Voronoi Diagrams
Robust and Efficient implementation
Master's Thesis Defense

Nirav Patel

Binghamton University
Department of Computer Science
Thomas J. Watson School of Engineering and Applied Science

May 5th, 2005

Motivation

"The Universal Spatial Data Structure" –Franz Aurenhammer

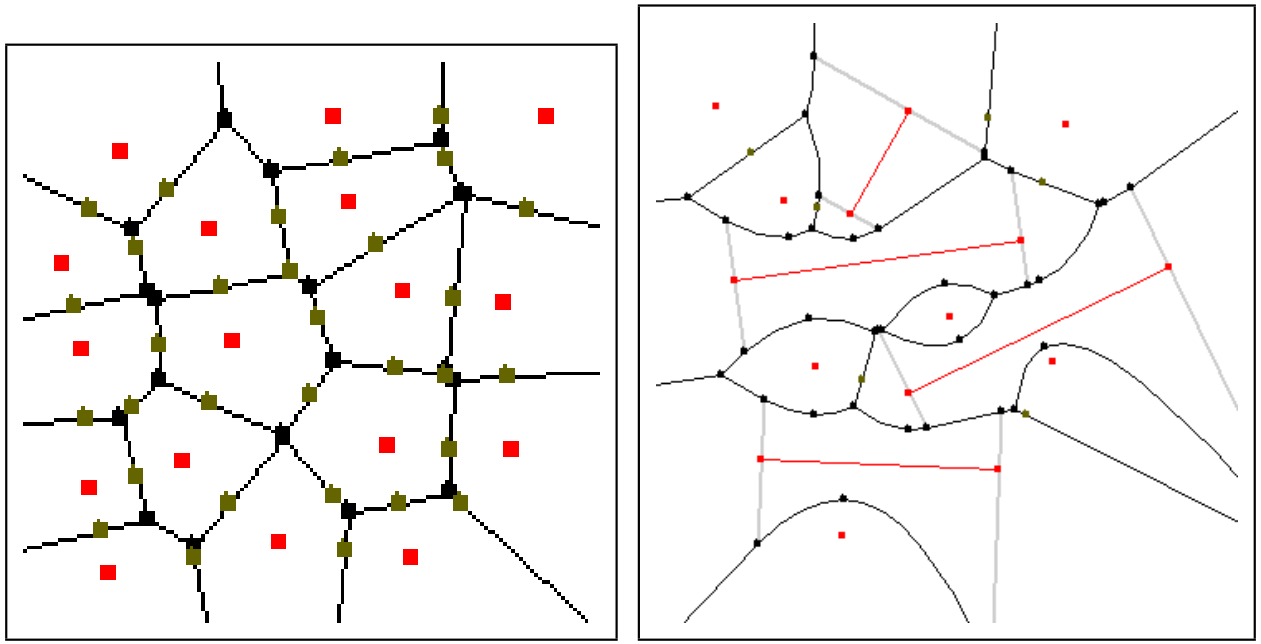
Primary motivation: To obtain skeletal representations for two dimensional shapes to be converted to embroidery data by an automated design system. Current methods employed by that system produced skeletons somewhat unreliably due to their inability to handle degenerate cases where intricate detail was present in the input shape.

Applications of Voronoi diagram

The Voronoi diagram is one of the most fundamental and versatile data structures in computational geometry. It's many applications include:

- Collision detection
- Pattern recognition
- Geographical optimization
- Geometric clustering
- Closest pairs algorithms
- k-nearest-neighbor queries

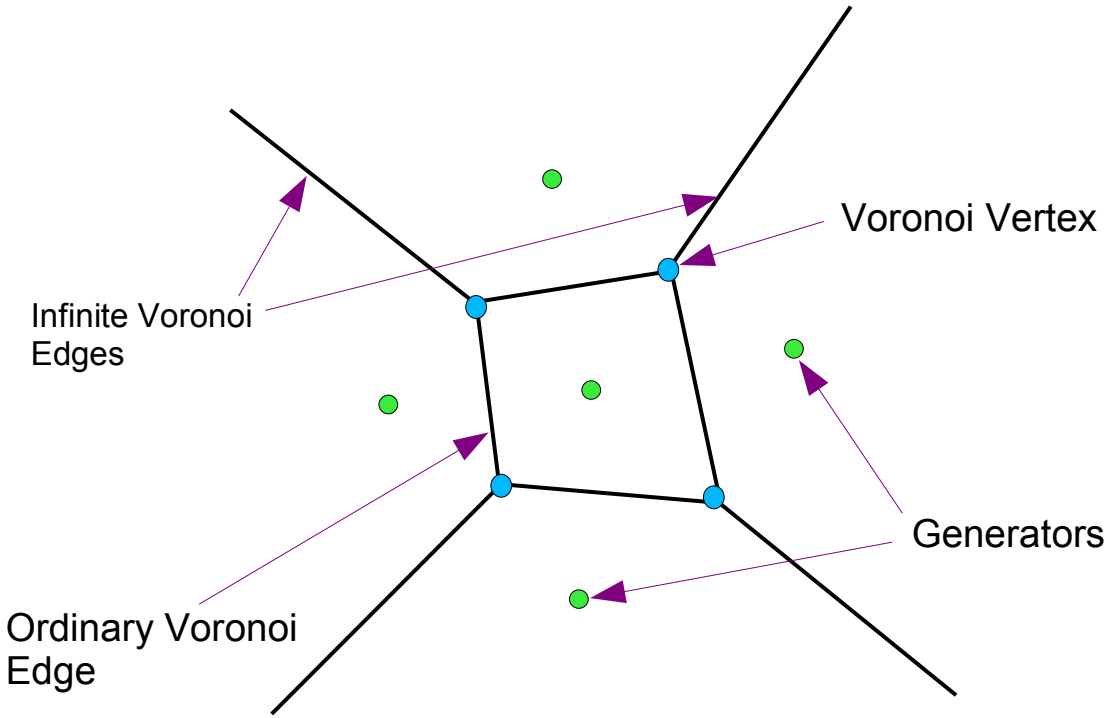
Definition of Voronoi Diagram



Definition

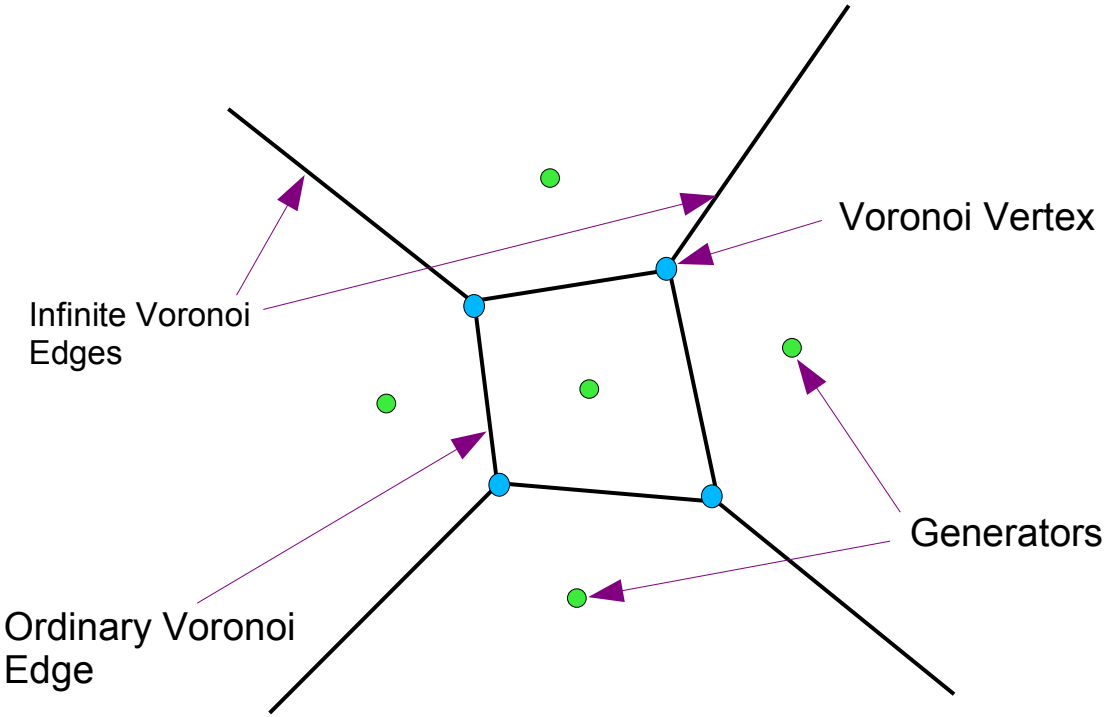
For a set P of points $p_1, p_2, \ldots, p_n$ in the Euclidean plane, the partition $Vor(P)$ of the plane into regions $R(p_1), R(p_2), \ldots, R(p_n)$ associated with each member of P, such that each point in a region $R(p_i), 1 \leq i \leq n$ is closer to p_i than any other point in P.

Structure of a Voronoi diagram



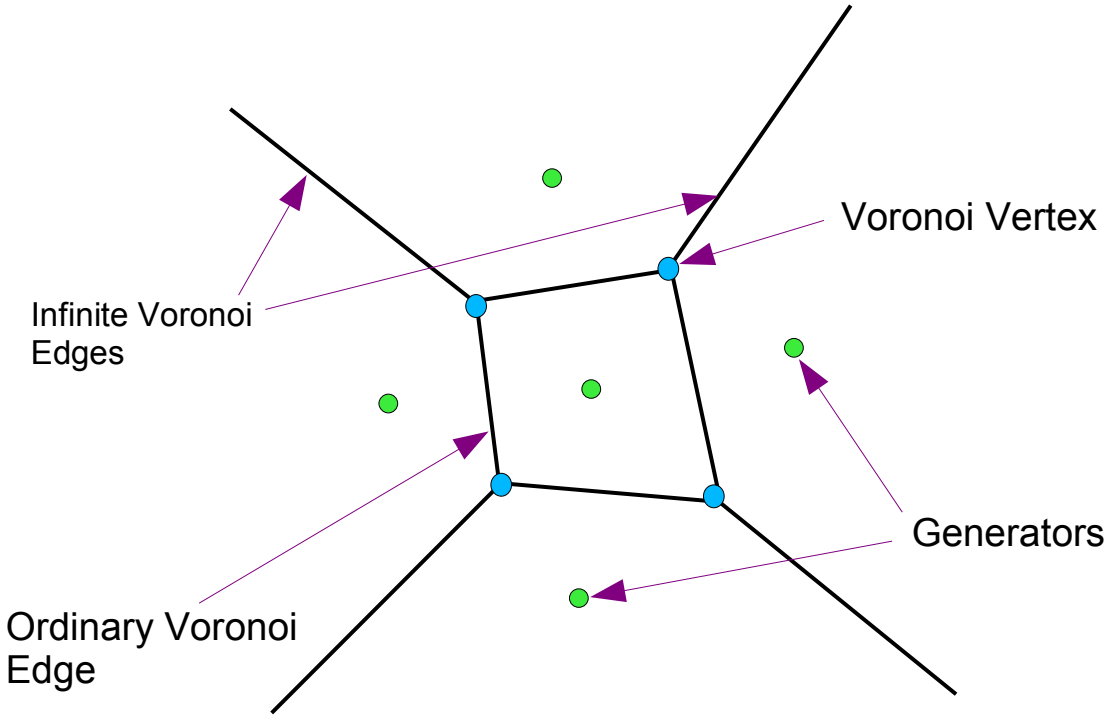
A region that corresponds to a generator p_i is called it's Voronoi region and is denoted as $R(p_i)$.

Structure of a Voronoi diagram



A common boundary of two Voronoi regions is called a Voronoi edge.

Structure of a Voronoi diagram



A point at which the boundaries of three or more Voronoi regions meet is called a Voronoi vertex.

Properties of Voronoi diagram

1. A Voronoi edge between two Voronoi regions $R(p_i)$ and $R(p_j)$ is a portion of the perpendicular bisector of the line segment connecting the two generators p_i and p_j.

Properties of Voronoi diagram

1. A Voronoi edge between two Voronoi regions $R(p_i)$ and $R(p_j)$ is a portion of the perpendicular bisector of the line segment connecting the two generators p_i and p_j.
2. A Voronoi vertex is the center of the circle that passes through the three generators whose regions are incident to the vertex, i.e., it is the circumcenter of the triangle with those generators as the vertices.

Properties of Voronoi diagram

1. A Voronoi edge between two Voronoi regions $R(p_i)$ and $R(p_j)$ is a portion of the perpendicular bisector of the line segment connecting the two generators p_i and p_j.
2. A Voronoi vertex is the center of the circle that passes through the three generators whose regions are incident to the vertex, i.e., it is the circumcenter of the triangle with those generators as the vertices.
3. A Voronoi region $R(p_i)$ is a convex (possibly unbounded) polygon containing the corresponding generator p_i.

Correlation between Voronoi diagram and Primary cycle structure

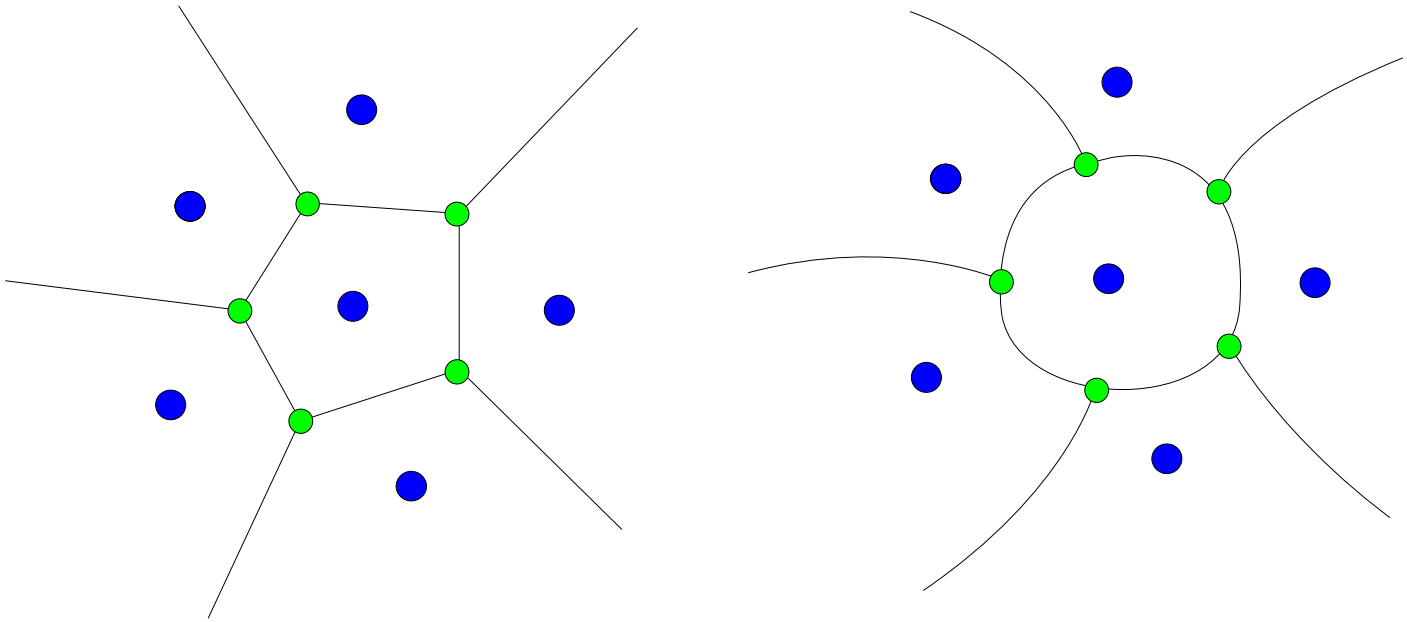


Figure: (a) Voronoi Diagram (V, E) (b) Corresponding primary cycle structure $G = (V, E, C)$

Various algorithms for Voronoi computation

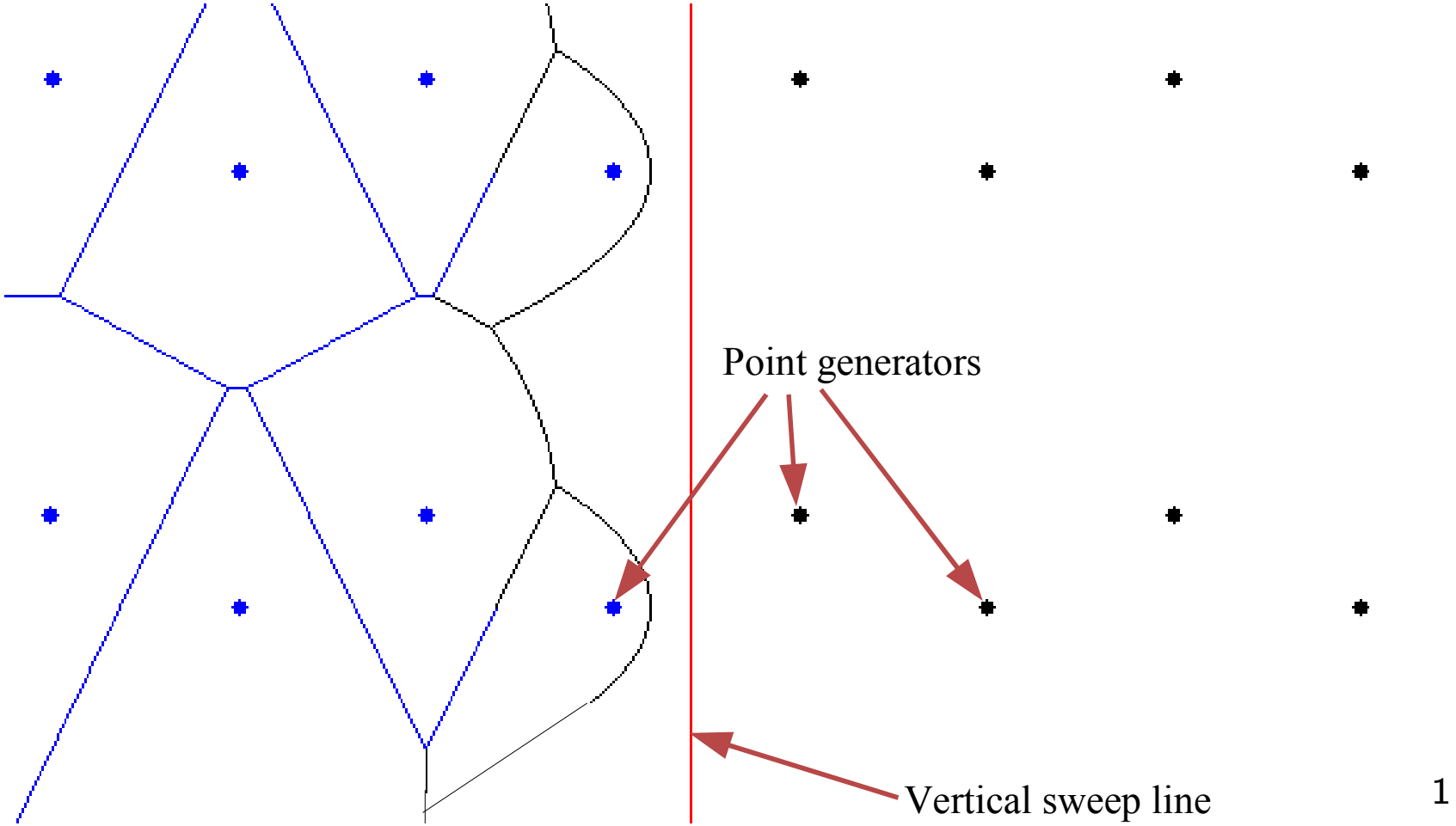
"It is notoriously difficult to obtain a practical implementation of an abstractly described geometric algorithm" –Steven Fortune

- Sweep line
- Divide-and-conquer
- Spiral-search
- Three dimensional convex hull
- Incremental

Sweep line algorithms

- Proposed by Steven Fortune.
- A vertical line (also called a sweep line) is swept across the plane, from left to right.
- The Voronoi diagram is incrementally constructed as point generators are encountered by the sweep line.

Snapshot of execution of Sweepline algorithm



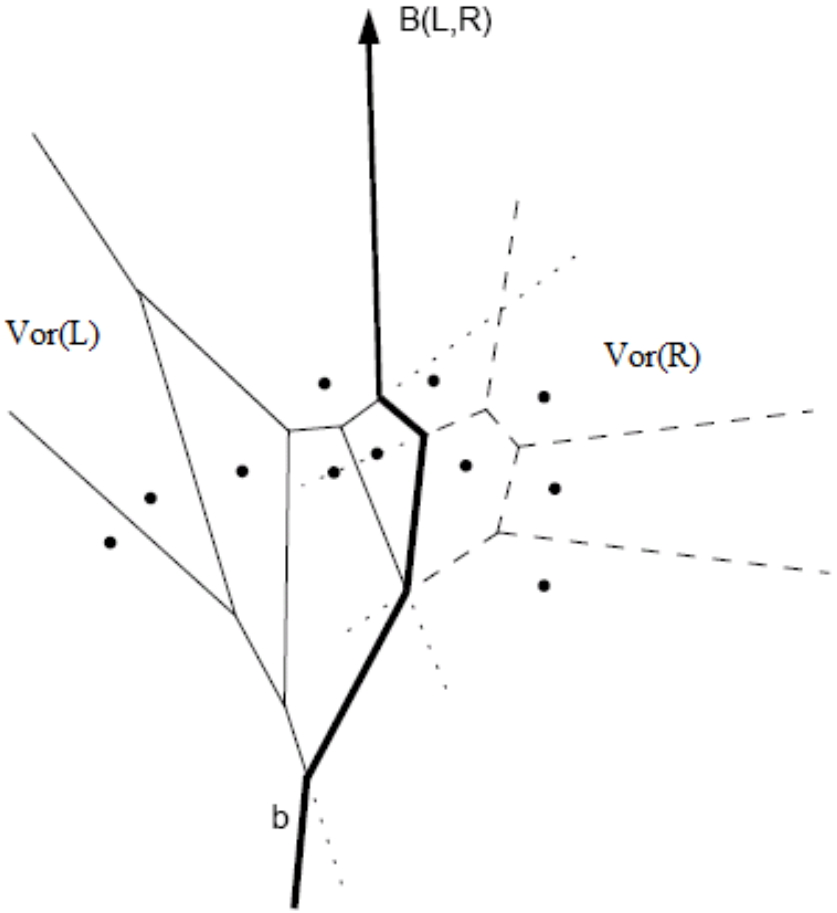
[1]

[1] Source:http://www.diku.dk/hjemmesider/studerende/duff/Fortune/

Properties of sweep line algorithms

- Elegant solution as in easy to understand and easy to implement
- Shortcomings when dealing with degenerate cases like more than three co-circular point generators
- Relies heavily on the accuracy of numerical calculations, which limits its use to theoretical purposes

Divide and conquer algorithms

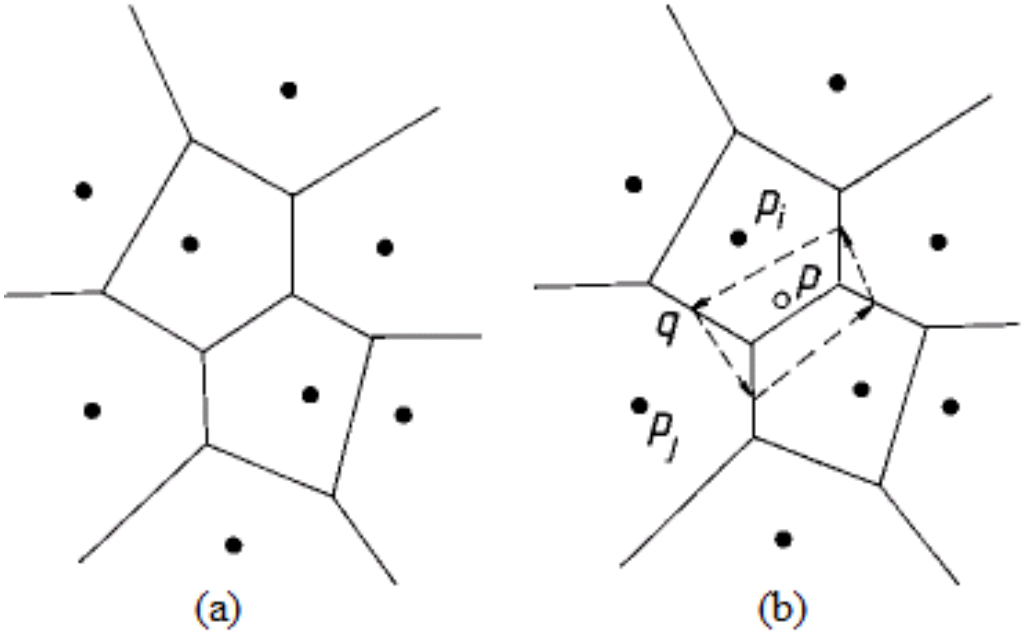


1. The set of point generators, P, is split by a dividing line into subsets L and R of about the same size.
2. The Voronoi diagrams $Vor\,(L)$

Properties of divide-and-conquer algorithms

- Implementation details are somewhat complicated
- Numerical errors are likely by construction
- The average and worst case time complexity is $\Omega(n \log n)$ and it is possible to achieve better performance using other methods.

Incremental algorithms



Obtain $Vor(P)$ from $Vor(P - \{q\})$ by inserting the site q.

Properties of incremental algorithms

- As the region of q can have up to $n-1$ edges, for $n = |P|$, this leads to a runtime of $O\left(n^2\right)$. Several authors have further tuned the technique of inserting Voronoi regions, producing efficient and numerically robust algorithms that have average time complexity of $O\left(n\right)$
- The implementation of incremental algorithms is simple compared to other techniques.

Summary of approaches to deal with numerical inaccuracy

Reliance on numerical values

Heavy

Exact Arithmetic

The topological structure is decided based on the signs of results of numerical computations. Two strategies:

1. Implementing the algorithms on top of a model of exact arithmetic capable of infinite precision computation.
2. Restricting the precision of input data (e.g., using only 24 bit integers) such that other techniques may be used that guarantee correctness of specific mathematical operations on that data.

Summary of approaches to deal with numerical inaccuracy

Reliance on numerical values

Medium

Tolerance based

Every numerical computation is accompanied by calculation of an upper bound on its error. Based on this, the result is judged to be either reliable or unreliable.

- Program code is comparatively complex because there must be two branches for processing after each calculation, one each for the reliable and unreliable case
- Software is less portable because the errors are often tied to a particular computation environment

Summary of approaches to deal with numerical inaccuracy

Reliance on numerical values

Small

Topology Oriented

Does not rely directly on numerical computations because of the assumption that there is error associated with all numeric computations.

- Derived topological properties are given higher priority and only the numerical computations consistent with that derived topology are accepted
- This approach can guarantee correctness at a relatively low cost since models of exact computation or restrictions on input are not required

Overall approach to ensure robustness

1. Reduce the algorithm into simple routines with precisely defined topological requirements.

Overall approach to ensure robustness

1. Reduce the algorithm into simple routines with precisely defined topological requirements.
2. Ensure every routine achieves its goals and leaves the Voronoi data structures in a topologically consistent state.

Overall approach to ensure robustness

1. Reduce the algorithm into simple routines with precisely defined topological requirements.
2. Ensure every routine achieves its goals and leaves the Voronoi data structures in a topologically consistent state.
3. Build upon the reliability of smaller routines to insert each generator while maintaining the topological consistency and correctness of the resulting Voronoi diagram.

Steps for robustness of math routines

1. Avoid using thresholds
2. Use all topological information available
3. Back-up routines for special case handling
4. For border-line cases use two different methods and choose a more reliable value based on topology

Example math routine

Computing point on a bisector for two line segments

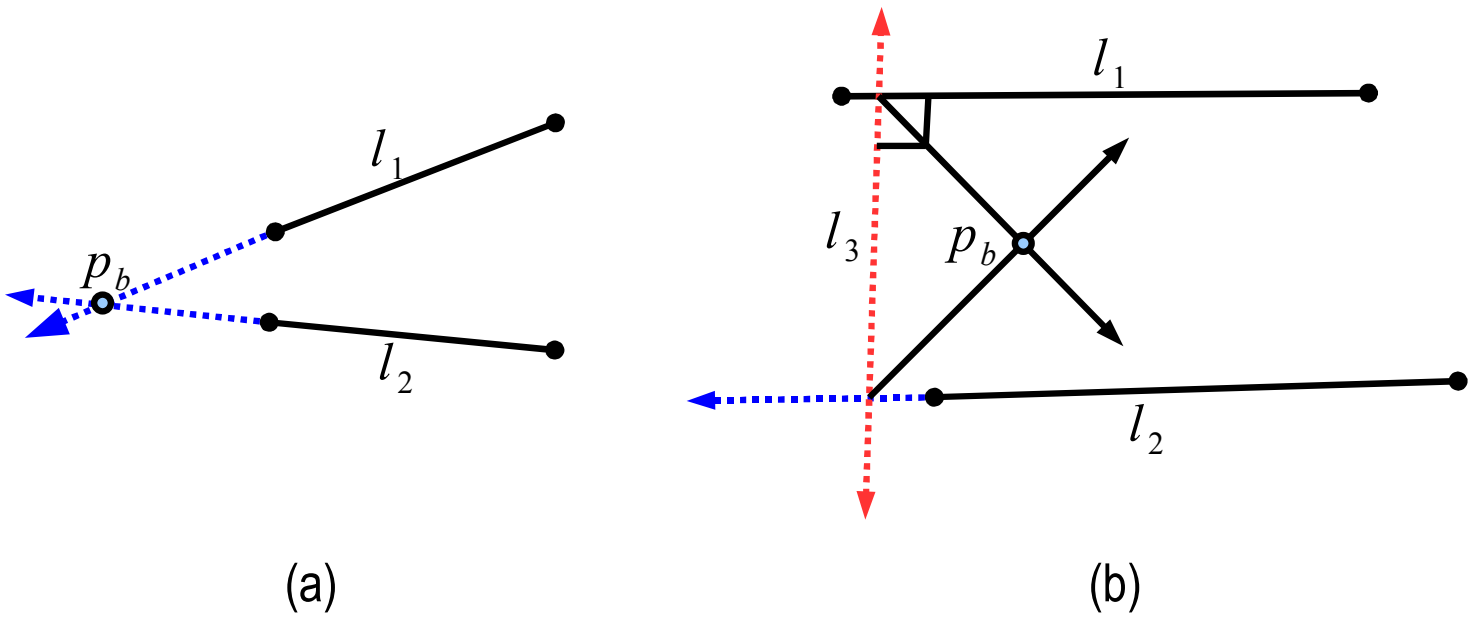
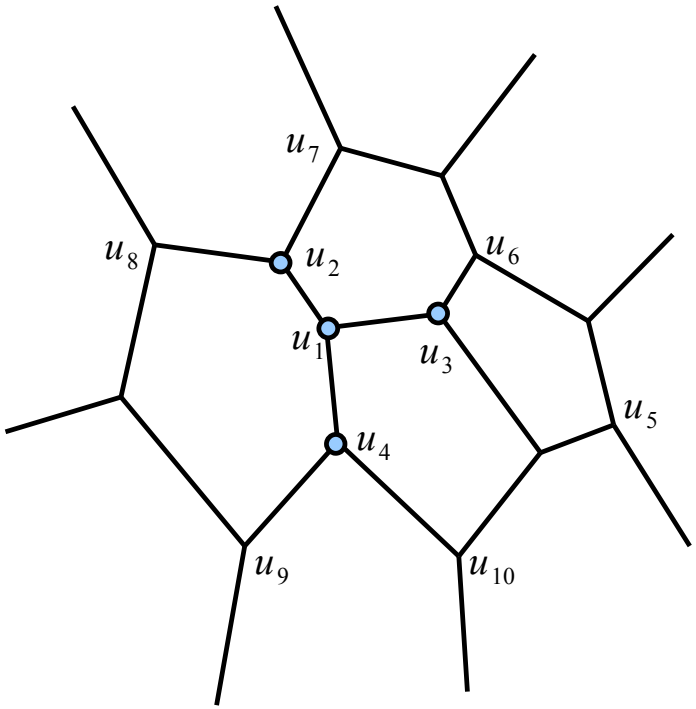


Figure: Computing bisector point p_b of lines l_1 and l_2 (a) normal case (b) nearly parallel lines, l_3 is the generated line

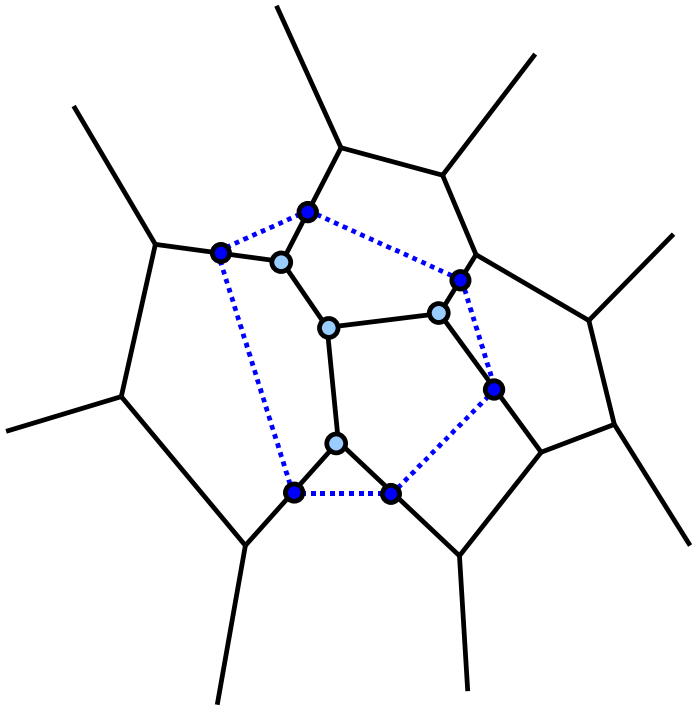
Numerical and Topological Process



(a)

1. u_1, u_2, u_3, u_4 are the vertices to be removed

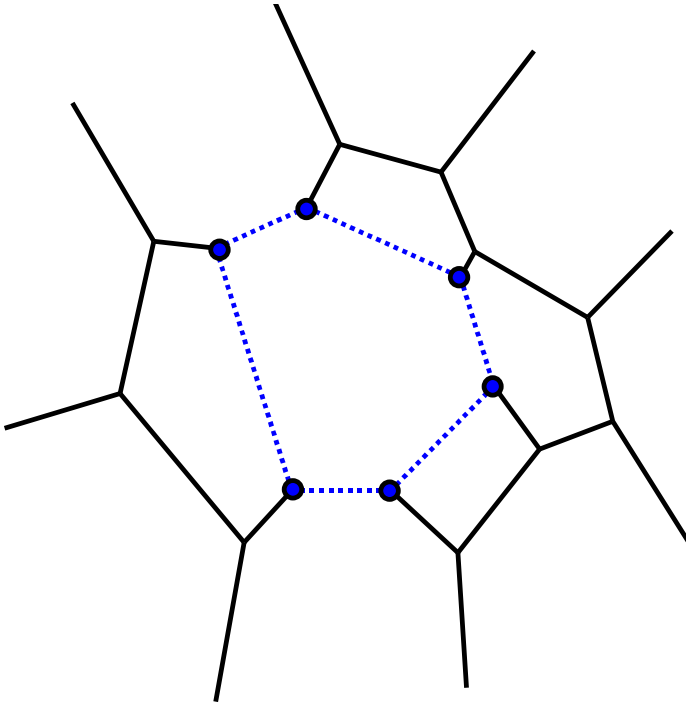
Numerical and Topological Process



(b)

2. New vertices and new edges are generated

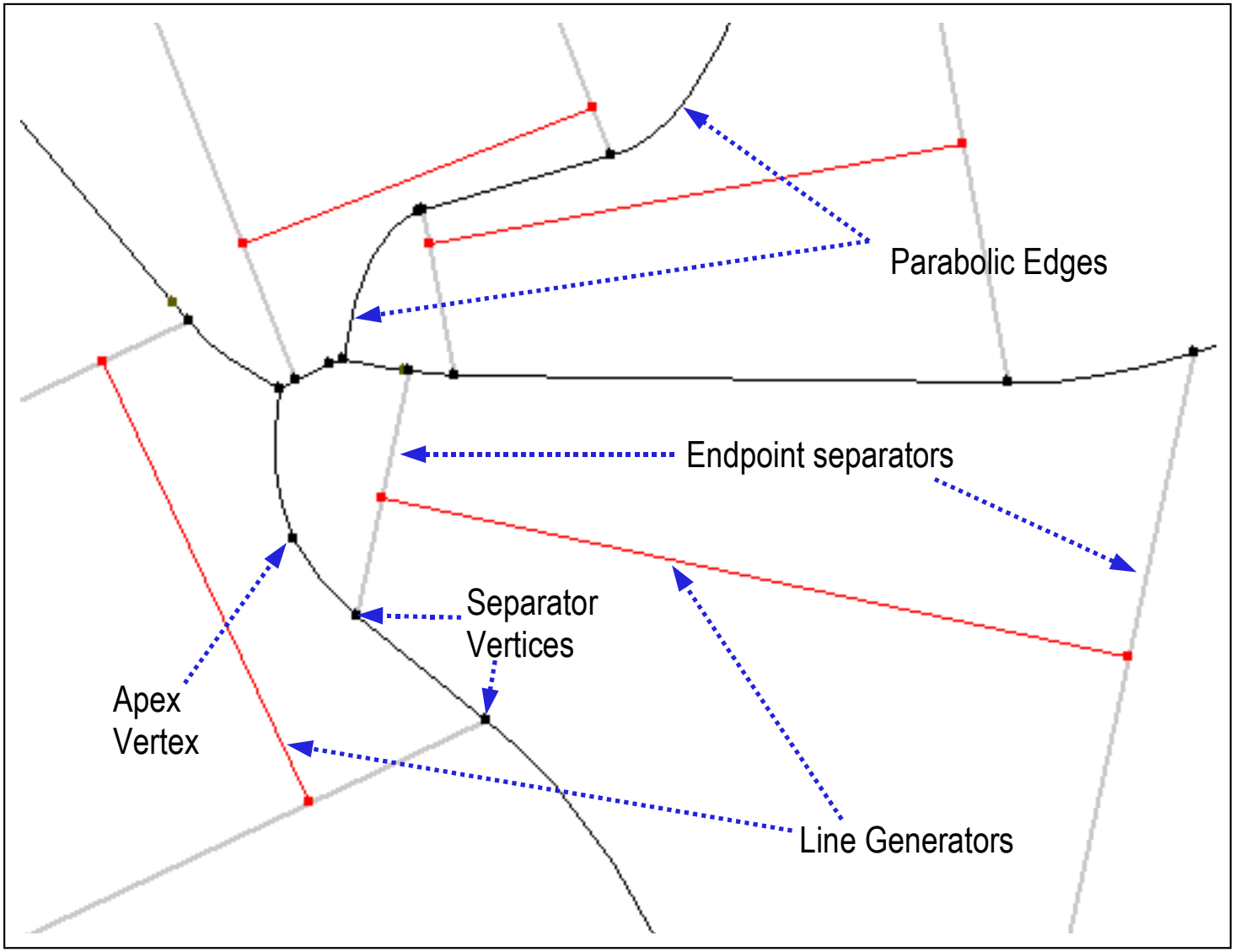
Numerical and Topological Process



(c)

3. New cycle is formed

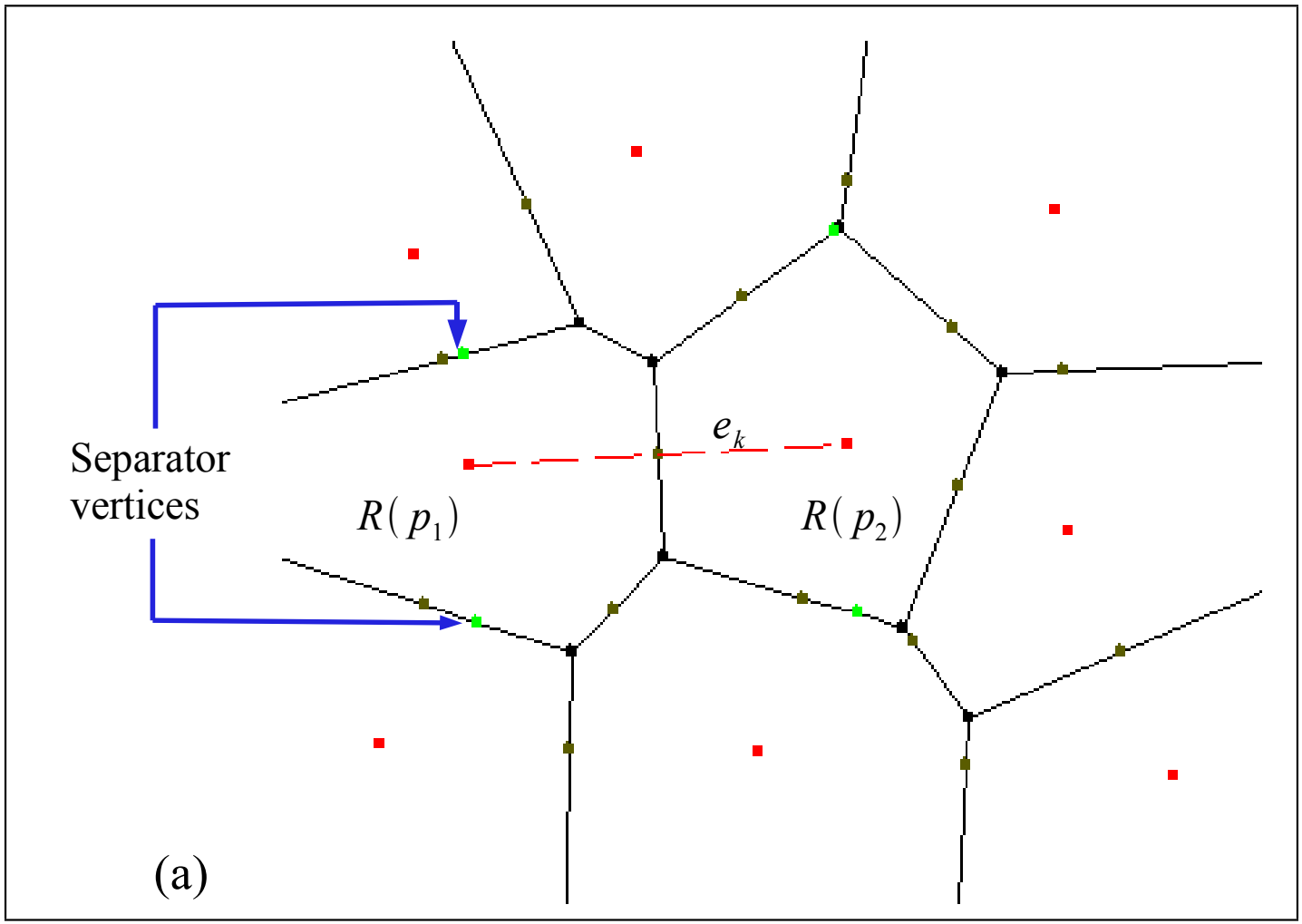
Types of Vertices and Edges



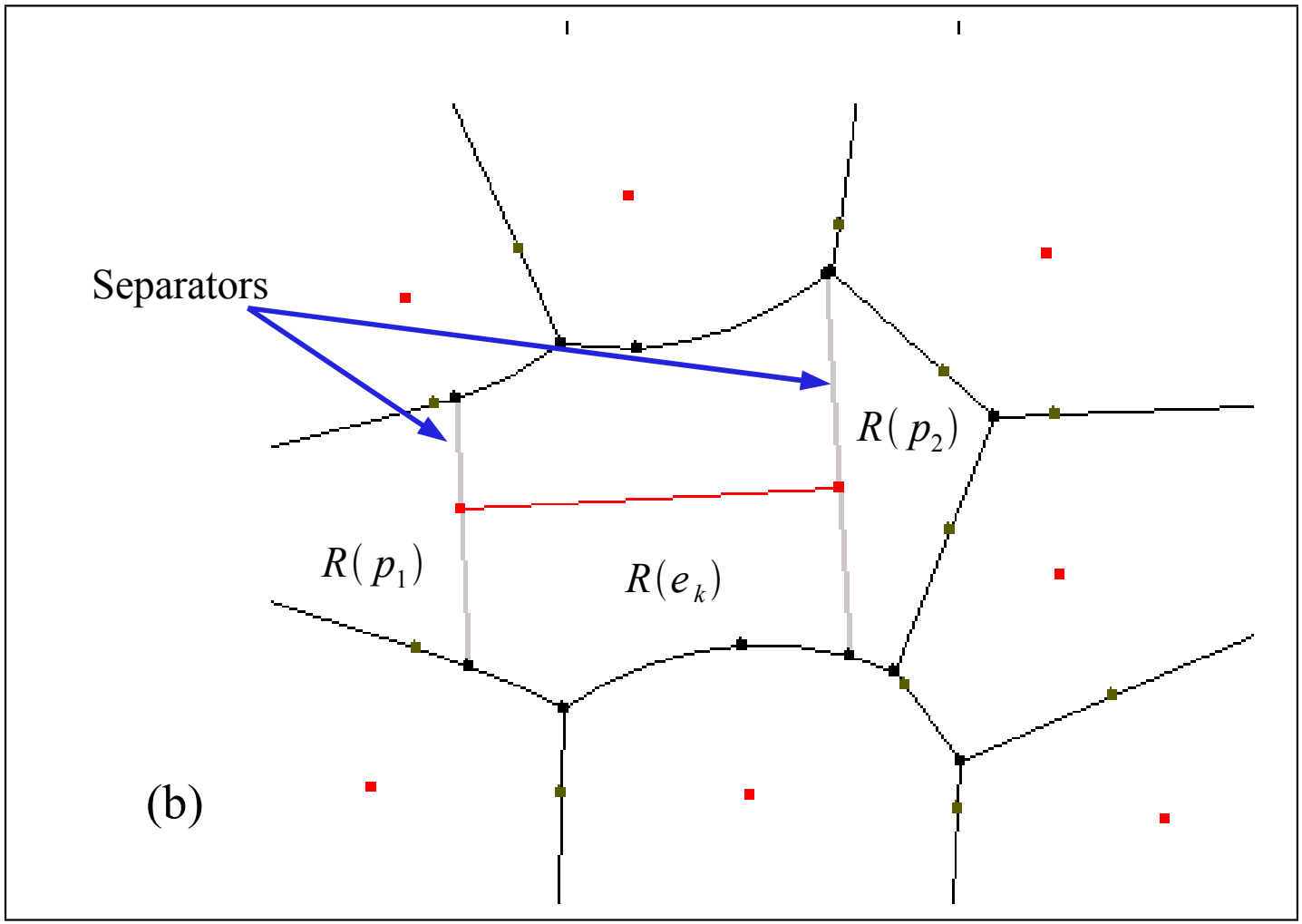
Making End point regions

When computing a Voronoi diagram the endpoints of line segments are considered to be separate generators which are already inserted into a previously created Voronoi diagram (i.e., the first step of this approach).
Topological requirement: To generate two new vertices on the primary cycle of each endpoint corresponding to the open segment.

Making End point regions



Making End point regions

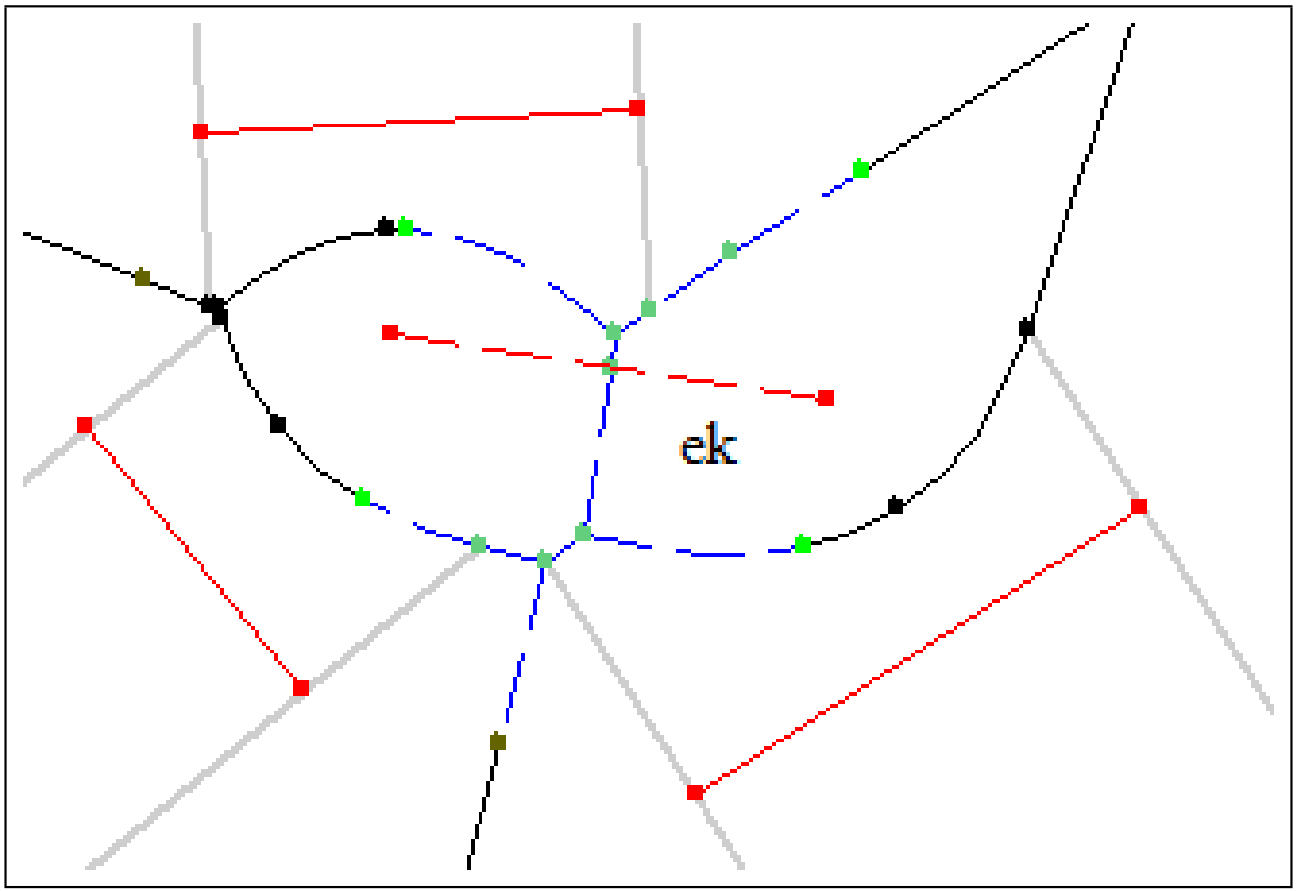


Marking vertices to be removed

Topological properties applicable to this routine:

- The vertices and edges to be removed form a tree structure.
- The tree structure has a unique path between the regions of the endpoints p_i and p_j.
- The path goes between the two groups of regions; one is to the right of the open segment e_k and the other is to the left.
- Any separator is not completely removed.

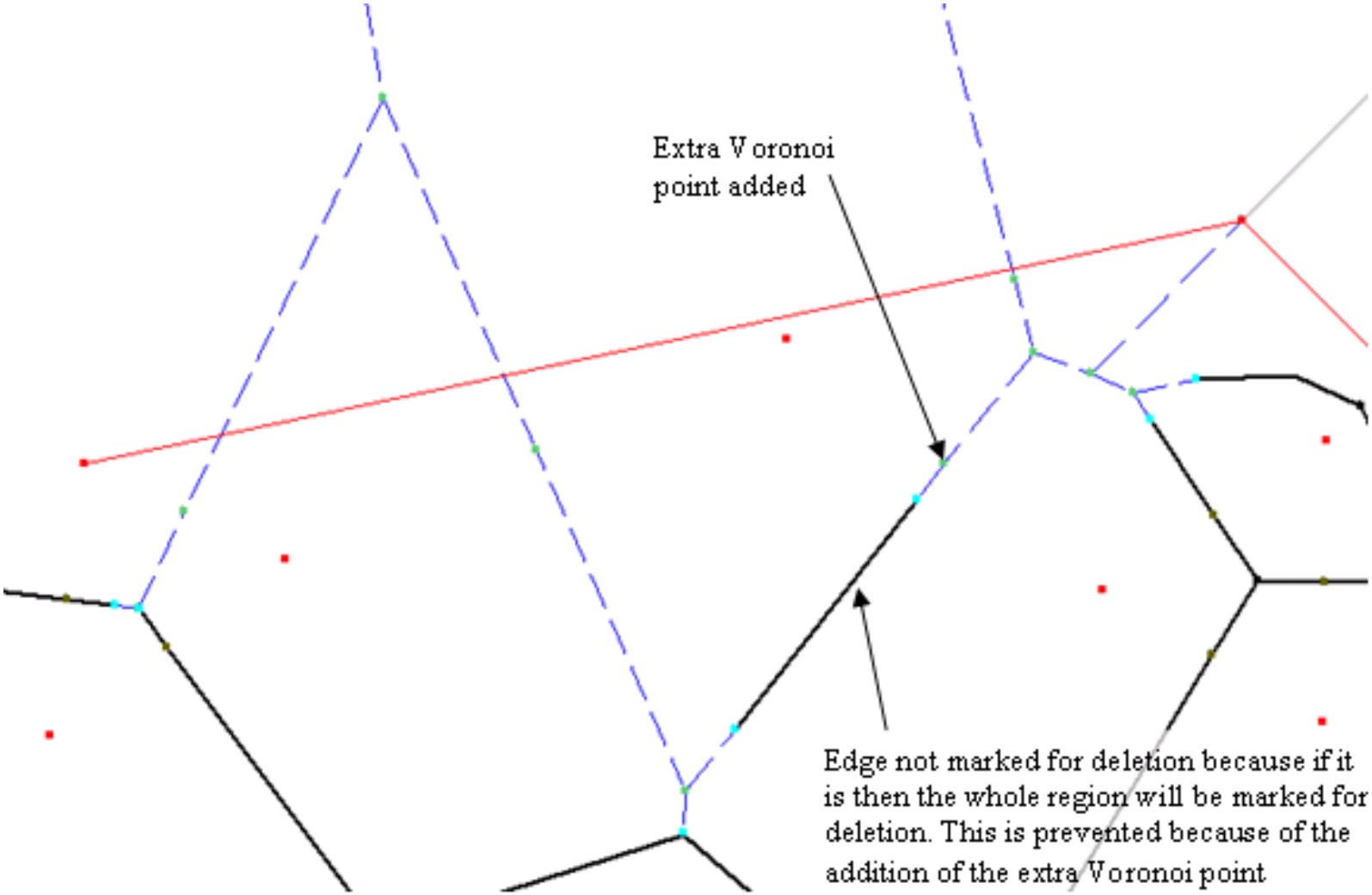
Marking vertices to be removed



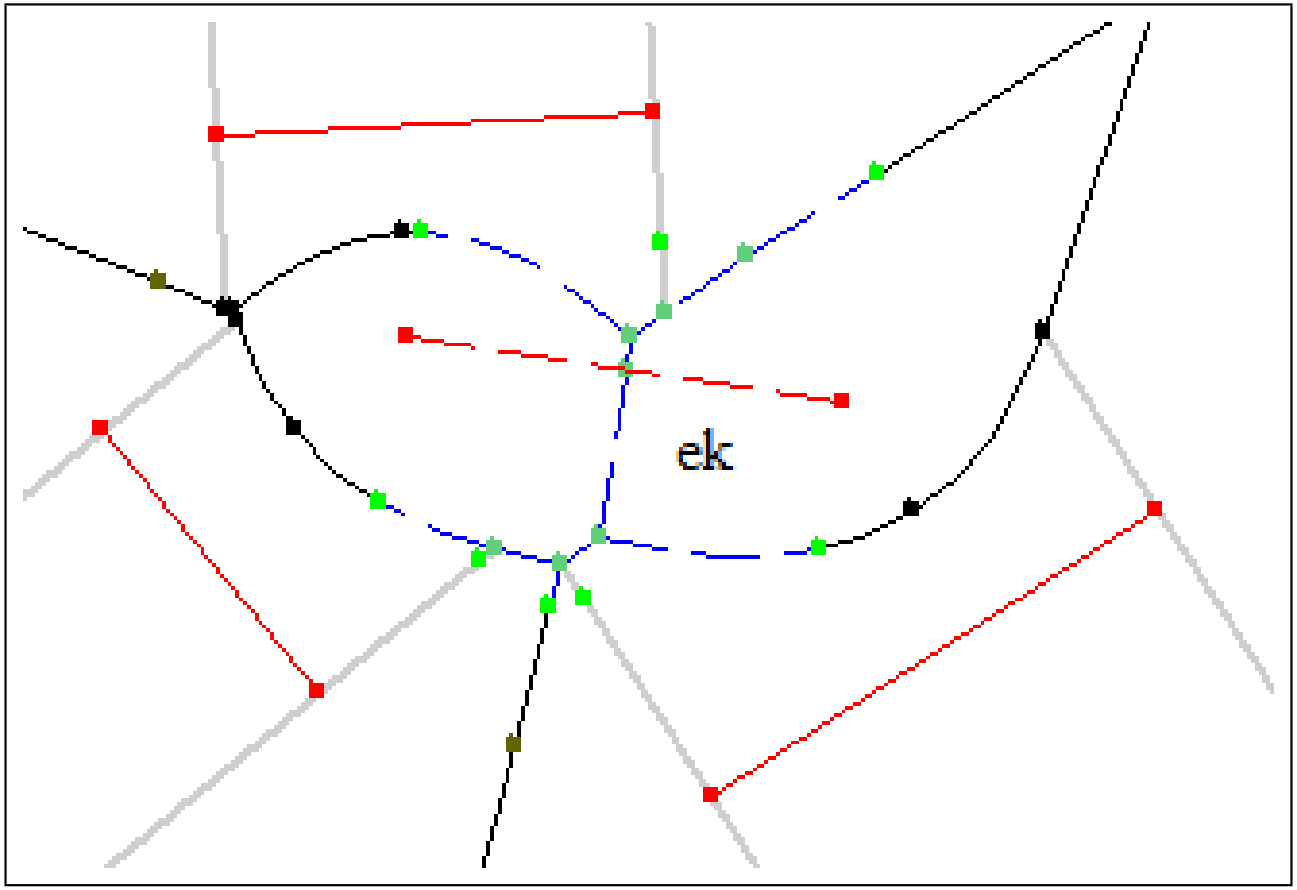
Removal of a whole region

- When marking vertices and edges for removal in some situations it is possible that all vertices corresponding to a particular cycle are marked for deletion.
- Extra Voronoi vertices are inserted into the diagram corresponding to the intersection of the line segment joining the Voronoi point generators and the bisector of that segment.

Removal of a whole region

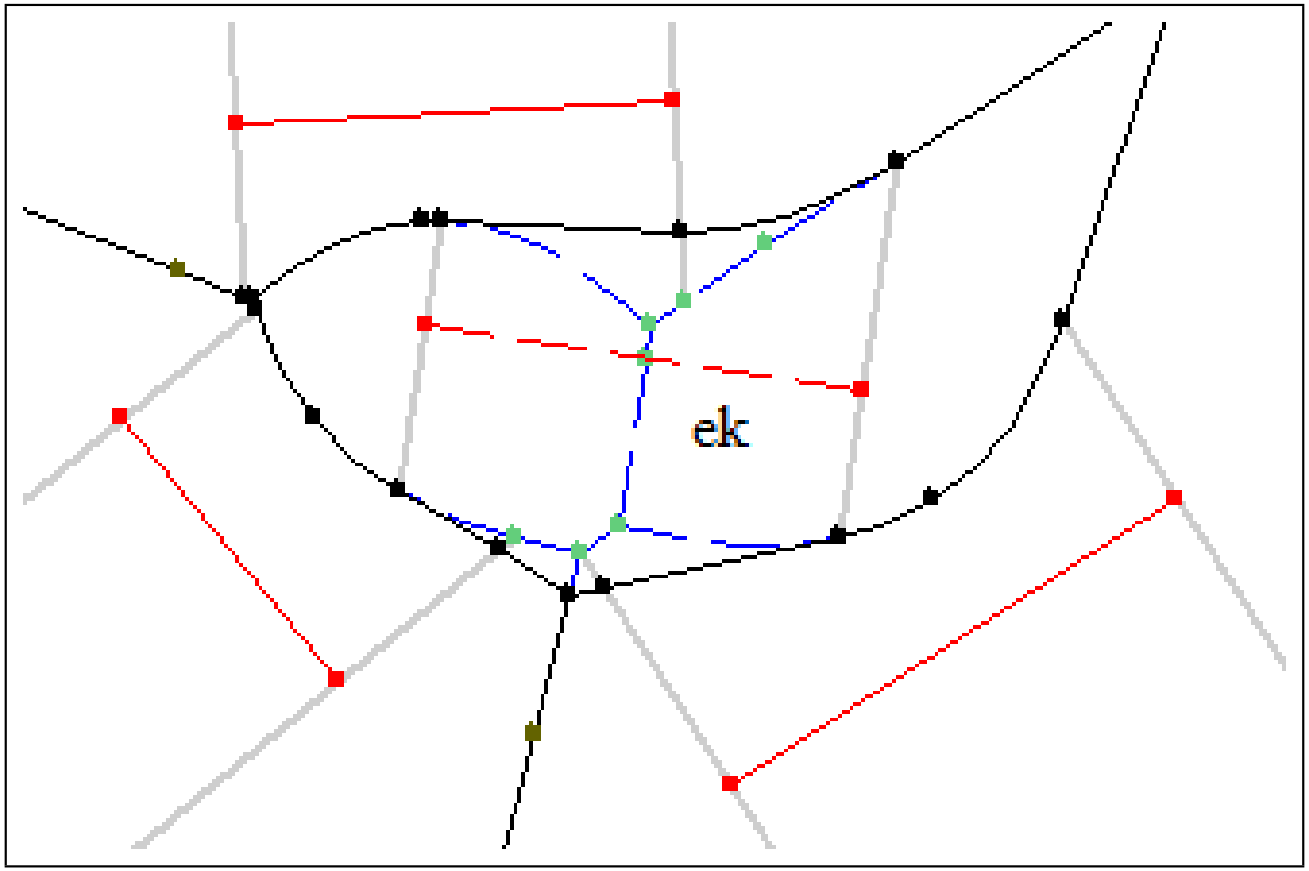


Generating new vertices



Generating new Voronoi vertices on edges connected to those marked for deletion while maintaining the topological consistency of regions surrounding the tree marked for deletion.

Forming the new cycle for generator



The primary cycle for the new generator e_k is created.

Finding the nearest generator

A method using grid data structures was implemented to find the nearest generator more efficiently. The generators already inserted within the Voronoi diagram are placed into in a regular grid (i.e., each cell is of equal size). When a new generator is to be inserted its corresponding cell in the grid is located. The generators in this cell and its surrounding cells are searched to determine the nearest generator. The grid resolution depends on the width w and height h of the bounding box of generators. There are $w\sqrt{n} * h\sqrt{n}$ cells in the grid.

Reordering of generators

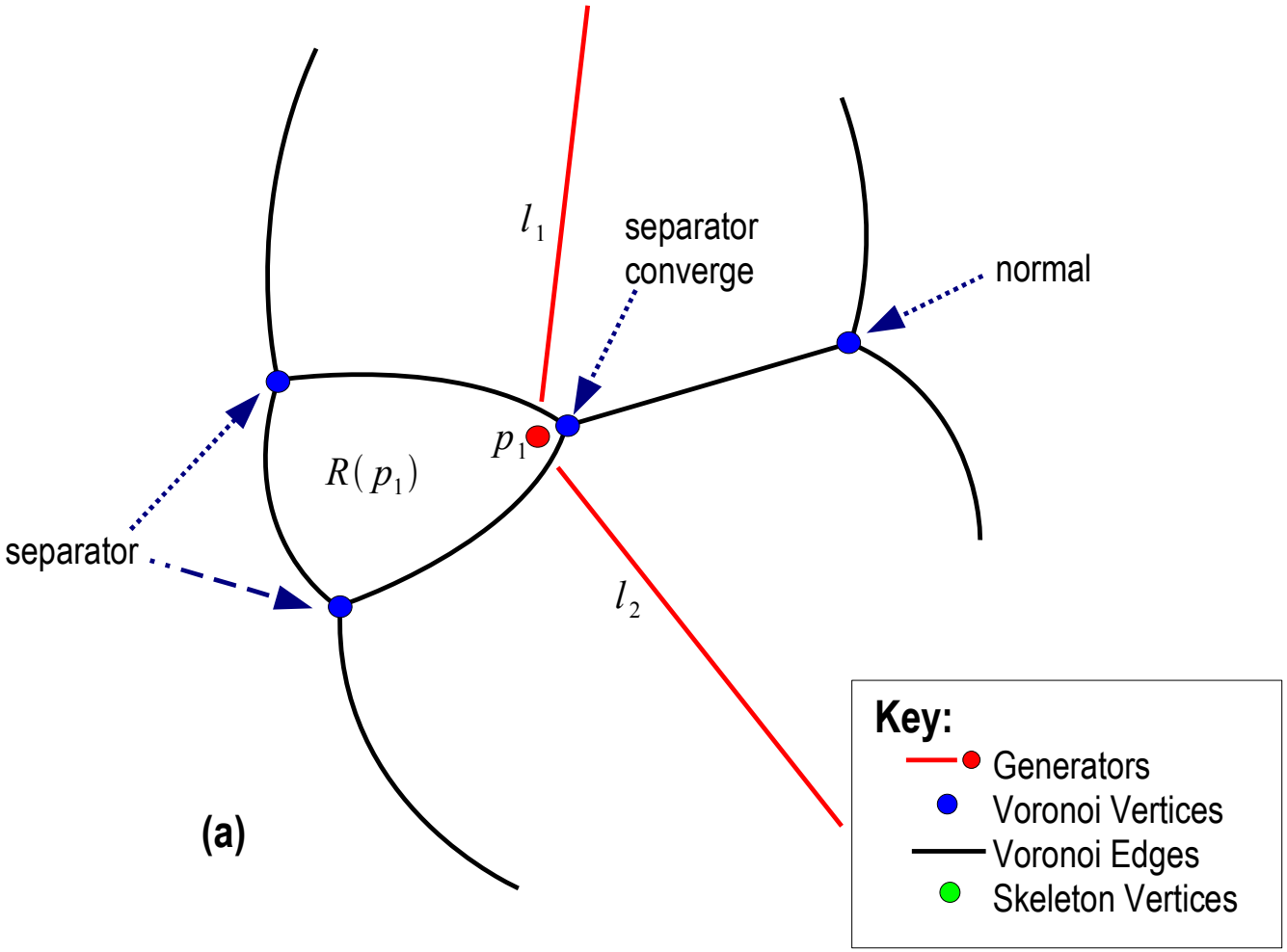
To order the insertion of generators such that the size of the structure to be deleted remains close to constant. This can be achieved if generators are uniformly distributed during insertion.
Method:

1. All generators are placed on a regular grid where each grid cell has an associated number m indicating the number of generators present within the cell.
2. The grid cells are then sorted in ascending order based on m.
3. The generators are ordered according to the index of their grid cell in the sorted list and by x-coordinate for generators that are within the same grid cell.

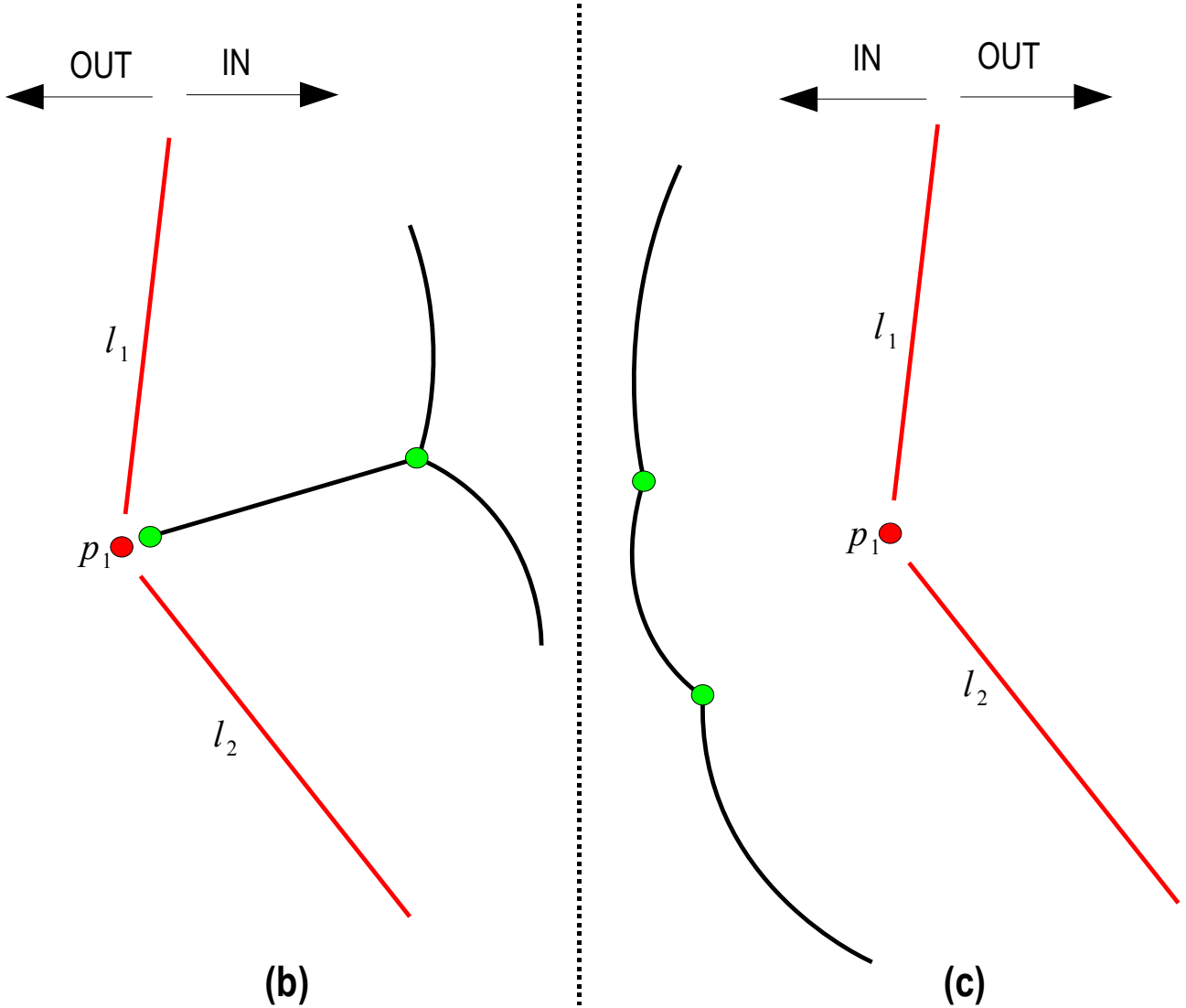
Trimming/Skeletonization algorithm

- A simple approach: Compute the orientation of each Voronoi vertex with respect to the closed contour associated with a region.
- Multiple Voronoi vertices lie on boundary points of the region. Due to numerical inaccuracy, determining the orientation of these points with respect to the contour becomes unreliable.
- Proposed approach: Label different types of vertices and edges when computing Voronoi diagram. Using this labeling if at least one of vertices can be determine to be "in" or "out" then all the other vertices in the primary cycle can be reliably determined.

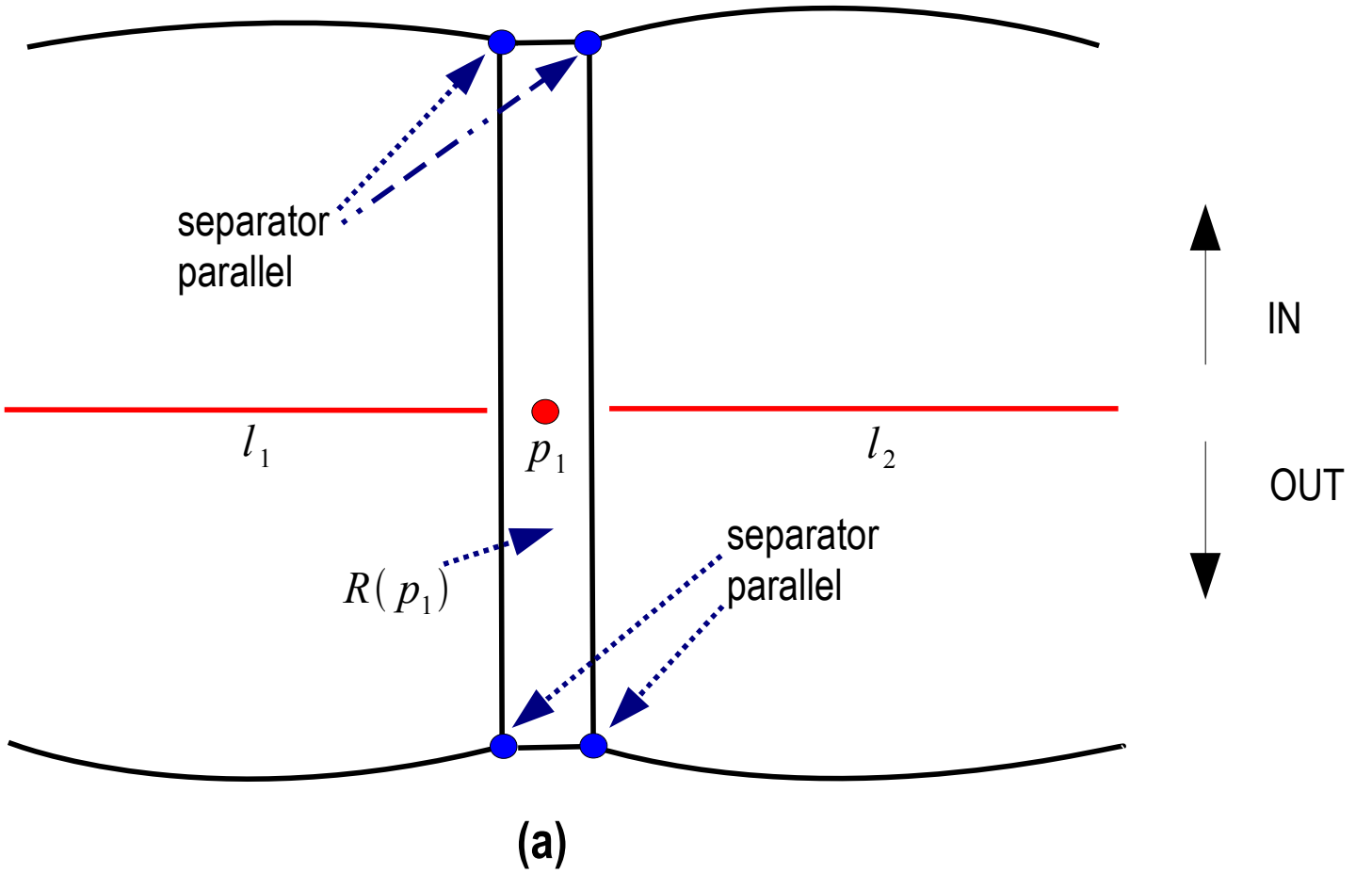
Converge Vertices - Primary cycle structure



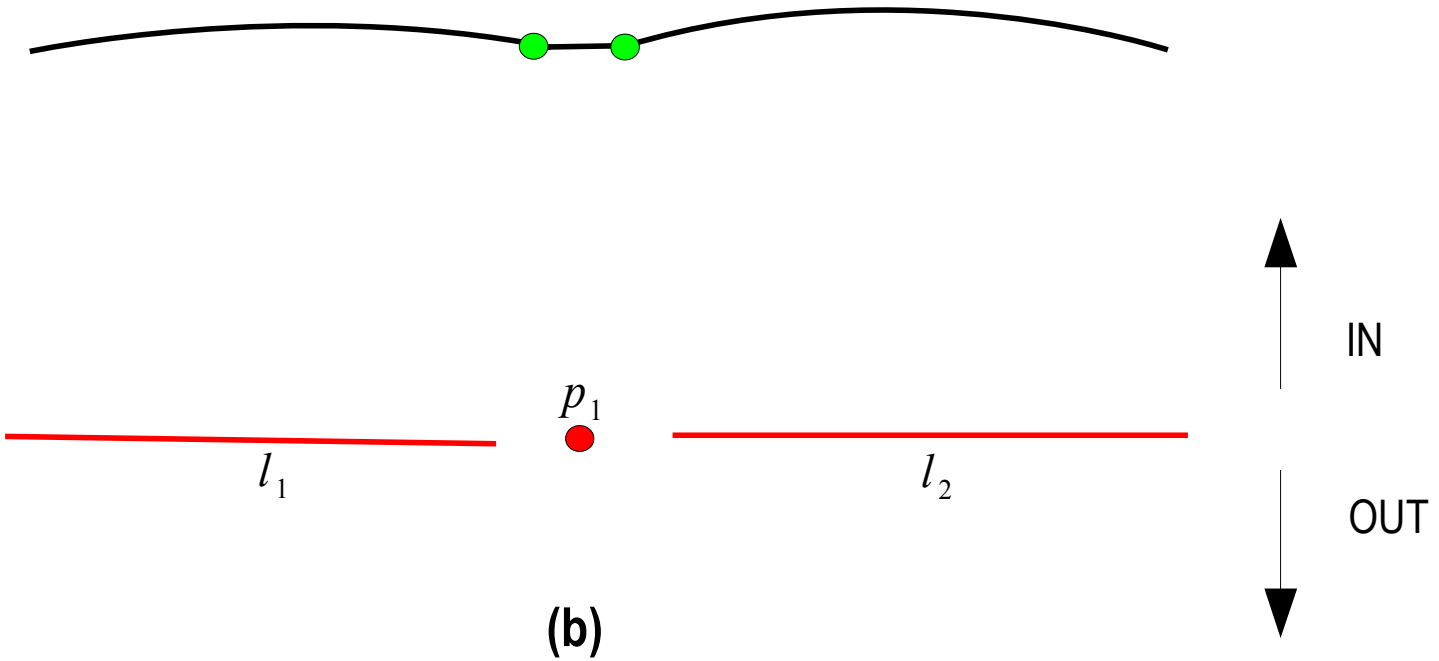
Converge Vertices - Skeleton



Parallel Vertices - Primary cycle structure



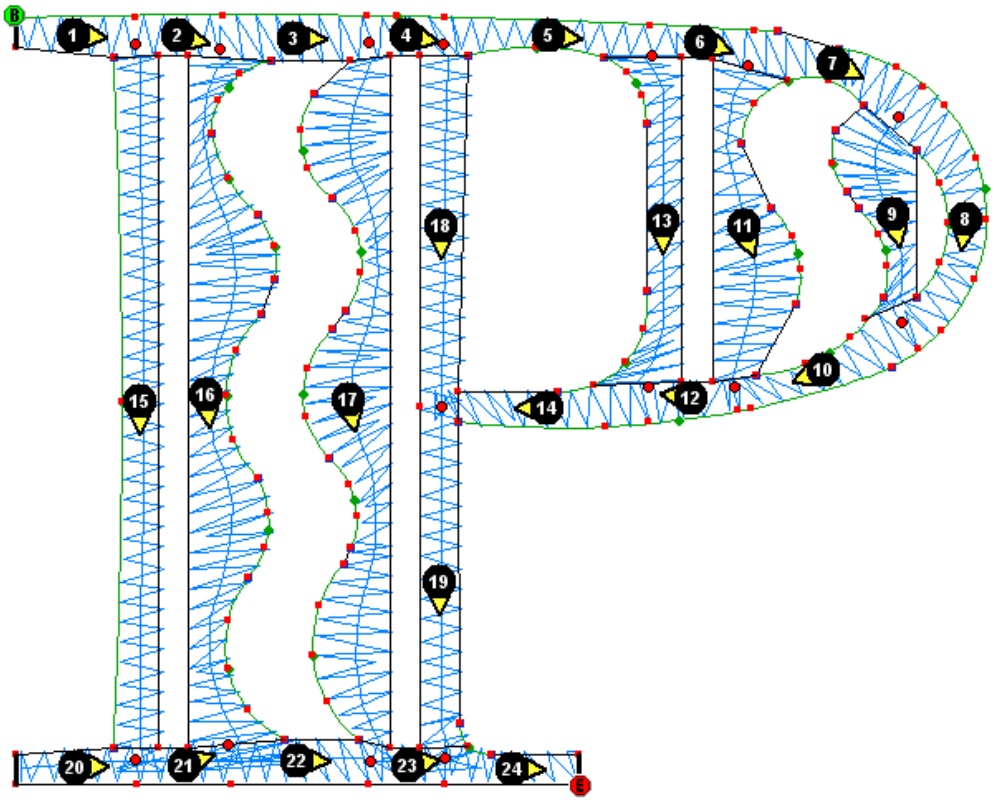
Parallel Vertices - Skeleton



Automated embroidery generation

1. Voronoi diagram is constructed for a given shape.
2. Skeleton for the given shape is generated from the Voronoi diagram.
3. The automation system then uses it in conjunction with outline and thickness/distance transform information to interpret and convert it to higher level embroidery design primitives.
4. These primitives are then used to generate detailed sequences of individual stitch locations that when sewn on embroidery equipment reproduce a visual representation of the shape in an aesthetically pleasing manner.

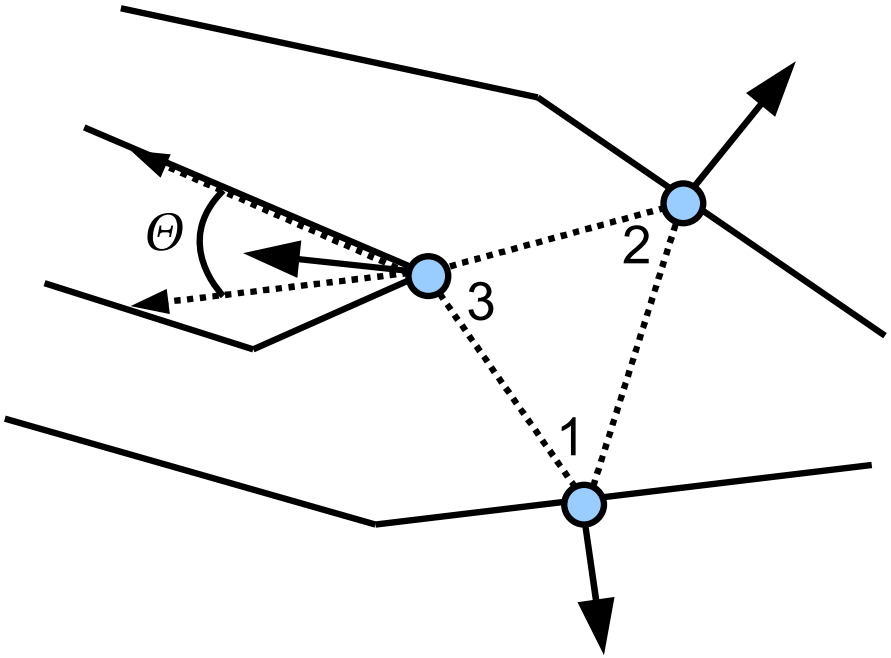
Generated stitch data for automated embroidery

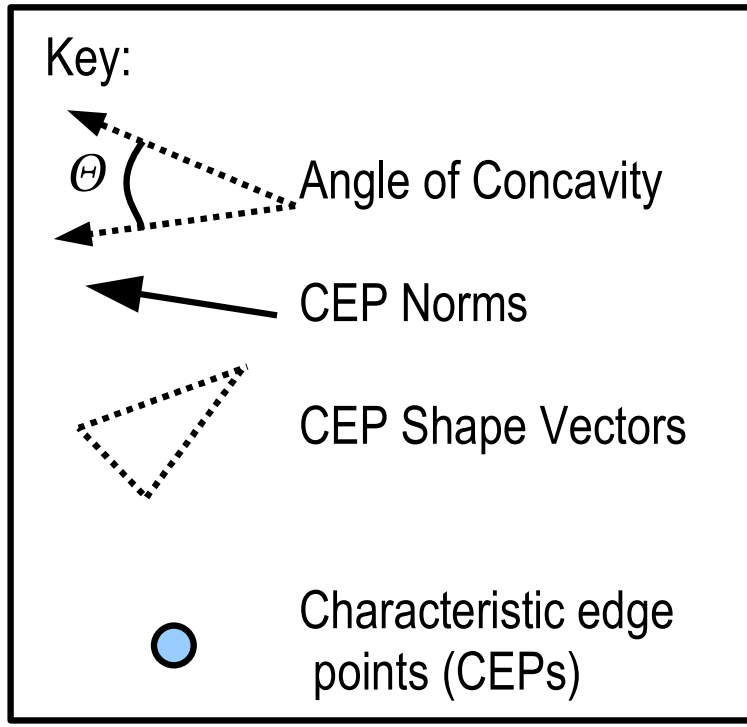


Structural Indexing

A rich source of information in visual data is present in the geometric structure of two dimensional shape. Structural Indexing tries to exploit this fact and the Voronoi skeletal is useful here to further decompose the structure of a shape. In the implementation presented here, Voronoi diagrams are used generate the skeletons of shapes using their edge contours as discussed previously. The edge contours are generated either from scanned images or from True Type Fonts.

Metrics used for structural indexing





Steps for structural indexing

1. The Voronoi skeletons are analyzed and converted to a set of skeletal nodes and connecting skeletal branches.
2. Several geometric metrics are computed from these skeletal nodes.
3. The metrics related to a skeleton node are encoded as a junction node and saved within a GTree (a database supporting multidimensional range queries).
4. Given the skeletal representation of an object, similar objects can be retrieved from the database.

Types of tests peformed

Manual input

- Testing particular parts of code.
- Creating degenerate cases.
- Gaining better visual understanding of the functioning of the algorithm.

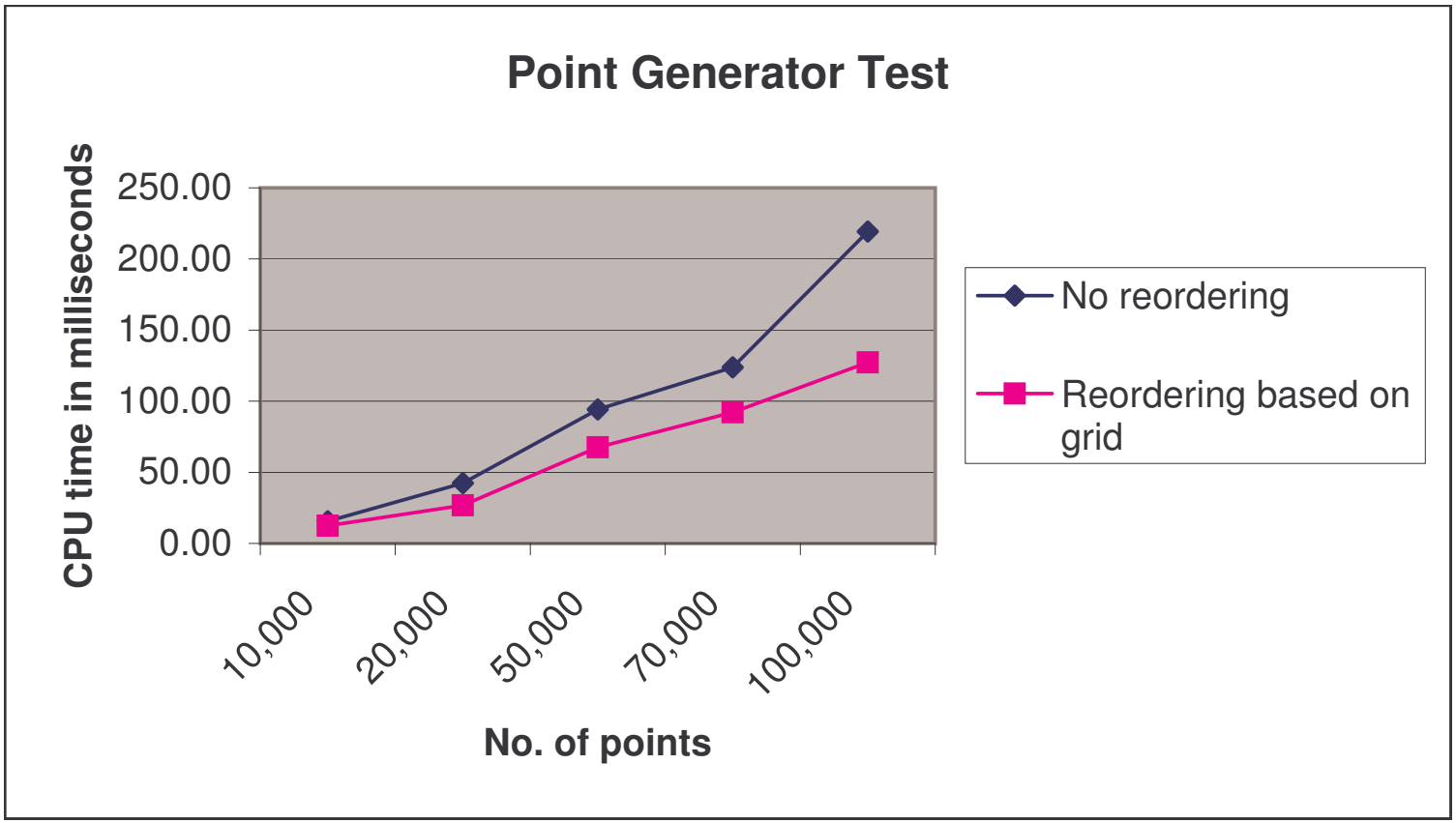
Random input

- Large scale testing of the code. The data-sets usually ranged from 10,000 to 100,000 generators.
- Comparing performance of code with other implementations.

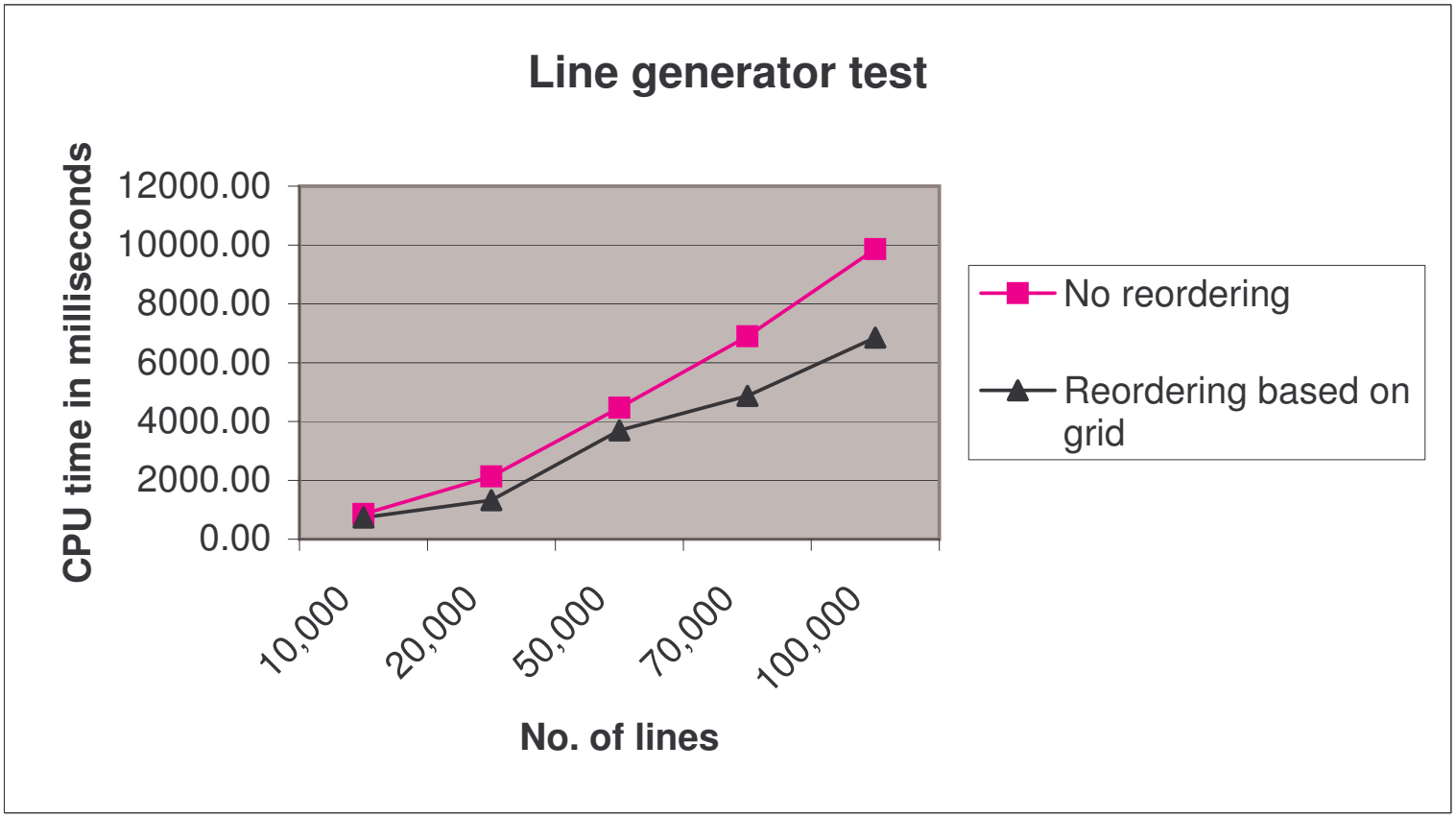
True Type Font outlines

- Automated testing of all the characters for a font at various sizes, orientations and positions in the plane.
- 12 million test cases in one such test.

Random point generators - Test results



Random segment generators - Test results

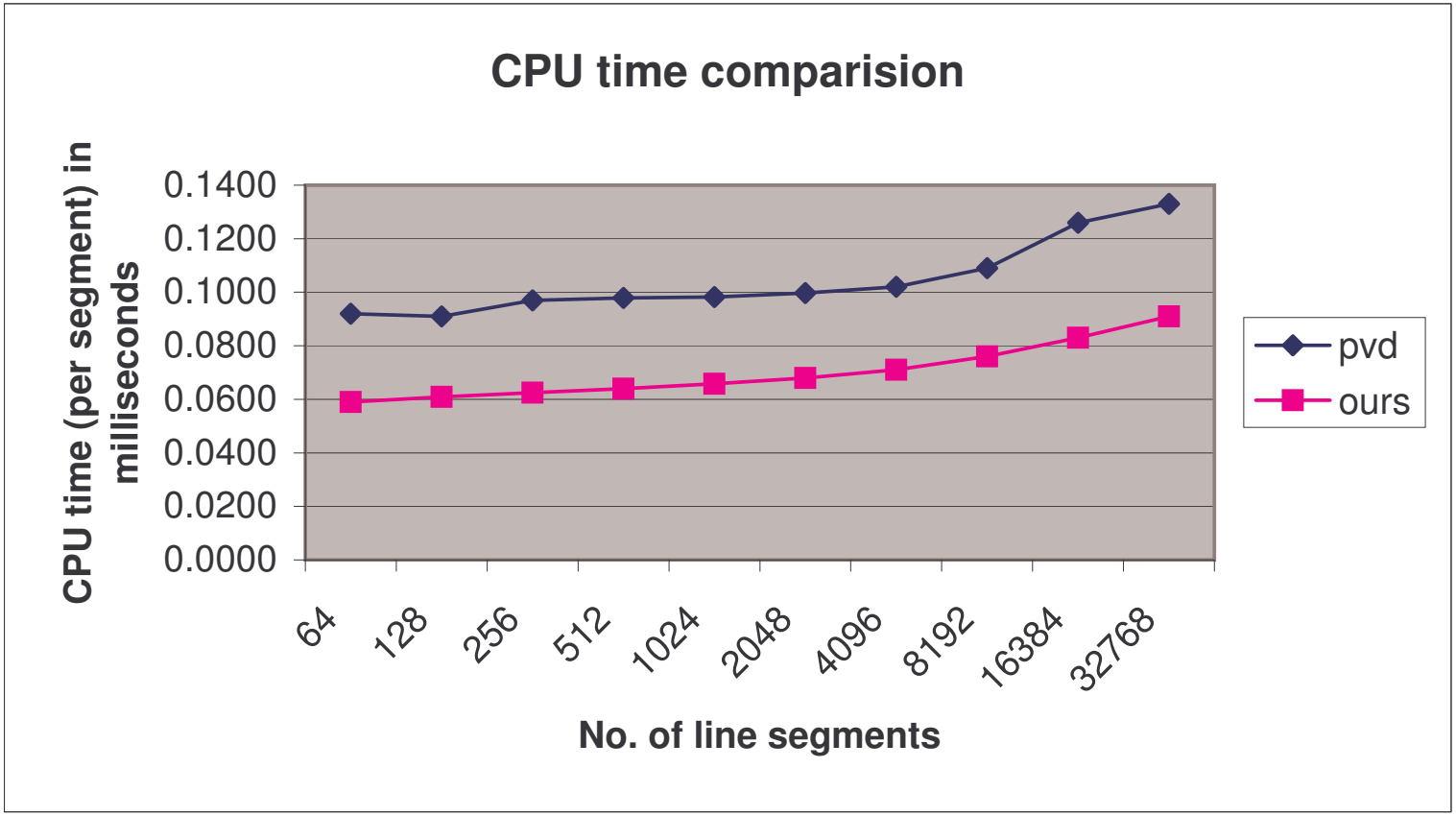


Comparision Tests

The code was then tested against existing publicly available implementations:

1. Seel's avd: This implementation is based on the exact arithmetic library from LEDA. Hence, the output of the algorithm can be verified if the implementation is correct.
2. Sethia's pvd: It is restricted to clean polygon data that forms the boundary of a multiple-connected area. pvd cannot deal with individual line segments.

Comparison Test results



Contributions of the thesis

- Implemented existing algorithms and techniques for Voronoi diagrams of point, segment and polygon generators by filling out missing details in the description of algorithms
- Extended the techniques to enhance efficiency of execution
- Ensuring robustness of the algorithm in presence of numerical inaccuracy resulting from floating point calculations
- Demonstrated the application of Voronoi diagrams

Future work

- Somehow manage to get Held's code for comparison!
- Extending the work to compute Voronoi diagrams for
  - Curves
  - Points, segments and polygons in 3-dimensional space
- Testing using different types of structures as input data like spikes, circles, eclipse.

Questions

Thank You