Karl-Franzens-Universität Graz & Technische Universität Graz

SPEZIALFORSCHUNGSBEREICH F 003

OPTIMIERUNG

und

KONTROLLE

Projektbereich

DISKRETE OPTIMIERUNG

Oswin Aichholzer Franz Aurenhammer
Siu-Wing Cheng Naoki Katoh Günter Rote
Michael Taschwer Yin-Feng Xu

Triangulations Intersect Nicely

Bericht Nr. 30, Version 2b — August 1996 to appear in *Discrete and Computational Geometry*

Triangulations Intersect Nicely

Oswin Aichholzer
Franz Aurenhammer
Michael Taschwer
Institute for Theoretical Computer Science
Graz University of Technology
Klosterwiesgasse 32/2, A-8010 Graz, Austria
e-mail: {oaich,auren}@igi.tu-graz.ac.at

Siu-Wing Cheng
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay
Hong Kong
e-mail: scheng@cs.ust.hk

Naoki Katoh
Department of Management Science, Kobe University of Commerce
Gakuen-Nishimachi 8-2-1, Nishi-ku, Kobe 651-21, Japan
e-mail: naoki@kobeuc.ac.jp

Günter Rote
Institut für Mathematik
Technische Universität Graz
Steyrergasse 30, A-8010 Graz, Austria
e-mail: rote@opt.math.tu-graz.ac.at

Yin-Feng Xu
School of Management
Xi'an Jiaotong University
Xi'an Shaanxi, 710049, P. R. China
e-mail: xucx@xjtu.edu.cn

August 22, 1996

Abstract

We show that there is a matching between the edges of any two triangulations of a planar point set such that an edge of one triangulation is matched either to the identical edge in the other triangulation or to an edge that crosses it. This theorem also holds for the triangles of the triangulations and in general independence systems. As an application, we give some lower bounds for the minimum weight triangulation which can be computed in polynomial time by matching and network flow techniques. We exhibit an easy-to-recognize class of point sets for which the minimum-weight triangulation coincides with the greedy triangulation.

1 Introduction

The aim of this paper is to prove and discuss some surprising and rather general intersection properties of planar triangulations.

Given two triangulations of a point set, we can find a matching between their edge sets such that matched edges either cross or coincide. This theorem and a few related statements will be proved in Section 2.

The remaining part of the paper deals with applications of this result to the computation of minimum-weight triangulations. In Section 3 we identify special cases of point sets for which the minimum-weight triangulation can be computed efficiently. An example is shown in Figure 4. Section 4 offers several lower bounds on the weight of a triangulation, and Section 5 describes some algorithms to compute these bounds. In Section 6 we will discuss possible applications of our results and some open questions.

The main matching result (Theorem 1) was discovered independently by two subsets of the current authors, see [AART95] and [CX95]. Naoki Katoh contributed the ideas for Section 4.3. This joint paper is a final version of [AART95] and some of the results of [CX95].

2 The matching theorems

2.1 Matchings of triangulations

Let P be a set of n points in the plane. We assume that not all points lie on one line. Consider the set E of all line segments connecting two points of P and containing no other points of P. (If P is in general position then $|E| = \binom{n}{2}$.) The elements of E are called *edges*. Two distinct edges are said to cross if they intersect in their interior. In particular, two edges sharing only one endpoint do not cross. A *triangulation* T of P is a maximal set of non-crossing edges. It dissects the convex hull of P into triangular faces. We recall a well-known fact on triangulations.

Lemma 1 *Every triangulation of P consists of $m = 3n - 3 - b$ edges, where b is the number of points in P which lie on the boundary of the convex hull of P. Every set of non-crossing edges consists of at most m edges.* □

The following is our main theorem. See Figure 1 for an illustration of the result.

Theorem 1 *Let P be a finite set of points in the plane and consider two triangulations R and B of P. There exists a perfect matching (a one-to-one assignment, a bijective mapping) between R and B, with the property that matched edges either cross or are identical.*

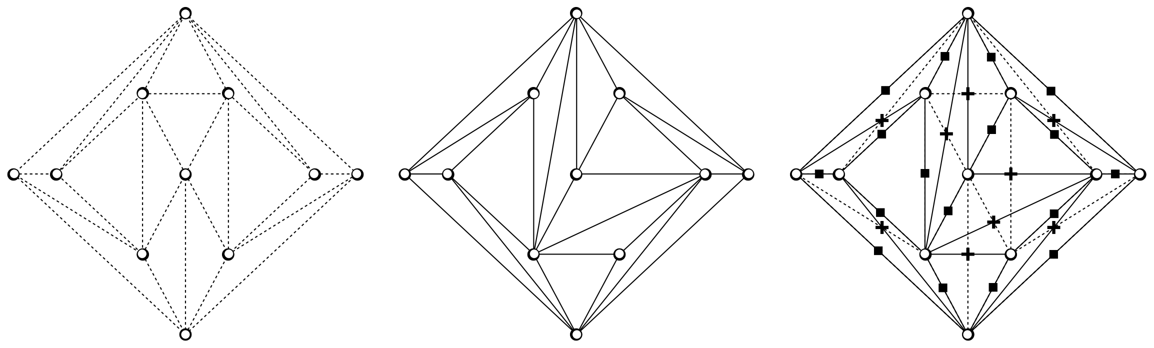

Figure 1: Two triangulations and a perfect matching between them. When two identical edges are matched this is indicated by a small box, and the intersection point of two crossing edges which are matched is marked by a cross.

Proof. For notational simplicity, let us color the edges in B blue and the edges in R red. We consider the intersection graph, G, of $R \cup B$. To avoid confusion, we will consistently speak of *edges* when we mean elements of R, B, E, etc., i. e., line segments considered as geometric objects. They correspond to *nodes* of the graph G, which are connected by *arcs*.

G represents each edge in R by a red node and each edge in B by a blue node. Two nodes are connected by an arc if the corresponding edges either cross or are identical. Clearly, there are no arcs between nodes of the same color as edges in a triangulation do not cross.

Hence G is bipartite. Note that, by Lemma 1, the number of red nodes equals the number of blue nodes.

We prove that G contains a perfect matching by showing that G fulfills the Hall condition of the marriage theorem (see for example Bollobás [B79]). This condition requires that, for each subset R_1 of red edges, the number of their blue neighbors in G is at least $|R_1|$.

Let $R_1 \subseteq R$, and let B_1 be the set of blue neighbors of R_1. With this notation, the Hall condition reads $|B_1| \geq |R_1|$. Let B_2 be the complement of B_1 in B. We claim that $B_2 \cup R_1$ is a non-crossing set of edges. The edges of R_1 do not cross each other, since $R_1 \subseteq R$, and likewise, the edges of B_2 do not cross each other. An edge b of B_2 cannot cross an edge of R_1 because b would belong to B_1, otherwise. For the same reason, R_1 and B_2 are disjoint, which gives

$$|R_1| + |B_2| = |R_1 \cup B_2| \leq m = |B|,$$

by Lemma 1. Thus we get $|B_1| = |B| - |B_2| \geq |R_1|$, and the Hall condition is proved. $\square$

A maximum cardinality matching in any bipartite graph can be found in polynomial time, and this holds in particular for the matching in Theorem 1. This will also be true of the other theorems in this section. They are formulated as existence theorems, but there are polynomial algorithms for constructing the matching in question. We will discuss specific algorithms and their time and space bounds in Section 5.1.

Corollary 1 *Let P be a finite set of points in the plane. Let R be a set of non-crossing edges between points of P and let T be a triangulation of P. Then there is a matching between the edges in R and some edges in T (an injective mapping from R to T) such that every edge of R is either matched with the identical edge in T or with an edge which crosses it.*

Proof. Since R can be extended to a triangulation the corollary follows from Theorem 1. $\square$

Our proof mainly exploits the property about the number of edges of triangulations expressed in Lemma 1. There is another version of Lemma 1 concerning triangles instead of edges.

Lemma 2 *Every triangulation of P dissects the plane into $2n-2-b$ interior triangular faces plus the exterior face, where b is the number of points in P on the boundary of the convex hull. Every set of non-crossing edges dissects the plane into at most $2n-1-b$ connected components.* $\square$

This enables us to prove, with exactly the same technique as above, the next theorem on triangles.

Theorem 2 *Let P be a finite set of points in the plane and consider two triangulations R and B of P. There exists a perfect matching between the set of triangles of R and the set of triangles of B, with the property that matched triangles either overlap or are identical.* $\square$

Figure 2 displays a perfect triangle matching for two triangulations of a convex point set. We can impose a stronger condition which requires the matched triangles to share a vertex. In Figure 2, for example, the triangle $p_2p_4p_7$ is matched to the triangle $p_3p_5p_6$. This would not be allowed in a matching according to Theorem 3.

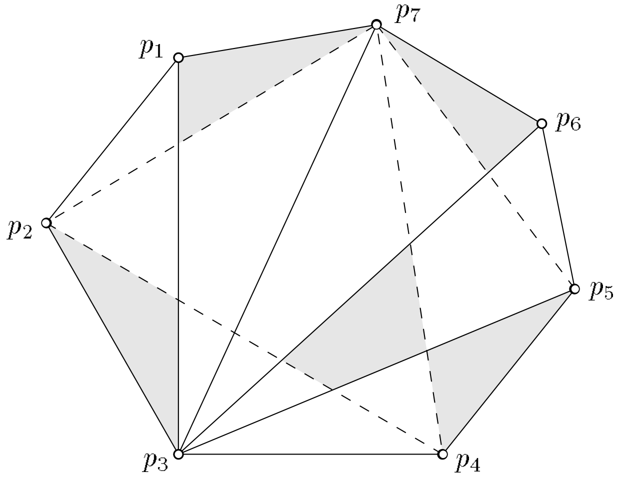

Figure 2: A perfect matching of overlapping triangles between two triangulations of a 7-gon. Every shaded area is the intersection between two matched triangles.

Theorem 3 *Let P be a finite set of points in the plane and consider two triangulations R and B of P. There exists a perfect matching between the set of triangles of R and the set of triangles of B, with the property that matched triangles*

1. *have common interior points, and*
2. *share at least one vertex.*

Proof. An equivalent formulation of the two conditions is that two triangles which are matched to each other have a vertex p in common, and the angular regions in the neighborhood of p which are covered by the two triangles overlap (have common interior points).

We can show the Hall condition more directly by an argument about sums of angles. Let R_1 be an arbitrary subset of triangles in R, and let B_1 be the set of triangles of B to which some triangle in R_1 can be matched. In other words, if a triangle in B shares a vertex and some overlapping angular region around that vertex with a triangle in R_1, then it belongs to B_1. We have to show $|B_1| \geq |R_1|$.

Fix a vertex $p \in P$. For a triangle Δ we denote by $\alpha(\Delta, p)$ the angle of Δ at p. If p is not a vertex of Δ then $\alpha(\Delta, p) = 0$. Every triangulation partitions the full angular region around p into disjoint sectors. (For vertices on the boundary of the convex hull, only the interior region is partitioned.) Consider the subsets $R_p \subseteq R_1$ and $B_p \subseteq B_1$ of triangles that have a vertex at p. At p, the union of the angles of triangles in B_p contains the union of the angles of triangles in R_p, by construction. Thus, the following inequality between sums of the angles holds for every vertex p:

$$\sum_{\Delta \in B_1} \alpha(\Delta, p) \geq \sum_{\Delta \in R_1} \alpha(\Delta, p).$$

We can take the sum over all vertices $p \in P$, and since $\sum_{p \in P} \alpha(\Delta, p) = \pi$ for every triangle Δ, we get $\pi|B_1| \geq \pi|R_1|$, and the Hall condition follows. □

2.2 A more general matching theorem for independence systems

Our matching theorems may be generalized to the framework of independence systems. An *independence system* $\mathcal{I}$ is a non-empty collection of subsets of a ground set E which is closed under taking subsets: if $A \in \mathcal{I}$ and $B \subset A$ then $B \in \mathcal{I}$. The elements of $\mathcal{I}$ are called the *independent* sets, the remaining subsets of E are called *dependent.* A *circuit* of $\mathcal{I}$ is a minimal dependent set.

In our example of triangulations, a set of non-crossing edges (or of non-overlapping triangles) may be considered independent. The circuits of this independence system have two elements; they are the pairs of crossing edges (or of overlapping triangles, respectively).

Theorem 4 *Let $R \in \mathcal{I}$ be any independent set, and let $B \in \mathcal{I}$ be an independent set of maximum cardinality in $\mathcal{I}$. Then there is an injective mapping $g\colon R \to B$ such that for every element $e \in R$ we have $g(e) = e$, or $\{g(e), e\}$ is contained in a circuit.*

Proof. The proof shows that the Hall condition is fulfilled, following the lines of the proof of Theorem 1. Let $R_1 \subseteq R$, and let $B_1 \subseteq B$ be the set of elements $f \in B$ such that $f = e$ or $\{e, f\}$ is contained in a circuit, for some element $e \in R_1$. To show that $|B_1| \geq |R_1|$ we first claim that $R_1 \cup (B - B_1)$ is independent. Otherwise, it would contain a circuit C. This circuit cannot be included in $R_1 \subseteq R$ or in $B - B_1 \subseteq B$ because these sets are independent. It follows that C contains an element $e \in R_1$ and an element $f \in B - B_1$. But this would mean that $f \in B_1$, a contradiction. So we have $|R_1 \cup (B - B_1)| \leq |B|$ because B is an independent set of maximum cardinality. R_1 and $B - B_1$ are disjoint by construction, and hence

$$|R_1 \cup (B - B_1)| = |R_1| + |B - B_1| = |R_1| + |B| - |B_1| \leq |B|,$$

which gives $|R_1| \leq |B_1|$. □

Theorems 1 and 2 (but not Theorem 3) are corollaries of Theorem 4. Cheng and Xu [CX95] have obtained additional results for independence systems where (as in the case of triangulations)

1. every circuit has cardinality two, and
2. all maximal (with respect to set inclusion) independent sets (bases) have the same cardinality.

For the case when $\mathcal{I}$ is a *matroid* (cf. e. g. Lawler [L76] for basic matroid-theoretic concepts), a stronger statement than Theorem 4 was discovered by Brualdi [B69]; see also Brylawski [B73] for a simple proof.

Theorem 5 *Let R and B be two bases of a matroid. Then there is a bijective mapping $g\colon R \to B$ such that for every element $e \in R$, $(R - \{e\}) \cup \{g(e)\}$ is a base.* □

It is straightforward to check that a matching fulfilling this condition fulfills the condition of Theorem 4, but not vice versa. Note that the condition of Theorem 5 is not symmetric in R and B.

3 Light triangulations

The intersection properties of planar triangulations expressed in the preceding section should have various applications in combinatorial and computational geometry. In this section and the next, two applications of Theorem 1 to minimum-weight triangulations are demonstrated. As before, let P be a set of n points in the plane, and let E be the set of all edges defined by P. The length (weight) of an edge $e = pq \in E$ is the Euclidean distance between

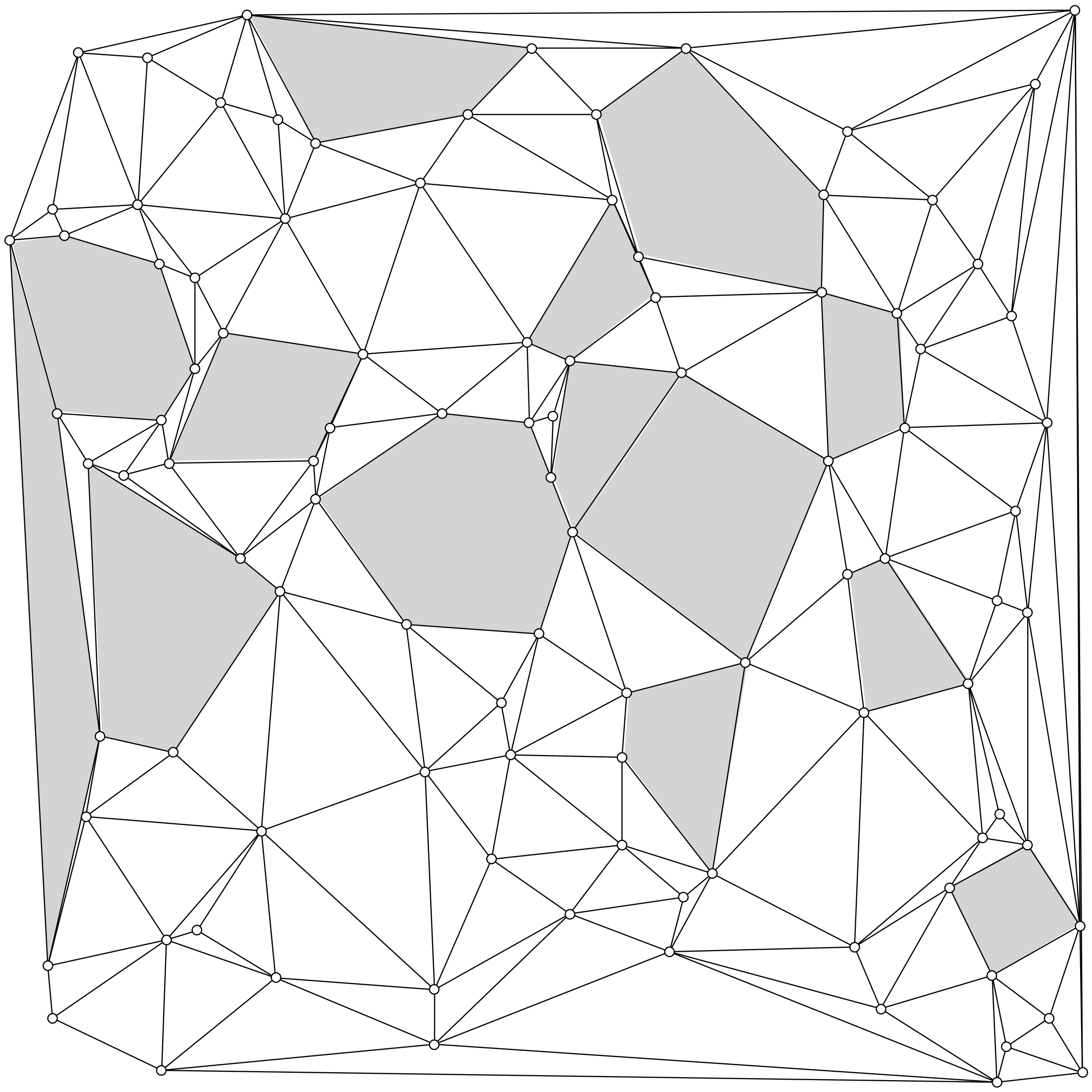

Figure 3: The light edges for a set of 100 points. The faces which are not triangles are shaded.

p and q and will be denoted by $|e|$ or $|pq|$. The weight $w(R)$ of a set R of edges is the sum of the lengths of its edges.

A *minimum-weight triangulation* T^* is defined to have $w(T^*) \leq w(T)$ for all triangulations T of P. Minimum-weight triangulations have some good properties (see Das and Joseph [DJ89]) and are, for example, useful for numerical approximation of bivariate data (Yoeli [Y75]). The complexity of computing a minimum-weight triangulation is unknown. This is in fact one of the longstanding open problems listed at the end of Garey and Johnson's book about NP-completeness [GJ79].

In this section, we exhibit a class of planar point sets where the minimum-weight triangulation can be computed in polynomial time.

Let us call an edge $e \in E$ *light* if any edge in E that crosses e is longer than e. Light edges obviously do not cross, so the set L of light edges can form at most a triangulation of P. Figure 3 shows the set L for a typical random point set. If L actually is a triangulation then we call L the *light triangulation* of P. See Figure 4 for an example.

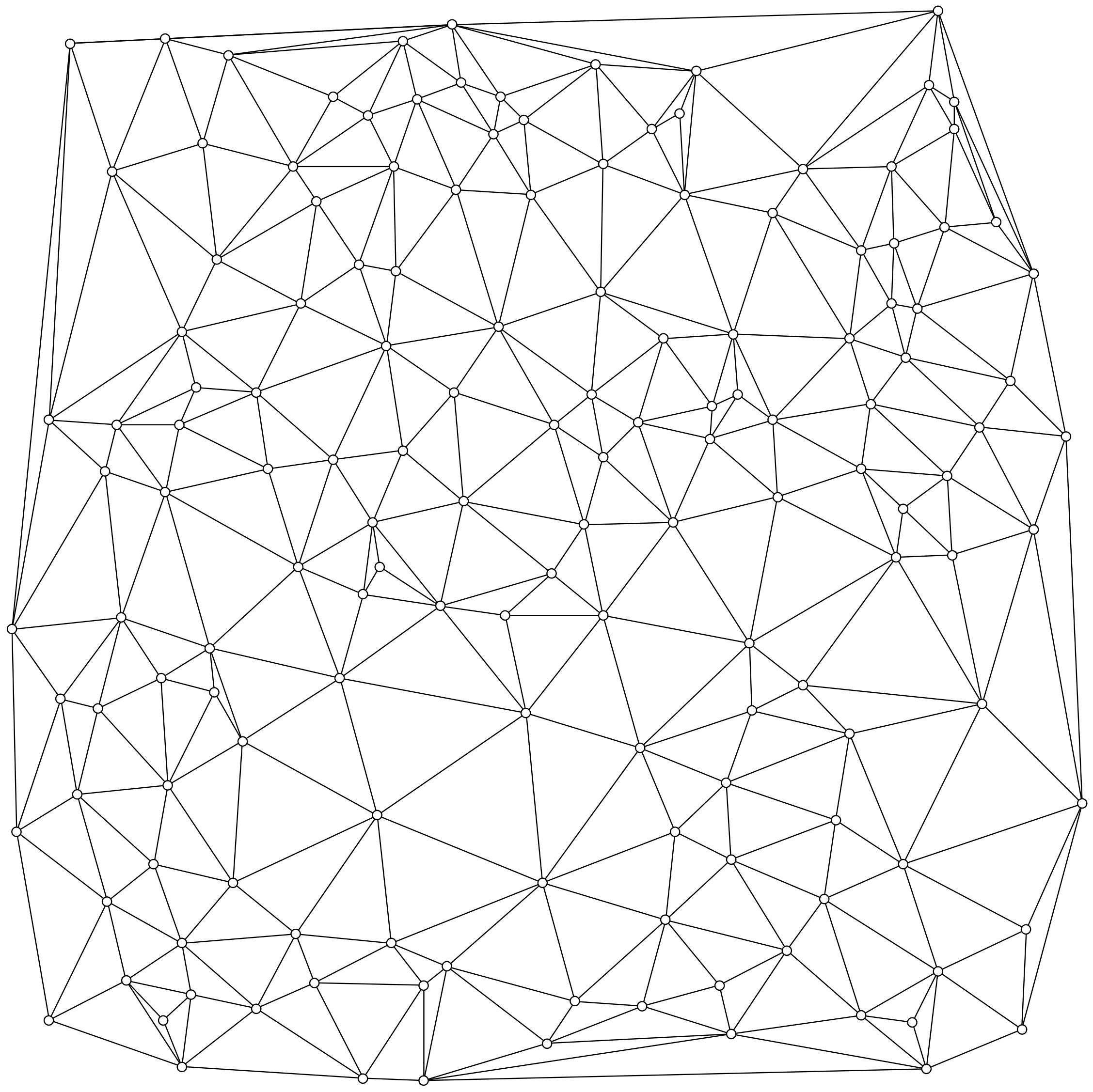

Figure 4: Minimum weight (light) triangulation for 150 points.

Light edges are related to the *greedy triangulation*, which is obtained by iteratively inserting the shortest edge of E that does not cross previously inserted edges. All light edges are contained in the greedy triangulation: a light edge e can never be blocked by previously inserted edges as E does not contain any shorter edge crossing e. Thus, if a light triangulation exists, it is identical to the greedy triangulation.

In light of Theorem 1, it is easy to prove length optimality.

Theorem 6 *If a planar point set P admits a light triangulation L then L is the minimum-weight triangulation for P.*

Proof. We show $w(T) \geq w(L)$, for any triangulation T of P. Consider a perfect matching as in Theorem 1 between the edges of T and the edges of L. For each matched pair of edges $e \in T$ and $e' \in L$, either $e = e'$ or e crosses e', in which case we know that $|e| > |e'|$ since L is light. Summing over all edges gives $w(T) \geq w(L)$. □

Since the light edges can easily be identified in polynomial time, the point sets admitting a light triangulation provide a polynomially solvable subclass of instances of the minimum-weight triangulation problem. In Section 5.2, we will show how to find the set of light edges in $O(n^2 \log n)$ time.

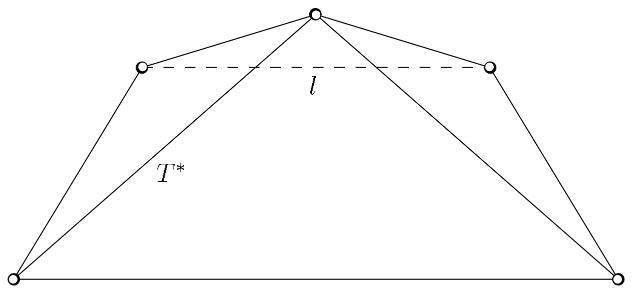

Figure 5: A light edge l not included in the minimum-weight triangulation T^*.

It is noteworthy that, in general, not all light edges occur in a minimum-weight triangulation T^*. Figure 5 gives an example.

4 Lower bounds for the minimum-weight triangulation

In this section, we offer new methods for computing lower bounds for the weight of planar triangulations. Such bounds can be used in a branch-and-bound algorithm for computing the minimum-weight triangulation. If the bounds are very tight they help to prune many branches of the branch-and-bound tree and thereby speed up the algorithm.

Apart from the trivial lower bound (the sum of the $|T^*|$ shortest edges in E), all known bounds require the knowledge of a set of edges which is a subset of T^*. Several local geometric criteria which guarantee that an edge belongs to T^* have recently been proposed, see [K94, CX96, YXY94], but since the resulting subsets of edges are usually small, the corresponding bounds are weak. We will see that Theorem 1 allows us to prove lower bounds for $w(T^*)$ in a completely different way, by solving an assignment problem and a minimum-cost flow problem, respectively.

4.1 Single assignment bounds

Theorem 7 (The single assignment bound) *Let R be a non-crossing set of edges and let T^* be a minimum-weight triangulation. Let $X(R, E)$ denote the set of all injective mappings (matchings) $g: R \to E$ with the properties required in Theorem 1: $g(e) = e$ or $g(e)$ crosses e. Then*

$$w(T^*) \geq \min_{g \in X(R,E)} \sum_{e \in R} |g(e)|. \tag{1}$$

Proof. Let $g^* \in X(R, E)$ be a matching $g^*: R \to T^*$ which exists by Theorem 1 and Corollary 1. We have

$$w(T^*) = \sum_{e \in T^*} |e| \geq \sum_{e \in R} |g^*(e)| \geq \min_{g \in X(R,E)} \sum_{e \in R} |g(e)|. \qquad \square$$

The set $X(R, E)$ is just the set of matchings with $|R|$ arcs in a bipartite graph. Optimizing over this set is an assignment problem (a special type of minimum-cost network flow problem, see e. g. [L76]), which can be solved in polynomial time. A faster algorithm with a running time of $O(n^3)$ will be developed in Section 5.3.

Note that edges in the set $g(R) = \{ g(e) : e \in R \}$ may cross each other. However, when R is a triangulation and $g_{\mathrm{opt}}(R)$ happens to be non-crossing for the optimal assignment g_{opt} in (1) then $g_{\mathrm{opt}}(R)$ must be minimum-weight:

Corollary 2 *Let R be a triangulation and let $g_{\mathrm{opt}} \in X(R,E)$ satisfy*

$$\sum_{e \in R} |g_{\mathrm{opt}}(e)| = \min_{g \in X(R,E)} \sum_{e \in R} |g(e)|.$$

If the set $g_{\mathrm{opt}}(R)$ is non-crossing then it is a minimum-weight triangulation. □

The optimal assignment g_{opt} has another nice property.

Theorem 8 *The optimal assignment matches no edge e to a longer edge, that is, $|g_{\mathrm{opt}}(e)| \leq |e|$ for all $e \in R$.*

Proof. Suppose that a matching $g \in X(R,E)$ has $|g(e)| > |e|$ for some edge e. Then we can improve g by assigning e to itself, i. e., by setting $g(e) = e$ and leaving g unchanged for the remaining edges. Since no other edge in R crosses e we still obtain a matching, and hence the original matching g cannot be optimal. □

If we relax the requirements on g in Theorem 7 and permit all injective mappings from R to E, we get the trivial lower bound: the sum of the $|R|$ shortest edges in E. This shows that our bound can never be worse than the trivial one.

4.2 Relaxed bounds

We may weaken the single assignment bound in a different way, by neglecting the fact that the matching g must assign different edges of R to different edges of T^*. The resulting bound is very easy to compute once some auxiliary information on the edges in R has been precomputed. For an edge $e \in E$, let us define its *excess* $\varepsilon(e)$ as

$$\varepsilon(e) = \max\{0,\ |e| - \lambda(e)\},$$

where $\lambda(e)$ is the length of the shortest edge crossing e. If e is crossed by no edge (for example, if e is an edge of the convex hull) then we set $\lambda(e) = \infty$, hence $\varepsilon(e) = 0$. Edges which have no crossing edges are called *unavoidable* edges; they belong to every triangulation, see Xu [X92]. Light edges have excess 0.

Theorem 9 *Let R be a non-crossing set of edges and let T^* be a minimum-weight triangulation. Then*

$$w(T^*) \geq w(R) - \sum_{e \in R} \varepsilon(e).$$

Proof. Let $g^*: R \to T^*$ be a matching as in the proof of Theorem 7. Then for all $e \in R$, either $g^*(e)$ crosses e or $g^*(e) = e$. In the first case, we have

$$|e| - \varepsilon(e) \leq |g^*(e)| \tag{2}$$

by the definition of the excess $\varepsilon(e)$. In the second case, the same inequality holds because $\varepsilon(e) \geq 0$. By summing (2) over all $e \in R$ we get

$$w(R) - \sum_{e \in R} \varepsilon(e) \leq \sum_{e \in R} |g^*(e)| \leq \sum_{e \in T^*} |e| = w(T^*).$$ □

The consequences of this theorem for the set $L \subset E$ of light edges are particularly interesting. Since L is a non-crossing set and $\varepsilon(e) = 0$ for all edges $e \in L$, we have:

Corollary 3

$$w(T^*) \geq w(L). \qquad \square$$

We stress the fact that, despite $L \not\subseteq T^*$ in general, summing up the light edges provides a valid lower bound. This bound covers the bounds in [K94, CX96, YXY94] as the subsets of T^* considered there are built of light edges only. If $R' \subset R$, then the bound given by R in Theorem 9 is at least as strong as the bound given by R'. Thus it pays to complete the non-crossing set R (or L) to a triangulation before applying Theorem 9. Note finally that if L happens to be a triangulation then $w(L) = w(T^*)$ and we obtain Theorem 6 of Section 3.

4.3 Double assignment bounds

We can improve the single assignment bound by starting with two different triangulations T_1 and T_2 instead of one set R.

Theorem 10 (The double assignment bound) *Let T_1 and T_2 be two triangulations and let T^* be a minimum-weight triangulation. With the notation of Theorem 7,*

$$w(T^*) \geq \min_{\substack{g_1 \in X(T_1,E) \\ g_2 \in X(T_2,E) \\ g_1(T_1)=g_2(T_2)}} \sum_{e \in T_1} |g_1(e)|, \tag{3}$$

and this bound can be computed in polynomial time.

Proof. Theorem 1 ensures the existence of bijective mappings $g_1^*: T_1 \to T^*$ and $g_2^*: T_2 \to T^*$ with the required properties, and the bound follows.

To get the polynomial-time result, we show that the required injective mappings g_1 and g_2 can be modeled as a flow in a network, and the optimal solution can be found by determining a minimum-cost flow. For basics of network flow terminology, see for example Tarjan [T87].

The network consists of four layers. The first layer contains a node for each edge of T_1; the second and third layers are identical: each of them contains a node for every edge in E; and the fourth layer contains a node for each edge of T_2. There is an arc from a node of the first layer to the second layer whenever the corresponding edges intersect or are identical. The third and fourth layer is connected in the same manner. There is an arc from a node in the second layer to a node in the third layer whenever the corresponding edges in E are identical. The arcs between the second and third layers carry costs which are equal to the lengths of the corresponding edges. The other arcs have cost 0.

All arcs have capacity 1, and we give a supply of 1 to each node in the first layer and a demand of 1 to each node in the fourth layer. Now, an integral flow in this network will decompose into unit flows running from sources to sinks along disjoint paths. The arcs between layers 1 and 2 which carry positive flow will induce the injective mapping g_1, and g_2 is obtained from the arcs between layers 3 and 4. The arcs between layers 2 and 3 ensure that $g_1(T_1) = g_2(T_2)$, and the cost of the flow is the total length of the edges in the set $g_1(T_1) = g_2(T_2)$. $\square$

When $T_1 = T_2$, we just get the single assignment bound (for the case when R is a triangulation). This means that, in general, the double assignment bound is stronger than the single assignment bound.

Similarly as in the previous subsection we obtain the following corollary.

Corollary 4 *Let g_1, g_2 be the optimal solution in* (3). *If the set $g_1(T_1) = g_2(T_2)$ is non-crossing then it is a minimum-weight triangulation.* $\square$

The following relaxation of the bound (3) is obtained from the double assignment bound by omitting the requirement that the functions g_i are injective, just as Theorem 9 is obtained from the single assignment bound.

Theorem 11 *Let T_1 and T_2 be two triangulations and let T^* be a minimum-weight triangulation. For $e \in T_1$ and $f \in T_2$ we define the cost $c(e, f)$ of assigning e to f as the length of the shortest edge which crosses or coincides with e and f. (Thus, if e and f cross, then both e and f are also candidates for the shortest edge.) If e and f do not cross and no common crossing edge exists, then $c(e, f) = \infty$. Let $\Pi(T_1, T_2)$ denote the set of all bijections between T_1 and T_2. Then*

$$w(T^*) \geq \min_{h \in \Pi(T_1,T_2)} \sum_{e \in T_1} c(e, h(e)). \tag{4}$$

Proof. From an optimal solution (g_1, g_2) of (3) in Theorem 10 we may obtain a feasible solution $h \in \Pi(T_1, T_2)$ by setting $h = g_2^{-1} \circ g_1$. This composition is well-defined since $g_1(T_1) = g_2(T_2)$. For all $e \in T_1$, the edge $g_1(e)$ crosses or coincides with e and $h(e)$, and hence $c(e, h(e)) \leq |g_1(e)|$. By summation of this inequality we get

$$\sum_{e \in T_1} c(e, h(e)) \leq \sum_{e \in T_1} |g_1(e)| \leq w(T^*). \qquad \square$$

The optimization problem (4) in the last theorem is just a standard assignment problem, i. e., a minimum-cost perfect matching problem in a bipartite graph. In terms of the network in the proof of Theorem 10, the weaker bound (4) can be obtained by giving the arcs from layer 2 to layer 3 unlimited capacity.

4.4 A matroid-theoretic interpretation

It is instructive to review the results of this section in terms of matroid theory. The single assignment bound can be formulated as the solution of a weighted matching problem in a bipartite graph with node classes R and E: we are looking for a matching which covers every node of R and has minimum total weight. However, in contrast to the standard setting of weighted matching problems, the costs are not associated with the *arcs* of the graph but with the *node* set E. This does not prevent standard matching algorithms from being applied, because we can set the cost of an arc (e, f) to $|f|$ and get the same result, but it opens an alternative approach.

The subsets of nodes in a graph which are matchable (for which there exists a matching which covers all of them) is a matroid, in our case of a bipartite graph a transversal matroid, see [L76, Corollary 7.4.3, p. 272, and Section 5.4, p. 192]. (This is in contrast to the independence system of the arc sets which form a matching.) Thus we have the problem of finding in the matroid a basis with minimum weight. It follows from basic matroid theory that it is possible to solve this problem by a greedy algorithm on the set E. We start with the empty matching and process the edges of E in order of increasing weight. For each edge, we try to augment the current matching to include the corresponding node. Such an augmentation may involve the replacement of some matching arcs by other arcs along an alternating path. A node which is matched in this process will never become unmatched. We may stop if we have $|R|$ matching pairs. Our efficient algorithm in Section 5.3 follows this approach.

In the double assignment bound, the solution set $g_1(T_1) = g_2(T_2)$ is in the intersection of *two* transversal matroids. There is a general algorithm for finding an optimal set in the intersection of any two matroids in polynomial time, see Lawler [L76, Chapter 8], whereas the matroid intersection problem for three matroids may already be NP-hard. This explains why the improvement from the single assignment bound to the double assignment bound cannot be pushed further to a "triple assignment bound" which is computable in polynomial time.

5 Algorithmic issues

In Sections 3 and 4 algorithmic issues were only discussed to the extent that the problems raised were reduced to well-studied combinatorial problems and polynomial time bounds were obvious. In some cases, we are able to improve over the time and space bound that one obtains by straightforward application of standard algorithms. Some familiarity with graph algorithms is desired for the understanding of this section.

5.1 Finding the matching efficiently

Let us consider the cost of computing the perfect matchings which are shown to exist in Theorems 1, 2, and 3. We first deal with Theorem 1. The crossings of a red triangulation R and a blue triangulation B of a set P of n points define a bipartite graph G, to which we can apply standard bipartite matching algorithms. This graph has $O(n)$ nodes and $K = O(n^2)$ arcs, where K is the total number of crossings between R and B plus the number of edges in $R \cap B$.

A direct application of the bipartite matching algorithm of Hopcroft and Karp [HK73] (see also [T87]) would result in $O(K\sqrt{n})$ time and $O(K)$ space. We will show how to reduce the space requirement to $O(n)$.

Theorem 12 *For two triangulations R and B of a planar point set P which have $K = O(n^2)$ crossings, we can find a perfect matching between the edge sets R and B which satisfies the condition of Theorem 1 in $O(K\sqrt{n}) = O(n^{5/2})$ time and $O(n)$ space.*

Proof. We have to show how the algorithm of Hopcroft and Karp can be carried out without explicitly storing the bipartite intersection graph G. For a detailed description of this algorithm we refer to the literature, but in order to enable the reader to see how the reduction of the space requirement works, we will present certain parts of the algorithm in more detail.

The algorithm of Hopcroft and Karp performs breadth-first search and depth-first search on certain directed subgraphs of G. The elementary operation in scanning a graph is accessing the next unexplored arc on the adjacency list of a given node. In [HK73], as in the standard setting of graph algorithms, it is assumed that the adjacency list of each node is stored explicitly as a sequential or linked list. Then it is easy to have a marker pointing to the current position in the list and to advance to the next arc when desired.

Although the algorithm works on subgraphs of G, it is not necessary to store explicit copies of these subgraphs, because when an arc of G is scanned, it can be checked in constant time whether it belongs to the subgraph in question, and if not, the arc is simply skipped.

In our geometric setting, we need not store adjacency lists at all but we extract them on-line. We store the two triangulations as plane graphs, such that standard operations like finding the two triangles incident to a given edge or finding the three edges bounding a given triangle can be executed in constant time. We also preprocess R and B to locate, for each point $p \in P$, the incident red edges among incident blue triangles. So we store for each red edge the first blue triangle which we enter as we walk from p along this edge. This can be done in $O(n)$ overall time by merging, for each p, the lists of red and blue edges in angular order around p. We also perform this preprocessing with the roles of red and blue reversed.

Now, to obtain the successive neighbors of, say, a red node e, we simply walk along the red edge it represents, proceeding from triangle to adjacent triangle in the blue triangulation and collecting all blue intersected edges. We can also "leave a marker" on e's adjacency list by remembering the current crossed blue edge. Later we can simply continue from there.

In this way, all graph exploration steps can be done in constant time without more than $O(n)$ storage. □

In case of face matchings (Theorem 3), we will see that the bipartite graph of "matchable" pairs of faces has only linearly many arcs. This leads to the following time bound.

Theorem 13 *For two triangulations R and B of a planar point set P, we can find a perfect matching between the triangular faces of R and B which satisfies the conditions of Theorem 3 (and hence also the condition of Theorem 2) in $O(n^{3/2})$ time and $O(n)$ space.*

Proof. Let G' be a bipartite graph with a node for each triangular face of R and B. Two nodes are connected by an arc if the corresponding triangles fulfill the two conditions of Theorem 3. We are looking for a perfect matching in G'.

A red triangle Δ_R can be matched with a blue triangle Δ_B if they have a common vertex p and share an angular region around p. Fix a vertex $p \in P$, and denote by $d_R(p)$ and $d_B(p)$ the *degree* of p in the triangulation R and B, respectively, i. e., the number of edges in R (or B) incident to p. We claim that the number of pairs (Δ_R, Δ_B) which can be matched because they share an angular region around p is at most $d_R(p) + d_B(p)$. Consider a ray sweeping around p. Each time it crosses an edge incident to p, we "generate" a new pair (Δ_R, Δ_B) consisting of the two triangles into which the ray points. Since we cross $d_R(p) + d_B(p)$ edges, we generate at most $d_R(p) + d_B(p)$ pairs. If we repeat this for all vertices p we generate all necessary pairs, i. e., all arcs of the graph G' in which we want to find the matching. Since $\sum_{p\in P} d_R(p) = \sum_{p\in P} d_B(p) = 2|R| = 2|B| = O(n)$, we generate only a linear number of arcs. The time bound of $O(n^{3/2})$ is now obtained by plugging this into the complexity of Hopcroft and Karp's algorithm. □

5.2 Computing excesses and light edges

Theorem 6 asks for a method to decide whether a given planar n-point set P admits a light triangulation, or in other words, whether the light edges triangulate P. Also, to compute the lower bound in Corollary 3, all light edges realized by P are required. More generally, the bound in Theorem 9 needs knowledge of the excess of all edges in a given non-crossing set.

We solve these problems in time $O(n^2 \log n)$ and space $O(n)$ by giving an algorithm for the following problem: given an arbitrary triangulation T of P, compute $\lambda(e)$ for each $e \in T$, where $\lambda(e)$ is the length of the shortest edge crossing e. As a by-product, all unavoidable edges are those with $\lambda(e)$ undefined.

We describe an $O(n \log n)$-time routine which, for a given point $p \in P$, computes

$$\lambda_p(e) = \min_{q\in P}\{\, |pq| : pq \text{ crosses } e \,\}$$

for all $e \in T$. Calling the routine for all $p \in P$ and maintaining the minimum for each e then gives $\lambda(e)$.

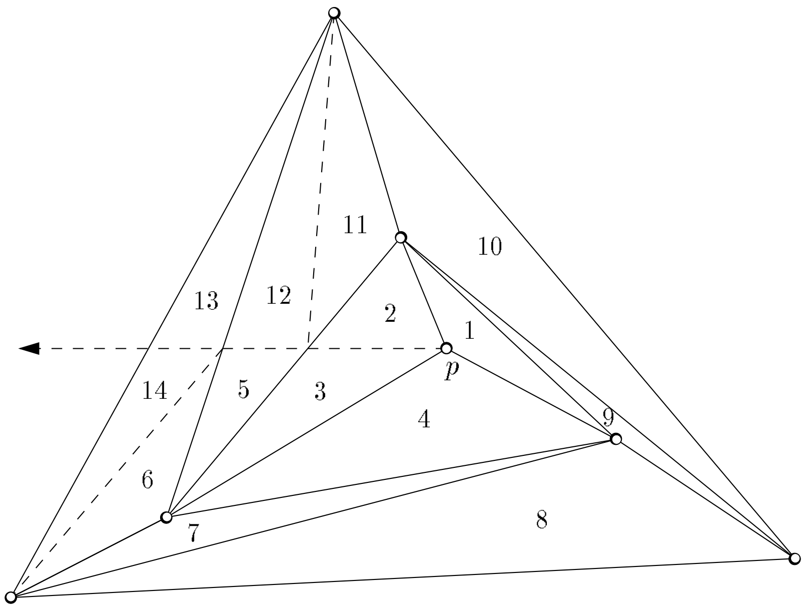

Figure 6: A triangulation which is cyclic when seen from p. The triangulation is cut by a horizontal ray from p to the left, and the resulting quadrangles are triangulated. The faces of the resulting triangulation are numbered in a topological order in which they can be processed. A (sub-)edge is processed when its incident triangle with the larger number is processed. Incidentally, when the point p is removed, the resulting triangulation is the minimum-weight triangulation, demonstrating that a minimum-weight triangulation is not necessarily "cycle-free".

Given the point p, we set up a semi-dynamic data structure for a point set Q and for the following type of queries: for a wedge V with apex p and angle less than π, return the closest point to p in $Q \cap V$. We initialize Q with P and allow only deletions from Q. The data structure we use is a binary search tree whose leaves store Q in cyclic order around p. Each interior node stores the minimum distance from p to all points in the subtree.

We then query the structure with wedges induced by edges e in the triangulation T. The queries and deletions are carried out in a specific order. This guarantees that, when we query for the wedge induced by edge e, Q contains no point that lies between e and p and includes all points that lie opposite to p with respect to e. To this end, a topological order for the edges of T with respect to the in-front/behind relation as seen from p is necessary. Problems may arise if this relation contains cycles, as shown in Figure 6. We circumvent this difficulty by cutting T with a ray emanating from p, and re-triangulating where necessary, in $O(n)$ time. For an edge e cut into subedges e' and e'' we clearly have $\lambda_p(e) = \min\{\lambda_p(e'), \lambda_p(e'')\}$.

A topological order is now obtained in $O(n)$ time by starting with the set of triangles incident to p, and adding adjacent triangles one by one while keeping the invariant that the boundary of their union U is star-shaped as seen from p. At each point in time, the search tree stores exactly the points of P exterior to U. As queries and deletions take $O(\log n)$ time each, the claimed $O(n \log n)$ time bound is obtained. A topological order of the edges is given in Figure 6.

Theorem 14 *Let P be a set of n points in the plane.*

(i) *Given a triangulation T of P, the excesses for all edges in T can be computed in $O(n^2 \log n)$ time and $O(n)$ space.*

(ii) *The excesses for all edges induced by P can be computed in $O(n^3 \log n)$ time and $O(n^2)$ space.*

(iii) *The set of light edges induced by P can be computed in $O(n^2 \log n)$ time and $O(n)$ space.*

Proof. Statement (i) follows from the preceding discussion. Statement (ii) follows from (i) since we can cover all edges by n triangulations: we simply connect a fixed point p to all other points and complete this to a triangulation. Repeating this for all points p, we will have covered every edge between two points of P at least once. For proving (iii), note that the greedy triangulation contains all light edges. So we first generate the greedy triangulation, and then check each of the $O(n)$ greedy edges for being light. Several algorithms exist which compute the greedy triangulation of P in the claimed time and space bounds, the easiest being perhaps the one of Goldman [G89]. The fastest algorithm for the greedy triangulation is due to Levcopoulos and Krznaric [LK94] and runs in $O(n \log n)$ time and $O(n)$ space; see also [DDMW94, DRA95] for fast expected-time algorithms. □

5.3 Computing the single assignment bound

The problem whose solution is required for computing the single assignment bound of Theorem 7 is an assignment problem between two edge sets R and E, where R is a given triangulation and E is the set of all edges between points of P. (R could be any non-crossing set but we restrict our attention to a triangulation.) The standard algorithm for the assignment problem leads to a complexity of $O(n^4)$ time and $O(n^3)$ space. However, we can follow the greedy approach outlined in Section 4.4 to obtain a faster algorithm that uses less space.

First, we generate the edges in E in increasing order of length until $|R|$ of them are matched. Dickerson et al. [DDS92] have shown that the m shortest edges in an n-point set P may be generated in $O((m+n)\log n)$ time and $O(m+n)$ space. By trying successively the values $m = n, 2n, 4n, 8n, \ldots$ we can make sure that we never generate more than twice as many edges as we actually need.

In the iterative step we have a partial matching between E and R, an edge $e_0 \in E$, and we try to augment the current matching to include the node corresponding to e_0.

Let us focus on the form of an augmenting path which we are looking for when processing an edge e_0. The path goes from e_0 to an adjacent node r_1 (an edge of R), from r_1 to the node e_1 to which r_1 is matched, from e_1 to an adjacent node r_2, from r_2 to the node e_2 to which r_2 is matched, and so on until it terminates in an unmatched node $r_i \in R$, $i \geq 1$. The augmentation consists of exchanging the $i-1$ matching arcs with the i non-matching arcs on this path.

We may search for such an augmenting path starting from e_0 using any graph search method, for example breadth-first search. We make several observations:

(i) We only need to explore the list of adjacent nodes for the edges of E; when we are at a node of R we simply proceed to the node of E to which it is matched. Since the neighbors of $e \in E$ are the edges of the triangulation R which cross e, we can find these neighbors by walking through the triangulation as described in Section 5.1, without any need to store adjacency lists. For each edge $e \in E$, there is an overhead of $O(\log n)$ for locating the first triangle in the triangulation R cut by e (as we walk along e from one endpoint).

(ii) Since we are using a greedy algorithm, if we fail to match a node of E, we can ignore it in the future. We need not remember to which nodes it is adjacent. Thus if $e \in E$ is not matched in the optimal solution, its incident arcs are explored only once during the whole algorithm.

(iii) If we have processed a node in R or E during an unsuccessful search for an augmenting path, we know that no unmatched node of R can be reached from this node, and we may skip the search from this node in all subsequent searches. This situation remains unchanged as long as no augmentation occurs. Thus, between two successful augmentations, every edge of the *current graph* is visited at most once. The current graph is the subgraph induced by the nodes R and the subset E' of nodes of E which are currently matched.

(iv) There are $|R| = O(n)$ successful augmentation steps. Hence, if $E_{\mathrm{opt}} = \{\, g_{\mathrm{opt}}(e) : e \in R\,\}$ denotes the set of matched nodes of E in the optimal matching g_{opt}, each of the arcs between R and E_{opt} is visited at most $O(n)$ times.

Putting everything together, we have the following theorem.

Theorem 15 *Let $E_{\mathrm{opt}} = \{\, g_{\mathrm{opt}}(e) : e \in R\,\}$ denote the set of edges to which the edges of R are assigned in the optimal matching g_{opt} of the single assignment bound. Let $K = O(n^2)$ denote the number of crossings between E_{opt} and R. Furthermore let $E_s \subseteq E$ be the set of edges which are not longer than the longest edge of E_{opt}, and let $L \le |E_s| \cdot |R| = O(n^3)$ denote the number of crossings between E_s and R. Then the optimal matching g_{opt} can be computed in $O(Kn+L+|E_s|\log n) = O((K+|E_s|)\,n) = O(n^3)$ time and $O(n+|E_s|) = O(n^2)$ space.*

Proof. By observation (iv), each of the K arcs is explored at most $O(n)$ times. In addition, each of the $L-K$ remaining arcs is explored only once. The term $|E_s|\log n$ accounts for generating the edges according to length and for locating the first triangle cut by each edge as mentioned in observation (i). □

The definition of E_s depends on the largest edge weight of E_{opt}. Since we do not know this beforehand, we may replace it, using Theorem 8, by the largest edge weight of R in the definition of E_s in order to get an a-priori upper estimate. In either case, unavoidable edges such as the boundary edges, which belong to every triangulation, can be ignored when computing the longest edge of E_{opt} or R.

The worst-case time bound of $O(n^3)$ is rather high, but the explicit parameters of the complexity indicate that for a good starting triangulation R the complexity might be quite good. Within the worst-case time of $O(n^3)$, E_s can also be enumerated in $O(n)$ space instead of $O(n+|E_s|)$ by a more primitive method.

Theorem 8 implies that we may skip every arc between $e \in E$ and a node $r \in R$ if r is shorter than e. This observation might speed up the algorithm, but we can make use of it only at the expense of storing the current graph.

5.4 Computing the double assignment bound

Theorem 16 *The double assignment bound of Theorem 10 for two triangulations T_1 and T_2 can be computed in $O(n \cdot (K_1 + K_2) + n^3 \log n) = O(n^4)$ time and $O(K_1 + K_2) = O(n^3)$ space, where K_1 (K_2) denotes the number of crossings between T_1 (T_2, respectively) and E.*

The relaxed double assignment bound of Theorem 11 can be computed in $O(n^3 + K_1 n) = O(n^4)$ time and $O(n^2)$ space.

Proof. As described in the proof of Theorem 10, the computation of the double assignment bound can be reduced to a minimum-cost network flow problem. The network has $2 \cdot |E| + O(n) = O(n^2)$ nodes and $K_1 + K_2 + O(n) + |E| = O(K_1 + K_2)$ arcs. We can easily construct it in $O(n^3)$ time by checking for each possible pair (e, g) of edges with $e \in T_1 \cup T_2$ and $g \in E$

whether it contributes an arc. The network can be stored in $O(K_1 + K_2)$ space. Since the total supply of the network is $|T_1|$, a minimum-cost flow can be computed in $|T_1| = O(n)$ flow augmentation steps along shortest augmenting paths, cf. [T87]. Each augmentation requires one shortest path computation in the network. A shortest path in a graph with v nodes and a arcs can be found in $O(a + v \log v)$ time. Therefore, $O(n)$ shortest path computations can be carried out in the claimed time complexity.

In the case of the relaxed bound of Theorem 11, most of the time is needed to compute the costs $c(e, f)$. In $O(n^3)$ time we can generate all $K_1 + O(n)$ pairs (e, g) of crossing or identical edges with $e \in T_1$ and $g \in E$. For each generated pair (e, g), we scan all edges $f \in T_2$, and if g crosses f or equals f, then g is one of the edges whose length contributes to the minimum in the definition of $c(e, f)$. This takes $O(n^3 + K_1 n)$ time. Finally, solving the assignment problem (4) takes $O(n^3)$ time. □

6 Conclusion and open problems

Theorems 1 and 2 are easily seen to hold for triangulations of arbitrary polygonal regions, possibly with holes. They can also be extended to triangulations of closed surfaces, when they are viewed as topological and graph-theoretic structures, as opposed to the geometric view taken in this paper.

It seems natural to try to prove Theorem 1 in a direct way, without resorting to the marriage theorem. For example, any triangulation of a planar point set can be changed into an arbitrary other one by repeated application of *edge flips*. An edge flip exchanges the diagonals of a convex quadrilateral in the current triangulation and thus naturally corresponds to a match of the involved edges. However, $\Omega(n^2)$ edge flips may be necessary to transform one triangulation of n points into another, see Hurtado, Noy, and Urrutia [HNU96]. This is an indication that there might be no proof of Theorem 1 based on the flipping paradigm.

We are planning to use our bounds in a branch-and-bound algorithm to compute the minimum-weight triangulation for arbitrary point sets. The bounds of Section 4 can be strengthened for subproblems where some specified edges are forced into the solution or excluded from the solution. We hope that a practically efficient algorithm for computing the minimum-weight triangulation may make experiments possible which lead to a better understanding of the properties of minimum-weight triangulations, with the ultimate goal of resolving the complexity status of the problem.

Another approach to compute the minimum-weight triangulation has recently been proposed by Dickerson and Montague [DM96]. They identify a subset of edges that must be part of every *locally minimal* triangulation. A locally minimal triangulation is a triangulation which cannot be improved by a single edge flip. Their very simple procedure — essentially they just use the definition of a locally minimal triangulation — is remarkably successful in identifying a large subset of edges which belongs to the minimum-weight triangulation. Levcopoulos and Krznaric [LK96] have recently obtained an approximation algorithm for the minimum-weight triangulation problem that achieves a constant approximation ratio.

Various open questions are raised by our results.

The point sets for which the light edges form a triangulation are interesting. What good properties do they and their associated triangulations have? Using a straightforward experimental program that finds and displays the light edges, we found it quite easy to draw light triangulations by choosing "well" distributed point sets. In fact, the light edges of a random point set chosen from a uniform distribution will in general leave just a few small polygonal regions which are not triangulated, see Figure 3. By putting a few more points into these regions, it is then not difficult to arrive at a point set with a light triangulation. This is how we obtained the point set in Figure 4. This leads to the following question. Given a planar n-point set P, can it always be extended by adding, say, $O(n)$ points so that it admits a light triangulation? Can we find such "enlightening" points in polynomial time? These questions might be interesting from the point of view of engineering applications, where only some points on the boundary rather than the complete set of points of the triangulation are fixed in advance.

In the single assignment bound it seems plausible that the "starting triangulation" R should have small weight (and therefore not be too different from the minimum-weight triangulation T^*) in order to get the best bounds. For the double assignment bound, however, we need two triangulations T_1 and T_2. Setting $T_1 = T_2$ does not yield an improvement over the single assignment bound. T_1 and T_2 should rather be chosen to be very different from each other, and thus they cannot both be close to T^*. How to choose T_1 and T_2 to get the best bounds is a question which is open to computational experiments.

Can we bound the quality of the bounds of Theorems 7 and 9 when R is, for instance, the greedy triangulation or the Delaunay triangulation? What is the maximum ratio between the two sides of the inequality?

The general Theorem 4 for independence systems can be applied for any special kind of independence system. However, we have not been able to find any interesting consequences of this matching result except for triangulations. Are there any applications to other areas?

The algorithms of Section 5.1 for finding matchings between edges or faces of two triangulations are the standard graph-theoretic matching algorithms. Geometry enters only in the implementation of adjacency lists in Theorem 12 and in the bound on the number of arcs in Theorem 13. It is conceivable that algorithms that exploit the geometric nature of the problem in a better way would be faster.

A triangulation can be viewed either as a set of edges or as a set of triangles. When we take the weight of a triangle to be its perimeter, the minimum-weight triangulation problem is identical for both formulations. In the triangle formulation, all edges are counted twice except for the boundary edges, which are counted once but are of fixed length. Nevertheless, we get two different greedy algorithms from the two formulations. We are not aware of any previous investigations of the "triangle-greedy" algorithm. Since there are $O(n^3)$ triangles but only $O(n^2)$ edges, it seems plausible that it may beat the usual "edge-greedy" algorithm in practice. We have examples which show that none of the two greedy algorithms beats the other for all problems. Some recent numerical investigations of many alternative greedy algorithms are reported in [AARX96]. For most point sets, the triangle-greedy algorithm does not beat the edge-greedy algorithm. This holds in particular for uniformly distributed point sets. How fast can the triangle-greedy triangulation be computed?

Acknowledgements. We thank Scot Drysdale, Herbert Edelsbrunner, Mordecai Golin, Bojan Mohar, András Pluhár, and Gerhard Woeginger for discussions, and a referee for many suggestions to improve the presentation.

This research was supported by the Spezialforschungsbereich F 003, *Optimierung und Kontrolle*. Research of Oswin Aichholzer was partially supported by the Austrian Ministry of Science and the Jubiläumsfond der Österreichischen Nationalbank. Research of Siu-Wing Cheng was partially supported by RGC grant HKUST 190/93E. Research of Yin-Feng Xu was partially supported by RGC grants HKUST 190/93E and HKUST 181/93E. Part of the work was done while Yin-Feng Xu visited the Department of Computer Science at the Hong Kong University of Science and Technology and the Institute for Theoretical Computer Science, Graz University of Technology.

References

[AART95] O. Aichholzer, F. Aurenhammer, G. Rote, and M. Taschwer, *Triangulations intersect nicely*, Proc. 11th Ann. Symp. on Computational Geometry, Vancouver, British Columbia, June 1995. Association for Computing Machinery, 1995; pp. 220–229.

[AARX96] O. Aichholzer, F. Aurenhammer, G. Rote, and Y.-F. Xu, *New greedy triangulation algorithms*, in preparation.

[B79] B. Bollobás, *Graph Theory. An Introductory Course*, Springer Verlag, 1979.

[B69] R. A. Brualdi, *Comments on bases in dependence structures*, Bull. Australian Math. Soc. **1** (1969), 161–167.

[B73] T. H. Brylawski, *Some properties of basic families of subsets*, Discrete Math. **6** (1973), 333–341.

[CX95] S.-W. Cheng and Y.-F. Xu, *Constrained independence system and triangulations of planar point sets*, in: D.-Z. Du, Ming Li, (eds.), Computing and Combinatorics, Proc. First Ann. Internat. Conf., COCOON'95, Xi'an, China, August 1995. Lecture Notes in Computer Science 959, Springer-Verlag, 1995, pp. 41–50.

[CX96] S.-W. Cheng and Y.-F. Xu, *Approaching the largest β-skeleton within a minimum-weight triangulation*, in: Proc. 12th Ann. Symp. on Computational Geometry, Philadelphia, Association for Computing Machinery, 1996, pp. 196–203.

[DJ89] G. Das and D. Joseph, *Which Triangulations Approximate the Complete Graph*, Proc. Internat. Symp. on Optimal Algorithms, Lecture Notes in Computer Science 401, Springer-Verlag, 1989, pp. 168–192.

[DDMW94] M. Dickerson, R. L. Drysdale, S. McElfresh, and E. Welzl, *Fast greedy triangulation algorithms*, Proc. 10th Ann. Symp. on Computational Geometry (1994), 211–220.

[DDS92] M. Dickerson, R. L. Drysdale, and J.-R. Sack, *Simple algorithms for enumerating interpoint distances and finding k nearest neighbors*, Internat. J. Computational Geometry & Appl. **3** (1992), 221–239.

[DM96] M. T. Dickerson and M. H. Montague, *A (usually?) connected subgraph of the minimum weight triangulation*, in: Proc. 12th Ann. Symp. on Computational Geometry, Philadelphia, Association for Computing Machinery, 1996, pp. 204–213.

[DRA95] R. L. Drysdale, G. Rote, and O. Aichholzer, *A simple linear time greedy triangulation algorithm for uniformly distributed points*, Report IIG-408, Institutes for Information Processing, Technische Universität Graz, February 1995, 16 pages.

[GJ79] M. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-completeness*, Freeman, 1979.

[G89] S. Goldman, *A space efficient greedy triangulation algorithm*, Inf. Process. Lett. **31** (1989), 191–196.

[HK73] J. E. Hopcroft and R. Karp, *An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs*, SIAM J. Comput. **2** (1973), 225–231.

[HNU96] F. Hurtado, M. Noy, and J. Urrutia, *Flipping edges in triangulations*, Proc. 12th Ann. Symp. on Computational Geometry, Philadelphia, Association for Computing Machinery, 1996, pp. 214–223.

[K94] M. Keil, *Computing a subgraph of the minimum weight triangulation*, Computational Geometry: Theory and Applications **4** (1994), 13–26.

[L76] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, 1976.

[LK94] C. Levcopoulos and D. Krznaric, *The greedy triangulation can be computed from the Delaunay in linear time*, Tech. Report LU-CS-TR:94-136, Dept. of Computer Science and Num. Analysis, Lund University, Lund, Sweden, 1994.

[LK96] C. Levcopoulos and D. Krznaric, *Quasi-greedy triangulations approximating the minimum weight triangulation*, in: Proc. 7th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA), Association for Computing Machinery, 1996, pp. 392–401.

[T87] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM Press, Philadelphia 1987.

[X92] Y.-F. Xu, *Minimum weight triangulation problem of a planar point set*, Ph. D. Thesis, Institute of Applied Mathematics, Academia Sinica, Beijing, 1992.

[YXY94] B.-T. Yang, Y.-F. Xu, and Z.-Y. You, *A chain decomposition algorithm for the proof of a property on minimum weight triangulations*, Proc. 5th International Symposium on Algorithms and Computation (ISAAC '94), Lecture Notes in Computer Science 834, Springer-Verlag, 1994, pp. 423–427.

[Y75] P. Yoeli, *Compilation of data for computer-assisted relief cartography*, in: J. C. Davis, M. J. McCullagh, (eds.), Display and Analysis of Spatial Data, Wiley, New York, 1975, pp. 352–367.