Enumerating Constrained Non-crossing Geometric Spanning Trees *

Naoki Katoh and Shin-ichi Tanigawa

Department of Architecture and Architectural Engineering, Kyoto University, Kyoto 615-8450 Japan, {naoki,is.tanigawa}@archi.kyoto-u.ac.jp.

Abstract. In this paper we present an algorithm for enumerating without repetitions all non-crossing geometric spanning trees on a given set of n points in the plane under edge constraints (i.e., some edges are required to be included in spanning trees). We will first prove that a set of all edge-constrained non-crossing spanning trees is connected via remove-add flips, based on the constrained smallest indexed triangulation which is obtained by extending the lexicographically ordered triangulation introduced by Bespamyatnikh. More specifically, we prove that all edge-constrained triangulations can be transformed to the smallest indexed triangulation among them by $O(n^2)$ times of greedy flips. Our enumeration algorithm is based on the reverse search paradigm of Avis and Fukuda, and it generates each output graph in $O(n^2)$ time and $O(n)$ space. This result improves the previous $O(n^3)$ bound by Avis and Fukuda for the unconstrained case by factor of $O(n)$. For the edge constrained case, the previous algorithm cannot be extended so as to cope with edge constraints. However, our algorithm can deal with the edge-constrained case in the same running time.

1 Introduction

Given a graph $G = (V, E)$ with n vertices and m edges where $V = \{1, \ldots, n\}$, G is a *spanning tree* if and only if G is connected containing no cycle. An embedding of the graph $G(P)$ on a set of points $P = \{p_1, \cdots, p_n\} \subset \mathbf{R}^2$ is a mapping of the vertices to points in the Euclidian plane $i \mapsto p_i \in P$. The edges (i, j) of G are mapped to straight line segments (p_i, p_j). An embedding is *non-crossing* if each pair of segments (p_i, p_j) and (p_k, p_l) have no point in common without their endpoints.

An embedded spanning tree on the point set is called *geometric spanning tree*. Here geometric non-crossing spanning tree is simply called non-crossing spanning tree (NST). We assume in this paper that spanning trees are embedded on fixed general point set P in $\mathbf{R}^2$, and x-coordinates

* Supported by JSPS Grant-in-Aid for Scientific Research on priority areas of New Horizons in Computing.

of them are all distinct. Let F be a set of non-crossing line segments on P. A spanning tree containing F is called *F-constrained spanning tree*. Then we give an algorithm for enumerating all the *F-constrained non-crossing spanning trees (F-CNST)*. We simply denote a vertex p_i by i and an edge p_ip_j by ij or (i, j).

Novelty. The algorithm we propose requires $O(n^2)$ time per output and $O(n)$ space. For the unconstrained case (i.e. $F = \emptyset$), the algorithm by Avis and Fukuda [7] requires an $O(n^3)$ time per output and $O(n)$ space. For this case, our algorithm improved their algorithm by $O(n)$ factor. Also it does not seem that the algorithm by [7] can be extended to the edge-constrained case. For the F-constrained case, in our recent paper [9], we proposed an algorithm for enumerating the F-constrained non-crossing minimally rigid frameworks embedded on a given point set in the plane. We remarked therein that based on a similar approach, we could develop an algorithm for enumerating F-CNSTs, although we have not given either any algorithm details or analysis of the running time. Although the details are omitted here, the running time of this algorithm is $O(n^3)$ time per output and it seems difficult to improve this running time.

Historical Perspective. Enumerating combinatorial and geometric objects are fundamental problems and several algorithms have been developed for, e.g. a set of triangulations [7, 12, 14], non-crossing spanning trees [7], pseudo-triangulations [4, 10, 13] and non-crossing minimally rigid frameworks [8, 9]. Let $\mathcal{O}$ be a set of objects to be considered. Two objects are *connected* iff they can be transformed to each other by a pre-defined *local operation*, where local operation generates one object from the other by means of small changes. Especially, local operation is sometimes called *(1-)flip* if they have all but one edge (or element) in common. Define a graph $\mathcal{G}_\mathcal{O}$ on $\mathcal{O}$ with a set of edges connecting between objects that can be transformed to each other by one local operation. Then the natural question is how we can design local operation so that $\mathcal{G}_\mathcal{O}$ is connected, or, if it is possible, how we can design $\mathcal{G}_\mathcal{O}$ with small diameter. There are several known results for these questions for triangulations (e.g. [17]), pseudo-triangulations [1], geometric matchings [16], some classes of simple polygons [15] and also for NSPs [1–3, 7]. Especially relevant to the historical context of our work are the results for NST by Avis and Fukuda [7], and Aichholzer et al. [1–3]. Let $\mathcal{SP}$ be a set of all NSTs on a set of n points. Avis and Fukuda [7] have developed simple 1-flip such that $\mathcal{G}_\mathcal{SP}$ is connected with diameter $2n - 4$. For the case of a local operation other

than 1-flip, Aichholzer et al. have described the operations with diameters $O(\log n)$ [2] and improved result [1]. Aichholzer et al. in [2] also tried to design 1-flip with the additional requirement, called *edge slide*, such that removed edge moves to the other one along an adjacent edge keeping one endpoint of the removed edge fixed. And Aichholzer and Reinhardt [3] have proved that all NSTs are connected with $O(n^2)$ edge slides. In this paper, we will propose 1-flip with the additional requirement such that removed and added edges are sharing one endpoint, and show that all F-CNSTs are connected by $O(n^2)$ flips sharing one endpoint plus $O(n)$ base exchange operations.

Main tools. Main tools we use are reverse search and the F-constrained smallest indexed triangulation. Reverse search is a memory efficient method for visiting all the nodes of $\mathcal{G}_{\mathcal{O}}$ developed by Avis and Fukuda [6, 7] has been successfully applied to a variety of combinatorial and geometric enumeration problems. It generates all the elements of $\mathcal{O}$ by tracing the nodes in $\mathcal{G}_{\mathcal{O}}$. To trace $\mathcal{G}_{\mathcal{O}}$ efficiently, it defines a *root* on $\mathcal{G}_{\mathcal{O}}$ and *parent* for each node except the root. Define the parent-child relation satisfying the following conditions: (1) each non-root object has a unique parent, and (2) an ancestor of an object is not itself. By this, iterating going up to the parent leads to the root from any other node in $\mathcal{G}_{\mathcal{O}}$ if $\mathcal{G}_{\mathcal{O}}$ is connected. The set of such paths defines a spanning tree, known as the *search tree*, and the algorithm traces it by depth-first search manner. So, the necessary ingredients to use the method are an implicitly described connected graph $\mathcal{G}_{\mathcal{O}}$ on the objects to be generated, and an implicitly defined search tree in $\mathcal{G}_{\mathcal{O}}$. In this paper we supply these ingredients for the problem of generating all F-CNSTs.

The idea of a smallest indexed triangulation is derived from a lexicographically order triangulation developed by Bespamyatnikh [12] for enumerating triangulations efficiently. We generalize it to an F-constrained triangulation by associating an appropriate index for each triangulation rather than lexicographical ordering. In this paper a smallest indexed triangulation plays a crucial role in the development of our enumeration algorithm. We conjecture that the general idea to use triangulations which is proposed in this paper can be extended to develop an efficient algorithm for enumerating non-crossing graphs other than non-crossing spanning trees and minimally rigid frameworks because any non-crossing graph can be augmented to a triangulation and there exists an efficient algorithm for enumerating triangulations based on a reverse search.

Organization. We review smallest indexed triangulation by Bespamyatnikh and extended it to the F-constrained smallest indexed triangulation in Section 2. Section 3 shows that F-constrained non-crossing spanning trees are connected by $O(n^2)$ flips. Section 4 gives an algorithm for enumerating F-constrained non-crossing spanning trees. Section 5 proves the correctness and gives detailed analysis of the algorithm.

2 Smallest Indexed Triangulation

In this section, we define new index for a set of triangulations and an optimal triangulation with respect to the associated index, which we call a *smallest indexed triangulation*. Then we show that any edge-constrained triangulations can be transformed into smallest index triangulation by $O(n^2)$ flips. We remark that our idea, that is the index for each triangulation, is derive from the lexicographical ordering developed by Bespamyatnikh although he had not extended his results to edge-constrained case. We will use the smallest indexed triangulation to enumerate edge-constrained non-crossing spanning trees. Before introducing the indexed triangulation, we first define several notations which we will use throughout paper.

2.1 Notations

Let P be a set of n points on the plane, and for simplicity we assume that the vertices $P = \{1, \ldots, n\}$ are labeled in the increasing order of x-coordinates with distinct x-coordinates. For two vertices $i, j \in P$, we use the notation, $i < j$, if x-coordinate of i is smaller than that of j. We use the notation P_i to represent $\{i+1, \ldots, n\} \subseteq P$ for $i \in P$.

We usually denote an edge between i and j with $i < j$ by (i, j). For three points i, j, k the signed area $\Delta(i, j, k)$ of a triangle Δijk is defined by $\Delta(i, j, k) = (x(j) - x(i))(y(k) - y(i)) - (x(k) - x(i))(y(j) - y(i))$, where $x(\cdot)$ and $y(\cdot)$ are x-coordinate and y-coordinate of each point. The sign of $\Delta(i, j, k)$ tells us that k is on the left and right side of a line through i and j by $\Delta(i, j, k) > 0$ and $\Delta(i, j, k) < 0$, respectively. Then the lexicographical ordering on a set of edges is defined as follows: for $e = (i, j)$ with $i < j$ and $e' = (k, l)$ with $k < l$, e is lexicographically smaller than e' (denoted by $e \prec e'$ or $e' \succ e$) iff $i < k$ or $i = k$ and $\Delta(i, j, l) < 0$[1], and denote by

[1] In general the lexicographical ordering for edge set is defined in such a way that $e = (i, j)$ is smaller than $e' = (k, l)$ iff either $i < k$ or $i = k$ and $j < l$ holds. But in this paper we adopt our lexicographical ordering for efficient enumeration described in Sections 4 and 5.

$e = e'$ when they coincide. Notice that, when $i = k$, the lexicographical ordering corresponds to the clockwise ordering around i in our definition.

For two vertices $i, j \in P$, j is *visible* from i with respect to a constrained edge set F when an edge (i, j) and any edge in F do not have a point in common except their endpoints. In this paper, we assume that j is visible from i when $(i, j) \in F$. And we denote a set of vertices of P visible from i with respect to F by $V_F(i, P)$.

Let $conv(P')$ be a convex hull of a point set P'. For a vertex i that is outside of the convex hull of P', i.e. $i \notin conv(P')$, $j \in P'$ is visible from i with respect to (the boundary of) $conv(P')$ when $j = (i, j) \cap conv(P')$ holds (see Fig. 1).

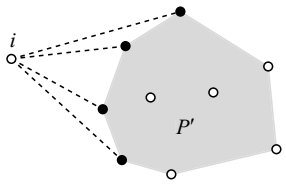


Fig. 1. Black vertices represent a set of vertices visible from i with respect to $conv(P')$.

For an edge $e = (i, j)$ with $i < j$, let $l(e)$ and $r(e)$ denote the left and right endpoints of e, i.e. $l(e) = i$ and $r(e) = j$, respectively. A straight line passing through i and j split $\mathbf{R}^2$ into two regions $(i, j)^+$ and $(i, j)^-$ that are open regions of left and right sides of (i, j), i.e. $e^+ = (i, j)^+ = \{p \in \mathbf{R}^2 \mid \Delta(i, j, p) > 0\}$ and $e^- = (i, j)^- = \{p \in \mathbf{R}^2 \mid \Delta(i, j, p) < 0\}$. Similarly, the closed regions $(i, j)^{+,0}$ and $(i, j)^{-,0}$ are defined. Moreover, considering a line through a vertex i perpendicular to x-axis, we can define $(i)^+ = \{p \in \mathbf{R}^2 \mid x(i) < x(p)\}$ and $(i)^- = \{p \in \mathbf{R}^2 \mid x(p) < x(i)\}$.

For $i \in P$, let $F(i)$ denote a set of constrained edges of F whose left endpoints are coincide with i. *Upper* and *lower hull edges*, (i, i^{up}) and (i, i^{low}), of i with respect to (the constrained edge set) F are defined as the upper and lower boundary edges of the convex hull of $\{i\} \cup V_F(i, P_i)$ incident to i. Notice that $V_F(i, P_i) \subset (i, i^{up})^{-,0}$ and $V_F(i, P_i) \subset (i, i^{low})^{+,0}$ hold. They define *empty region* in which no point of P exists as we describe below. Let l_i be a line perpendicular to x-axis passing through i, and let f_1 and f_2 be the closest edges from i among F intersecting with l_i in the upper and lower side respecting i (if such edge exists). Then there exists no point of P inside the region bounded by l_i, f_1 (and f_2), and the line through i and i^{up} (and i^{low}, respectively). When f_1 (and f_2) does not

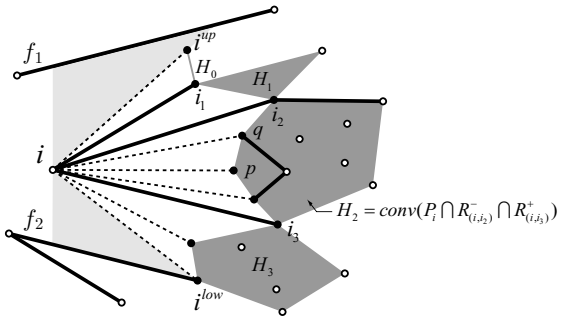


Fig. 2. The part of CSIT around i, where bold edges represent edges of F. Upper and lower light gray regions are empty regions of an upper hull edge (i, i^{up}) and a lower hull edge (i, i^{low}), respectively. CSIT has edges between i and black vertices.

exist, it is defined by the region bounded by l_i and the line through i and i^{up} (and i^{low}). We call this fact *empty region properties* of the upper and lower hull edges.

2.2 Constrained Smallest Indexed Triangulation

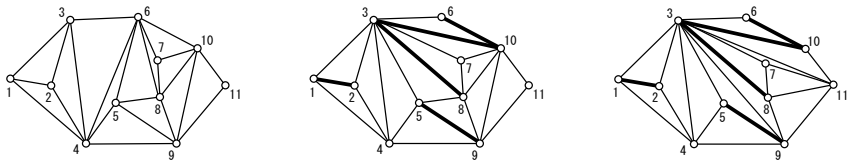
Although we would be better of introducing the result of Bespamyatnikh, we omit it in this extended abstract (see [12]). And let us extend his result to edge-constrained triangulations. F-constrained smallest indexed triangulation denoted by CSIT or F-CSIT is defined as follows:

Definition 1. *Let (i, i^{up}) and (i, i^{low}) be the upper and lower hull edges of $i \in P$ respecting F, and denote a set of edges of $F(i) \cup \{(i, i^{up}), (i, i^{low})\}$ by $(i, i_0), (i, i_1), \ldots, (i, i_k)$ arranged in clockwise order around i, (where $i_0 = i^{up}$ and $i_k = i^{low}$ holds). Then F-constrained smallest indexed triangulation (CSIT) has an edge (i, j) with $i < j$ if and only if j is visible from i with respect to the convex hull $H_l = conv(P_i \cap (i, i_l)^{-,0} \cap (i, i_{l+1})^{+,0})$ for some l with $0 \leq l \leq k-1$ (see Fig. 2).*

We give an example of CSIT in Fig.3(b) for 11 points, and also give an example when $F = \emptyset$ in Fig. 3 which is called SIT. We remark that CSIT always has the edges of $F(i) \cup \{(i, i^{up}), (i, i^{low})\}$ for all $i \in P$.

Lemma 1. CSIT is a triangulation of the point set P.

Proof. For $i \in P$, let $i_0 = i^{up}, i_1, \ldots, i_k = i^{low}$ denote the vertices as defined in Definition 1. Assume that all faces of the subgraph T' of CSIT induced by P_i are triangles, and we show that all new faces obtained by adding i and connecting it with T' by the edges defined in Definition 1 are



(a) (b) (c)

Fig. 3. (a)SIT. (b)CSIT. (c)non-CSIT.

also triangles. Then, all faces of CSIT are triangles by the induction from $i = n - 1$ to 1, which implies that CSIT is triangulation. (Clearly CSIT has the boundary edges of $conv(P)$.) Let us consider $H_l = conv(P_i \cap (i, i_l)^{-,0} \cap (i, i_{l+1})^{+,0})$ for $0 \leq l \leq k - 1$, and a set of vertices of H_l visible from i with respect to H_l which we denote $V_{H_l}(i) \subset H_l$ (see Fig. 2). From Definition 1, CSIT has edges between i and vertices of $V_{H_l}(i)$ without intersection since each $V_{H_l}(i)$ is visible from i respecting all edges of T'. And CSIT also has a part of the boundary of H_l connecting the sequence of vertices of $V_{H_l}(i)$ from i_l to i_{l+1}. It is because, for $p, q \in V_{H_l}(i)$ with $p < q$ such that p and q is adjacent on the boundary of H_l, (p, q) is at least one of the upper and lower hull edges of p, (otherwise $q \notin V_{H_l}(i)$). Then all new faces incident to i are triangles. □

2.3 Greedy Flipping in Constrained Triangulations

Let CT^* denote CSIT on a given point set. For any F-constrained triangulation, an *index* of T is defined as a pair of integers $n - c$ and d, and denoted by $index(T) = (n - c, d)$, where $c = c(T) \in \{1, \ldots, n - 1\}$ and $d = d(T) \in \{1, \ldots, n - 3\}$ are the label of the *critical vertex* of T and the *critical degree* of the critical vertex, respectively. The *critical vertex* is the smallest label of a vertex whose incident edges differ from the corresponding set of incident edges in CT^*. The *critical degree* is the number of edges incident to the critical vertex not contained in CT^*. The index of CT^* is defined to be $(0, 0)$. Then, for two triangulations T and T' of $index(T) = (n - c, d)$ and $index(T') = (n - c', d')$, T has smaller index than that of T' when $n - c < n - c'$, or $c = c'$ and $d < d'$ hold. Note that the index decreases as the label of the critical vertex increases. For example, a triangulation in Fig. 3(c) has an index $(3, 2)$.

For an edge e in a triangulation T, e is *flippable* when two triangles incident to e in T form a *convex* quadrilateral Q. *Flipping* e in T generates a new triangulation by replacing e of T with the other diagonal of Q. Such operation is called *improving flip* if the triangulation obtained by

flipping e has a smaller index than the previous one, and e is called *improving flippable*. Now let us show that the greedy flipping property of the constrained triangulations.

Lemma 2. Let T be the F-constrained triangulation with $T \neq CT^*$ and c be the critical vertex of T. Then there exists at least one improving flippable edge incident to c in $T \setminus CT^*$.

Proof. Let $F(c)$ and $CT^*(c)$ be sets of edges of F and CT^* whose left endpoints coincide with c, respectively. And let (c, c_0) and (c, c_k) be the upper and lower hull edges of c respecting F.

Now we show that T contains all edges of $CT^*(c)$. First let us show that $(c, c_0) \in T$ ((c, c_k) can be similarly proofed). Suppose that (c, c_0) is missing in T. Since T is a triangulation, T has some edge $e \in T \setminus CT^*$ intersecting (c, c_0). Then, from the empty region property of (c, c_0) discussed in Section 2.1, $l(e) < c$ holds, which implies that the vertex $l(e)$ is incident to $e(\neq CT^*)$ and contradicts that c is the critical vertex.

Next let us show that each edge $(c, v) \in CT^*(c)$ other than $F(c) \cup \{(c, c_0), (c, c_k)\}$ is contained in T. Suppose that (c, v) is missing in T. Then there exists some edge $e \in T \setminus CT^*$. Let $(c, c_0), (c, c_1), \ldots, (c, c_k)$ be the edges of $F(c) \cup \{(c, c_0), (c, c_k)\}$ arranged in clockwise ordering around c as denoted in Definition 3. Since $(c, v) \in CT^*(c)$, there exists a unique l with $0 \leq l \leq k-1$ of a convex hull, $H_l = conv(P_c \cap (c, c_l)^{-,0} \cap (c, c_{l+1})^{+,0})$, such that v is on the boundary of H_l. Then the edges $(c, c_l), (c, c_{l+1})$ and the part of the boundary edges of H_l (convex chain) from c_l to c_{l+1} forms a pseudo-triangle with three corners c, c_l and c_{l+1}. Since there exists no point of P inside of such pseudo-triangle, what e intersects (c, v) implies that e also intersects at least one of (c, c_l) and (c, c_{l+1}), which contradicts that T contains all edges of $F(c) \cup \{(c, c_0), (c, c_k)\}$. Hence T contains $CT^*(c)$.

Now let us show that there exists at least one improving flippable edge $e^* \notin CT^*$ incident to c. Since c is a critical vertex, T has at least one edge $(c, p) \notin CT^*$. Let (c, c'_0) and (c, c'_k) be a pair of edges of $CT^*(c)$ such that p exists between (c, c'_0) and (c, c'_k) and an angle $\angle c'_0 c c'_k$ is minimum for all pairs of edges in $CT^*(c)$ (see Fig. 4). Consider a set of edges in T incident to c between (c, c'_0) and (c, c'_k), and denote them by $(c, c'_1), \ldots, (c, c'_{k-1})$ in clockwise order around c. Note that $(c, c'_j) \in T \setminus CT^*$ holds for all $j = 1, \ldots, k-1$, and no vertex of $\{c'_1, \ldots, c'_{k-1}\}$ is inside of the triangle $\Delta c c'_0 c'_k$, since otherwise CT^* is not triangulated. Therefore, all edges $(c, c'_1), \ldots, (c, c'_{k-1})$ intersect the edge (c'_0, c'_k). Let c'_{j^*} be a vertex furthest from the line through c'_0 and c'_k among c'_j for $j = 1, \ldots, k-1$. Then a

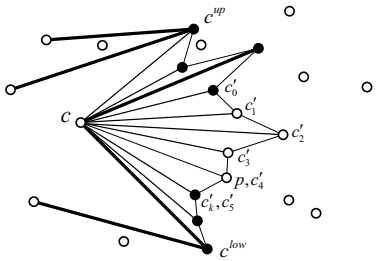


Fig. 4. Existence of an improving flippable edge (c, c'_{j^*}). Bold edges represent the edges of F, and black vertices represent the vertices incident to c in CT^*. In this figure, a quadrilateral $cc'_1c'_2c'_3$ is convex and (c, c'_2) is improving flippable.

quadrilateral $cc'_{j^*-1}c'_{j^*}c'_{j^*+1}$ is convex, and flipping (c, c'_{j^*}) to (c'_{j^*-1}, c'_{j^*+1}) produces a new triangulation with a smaller index than that of the previous one because $c < c'_{j^*-1}$ and $c < c'_{j^*+1}$ hold. □

Theorem 3. *Every F-constrained triangulation T can be transformed into CT^* by $O(n^2)$ flips.*

Proof. From Lemma 2, T has an improving flippable edge incident to the critical vertex, and flipping such edge reduces the index of T. Since the number of distinct indices is $O(n^2)$, T can be transformed into CT^* by $O(n^2)$ improving flips. □

3 Constrained Non-crossing Spanning Tree

Let F be a non-crossing edge set on P, and we assume that F is a forest. In this section we show that a set of F-constrained spanning trees on P, denoted by $\mathcal{SP}$, is connected by $O(n^2)$ flips.

Let $E = \{e_1 \prec e_2 \prec \ldots \prec e_m\}$ and $E' = \{e'_1 \prec e'_2 \prec \ldots \prec e'_m\}$ be lexicographically ordered edge lists. Then E is lexicographically smaller than E' if $e_i \prec e'_i$ for the smallest i such that $e_i \neq e'_i$.

Consider the F-constrained smallest indexed triangulation (F-CSIT), which is denoted by $T(F)$ in what follows. *F-constrained smallest indexed spanning tree (F-CSISP)* is a F-constrained spanning tree which is a subset of $T(F)$, and we denote a set of all F-CSISPs by $\mathcal{CSISP}$. Let SP^* be a spanning tree consisting of lexicographically smallest edge list among $\mathcal{CSISP}$. The following lemma holds from the known fact about matroid (see e.g. [20]):

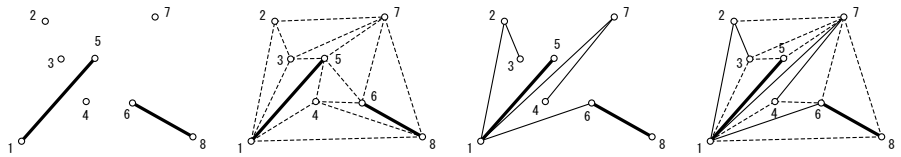


Fig. 5. (a) F, (b) $T(F)$, (c) SP, and (d) $T(SP)$, where bold edges represent F and dotted edges represent added edges for triangulations.

Lemma 4. Every non-crossing spanning tree of $\mathcal{CSISP}$ can be transformed to SP^* by at most $n-1$ flips.

Proof. Let $SP \in \mathcal{CSISP}$. Then SP is a base of graphic matroid restricted to the edge set of $T(F)$. Since all bases are connected via base exchange, the statement holds. In fact, planarity is maintained since any $SP \in \mathcal{CSISP}$ is subset of $T(F)$.

Now we will define an index for each spanning tree $SP \notin \mathcal{CSISP}$ to represent how far it is from one of $\mathcal{CSISP}$. For each F-constrained triangulation T we have defined its index with respect to F by $index_F(T) = (n - c_F, d_F)$, which represents how far T is from $T(F)$, i.e. c_F is the smallest vertex in T incident to edges not contained in $T(F)$, and d_F is the number of edges incident to c_F not contained in $T(F)$. We associate SP-constrained smallest indexed triangulation $T(SP)$ with each spanning tree SP, and define an index of SP (denoted by $index(SP) = (c_{SP}, d_{SP})$) as $index(SP) = index_F(T(SP))$, We also call c_{SP} the critical vertex of SP. Fig. 5 shows an example of SP whose critical vertex is 1 and $index(SP)$ is $(7, 2)$.

Let $SP(i)$ and $T(SP; i)$ be the edges of SP and $T(SP)$ whose left endpoints coincide with $i \in P$, respectively. The next observation immediately follows from the definition of CSIT.

Observation 5. For $i \in P$ all edges of $T(SP; i) \setminus (SP(i) \cup \{(i, i_0), (i, i_k)\})$ are not flippable in $T(SP)$, where (i, i_0) and (i, i_k) are upper and lower hull edges of i respecting SP.

Proof. It is clear from the definition of CSIT (Definition 1) that the added edges for obtaining $T(SP)$ from SP are not flippable in $T(SP)$ except for the upper and lower hull edges.

Then we derive the followings from Theorem 3 and Observation 5:

Lemma 6. Let $SP \notin \mathcal{CSISP}$ and c be the critical vertex of SP. Then (i) there exists at least one improving flippable edge in $T(SP) \setminus T(F)$, and (ii) an edge $e \in T(SP)$ is improving flippable if and only if e is flippable and $e \in SP(c) \setminus \{(c, c_0), (c, c_k)\}$, where (c, c_0) and (c, c_k) are upper and lower hull edges of c respecting SP.

Proof. Since $T(SP) \neq T(F)$ holds, from Lemma 2, there exists at least one improving flippable edge incident to the critical vertex, which implies (i).

Let us show (ii). From the definition of the index of $T(SP)$, we notice that flipping an improving flippable edge decreases the number of edges incident to c. Hence all improving flippable edges must be incident to c. From Observation 5 all edges of $T(c) \setminus (SP(c) \cup \{(c, c_0), (c, c_k)\})$ are not flippable. Then the proof is completed by showing that none of (c, c_0) and (c, c_k) are improving flippable edges. Suppose that (c, c_0) is improving flippable, there exists two triangles Δcc_0v_1 and Δcc_0v_2 incident to (c, c_0) in T and $c < v_1$ and $c < v_2$ hold. Otherwise, assuming $v_1 < c$, v_1 become the critical vertex and index increases after replacing (c, c_0) to (v_1, v_2). However, either v_1 or v_2 is in the empty region of (c, c_0), which is a contradiction. (The same remark holds for (c, c_k).)

Lemma 7. Every F-constrained non-crossing spanning tree $SP \notin \mathcal{CSISP}$ can be transformed into a spanning tree in $\mathcal{CSISP}$ by at most $O(n^2)$ flips.

Proof. Let c be a critical vertex of SP. From Lemma 6 there exists an edge $e_1 = (c, c^*) \in SP \setminus F$ such that e_1 is improving flippable in $T(SP)$. There exist two vertices, denote c_l^* and c_r^*, incident to both c^* and c in $T(SP)$. When removing e_1 from SP, the set of vertices of $SP - e_1$ is partitioned into two components, where c^* and c belong to different components, and c_l can belong to only one of them. Therefore adding one of (c, c_l^*) or (c_l^*, c^*) to $SP - e_1$, we obtain a new non-crossing spanning tree SP'. Note that index of $T(SP')$ is smaller than that of $T(SP)$ because $T(SP')$ does not have (c, c^*) but has (c_l^*, c_r^*) instead and the critical degree decreases by one. Repeating this procedure, the underlying triangulation eventually reaches the smallest indexed triangulation which is the required F-CSISP. Since the number of distinct indices is $O(n^2)$, $O(n^2)$ flips occur.

From Lemmas 4 and 7 we conclude this section with the following theorem:

Theorem 8. *Every F-constrained non-crossing spanning tree is connected by at most $O(n^2)$ flips.*

4 Enumerating Constrained Non-crossing Spanning Trees

Let SP^* be an F-CSISP with the lexicographically smallest edge list. And let I_{SP} be a set of improving flippable edges in SP. We define the following *parent function* $f : \mathcal{SP} \setminus \{SP^*\} \to \mathcal{SP}$ based on the results of the previous section.

Definition 2. *(Parent function) Let $SP \in \mathcal{SP}$ with $SP \neq SP^*$, and c be a critical vertex of SP. $SP' = SP - e_1 + e_2$ is the parent of SP, where*
Case 1: $SP \in \mathcal{CSISP}$,
• $e_1 = \max\{e \mid e \in SP \setminus SP^*\}$, *and*
• $e_2 = \min\{e \in SP^* \setminus SP \mid SP - e_1 + e \in \mathcal{SP}\}$,
Case 2: $SP \notin \mathcal{CSISP}$,
• $e_1 = (c, c^*) = \min\{e \in SP \setminus F \mid e \in I_{SP}\}$, *and*
• e_2 *is either* (c, c_l^*) *or* (c_l^*, c^*) *such that* $SP - e_1 + e_2 \in \mathcal{SP}$, *where* c_l^* *is a vertex such that* $\Delta cc^*c_l^*$ *exists in* $T(SP)$ *with* $\Delta(c, c^*, c_l^*) > 0$.

Note that $I_{SP} \neq \emptyset$ and I_{SP} is the subset of $SP \setminus F$ incident to c from Lemma 6. Therefore e_1 in Case 2 of Definition 2 always exists. There exist two vertices, c_l^* and c_r^*, incident to both c^* and c in $T(SP)$ with $\Delta(c^*, c, c_l^*) > 0$ and $\Delta(c^*, c, c_r^*) < 0$, respectively. Here, we adopt c_l^* in Definition 2 in order to define the unique parent. In Fig. 6 we show how the parent function works for $SP \in \mathcal{SP} \setminus \mathcal{CSISP}$ with $index(SP) = (6, 2)$ and $F = \emptyset$. Removing 27 and adding 57 we obtain a new spanning tree with $index = (6, 1)$. From Lemmas 4 and 7 these parent-child relationships form the search tree of $\mathcal{SP}$ explained in Section 1. To simplify the notations, we denote the parent function depending on Cases 1 and 2 by $f_1 : \mathcal{CSISP} \setminus \{SP^*\} \to \mathcal{CSISP}$ and $f_2 : \mathcal{SP} \setminus \mathcal{CSISP} \to \mathcal{SP}$, respectively.

For $SP' \in \mathcal{SP}$ the local search is given by an *adjacency function*, Adj, defined as follows:

$$Adj(SP', e_{\rm rem}, e_{\rm add}) := \begin{cases} SP' - e_{\rm rem} + e_{\rm add} & \text{if } SP' - e_{\rm rem} + e_{\rm add} \in \mathcal{SP}, \\ null & \text{otherwise,} \end{cases}$$

where $e_{\rm rem} \in SP' \setminus F$ and $e_{\rm add} \in K_n \setminus SP'$. Let $\mathsf{elist}_{SP'}$ and elist_{K_n} be the list of edges of SP' and K_n ordered lexicographically, and let $\mathsf{elist}_{SP'}(i)$ and $\mathsf{elist}_{K_n}(i)$ be the i-th element of $\mathsf{elist}_{SP'}$ and elist_{K_n}, respectively. We also denote the above defined adjacency function by $Adj(SP', i, j)$ for which $e_{\rm rem} = \mathsf{elist}_{SP'}(i)$ with $e_{\rm rem} \notin F$ and $e_{\rm add} = \mathsf{elist}_{K_n}(j)$ with $e_{\rm add} \notin SP'$. Then, based on the algorithm in [6, 7], we describe our algorithm in Fig. 7.

Both the parent function and the adjacency function need $O(n)$ time for each process by simply checking non-crossing property. Then, the

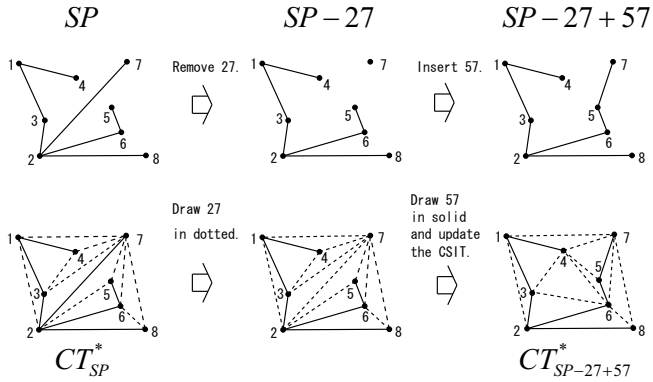


Fig. 6. An example of the parent function for $SP \notin \mathcal{CSISP}$, where $F = \emptyset$.

while-loop from line 4 to 17 has $|SP'| \cdot |K_n|$ iterations which require $O(n^4)$ time if simply checking lines 8 and 10. In the following sections we give an improved $O(n^2)$ time algorithm. Especially, we will show that the inner-while loop from lines 6 to 16 can be implemented in $O(n)$ time. Hence, from the correctness of the algorithm given in the following sections, we obtain the theorem:

Theorem 9. *The set of all F-constrained non-crossing spanning trees on a given point set can be reported in $O(n^2)$ time per output using $O(n)$ space.*

5 Correctness and Analysis of the Algorithm

We will devote this section to the proof of Theorem 9. Let SP and SP' be two distinct spanning trees for which $SP = Adj(SP', e_{\rm rem}, e_{\rm add})$ for $e_{\rm rem} \in SP' \setminus F$ and $e_{\rm add} \in K_n \setminus SP'$. Now we want to efficiently determine whether SP' is the parent of SP. More specifically we want to characterize the pair of edges, $e_{\rm rem}$ and $e_{\rm add}$, satisfying either $f_1(SP) = SP'$ or $f_2(SP) = SP'$. Lemma 10 and Lemma 12 show necessary and sufficient conditions for which $e_{\rm rem}$ and $e_{\rm add}$ satisfy $f_1(SP) = SP'$ and $f_2(SP) = SP'$, respectively. Lemma 11 shows that for each $e_{\rm rem} \in SP' \setminus F$ we can enumerate, in $O(n)$ time with $O(n)$ space, a set of edges E_1 satisfying all conditions of Lemma 10. Lemmas 18 and 19 show that for each $e_{\rm rem} \in SP' \setminus F$ we can enumerate, in $O(n)$ time with $O(n)$ space, a set of edges E_2 satisfying the conditions of Lemma 12. Then we can enumerate all the edge pairs of $(e_{\rm rem}, e_{\rm add})$ such that $SP' - e_{\rm rem} + e_{\rm add}$ is child of SP' are obtained in linear time for each $e_{\rm rem}$. Since the number of possible elements for $e_{\rm rem}$ is $O(n)$, Theorem 9 holds.

Algorithm Enumerating F-constrained non-crossing geometric spanning trees.

```
1: SP* := F-CSISP with lexicographically smallest edge list;
2: SP' := SP*; i, j := 0; Output(SP');
3: repeat
4:    while i ≤ |SP'| do
5:       i := i + 1;
6:       while j ≤ |K_n| do
7:          j := j + 1;
8:          if elist_{SP'}(i) ∉ F, elist_{K_n}(j) ∉ SP' and Adj(SP', i, j) ≠ null then
9:             SP := Adj(SP', i, j);
10:            if f_1(SP) = SP' or f_2(SP) = SP' then
11:               SP' := SP; i, j := 0;
12:               Output(SP');
13:               go to line 4;
14:            end if
15:         end if
16:      end while
17:   end while
18:   if SP' ≠ SP* then
19:      SP := SP';
20:      if SP ∈ CSISP then SP' := f_1(SP);
21:      else SP' := f_2(SP);
22:      determine integer pair (i, j) such that Adj(SP', i, j) = SP;
23:      i := i − 1;
24:   end if
25: until SP' = SP*, i = |SP'| and j = |K_n|;
```

Fig. 7. Algorithm for enumerating F-constrained non-crossing geometric spanning trees.

5.1 Checking $f_1(Adj(SP', e_{\rm rem}, e_{\rm add}) = SP'$

First we show the following lemma which contributes to efficient check of whether $f_1(SP) = SP'$ holds or not.

Lemma 10. Let SP and SP' be two distinct F-CSISP which are subgraphs of $T(F)$ for which $SP = Adj(SP', e_{\rm rem}, e_{\rm add})$ for $e_{\rm rem} \in SP' \setminus F$ and $e_{\rm add} \in K_n \setminus SP'$. Then, $f_1(SP) = SP'$ holds if and only if $e_{\rm rem}$ and $e_{\rm add}$ satisfy the following conditions:

(A) $e_{\rm rem} \in SP^*$,
(B) $e_{\rm add} \in T(F) \setminus (SP^* \cup SP')$,
(C) $e_{\rm rem} \prec \min\{e \in SP^* \setminus SP' \mid SP' - e_{\rm rem} + e \in \mathcal{SP}\}$,
(D) $e_{\rm add} \succ \max\{e \mid e \in SP' \setminus SP^*\}$.

Proof. We first remark that all SP, SP' and SP^* are subgraphs of SIT. Then all edges in $SP \cup SP' \cup SP^*$ are non-crossing.

("only if"-part.) Since $f_1(SP) = SP'$ holds, $e_{\rm rem}$ and $e_{\rm add}$ must be chosen as e_2 and e_1 in Case 1 of Definition 2. From Definition 2, $e_{\rm add}(= e_1) \in SP \setminus SP^*$ holds. Since $SP \in \mathcal{SP}$, $SP \subset T(F)$ and $e_{\rm add} \in T(F) \setminus SP^*$

holds, proving (B). Similarly since $e_{\rm rem}(= e_2) \in SP^* \setminus SP \subset SP^*$, we have (A). From $e_{\rm rem} = e_2$, we have

$$SP' - e_{\rm rem} = (SP - e_1 + e_2) - e_{\rm rem} = SP - e_1. \tag{1}$$

Let $e' = \min\{e \in SP^* \setminus SP' \mid SP' - e_{\rm rem} + e \in \mathcal{SP}\}$. Suppose (C) does not hold. Then $e' \prec e_{\rm rem}$ holds. (Note that the equality does not hold since $e_{\rm rem} \in SP' \setminus F$.) We have

$$\begin{aligned} e' &= \min\{e \in SP^* \setminus SP' \mid SP' - e_{\rm rem} + e \in \mathcal{SP}\} \\ &= \min\{e \in SP^* \setminus (SP - e_1 + e_2) \mid SP - e_1 + e \in \mathcal{SP}\} && \text{(from (1))} \\ &= \min\{e \in SP^* \setminus SP \mid SP - e_1 + e \in \mathcal{SP}\} && \text{(from } e' \prec e_2 \prec e_1\text{)}. \end{aligned}$$

Thus, e' would have been selected instead of $e_{\rm rem}$ when the parent function f_1 is applied to SP, which contradicts $e_{\rm rem} = e_2$. Hence, (C) holds.

Finally, let $e'' = \max\{e \mid e \in SP' \setminus SP^*\}$, and suppose that (D) does not hold and $e_{\rm add} \prec e''$ holds. (Note that the equality does not hold since $e_{\rm add} \notin SP'$.) Since $e_2 \prec e_1 = e_{\rm add} \prec e''$, we have

$$\begin{aligned} e'' &= \max\{e \mid e \in SP' \setminus SP^*\} \\ &= \max\{e \mid e \in (SP - e_1 + e_2) \setminus SP^*\} \\ &= \max\{e \mid e \in SP \setminus SP^*\}. \end{aligned}$$

Then e'' would have been selected instead of $e_{\rm add}$ when the parent function f_1 is applied to SP, which contradicts $e_{\rm add} = e_1$. Thus, (D) holds.

("if"-part.) From (A) and (B), $SP = SP' - e_{\rm rem} + e_{\rm add} \in \mathcal{CLOSP}$. Since $e_{\rm rem} \in SP^*$ from (A), (D) implies that

$$\begin{aligned} e_{\rm add} &\succ \max\{e \mid e \in SP' \setminus SP^*\} \\ &= \max\{e \mid e \in (SP + e_{\rm rem} - e_{\rm add}) \setminus SP^*\} \\ &= \max\{e \mid e \in (SP - e_{\rm add}) \setminus SP^*\}. \end{aligned}$$

Thus, $e_{\rm add} = \max\{e \mid e \in SP \setminus SP^*\}$ holds, and hence f_1 chooses $e_{\rm add}$ for an edge e_1 to be deleted from SP. From this we have $SP - e_1 = SP' - e_{\rm rem} + e_{\rm add} - e_1 = SP' - e_{\rm rem}$. Since $e_{\rm add} \notin SP^*$ from (B), (C) implies

$$\begin{aligned} e_{\rm rem} &\prec \min\{e \in SP^* \setminus SP' \mid SP' - e_{\rm rem} + e \in \mathcal{SP}\} \\ &= \min\{e \in SP^* \setminus (SP + e_{\rm rem} - e_{\rm add}) \mid SP - e_1 + e \in \mathcal{SP}\} \\ &= \min\{e \in SP^* \setminus (SP + e_{\rm rem}) \mid SP - e_1 + e \in \mathcal{SP}\}. \end{aligned}$$

Since $e_{\rm rem} \in SP^* \setminus SP$, $e_{\rm rem} = \min\{e \in SP^* \setminus SP \mid SP - e_1 + e \in \mathcal{SP}\}$. Thus, f_1 chooses $e_{\rm rem}$ for an edge to be added, and $f_1(SP)$ returns SP'.

Lemma 11. Let $SP' \in \mathcal{SP}$ and $e_{\rm rem} \in SP' \setminus F$. Then the conditions (A) and (C) in Lemma 10 can be checked in linear time for each $e_{\rm rem}$. Moreover we can enumerate a set of edges $E_1 \subset K_n \setminus SP'$ that satisfy the conditions (B) and (D) in linear for each $e_{\rm rem}$ with linear space,.

Proof. We assume that SP^* and $T(F)$ are pre-computed in the preprocessing phase before enumeration, and the sets of edges in SP', SP^* and $T(F)$ are maintained in lexicographically ordered edge lists. So the condition (A) can be checked in $O(\log n)$ time for each $e_{\rm rem}$. For the edges to be added, from the condition (B), it is sufficient to check only the edges of $T(F) \setminus (SP^* \cup SP')$. Using linear time we compute a list of edges of $T(F) \setminus (SP^* \cup SP')$ stored lexicographically, and denote it by $\mathsf{elist}_{\rm add}$. And we compute the edge $e' = \min\{e \in SP^* \setminus SP' \mid SP' - e_{\rm rem} + e \in \mathcal{SP}\}$ in $O(n)$ time by checking each edge of $SP^* \setminus SP'$ one by one whether it spans the different components of $SP' - e_{\rm rem}$. (Note that all edges in $SP^* \cup SP'$ are non-crossing.) Also we can compute the edge $e'' = \max\{e \mid e \in SP' \setminus SP^*\}$ in $O(n)$ time. By using e' and e'' the condition (C) can be checked in $O(1)$ time and the set of edges in $\mathsf{elist}_{\rm add}$ satisfying the condition (D) is obtained in $O(n)$ time. Thus, the lemma follows.

5.2 Checking $f_2(Adj(SP', e_{\rm rem}, e_{\rm add})) = SP'$

Next we will explain how we can efficiently check whether $f_2(SP) = SP'$ holds or not. Consider the situation that we remove $e_{\rm rem} = (a, b)$ with $a < b$ from SP' and add $e_{\rm add} = (x, y)$ (or (y, x)) to $SP' - e_{\rm rem}$ such that $SP' - e_{\rm rem} + e_{\rm add} \in \mathcal{SP}$. From Definition 2, $e_{\rm rem}$ and $e_{\rm add}$ must share exactly one endpoint. Then, we assume without loss of generality that either $y = a$ or $y = b$ holds. Especially, we will characterize the other endpoint x of $e_{\rm add}$ depending on $e_{\rm rem} = (a, b)$. The endpoint of $e_{\rm add}$ other than y is denoted by x throughout this section.

Now let us start to characterize the edges $e_{\rm rem}$ and $e_{\rm add}$.

Lemma 12. Let SP and SP' be two distinct F-constrained spanning trees for which $SP = Adj(SP', e_{\rm rem}, e_{\rm add})$ for $e_{\rm rem} = (a, b) \in SP' \setminus F$ and $e_{\rm add} \in K_n \setminus SP'$ that is incident to either a or b. Then, $f_2(SP) = SP'$ holds if and only if $e_{\rm rem}$ and $e_{\rm add}$ satisfy the following conditions:

(A) $e_{\rm add} = \min\{e \in SP \setminus F \mid e \in I_{SP}\}$,
(B) $l(e_{\rm add}) \leq a$, (i.e. $l(e_{\rm add}) \leq l(e_{\rm rem})$),
(C) $a \in {e_{\rm add}}^+$ (or $b \in {e_{\rm add}}^+$) holds when $e_{\rm rem}$ and $e_{\rm add}$ share b (or a, respectively), and
(D) the triangle Δabx exists in $T(SP)$.

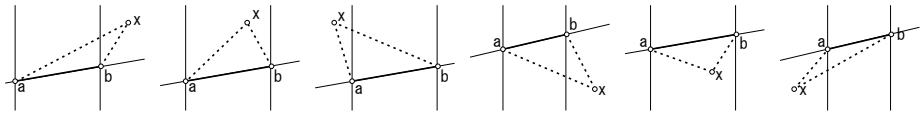


Fig. 8. Six cases of relative position of x.

Proof. The necessary and sufficient conditions for $f_2(SP) = SP'$ is that $e_{\mathrm{rem}} = e_2$ and $e_{\mathrm{add}} = e_1$ hold, where e_1 and e_2 are those defined in Case 2 of Definition 2. Then, replacing e_1 and e_1 of Definition 2 by e_{add} and e_{rem}, respectively, we obtain the conditions (A), (C) and (D). Moreover, we have $l(e_1) \leq l(e_2)$ since e_1 is a improving flippable edge in $T(SP)$ from Definition 2 and we know that all vertices of two triangles incident to e_1 in $T(SP)$ is lexicographically larger than $l(e_1)$. Thus (B) holds.

Remember that e_{rem} and e_{add} must share one endpoint, so the number of candidate edges, e_{add}, satisfying Lemma 12 for each $e_{\mathrm{rem}} \in SP' \setminus F$ is $O(n)$. However, Lemma 12 does not directly provide an efficient algorithm, and we need more precise analysis for each condition in Lemma 12.

Checking the conditions (B), (C) and (D) Now let us first analyze the conditions (B) and (C) in Lemma 12. Considering the intersections of open halfspaces defined by the three lines, one is passing through a and b and the others are through a and b perpendicular to x-axis, the following six cases are possible depending on the position of x (see Fig 8):

Case 1: $x \in R^{+++}$, where $R^{+++} = (a,b)^+ \cap (a)^+ \cap (b)^+$,
Case 2: $x \in R^{++-}$, where $R^{++-} = (a,b)^+ \cap (a)^+ \cap (b)^-$,
Case 3: $x \in R^{+--}$, where $R^{+--} = (a,b)^+ \cap (a)^- \cap (b)^-$,
Case 4: $x \in R^{-++}$, where $R^{-++} = (a,b)^- \cap (a)^+ \cap (b)^+$,
Case 5: $x \in R^{-+-}$, where $R^{-+-} = (a,b)^- \cap (a)^+ \cap (b)^-$,
Case 6: $x \in R^{---}$, where $R^{---} = (a,b)^- \cap (a)^- \cap (b)^-$.

Neither of Case 1 or 2 happen. It is because that, from the condition (B), e_{add} must be (a,x) in these cases. However, if so, $b \notin {e_{\mathrm{add}}}^+$ and the condition (C) is not satisfied. Simply checking the conditions (B) and (C) for each case, the endpoint of e_{add} other than x that is either a or b is uniquely determined depending on the position of x as follows:

Lemma 13. Let SP, SP', $e_{\mathrm{rem}} = (a,b)$ and e_{add} be as in Lemma 12. The conditions (B) and (C) hold if and only if $x \in R^{+--} \cup R^{-++} \cup R^{-+-} \cup R^{---}$ and x satisfies the following conditions:

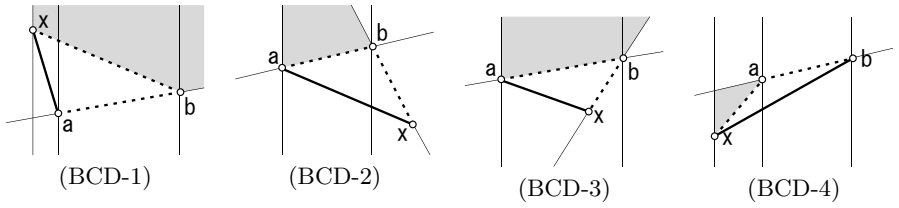


Fig. 9. Each shaded region represents the forbidden region to be a point in P visible to (a, b) when removing (a, b) and adding the bold edge.

(BC-1) $e_{\text{add}} = (x, a)$ when $x \in R^{+--}$,
(BC-2) $e_{\text{add}} = (a, x)$ when $x \in R^{-++}$,
(BC-3) $e_{\text{add}} = (a, x)$ when $x \in R^{-+-}$,
(BC-4) $e_{\text{add}} = (x, b)$ when $x \in R^{---}$.

Proof. Immediate from the conditions (B) and (C).

For a point p and an edge $e = (a, b)$, p is *completely visible* from e with respect to SP' when p is visible from all points on (a, b) with respect to the edges of SP'. We denote a set of such points by $CV_{SP'}(e)$ or $CV_{SP'}(a, b)$. For each case from (BC-1) through (BC-4), the following lemma tells when condition (D) holds:

Lemma 14. Let SP, SP', $e_{\text{rem}} = (a, b)$ and e_{add} be as in Lemma 12. The conditions (B), (C) and (D) hold if and only if $x \in CV_{SP'}(a, b)$ and one of the following conditions is satisfied:

(BCD-1) when $x \in R^{+--}$, $e_{\text{add}} = (x, a)$ and $CV_{SP'}(a, b) \cap ((x)^+ \cap (a, b)^+ \cap (x, b)^+) = \emptyset$,
(BCD-2) when $x \in R^{-++}$, $e_{\text{add}} = (a, x)$ and $CV_{SP'}(a, b) \cap ((a)^+ \cap (a, b)^+ \cap (b, x)^-) = \emptyset$,
(BCD-3) when $x \in R^{-+-}$, $e_{\text{add}} = (a, x)$ and $CV_{SP'}(a, b) \cap ((a)^+ \cap (a, b)^+ \cap (x, b)^+) = \emptyset$,
(BCD-4) when $x \in R^{---}$, $e_{\text{add}} = (x, b)$ and $CV_{SP'}(a, b) \cap ((x)^+ \cap (a, b)^- \cap (x, a)^+) = \emptyset$.

Proof. Fig. 9 shows an example of *forbidden region* in which there exists no point of P completely visible to (a, b) with respect to SP'. (Since $x \in CV_{SP'}(a, b)$ is required in Lemma 14, you may add a triangle Δabx to each forbidden region.) We here show only the case (BCD-1) and omit the other cases in order to avoid similar arguments.

("only if"-part.) From Lemma 13, e_{add} must be (x, a). Let $SP = SP' - e_{\text{rem}} + e_{\text{add}}$. We show that if there exists some point in $(x)^+ \cap (a, b)^+ \cap$

$(x,b)^+$ completely visible from (a,b), the updated triangulation $T(SP)$ contains some edge intersecting (x,b). Let $p \in (x)^+ \cap (a,b)^+ \cap (x,b)^+$ such that the angle $\angle pab$ around a is largest. When $x < p < a$ holds, (p,a) is the lower hull edge of p in SP since $(x,a) \in T(SP)$, and hence $T(SP)$ has (p,a). When $a < p$ holds, (a,p) is the upper hull edge of a in SP since (a,p) has the largest angle around a among the edges connecting a and the points of $V_{SP}(a,P_a)$. Then $T(SP)$ has an edge (a,p). In both cases $T(SP)$ has some edge intersecting (x,b). Therefore we cannot have Δabx in $T(SP)$.

("if"-part.) When the conditions in the statement hold, we can observe that there exists no edge in $T(SP)$ intersecting Δabx except their endpoints (see Fig. 9). In fact, $b \in V_{SP}(x,P_x)$ holds, and $T(SP)$ contains Δabx.

Now let us analyze the time complexity for efficiently checking the conditions (B), (C) and (D) in Lemma 12. Let E_{BCD} be a subset of $K_n \setminus SP'$ satisfying the conditions (B), (C) and (D) in Lemma 12. First we will describe how to obtain $CV_{SP'}(a,b)$ for an edge $(a,b) \in SP$ in linear time in Lemma 17. And then we will show how to enumerate E_{BCD} in linear time in Lemma 18.

For a simple polygon $\mathcal{P}$ and a point p in the inside of $\mathcal{P}$, the *visibility polygon* of p in $\mathcal{P}$ is the set of all points visible in $\mathcal{P}$ from p with respect to the edges of $\mathcal{P}$. For a set of line segments F and a point p, the *visibility polygon* of p in (the arrangement of) F is defined by $\mathcal{VP}_F(p) = \{q \in \mathbf{R}^2 \mid q \text{ is visible from } p \text{ with respect to } F\}$. Similarly, replacing the the term "visible" by "completely visible", *complete visibility polygon* is defined. The following two facts are known:

Fact 15. ([18, 19]) Let $\mathcal{P}$ be a simple polygon. Then a visibility polygon of a point p in $\mathcal{P}$ can be found in linear time.

Fact 16. ([5]) Let F be a set of line segments. Then a complete visibility polygon of a line segment $(a,b) \in F$ is the visibility polygon of b in $\mathcal{VP}_F(a)$.

From Facts 15 and 16, $CV_{SP}(a,b)$ can be found in linear time if $\mathcal{VP}_{SP}(b)$ is obtained in linear time. In general, it is known that it takes $O(n \log n)$ time to compute the visibility polygon of a point in the arrangement of line segments. However, computing the visibility polygon of a point in a spanning tree in linear time is possible as will be shown in the following lemma.

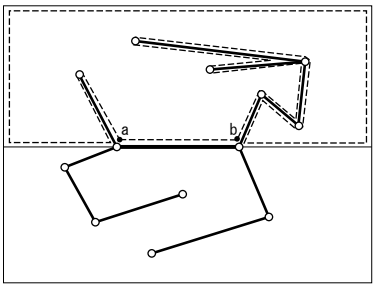


Fig. 10. Spanning tree SP contained in large rectangle R. Dotted simple polygon $\mathcal{P}$ is obtained by tracing the edges of SP, R and the line through a and b.

Lemma 17. Let SP be a non-crossing spanning tree on a point set P. Then, a completely visibility polygon of an edge $(a, b) \in SP$ can be found in linear time with linear space.

Proof. Suppose that SP is contained in the large rectangle R, and let us find a completely visible polygon of (a, b) inside $(a, b)^{+,0}$. Let L_{ab} be a straight line through a and b. From Fact 16, if we obtain a visibility polygon $VP_{SP \cup R \cup l_{ab}}(b)$ that is a visibility polygon of b in the edge set SP bounded by R and l_{ab}, in linear time, a completely visible polygon of (a, b) is found in linear time from Fact 15. We can view the problem of finding a visibility polygon of b in the edge set SP as the one of finding a visibility polygon in the polygon $\mathcal{P}$, where $\mathcal{P}$ is a simple polygon obtained by tracing the edges of SP, R and the line through a and b as in Fig. 10. Thus, the lemma follows.

Lemma 18. We can enumerate the edge set $E_{BCD} \subset K_n \setminus SP'$ each of which satisfies the conditions (B)(C) and (D) in Lemma 12 in the lexicographically sorted edge list in $O(n)$ time.

Proof. From Lemma 14, it is sufficient to determine those among the edges each of whose one endpoint is in $CV_{SP'}(a, b)$ and the other is either a or b. From Lemma 17, we can find the set of vertices, $CV_{SP'}(a, b)$, completely visible from (a, b) in linear time. Moreover the algorithm described in Lemma 17 computes the visibility polygon of a in the visibility polygon of b. It is well known that the visibility polygon is *star shaped* (see e.g. [5]). Then we obtain, from the visibility polygon of b, two sorted lists of vertices of $CV_{SP'}(a, b)$, one contains the vertices of $CV_{SP'}(a, b) \cap (a, b)^+$ and the other contains those of $CV_{SP'}(a, b) \cap (a, b)^-$ in $O(n)$ time by the

angles $\angle bax$ around a for elements $x \in CV_{SP'}(a,b)$. Denote these lists by $\mathsf{cv\text{-}list}^+$ and $\mathsf{cv\text{-}list}^-$, respectively.

Let us first focus our attention on the points in $\mathsf{cv\text{-}list}^+$. From Lemma 13 the points in $R^{+++} \cup R^{++-}$ (which correspond to Cases 1 and 2 in Fig. 8) should be removed from $\mathsf{cv\text{-}list}^+$. Then, for each point $x \in \mathsf{cv\text{-}list}^+$, the edge to be chosen as e_{add} is determined, i.e. (x,a) must be chosen as e_{add} when $x \in R^{+--}$. So, let us consider how to efficiently check the remaining part of the condition (BCD-1) of Lemma 14, that is, how to enumerate the set of $x \in \mathsf{cv\text{-}list}^+$ for which $CV_{SP'}(a,b) \cap ((x)^+ \cap (a,b)^+ \cap (x,b)^+) = \emptyset$ holds in $O(n)$ time. It can be done easily by preprocessing before removing the points in $R^{+++} \cup R^{++-}$ from $\mathsf{cv\text{-}list}^+$. In this preprocessing phase, we calculate the largest x-coordinate point x_1 among $\mathsf{cv\text{-}list}^+$. Then, for $x \in R^{+--}$, $CV_{SP'}(a,b) \cap ((x)^+ \cap (a,b)^+ \cap (x,b)^+) = \emptyset$ holds if and only if $x_1 \notin (x)^+ \cap (a,b)^+ \cap (x,b)^+$ holds. Actually, only x_1 satisfies the condition (BCD-1) of Lemma 14 if $x_1 \in R^{+--}$, and there exists no point satisfying that condition otherwise.

Next let us consider the points in $\mathsf{cv\text{-}list}^-$. By the same way as in the above discussion for $\mathsf{cv\text{-}list}^+$, we can determine the edge to be chosen as e_{add} for each $x \in \mathsf{cv\text{-}list}^-$ depending on the position of x. In the preprocessing phase we calculate, in linear time, two points x_2 and x_3 such that x_2 is the point with the minimum angle of $\angle abx$ around b among $x \in CV_{SP'}(a,b) \cap R^{++-}$, and x_3 is the point with the minimum angle of $\angle abx$ around b among $x \in CV_{SP'}(a,b) \cap (R^{++-} \cup R^{+++})$. Each can be found in linear time by simply checking all points in $\mathsf{cv\text{-}list}^+$. By this, for each $x \in \mathsf{cv\text{-}list}^+$, we can check from scratch the conditions of Lemma 14 in $O(1)$ time as follows. **(BCD-2)**When $x \in R^{-++}$, $CV_{SP'}(a,b) \cap ((a)^+ \cap (a,b)^+ \cap (b,x)^-) = \emptyset$ holds if and only if $x_2 \in (b,x)^+$ (or x_2 does not exist), see Fig. 9. Therefore, the algorithm can check the condition (BCD-2) in Lemma 14 in $O(1)$ time by checking only whether $\Delta(b,x,x_2) > 0$ holds or not. **(BCD-3)**When $x \in R^{-+-}$, similarly the algorithm can check the condition (BCD-3) in Lemma 14 in $O(1)$ time by checking only whether $\Delta(x,b,x_3) < 0$ holds or not (see Fig. 14). **(BCD-4)**When $x \in R^{---}$, which is the last part of the algorithm, we assume that the above process for $x \in R^{-++} \cup R^{-+-}$ is already done and we have only the points of R^{---} in $\mathsf{cv\text{-}list}^-$. Remember that the points of $\mathsf{cv\text{-}list}^-$ is stored in the sorted ordering by the angle around a. The algorithm will check the condition (BCD-4) for each x in $\mathsf{cv\text{-}list}^-$ in the decreasing order of angle $\angle bax$, i.e. counterclockwise order around a in R^{---}. At each time we remember the largest point x_4 among the already scanned points. Then we have that $CV(a,b) \cap ((x)^+ \cap (a,b)^- \cap (x,a)^+) = \emptyset$ holds

if and only if $x_4 < x$ holds (see Fig.9). Thus, the algorithm can check the condition (BCD-4) in Lemma 14 in $O(1)$ time for each by checking only whether $x_4 < x$ holds or not. After checking it x_4 is updated if $x_4 < x$ holds in $O(1)$ time.

Finally we remark that we can obtain the lexicographically ordered edge list satisfying the condition (B)(C) and (D) in $O(n)$ from the edges obtained after performing the above processes. It is because that the edges obtained after performing them are sorted in the counterclockwise order around a, so the edges (a, x) satisfying (BCD-2) or (BCD-3) are already sorted in lexicographical order. The edges (x, a) satisfying (BCD-1) and (x, b) satisfying (BCD-4) have distinct left endpoints, so they can be sorted lexicographically by putting each edge e of them on the $l(e)$-th entry of one dimensional array of length n.

Checking the Condition (A) of Lemma 12 Our goal of this section is to show the following lemma:

Lemma 19. Let SP' be a F-constrained spanning tree and $e_{\rm rem} = (a, b) \in SP' \setminus F$. We can enumerate the edge set $E_2 \subseteq E_{BCD}$, in linear time with linear space, that satisfies the condition (A) in Lemma 12.

Based on four cases for edges E_{bcd} considered in Lemma 14, we can divide the proofs into the following six cases, where c' denote the critical vertex of SP':

Case 1 $l(e_{add}) = a$, which corresponds to (BCD-2) or (BCD-3), and one of **(1-a)** $c' < l(e_{\rm add})$, **(1-b)** $l(e_{\rm add}) = c'$ or **(1-c)** $l(e_{\rm add}) < c'$ holds, and

Case 2 $l(e_{add}) = x$, which corresponds to (BCD-1) or (BCD-4), and one of **(2-a)** $c' < l(e_{\rm add})$, **(2-b)** $l(e_{\rm add}) = c'$ or **(2-c)** $l(e_{\rm add}) < c'$ holds.

For simplicity, we abbreviate $SP' - e_{\rm rem}$ and $SP' - e_{\rm rem} + e_{\rm add}$ to SP'_- and SP'_{-+}, respectively. The condition (A) claims that $e_{\rm add}$ is the lexicographically smallest improving flippable edge in $T(SP'_{-+})$. Then we need to provide the efficient way to check the followings for each $e_{\rm add}$,

(A-1): $e_{\rm add}$ is improving flippable in $T(SP'_{-+})$, and

(A-2): there exists no improving flippable edge in $T(SP'_{-+})$ lexicographically smaller than $e_{\rm add}$.

We will supply how to check these in $O(1)$ time for each $e_{\rm add} \in E_{BCD}$ with fixed $e_{\rm rem}$ with one exception of Case 2-b. For this purpose, it must be sufficient to observe how a set of edges incident to a vertex i

in $T(SP'_{-+})$, (denoted by $T(SP'_{-+}; i)$), changes not for all $i \in P$ but only for $i \leq \{1, \ldots, l(e_{\rm add})\} \subseteq P$. Therefore let us consider how we can update a set of edge incident to i for $i \leq l(e_{\rm add})$, that is, from $T(SP'; i)$ to $T(SP'_{-}; i)$ when removing $e_{\rm rem} = (a, b)$ from $T(SP')$, and from $T(SP'_{-}; i)$ to $T(SP'_{-+}; i)$ when adding $e_{\rm add}$ to $T(SP'_{-})$. Of course, there exists no literature concerning how the CSIT can be constructed dynamically when removing and adding constrained edges. So we should present several Lemmas answering these matters although their rigorous proofs will be provided in the next section.

Now let us first consider removing one edge $e_{\rm rem}$ from $T(SP')$.

Lemma 20. $T(SP'_{-}; i) = T(SP'; i)$ holds for $i \in \{1, \ldots, l(e_{\rm rem}) - 1\}$.

This lemma implies that a set of edges incident to a vertex i on the left side of $l(e_{\rm rem})$ does not change at all when removing the constrained edge $e_{\rm rem}$. Similar argument is also hold when adding $e_{\rm add}$ to $T(SP'_{-})$.

Lemma 21. $T(SP'_{-+}; i) = T(SP'_{-}; i)$ holds for $i \in \{1, \ldots, l(e_{\rm add}) - 1\}$.

As a consequence, from $l(e_{add}) < l(e_{rem})$ (which is the condition (B)), we have the following observation:

Observation 22. $T(SP'_{-+}; i) = T(SP'; i)$ holds for $i \in \{1, \ldots, l(e_{\rm add}) - 1\}$.

Remember that we are now concerned about $T(SP'_{-+}; i)$ for $i \in \{1, \ldots, l(e_{add})\}$ for checking (A-1) and (A-2). Then we also need the following two lemmas for calculating $T(SP'_{-}; l(e_{\rm rem}))$ and $T(SP'_{-+}; l(e_{\rm rem}))$.

Lemma 23. It takes linear time to update $T(SP'_{-}; l(e_{\rm rem}))$ from $T(SP')$ with linear space.

Lemma 24. It takes linear time to update $T(SP'_{-+}; l(e_{\rm add}))$ from $T(SP'_{-})$ with linear space.

Notice that, to achieve the efficient algorithm, namely $O(n^2)$ time algorithm per output, we cannot call the method of Lemma 24 using $O(n)$ time for every $e_{\rm add}$. However this argument will tell us a useful tuition into (A-1) as follows.

Let us consider how the edge set $T(SP'_{-+}; l(e_{\rm add}))$ is obtained from $T(SP'_{-}; l(e_{\rm add}))$ when adding $e_{\rm add}$ to $T(SP'_{-})$, (assuming that we have $T(SP'_{-})$). We can find that there exists two edges, $e_1 = (l(e_{\rm add}), v_1)$ and $e_2 = (l(e_{\rm add}), v_2)$, of $T(SP'_{-}))$ such that $r(e_{\rm add})$ exists between e_1 and e_2 and an angle $\angle v_1 l(e_{\rm add}) v_2$ is minimum for all pairs of edges in $T(SP'_{-})$.

Then the update occurs locally only inside of $(l(e_{\text{add}}))^+ \cap e_1^- \cap e_2^+$. We compute two convex hulls, H_1 and H_2, of $P_{l(e_{\text{add}})} \cap e_1^- \cap e_{\text{add}}^+$ and $P_{l(e_{\text{add}})} \cap e_{\text{add}}^- \cap e_2^+$, and connect $l(e_{\text{add}})$ and each vertices visible from $l(e_{\text{add}})$ respecting H_1 and H_2, respectively. Then we obtain $T(SP'_{-+}; l(e_{\text{add}}))$.

Notice that in the above construction newly added edges (connecting the vertices of two convex chains with $l(e_{\text{add}})$) are not flippable. Then we derive the following observation:

Observation 25. Every flippable edges $e \in T(SP'_{-+}; l(e_{\text{add}}))$ is either flippable in $T(SP'_-)$ or equal to e_{add}.

Furthermore we can easily see the following observation:

Observation 26. e_{add} is flippable in $T(SP'_{-+})$ if $e_{\text{add}} \notin T(SP'_-)$.

Although the proof is left in the next section, we can find the necessary and sufficient condition to reply (A-1) developing Observations 22 and 26. Notice that, from Observation 26, e_{add} is improving flippable only if $e_{\text{add}} \notin T(SP'_-)$ holds, and, from Observation 22, all edges of $T(SP'_{-+}; i)$ for $i \in \{1, \ldots, l(e_{\text{add}}) - 1\}$ are edges of $T(F)$ and $l(e_{add})$ may become the critical vertex when $l(e_{\text{add}}) \leq c$ while c remains critical vertex if $c < l(e_{\text{add}})$ holds.

Lemma 27. Let c be a critical vertex of $T(SP')$, and let $e_{\text{add}} \notin SP' - e_{\text{rem}}$. Then e_{add} is improving flippable in $T(SP'_{-+})$ if and only if both $l(e_{\text{add}}) \leq c$ and $e_{\text{add}} \notin T(SP'_-)$ hold.

Then let us start the proof of Lemma 19

Proof (Proof of Lemma 19). Let $e_{\text{rem}} = (a, b)$. Let c be a critical vertex of SP' and $e^* = \min\{e \in SP' \setminus F \mid e \in I_{SP'}\}$. From Lemma 27, an edge $e_{\text{add}} \in E_{BCD}$ cannot be improving flippable in $T(SP'_{-+})$ if $e_{\text{add}} \in T(SP'_-)$, and then cannot satisfy the condition (A). So only the edges of $E_{BCD} \setminus T(SP'_-)$ may be the edges satisfying all conditions in Lemma 12. Denote $E_{BCD} \setminus T(SP'_-)$ by E'_{BCD} and let us consider how to get E'_{BCD} from E_{BCD} in $O(n)$ time. Now, from Lemmas 20 and 23, we can obtain $T(SP'_-; i)$ for all $i \in \{1, \ldots, a\}$ from $T(SP')$ in $O(n)$ time. Although no edges of $T(SP'_-; i)$ for $i \in \{a + 1, \ldots, n\}$ are computed, E'_{BCD} can be obtained because $l(e_{\text{add}}) \leq a$ must hold from the condition (b) of Lemma 12. Note that E_{BCD} obtained from the algorithm in Lemma 18 is already sorted in lexicographical order, and also $T(SP'_-; i)$ is updated in lexicographical order from $T(SP')$. Then $E'_{BCD} = E_{BCD} \setminus T(SP'_-)$ is obtained in $O(n)$ time.

Let us consider how to check whether e_{add} satisfies the conditions (A) (more precisely (A-1) and (A-2) discussed above) in $T(SP'_{-+})$ for each

$e_{\text{add}} \in E'_{BCD}$. We consider the problem according to the cases mentioned above.

Case 1 $x \in R^{-++} \cup R^{-+-}$, which is the case (BCD-2) or (BCD-3) and $e_{\text{add}} = (a, x)$, we further divide the proof into three cases depending on the relative position of a. (Note that $a = l(e_{\text{rem}}) = l(e_{\text{add}})$ in this case.)

(1-a): When $c < a$ holds, from Lemma 27, we notice that e_{add} is not improving flippable in $T(SP'_{-+})$. In this case $e_{\text{add}} \in E'_{BCD}$ does not satisfy the condition (A).

(1-b): When $a < c$ holds, from Lemma 27, e_{add} is improving flippable in $T(SP'_{-+})$ and (A-1) holds. So let us show (A-2) that means there exists no improving flippable edge smaller than e_{add}. We claim that $e_{\text{rem}} \in T(F)$ holds when $a = l(e_{\text{rem}}) < c$ since all edges incident to vertices smaller than c in $T(SP')$ are coincide with the corresponding edges in $T(F)$ from the definition of the critical vertex. Therefore, $T(SP'_{-}) = T(SP')$ holds from Lemma 29 and the critical vertex of SP'_{-} is still c. And then $T(SP'_{-}; a) = T(F; a)$ holds for $a < c$, which implies that no edge of $T(SP'_{-}; a) \setminus F(a)$ is flippable except for the upper and lower hull edges because all added edges for constructing $T(F)$ from F are not flippable from the definition of CSIT (Observation 5). Since the upper and lower hull edges are not improving flippable from Lemma 6, only e_{add} is improving flippable edge in $T(SP'_{-+})$. Hence, in this case, $e_{\text{add}} \in E'_{BCD}$ always satisfies the condition (A).

(1-c): When $c = a$ holds, we again notice that $c = a = l(e_{\text{rem}}) = l(e_{\text{add}})$ hold. Similarly, from Lemma 21, e_{add} is improving flippable in $T(SP'_{-+})$ and the critical vertex of SP'_{-+} is $l(e_{\text{add}})$. Then let us show how to check whether there exists a lexicographically smaller improving flippable edge in $T(SP'_{-+}; l(e_{\text{add}}))$ than e_{add}. Let $(a, a'_0), (a, a'_1), \ldots, (a, a'_k)$ be the edges of $T(SP'; a)$ in clockwise order around a, and assuming that $b(= r(e_{rem})) = v_l$. Then we can observe that $T(SP'_{-+})$ still contains the triangles $\Delta a'_0 a'_1 b, \ldots, \Delta a'_{l-1} ab$ because the triangle Δaxb is always contained in $T(SP'_{-+})$ from the condition (D) in Lemma 12. (Imagine that we continue having the constrained edge $e_{\text{rem}} = (a, b)$ when updating the triangulation from $T(SP')$ to $T(SP'_{-+})$). By using this fact we can show that e_{add} satisfies the condition (A) if and only if either (i)$e_{\text{add}} \prec e^*$ holds, or (ii)$e^* = e_{\text{rem}} (\prec e_{\text{add}})$ holds.

If (i) holds, we show that there exists no edge in $T(SP'_{-+}; a)$ such that it is improving flippable and lexicographically smaller than e_{add} in $T(SP'_{-+})$. First we remark that $e_{\text{rem}} = (a, b)$ is next to $e_{\text{add}} = (a, x)$

in lexicographical increasing order among $T(SP'_{-+}; a)$ from the fact that $T(SP'_{-+})$ contains Δaxb. Then we show that no edge smaller than $e_{\rm rem}$ in $T(SP'_{-+}; a)$ is improving flippable in $T(SP'_{-+})$. In fact, e^* is lexicographically smallest improving flippable edge in $T(SP')$, that is, no edge of $T(SP')$ smaller than $e_{\rm add}$ is improving flippable in $T(SP')$. Remember the above mentioned fact that all edges smaller than $e_{\rm rem}(\succ e_{\rm add})$ in $T(SP')$ are still contained in $T(SP'_{-+})$ and incident to the same triangles as in $T(SP')$. Then there exists no improving flippable edge lexicographically smaller than $e_{\rm add}$ in $T(SP'_{-+})$. If (ii) holds, by the same reason as in (i), no edge smaller than $e_{\rm rem} = e^*$ in $T(SP'_{-+}; a)$ is not improving flippable in $T(SP'_{-+})$. And $e_{\rm rem} \notin SP'_{-+}$ is not flippable in $T(SP'_{-+})$ from Observation 5. Hence, no edge smaller than $e_{\rm add}$ in $T(SP'_{-+}; a)$ is improving flippable in $T(SP'_{-+})$. (No edge exists between $e_{\rm add}$ and $e_{\rm rem}$ from $\Delta axb \in T(SP'_{-+})$.)

Conversely, suppose that $e^* \prec e_{\rm add}$ and $e_{\rm rem} \neq e^*$ holds. As already noticed, $e_{\rm add} = (a, x)$ is next to $e_{\rm rem} = (a, b)$ in lexicographical ordering among $T(SP'_{-+}; a)$, and we have $e^* \prec e_{\rm rem} \prec e_{\rm add}$. From the fact that all edges smaller than $e_{\rm rem}(\prec e_{\rm add})$ in $T(SP'; a)$ are still contained in $T(SP'_{-+}; a)$ and incident to the same triangles as in $T(SP')$, $e^*(\prec e_{\rm add})$ remains improving flippable in $T(SP'_{-+})$. Hence, either $e_{\rm add} \prec e^*$ or $e^* = e_{\rm rem}$ is necessary for $e_{\rm add}$ to satisfy the condition (A).

As a result, $e_{\rm add}$ with $x \in R^{-++} \cup R^{-+-}$ satisfies the condition (A) iff $e_{\rm add} \prec e^*$ or $e_{\rm rem} = e^*$ holds, and this can be checked in $O(1)$ time for each $e_{\rm add} \in E'_{BCD}$.

Case 2 $x \in R^{+--} \cup R^{---}$, which is the case (BCD-1) or (BCD-4). Notice that $e_{\rm add} = (x, a)$ holds when $x \in R^{+--}$ and $e_{\rm add} = (x, b)$ holds when $x \in R^{---}$, and $l(e_{\rm add}) < l(e_{\rm rem})$ in both cases. When $c < x$ holds, by the same reason in the case (1-a), $e_{\rm add}$ cannot satisfy the condition (a). Similarly, when $x < c$ holds, $e_{\rm add}$ always satisfies the condition (A). And, these can be checked in $O(1)$ time for each $e_{\rm add} \in E'_{BCD}$.

(2-b): When $x = c$ holds, we calculate $T(SP'_{-+}; x)$ from $T(SP'-)(x)$ in $O(n)$ time by the algorithm described in Lemma 24. Since there exists no improving flippable edge in $T(SP'_{-+}; i) = T(SP'; i)$ for $i \in \{1, \ldots, x - 1(= c-1)\}$ (from Lemmas 20 and 21), we can check whether $e_{\rm add}$ is lexicographically smallest improving flippable edge in $T(SP'_{-+})$ by comparing only the edges of $T(SP'_{-+}; x)$ in $O(n)$ time. For the analysis of the time complexity, notice that all x of edges in E_{BCD} are distinct. Then, we have at most one time when $x = c'$ holds. Thus the total time for finding the edges satisfying the condition (A) is $O(n)$. This completes the proof.

we can see that e_{add} is neither the upper nor lower hull edge of $l(e_{\text{add}})$ respecting SP'_{-+} since, otherwise it remains the upper or lower hull edge respecting SP'_{-} when removing e_{add} from SP'_{-+}, and $e_{\text{add}} \in T(SP'_{-})$ holds.

References

1. O. Aichholzer, F. Aurenhammer, C.Huemer and H. Krasser. Transforming spanning trees and pseudo-triangulations. *Inf. Process. Lett.*, 97(1):19–22, 2006.
2. O. Aichholzer, F. Aurenhammer and F. Hurtado. Sequences of spanning trees and a fixed tree theorem. *Comput. Geom.*, 21(1-2):3–20, 2002.
3. O. Aichholzer and K. Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. In *Proc. 20th European Workshop on Computational Geometry (EWCG04)*, pages 13-16, Sevilla, Spain, 2004.
4. O. Aichholzer, G. Rote, B. Speckmann, and I. Streinu. The zig-zag path of a pseudo-triangulation. In *Proc. 8th International Workshop on Algorithms and Data Structures (WADS)*, Lecture Notes in Computer Science 2748, pages 377–388, Ottawa, Canada, 2003. Springer Verlag.
5. T. Asano, S. K. Ghosh and T. Shermer. Visibility in the plane. In J.-R Sack and J. Urrutia eds, *Handbook in Computational Geometry*, Elsevier, Chapter 19, pp. 829–876, 2000.
6. D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8:295–313, 1992.
7. D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, March 1996.
8. D. Avis, N. Katoh, M. Ohsaki, I. Streinu, and S. Tanigawa. Enumerating non-crossing minimally rigid graphs. In *Proc. 12th Annual International Conference Computing and Combinatorics (COCOON 2006)*, LNCS 4112, pages 2005–215, Taipei, 2006
9. D. Avis, N. Katoh, M. Ohsaki, I. Streinu, and S. Tanigawa. Enumerating non-crossing constrained minimally rigid frameworks. `http://arxiv.org/PS_cache/math/pdf/0608/0608102.pdf`
10. S. Bereg. Enumerating pseudo-triangulations in the plane. *Comput. Geom. Theory Appl.*, 30(3):207–222, 2005.
11. M. Bern and D. Eppstein. Mesh generation and optimal triangulation. *Computing in Euclidean Geometry, 2nd Edition, Du and Hwang eds.*, 23–90, 1992.
12. S. Bespamyatnikh. An efficient algorithm for enumeration of triangulations. *Comput. Geom. Theory Appl.*, 23(3):271–279, 2002.
13. H. Brönnimann, L. Kettner, M. Pocchiola, and J. Snoeyink. Enumerating and counting pseudo-triangulations with the greedy flip algorithm. In *Proc. ALENEX*, Vancouver, Canada, 2005.
14. A. Dumitrescu, B. Gärtner, S. Pedroni, and E. Welzl. Enumerating triangulation paths. *Computational Geometry: Theory and Applications*, 20(1-2):3–12, 2001.
15. M. C. Hernando, M. E. Houle and F. Hurtado. On Local Transformation of Polygons with Visibility Properties. In *Proc.6th Annual International Conference Computing and Combinatorics,COCOON 2000*, LNCS 1858, pages 54–63. Springer, 2000.

16. C. Hernando, F. Hurtado and M. Noy. Graphs of Non-Crossing Perfect Matchings., *Graphs and Combinatorics*, 18(3):517–532, 2002.
17. F. Hurtado, M. Noy and J. Urrutia. Flipping Edges in Triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.
18. B. Joe and R. B. Simpson. Corrections to Lee's Visibility Polygon Algorithm. *BIT*, 27(4):458–473, 1987.
19. D. T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207–221, 1983.
20. D. J. A. Welsh. Matroids: Fundamental Concepts In *Handbook of Combinatorics Vo.I, R.L.Graham, M.Grötschel, and L.Lovász eds.* North-Holland, 1995, 481-526.

A Deletion and insertion of the constrained edges

In this section we give the rigorous proofs of Lemmas described in Section **??**. First let us consider how $T(SP')$ changes to $T(SP'_-)$ when removing a constrained edge $e_{\rm rem}$. Althoug the following two lemmas have not been claimed in the main sentence, we need them.

Lemma 28. $T(SP'_{-+}) = T(SP'_-)$ holds if $e_{\rm add} \in T(SP'_-)$.

Proof. We show that the upper (and lower) hull edges of i respecting SP'_-, (i, i_{0,SP'^-}), and that respecting SP'_{-+}, $(i, i_{0'})$, are consider for all $i \in P$. If not, and suppose that i_0 is not visible from i with respect to SP'_{-+}. From this fact, we find that (i, i_0) intersect $e_{\rm add}$, which contradicts the fact that both $(i, i_0) \in T(SP'_{-+})$ and $e_{\rm add} \in T(SP'_-)$ hold. Then, by the definition of CSIT, the lemma follows.

Lemma 29. Let $e_{\rm rem} \in SP' \setminus F$. $T(SP'_-) = T(SP')$ holds if $e_{\rm rem} \in T(F) \setminus F$.

Proof. Let $e_{\rm rem} = (a, b) \in T(F) \setminus F$. It is sufficient to show that $e_{\rm rem}$ is still contained in $T(SP'_-)$ from Lemma 28 (replacing $e_{\rm add}$ by $e_{\rm add}$). We first remark that, as for the set of points of P_a visible from a, $V_{SP'_-}(a, P_a) = V_{SP'}(a, P_a) \subseteq V_F(a, P_a)$ holds from $F \subset SP'$. If $e_{\rm rem}$ is the upper or lower hull edge of a in F (denoted by $(a, a_0), (a, a_k)$), $e_{\rm rem}$ is also the upper or lower hull edge of a in SP'^- from the fact that $b \in V_{SP'_-}(a, P_a) \subseteq V_F(a, P_a)$. Hence $T(SP'_-)$ contains $e_{\rm rem}$ from Definition 1.

If $e_{\rm rem}$ is neither (a, a_0) nor (a, a_k), $e_{\rm rem}$ is not flippable in $T(F)$ from Observation 5 (by replacing SP by F in the statement of Observation 5). Consider $T(F)$ and let us denote the edges of $F(c) \cup \{(a, a_0), (a, a_k)\}$ by $(a, a_0), (a, a_1), \ldots, (a, a_k)$ in clockwise order around a. Since $e_{\rm rem}$ is not flippable in $T(F)$, there exists a unique l with $0 \leq k-1$ of one convex hull $H_l = conv(P_a \cap (a, a_l)^{-,0} \cap (a, a_{l+1})^{+,0})$ such that $b = (a, b) \cap H_l$ holds.

Similarly, consider $T(SP')$ and let $(a,\tilde{a}_0),(a,\tilde{a}_1),\ldots,(a,\tilde{a}_{k'})$ be the edges of $SP(c)\cup\{(a,\tilde{a}_0),(a,\tilde{a}_{k'})\}$ arranged in clockwise order around a, where $(a,\tilde{a}_0)$ and $(a,\tilde{a}_{k'})$ are the upper and lower hull edges of a in SP'_-. Then there exists a convex hull $\tilde{H}_{l'} = conv(P_a\cap(a,\tilde{a}_{l'})^{-,0}\cap(a,\tilde{a}_{l'+1})^{+,0})$ for $0\le l'\le k'-1$ such that $b\in\tilde{H}_{l'}$. Because of $F(c)\subseteq SP'(c)$, $(a,\tilde{a}_{l'})^{-,0}\cap(a,\tilde{a}_{l'+1})^{+,0}\subseteq((a,a_l)^{-,0}\cap(a,a_{l+1})^{+,0})$ holds, and hence $\tilde{H}_{l'}\subseteq H_l$ holds. Recalling that $b=(a,b)\cap H_l$, we have $b=(a,b)\cap\tilde{H}_{l'}$. Hence $e_{\rm rem}=(a,b)$ is contained in $T(SP'-)$ from definition of CSIT.

Proof (Proof of Lemma 20). Let (i,i_0) and (i,i_k) be the upper and lower hull edges of i in SP'. We first show that (i,i_0) is still the upper hull edge of i in SP'_-, and $T(SP'_-)$ contains (i,i_0). (The same arguments can be applied to the lower hull edge.) Suppose that the upper hull edge, (i,i'_0), of i in SP'_- is different from (i,i_0). Since i'_0 is not visible from i in SP' but in SP'_-, $e_{\rm rem}$ intersects (i,i'_0), which implies $l(e_{\rm rem})<i$ from the empty region property of (i,i'_0) that was mentioned in Section 2.2. This contradicts $i<l(e_{\rm rem})$.

Now the set of edge of SP'_- incident to i is equal to that of SP' from $i<l(e_{\rm rem})$. So we can see that $SP'_-(i)\cup\{(i,i'_0),(i,i'_k)\}=SP'(i)\cup\{(i,i_0),(i,i_k)\}$, and the statement follows from Definition 1.

Proof (Proof of Lemma 23). Let $e_{\rm rem}=(a,b)$. We assume that the set of edges of $T(SP';a)$ is maintained in lexicographical ordering (which is defined as clockwise ordering in Section 2.1), and show that $T(SP'_-;a)$ can be updated in $O(n)$ time so that this assumption remains to hold. Let the vertices of $T(SP';a)$ be $(a,a'_0),(a,a'_1),\ldots,(a,a'_k)$ arranged in clockwise ordering, where (a,a'_0) and (a,a'_k) are upper and lower hull edges of a in SP'. If $e_{\rm rem}=(a,a'_0)$ or $e_{\rm rem}=(a,a'_k)$ holds, we have $T(SP'_-;a)=T(SP';a)$ as was shown in the proof of Lemma 29. It is because that (a,a'_0) and (a,a'_k) are still the upper and lower hull edges of a in SP'_- from the empty region properties of (a,a'_0) and (a,a'_k). So, let us consider the case that $e_{\rm rem}=(a,a'_i)$ with $i\neq 0,k$. First we compute, in $O(n)$ time, two edges (a,a'_s) and (a,a'_t) in $(SP'(a)-e_{\rm rem})\cup\{(a,a'_0),(a,a'_k)\}$ with $a'_s\in(a,a'_i)^+$ and $a'_t\in(a,a'_i)^-$ such that the angle $\angle a'_saa'_t$ around a is minimum (see Fig. 11(a)). Let $P_{st}=\{a'_s,a'_{s+1},\ldots,a'_t\}$, (which is arranged in lexicographical ordering). Computing $conv(P_{st})$ and finding the set of vertices visible from a with respect to $conv(P_{st})$ in $O(n)$ time by tracing the boundary of $conv(P_{st})$ from a'_s to a'_t. Connecting these vertices with a, we obtain $T(SP'_-;a)$ maintaining the lexicographical ordering for edges.

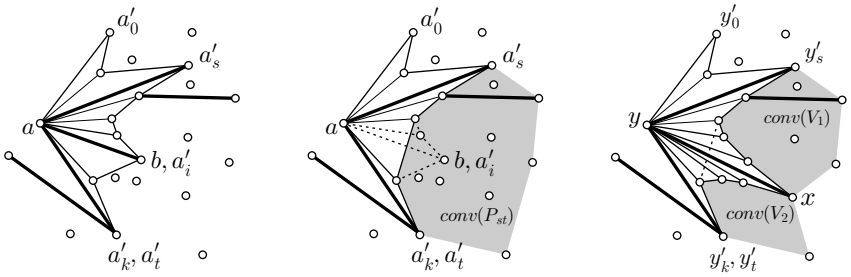


Fig. 11. (a)The part of $T(SP')$, where the bold edges represent those of SP', (b)the part of updated triangulation when removing $e_{\rm rem} = (a, b)$ from SP' and (c)when inserting $e_{\rm add} = (y, x)$ into SP'_-.

Next let us consider how CSIT changes when we insert $e_{\rm add}$ into $SP' - e_{\rm rem}$. The following Lemmas 30 and 21 can be proved in the same manners as in those of Lemma 29 and 20, respectively.

Lemma 30. $T(SP'_{-+}) = T(SP'_-)$ holds if $e_{\rm add} \in T(F)$.

Proof (Proof of Lemma 24). Let $e_{\rm add} = (y, x)$. When $e_{\rm add} \in T(SP'_-)$ holds, $T(SP'_{-+}; y) = T(SP'_-; y)$ holds from Lemma 28. So, we assume $e_{\rm add} \notin T(SP'_-)$. Denote the edges of $T(SP'_-; y)$ arranged in clockwise order around y by $(y, y'_0), (y, y'_1), \ldots, (y, y'_k)$. Note that $x \notin (y, y'_0)^+$ and $x \notin (y, y'_k)^-$ hold from the empty region properties of the upper or lower hull edge in $T(SP'_-)$.

By the same way as in the proof of Lemma 23 let us compute in $O(n)$ time two edges (y, y'_s) and (y, y'_t) in $(SP'(y) - e_{\rm rem}) \cup \{(y, y'_0), (y, y'_k)\}$ with $y'_s \in (y, x)^+$ and $y'_t \in (y, x)^-$ such that $\angle y'_s y y'_t$ around y is minimum (see Fig. 11(c)). Then update occurs only in the region $(y, y'_s)^- \cap (y, y'_t)^+$. It can be done as follows according to the definition of CSIT. First we compute two sets of vertices of $P_y \cap (y, y'_s)^{-,0} \cap (y, x)^{+,0}$ and $P_y \cap (y, x)^{-,0} \cap (y, y'_t)^{+,0}$ visible from y with respect to SP'_{-+}, (denoted by V_1 and V_2, respectively), in $O(n)$ time by the algorithm described in Lemma 17. (These vertices are obtained in lexicographical ordering). Then, we compute, in linear time, two convex hulls of V_1 and V_2 and then compute the sequences of vertices (two convex chains) by tracing the boundaries of $conv(V_1)$ and $conv(V_2)$. Thus $T(SP'_{-+}; y)$ is obtained by connecting these vertices with y in $O(n)$ time (in lexicographical order).

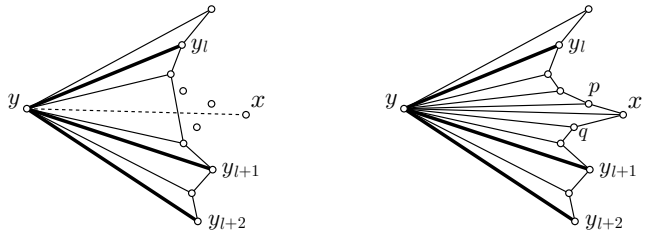


Fig. 12. (a)(b) Adding $e_{\text{add}} = (y, x)$ with $e_{\text{add}} \notin T(SP'_-)$ results in the triangulation in which e_{add} is flippable.

B The Necessary and Sufficient Consition for e_{add} to be Improving Flippable in $T(SP'_{-+})$

Proof (Proof of Lemma 27). Let $e_{\text{add}} = (y, x)$. ("if"-part.) From Lemma **??** e_{add} is flippable in $T(SP'_{-+})$ when $e_{\text{add}} \notin T(SP'_-)$ holds. Then, from Lemma 6, e_{add} is improving flippable in $T(SP'_{-+})$ if $e_{\text{add}} \in SP'_{-+}(c') \setminus \{(c', c'_0), (c', c'_k)\}$ holds, where c' be the critical vertex of SP'_{-+} and $(c'c'_0)$ and (c', c'_k) are the upper and lower hull edges of c' respecting SP'_{-+}. To prove this, we show the following two things:(1) y is the critical vertex in SP'_{-+} and (2) e_{add} is neither the upper nor lower hull edges of y in SP'_{-+}.

(1)Now we have $y = l(e_{\text{add}}) \leq c$ and $y = l(e_{\text{add}}) \leq e_{\text{rem}}$. Then $T(SP'; i) = T(SP'_{-+}; i)$ holds for $i \in \{1, \ldots, y-1\}$ from Lemmas 20 and Lemma 21. Since all edges of $T(SP'; i)$ for $i \leq c$ are edges of $T(F)$ from the definition of the critical vertex, they remains in $T(F)$ for $i \in \{1, \ldots, y-1\}$. Thus no edge of $T(SP'_{-+}; i)$ for $\{1, \ldots, y-1\}$ is improving flippable in $T(SP'_{-+})$, and the critical vertex of $T(SP'_{-+})$ become y if $e_{\text{add}} \notin T(F)$. To show this we suppose that $e_a \in T(F)$. Then $T(SP'_-) = T(SP'_{-+})$ holds from Lemma 23, and $e_{\text{add}} \in T(SP'_-)$ holds, which is contradiction

(2)We show that the upper and lower hull edges of y respecting SP'_{-+} coincides with those respecting SP' from $V_{SP'_{-+}}(y, P_y) = V_{SP'}(y, P_y)$. If so, e_{add} is not equal to them because $e_{\text{add}} \notin T(SP'_-)$. As we noticed above, $T(SP'_{-+}; i) = T(SP'; i)$ holds now for $i \in \{1, \ldots, y-1\}$, which implies $V_{SP'_{-+}}(y, P_y) = V_{SP'}(y, P_y)$ and the upper and lower hull edges does not change. Thus "if"-part is proofed.

("only if"-part.) Suppose $e_{\text{add}} \in T(SP'_-)$ holds, we have $T(SP'_{-+}) = T(SP'_-)$ from Lemma 28. And, from Lemma 6, all improving flippable edges in $T(SP'_-)$ are in SP'_-. Then, since $e_{\text{add}} \notin SP'_-$ from the lemma assumption, e_{add} is not improving flippable in $T(SP'_-) = T(SP'_{-+})$. Hence $e_{\text{add}} \in T(SP'_-)$ must be hold. Again from Lemma 6, all improv-

ing flippable edges in $T(SP'_{-+})$ are incident to the critical vertex of SP'_{-+}, so $y(= l(e_{\text{add}}))$ must be the critical vertex of SP'_{-+}. Thus we need $l(e_{\text{add}}) \leq c$.