The obtained set of patterns can be potentially used as a verification set for MB-OPC.

In step 4 of the pattern selection procedure, we have considered a marker to be covered in three different ways: (A) when the geometric center of the marker is strictly inside the corresponding window of a gauge, we say that the gauge covers the marker, (B) when the whole marker rectangle is completely inside the corresponding window of the gauge, and (C) when the marker rectangle overlaps with the gauge window.

Remark: Note that (B) may miss many markers because very long marker or wide marker may not fit completely inside windows that we considered.

The results of our experiment for our three different methods (A), (B), and (C) of marker covering are shown in Table 1, Table 2, and Table 3 respectively in Section 5.

To analyze the quality of our gauges we further investigated the number of windows required to cover all the markers. We implemented a simple heuristic that takes a window of different sizes and goes over the layout to cover the markers. The input to our heuristic algorithm is a set of points, and the output is a set of windows, which covers the input set of points. Let P be an input set of points to our heuristic algorithm. In the context of our experiments, the set of points can be a set of marker centers, or a set of gauge locations. The description of the heuristic algorithm follows:

1. Find the leftmost x and bottommost y coordinate in P and let this point be (l_x, b_y). Construct a window with (l_x, b_y) as its bottom left corner.
2. For each point in P, remove the point if it is inside the window.

3. If $\mid P \mid \neq 0$ (remaining list is not empty), then increment the window counter by 1 and repeat steps 1 and 2; otherwise report the number of windows required to cover all the points in the given set.

To further minimize the number of patterns, we apply this heuristic to gauge locations, and obtain a reduced set of patterns. The next section provides the experimental results.

5 Experimental results

We have done experiments for pattern selection on a portion of 22nm random logic design layout, provided with a state of the art markers for the corresponding layout. The 22nm layout had 38584 design polygons, and 7079 markers. The experiments are executed on a MacBook Pro 2.2 GHz Intel i7 with 4 GB RAM.

5.1 Marker coverage

We check the quality of the gauges by counting the number of gauges needed to cover all the markers. We considered four window sizes (in *pitches* $\times$ *pitches*), 5×5, 6×6, 7×7, and 8×8. We observed that the smaller windows were not able to cover all the markers. As we increased the window size the covering of marker increased. Our results are summarized in Tables 1 , 2, and 3, and the graph shown in figures 6, and 7.

The notations in Tables 1, 2, and 3 are as follows: W_s = window size = *pitches* $\times$ *pitches*, M_c = Number of markers covered, G_u = Number of gauges used, r_u = Normalized range of scores of useful gauges (normalized gauge score = $\frac{g_s}{H_s}$, where g_s is the score of a gauge, and H_s is the highest recorded score among all considered gauges), G_f = Number of gauges failed to detect any marker, r_f = Normalized range of scores of failed gauges, p_d = Probability of detection of markers by the provided gauge set = $\frac{M_c}{7079} \times 100$, p_m = Probability that markers will be missed by the given gauge set will be $1 - p_d$, u_g = percentage of useful gauges those detect at least one marker (this basically evaluates the scoring function) = $\frac{G_u}{G_u + G_f} \times 100$, and T = Time taken to run the experiment on a MacBook Pro 2.2 GHz Intel i7 with 4 GB RAM.

In Table 1, we show results of our experiment for method (A) of marker covering, where we vary the window size and check the covering of the markers by the windows extracted by using gauge locations. Recall that for method (A), a marker is considered covered, if the geometrical center of the marker is within some window. The gauge utilization, that is the percentage of gauges which successfully covered at least some marker, increases with the increase in the window size. The larger window has more chance to cover more marker centers, provided the location of the window is chosen logically, and in this case the windows are chosen with priority, based on the priority of the guages, which in turn depends on shapes and the proximity information of the shapes in the design. The probability of marker coverage increases with the increase in the window size. We observe 100% marker coverage for the window size of 7×7 and 8×8. The run time of the experiment decreases with the increase in the window size, and this is because the markers are exhausted quickly with the increasing window size. For all the different window sizes the run time is within 20 seconds. The variation of marker coverage with window size for this case is shown in Fig 6.

Table 2 shows the result for method (B) of marker covering, in which a marker is considered covered, if the marker area is completely within some window. The marker coverage increases

Table 1 Covering of marker centers by windows generated by gauge set

W_s (*pitches* $\times$ *pitches*)	M_c	G_u	r_u	G_f	r_f	p_d	u_g	T
5×5	6968	5708	[0.05 - 1.0]	1638	[0.05 - 0.1]	98.43%	77.70 %	18.208 sec
6×6	7070	5306	[0.05 - 0.55]	675	[0.05 - 0.1]	99.87 %	88.71%	17.666 sec
7×7	7079	5029	[0.05 - 0.2]	383	[0.05 - 0.1]	100%	92.92%	16.768 sec
8×8	7079	4767	[0.05 - 0.16]	354	[0.05 - 0.1]	100%	93.08%	16.056 sec

Table 2 Covering of marker rectangles by windows generated by gauge set

W_s (*pitches* $\times$ *pitches*)	M_c	G_u	r_u	G_f	r_f	p_d	u_g	T
5×5	6101	5202	[0.05 - 0.6]	2278	[0.05 - 0.1]	86.18%	69.54%	68.63sec
6×6	6711	5263	[0.05 - 0.55]	691	[0.05 - 0.1]	94.8%	88.39 %	61.93sec
7×7	6908	5126	[0.05 - 0.4]	334	[0.05 - 0.1]	97.58%	93.88%	59.21sec
8×8	7014	4899	[0.05 - 0.2]	370	[0.05 - 0.1]	99.08%	92.97%	57.12sec

with the increase in window size. The gauge utilization first increases with the increase in window size (from 5×5 to 7×7), and then decreases as we further increase the window size (from 7×7 to 8×8), this is because the bigger windows has potential to cover more markers but we need to check with more number of windows, as there are many windows which do not completely contain any marker. In this case also we observe that the probability of marker coverage increases with the increase in window size. We were not able to cover 100% markers, mainly due to the fact that some markers were very long or wide and were not fitting in completely within any acceptable window size. The best case was 99.08% for the window size of 8×8. For all the different window sizes the run time is within 70 seconds.

Table 3 shows the result for method (C) of marker covering, in which a marker is considered covered, if the marker area is overlapping within some window. The probability of marker coverage increases with the increase in window size. We observe 100% marker coverage for the window size of 7×7 and 8×8. For all the different window sizes the run time is within 4 minutes. The time taken in this case is more compared to the other two cases as the predicate to determine overlap between window and design shapes takes more time that the predicate that determines if a point is inside a window or a rectangle is completely inside a window.

We observe that the window size of 7×7 and 8×8 gives fairly acceptable results in terms of marker coverage in all cases. All types of gauges has the most critical gauge score of highest priority (0.05). A clear observation from the range of scores for G_u is that, with the increase of window size, there is a decrease in the requirement of low priority gauges. In all cases the range of score for G_f is [0.05 - 0.1], which suggests that the gauge (which belongs to G_f) locations are critical and there is a possibility of finding a problematic pattern around these gauges, although in this specific layout they did not cover any ORC marker. The range of scores for G_f in all cases indicates that the scoring method captures the pattern criticality and is also able to capture more patterns that may be needed to minimize the probability of missing problematic patterns.

The variation of marker coverage with window size for all cases are shown in Fig 6. The

Table 3 Covering (overlapping) of marker rectangles by windows generated by gauge set

W_s (*pitches* $\times$ *pitches*)	M_c	G_u	r_u	G_f	r_f	p_d	u_g	T
5×5	7011	5607	[0.05 - 0.6]	1627	[0.05 - 0.1]	99.03%	77.50%	3m 57s
6×6	7073	5180	[0.05 - 0.4]	341	[0.05 - 0.1]	99.91%	93.82%	3m 44s
7×7	7079	4910	[0.05 - 0.2]	319	[0.05 - 0.1]	100%	93.89%	2m 25s
8×8	7079	4686	[0.05 - 0.16]	339	[0.05 - 0.1]	100%	93.25%	1m 28s

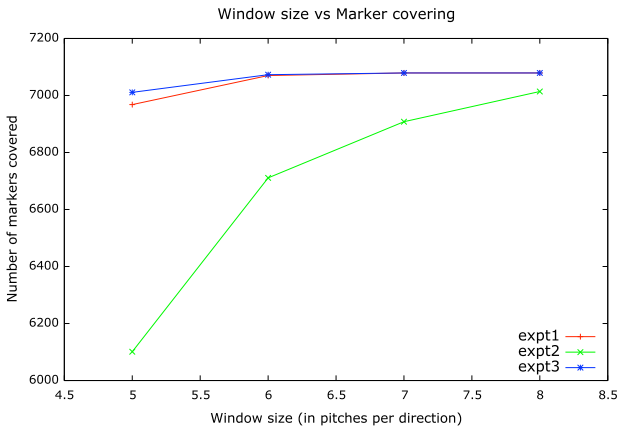

Fig 6 Variation of markers covered w.r.t the window size.

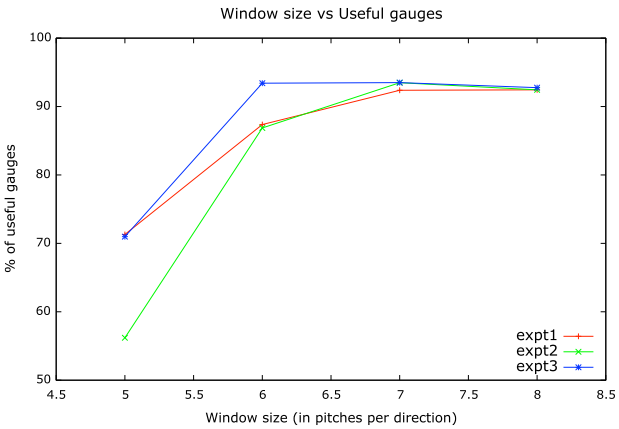

Fig 7 Variation of useful gauges w.r.t the window size.

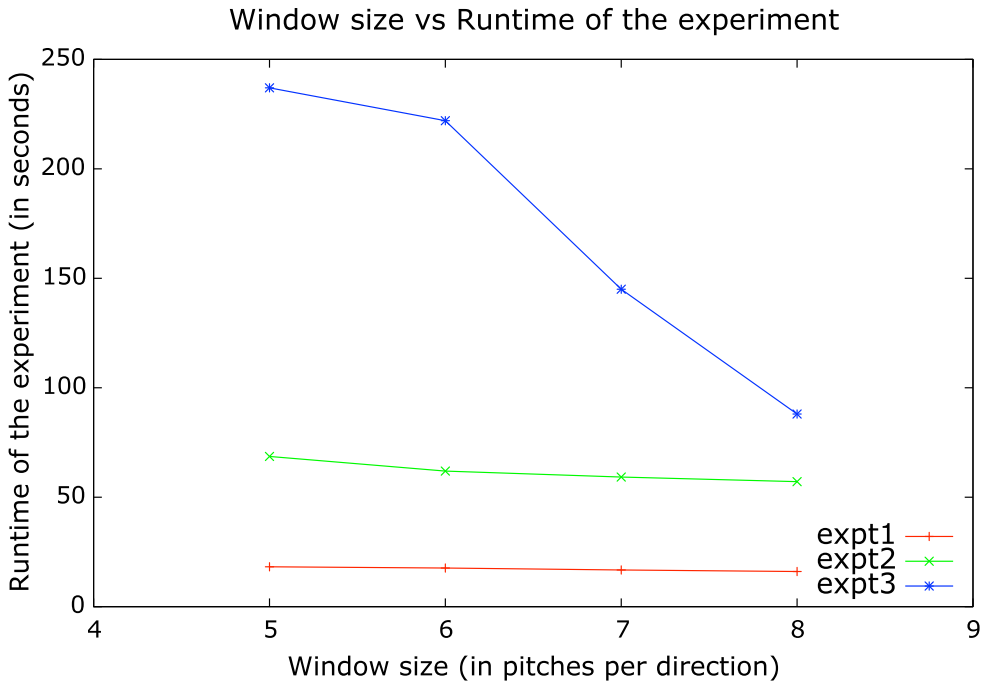

Fig 8 Variation of runtimes of the experiments w.r.t the window size.

variation of useful gauges with window size for all cases are shown in Fig 7. We have compared the runtime of the experiments based on three different ways of marker covering in the Figure 8.

5.2 Gauge distribution

We performed experiments to find out the distribution of gauges covering the markers. The notations used in the Table 4 and 5: r_i = Normalized range of scores of internal gauges, r_e = Normalized range of scores of external gauges, r_s = Normalized range of scores of sandwich gauges, r_t = Normalized range of scores of T gauges. In Table 4, we give detailed results on the marker coverage by the different types of gauges with respect to the experiment on marker coverage. We observe that all types of gauges are useful, as there are at least some gauges of each type covering

Table 4 Gauge distribution for marker covering: A, B, C are gauge distribution for three different ways of covering markers that is, marker center inside the window, markers are completely inside the window respectively, and markers are overlapping with windows.

	W_s	Intra	r_i	Extra	r_e	Sandwich	r_s	T	r_t	Total
A	5×5	30	[0.2 - 0.4]	38	[0.05 - 1.0]	5388	[0.075 - 0.3]	252	[0.16 - 0.32]	5708
	6×6	6	[0.2 - 0.4]	13	[0.05 - 0.55]	5240	[0.075 - 0.1]	47	[0.16 - 0.16]	5306
	7×7	1	[0.2 - 0.2]	7	[0.05 - 0.1]	5005	[0.075 - 0.1]	16	[0.16 - 0.16]	5029
	8×8	0	[0 - 0]	6	[0.05 - 0.05]	4753	[0.075 - 0.1]	8	[0.16 - 0.16]	4767
B	5×5	57	[0.2 - 0.4]	56	[0.05 - 0.6]	4686	[0.075 - 0.3]	403	[0.16 - 0.16]	5202
	6×6	12	[0.2 - 0.4]	18	[0.05 - 0.55]	5141	[0.075 - 0.2]	92	[0.16 - 0.16]	5263
	7×7	4	[0.2 - 0.4]	9	[0.05 - 0.4]	5078	[0.075 - 0.2]	35	[0.16 - 0.16]	5126
	8×8	1	[0.2 - 0.2]	7	[0.05 - 0.1]	4880	[0.075 - 0.1]	11	[0.16 - 0.16]	4899
C	5×5	15	[0.2 - 0.4]	29	[0.05 - 0.6]	5424	[0.075 - 0.2]	139	[0.16 - 0.32]	5607
	6×6	4	[0.2 - 0.4]	11	[0.05 - 0.4]	5135	[0.075 - 0.1]	30	[0.16 - 0.16]	5180
	7×7	1	[0.2 - 0.2]	7	[0.05 - 0.1]	4888	[0.075 - 0.1]	14	[0.16 - 0.16]	4910
	8×8	0	[0 - 0]	6	[0.05 - 0.05]	4673	[0.075 - 0.1]	7	[0.16 - 0.16]	4686

Table 5 Distribution of gauges do not covering any markers.

	W_s	Intra	r_i	Extra	r_e	Sandwich	r_s	T	r_t	Total
A	5×5	0	[0 - 0]	3	[0.05 - 0.05]	1635	[0.1 - 0.1]	0	[0 - 0]	1638
	6×6	0	[0 - 0]	3	[0.05 - 0.05]	672	[0.1 - 0.1]	0	[0 - 0]	675
	7×7	0	[0 - 0]	3	[0.05 - 0.05]	380	[0.1 - 0.1]	0	[0 - 0]	383
	8×8	0	[0 - 0]	3	[0.05 - 0.05]	351	[0.1 - 0.1]	0	[0 - 0]	354
B	5×5	0	[0 - 0]	3	[0.05 - 0.05]	2275	[0.1 - 0.1]	0	[0 - 0]	2278
	6×6	0	[0 - 0]	3	[0.05 - 0.05]	688	[0.1 - 0.1]	0	[0 - 0]	691
	7×7	0	[0 - 0]	3	[0.05 - 0.05]	331	[0.1 - 0.1]	0	[0 - 0]	334
	8×8	0	[0 - 0]	3	[0.05 - 0.05]	367	[0.1 - 0.1]	0	[0 - 0]	370
C	5×5	0	[0 - 0]	3	[0.05 - 0.05]	1624	[0.1 - 0.1]	0	[0 - 0]	1627
	6×6	0	[0 - 0]	3	[0.05 - 0.05]	338	[0.1 - 0.1]	0	[0 - 0]	341
	7×7	0	[0 - 0]	3	[0.05 - 0.05]	316	[0.1 - 0.1]	0	[0 - 0]	319
	8×8	0	[0 - 0]	3	[0.05 - 0.05]	336	[0.1 - 0.1]	0	[0 - 0]	339

some markers. For this particular design layout, the sandwich gauge turned out to cover most of the markers. In other design layouts other type of gauges may provide more useful windows to cover markers. We do not overlook any type of gauges, however small is their contribution, as our goal is to cover all the markers. In Table 5, we give the distribution of gauges which fail to cover any marker. We observe that the failed gauges are of the extra or the sandwich type. The internal and T type of gauges have no failure.

Remark: In Tables 4 and 5 we have not included any information on comb type of gauges. This is because we have observed that neither comb gauges were used to cover any marker nor any comb gauge was failed to cover any marker. The normalized gauge score for comb gauge is [0.15 - 0.15]. No comb gauge was found to fail in covering some marker, that is every comb gauge was found to cover some marker. The score range of comb gauges suggests that they should be useful, but they did not appear in the distribution of useful gauges, because they cover the markers, which were already covered by gauges with greater priority (normalized score ≤ 0.15) than them.

5.3 Reduction of patterns by a simple heuristic

We tried to further reduce the number of patterns by applying the simple heuristic algorithm of Section 4. First, we took a list of geometrical centers of all the markers as an input to the heuristic.

Table 6 Covering of near by gauges by a greedy algorithm

W_s (*pitches* $\times$ *pitches*)	number of windows before	number of windows after	Reduction in windows
5×5	5709	4102	28.14%
6×6	5306	3555	33.00%
7×7	5029	3221	35.95%
8×8	4767	3013	36.79%

The heuristic reports 3783 windows to cover all the marker centers. Then to further minimize the number of windows, we applied the heuristic algorithm that tried to put the near by gauge centers in to one window. The gauges obtained in experiment 1 using method (A) for marker covering was input to the heuristic algorithm and then we obtained the number of windows to cover those gauge centers. We observe a decrease in the required number of windows to cover the marker centers by nearly 30%. We report the results in Table 6.

6 Conclusions

We discussed a new method to select a set of problematic patterns which is based on topological information extracted from the Voronoi diagram of layout shapes. Our method is fast and automatic way to discover the potentially problematic areas when printing VLSI layout patterns. We verified our windows by covering ORC-generated markers. Using our pattern selection tool, we covered 7079 ORC markers for a design layout with 38584 design polygons by nearly 5000 extracted patterns. Applying a simple heuristic algorithm we further reduced the number of patterns by nearly 30%. The variety of gauges has a potential to extract topology and context based interesting features of patterns, which laid a foundation for our future work of obtaining calibration and verification set of patterns for MB-OPC.

References

1 G. Viehoever, B. Ward, and H. J. Stock, "Pattern selection in high-dimensional parameter spaces," in *Proc. SPIE*, **8326**, 832618 (2012).

2 D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **30**(11), 1621–1634 (2011).

3 J. R. Gao, B. Yu, and D. Z. Pan, "Accurate lithography hotspot detection based on pca-svm classifier with hierarchical data clustering," in *Proc. SPIE*, **9053**, 90530E–90530E–10 (2014).

4 H. Nosato, H. Sakanashi, E. Takahashi, M. Murakawa, T. Matsunawa, S. Maeda, S. Tanaka, and S. Mimotogi, "Hotspot prevention and detection method using an image-recognition technique based on higher-order local autocorrelation," *Journal of Micro/Nanolithography, MEMS, and MOEMS* **13**(1), 011007 (2014).

5 A. Abdo and R. Viswanathan, "The feasibility of using image parameters for test pattern selection during OPC model calibration," in *Proc. SPIE*, **7640**, 76401E (2010).

6 G. Yoo, J. Kim, T. Lee, A. Jung, H. Yang, D. Yim, S. Park, K. Maruyama, M. Yamamoto, A. Vikram, and S. Park, "OPC verification and hotspot management for yield enhancement through layout analysis," in *Proc. SPIE*, **7971**, 79710H–79710H–11 (2011).

7 J. Y. Wuu, F. G. Pikus, A. Torres, and M. Marek Sadowska, "Rapid layout pattern classification," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, *ASPDAC '11*, 781–786, IEEE Press, (Piscataway, NJ, USA) (2011).

8 M. C. Simmons, J. H. Kang, Y. Kim, J. I. Park, S. W. Paek, and K. S. Kim, "A state-of-the-art hotspot recognition system for full chip verification with lithographic simulation," in *Proc. SPIE*, **7974**, 79740M–79740M–9 (2011).

9 S. Shim, W. Chung, and Y. Shin, "Synthesis of lithography test patterns through topology-oriented pattern extraction and classification," in *Proc. SPIE, Design-Process-Technology Co-optimization for Manufacturability VIII*, **9053**, 905305–905305–10 (2014).

10 Y. Sun, Y. M. Foong, Y. Wang, J. Cheng, D. Zhang, S. Gao, N. Chen, B. I. Choi, A. J. Bruguier, M. Feng, J. Qiu, S. Hunsche, L. Liu, and W. Shao, "Optimizing OPC data sampling based on orthogonal vector space," in *Proc. SPIE*, **7973**, 79732K (2011).

11 N. Casati, M. Gabrani, R. Viswanathan, Z. Bayraktar, O. Jaiswal, D. L. DeMaris, A. Y. Abdo, J. Oberschmidt, and R. A. Krause, "Automated sample plan selection for OPC modeling," in *Optical Microlithography XXVII, Proceedings of SPIE*, **9052** (2014).

12 R. Viswanathan, O. Jaiswal, M. Gabrani, N. Casati, A. Y. Abdo, J. Oberschmidt, and J. Watts, "Experiments using automated sample plan selection for OPC modeling," in *Optical Microlithography XXVIII, Proceedings of SPIE*, **9426** (2015).

13 M. Mukherjee, Z. Baum, J. Nickel, and T. G. Dunham, "Optical rule checking for proximity-corrected mask shapes," in *Proc. SPIE, Optical Microlithography XVI*, **5040**, 420–430 (2003).

14 F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)* **23**(3), 345–405 (1991).

15 F. Aurenhammer, R. Klein, and D. T. Lee, *Voronoi Diagrams and Delaunay Triangulations*, World Scientific Publishing Company, Singapore (2013).

16 MATLAB, *version 8.4 (R2014b)*, The MathWorks Inc., Natick, Massachusetts (2014).

17 The CGAL Project, *CGAL User and Reference Manual*, CGAL Editorial Board, 4.5 ed. (2014). Available at: http://doc.cgal.org/4.5/Manual/packages.html.

18 M. Karavelas, *2D segment Delaunay graphs*, CGAL Editorial Board, 4.5 ed. (2014). Available at: http://doc.cgal.org/latest/Segment_Delaunay_graph_2/index.html.

19 "Voronoi CAA: Voronoi Critical Area Analysis." IBM CAD Tool, Department of Electronic Design Automation, IBM Microelectronics Division, Burlington, VT. Initial patents: US6178539, US6317859.

20 E. Papadopoulou and D. T. Lee, "The L_∞ Voronoi diagram of segments and VLSI applications," *International Journal of Computational Geometry and Application* **11**(5), 502–528 (2001).

21 E. Papadopoulou, "Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **30**(5), 704–716 (2011).

22 P. Cheilaris, S. K. Dey, M. Gabrani, and E. Papadopoulou, "Implementing the L_∞ segment Voronoi diagram in CGAL and applying in VLSI pattern analysis," in *Proc. 4th International Congress on Mathematical software (ICMS'14)*, *Lecture Notes in Computer Science* **8592**, 198–205, Springer (2014).

23 P. Cheilaris, S. K. Dey, and E. Papadopoulou, *L_∞ Segment Delaunay graphs*, Code documentation is available at: http://compgeom.inf.usi.ch/cgal_doc/Segment_Delaunay_graph_Linf_2/ (2013). Code can be available on request.