An Optimal Algorithm for Greedy Triangulation of a Set of Points [1]

Cao An Wang
Department of Computer Science
Memorial University of Newfoundland, Canada A1C 5S7

Abstract
In this paper, we present an algorithm for finding the greedy triangulation of a set of n points in E^2. The algorithm requires $\Theta(n \log n)$ time and $\Theta(n)$ space, which is worst-case optimal.

1 Introduction

A **triangulation** of a set S of points (call them *sites*) in E^2 is a subdivision of the plane such that the vertices of this subdivision are the sites of S and every region internal to the convex hull of S is a triangle.

Let L be the set of line segments connecting all pairs of the sites of S. The **greedy triangulation** of S, denoted by $GT(S)$, is a triangulation of S obtained by repeatedly selecting the shortest segment in the remainders of L and inserting it into the partial $GT(S)$ if it does not cross any segment previously inserted (called the *greedy edge*).

Using a brute-force method to test the crossings of the selected segments and the greedy edges in the partial $GT(S)$ for finding $GT(S)$ may take $O(n^3)$ time and $O(n^2)$ space. The first non-trivial algorithm for speeding up this test was proposed by Gilbert [3], which requires $O(n^2 \log n)$ time and $O(n^2)$ space. Whether or not one can design an $o(n^2 \log n)$ time and $o(n^2)$ space algorithm for the problem became an open problem [1]. These complexity bounds remained unchanged for almost a decade until Lingas [8] (and later, Goldman [5] independently) improved the space bound to $O(n)$ and Levcopoulos and Lingas [10] (and later, Wang [15] independently) improved the time bound to $O(n^2)$.

Levcopoulos and Lingas [11] proposed a linear expected-time algorithm for constructing the greedy triangulation of S if S is uniformly distributed in a unit square. Dickerson, Drysdale, McElfresh, and Welzl [4] proposed an $O(n \log n)$ expected-time algorithm for the problem when S is uniformly distributed in any convex shape.

In this paper, we proposed an optimal deterministic algorithm to solve this problem. For simplicity, we use CDT and GT to denote constrained Delaunay triangulation and Greedy triangulation, respectively. The edges and the triangles of GT (CDT) are called the greedy (Delaunay) edges and greedy (Delaunay) triangles, respectively. All the proofs of the lemmas are omitted in this conference version. Those readers interested in the paper can find the details in reference [16].

[1]This work is supported by NSERC grant OPG0041629.

2 Preliminaries

Let L be a set of non-intersecting obstacle line segments, and let S be the sites containing all the endpoints of these obstacles.

Definition: The **CDT** of S in the presence of L, denoted by $CDT(S, L)$, is a triangulation of S such that the edge set of $CDT(S, L)$ contains all the elements of L, and the interior of the circumcircle (called Delaunay circle) of any triangle of $CDT(S, L)$, say $\Delta ss's''$, does not contain any element of $S - \{s, s', s''\}$ visible to s, s', and s''. The **CDT** of a simple polygon P, denoted by $CDT(P)$, is defined similarly, where the vertices of P are the vertices of the triangulation and the boundary edges of P are the obstacles. For our purpose, only the part of $CDT(P)$ internal to P is considered.

We use the following notations in the rest of the paper:
- $DT(S)$ denotes the standard Delaunay triangulation of S, $E_{DT(S)}$, its edge set.
- n' denotes the number of edges of $GT(S)$; $E_g(n')$ denotes the edge set of $GT(S)$.
- the process of finding the $GT(S)$ is said to be *in stage* k for $1 \leq k \leq n'$ if the kth shortest greedy edge has just been inserted into the partial greedy triangulation.

A notation with argument k means that the object represented by this notation is in the kth stage of the process.
- $E_g(k)$ denotes the subset of $E_g(n')$ which consists of the first k shortest edges; The edges of $E_g(k)$ are regarded as obstacles in stage $k \leq l \leq n'$; Consequently, a greedy triangle in stage k is formed by its three elements belonging to $E_g(k)$.
- $CDT(S, E_g(k))$ denotes the CDT of S in the presence of obstacles $E_g(k)$.
- $e_g(k+1)$ denotes the $(k+1)th$ greedy edge to be inserted to $GT(S)$.
- $E_{cd}(k)$ denotes the subset of the edge set of $CDT(S, E_g(k))$ such that $E_{cd}(k) \cap E_g(k) = \emptyset$.
- $E_d(k)$ denotes the remainder of $E_{DT(S)}$ in stage k. Note that $E_d(0) = E_{cd}(0) = E_{DT(S)}$.
- $E_c(k)$ denotes the set of edges such that each edge of $E_c(k)$ together with two edges of $E_g(k)$ form a triangle, and no edge of $E_c(k)$ belongs to $E_g(k) \cup E_{DT(S)}$ or crosses any edge of $E_g(k)$.
- Let $()ss'$ denote the *lune* with radius $\overline{ss'}$ and with centers s and s'.
- Let Oss' denote the *circle* with $\overline{ss'}$ as the diameter and with s and s' on its boundary.

Lemma 2.1: [Theorem 3.1, 10] Greedy edge $e_g(k+1)$ is either an edge of $E_{cd}(k)$ or an edge of $E_c(k)$ for $0 \leq k < n'$.

By Lemma 2.1, the next greedy edge $e_g(k+1)$ can be found from $E_{cd}(k) \cup E_c(k)$. By adding $e_g(k+1)$ to $E_g(k)$ and finding $CDT(S, E_g(k) \cup \{e_g(k+1)\})$ for $k = 1$ to $n' - 1$, we obtain $GT(S)$. Clearly, the time complexity of this method is dominated by the time for finding $CDT(S, E_g(k) \cup \{e_g(k+1)\})$ for $k = 1$ to $n' - 1$ when $CDT(S, E_g(k))$ and $e_g(k+1)$ are available.

A further inspection shows that we need not update the entire $CDT(S, E_g(k))$. If $e_g(k+1)$ is an edge of $E_{cd}(k)$, then the edge set of $CDT(S, E_g(k) \cup \{e_g(k+1)\})$ remains the same as that of $CDT(S, E_g(k))$, while if $e_g(k+1)$ is an edge of $E_c(k)$, then the two edge sets are different only locally. In the latter case, let edge $e_g(k+1)$ together with edges e and e' of $E_g(k)$ form a greedy triangle and let u be the endpoint shared by e and e'. Then, $e_g(k+1)$ crosses some of those Delaunay triangles of $CDT(S, E_g(k)$ incident at u, say T. In order to update $CDT(S, E_g(k) \cup \{e_g(k+1)\})$, it is sufficient to re-build the CDT of the area that covers T, where the area forms a special polygon P.

Algorithm 1 formalizes the above idea. Let P denote the polygon induced by $e_g(k+1)$. Let $CDT(P)$ denote the internal part of CDT of P. Operator min chooses the shortest edge in a set.

Algorithm 1
INPUT: S
OUTPUT: $GT(S)$
METHOD:
1. $k \leftarrow 0$; $E_g(k) \leftarrow \emptyset$; $CDT(S, E_g(k)) \leftarrow DT(S)$.
   (* after initialization, $E_c(0) = \emptyset$, $E_g(k) = \emptyset$, and $E_{cd}(0) = E_{DT(S)}$ *)
2. *While* ($E_g(k)$ is not a complete edge set of $GT(S)$) *Do*
   (a) $e_g(k+1) \leftarrow min(E_{cd}(k) \cup E_c(k))$; $E_g(k+1) \leftarrow E_g(k) \cup \{e_g(k+1)\}$;
   (* select a greedy edge according to Lemma 2.1 *)
   (b) *If* ($e_g(k+1) \epsilon E_c(k)$) *Then* find $CDT(S, E_g(k) \cup \{e_g(k+1)\})$ by finding $CDT(P)$;
   (* find $CDT(P)$ in linear-time using the method in [10,15] *)
   (c) $k \leftarrow k+1$;
   EndDo.

Theorem 2.1: [Theorem 3.2, 10, 15] Algorithm 1 finds $GT(S)$ in $O(n^2)$ time and $O(n^2)$ space.

3 Finding the GT of S

As $CDT(S, E_g(k))$ is a planar graph, the number of edges in $CDT(S, E_g(k))$ is $O(n)$. Then, the number of edges in $E_c(k)$ is also $O(n)$ by the definition of $E_c(k)$. Thus, Algorithm 1 requires at least $O(n^2)$ time. This is because there may exist $O(n)$ greedy edges from the edges of $E_c(k)$ for $k < n'$ such that each of them crosses $O(n)$ Delaunay edges, which requires $\Omega(n^2)$ update operations to obtain $CDT(S, E_g(k) \cup \{e_g(k+1)\})$ for $k = 0$ to $O(n)$.

We shall resolve this difficulty by avoiding updates of the constrained Delaunay triangulations in Step 2(b) of Algorithm 1. To do so, in Section 3.1, we introduce a special region, called the *nest* region, which is the union of the special polygons. Each of these special polygons is bounded by some edges of $E_d(0)$, and is induced by inserting a greedy edge originally belonging to $E_c(k)$. Inside such a nest region, the Delaunay edges of $E_{cd}(0)$ are removed and the region will not be further Delaunay-triangulated. Our new algorithm also starts at $DT(S)$, picks up the greedy edges in ascending order with their lengths and inserts them into $DT(S)$ and its subsequent figures one by one, until the figure becomes $GT(S)$. At the kth stage, the subsequent figure consists of a set of nest regions and their subregions, called *cores* bounded by some of $E_d(k) \cup E_g(k)$. The remaining problem is how to find the next greedy edge since Step 2(a) of Algorithm 1 is no longer functioning. This is because Lemma 2.1 does not work due to the nest regions are not Delaunay-triangulated. We shall prove in Lemma 3.6 that the next greedy edge $e_g(k+1)$ must belong to $E_d(k) \cup E_c(k) \cup SP(k)$, where $SP(k)$ is one type of candidates of greedy edges called *spines*. Thus, at the kth stage, the 'configuration' of our algorithm consists of the subsequent figure and a set of extra edges: $E_c(k)$ and $SP(k)$ with respect to the boundary edges of the nest regions and cores. We shall further show some properties of nest regions and cores in Sections 3.1 and 3.2 so that the spines can be calculated quickly. Consequently, $GT(S)$ can be found quickly.

3.1 Some properties of cores and nest regions

Definition: A **nest-region** (in stage k), denoted by NR , is a connected region without holes such that its boundary consists of some edges of $E_d(k)$ and some edges of $E_g(k)$ originally belonging to $E_d(0)$. A **core** of an NR (in stage k) is a subregion of the NR separated by some edges of $E_g(k)$. A **convex** subchain of core $\mathbf{c}$ (in stage k) is a piece of the boundary of $\mathbf{c}$ such that every edge of the subchain belongs to $E_g(k)$ and the inner angle of every interior vertex of the subchain is reflex. Two distinct nest regions are to be **merged** into an new nest region if they share an edge e of $E_d(k)$ in stage k and if e is crossed by a greedy edge of $E_g(j)$ for $k < j \leq n'$.

(**Remark:** The purpose of introducing the concepts of NR and *core* is to design a data structure for ray-shooting such that the candidates $SP(k)$ can be found quickly. For simplicity, we omit term 'in stage k' for an NR or a *core* while it is understandable from the context.)

Definition: Let e, e', and $e_g(k)$ of $E_g(k)$ lie on an NR such that they form a greedy triangle, where $e_g(k)$ originally belongs to $E_c(k-1)$. Let also $\vec{b}$ be the ray on the perpendicular bisector of e, emitting from the midpoint of e towards $e_g(k)$, and let $\vec{b'}$ w.r.t. e' be defined similarly. Then, $\vec{b}$ and $\vec{b'}$ will cross the boundary of the NR as well as the boundary of the core $\mathbf{c}$, where the boundary of $\mathbf{c}$ contains $e_g(k)$. Let p' and q' be the two first crossover points on the boundary of this NR and let p and q be the two first crossover points on the boundary of this $\mathbf{c}$. If $\vec{b}$ ($\vec{b'}$) reaches p' (q') before it crosses $\vec{b'}$ ($\vec{b}$), then p' and q' delimit a piece of the boundary of NR not containing e and e', denoted by $\mathbf{c}'_s = \widehat{p'q'}$, and p and q delimit a piece of the boundary of core $\mathbf{c}$, denoted by $\mathbf{c}_s = \widehat{pq}$ and called the **upper base.** Upper base $\mathbf{c}_s$ is defined to be empty if it does not contain any vertex of the NR or $\vec{b}$ and $\vec{b'}$ cross each other before they reach p' and q', respectively. A convex subchain of $\mathbf{c}$ with an endpoint of $e_g(k)$ as an extreme is denoted by $\mathbf{c}_c$ and called a **lower base.** There exist exactly two lower bases and at most one upper base of $\mathbf{c}$ w.r.t. $e_g(k)$.

Definition: The **spine** of the core of the NR w.r.t. $e_g(k)$ of $E_c(k-1)$ is the shortest edge between the vertices of $\mathbf{c}_s$ and the vertices of $\mathbf{c}_c$ such that the endpoints of the edge are visible each other. Clearly, the spine must lie inside the core and may not exist if $\mathbf{c}_s$ is empty. Let $SP(k)$ denote the set of spines in stage k. (Refer to Fig. 3.1 (a) and (b). for these definitions.)

Definition: Let $\overline{ss'}$ be the greedy edge $e_g(k+1)$ not belonging to $E_d(k) \cup E_c(k)$. Let B be a subset of $E_g(k)$, each of which intersects $()ss'$ so that any site lying inside $()ss'$ is not visible to s and s' due to the collective blockages of these edges. The **critical blocking edges** are a subset of B, denoted by E_b each of E_b intersects Oss', blocks the lines of sight of at least one site in $()ss'$ to s and s', and its two endpoints lie on or outside $()ss'$ such that the interior of the area bounded by $\overline{ss'}$, $\overline{s_is_z}$, $\widehat{ss_i}$, and $\widehat{s's_z}$ (or the interior of the triangle $\Delta ss's_z$) contains no sites, where s_i and s_z are the intersection points of this edge and the boundary of $()ss'$. A **pseudo** critical blocking edge is a special blocking edge such that only one of its endpoints, say s_x, lies in $()ss'$ and s_x is immediately blocked by a critical blocking edge. For simplicity, both the critical and pseudo critical blocking edges are denoted by E_b. (Refer to Fig.3.2(a)(ii), Fig.3.2(b)(iv), and Fig.3.2(b)(v) for the definition.)

The following five lemmas lead to that the $(k+1)$th greedy edge is an edge of $E_d(k) \cup E_c(k) \cup SP(k)$.

Lemma 3.1: Let $\overline{ss'}$ be the greedy edge $e_g(k+1)$ not belonging to $E_d(k) \cup E_c(k)$. Then, there exists a set of edges E_b in stage $j \leq k$, such that none of the sites lies inside $()ss'$ is visible to s and s' in stage j due to the blockage of E_b. E_b contains at least an edge e_r which block the line of sight of a site r in Oss' to s and s'.

Lemma 3.2: Let $\overline{ss'}$ be the greedy edge $e_g(k+1)$ not belonging to $E_d(k) \cup E_c(k)$. Then, s and s' belong to the boundary of the same NR in stage $j \leq k$, and the NR contains an e_r for some r in Oss'.

Lemma 3.1 and Lemma 3.2 imply that there exists an $e_r \epsilon E_b$ in stage $j \leq k$ such that every point of e_r in $()ss'$ is visible to s and s'.

Let $\overline{tt'}$ be the segment of the line supporting Oss' and t and t' lie on the boundary of $()ss'$. Let $\overline{gg'}$ and $\overline{g's}$ be defined similarly. Let c be an intersection point of circles $O_{s,\overline{ss'}}$ and $O_{s',\overline{ss'}}$ with s (s') as center and $\overline{ss'}$ as radius. Let d be the intersection of $\overline{ss'}$ and $\overline{tt},$. (Refer to Fig. 3.2 (b).)

Lemma 3.3: Let $\overline{ss'}$ be the greedy edge $e_g(k+1)$ not belonging to $E_d(k) \cup E_c(k)$. Then, by Lemma 3.1 there exists a set E_b. Then, each of E_b can be in only one of the following three cases: (1) it crosses arc $\widehat{sc}$ exactly twice or it crosses arc $\widehat{cs'}$ twice, (2) it crosses arc $\widehat{st}$ and arc $\widehat{t'c}$ respectively or it crosses arc $\widehat{s'g}$ and arc $\widehat{g'c}$ respectively, and (3) it crosses arc $\widehat{s'g}$ and arc $\widehat{st}$ respectively. (Refer to Fig.3.2.)

Lemma 3.3 implies that there always exists a critical (or pseudo critical) blocking edge e_r of E_b which determines two perpendicular bisectors such that the two bisectors determine the upper and lower bases w.r.t. e_r.

Lemma 3.4: Let $\overline{ss'}$ be the greedy edge $e_g(k+1)$ not belonging to $E_d(k) \cup E_c(k)$. Then, s and s' belongs to the vertices of the upper and lower bases with respect to some edge of E_b, respectively, and $\overline{ss'}$ is a spine on the core with respect to this edge of E_b.

Lemma 3.5. Edge $e_g(k+1)$ belongs to $E_c(k) \cup E_d(k) \cup SP(k)$.

3.2 The final algorithm

The remaining problem is how fast the spines can be found. The following two lemmas show that the spines can be found in logarithmic time using ray shooting method.

Lemma 3.6. Let $e_g(k+1)$ $(=\overline{ss'})$ be the spine of the core $\mathbf{c}$ of the NR w.r.t. $e_r(=\overline{s_i s_z})$, then s lies on the convex subchain $\mathbf{c}_c$ and s' lies on a convex subchain $\mathbf{c}'$ of $\mathbf{c}_s$, and $\overline{ss'}$ can be determined in logarithmic time if $\mathbf{c}_s$ and $\mathbf{c}_c$ are available. (Refer to Fig.3.3.)

The remaining problem is to show that the bases of the spine w.r.t. $e_g(k+1)$ ($\epsilon E_c(k)$) for $k\epsilon[1,n']$ can be calculated in logarithmic time so that the spine can be determined in logarithmic time by Lemma 3.6. Note that the number of spines created is proportional to that of edges of $E_g(k+1)$ originally belonging to $E_c(k)$, then finding these spines takes at most $O(n \log n)$ time.

Note by definition that $\mathbf{c}_s$ w.r.t. $e_g(k+1)$ ($\epsilon E_c(k)$) belongs to the core $\mathbf{c}$ of the NR containing $e_g(k+1)$, and it is delimited by the crossover points of the rays $\vec{b}$ and $\vec{b'}$ of e and e', where e, e', and $e_g(k+1)$ form a greedy triangle. These crossover points can be found by directly applying the ray-shooting method to the core. However, because the cores may expand, shrink, and split, it is difficult to maintain such a ray-shooting data structure within $O(n \log n)$ time bound.

Alternatively, we can use the portion of $DT(S)$ on the nest regions as a data structure to support the ray-shooting operation on an NR. Such a data structure is easy to maintain because the boundary of an NR belongs $E_d(0)$ and an NR can only expand. We then use another data structure to connect NR and its cores. Using this connecting data structure, we can quickly locate the corresponding crossover point of the core and the shooting ray if the crossover point of the NR and the ray is known.

Lemma 3.7: Let $e_g(k+1)$ belong to $E_c(k)$ and let $\mathbf{c}$ be the core containing $e_g(k+1)$ and NR be the nest region containing $\mathbf{c}$. Then, the corresponding $\mathbf{c}_s$ and $\mathbf{c}_c$ of $\mathbf{c}$ can be found in $O(\log \mid NR \mid)$ time.

(**Remark:** The best known ray-shooting algorithm for a simple polygon with k holes and total of n vertices takes $O(\sqrt{(k)} \log n)$ time [14]. In our particular case, the ray may cross several nest regions, but these nest regions are also crossed by $e_g(k+1)\epsilon E_c(k)$, which allows us to achieve logarithmic time bound.)

Theorem 1: $GT(S)$ can be found in $O(n \log n)$ time and $O(n)$ space, where $n = \mid S \mid$.

4 References

[1] Aggarwal A., *Computational Geometry*, MIT Lecture Notes 18.409 (1988).

[2] Chazelle B. and Guibas L., *Visibility and Intersection Problems in plane geometry*, Discr. and Comp. Geometry 4 (1989), pp. 551-581.

[3] Gilbert P., *New results on planar triangulations*, *Tech. Rep.* ACT-15 (1979), Coord. Sci. Lab., University of Illinois at Urbana.

[4] Dickerson M., Drysdale R., McElfresh S., and Welzl E., *Fast greedy triangulation algorithms*, To appear in the Proc. of 10th ACM symposium on Computational Geometry. (1994)

[5] Goldman S., *A space efficient Greedy triangulation algorithm*, IPL, 31 (1989), pp.191-196.

[6] Kirkpatrick D., *A note on Delaunay and optimal triangulations*, IPL, 10(3) (1980), pp.127-128.

[7] Lee D. and Schachter B., *Two algorithms for constructing Delaunay triangulations*, Int. Journal on Computer and Information Sciences, 9(3) (1980), pp.219-243.

[8] Lingas, A., *A space efficient algorithm for the greedy triangulation*, Lecture Notes in Control and Information Sciences, Vol. 113 (1987), pp. 359-364.

[9] Levcopoulos C. and Lingas, A., *On approximation behavior of the greedy triangulation for convex polygon*, Algorithmica, 2 (1987), pp.175-193.

[10] Levcopoulos C. and Lingas, A., *Fast algorithms for greedy triangulation*, **BIT**, 32 (1992), pp. 280-296.

[11] Levcopoulos C. and Lingas, A., *Greedy triangulation approximates the optimum and can be implemented in linear time in the average case*, Lecture Notes in CS, 497 (1991), pp.139-148.

[12] Preparata F. and Shamos M., (1985), *Computational Geometry*, Springer-Verlag.

[13] Seidel R., *Constrained Delaunay triangulations and Voronoi diagrams with obstacles*, Rep. 260, IIG-TU Graz, Austria (1988), pp. 178-191.

[14] Suri S. and Hershberger J., (1993), *A pedstrian approach to ray shooting: Shoot a ray, take a walk*, The Proc. of 4th ACM-SIAM symposium on Discrete Algorithms, pp.

[15] Wang C., *Efficiently updating the constrained Delaunay triangulations*, **BIT**, 33 (1993), pp. 238-252.

[16] Wang C., *An optimal algorithm for greedy triangulation of a set of points*, Technical Report, Memorial University of Newfoundland, (1993).

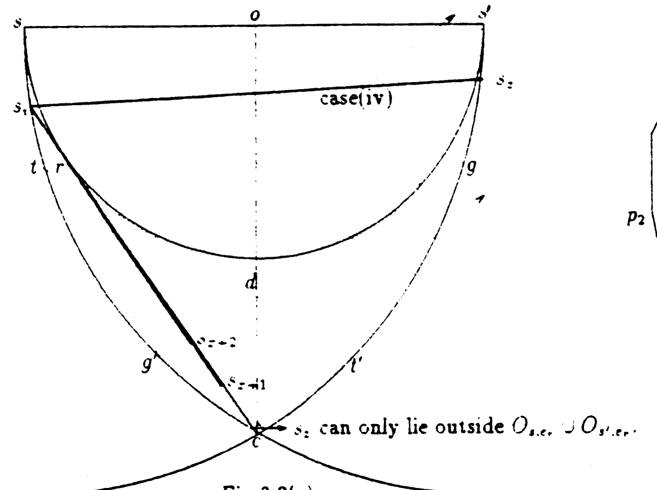

Fig.3.2(c)

$s_r, s_{r+1}, \ldots$ must form a convex chain facing s'. s_r must lie outside $O_{s,e_r} \cup O_{s',e_r}$ and s_z must lie outside $O_{s_{r+1},\overline{s_1 s_{r+1}}}$.

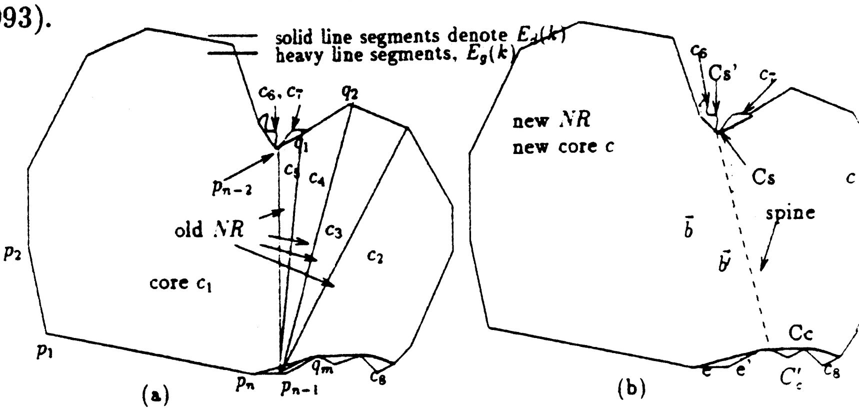

Figure 3.1 The old $\mathcal{NR}$s in case (a) are merged to a new $\mathcal{NR}$ in case (b) induced by greedy edge $\overline{p_n q_m}$. The new $\mathcal{NR}$ is $(p_1, \ldots, p_{n-2}, \ldots, q_1, \ldots, q_m, p_{n-1}, p_n)$. The old cores $(c_1, c_2, c_3, c_4, c_5)$ are merged to a new core c. Symbols $c_s, c_s', c_e, c_e', \bar{b}, \bar{b}'$, and spine are shown in case (b).

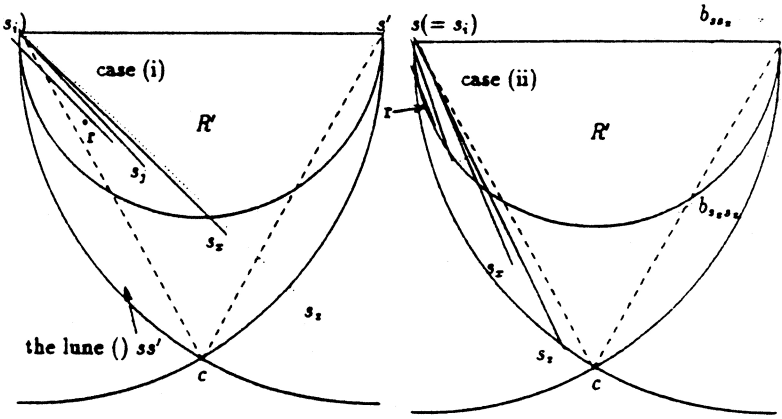

Figure 3.2(a) Case (a) of Lemma 3.3.

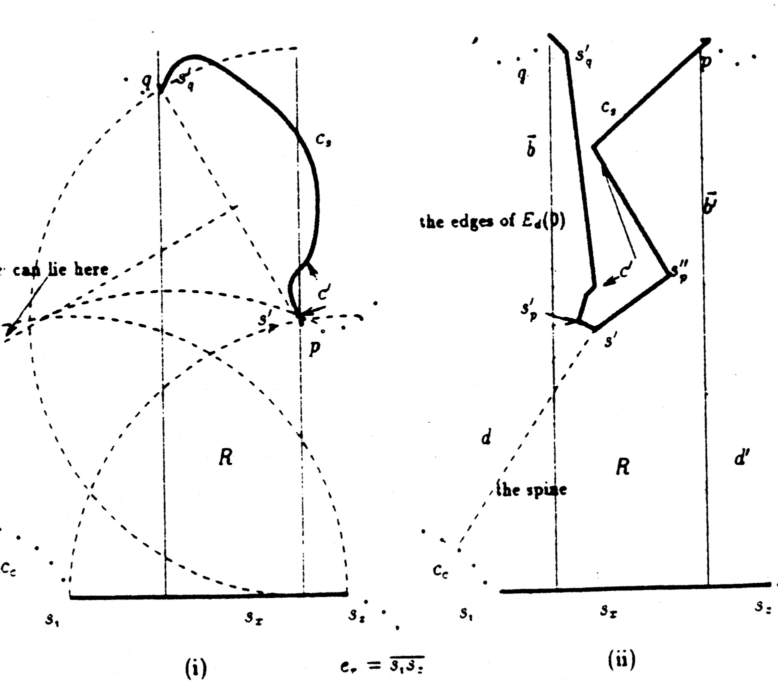

Fig. 3.3.

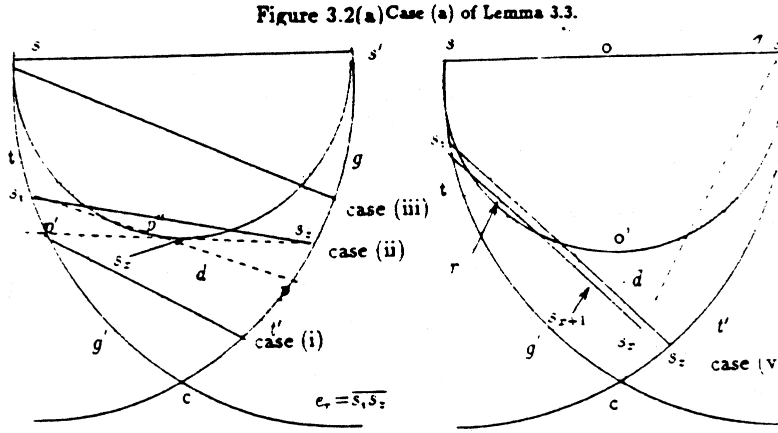

Figure 3.2 (b) Case (b) of Lemma 3.3.