1994

Constructing Voronoi diagrams in L(1) and L(infinity) metrics with two plane sweeps.

Jianan. Wang
University of Windsor

Constructing Voronoi Diagrams in L_1 and L_∞ Metrics with Two Plane Sweeps

by

Jianan Wang

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science in Partial
Fulfillment of the Requirements for the Degree of
Master of Science at the
University of Windsor
Windsor, Ontario, Canada
1994

ISBN 0-612-01426-6

Name Jianan Wang

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Computer Science
SUBJECT TERM

0989
SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Library Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0998
Physical	0523
Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce-Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578
Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia, Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family Studies	0628
Industrial and Labor Relations	0629
Public and Social Welfare	0630
Social Structure and Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and Nutrition	0475
Animal Pathology	0476
Food Science and Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Biostatistics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996
Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Paleobotany	0345
Paleoecology	0426
Paleontology	0418
Paleozoology	0985
Palynology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0300
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575
Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463

Applied Sciences

Applied Mechanics	0346
Computer Science	0984
Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0384
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0451

Abstract

A plane-sweep method without using transformation but using two sweeps is used to construct the Voronoi diagram in L_1 (L_∞, respectively) metric of a set of n point sites in $O(n\log n)$ time and $O(n)$ space. The two sweeps advance from opposite directions and produce two symmetrical data structures called the Left-to-Right Shortest-Path-Map and Right-to-Left Shortest-Path-Map. The two maps are then tailored to produce the desired Voronoi diagram.

To my wife Qiu and son Zhen

Acknowledgments

I would like to take this opportunity to give the deepest thanks to my supervisor Dr. Y.H.Tsin for his constant guidance, encouragement, understanding, and financial support.

I extend my gratitude to my thesis internal examiner Dr. H.Towes and external examiner Dr. H.K.Kwan for providing valuable comments.

My thanks go as well to Dr. R.Frost, Dr. S.Bandyopadhay, Dr. J.Morrissey, Dr. R.Kent, Dr. L.Li, and Dr. Y.Park for their good advice and inspiration.

I would also like to thank Steve Karamatos, Walid Nyamneh and our secretaries Mary and Josette for their help.

Finally, I cannot miss this opportunity to express my thanks to all my graduate lab fellows who sit down with me day and night in front of the computers.

TABLE OF CONTENTS

List of Figures

Chapter 1 An Overview of Voronoi diagrams

Section 1 An Overview

The Voronoi diagram for a set of n points in the Euclidean plane is a classic geometric object named after the Russian mathematician G.Voronoi [32, 33]. It was introduced to the field of computer science in 1975 by Shamos and Hoey [27]. As the Voronoi diagram has many useful applications in various areas of science (eg. pattern recognition, robotics, etc), it has drawn considerable research interest over the last two decades. Many varieties of Voronoi diagrams have been proposed and many efficient algorithms have been designed to construct them.

The Original Voronoi Diagram Problem

Definition: Let S be a finite set of points on the plane R^2. For each site $s \in S$, the Voronoi region of s is the set:

$$V(s) = \{p \in R^2 | d(p,s) \leq d(p,t),\ for\ every\ \ t \in S\}$$

where d is the Euclidean distance function. The Voronoi diagram of S is the set:

$$Vor(S) = \{V(S) | s \in S\}$$

Clearly, the Voronoi diagram for a set of points on the plane is a partition of the plane into n polygon regions, each of which is associated with a given point.

The region associated with a point is the locus of points closer to that point than to any other given point in S.

Shamos and Hoey [27] used the divide-conquer method to create the $Vor(S)$. Fortune [8] used a plane sweep method with transformation to construct the same diagram. Both achieved the $O(n\log n)$ time and $O(n)$ space bound which have been shown to be optimal [29].

Once the Voronoi diagram is available, many important proximity problems such as the all-nearest-neighbor problem, the triangulation problem, and the EMST (Euclidean Minimum Spanning Tree) problem can be solved in $O(n)$ time.

Voronoi Diagram in L_p Metric

The Voronoi diagram discussed above is referred to as the Euclidean metric, which is called L_2 metric in Minkowski's formulation. The diagram can be naturally extended to the L_p metric where $1 \leq p \leq \infty$. The definition of L_p metric, $1 \leq p \leq \infty$, is as follow: given two points q_i and q_j in the plane R^2 with coordinates $q_i(x_i, y_i)$ and $q_j(x_j, y_j)$, the distance between q_i and q_j is defined as:

$$d_p(q_i, q_j) = \left(|x_i - x_j|^p + |y_i - y_j|^p\right)^{1/p},\ 1 \leq p \leq \infty.$$

F.K.Hwang [10] gave an algorithm for constructing the Voronoi diagram in L_1 metric in $O(n\log n)$ time. D.T.Lee and C.K.Wong [20] presented an algorithm for L_1 and L_∞ metrics in $O(n\log n)$ time, and proved that the Voronoi diagram in L_∞ could be transferred linearly into L_1 metric. Later, D.T.Lee [17] also presented an algorithm for L_p $1 \leq p \leq \infty$ metric in $O(n\log n)$ time. P.Chew and R.Drysdale [4] generalized L_p–metric to convex distance functions and produced an $O(n\log n)$ time and $O(n)$ space algorithm for constructing the corresponding Voronoi diagram. All the above mentioned algorithms use the divide-conquer method.

Voronoi Diagrams and Convex Hulls

Brown [3] was the first one to show a linkage between Voronoi diagrams and convex hulls. By using a geometric transformation technique named *inversion*, the problem of constructing a planar Voronoi diagram for a n point set can be transferred to the problem of constructing the convex hull of n points in 3–dimensional space. The result can be generalized to the higher-dimensional space. Specifically, the Voronoi diagram of a set S in R^d (d-dimensional space) corresponds to the convex hull of $f(S)$ in R^{d+1} through an inversion transform f.

Other Generalizations of Voronoi Diagram

The Voronoi Diagram of a Set of Objects other than Points The Voronoi diagram can be generalized in many ways. For instances, the sites can be polygons, circles or other geometric figures rather than points.

R.L.Drysdale,III and D.T.Lee [6] presented an $O\left(Nc^{\sqrt{logn}}\right)$ algorithm for constructing the Voronoi diagram for circles and line segments in R^2, Three years later, they presented a better result, that is $O\left(N(logN)^2\right)$ [5]. This result was improved to $O(n\log n)$ by Kirkpatrik [12].

Imai et al [11] extended the ordinary Euclidean geometry for n points to the one in the Laguerre geometry for n circles in the plane, where the distance between a circle and a point is defined by the length of the tangent line from that point to the circle. They presented an $O(n\log n)$ time and $O(n)$ space algorithm using the divide-conquer method. The Voronoi diagram in the Laguerre geometry can be used in determining if a point belongs to the union of n circles, finding the connection of n circles, and finding the contour of the union of n circles.

Higher-order Voronoi Diagrams (in a Plane) For the ordinary Voronoi diagram, each polygon is the locus of points nearest to one point (order-1). The higher-order diagrams consider the k sites which are closest to a query point. It can be used for interpolation, contouring, and classification.

The definition of the generalized Voronoi polygon $V(T)$ of a subset T of points is:

$$V(T) = \{p : \forall v \in T \ \forall w \in S - T, \ d(p,v) < d(p,w)\}$$

We can see that each point of the subset T is closer to p than the points not in T. For some T, $V_k(T)$ may be empty. If one lets k become n-1, one obtains the farthest point Voronoi diagram. D.T.Lee [16] presented an $O(k^2 nlogn)$ time algorithm. Edelsbrunner, O'Rourke, and Seidel [7] gave an $O(n^3)$ method for constructing the family of all higher-order Voronoi diagrams in R^d.

The farthest neighbor Voronoi diagrams (order-(n-1) Voronoi diagram) for n points can be constructed in $O(nlogn)$ time [13, 26] and has been used to solve the diameter problem.

Weighted Voronoi Diagrams In a weighted Voronoi diagram, every point is associated with a positive weight. Each polygon of a weighted Voronoi diagram is the locus of points whose weighted distance to a given point (site) is minimum [1]. Aurenhammer gave an $O(n^2)$ algorithm for constructing a multiplicative weighted Voronoi diagram.

Voronoi Diagram in High Dimensions Preparata found out that a Voronoi diagram in R^3 with n points can have $O(n^2)$ edges [23]. As mentioned above, there exists a technique called inversion transform f, which could transform a Voronoi diagram of a set S in R^d to a convex hull of $f(S)$ in R^{d+1} [3]. In this way we can use the convex hull algorithm to obtain a Voronoi diagram in higher dimensions [3].

Voronoi Diagrams Involving Obstacles C.Wang and L.Sehubert [34], R.Seidel [24], and A.Lingas [21] considered constrained Voronoi diagrams with obstacles and presented $O(n\log n)$ time and $O(n)$ space algorithm. Y.H.Tsin and C.Wang [31] gave an $O((m+n)\log(m+n))$ algorithm (m is the number of obstacles, n is the number of sites) for constructing the geodesic Voronoi diagram *Vor(S,O)*, where O is a set of parallel line segments and S is a set of sites. When m=0, *Vor(S,O)* goes back to the original Voronoi diagram *Vor(S)*. Their algorithm uses a left to right sweep and right to left sweep to build the geodesic Voronoi diagram.

Section 2 Techniques for Constructing the Voronoi Diagram

The lower bound for constructing a Voronoi diagram on n points in the plane must take $\Omega(n\log n)$ time and $\Omega(n)$ space [29].

A *brute-force method* to construct a Voronoi diagram is to construct its polygons one at a time. Each polygon is the intersections of n-1 half-planes if n is the number of sites in the plane, and it can be constructed in $O(n\log n)$ time [29]. So the total time needed to construct the Voronoi diagram is $O(n^2\log n)$.

The Incremental approach

The incremental algorithms are relatively simple which construct the Voronoi diagrams by adding a site at a time, and has worst-case time complexity $O(n^2)$ [9].

Divide–conquer approach

Shamos and Hoey [27] used a divide–conquer method to construct the Voronoi diagram in $O(n\log n)$ time and $O(n)$ space where n is the number of the sites of a set S in the plane. We briefly outline the algorithm below:

1. Partition the set S into two subsets S_L and S_R with approximately equal size according to the x-coordinate.

2. Recursively constructs $Vor(S_L)$ and $Vor(S_R)$.

3.1 Find the polygonal chain T which separates $Vor(S_L)$ and $Vor(S_R)$.

3.2 The edges of $Vor(S_L)$ located at the right side of polygonal chain T, and the edges of $Vor(S_R)$ located at the left side of polygonal chain T are deleted. The remainder is the Voronoi diagram of the set S.

Figure 1 is a set of sites in a plane which is divided into two approximately equal subsets.

Figure 1 A set of sites in a plane is divided into two approximately equal parts

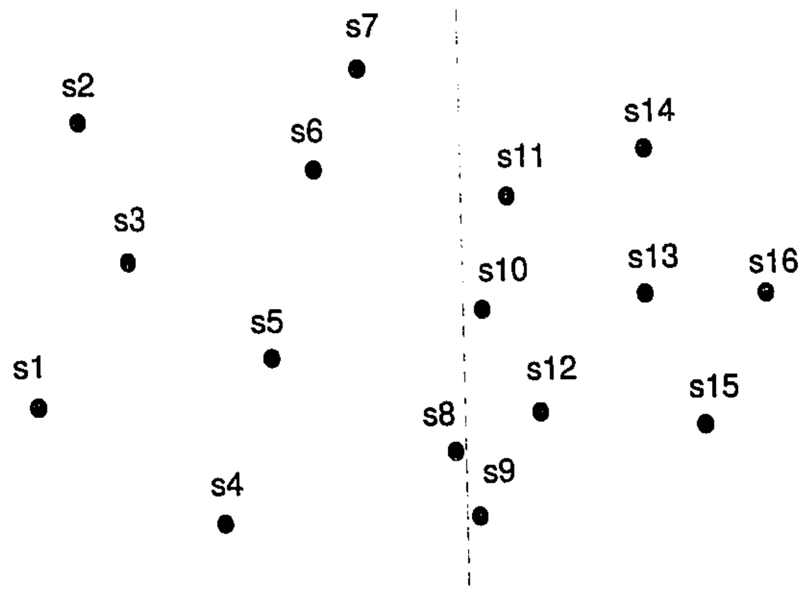
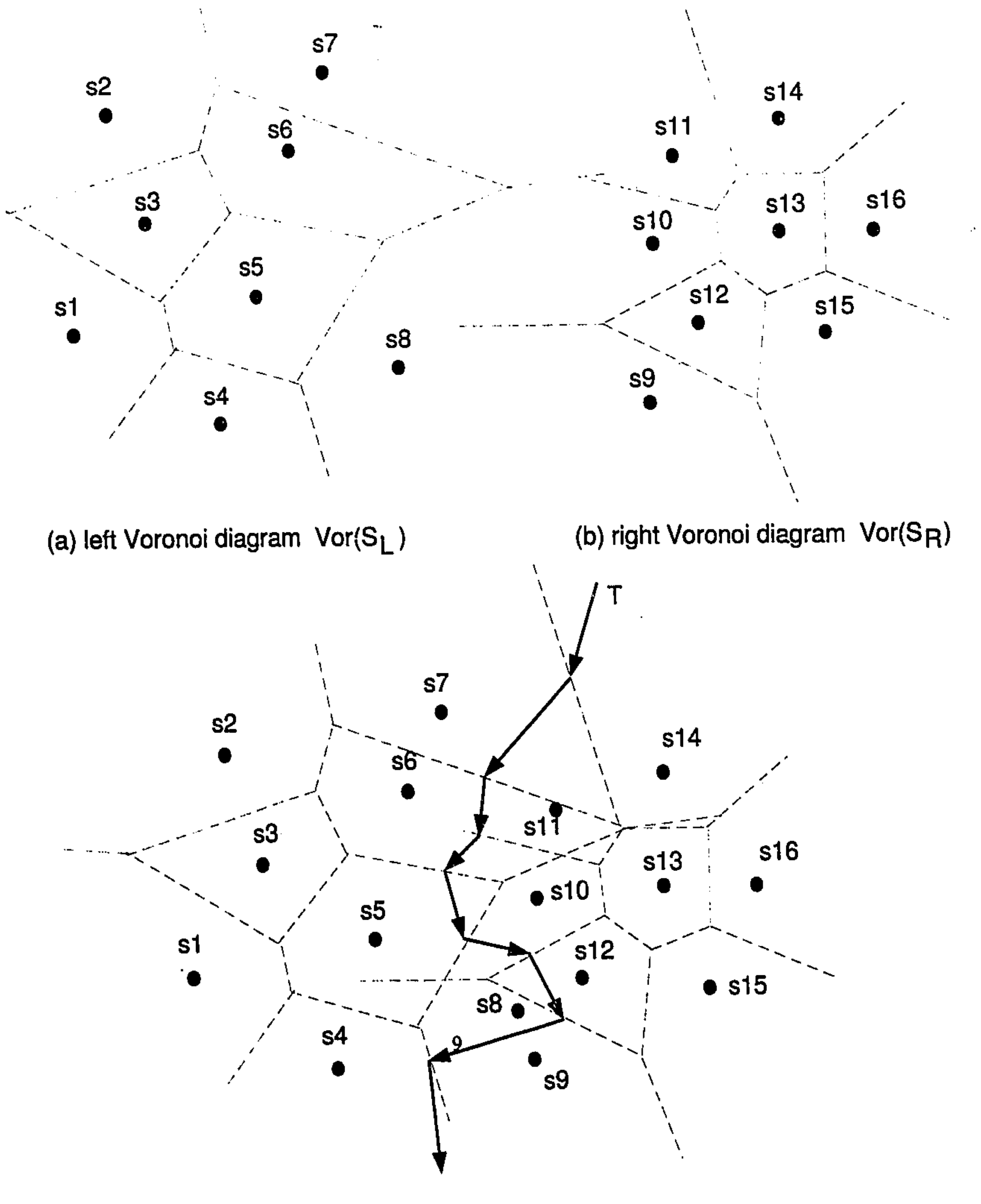


Figure 2 shows how the left Voronoi diagram *Vor(*S_L*)* and the right Voronoi diagram *Vor(*S_R*)* are merged into one by constructing the polynomial chain *T*.

Figure 2 The Voronoi diagram of the left subset *Vor(S_L)* and the Voronoi diagram of the right subset *Vor(S_R)* are merged into one



(a) left Voronoi diagram Vor(S_L)

(b) right Voronoi diagram Vor(S_R)

(c) A polygonal chain T used to merge Vor(S_L) and Vor(S_R)

Step 1 and step 3.2 can be done in $O(n)$ time. If step 3.1 can be done in $O(n)$ time, the time complexity would be $T(n) = 2T(n/2) + O(n) = O(nlogn)$ because of the recursive step 2.

The key point is to find the polygonal chain T which separates S_L and S_R (to merge the two parts into one) in $O(n)$ time.

First, the two semi-infinite rays of the polygonal chain T are found: A semi-infinite ray is the perpendicular bisector of a supporting line segment of the two convex hulls $CH(S_L)$ and $CH(S_R)$. Since each unbounded edge of the Voronoi diagram belongs to the perpendicular bisector of a convex hull edge, the $CH(S_L)$ and $CH(S_R)$ are the by-product of $Vor(S_L)$ and $Vor(S_R)$. The supporting line segments of $CH(S_L)$ and $CH(S_R)$ can be found in linear time [27].

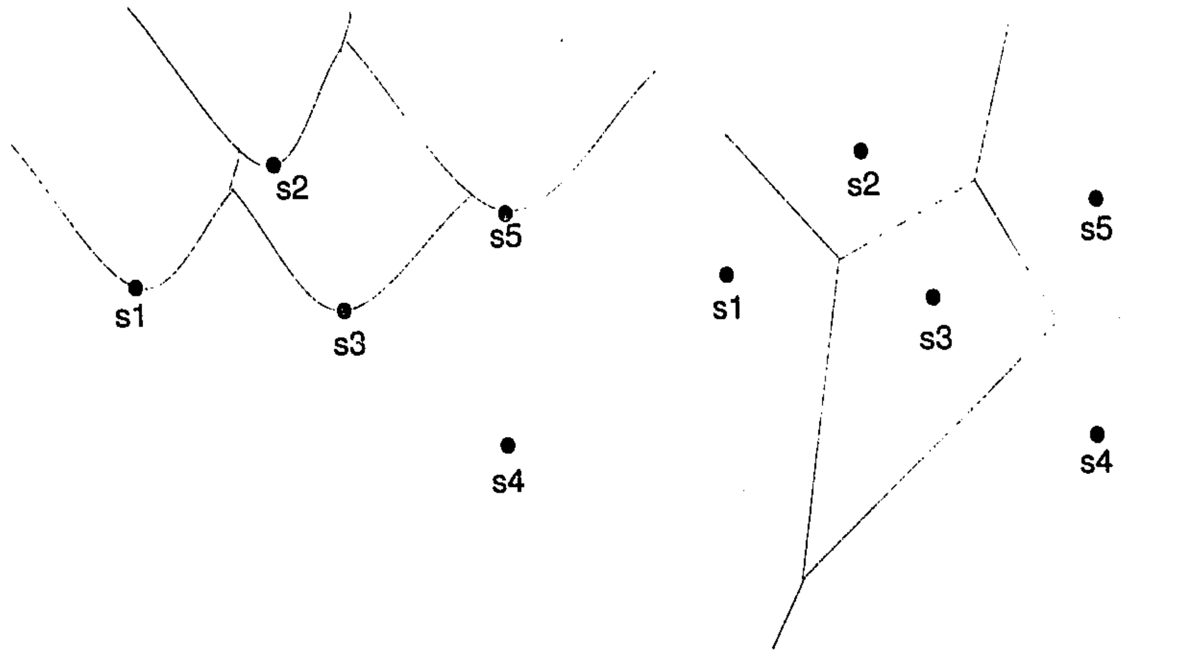
Then following one of the semi-infinite rays, the polygonal chain T is constructed edge by edge, until the other semi-infinite ray is reached. Each edge on T is the bisector of $Vor(S_L)$ and $Vor(S_R)$. A method without backtracking is used to avoid repeatedly scanning all the edges of a Voronoi polygon when the zig-zag segments of T walk several times inside the Voronoi polygon. Therefore, the polygonal chain T can be constructed in $O(n)$ time, the total time for constructing the Voronoi diagram is thus $O(n\log n)$.

Sweep line approach

Another algorithm presented by Fortune [8] used the plane sweep method augmented with transformation to construct the Voronoi diagram in *O(nlogn)* time and *O*(*n*) space.

The algorithm works as follow: The plane sweep proceeds from bottom to top. Whenever a site is reached, a transformed Voronoi region for that site is built. The transformed Voronoi region has the property that its bottom–most points coincides with the site. Furthermore, the boundary of the region is computed through properly maintained information between adjacent regions along the sweep line. Then by using the same transformation, the actual Voronoi diagram can be constructed. Figure 3 (a) shows a transformed Voronoi diagram constructed by a sweep line from bottom to top. Figure 3 (b) shows the original Voronoi diagram.

Figure 3 A geometric transformation to Voronoi diagram



(a)A transformation of a real Voronoi diagram

(b)The correspondent Voronoi diagram

Two plane sweep approach

Tsin,Y.H. and Wang,C. [31] [30] used a two plane-sweep technique to construct the geodesic Voronoi diagram *Vor(S, O)* for a set of sites S with a set of parallel line segments O as obstacles. Their method can be outlined as follows: The two sweeps proceeding from two opposite directions generate two data structures, called shortest-path-maps. From these two maps, for each $p \in R^2$, the closest site to its left and the closest site to its right can be determined.

Since the closest site of p must be one of these two closest sites, the two maps provide enough information to construct the Voronoi diagram. Tsin and Wang presented an $O((m+n)log(m+n))$ time $O(m+n)$ space algorithm where $|S|=n$ and $|O|=m$. When $m=0$, $Vor(S,O)$ becomes the original Voronoi diagram $Vor(S)$, and the time complexities are $O(nlogn)$ and $O(n)$ respectively.

A comparison of three methods

The divide-conquer method, although elegant, has been discovered to be very difficult to implement. One of the reasons is that the assumption that the sites are in general position is difficult to remove.

Fortune's plane sweep method is elegant, and has been successfully implemented, however, good transformations for other types of Voronoi diagrams are difficult to find.

The two sweep method of Tsin and Wang avoids the use of transformation at the cost of an extra sweep.

Chapter 2 Voronoi Diagrams in L_1 and L_∞ Metrics

Section 1 Voronoi Diagrams in L_1 and L_∞ Metrics

L_1 and L_∞ metrics are special cases of the L_p ($1 \le p \le \infty$) metrics (Minkowski metrics) defined as follows:

Definition (L_p metric): Given two points $q_i(x_i, y_i)$ and $q_j(x_j, y_j)$ in the plane R^2. The distance between q_i, q_j; under the L_p ($1 \le p \le \infty$) metrics is the distance

$d_p(q_i, q_j) = (|x_i - x_j|^p + |y_i - y_j|^p)^{1/p}$ $(p = 1, 2, \ldots, \infty)$.

Thus, the L_2 metric is the conventional Euclidean metric, that is:

$$d_2(q_i, q_j) = \left(|x_i - x_j|^2 + |y_i - y_j|^2\right)^{1/2}$$

.

L_1 Metric

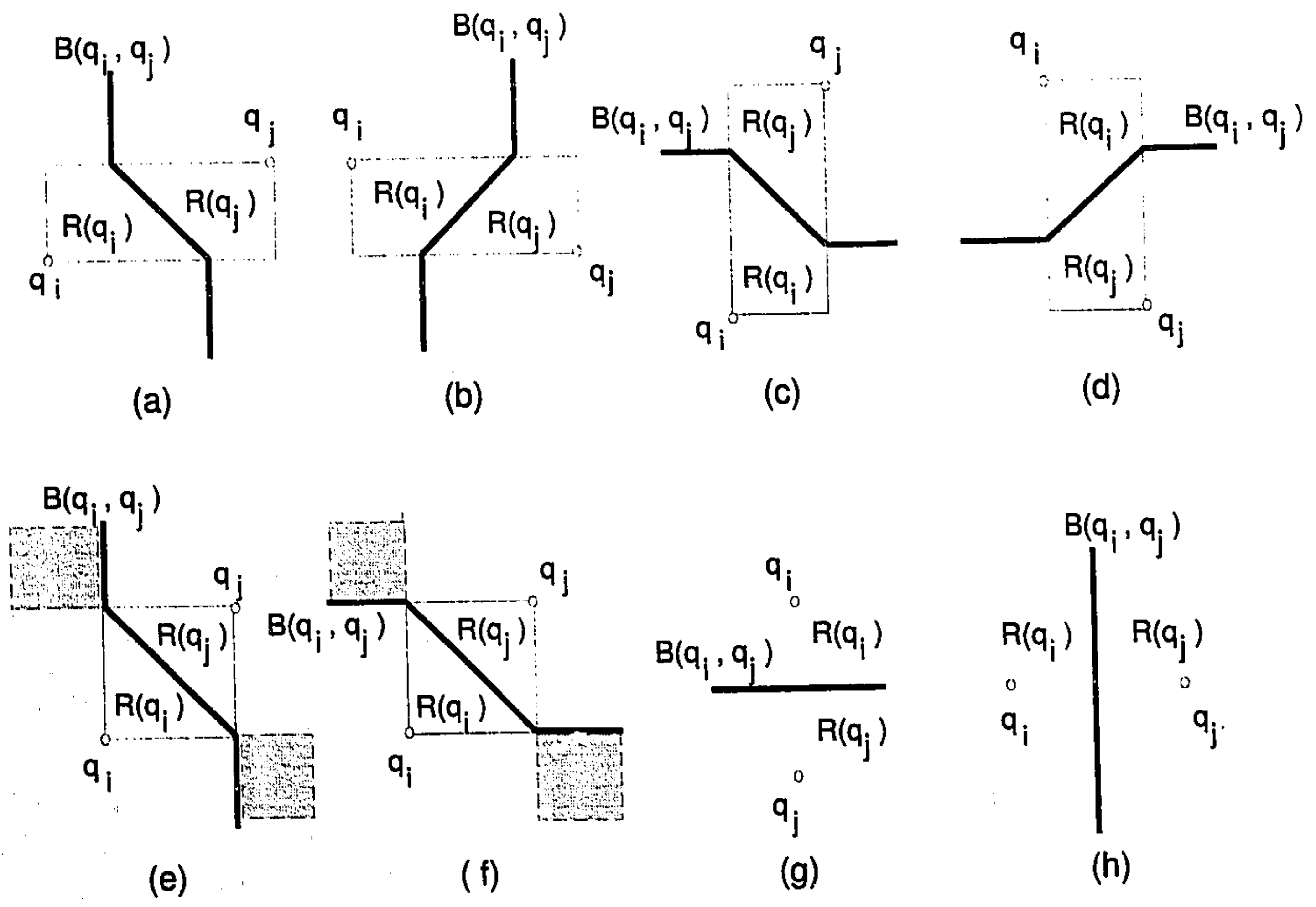
In L_1 metric, the distance between points $q_i(x_i, y_i)$ and $q_j(x_j, y_j)$ is defined as:

$d_1(q_i, q_j) = (|x_i - x_j| + |y_i - y_j|)$

It is a rectilinear distance between the two points q_i and q_j. The bisector of q_i and q_j, $B(q_i, q_j)$, partitions the plane into two regions, $R(q_i)$ and $R(q_j)$. All the points in region $R(q_i)$ ($R(q_j)$ respectively) is closer to point q_i (q_j respectively) than to point q_j (q_i respectively).

Figure 4 shows the different types of bisectors of q_i and q_j that divide the plane into regions $R(q_i)$ and $R(q_j)$:

Figure 4 Different types of bisectors of q_i and q_j that divide the plane into region $R(q_i)$ and $R(q_j)$



From figure 4 (e) and (f), we notice that part of the bisector has become a 2–dimentional region when $|x_i - x_j| = |y_i - y_j|$. We will adopt, for clarity, the solid lines shown in the figure (f) as the bisector in subsequent discussion. Figure 4 (g) and (h) are the degenerate cases where the bisector becomes a horizontal line and a vertical line when $x_i = x_j$ and $y_i = y_j$ respectively.

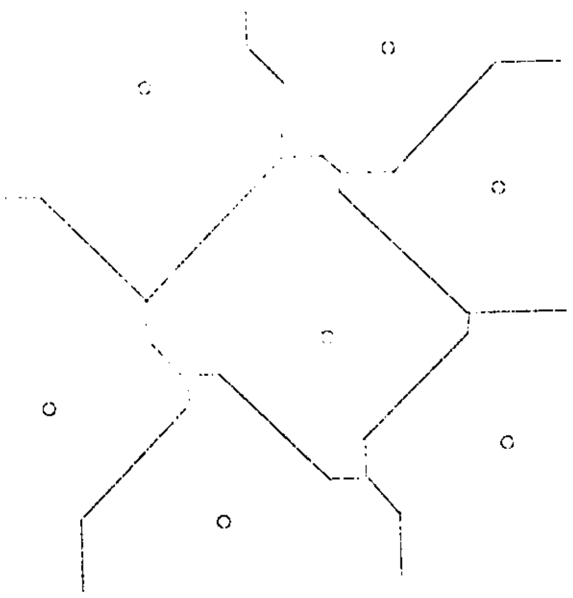
In summary, bisector consists of at most three types of segments: horizontal, vertical and 45° line segments.

Definition (Voronoi diagrams in L_1 metric): Given a set S of n points $\{p_1, p_2, \ldots, p_n\}$ in the plane R^2 under L_1 metric, a Voronoi region (or Voronoi polygon) $V(p_i)$ associated with a site p_i is a polygon, defined as

$$V(p_i) = \{r \in R^2 | d_1(r, p_i) \leq d_1(r, p_j), \forall j, 1 \leq j \leq n, \ i \neq j\}$$

Clearly, the n polygons $V(p_i)$ $(1 \leq i \leq n)$ form a partition of the plane, called the Voronoi diagram of S in L_1 metric, denoted by $Vor(S)$ (figure 5). We call the intersection points of the bisectors as Voronoi points, the segments between two Voronoi points as Voronoi edges.

Figure 5 Voronoi diagram in L_1 metric.



L_∞ metric

L_∞ is the distance measure in plane R^2_∞:

$$d_\infty(q_i, q_j) = max(|x_i - x_j|, |y_i - y_j|)$$

It is well known that L_∞ can be mapped by a linear transformation from L_1 metric [20]. Therefore, the construction of Voronoi diagram in $L\infty$ metric can be transformed to that in L_1 metric.

The Isometry of L_1 and L_∞

Both L_1 and L_∞ metrics are isometry. This can be explored as follows (figure 6): Two points $q_i(x_i, y_i)$ and $q_j(x_j, y_j)$ of R^2_∞ are mapped to points $q'_i(x'_i, y'_i)$ and

√

$q'_j(x'_j, y'_j)$ of R_1^2 respectively by the linear mapping $f : R_\infty^2 \rightarrow R_1^2$ such that $x'=(y+x)/2$ and $y'=(y-x)/2$. In figure 6, it is easily seen that the distance between q_i and q_j in R_∞^2 is

$$d_\infty(q_i, q_j) = |y_i - y_j| = 8$$

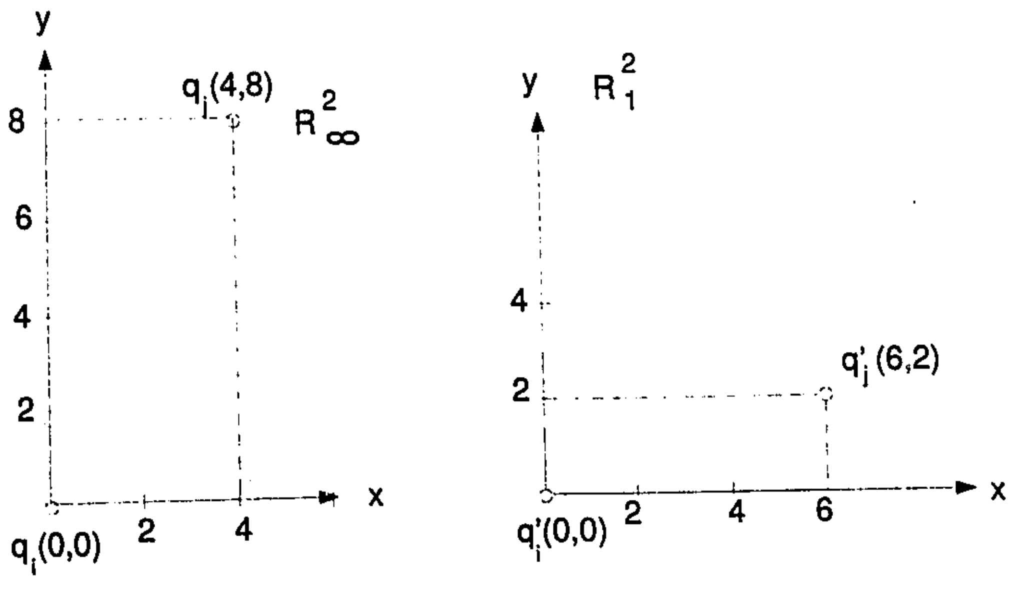
and the distance between q'_i and q'_j in R_1^2 is:

$$d_1(q'_i, q'_j) = (|x'_i - x'_j| + |y'_i - y'_j|) = 8$$

.

Since the distant measure in R_∞^2 can be mapped to the distant measure in R_1^2 and vice versa [20]. In the rest of this work we will only discuss the problems referring to L_1 metric.

Figure 6 The isometry of L_1 and L_∞ metrics



The applications of L_1 and L_∞ metrics in two-dimensional storage.

Lee and Wong [20] discovered that the Voronoi diagrams in L_1 and L_∞ metrics can be used to speed up retrieval in two-dimensional storage systems.

They pointed out that for a batch of *n* I/O requests [2] [28], scheduling the read/write head movement to visit all the records in a two-dimensional storage system in minimum time is an open path problem (OPP). They showed that using the Voronoi diagram in L_1 and L_∞ metrics, one can build a near-optimal path through each point either by constructing a minimum spanning tree or by the closest insertion method.

Section 2 Construction the Voronoi diagram in L_1 and L_∞ metrics by Divide and Conquer

D.T.Lee and C.K.Wong presented the divide and conquer method to construct the Voronoi diagram in L_1 metric in $O(n\log n)$ time [20]. We briefly outline the method below:

A preprocessing is needed to sort the n sites in lexicographical order.

1. Partition the set S into a left subset S_L and a right subset S_R with approximately equal size by the median x-coordinate.

2. Construct $Vor(S_L)$ and $Vor(S_R)$ recursively.

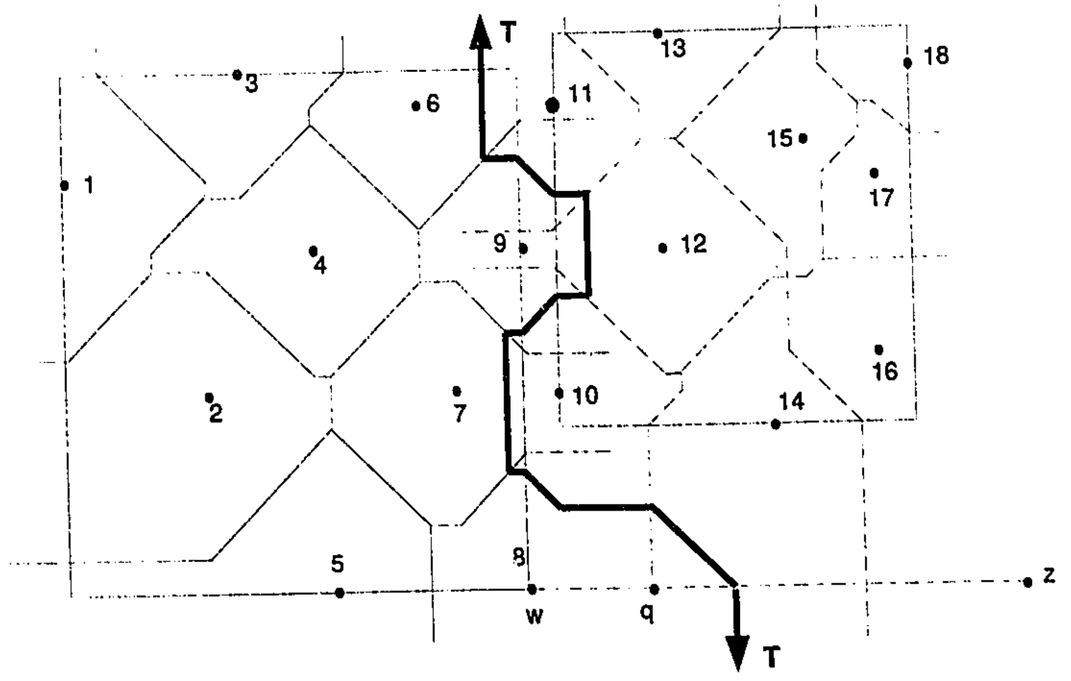
3.1 Construct the polygon line T which separates S_L and S_R.

3.2 Discard all edges of $Vor(S_L)$ that lie to the right of T, and all edges of $Vor(S_R)$ that lie to the left of T. What remains forms the Voronoi diagram $Vor(S)$.

The method for building the polygon line T in $O(n)$ time is as follows:

Figure 7 is an example illustrating the *merging* process.

Figure 7 Merging left subset and right subset.



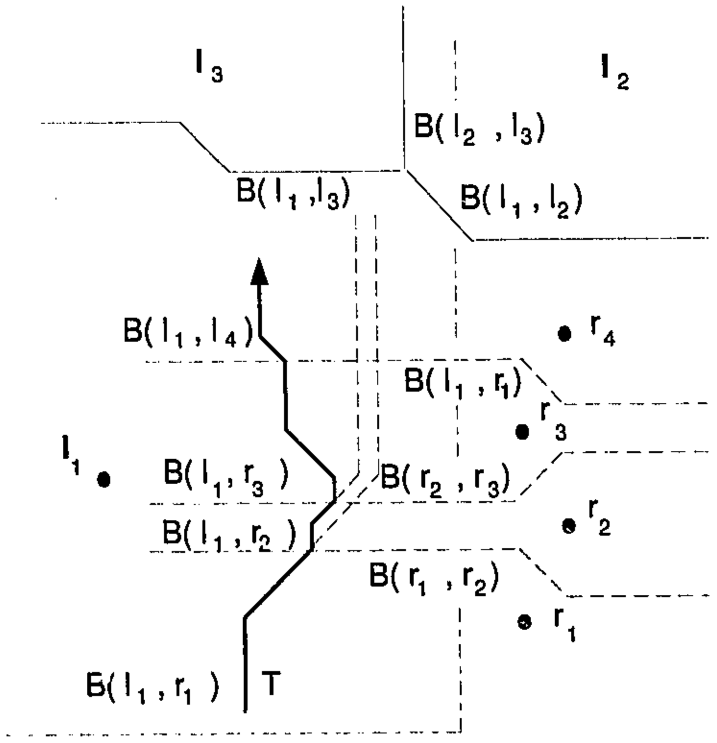
The first step is to find the starting segment of the polygon line *T*.

A set of 18 points with left subset S_L={1,2, . . . , 9} and right subset S_R={10, 11, . . . , 18}. The left subset S_L is enclosed by the smallest rectangle which includes all the sites of the S_L. The right subset S_R is enclosed by the smallest rectangle which includes all the sites of S_R. Find the rightmost site *w* with the smallest index of S_L. This means that the lower right corner of the smallest rectangle of the left subset S_L is inside *Vor(w)*. $\overline{wz}$ is the horizontal line segment with point *z* far enough to ensure intersection with the starting line segment of the polygon line *T*. Find the nearest neighbor *r* of *w* among the set S_R (that is point *10* in S_R). This takes *O*(n) time. Scan edges of *V*(10) to find an edge intersecting line segment $\overline{wz}$. In figure 7, the bisector *B*(10, 14) intersects $\overline{wz}$ at *q*. Compare the distance between point *w* and point *q* with that between point *w* and point *10*. Since the distance between point *w* and *q* is less than the distance between point *q* and point *10*, so the bisector of point *w* and point *10* does not intersect segment $\overline{wz}$. Then try another point in the right subset, that is point *14*. The bisector of point *w* and point *14* intersects segment $\overline{wz}$. This is the starting line segment of the polygon line *T*.

The second step is to follow a zig-zag path to construct the polygonal line *T* until it goes out of the smallest rectangle. At any time, an imaginary point *t* (on the polygon line *T*) always lies in two Voronoi polygons, one is in $Vor(S_L)$, the other is in $Vor(S_R)$. Whenever a new Voronoi point is created, the polygon line *T* will enter a new Voronoi polygon, and has a new direction. By scanning the boundary of *V(i)* in $Vor(S_L)$ and *V(j)* in $Vor(S_R)$, the next point can be determined. Sometimes, polygon line *T* will remain inside a polygon *V(i)*, turning many times before crossing an edge of *V(i)*. In this case, we need not to scan all the edges of *V(i)* each time (otherwise, this procedure will take as much as quadratic time). A method without back tracking is used to save time. Figure 8 illustrates the polygon line *T* penetrating into *V(i)*, meeting the edges $V(r_1)$, $V(r_2)$, $V(r_3)$, and $V(r_4)$ with counterclockwise turning without scanning all the V(i) edges each time.

Thus the total number of edges visited in constructing the polygon line *T* is proportional to the number of bisectors plus the number of Voronoi points on *T*. Both of them are *O(n)*. The merge process therefore takes *O*(*n*) time. Hence, the total time required to construct the Voronoi diagram of *n* sites is *O(n*log*n*), which is optimal in the worst case.

Figure 8 Tracing *T* in $V(L_l)$ without backtracking.



Chapter 3 Using Two Sweeps to Construct the Voronoi Diagrams in L_1 Metric

Section 1 Shortest–Path–Map

In order to generate the Voronoi diagram of a set of site S in the plane R^2, we need to know the closest site of each point $p \in R^2$. The two plane sweep technique [31] is used here to create two data structures, called the left-to-right Shortest–Path–Map $SPM_{\mapsto}(S)$, and the right–to–left Shortest–Path–Map $SPM_{\mapsfrom}(S)$, represented by a Doubly-Connected-Edge-List (DCEL) structure [29].

From the $SPM_{\mapsto}(S)$ and $SPM_{\mapsfrom}(S)$, the left closest site and the right closest site of each point $p \in R^2$ can be determined, using these information, the closest site of the p can be determined.

The total order $\prec$ is introduced here to sort the points in S. It is defined as $\alpha \prec \beta$ iff $x(\alpha) < x(\beta)$ or $x(\alpha)=x(\beta)$ and $y(\alpha)< y(\beta)$, where $x(\alpha)$, $y(\alpha)$ stand for the x and y coordinates of α, respectively. $\alpha \preceq \beta$ iff $\alpha \prec \beta$ or $\alpha = \beta$.

For clarity, we assume without loss of generality that no two or more sites have the same x-coordinate.

Definition: Let S be a set of sites in the plane R^2. The Left–to–Right Shortest–Path–Map of a set of sites S in the plane R^2 is a set of regions

$\{\Delta(s_i) | s_i \in S, 1 \leq i \leq n\}$ such that region $\Delta(s_i)$ is the locus of the point p such that:

i) $x(s_i) \leq x(p)$;

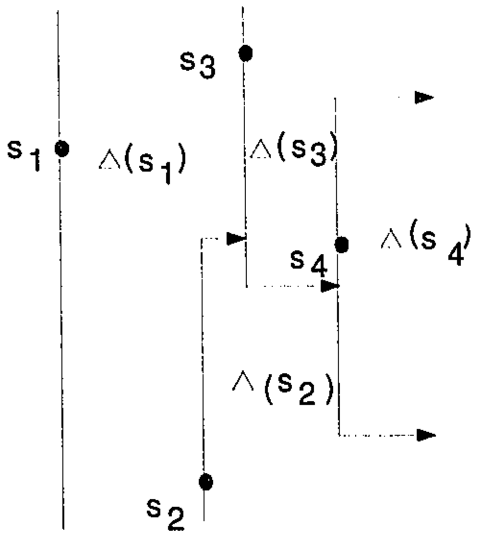
ii) s_i is the left closest site of point p among all the sites s', no lying to the right of s_i, i.e.,

$$d_1(s_i, p) \leq d_1\left(s', p\right) \ \forall s', \ x\left(s'\right) \leq x(p)$$

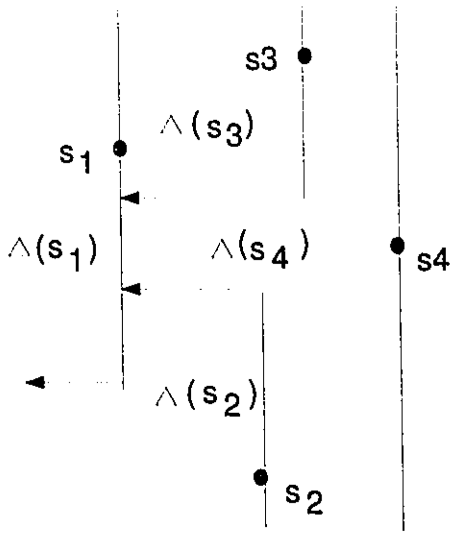
.

The Right–to–Left Shortest–Path–Map of a set of sites S in the plane R^2 can be defined in the similar way.

Figure 9 A left-to-right shortest-path-map and a right-to-left shortest-path-map



(a) A left to right Shortest Path Map



(b) A right to left Shortest Path Map

Definition:

The edges in a Shortest–Path–Map are called SPM arcs. The horizontal segments (part of the bisectors) are called horizontal SPM arcs (h_SPM arcs) which sit at the same side of their two sites. The starting point of a h_SPM arc is the corner of the region of a site. The ending point is at (1) infinity or (2) on the vertical boundary of a site. The vertical segments (sweep line or part of the sweep line) are called vertical SPM arcs (v_SPM arcs).

Section 2 The Construction of Shortest–Path–Map (SPM)

To construct the $SPM_{\vdash\!\to}(S)$ ($SPM_{\leftarrow\!\dashv}(S)$ respectively), a preprocessing is needed to sort all the sites $s_i \in S(1 \leq i \leq n)$ in lexicographical order $\prec$. The construction will start from the leftmost site s_1 and terminate after passing through the rightmost site s_n.

Lemma 3.1 Let $s_i \in S$, and the sweep line is positioned at site s_i. Let v_h be a point on l_i such that $v_h \in \Delta(S_h)$ where $s_h \in S$. Then

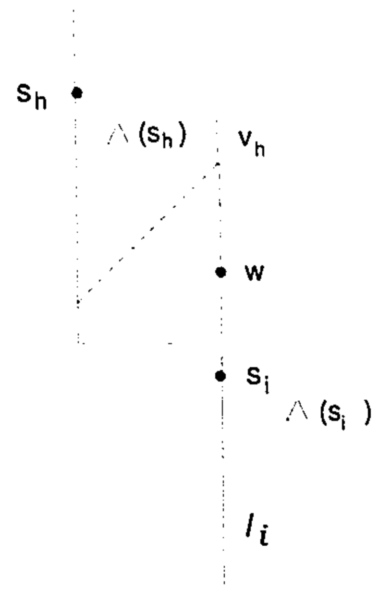
$$d_1(s_i, v_h) \leq d_1(s_h, v_h) \Rightarrow d_1(s_i, w) < d_1(s_h, w) \; \forall w \in \overline{s_i v_h} - \{v_h\}$$

Proof:

$$d_1(s_i, w) + d_1(w, v_h) = d_1(s_i, v_h) \leq d_1(s_h, v_h) < d_1(s_h, w) + d_1(w, v_h)$$

$$\Rightarrow d_1(s_i, w) < d_1(s_h, w) \blacksquare$$

Figure 10 $d_1(s_i, w) < d_1(s_h, w)$

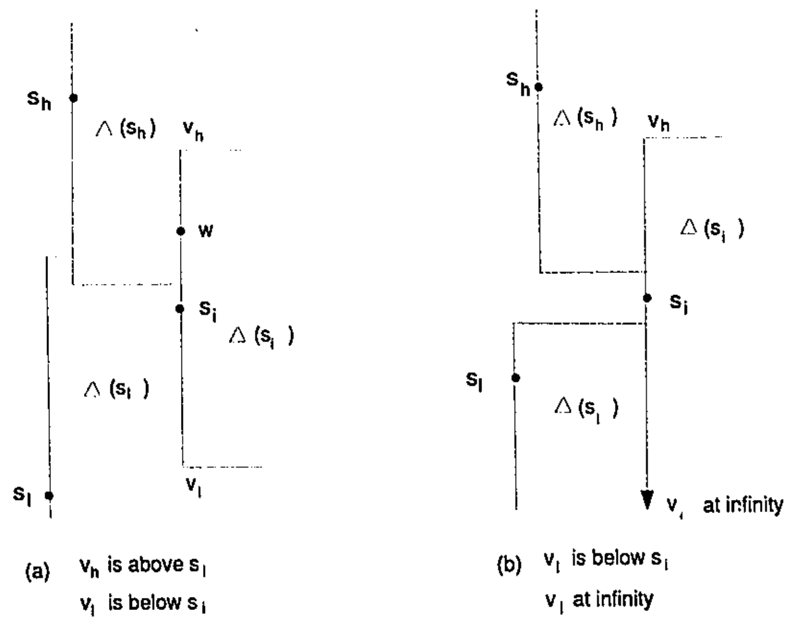


Lemma 3.2 Let $s_i \in S$, the initial region of $\Delta(s_i)$ is bounded on the left by the line segment $\overline{v_h v_l}$ where v_h (v_l, respectively) is a point above (below, respectively) s_i such that either v_h (v_l, respectively) is at infinity or $v_h \in \Delta(s_h)$ ($v_l \in \Delta(s_l)$, respectively) such that $d_1(s_h, v_h) = d_1(s_i, v_h)$ ($d_1(s_l, v_l) = d_1(s_i, v_l)$, respectively).

In the latter case, $\Delta(s_i)$ is bounded above (below, respectively) by a bisector between s_i and s_h (s_l, respectively).

Proof: The lemma follows from lemma 3.1 ■

Figure 11 The upper boundary and lower boundary of an initial region $\Delta(s_i)$



Definition: The point v_h is called the upper corner of $\Delta(s_i)$ and point v_h is called the lower corner of $\Delta(s_i)$.

It is clearly seen that all the bisectors in $SPM_{\mapsto}(S)$ and $SPM_{\mapsfrom}(S)$ are horizontal line segments, so that they are monotone with respect to the x-axis.

Some data structures we used here are listed below:

a) Queue

b) Balanced Binary Searching Tree

c) Doubly-Connected-Edge-List

Queue (Q)

The Queue Q stores all the sites of S in the lexicographical order $\prec$. The sweep treats each site as an event point. When the sweep line moves from left to right, these sites will be dequeued one by one in order starting from the leftmost site s_1 to the rightmost site s_n.

Balanced Binary Tree (T)

The balanced binary tree T stores all the bisectors intersected by the sweep line. The bisectors are ordered from bottom to top according to their y-coordinates. Each node of the tree stores the same information of a bisector that stored in the Doubly-Connected-Edge-List (DCEL) including the y-coordinate of the bisector, F_L (field name of left side of the bisector), F_R (field name of right side of the bisector), the coordinates of the two sites of the bisector.

Doubly-Connected-Edge-List

The Doubly-Connected-Edge-List is used to store the information of a planar graph embedded in the plane [29]. Each edge of the graph is described by a list of items:

1. v_s: starting vertex.
2. v_e: ending vertex.
3. F_L: field name of left side of the edge.
4. F_R: field name of right side of the edge.

5. Field: field name to which the edge belongs
6. h_SPM: horizontal SPM arc (part of a bisector).
7. v_SPM: vertical SPM arc. (sweepline or part of the sweepline).
8. t2–v: type2 Voronoi arc (part of a bisector).
9. site (pointer):
    a. if the edge is a vertical line segment, this item has one pointer pointing to the site through which the vertical segment passes .
    b. if the edge is a h_SPM (part of the bisector), this item includes two pointers pointing to two sites corresponding to this edge.

It is easy to find all the edges that enclose a certain field, or a vertex at which the edges meet.

A DCEL is represented in an edge-list form. Figure 12 is an illustration of DCEL.

The Construction of $SPM_{\mapsto}(S)$ $\left(SPM_{\leftarrow\!|}(S)\right)$

First, the leftmost site s_1 is removed from the queue Q. This effectively positions the sweep line over s_1. Starting from s_1 the sweep line moves towards the right, stopping at every event point until it passes through the last (i.e. rightmost) element s_n in Q.

As the sweep line is moving from the leftmost site s_1 to the right, all the information of the edges that form the $SPM_{\mapsto}(S)$ are recorded into the data structure DCEL. An example below (Figure 12) shows how the $SPM_{\vdash}(S)$ is constructed and how the DCEL is formed.

Figure 12 The beginning of the $SPM_{\vdash}(S)$

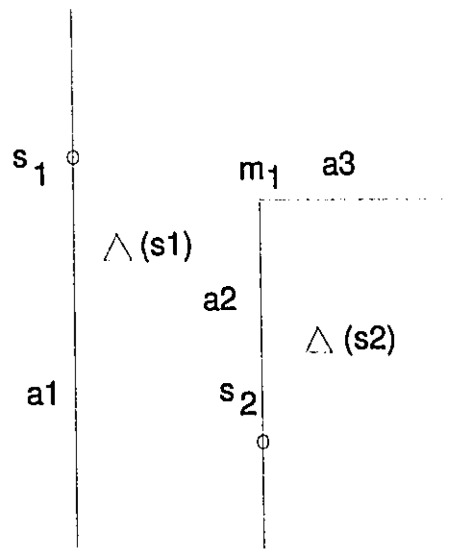


Figure 13 The corresponding DCEL represents part of the $SPM_{\vdash}(S)$.

	v_s	v_e	F_L	F_R	h_SPM	t2_v	v_SPM	site(ptr)
a1	$-\infty$	∞	0	1	0	0	1	1
a2	$-\infty$	m1	1	2	0	0	1	2

Note that edge *a3* may be truncated by the other edge at later stages. The information of edge *a3* is now saved in the binary tree T (figure 16). It will be saved in the DCEL only after its ending point v_e is certain (whether at ∞ or truncated by a vertical edge).

At the time the sweep line stops at site s_3, the intersection point *m3* between the sweep line l_3 and the bisector *a3* is found. Site s_3 is in the initial region of $\Delta(s_1)$, the bisector of (s_1) and (s_3) has no intersection with the sweep line l_3. Therefore, site s_3 is bounded above at infinity. The intersection point *m3* on the sweep line l_3 is below site s_3. All the points on the sweep line l_3 below *m3* is in the initial region $\Delta(s_2)$. The bisector of s_2 and s_3 has an intersection point *m2* on l_3. Edge *a6* is part of the bisector, it is a h_SPM arc. Now, *a4, a5, a6* are added into the list, edge *a6* is part of the bisector of s_2 and s_3. Edge *a3* is truncated by the sweep line l_3 which passes through site s_3. Edge *a3* is added to the DCEL. The updated $SPM_{|\rightarrow}(S)$ and DCEL are showed below. Note that the ending point of *a6* is uncertain at this stage, and hence cannot be saved into DCEL.

Figure 14 $SPM_{|-}(S)$ when sweep line l_3 passes through s_3

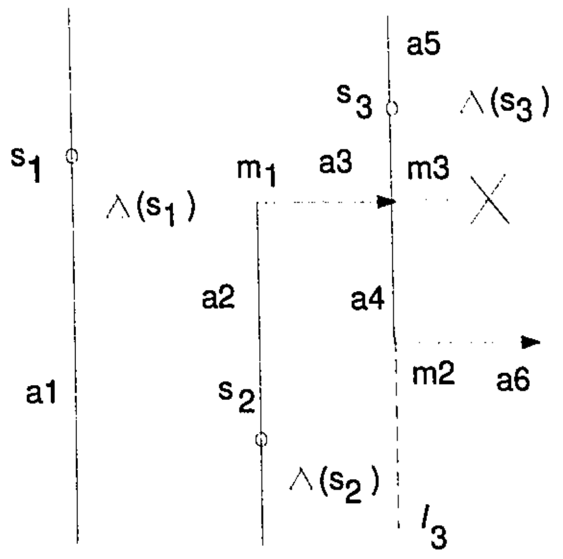


Figure 15 The corresponding DCEL represents the $SPM_{|-}(S)$. when sweep line passes through s_3

	v_s	v_l	F_L	F_R	h_SPM	t2_v	v_SPM	site(ptr)
a1	-∞	∞	0	1	0	0	1	1
a2	-∞	m1	1	2	0	0	1	2
a3	m1	m3	1	2	1	0	0	1/2
a4	m2	m3	2	3	0	0	1	3
a5	m3	∞	1	3	0	0	1	3

Lemma 3.3 Finding the upper boundary and lower boundary of the initial region $\Delta(s_i)$ when the sweep line is positioned at site s_i can be done in $O(\log n)$ time.

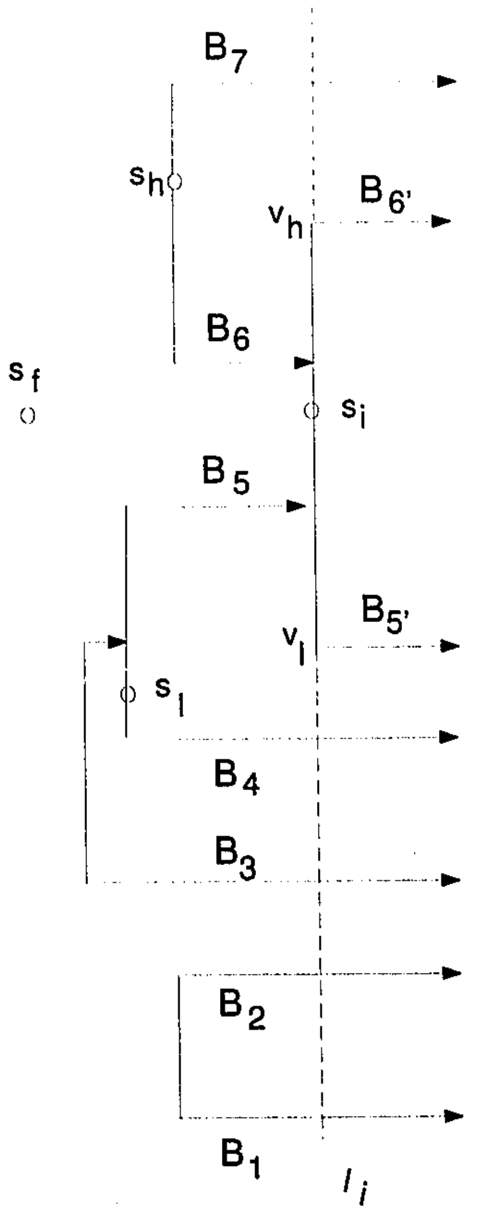
Proof: At the time the sweep line is positioned at the event point s_i, the binary search tree records the order of the bisectors that intersect the sweepline from bottom to top according to their y-coordinates.

Using the y-coordinate of site s_i as the key, search T for the region $\Delta(s_i)$ containing s_i. Then examining the bisectors above s_i (below s_i respectively) along the sweep line l_i. If no bisector above s_i (below s_i respectively) is found, that means the upper corner v_h (lower corner v_l, respectively) is at positive infinity (negative infinity respectively) of y-coordinate. Otherwise, v_h (v_l, respectively) is determined and the initial region $\Delta(s_i)$ is bounded on the left by the line segment $\overline{v_h v_l}$, bounded above (below, respectively) by a h_SPM arc originated at v_h (v_l, respectively) if v_h (v_l, respectively) exits. All the bisectors that intersect the segment $\overline{v_h v_l}$ are truncated by $\overline{v_h v_l}$, and are thus deleted from the balanced binary search tree T. The newly generated bisectors are then added into T (figure 16).

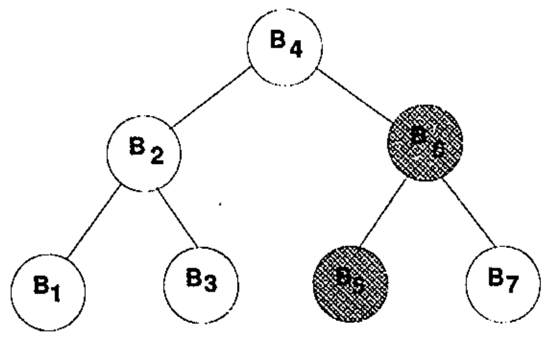
Special cases discussion: It is possible that no bisectors (h_SPM arcs) are met by the sweep line passing through a site. This happens: (a) at the beginning of the sweep, when the sweep line l_1 is positioned at site s_1, or (b) when the vertical SPM arc passes through site s_i truncating all the bisectors and without generating a new bisector, (Figure 17 (a) and (b)). When it happens, the binary search tree T is an empty tree.

Figure 16 is an example to illustrate how to use the balanced binary searching tree T to find the boundaries enclosing the initial region $\Delta(s_i)$ in $O(\log n)$ time.

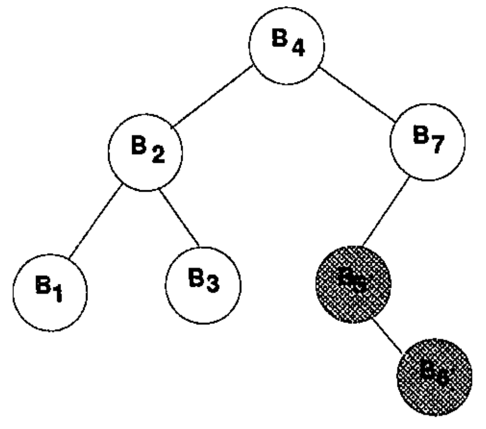
Figure 16 Using the balanced binary search tree to find the initial region $\Delta(s_i)$ in $O(\log n)$ time.



(a) A sweep line positioned at the site s_i, vertical segment $\overline{v_h v_l}$ truncates the bisectors B_5, B_6.

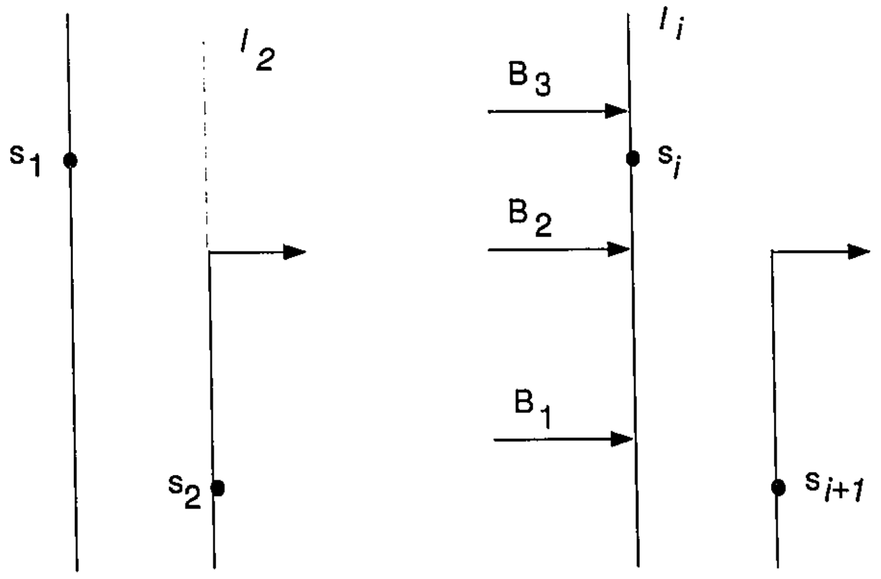


(b) A binary search tree, nodes $\textcircled{5}$ and $\textcircled{6}$ are to be deleted



(c) A binary search tree after adding nodes $\textcircled{5'}$ and $\textcircled{6'}$

Figure 17 The balanced binary search tree T is an empty tree.

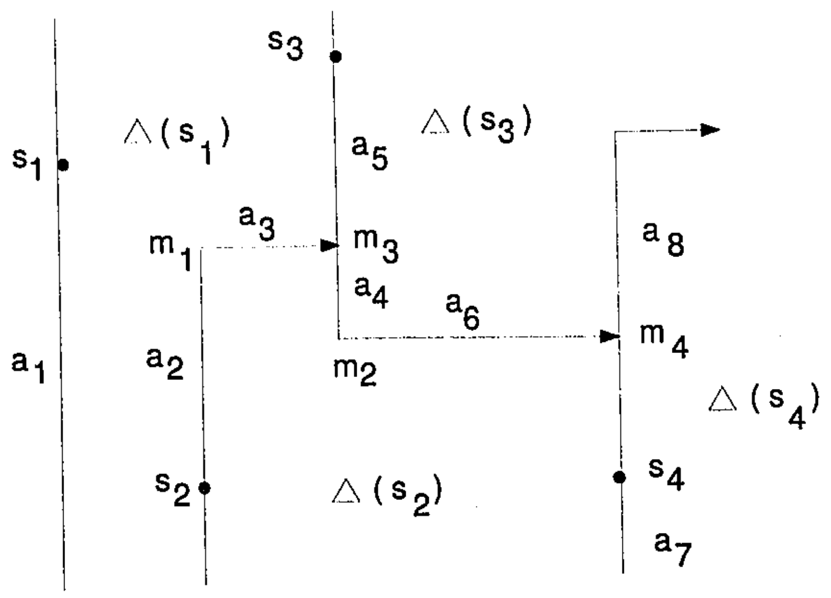


(a) The binary searching tree is empty at the eventpoint s_1

(b) The binary searching tree is empty at the event point s_i

If a h_SPM arc (part of the bisector of two sites) is truncated by $\overline{v_h v_l}$, the ending point of the h_SPM arc (the intersection between the h_SPM arc and the segment $\overline{v_h v_l}$) is fixed. This edge will be saved into the DCEL. In order to find the type 2 Voronoi arcs in each region of the plane (this will be discussed in Section 3) efficiently, edges of $SPM_{\mapsto}(S)$ in DCEL are grouped according to the fields, that is, all the edges of $SPM_{\mapsto}(S)$ enclosing a certain field are grouped together. Figure 18 is an example, field $\Delta(s_1)$ is enclosed by edges *a1*, *a2*, *a3*, and *a5*. Field $\Delta(s_2)$ is enclosed by edges *a2*, *a3*, *a4*, *a6*. *a7*. Note that each edge relates to two fields: left side field (F_L) and right side field (F_R) (except the edges passing through the leftmost site s_1). Each edge (except the leftmost edge and the rightmost edge) will appear twice on the DCEL.

Figure 18 The edges enclosing one field are grouped together.



	v_s	v_e	F_L	F_R	Field	h_SPM	t2_v	v_SPM	site(ptr)
a_1	$-\infty$	∞	0	1	1	0	0	1	1
a_2	$-\infty$	m_1	1	2	1	0	0	1	2
a_3	m_1	m_3	1	2	1	1	0	0	1/2
a_5	m_3	∞	1	3	1	0	0	1	3
a_2	$-\infty$	m_1	1	2	2	0	0	1	2
a_3	m_1	m_3	1	2	2	1	0	0	1/2
a_4	m_2	m_3	2	3	2	0	0	1	3
a_6	m_2	m_4	3	2	2	1	0	0	2/3
a_7	$-\infty$	m_4	2	4	2	0	0	1	4

The entire $SPM_{\mapsto}(S)$ is constructed after the sweep line passes through all the sites from left to right and the corresponding DCEL is built for later use.

The right to left $SPM_{\hookleftarrow}(S)$ can be constructed in the similar way.

Lemma 3.4 Constructing the Shortest–Path–Map from left–to–right $SPM_{\mapsto}(S)$ (right–to-left $SPM_{\hookleftarrow}(S)$ respectively) of n sites can be done in $O(n \log n)$ time.

Proof: The preprocessing to sort the n sites in lexicographical order takes $O(n \log n)$ time. It is easily verified that every edge in $SPM_{\mapsto}(S)$ is (a) the leftmost vertical boundary edge of a region, or (b) the upper boundary originated at a corner, or (c) the lower boundary originated at a corner. Therefore, the total number of edges in $SPM_{\mapsto}(S)$ is at most $3n$. As a result $|T|=O(n)$, and each insertion and deletion operation performed on T takes $O(\log n)$ time. For each site s_i, finding the upper corner and lower corner of its initial region $\Delta(s_i)$ takes $O(\log n + k_i \log n)$ time where k_i is the number of edges truncated. since every bisector can be truncated at most once, and they are $O(n)$ bisectors $\sum_{i=2}^{n} k_i = O(n)$. The total amount of time required to construct $SPM_{\mapsto}(S)$ is thus
$$O\left(nlogn + \sum_{i=2}^{n} k_i logn\right) = O(nlogn).$$

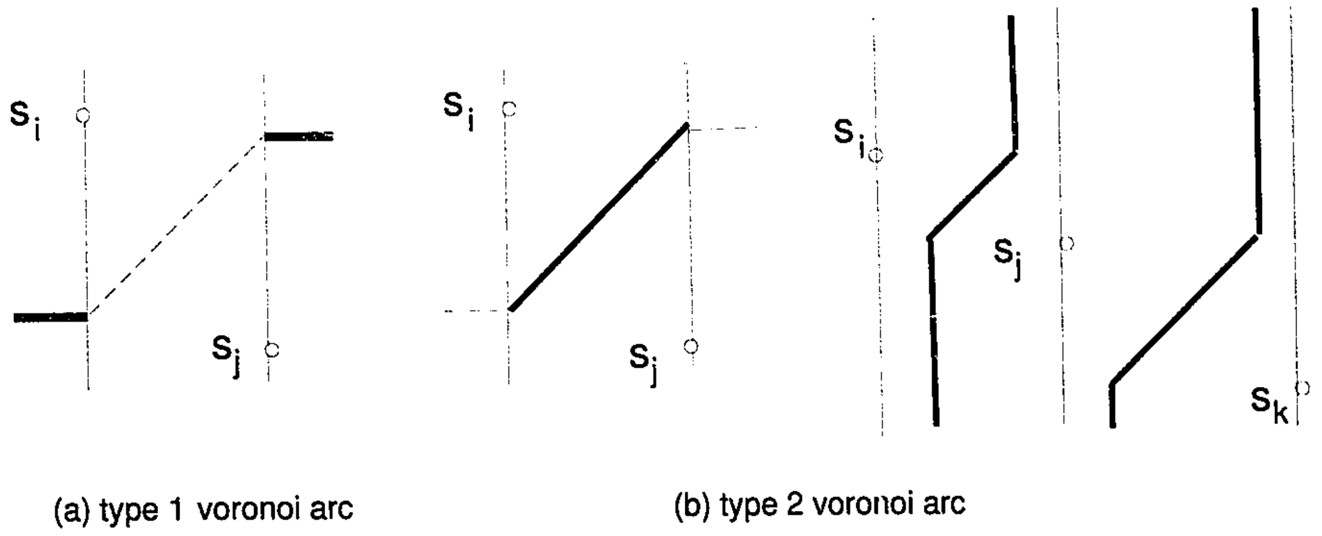
Section 3 Constructing the Voronoi Diagram from two Shortest–Path–Maps

Definition: A type–1 Voronoi arc is a Voronoi arc that lies at the same side of the two sweep lines passing through the two sites which determine the arc. Obviously, every type-1 Voronoi arc is either a h_SPM arc or a section of a h_SPM arc.

A type-2 Voronoi arc is a Voronoi arc that lies between the two sweep lines passing through the two sites determining the arc. We distinguish two cases: (1)

The Voronoi arc intersects the two sweep lines passing through its two sites, (2) The Voronoi arc does not intersect the two sweep lines passing through its two sites.

Figure 19 Type 1 and type 2 Voronoi arcs



Finding the type 1 Voronoi arcs

Definition: $SPM_{\vdash}(S)$ is the opposite SPM of $SPM_{\dashv}(S)$, and vice versa. Entities associated with the opposite SPM are subscribed with *opp*.

Lemma 3.5:

Let edge e be a horizontal SPM arc (h_SPM arc) determined by two sites s_i and s_j. Let u be a point on edge e, $u \in \Delta_{opp}(s_k)$ and $d_1(s_i,u) = d_1(s_j,u) \geq d_1(s_k,u)$. Then for any point p on edge e which is between u and the ending part of e, $p \in \Delta_{opp}(s_k)$ and

$d_1(s_i,p) = d_1(s_j,p) > d_1(s_k,p)$.

Proof: Each site region is an area surrounded by vertical SPM arcs and horizontal SPM arcs. If one point on the horizontal SPM arc is in an opposite region, then every point on this horizontal SPM arc is in that opposite region. Edge e is a horizontal SPM arc, which is a section of the bisector of site s_i and s_j, and u is a point on edge e such that $u \in \Delta_{opp}(s_k)$, and $d_1(s_i,u) = d_1(s_j,u) \geq d_1(s_k,u)$. Clearly, u is equal-distant to sites s_i and s_j. Let p be a point on edge e between u and the ending point of e (that is point g). From figure 20:

$$d_1(s_i,u) = d_1(s_j,u) \geq d_1(s_k,u)$$

$$d_1(s_i,u) = \overline{s_ic} + \overline{cu}$$

$$d_1(s_k,u) = \overline{s_kd} + \overline{du}$$

$$d_1(s_i,p) = d_1(s_i,u) + \overline{cb}$$

$$d_1(s_k,p) = d_1(s_k,u) - \overline{cb} \leq d_1(s_i,u) - \overline{cb}$$

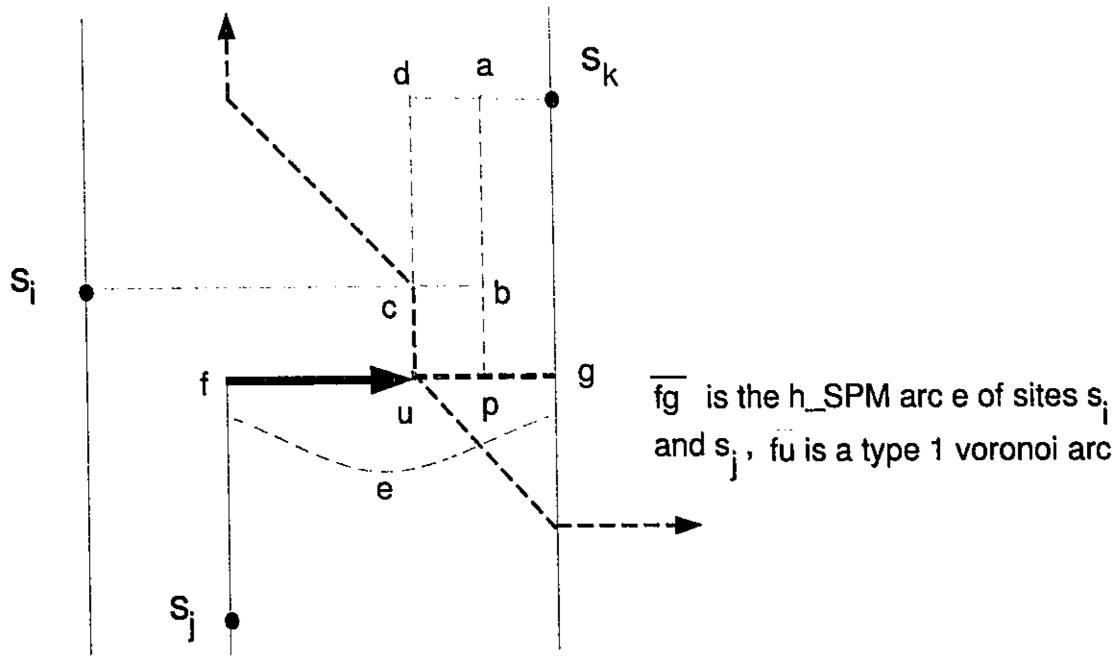
Therefore

$$d_1(s_i,p) > d_1(s_k,p)$$

Consequently

$$d_1(s_i,p) = d_1(s_j,p) > d_1(s_k,p) \ \blacksquare$$

Figure 20 Point u is the centre of sites s_i, s_j, and s_k.



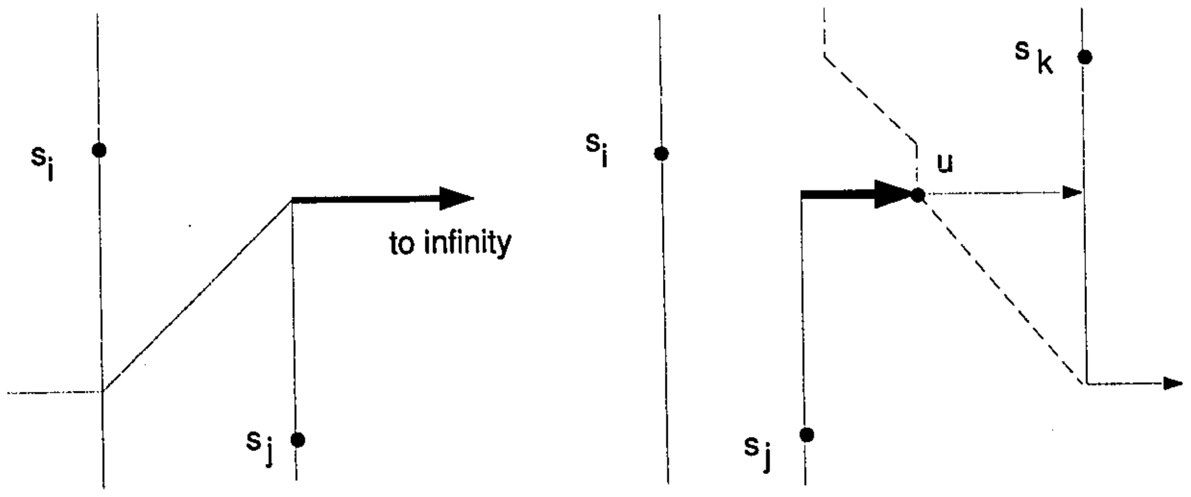
d
a
s_k
s_i
b
c
f
g
u
p
e
s_j
$\overline{fg}$ is the h_SPM arc e of sites s_i and s_j, fu is a type 1 voronoi arc

Lemma 3.6: The starting points of the type-1 Voronoi arc in the SPMs are the corners of the regions. The ending points of the type-1 Voronoi arcs are: (1) at infinity (figure 21(a)); (2) the intersections between h_SPM arcs and type 2 Voronoi arcs (figure 21(b)).

Proof: The first statement follows immediately from the definition of the bisector in L_1–metric (figure 4). If the h_SPM arc of the two sites does not meet the vertical line segment of any site region in the SPM, the ending point of the h_SPM arc must end at infinity (figure 21 (a)). Obviously, the h_SPM arc is a type-1 Voronoi arc.

If the h_SPM arc of two sites s_i, s_j meets a type 2 Voronoi arc at a point u, the type-2 Voronoi arc must be determined by a site s_k and one of s_i and s_j (figure 21 (b)). Clearly, u is equidistant to s_i, s_j and s_k. By *Lemma 3.5*, every point lying between u and the ending point of the h_SPM arc is closer to s_k than s_i and s_j, and every point lying between u and the starting point of the h_SPM arc is closer to s_i and s_j than s_k. Therefore, the section of the h_SPM arc from the starting point to u is a type-1 Voronoi arc ■

Figure 21 Type 1 Voronoi arcs



(a) a type 1 voronoi arc to infinity

(b) a type 1 voronoi arc meets the type 2 voronoi arc (dash line)

Finding the type 2 Voronoi arcs

Lemma 3.7: An edge e is a type-2 Voronoi arc of sites s_i and s_j iff

$$\exists s_i,\ s_i \in S,\ \ e = \Delta(s_i) \cap \Delta_{opp}(s_j) \cap b(s_i, s_j) \quad \neq \emptyset$$

where $b(s_i,\ s_j)$ is the bisector of s_i, and s_j.

Proof:

$\Longrightarrow$) The type–2 Voronoi arc e is a section of the bisector $b(s_i,\ s_j)$ within the area $\Delta(s_i) \cap \Delta_{opp}(s_j)$, therefore

$$e = \Delta(s_i) \cap \Delta_{opp}(s_j) \cap b(s_i, s_j) \quad \neq \emptyset$$

$\Longleftarrow$) Trivial ■

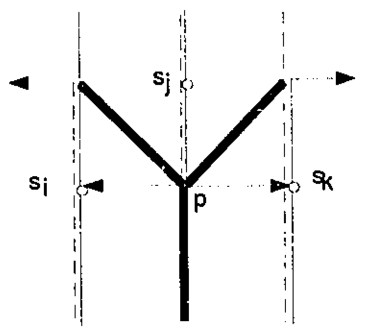
A special case discussion:

Three type-2 Voronoi arcs meet together.

Case 1: Two sites s_i and s_k have the same y-coordinate, and site s_j is on the vertical bisector of site s_i and site s_k, such that $\overline{s_ip} = \overline{s_kp} = \overline{s_jp}$ (figure 22 (a)), or $\overline{s_ip'} = \overline{s_kp'} < \overline{s_jp'}$ (figure 22 (b)). However in (figure 22 (c)) if $\overline{s_ip'} = \overline{s_kp'} > \overline{s_jp'}$, the bisectors of site s_i and site s_j , the bisector of site s_j and site s_k would not meet.

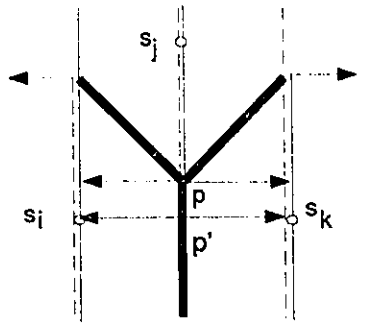
Case 2: The bisector $\overline{ab}$ of site s_i and site s_j, the bisector $\overline{bc}$ of site s_j and site s_k and the bisector $\overline{bd}$ of site s_i and site s_k meet together at point b (figure 22 (d)).

Figure 22 Three type 2 Voronoi arcs meet at one point



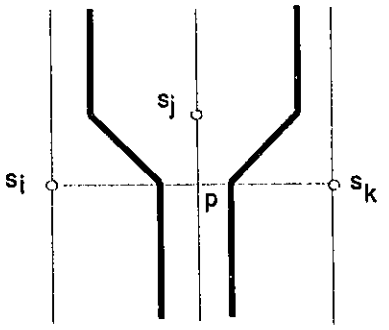
(a) Three type 2 voronoi arcs meet at point p

$\overline{s_i p} = \overline{s_j p} = \overline{s_k p}$



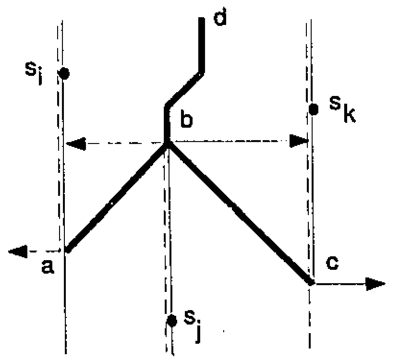
(b) Three type 2 voronoi arcs meet at point p

$\overline{s_i p'} = \overline{s_k p'} < \overline{s_j p'}$



(c) Type 2 voronoi arcs do not meet

$\overline{s_i p'} = \overline{s_k p'} > \overline{s_j p'}$



(d) Three type 2 voronoi arcs meet at point b.

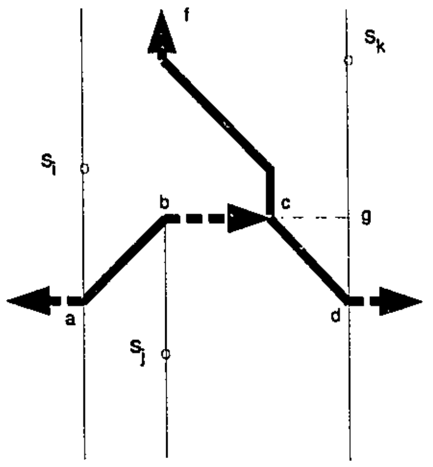
Lemma 3.8: Let e be a type-2 Voronoi arc such that

$e = \Delta(s_i) \cap \Delta_{opp}(s_j) \cap b(s_i, s_j) \neq \emptyset$ for some sites s_i and s_j.

The end points of e are (1) at infinity; or (2) on the corners of the regions $\Delta(s_i)$ or $\Delta_{opp}(s_j)$; or (3) on the horizontal boundary of the region $\Delta(s_i) \cap \Delta_{opp}(s_j)$; or (4) at the intersection of three type 2 Voronoi arcs.

Proof: The proof is trivial, see figure 23 for case (1), (2) and (3). See page 46 and figure 22 (a), (b), (d) for case (4) ■

Figure 23 Type 2 Voronoi arcs



case (1): edge $\overline{fc}$, a type 2 voronoi arc, an end point f at infinity.

case (2): edge $\overline{cd}$, a type 2 voronoi arc, an end point d is on the vertical boundary.

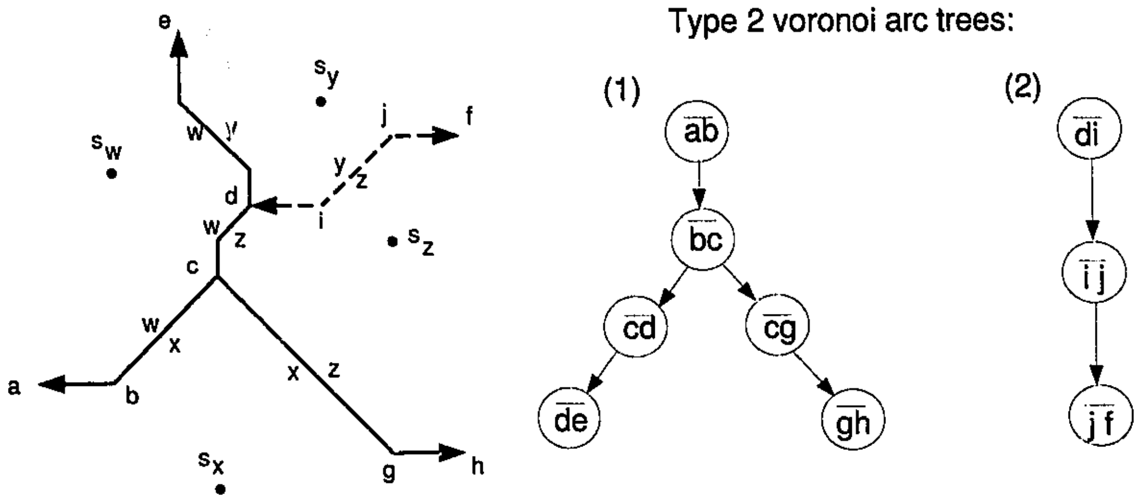
case (3): edge $\overline{fc}$, a type 2 voronoi arc, its end point c on the horizontal SPM arc connecting to another type 2 voronoi arc $\overline{cd}$.

Definition: A type 2 Voronoi arc tree is a tree in which every internal node corresponds to a type 2 Voronoi arc and every leaf corresponds to a type-1 Voronoi arc, or a type-2 Voronoi arc which has an end point at infinity. Moreover, the Voronoi arcs in two adjacent nodes share a common endpoint.

Figure 24 shows a type 2 Voronoi arc tree. It starts from a type 1 Voronoi arc $\overline{ab}$, follows a type 2 Voronoi arcs $\overline{bc}$ and then has two branches. One goes along $\overline{cd}$, and terminates at a type 2 Voronoi arc $\overline{de}$, while the other goes along $\overline{cg}$ and terminates at a type 1 Voronoi arc $\overline{gh}$.

Edges $\overline{di}$, $\overline{ij}$ and $\overline{jf}$ also form a type 2 Voronoi arc tree. It starts from $\overline{di}$ (type 1 Voronoi arc), follows $\overline{ij}$ (type 2 Voronoi arc) and ends at $\overline{jf}$ (type 1 Voronoi arc).

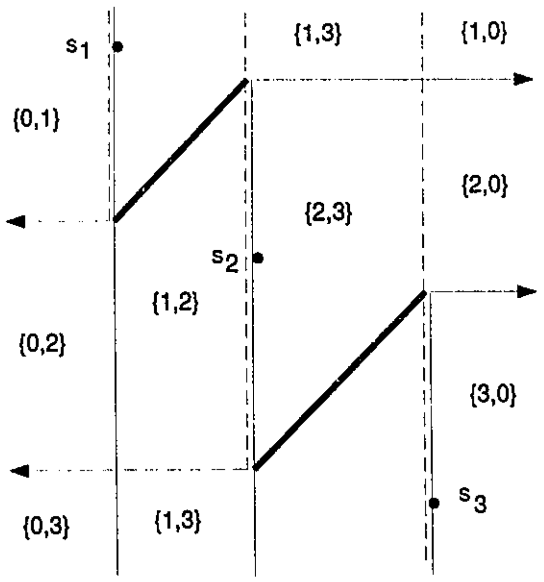
Figure 24 Type 2 Voronoi arc tree.



Section 4 Algorithms

After two sweeps, $SPM_{\mapsto}(S)$ and $SPM_{\mapsfrom}(S)$ are constructed. An area $\Delta(s_i) \cap \Delta_{opp}(s_j)$, $s_i, s_j \in S$, denoted by {i, j} is the intersection (intersections), (Figure 25 {1, 3}) between a left to right sweep region $\Delta(s_i)$ and right to left sweep region $\Delta_{opp}(s_j)$. Site s_i is the closest left site of every point inside area {i, j}, and site s_j is the closest right site of every point insid area {i, j}. An area {i, j} may be a closed area enclosed by the SPM arcs, or with some edges open at infinity. The areas located at the left side of the sweep line passing through the leftmost site s_1 belong to only one region, that is area {0, 1}. The area located at the right side of a sweep line passing through the right most site s_n belong to only one region, that is area {n, 0}. Obviously, there is no type 2 Voronoi arc inside the areas {0, 1} or {n, 0}, which has only one closest site s_1 or s_n. The horizontal boundaries of the one site area {0, 1} or {n, 0} are type 1 Voronoi arcs. Note that some areas {i, j} do not have the type 2 Voronoi arc inside, the bisector of site s_i and s_j do not cross through these two site areas, (figure 25).

Figure 25 Areas $\Delta(s_1) \cap \Delta_{opp}(s_3)$ represented by $\{1, 3\}$ have no type 2 Voronoi arcs inside.



Viewing the plane divided by the two SPMs as a collection of areas $\Delta(s_i) \cap \Delta_{opp}(s_j)$, $s_i, s_j \in S$ will help us in finding the type 1 and type 2 Voronoi arcs.

Three DCELs are used to store the information when tracing the type 2 Voronoi arc tree:

D_{st}: stores the h_SPM arcs which have one ending point at infinity and the type-2 Voronoi arcs which have one ending point at infinity as the starting edges to trace the type 2 Voronoi arc trees. The type-2 Voronoi arcs which are starting edges at the branch of the type–2 Voronoi arc tree are also stored in D_{st} (figure 24, $\overline{cg}$ or $\overline{cd}$).

D_{end}: stores the h_SPM arcs and type 2 Voronoi arcs which terminate the type 2 Voronoi arc trees.

D_v: stores all the edges of the Voronoi diagram.

After the left to right sweep and the right to left sweep, we obtain two shortest–path–maps: $SPM_{\mapsto}(S)$ and $SPM_{\leftarrow\!|}(S)$, which are represented by Doubly-Connected-Edge-List (DCEL). All h_SPM arcs whose one ending point is at infinity and all type-2 Voronoi arcs whose one ending point is at infinity (extreme edges) are copied to two DCEL structures: D_{st} (D_{st} is a DCEL data structure for storing the extreme edges for tracing type 2 Voronoi arc trees), and D_v (D_v is a DCEL data structure for storing the Voronoi diagram).

Pick up the extreme edges one by one from D_{st}, trace back into the areas one after another to find out all the type 2 Voronoi arcs in the tree (this will be discussed later). The leaf edge of a type 2 Voronoi arc tree is a type 1 Voronoi arc which follows a type 2 Voronoi arc ended at the vertical boundary of the area, or a type 2 Voronoi arc which ended at infinity. The intersections between type 2 Voronoi arcs and h_SPM arcs (an edge on the boundary of the area in which the type 2 Voronoi arc locates) are the heads of type 1 Voronoi arcs. These type 1 Voronoi arcs are also roots of type 2 Voronoi arc trees. Store these type 1 Voronoi arcs into D_{st} (DCEL) for later tracing. In cases where three type 2 Voronoi arcs meet together at the corners of the areas (figure 22), pick one type 2 Voronoi arc for continuous tracing, store the other type 2 Voronoi arc into D_{st} (DCEL) for later tracing. Pick every edge stored in the D_{st} (DCEL) and trace out its type 2 Voronoi arc tree until all the edges are traced. Store every type 1 Voronoi arc and type 2 Voronoi arc of the type 2 Voronoi arc tree into D_v during the tracing. D_v is the required DCEL storing the Voronoi diagrams.

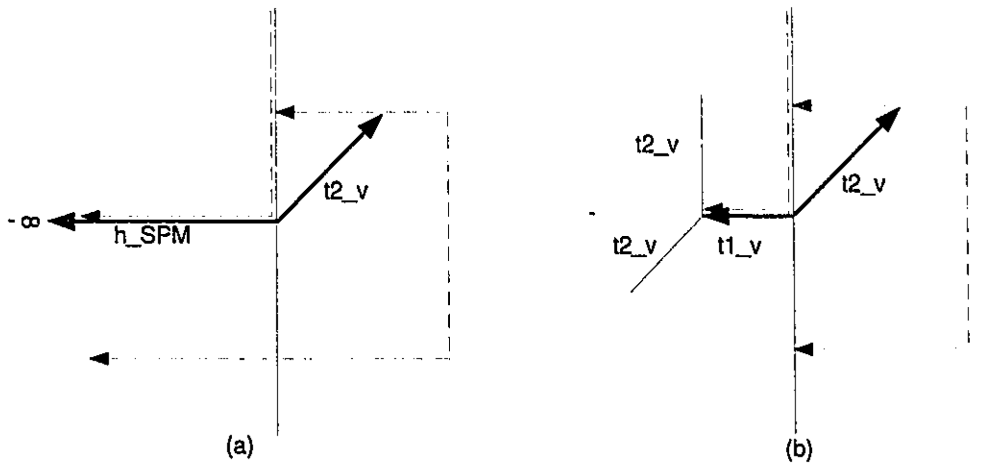
We can easily find out whether there is a type 2 Voronoi arc lying inside an area, and find out the intersections between the type 2 Voronoi arc and the enclosing edges of the area {i, j} by searching under the fields $\Delta(s_i)$ and $\Delta_{opp}(s_j)$ of DCEL which stores all the information of $SPM_{\vdash}(S)$ and $SPM_{\dashv}(S)$.

To trace back all the type 2 Voronoi arc trees, six different cases are discussed below:

Case 1: (figure 26) Pick up an edge from D_{st}. Regardless of whether it is a h_SPM arc with its ending point at infinity or a type 1 Voronoi arc, its starting point is on a vertical SPM arc (v_SPM) and will be followed by a type 2 Voronoi arc. The area in which the following type 2 Voronoi arc locates is the intersection of two fields which are the same as the left field and the right field of the previous h_SPM arc.

When the area in which the type 2 Voronoi arc locates is known, that means the two fields are known . Get the grouped edges of field $\Delta_{opp}(s_j)$ of the area from the DCEL, compute the type 2 Voronoi arc which is inside of the area, the starting point of the type 2 Voronoi arc is the same as the starting point of the h_SPM arc (or the type 1 Voronoi arc). The ending point of the type 2 Voronoi arc is the intersection between the type 2 Voronoi arc and one of the edges of field $\Delta_{opp}(s_j)$ of the area.

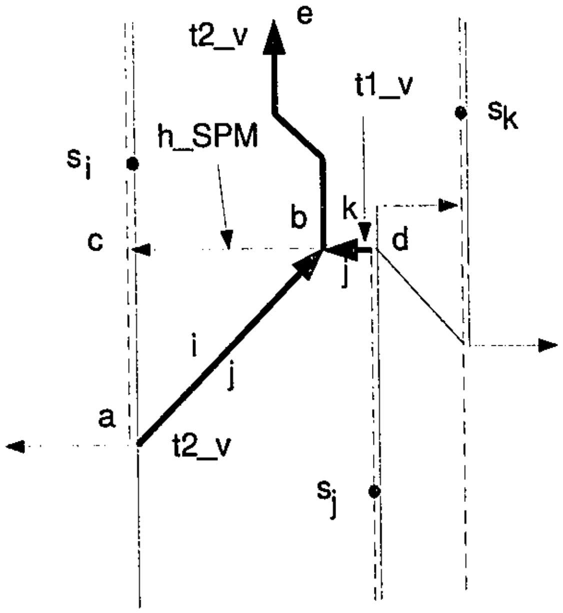
Figure 26 Start tracing back the type 2 Voronoi arc tree from (a) a h_SPM arc with the ending point at infinity, or from (b) a type 1 Voronoi arc.



Case 2: (figure 27) An ending point of a type 2 Voronoi arc $\overline{ab}$ is on a h_SPM arc $\overline{dc}$ which is the boundary of the opposite field $\Delta_{opp}(s_j)$ of the area in which the type 2 Voronoi arc resides. The left (right) side field of type 2 Voronoi arc $\overline{ab}$ is $\Delta(s_i)$ ($\Delta(s_j)$). The left (right) side of h_SPM arc $\overline{dc}$ is $\Delta(s_j)$ ($\Delta(s_k)$). So the next type 2 Voronoi arc $\overline{be}$ is in the area {i, k}. The processing is: compute the type 2 Voronoi arc $\overline{ab}$ in area {i, j}, get the ending point *b* of the type 2 Voronoi arc $\overline{ab}$ (*b* is on a h_SPM arc which is the boundary of the opposite field $\Delta_{opp}(s_j)$

of area {i, j}), get the type 1 Voronoi arc $\overline{db}$, check if the h_SPM arc $\overline{dc}$ is in the D_{end} (DCEL). If $\overline{dc}$ is inside D_{end}, delete $\overline{dc}$ from D_{end}, and save type 1 Voronoi arc $\overline{db}$ into D_v. If $\overline{dc}$ is not in the D_{end}, store $\overline{db}$ into D_{st} (for later tracing) and D_v.

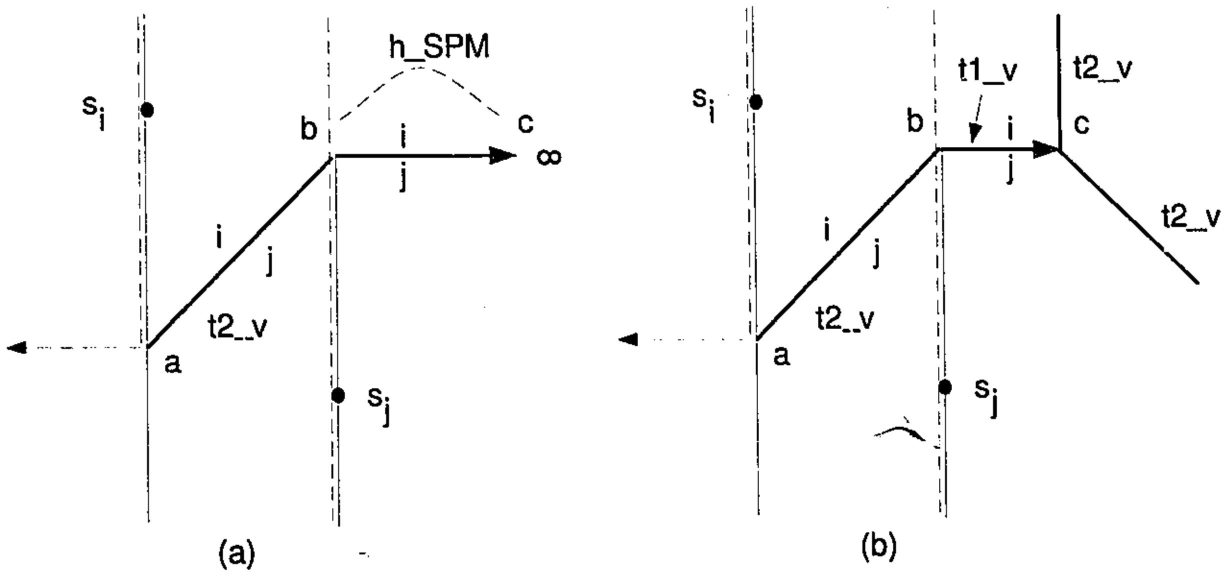
Figure 27 Tracing back from a type 2 Voronoi arc to another type 2 Voronoi arc, and meet a type 1 Voronoi arc as a root of a new type 2 Voronoi arc tree.



Case 3: (figure 28 (a)) At the time of tracing back along the type 2 Voronoi arc tree, if the ending point b of a type 2 Voronoi arc $\overline{ab}$ is on the vertical boundary of the area in which the type 2 Voronoi arc locates, the next edge is a h_SPM arc. The left side field and the right side field of this h_SPM arc $\overline{bc}$ is the same as the left side field and the right side field of $\overline{ab}$.

If the ending point of the h_SPM arc $\overline{bc}$ is at infinity, $\overline{bc}$ must already exist in D_{st}, then delete it from D_{st}.

Figure 28 Tracing back from a type 2 Voronoi arc to (a) a h_SPM arc with the ending point at infinity. (b) a type 1 Voronoi arc which is met by other type 2 Voronoi arc tree.



Case 4: (figure 28(b)) If the ending point b of a type 2 Voronoi arc $\overline{ab}$ is on the vertical boundary of the area in which the type 2 Voronoi arc locates, the next edge is a h_SPM arc $\overline{bc}$. The left side field and right side field of $\overline{bc}$ is the same as the left side field and right side field of $\overline{ab}$. Check whether this h_SPM arc $\overline{bc}$ is inside the D_{st} (DCEL): (1) If it exists in the D_{st} already, it means this h_SPM arc was intersected by another type 2 Voronoi arc tree, and $\overline{bc}$ was saved into D_{st} before the current tracing of type 2 Voronoi arc tree along $\overline{ab}$ and $\overline{bc}$. Delete it

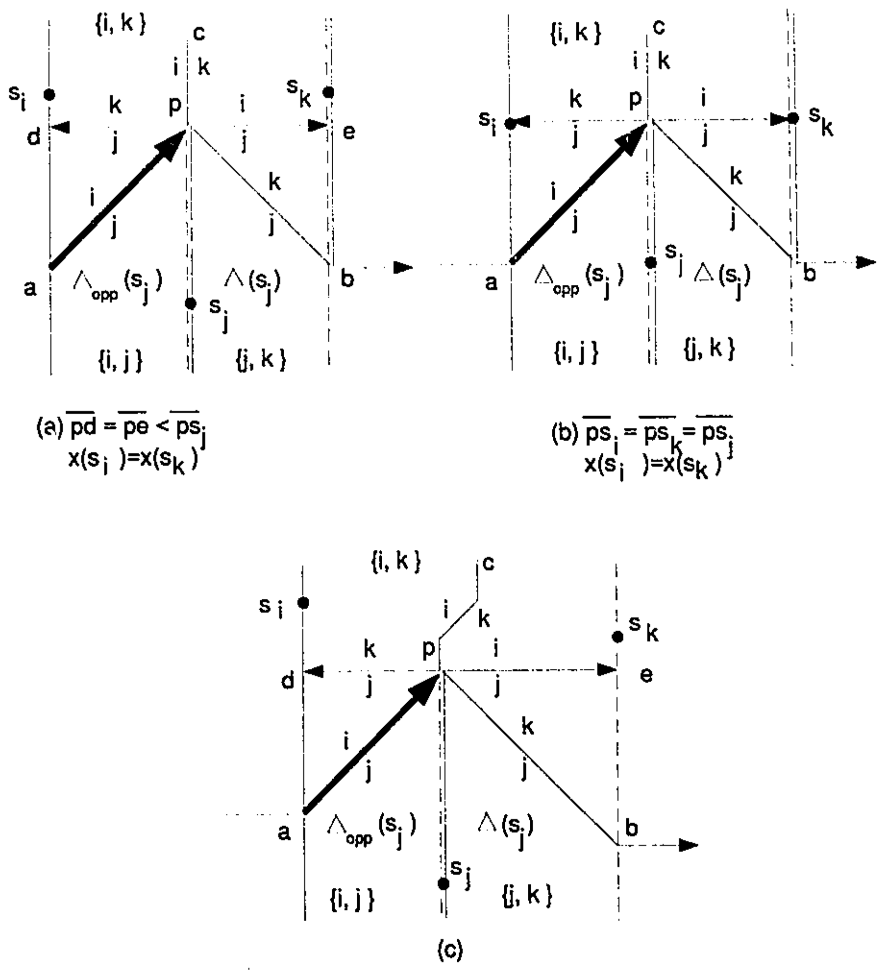
from D_{st}, because this type 2 Voronoi arc tree has been traced by now. (2) If it doesn't exist in the D_{st}, save it in the D_{end}, it indicates that the type 2 Voronoi arc tree starting from this h_SPM arc $\overline{bc}$ needs not be traced again. When the h_SPM arc $\overline{bc}$ is intersected by the other type 2 Voronoi arc tree at point c later on, whether the type 2 Voronoi arc tree started from $\overline{bc}$ has been traced or not can be known by checking if $\overline{bc}$ is inside D_{end} or not.

Lemma 3.9 Let s_i, s_j and s_k be three sites in the plane. If the end point of a type 2 Voronoi arc of the area {i, j} is at the corner p of the area {i, j}, then the corner p must also be the corner of adjacent area {j, k}, where site s_j is on the boundary of both area {i, j} and {j, k}. If two sites s_i, s_k have the same x-coordinates, the three sites s_i, s_j and s_k are vertices of an isosceles triangle.

Proof: (figure 29) Let p be the corner of area {i, j}, $\overline{ap}$ be a type 2 Voronoi arc of area {i, j}, $\overline{pd}$ be a h_SPM arc as part of the boundary of area {i, j}, so that the starting point a of $\overline{ap}$ is on the vertical line passing through site s_i, the ending point p of $\overline{ap}$ is on the vertical line passing through site s_j. According to the method of constructing the SPM, the edges $\overline{ps_j}$ and $\overline{pe}$ are part of the boundaries of field $\Delta(s_j)$. From lemma 3.6, $\overline{pe}$ is a h_SPM arc starting from the corner p of region $\Delta(s_j)$ with left side field $\Delta(s_i)$ and right side field $\Delta(s_j)$. According to the method of constructing the SPM, $\overline{pd}$ and $\overline{ps_j}$ must be part of the boundaries of field $\Delta_{opp}(s_j)$. Edge $\overline{pd}$ (a h_SPM arc) is part of the bisector of site s_j and the other site, say s_k. The left side field of $\overline{pd}$ is field $\Delta_{opp}(s_j)$, the right side field of

$\overline{pd}$ is field $\Delta_{opp}(s_k)$. $\overline{pe}$ and $\overline{pd}$ are both h_SPM arcs, and belonging to field $\Delta(s_j)$ and field $\Delta_{opp}(s_j)$ respectively, so that corner p is the centre of three sites s_i, s_j and s_k. If site s_i and site s_k have the same x-coordinates, then s_i, s_j and s_k are three vertices of an isosceles triangle, and $|x(s_i) - x(s_j)| = |x(s_j) - x(s_k)|$, $y(s_i) = y(s_k)$, (figure 29 (a) (b)). If site s_i and site s_k have different x-coordinates, then the situation is depicted in figure 29 (c) ■

Figure 29 Three type 2 Voronoi arcs meet together.



(a) $\overline{pd} = \overline{pe} < \overline{ps_j}$
$x(s_i) = x(s_k)$

(b) $\overline{ps_i} = \overline{ps_k} = \overline{ps_j}$
$x(s_i) = x(s_k)$

(c)

Case 5: (figure 29(a)) If the ending point of a type 2 Voronoi arc $\overline{ap}$ is on the corner of the area {i, j} in which the type 2 Voronoi arc $\overline{ap}$ resides. From *lemma 3.9*, *p* is equidistant to the three sites. Check from D_{st} if the edge $\overline{ap}$ is already recorded inside, that is, check if two type 2 Voronoi arcs $\overline{bp}$ and $\overline{pc}$ are traced in the other type 2 Voronoi arc tree already. If it is inside D_{st}, delete edge $\overline{ap}$ from

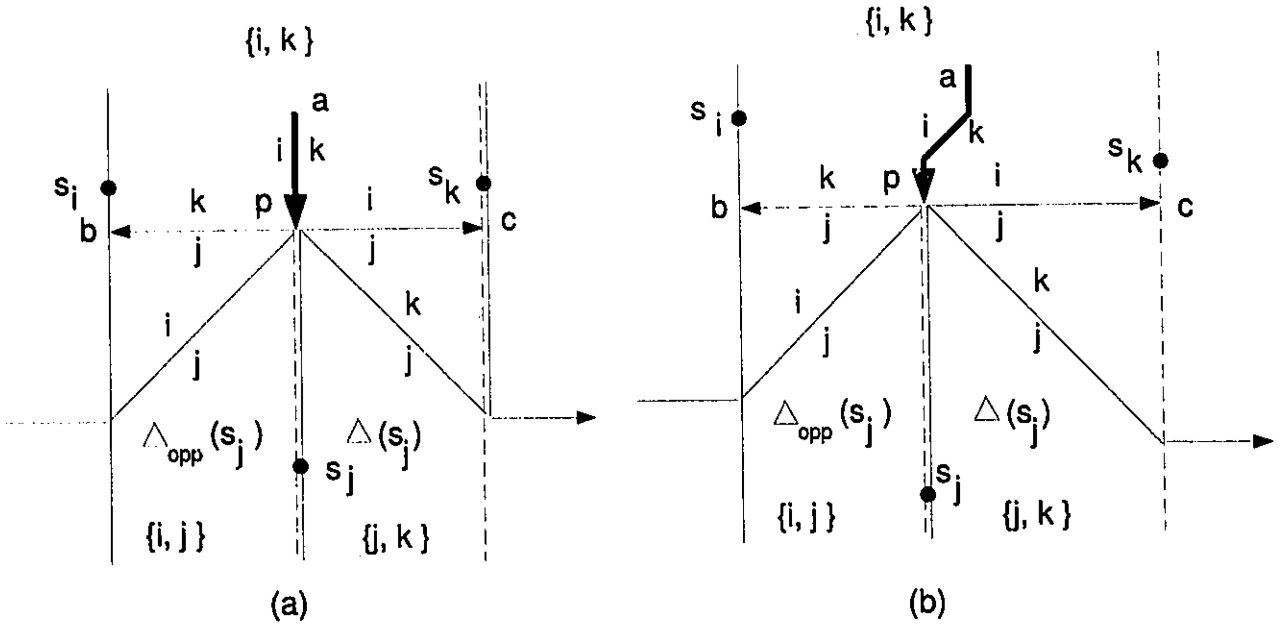
D_{st}. Otherwisw, do the following: get the h_SPM arc $\overline{pd}$, find out the left and right field of $\overline{pd}$, that is $\Delta_{opp}(s_j)$ and $\Delta_{opp}(s_k)$. Then the neighboring area of area {i, j} will be area {j, k} and area {i, k}. Compute the type 2 Voronoi arcs in both area {j, k} and area {i, k}, save one type 2 Voronoi arc in D_{st} for later tracing, and trace back along another type 2 Voronoi arc.

Figure 29 (b) is a special situation of *case 5* (figure 29 (a)), where $\overline{s_i p} = \overline{s_j p} = \overline{s_k p}$.

Case 6: If the two sites s_i and s_k of an area {i, k} have the same y-coordinate, (figure 30 (a)) the bisector is a straight line (figure 4 (h)). Let $\overline{ap}$ be a type 2 Voronoi arc of area {i, k}. If the ending point p of $\overline{ap}$ is the position where two h_SPM arcs $\overline{pb}$, $\overline{pc}$ join (apparently, these two h_SPM arcs are on a straight line). This is the same situation as in case 5, that three sites , s_i, s_j and s_k are vertices of a isosceles triangle. Check from D_{st} (DCEL), if this edge is already recorded inside, that is, check if the other two type 2 Voronoi arcs are already traced in a type 2 Voronoi arc tree. If it is inside D_{st}, delete edge $\overline{ap}$ from D_{st} (DCEL). Get the h_SPM arc $\overline{pb}$ with the left (right) side field $\Delta_{opp}(s_j)$ ($\Delta_{opp}(s_k)$), get the h_SPM arc $\overline{pc}$ with the left (right) side field $\Delta(s_i)$ ($\Delta(s_j)$). Determine the two adjacent areas, {i, j} and {j, k}. Compute their type 2 Voronoi arcs, save one type 2 Voronoi arc in D_{st} for later tracing, and trace back along another type 2 Voronoi arc.

The same method can be applied in Figure 30 (b). The only difference is that the two sites s_i and s_k of an area {i, k} have different y-coordinates, and their bisector is not a straight line.

Figure 30 Three type 2 Voronoi arcs meet together. Tracing type 2 Voronoi arc trees from edge $\overline{ap}$



Theorem 3.10

The Voronoi diagram with n sites in L_1 metric can be constructed in $O(n\log n)$ time and $O(n)$ space.

Proof: From lemma 3.4, the construction of the Shortest–Path–Maps, $SPM_{\mapsto}(S)$ and $SPM_{\leftarrow\!\dashv}(S)$ takes $O(n\log n)$ time. After the left–to–right sweep and right–to–left sweep, all the edges of SPMs are grouped into fields and stored in the Doubly-Connected-Edge-List (DCEL).

Tracing the type 1 Voronoi arcs and type 2 Voronoi arcs to form the Voronoi diagram starts from: (1) the h_SPM arcs with the ending points at infinity, or the type-2 Voronoi arc with one ending point at infinity (these h_SPM arcs and type 2 Voronoi arcs are saved in D_{st} (DCEL) during the two sweeps), or from (2) each newly found type 1 Voronoi arc (found at the time of tracing type 2 Voronoi arc trees and is treated as a starting edge of a type 2 Voronoi arc tree, or a terminating edge of a type 2 Voronoi arc tree), or from (3) each newly found type 2 Voronoi arc (three type 2 arcs meet together), and then tracing into the areas one after another.

Picking up the starting arc of a Voronoi arc tree from D_{st} takes $O(1)$ time.

Fetching the SPM arcs which enclose an area from DCEL (an area is enclosed by a group of SPM arcs) takes $O(1)$ time.

The total number of SPM arcs in the plane is $O(n)$.

Computing the intersections in each area between a type-2 Voronoi arc and the enclosing SPM arcs takes $O(n_k)$ time where n_k is the number of enclosing SPM arcs. As every SPM arc is used a constant number of time in this way and $\sum n_k = O(n)$. Finding the intersections for all the areas takes $O(n)$.

The total number of type 1 Voronoi arcs and type 2 Voronoi arcs in the Voronoi diagram is $O(n)$. Therefore, the total time required to trace out all the type 1 and type 2 Voronoi arcs takes $O(n)$ time. Hence, the total amount of time required to construct the Voronoi diagrams of a set of n site in L_1 metric is $O(n\log n)$ ■

Chapter 4 Conclusions

We presented an *O(n*log*n)* time and *O*(*n*) space algorithm for the Voronoi diagram of a set of *n* sites in L_1 metrtic using a *two plane sweep* method. The two sweeps produce two data structures, called the left–to–right Shortest–Path–Map ($SPM_{\mapsto}(S)$) and the right–to–left Shortest–Path–Map ($SPM_{\leftarrow\!\dashv}(S)$). Each of the maps partitions the plane into regions. They contain enough information to determine the left closest site and right closest site of each point in the plane. Then by tailoring the two maps with chains of Voronoi adges, the Voronoi diagrams is constructed.

A method for storing the SPM arcs in groups according to the fields they belong in the DCEL data structure is used to speed up the process.

A balanced binary research tree (T) is adopted for efficiently updating the status of the sweepline.

It is interesting to investigate if this technique can be applied to Voronoi diagrams in other metrics.

APPENDIX 1 An Algorithm for Constructing the Left to Right Shortest Path Map ($SPM_{\mapsto}(S)$)

```
while (queue is not empty) do
    • dequeue the head element $s_i$. /* the sweep line is positioned at event point
    $s_i$ */
    • Determine the upper boundary $v_h$ of the initial region $\Delta(s_i)$ on the sweep
    line $l_i$ using a balanced binary search tree ($T$).
    if ($v_h$ is at infinity) then
        There is no horizontal bisector above site $s_i$.
        Compute the type-2 Voronoi arc with one ending point at infinity, and
        save it in $D_{st}$.
    else $v_h$ is not at infinity
        Computes the bisector $B_h(s_i)$ that starts from $v_h$ and ends at $+\infty$.
        Insert $B_h(s_i)$ into T.
    endif
    • Determine the lower boundary $v_l$ of the initial region $\Delta(s_i)$ on the sweep
    line $l_i$ by a balanced binary search tree ($T$).
    if ($v_l$ is at infinity) then
        There is no horizontal bisector below site $s_i$.
        Compute the type-2 Voronoi arc with one ending point at infinity, and
        save it in $D_{st}$.
    else $v_l$ is not at infinity
        Computes the bisector $B_l(s_i)$ that starts from $v_l$ and ends at $+\infty$.
        Insert $B_l(s_i)$ into T
    endif
    • The bisectors truncated by the segment $\overline{v_h v_l}$ are deleted from the tree (T).
    • Save the segment $\overline{v_h v_l}$ (including $v_h$, $v_l$ at infinity) into two DCEL groups
    according to their left side and right side field.
    • Save each of the bisectors truncated by the segments $\overline{v_h v_l}$ into the DCEL
    field groups according to their left side field and right side field.
    if ($s_i$ is not the last element in the ST) then
        loop back
    else
        Save each of the bisectors (their ending point are at $+\infty$ in the tree (T)
        into two DCEL groups according to their left side field and right side field.
        Save them into $D_{st}$ (DCEL). /* starting edges for tracing the type 2
        Voronoi arc trees */
    endif
```

```
endwhile
```

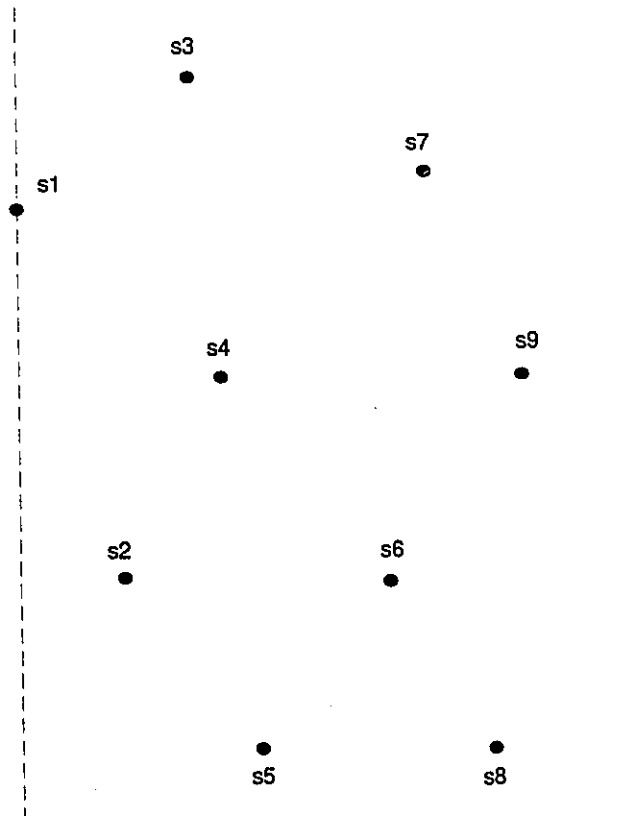
APPENDIX 2 An Algorithm for Constructing the Voronoi Diagram in L_1 Metric

```
while ( $D_{st}$ is not empty) do

Step 1:

    Dequeue $e_{st}$ from $D_{st}$ (DCEL) /* $e_{st}$ is an element in $D_{st}$, a starting edge for
    tracing a type 2 Voronoi arc tree */

    if ($e_{st}$ is a type 1 Voronoi arc) then

        /* see figure 26 regardless of whatever the ending point is at infinity */

        The next area {L, R} where the next edge lies within is determined by
        the left side field $F_L$ and the right side field $F_R$ of $e_{st}$.

    else $e_{st}$ is a type 2 Voronoi arc

        /* that is the situation where three edges meet at a point (see Lemma
        3.9). The left side field $F_L$ and right side field $F_R$, that is, the area {L,
        R} is known */

    endif

Step 2:

    if (the bisector of the two sites of the area is a vertical line segment (i.e.
    the y-coordinates of the two sites of the area are the same) (figure 4 (h)), or
the bisector does not intersect the vertical line passing through the two sites
(figure 4 (a) (b)), **then**

    Get all the edges of the two field groups from $D_{L\_R}$ (DCEL) and $D_{R\_L}$
    (DCEL)
    Compute the type 2 Voronoi arc in the current area.
    Compute the intersection between the type 2 Voronoi arc and edges of
    the two field groups from $D_{L\_R}$ (DCEL) and $D_{R\_L}$ (DCEL)
    Save the type 2 Voronoi arc in $D_v$.

**else** /* the bisector of the two sites of the area intersects the vertical line
passing through two sites (figure 4 (c) (d)) */

    Get the edges of one field group from $D_{L\_R}$ (DCEL) or $D_{R\_L}$ (DCEL)
    which opposites the direction of tracing along the type 2 Voronoi arc.
    Compute the type 2 Voronoi arc in the current area.
    Compute the intersection between the type 2 Voronoi arc and edges of
    one field group which opposites the direction of tracing along the type
    2 Voronoi arc.

**endif**
**Case 1**: the type 2 Voronoi arc meets the h_SPM arc of $\Delta_{opp}(s_j)$, (case 2 in
*Algorithm* and figure 27),
Get the type 1 Voronoi arc which is part of the h_SPM arc and the next area
in which the next type 2 Voronoi arc lies inside.
    **if** (type 1 Voronoi arc is in $D_{end}$ (DCEL)) **then**
        Delete type 1 Voronoi arc from $D_{end}$ (DCEL).
    **else** save it in $D_{st}$ and $D_v$.
    **endif**
    Trace the next type 2 Voronoi arc, go back to the beginning of STEP 2.
**endcase 1**


**Case 2**: the type 2 Voronoi arc meets the v_SPM arc of $\Delta_{opp}(s_j)$. (see case
3 in *Algorithm* and figure 28), the next edge is a type 1 Voronoi arc.
    **if** (the type 1 Voronoi arc is in the $D_{st}$) **then**
        delete it from $D_{st}$
    **else** save it into $D_{end}$
    **endif**
    Go back to the beginning of **while do**.
**endcase 2**
```

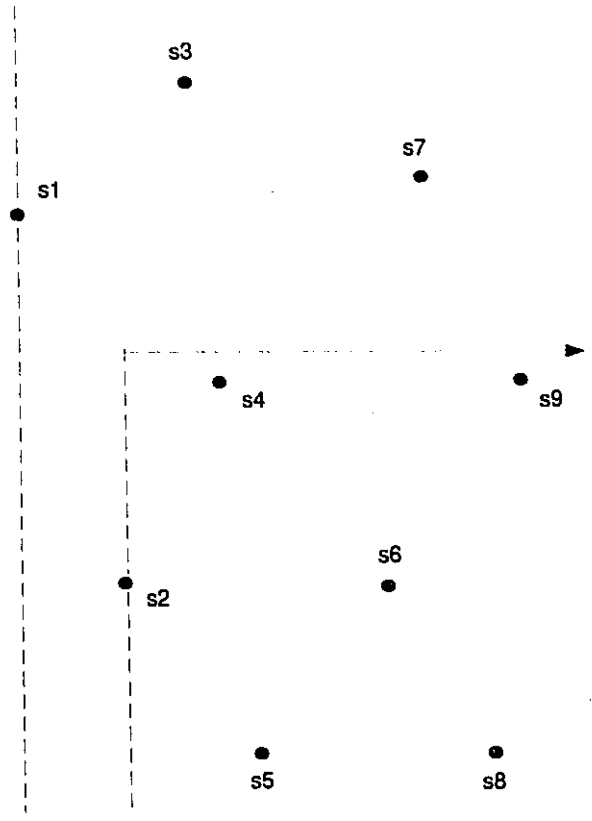
```
Case 3: the type 2 Voronoi arc meets at the corner of a site field, (cases 5,
6 in algorithmLemma 3.9 and figures 29, 30), the corner is the intersection
of three type 2 Voronoi arcs.
    if (this type 2 Voronoi arc is in $D_{st}$) then
        delete it from $D_{st}$ /* the other two type 2 Voronoi arc was traced
        before */
        go back to beginning of while do.
    else save one type 2 Voronoi arc into $D_{st}$ (and $D_v$), trace the other type
    2 Voronoi arc.
    endif
  Go back to Step 2.
  endcase 3
endwhile
```

APPENDIX 3 An Example of Constructing the Voronoi Diagram in L_1 Metric

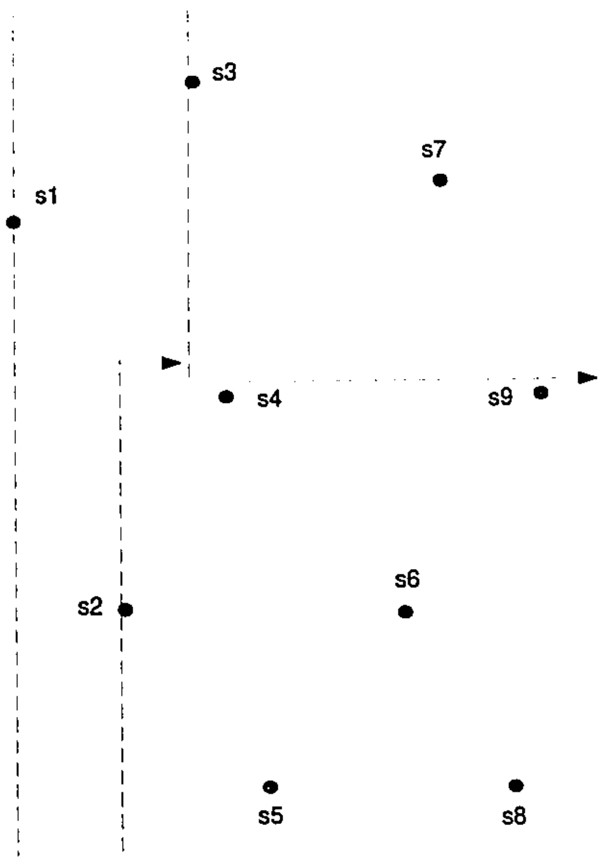
The example below illustrates how the left to right sweep and the right to left sweep work, how the *areas* are formed by the two sweeps, and how the Voronoi arcs are constructed.



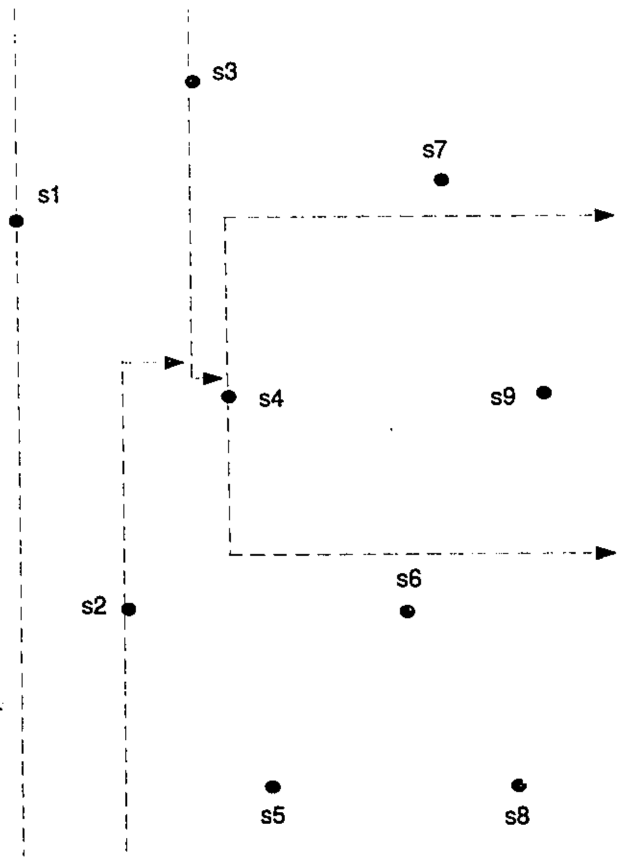
(a) Sweep line at site s1



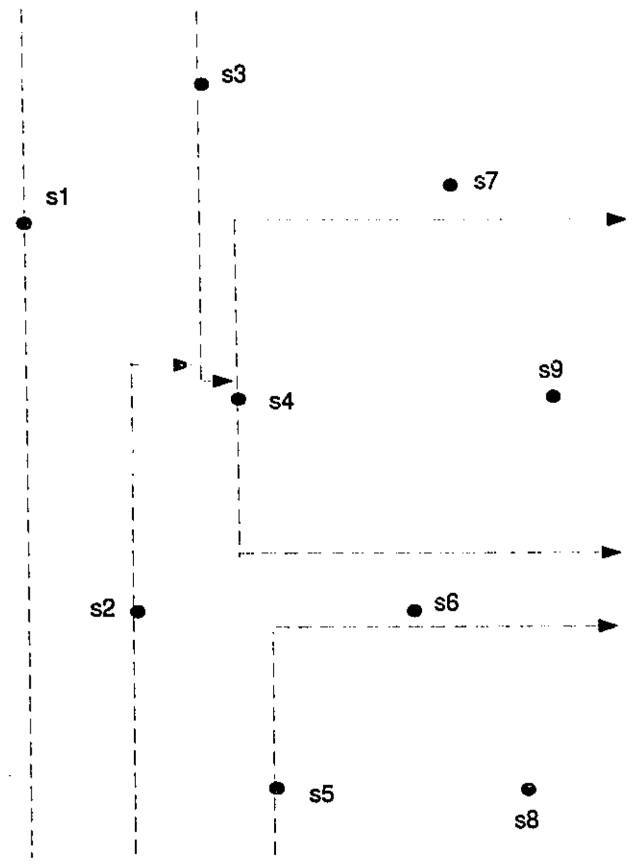
(b) Sweep line at site s2



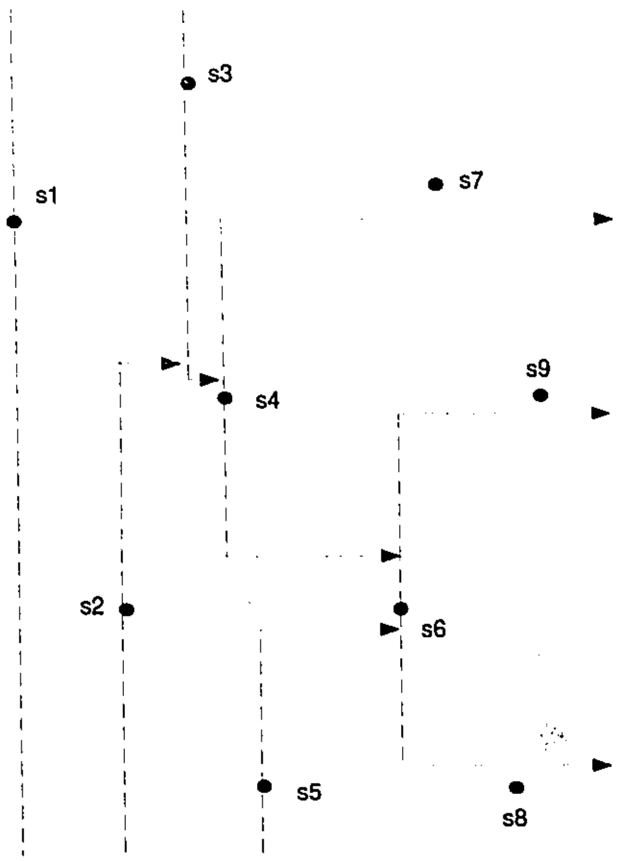
(a) Sweep line at site s3



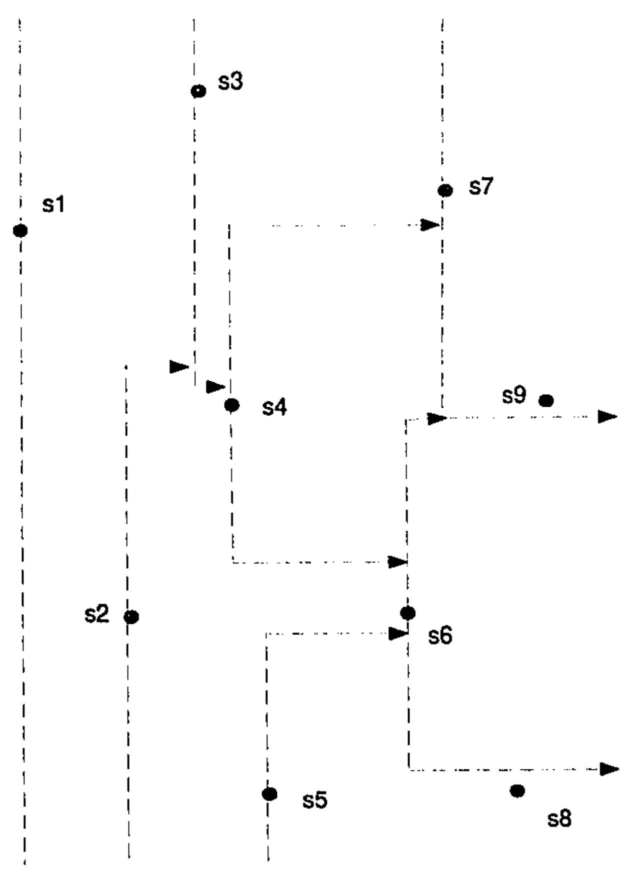
(b) Sweep line at site s4



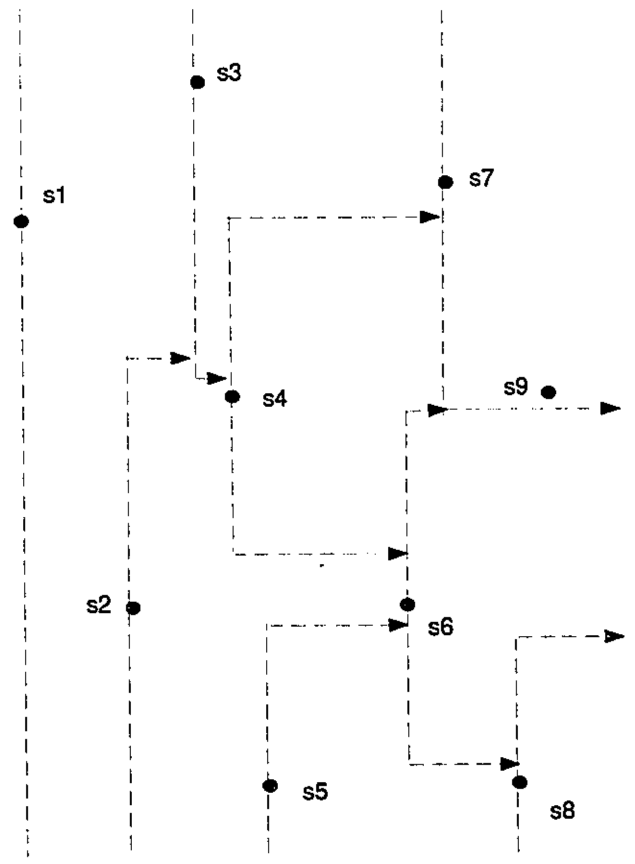
(a) Sweep line at site s5



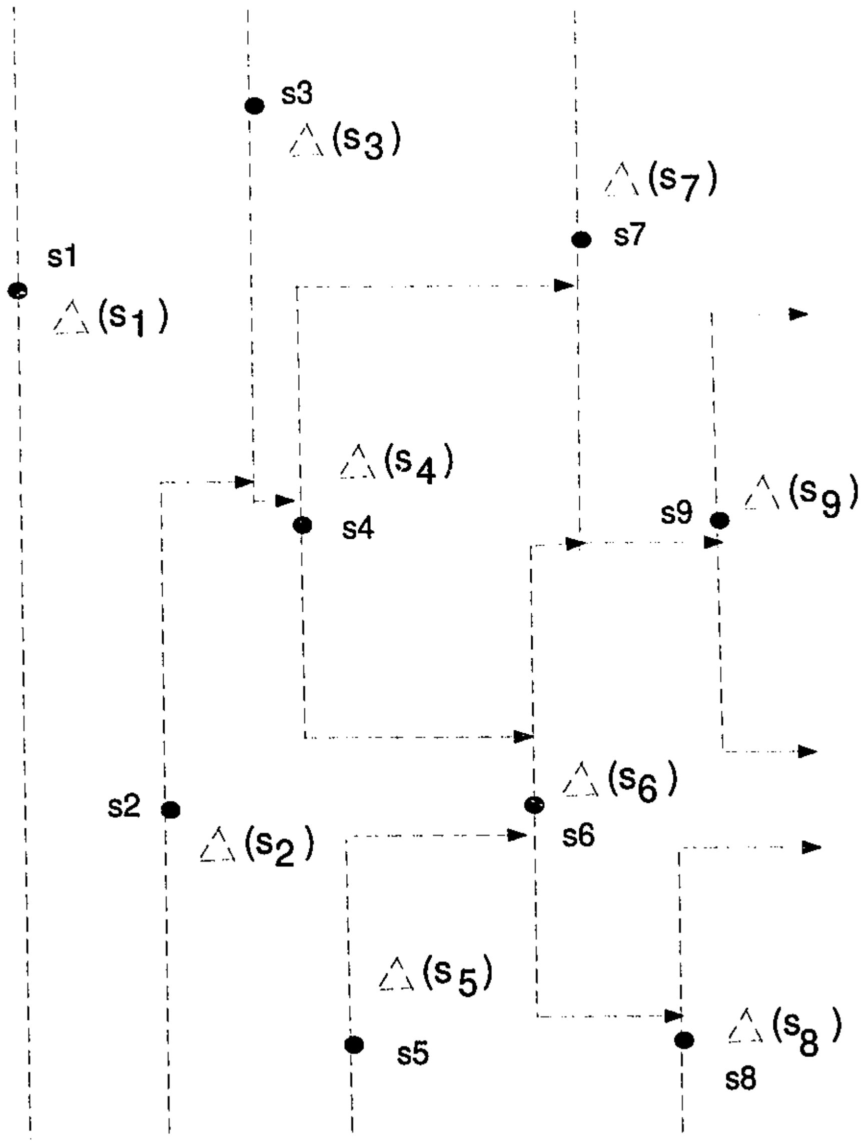
(b) Sweep line at site s6



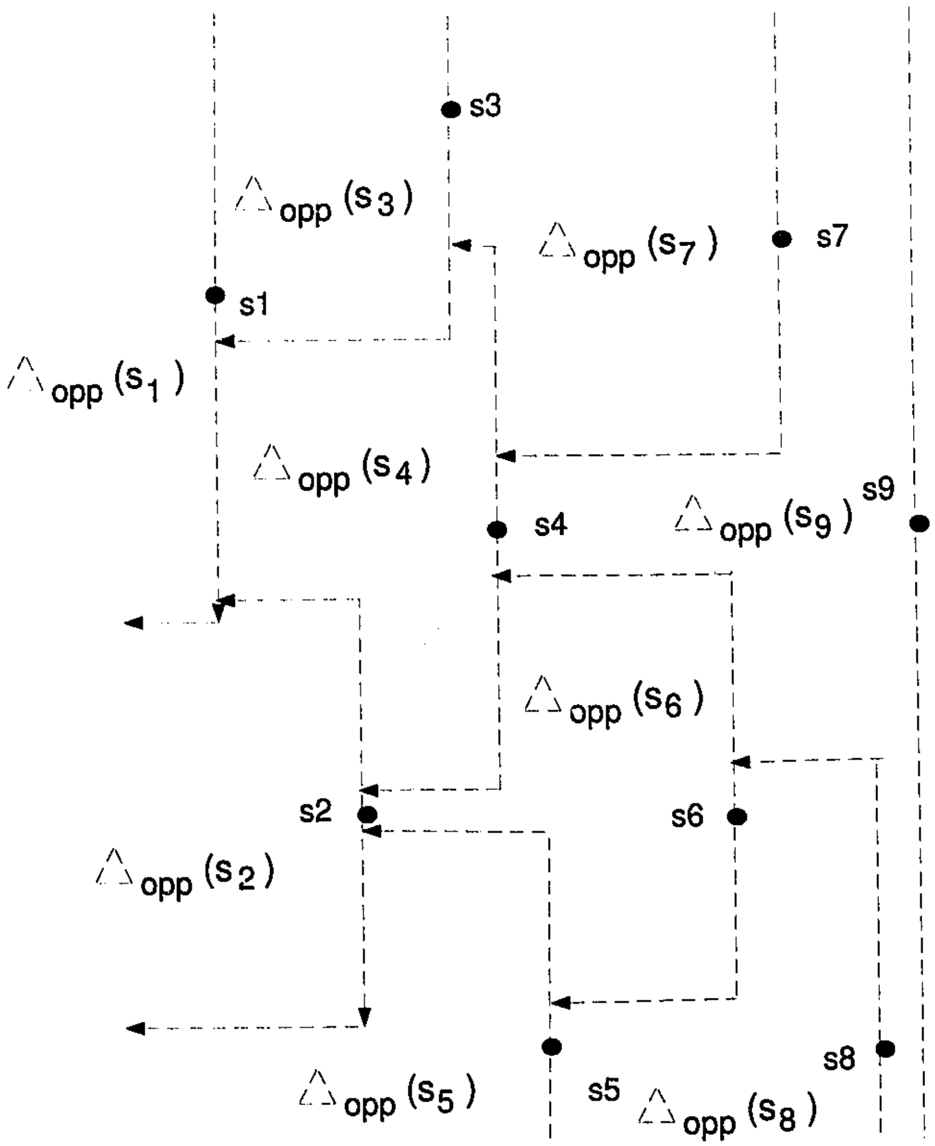
(a) Sweep line at site s7



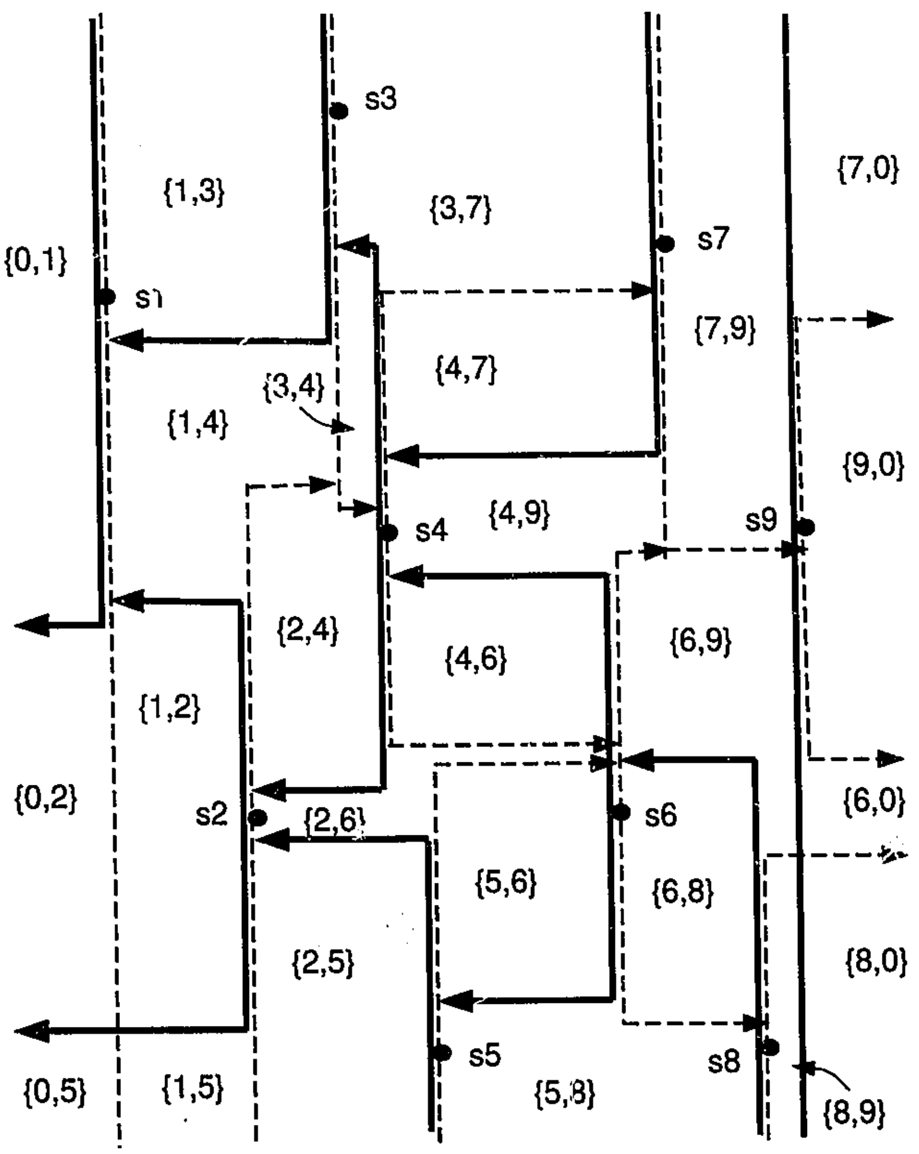
(b) Sweep line at site s8



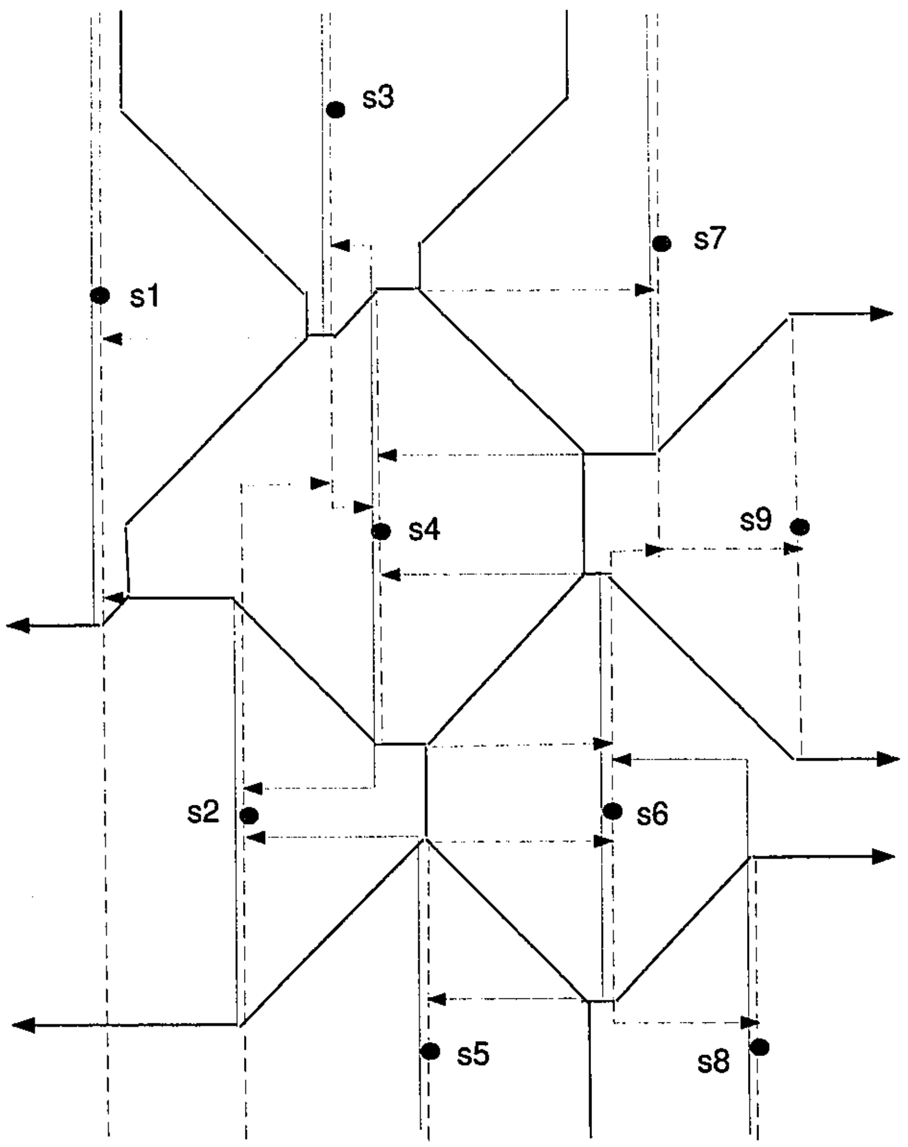
The left to right SPM $\mapsto(S)$



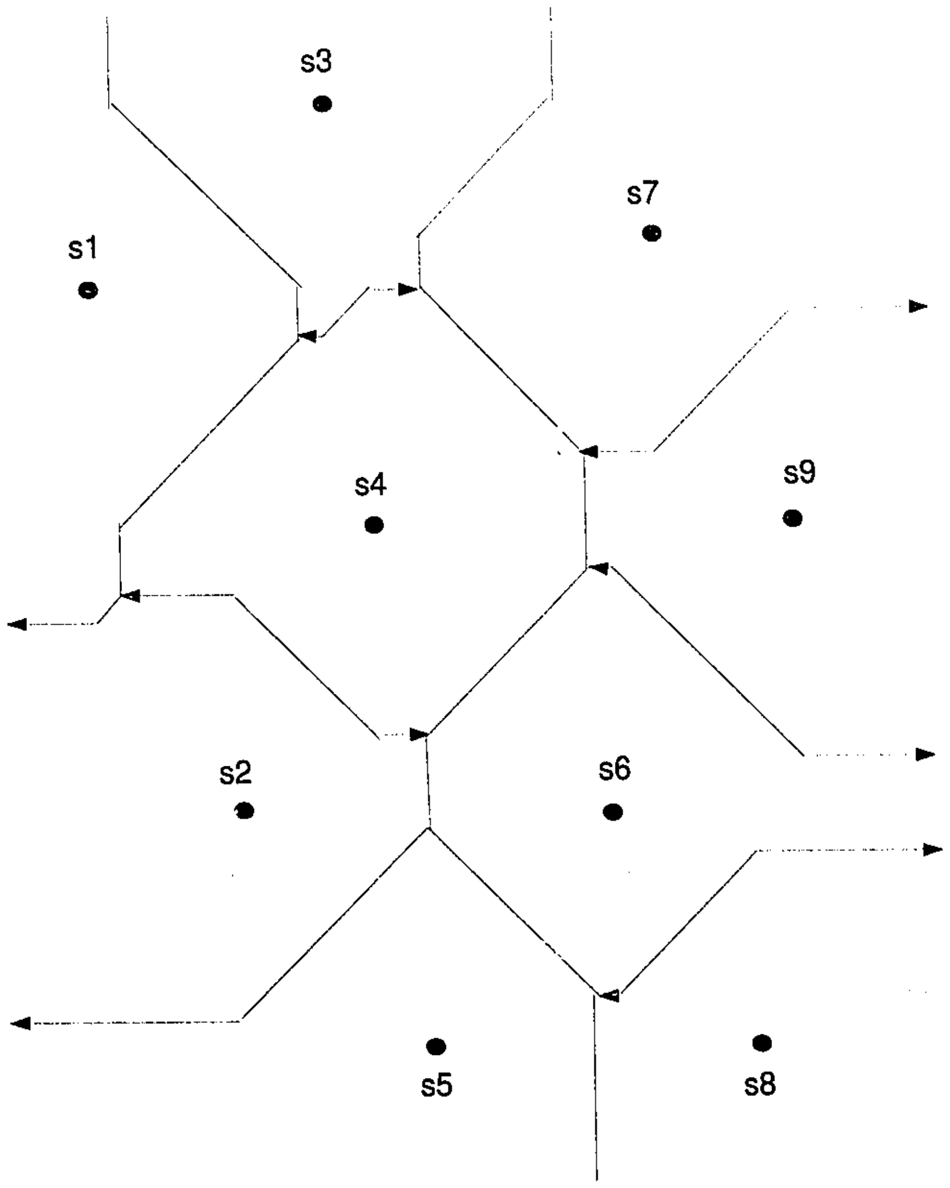
The right to left SPM$_{\hookleftarrow}$(S)



Areas formed by the SPM$_{\mapsto}$ (S) and SPM$_{\leftarrow}$ (S)



Voronoi arcs are traced



Voronoi diagram in L_1 metric

BIBLIOGRAPHY

[1] F. Aurenhammer and H. Edelsbrunner. An optimal algorithm for constructing the weighted voronoi diagrams in the plane. *Pattern Recognition, vol. 17, no. 2, pp. 251–257*, 1984.

[2] J.R. Bitner and C.K. Wong. Optimal and near-optimal scheduling algorithms for batched processing in linear storage. *SIAM J. on Computing 8, pp. 479–499*, 1979.

[3] K.Q. Brown. Voronoi diagram from convex hulls. *Inform. Process Letter, 9, pp. 223–228*, 1979.

[4] P. Chew and R.L. Drysdale III. Voronoi diagrams based on convex distance functions. *1st ACM Annual Symposium on Computational Geometry, pp.235–244*, 1985.

[5] R.L.III Drysdale and D.T. Lee. Generalization of voronoi diagram in the plane. *SIAM J. Comput. vol. 10, no. 1,*, Feb. 1981.

[6] R.L.III Drysdale and D.T. Lee. Generalized voronoi diagram in the plane. *Proc. of the 16th Annual Allerton Conference on Communications, Control and Computing, pp. 833–842*, Oct. 1978.

[7] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput. 15, pp.341–363*, 1986.

[8] S.J. Fortune. A sweep line algorithm for voronoi diagrams. *Proc. 2nd Ann. ACM Symp. Comput. Geom. pp.313–322.*, 1986.

[9] P. Green and R. Sibson. Computing dirichlet tesselations in the plane. *Compt. J. 21, pp.168–173.*

[10]F.K. Hwang. An o(n log n) algorithm for rectilinear minimal spanning trees. *ACM vol. 26, no. 2, pp.177–182*, April 1979.

[11]H. Imai, M. Iri, and K. Murota. Voronoi diagram in the laguerre geometry and its applications. *SIAM J. Comput. vol.14, no.1, pp. 93–105*, Feb. 1985.

[12]D.G. Kirkpatrik. Efficient computation of continuous skeletons. *Proc. 20th IEEE Annu. Symp. Found. Comput. Sci., pp.18–27*, Oct. 1979.

[13]D.T. Lee. Farthest neighbor voronoi diagrams and applications. *Dep. Elec. Eng. Comput. Sci., Northwestern Univ., Evanston, IL. Tech. Rep. 80–11–FC-04*, 1980.

[14]D.T. Lee. On k-nearest neighbor voronoi diagram in the plane. *IEEE Trans. on Comput., 35(6), pp. 478–487*, June 1982.

[15]D.T. Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE Transaction on Computers, 35(6), pp. 478–487*, June 1982.

[16]D.T. Lee. On finding k-nearest neighbors in the plane. *Technical report, Coordinated Sci. Lab., Univ. of Ill., Urbana, Illinois*, May 1976.

[17]D.T. Lee. Two dimensional voronoi diagram in the l_p metric. *J. of ACM, vol. 27, no.4, pp. 604–618*, Oct. 1980.

[18]D.T. Lee and F.P. Preparata. The all nearest neighbor problem for convex polygons. *Info. Proc. Lett., 7, pp. 189–192*, 1978.

[19]D.T. Lee and F.P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks, vol.14, pp. 393–410*, 1984.

[20]D.T. Lee and C.K. Wong. Voronoi diagrams in l_1 and l_{00} metrics with 2–dimentional storage applications. *SIAM J. on Comput. vol. 9, No.1, pp.200–212*, Feb. 1980.

[21]A. Lingas. Voronoi diagrams with barriers and shortest diagonal problem. *Information Processing Letters, 32:191–198*, 1989.

[22]D.E. Muller and F.P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science, 7(2), pp. 217–236*, Oct. 1978.

[23]F.P. Preparata. Steps into computational geometry. *Tech. Report, Coordinated Science Lab., Univ. of Illinois, Urbana, Il.*, 1977.

[24]R. Seidel. Constrained delaunay triangulation and voronoi diagrams with obstacles. *Rep.260, pp.178–191, IIG-TV Graz, Austria*, 1988.

[25]M.I. Shamos. Computational geometry. *PhD thesis, Dept. of Comput. Sci., Yale Univ.*, 1978.

[26]M.I. Shamos. Geometric complexity. *Proc. 7th ACM Annu. Symp. Theory Comput. pp.224–233*, May 1975.

[27]M.I. Shamos and D. Hoey. Closest–point problems. *Proc. 16th IEEE Ann. Symp. Found. Comput Sci. pp. 208–215*, 1976.

[28]B. Shneiderman and V. Goodman. Batched searching of sequential and tree structured files. *ACM Trans. Database Sys. 1, pp. 268–275*, 1976.

[29]M.I. Shomas and F.P. Preparata. Computational geometry. *Spring Verlag,* 1985.

[30]Y.H. Tsin and C. Wang. Finding constrained and weighted voronoi diagrams in the plane. *Proceedings of the Second Canadian Conference in Computational Geometry, pp. 200–303*, 1990.

[31]Y.H. Tsin and C. Wang. Finding geodesic voronoi diagram of points in the presence of rectilinear barriers. 1990.

[32]G. Voronoi. Nouvelles applications des paramétres continues á la théorie des formes quadratiques; premier mémoire: sur quelques propriétés des formes quadratiques positives parfaites. *J. Renie Angew. Math. 133, pp. 97–178*, 1907.

[33]G. Voronoi. Nouvelles applications des paramétres continues á la théorie des formes quadratiques; deuxiéme mémoire; recherches sur les parallélloédres primitifs. *J. Reine Angrew. Math. 134, pp.198–287*, 1908.

[34]C. Wang and L. Schubert. An optimal algorithm for constructing the delaunay triangulation of a set of line segments. *3rd ACM Annual Symposium on Computational Geometry, pp.223–232*, 1987.

VITA AUCTORIS

Jianan Wang graduated from the Department of Electrical Engineering and Computer Science at Shanghai Jiao Tong University with a Bachelor of Science in 1982. He then worked at The Ninth Design and Research Institute China State Shipbuilding Corporation, and later at Shanghai China Cooperation for Foreign Economic and Technological Corporation (Australia Branch).

He is currently a candidate for the Master's degree in Computer Science at the University of Windsor and expected to graduate in Sept. 1994.