A New Approximation Algorithm for Finding Heavy Planar Subgraphs

Gruia Călinescu * Cristina G. Fernandes † Howard Karloff ‡ Alexander Zelikovsky §

Abstract

We provide the first nontrivial approximation algorithm for MAXIMUM WEIGHT PLANAR SUBGRAPH, the NP-hard problem of finding a heaviest planar subgraph in an edge-weighted graph G. This problem has applications in circuit layout, facility layout, and graph drawing. No previous algorithm for MAXIMUM WEIGHT PLANAR SUBGRAPH had performance ratio exceeding $1/3$, which is obtained by any algorithm that produces a maximum weight spanning tree in G. Based on the Berman-Ramaiyer Steiner tree algorithm, the new algorithm has performance ratio at least $1/3 + 1/72$ and at most $5/12$. We also show that if G is complete and its edge weights satisfy the triangle inequality, then the performance ratio is at least $3/8$. Furthermore, we derive the first nontrivial performance ratio ($7/12$ instead of $1/2$) for the NP-hard MAXIMUM WEIGHT OUTERPLANAR SUBGRAPH problem.

1 Introduction

MAXIMUM WEIGHT PLANAR SUBGRAPH is this problem: given a graph G with a nonnegative weight defined for each edge, find a planar subgraph of G of maximum weight. This problem—especially this weighted version—has applications in graph drawing, circuit layout, and facility layout.

Graph drawing is the process of drawing a graph, usually on a two-dimensional medium, as "nicely" as possible, where "nicely" is defined by the application. A long annotated bibliography on the large class of graph-drawing algorithms is found in [BETT94]. A plane embedding of a planar graph G could naturally be viewed as "nice." Graph-drawing experts have studied many different ways of drawing planar graphs in the plane, so as to maximize some optimization criterion [CON85, CYN85, LNS85, Tu63]. For a nonplanar graph G, one can start by drawing a planar subgraph of G, using some preferred method for drawing planar graphs [TDB88, BNTT84], and then drawing the remaining edges. *Planarization* is the process of obtaining a planar graph from a nonplanar one [CNS79, OT81, JTS86]. If different edges are not equally important, each edge might come with a weight reflecting the importance of drawing that edge; the edges in a maximum weight planar subgraph would be drawn first. Algorithms designed specifically for this weighted planar subgraph problem have been studied by Eades, Foulds and Griffin [EFG82], Jünger and Mutzel [JM93], Dyer, Foulds and Frieze [DFF85], and Foulds and Robinson [FR78]. Two other references dealing with finding planar subgraphs are [JM96] and [Li01].

Another application of MAXIMUM WEIGHT PLANAR SUBGRAPH is *architectural floor planning*: the laying out, in some optimal or near-optimal way, of a set of facilities [DFF85, F92, FR78, Mu94]. Krejcirik [Kr69] presented the first graph-theoretic approach to architectural floor planning, which sometimes yielded improved solutions [HW85]. For each pair of facilities (i, j), m_{ij} denotes the desirability of placing i adjacent to j. The "desirability" of the layout is defined to be the sum of the m_{ij}'s of adjacent facilities i, j. The goal is to produce a layout with sum as high as possible. This problem can be modeled as an instance of MAXIMUM WEIGHT PLANAR SUBGRAPH: the vertices are the facilities and the weight of the edge between facilities i and j is m_{ij}.

Architectural floor planning—graph-theoretic and not—has traditionally been used in the design of industrial plants [DFF85, F92, FR78, Mu94], but also in the design of airports [BFP95], hospitals [E77], universities [DH72], and libraries [FT86]. Often m_{ij} is the number of people expected to travel from facility i to facility j.

Another important application of MAXIMUM WEIGHT PLANAR SUBGRAPH is *circuit layout* [Ma78, Mu94]. The goal is to lay out a circuit on a one-layer board to maximize the number of connections between components. Now m_{ij} represents the number of desired connections between component i and component j. The maximum weight planar subgraph gives a largest subset of connections that can be laid out without wire crossings on a board.

*Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616; calinesc@iit.edu. Research performed while at the Georgia Institute of Technology and supported in part by NSF grant CCR-9319106.

†Department of Computer Science, University of São Paulo, Rua do Matao, 1010, 05508-900 Sao Paulo - Brazil; cris@ime.usp.br. Research supported in part by the CNPq (Brazil), under contract 200975/92-7, FAPESP (Proc. 96/04505–2) and ProNEx 107/97 MCT/FINEP. Work performed while at the Georgia Institute of Technology.

‡AT&T Laboratories–Research, Room C231, 180 Park Ave., Florham Park, NJ 07932; howard@research.att.com. Research performed while at the Georgia Institute of Technology and supported in part by NSF grant CCR-9319106.

§Department of Computer Science, Georgia State University, Atlanta, GA 30303; alexz@cs.gsu.edu. Research performed while at the University of Virginia and supported in part by the Packard Foundation.

These applications justify the interest in solving MAXIMUM WEIGHT PLANAR SUBGRAPH. Unfortunately, because MAXIMUM WEIGHT PLANAR SUBGRAPH is NP-hard [LG77], a polynomial-time exact algorithm is unlikely to exist. For this reason, we study approximation algorithms for MAXIMUM WEIGHT PLANAR SUBGRAPH.

We define the *performance ratio* of an algorithm A that takes weighted graphs G and produces planar subgraphs H of G as the infimum, over all weighted graphs G, of the ratio between the weight of H and the weight $Opt(G)$ of a heaviest planar subgraph of G; the performance ratio is at most one. In fact, MAXIMUM WEIGHT PLANAR SUBGRAPH is known to be Max SNP-hard even when all the edge weights are one [CFFK98]. This means there is an $\epsilon > 0$ such that no approximation algorithm achieves a ratio of $1-\epsilon$ for the problem, unless P=NP. The best (largest) known ϵ, however, is tiny, making $1-\epsilon$ very far from the best previously known performance ratio for MAXIMUM WEIGHT PLANAR SUBGRAPH, which was $1/3$ [Ci97]. Any algorithm that produces a maximum weight spanning tree (see Section 3.4) achieves the $1/3$ ratio. See [DFF85] for three algorithms for the weighted case and [CFFK98] for a list of three others for the unweighted case, all having performance ratio at most $1/3$. In this paper, we present the first algorithm for MAXIMUM WEIGHT PLANAR SUBGRAPH to cross the $1/3$ threshold: its performance ratio is at least $1/3 + 1/72$. We also prove that its performance ratio is at most $5/12$.

In the unweighted case, any algorithm producing any spanning tree of G has performance ratio $1/3$ (a spanning tree has $n-1$ edges and a planar subgraph has at most $3n-3$), and a recent paper exhibited a polynomial-time algorithm with a performance ratio of $4/9$ [CFFK98]. However, the weighted case seems much more difficult. The key idea in the algorithms of [CFFK98] is to produce a graph all of whose cycles are triangles (a *triangular structure*, which is necessarily planar), having many triangles. Each triangle gives one more edge than a spanning tree would have—the more triangles, the better. A simple algorithm greedily choosing triangles and then connecting them achieves $1/3+\epsilon$ for a fixed positive ϵ [CFFK98]. To get $4/9$, one finds a *largest* triangular structure in the input graph, which, surprisingly, can be done in polynomial time.

Unfortunately, the algorithms for the unweighted case do not work in the weighted case. First, one clearly cannot take arbitrary triangles, regardless of their weight. One must focus, of course, on heavy triangles. How? Recall that in the unweighted case, one can choose triangles (never creating any cycles of length exceeding three) and then arbitrarily tie together the components into a connected graph. In the weighted case, focusing on heavy triangles early on, without regard for the later step of *connecting* the triangles, might force the algorithm to use overly light edges to connect the triangles. Second, it is not known whether a *heaviest* triangular structure in G can be found in polynomial time. Graphical matroid parity is used to find a largest triangular structure in an unweighted graph [CFFK98], graphical matroid parity being solvable in polynomial time [GS85]; *weighted* graphical matroid parity is not known to be NP-hard or in P. Third, although in the unweighted case, proving that a heaviest (i.e., largest) triangular structure in G has weight at least $(1/3+\epsilon)Opt(G)$ is not hard (indeed, a greedily-constructed triangular structure suffices), whether the same property held for weighted graphs was not known. In fact, we can prove that this is so only indirectly by noting that our algorithm has performance ratio at least $1/3 + 1/72$, and that its output is a triangular structure.

For most approximation algorithms proposed recently, the performance ratios for the weighted cases are the same as those for the unweighted cases [CST95, S95, GW95, FG95, Ho82, Ho97]. We, however, can prove only that the performance ratio for MAXIMUM WEIGHT PLANAR SUBGRAPH is at least $1/3 + 1/72$, whereas the performance ratio for MAXIMUM WEIGHT PLANAR SUBGRAPH in the unit-weight case is at least $4/9$. Of course, these are only the best bounds *known*. We leave to the reader the (open) problem of proving that the best *possible* performance ratios are the same.

Our algorithm provides two other results. First, in the important special case in which G is a complete graph whose edge weights satisfy the triangle inequality, we can prove a better performance ratio of $3/8 = 0.375$. Second, since our algorithm produces triangular structures and any triangular structure is outerplanar (i.e., can be drawn in the plane with all vertices on the boundary of the outer face), we have an approximation algorithm for MAXIMUM WEIGHT OUTERPLANAR SUBGRAPH. We can prove a performance ratio of at least $7/12$, whereas producing a spanning tree gives a performance ratio of $1/2$. (The best ratio known in the unweighted case is $2/3$ [CFFK98].)

Our algorithm exploits an analogy between Steiner trees and triangular structures. Modeled on the two-phase Berman-Ramaiyer Steiner tree algorithm [BR94], the algorithm is clearly polynomial-time. However, its analysis is difficult. A "road map" guiding the reader through the analysis appears in Section 3.1.

2 The Algorithm

MAXIMUM WEIGHT PLANAR SUBGRAPH is this problem: given a graph G with a nonnegative weight defined for each edge, find a planar subgraph of G of maximum weight.

Let $G = (V, E)$ be a connected graph with a nonnegative weight function w on the edges. Recall that a triangular structure is a graph all of whose cycles (if any) are triangles. A *triangular structure in a graph G* is a subgraph of G which is a triangular structure. We will describe an algorithm which produces a heavy triangular structure in G.

Before precisely describing the algorithm, we need some notation. The vertex sets in all graphs will be obvious, so we will identify a graph with its edge set.

The algorithm has two phases, the *evaluation phase* and the *construction phase*. The evaluation phase (executed first) maintains a tree whose edges are not only from the original graph, but also some *artificial* edges. The artificial edges have weight, also denoted by w. We denote by N_i the set of two artificial edges added in the i^{th} augmentation (which will be defined soon).

We start with T_0, a maximum spanning tree in $G = G_0$. We will maintain the invariant that T_i is a maximum spanning tree in $G_i = (V, E \cup (\cup_{j=1}^{i} N_j))$. G_i is, in fact, a multigraph.

Let Δ be a triangle of G, the original graph. In the spanning tree T_{i-1}, there is a unique vertex s such that the paths in T_{i-1} from s to each of the three vertices of Δ are edge-disjoint. For each of these three paths (at most one of which may be trivial), choose an edge of minimum weight on the path, if possible. Now label the vertices of Δ as x, y, z such that the weight of the minimum weight edge e_1 on the $s-x$ path is at most the weight of the minimum weight edge e_2 on the $s-y$ path, which is at most the weight of the minimum weight edge e_3 on the $s-z$ path. (If this last path is trivial, leave e_3 undefined and, by an abuse of notation, say that $w(e_3) = \infty$.) The edges of the triangle Δ are unnamed.

One can easily check that e_1 and e_2 are two edges of T_{i-1} such that each of the three connected components of $T_{i-1} \setminus \{e_1, e_2\}$ contains one vertex of Δ. Moreover, among all the choices of pairs (e, f) of edges of T_{i-1} whose removal pairwise disconnects x, y, and z, the pair (e_1, e_2) has the least total weight. We define the *gain $g_{i-1}(\Delta)$ of the triangle Δ with respect to T_{i-1}* to be $g_{i-1}(\Delta) = w(\Delta) - w(e_1) - w(e_2)$, where by $w(\Delta)$ we mean the sum of the weights of the three (original) edges of the triangle Δ.

We call the following operation *augmenting the tree T_{i-1} by the triangle Δ*. This is done only if $g_{i-1}(\Delta) > 0$, so let us assume now that $g_{i-1}(\Delta) > 0$. An augmentation produces a tree T_i of greater weight.

Let e'_1 be a new artificial edge with endpoints x and z and $w(e'_1) = w(e_1) + g_{i-1}(\Delta)$. Let e'_2 be a new artificial edge with endpoints y and z and $w(e'_2) = w(e_2) + g_{i-1}(\Delta)$. Note that both e'_1 and e'_2 are parallel to existing (unnamed) edges of the triangle Δ. Let $N_i = \{e'_1, e'_2\}$. Then replace e_1 by e'_1 and replace e_2 by e'_2 to get T_i: $T_i = (T_{i-1} \setminus \{e_1, e_2\}) \cup N_i$.

Note that e_1 and e_2 might themselves be artificial edges. Because vertices x, y, z are in separate components of $T_{i-1} \setminus \{e_1, e_2\}$, T_i is a tree. We will see later that T_i is a maximum spanning tree of G_i.

2.1 The Evaluation Phase

The evaluation phase starts with a maximum spanning tree T_0 in $G_0 = G$, and sets $\mathcal{T}$ to be the set of all triangles in G. Repeatedly, the algorithm removes any one triangle Δ from $\mathcal{T}$, and calculates its gain $g_{i-1}(\Delta)$ relative to the current tree T_{i-1}. If nonpositive, it does nothing, while if positive, it augments T_{i-1} by Δ to get T_i (and N_i), sets $\Delta_i \leftarrow \Delta$, and increments i. This process continues until $\mathcal{T}$ is exhausted.

Here is the evaluation phase.

Evaluation Phase

Let T_0 be a maximum spanning tree in G; $i \leftarrow 1$
Let $\mathcal{T}$ be the set of all triangles in G
while $\mathcal{T} \neq \emptyset$ **do**
    Choose any $\Delta \in \mathcal{T}$
    if $g_{i-1}(\Delta) > 0$ **then**
        Augment T_{i-1} by Δ to get T_i and N_i
        $\Delta_i \leftarrow \Delta$
        $i \leftarrow i + 1$
    endif
    $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\Delta\}$
endwhile
$f \leftarrow i - 1$.

2.2 The Construction Phase

The goal of the construction phase is to use the spanning tree T_f, the sets $N_f, N_{f-1}, N_{f-2}, \ldots, N_1$, and the triangles $\Delta_f, \Delta_{f-1}, \Delta_{f-2}, \ldots, \Delta_1$ from the evaluation phase to build a connected triangular structure in $G_0 = G$. This is done iteratively by maintaining a connected triangular structure S_i in G_i, for $i = f, f-1, f-2, \ldots, 0$, where

$G_i = (V, E \cup (\cup_{j=1}^{i} N_j))$. The spanning tree T_f in G_f serves as the connected triangular structure S_f in G_f to get us started. From that point on, in the i^{th} iteration ($i = f, f-1, \ldots, 1$), consider the 2-element set N_i. We must ensure that $N_i \cap S_{i-1} = \emptyset$, since S_{i-1} must be a subgraph of G_{i-1} and the edges of N_i do not appear in G_{i-1}. What happens next depends on $|N_i \cap S_i|$. If $|N_i \cap S_i| = 0$, we simply set $S_{i-1} \leftarrow S_i$; $|S_{i-1}| = |S_i|$. If $|N_i \cap S_i| = 1$, then we set $S_{i-1} \leftarrow (S_i \setminus N_i) \cup \{h\}$, where h is a heavy edge chosen to guarantee that S_{i-1} is connected; again, $|S_{i-1}| = |S_i|$. The most interesting case is the one in which $|N_i \cap S_i| = 2$, that is, $N_i \subseteq S_i$. Then we set $S_{i-1} \leftarrow (S_i \setminus N_i) \cup \Delta_i$. Since $|\Delta_i| = 3$, we have $|S_{i-1}| = |S_i| + 1$. It is through this step that we add triangles to the final S_0 and improve on the maximum spanning tree.

Here is the construction phase.

Construction Phase

$S_f \leftarrow T_f$
for $i = f$ **downto** 1 **do**
    if $|N_i \cap S_i| = 0$ **then** $S_{i-1} \leftarrow S_i$ /* $|S_{i-1}| = |S_i|$ */
    if $|N_i \cap S_i| = 1$ **then**
        $S_i' \leftarrow S_i \setminus N_i$ /* S_i' has two connected components */
        Find edge h of G_{i-1} of maximum weight with endpoints in different components of S_i'
        $S_{i-1} \leftarrow S_i' \cup \{h\}$ /* $|S_{i-1}| = |S_i|$ */
    endif
    if $|N_i \cap S_i| = 2$ **then** $S_{i-1} \leftarrow (S_i \setminus N_i) \cup \Delta_i$ /* $|S_{i-1}| = |S_i| + 1$ */
endfor
Output S_0

3 Analysis

3.1 Road Map

The output of our algorithm, when the input is a weighted connected graph G, is S_0. If S is a subgraph of G or a set of edges of G, then we denote by $w(S)$ the sum of $w(e)$ over all e in S. Our analysis proceeds as follows.

1. First, we prove that S_0 is a (connected) triangular structure in G (Section 3.2). This ensures that the output is a planar subgraph of G.

2. We prove in Section 3.3 that the weight of S_0 is at least the average of the weight of T_0, a maximum spanning tree in G, and the weight of T_f, a spanning tree (indeed, *maximum* spanning tree) in the supergraph G_f of G obtained during the evaluation phase.

   We will see shortly that $w(T_0) \geq (1/3)Opt(G)$, where $Opt(G)$ is the weight of a heaviest planar subgraph of G. We circuitously prove that $w(S_0) \geq (1/3 + 1/72)Opt(G)$ via two lower bounds on $w(T_0)$ (the trivial bound that $w(T_0) \geq (1/3)Opt(G)$ and a more difficult one), and an easy lower bound on $w(T_f)$.

3. In Section 3.4, we prove the trivial fact that $w(T_0) \geq (1/3)w(P)$ for any planar subgraph P of G.

4. Section 3.5 contains a plethora of lemmas needed to prove the easy lower bound on $w(T_f)$.

5. In Section 3.6, we introduce the concept of a "good" set $\mathcal{S}$ of triangles in a plane graph P. Let $b(\Delta)$ denote the median weight of the edges in triangle Δ. Section 3.6.1 provides a proof of the easy lower bound on $w(T_f)$:

$$w(T_f) \;\geq\; \frac{1}{3}w(P) + \frac{1}{12}\left[\sum_{\Delta \in \mathcal{S}} b(\Delta)\right]$$

   provided only that $\mathcal{S}$ is a good set of triangles in a plane subgraph P of G.

6. In Section 3.6.2, the reader will find the technically most important fact of the analysis, Corollary 20, which has Lemma 19 as its crux. Corollary 20 states that if P is a weighted plane subgraph of G, then there is a good family $\mathcal{S}$ of triangles of P such that

$$w(T_0) \;\geq\; \frac{1}{2}w(P) - \frac{1}{2}\left[\sum_{\Delta \in \mathcal{S}} b(\Delta)\right].$$

This is the difficult lower bound.

7. Letting P be any plane subgraph of G and letting $\beta = \sum_{\Delta \in \mathcal{S}} b(\Delta)$, for the good family $\mathcal{S}$ of triangles of P given by Corollary 20, we have

   (a) $w(T_0) \geq (1/3)w(P)$,

   (b) $w(T_0) \geq (1/2)w(P) - (1/2)\beta$, and

   (c) $w(T_f) \geq (1/3)w(P) + (1/12)\beta$.

   The last two conditions and $w(S_0) \geq (1/2)[w(T_0) + w(T_f)]$ say that $w(S_0)$ is at least $(1/3 + \epsilon)w(P)$ unless β is large. However, when β is large, the first and third conditions, taken with $w(S_0) \geq (1/2)[w(T_0) + w(T_f)]$, say that $w(S_0) \geq (1/3 + \epsilon)w(P)$. The formal statement that

   $$\frac{1}{2}\left[w(T_0) + w(T_f)\right] \;\geq\; \left(\frac{1}{3} + \frac{1}{72}\right) w(P),$$

   whatever the value of β, appears in Section 3.7. Taking P to be a heaviest planar subgraph of G, we are done.

8. Section 3.9 proves that the performance ratio is at least 3/8 if the edge weights in the input graph G satisfy the triangle inequality.

9. Section 3.10 gives a 7/12 bound for the problem of finding a heaviest outerplanar subgraph in G. A ratio of 1/2 is obtained by any maximum spanning tree.

3.2 S_0 is a Connected Triangular Structure in G

Let us concentrate on the construction phase. When going from S_i to S_{i-1}, we make sure that $S_{i-1} \cap N_i = \emptyset$; thus S_{i-1} is a subgraph of G_{i-1}. In particular, S_0 is a subgraph of $G = G_0$. The fact that S_i is a connected triangular structure for all i follows from the following lemma.

Lemma 1 *For all $i \in \{f, f-1, f-2, \ldots, 0\}$, S_i is a connected graph all of whose cycles are triangles in the original graph G.*

Proof. S_f is a spanning tree, so for $i = f$, the lemma is true. Let us assume that the lemma holds for some i, $f \geq i \geq 1$, and prove it holds for $i - 1$. We follow the cases of each iteration of the construction phase.

If $|N_i \cap S_i| = 0$, then $S_{i-1} = S_i$, which has the desired properties.

Suppose now that $|N_i \cap S_i| = 1$. Then S_i' is obtained from S_i by removing an artificial edge. This edge, being artificial, by the inductive hypothesis does not belong to any cycle. It follows that S_i' has two components. Then in line 7 one edge is added between the two components of S_i'. Hence this new edge does not belong to any cycle of S_{i-1} and therefore all cycles of S_{i-1} are cycles of S_i. Also, S_{i-1} is connected.

Suppose for the remainder of the proof that $|N_i \cap S_i| = 2$. We use the notation from the definition of augmentation. $N_i = \{e_1', e_2'\}$, where e_1' has endpoints x and z and e_2' has endpoints y and z. By the inductive hypothesis, since both e_1' and e_2' are artificial edges, neither appears in any cycle of S_i. Let f_1, f_2, and e be the edges of the original (simple) graph with endpoints x and z, y and z, and x and y, respectively. Consider $S' = (S_i \setminus N_i) \cup \{f_1, f_2\}$ (the two edges of N_i are replaced by two edges parallel to them). Then in S' all cycles are triangles of the original graph G and the edges f_1 and f_2 do not appear in any cycle. We have $S_{i-1} = S' \cup \{e\}$.

We know that before adding e, f_1 and f_2 did not appear in any cycle. We will show that adding e only creates the cycle $< f_1, f_2, e >$. Any cycle we might create by adding e has to contain e, of course. Suppose, for a contradiction, that adding e creates a cycle C other than $< f_1, f_2, e >$. Assume first that C contains neither f_1 nor f_2. By replacing e in C by f_1 and f_2, we either get a cycle containing f_1 and f_2 but not e, a contradiction, or a union of two cycles (this is the case in which C goes through the common endpoint of f_1 and f_2). But one of these cycles contains f_1 and does not contain e, a contradiction. If C contains f_1 and e, but not f_2, then by replacing f_1 and e in C by f_2, we get a cycle containing f_2 and not e, contradicting the fact that before adding e, f_1 and f_2 did not appear in any cycle. The case in which C contains f_2 and e, but not f_1, is similar.

It follows that in S_{i-1} all cycles are triangles of the original graph. Also, since S_i is connected, S' is connected and therefore S_{i-1} is connected. ■

Corollary 2 *S_0 is a connected triangular structure in G and hence planar.*

3.3 A Lower Bound on the Weight of S_0

We need the following lemma and its corollary, analogues of Steiner tree Lemmas 3.5 and 3.7 in [BR94].

Lemma 3 *For $i = 1, \ldots, f$, $w(S_{i-1}) \geq w(S_i) - g_{i-1}(\Delta_i)$.*

Proof. The proof follows the three cases from each iteration of the construction phase. As above, let $\Delta_i = xyz$ and let e_1 and e_2 be the two edges removed from T_{i-1} during the i^{th} augmentation. $N_i = \{e'_1, e'_2\}$ is the set of artificial edges added during the i^{th} augmentation. Also $w(e'_1) = w(e_1) + g_{i-1}(\Delta_i)$ and $w(e'_2) = w(e_2) + g_{i-1}(\Delta_i)$.

If $|N_i \cap S_i| = 0$, then $S_{i-1} = S_i$ and the lemma follows from the fact that $g_{i-1}(\Delta_i) > 0$.

If $|N_i \cap S_i| = 2$, then $w(S_{i-1}) = w(S_i) - w(e'_1) - w(e'_2) + w(\Delta_i) = w(S_i) - w(e_1) - g_{i-1}(\Delta_i) - w(e_2) - g_{i-1}(\Delta_i) + w(\Delta_i) = w(S_i) - g_{i-1}(\Delta_i)$, since $g_{i-1}(\Delta_i) = w(\Delta_i) - w(e_1) - w(e_2)$.

Now suppose that $|N_i \cap S_i| = 1$. We only consider the case $N_i \cap S_i = \{e'_1\}$; in the case $N_i \cap S_i = \{e'_2\}$, a similar argument applies. The endpoints of e'_1 are x and z and therefore in $S'_i = S_i \setminus \{e'_1\}$, x and z are in two different components (by Lemma 1). Consider the (unique) path in T_{i-1} from x to z. There is at least one edge on this path which has endpoints in two different components of S'_i; choose such an edge and call it e. We conclude that the edge h selected by the algorithm in the construction phase satisfies $w(h) \geq w(e)$. Note also that e_1 is the minimum weight edge on the path in T_{i-1} from x to z, so $w(e_1) \leq w(e)$. Then we have $w(h) \geq w(e_1) = w(e'_1) - g_{i-1}(\Delta_i)$. We conclude that $w(S_{i-1}) = w(S_i) - w(e'_1) + w(h) \geq w(S_i) - g_{i-1}(\Delta_i)$. ■

Corollary 4 *$w(S_0) \geq \frac{1}{2}(w(T_0) + w(T_f))$.*

Proof. It is immediate that for $i = 1, \ldots, f$, $w(T_i) = w(T_{i-1}) + 2g_{i-1}(\Delta_i)$, and thus $w(T_f) = w(T_0) + 2\sum_{i=1}^{f} g_{i-1}(\Delta_i)$. It follows, by Lemma 3, that $w(S_0) \geq w(S_f) - \sum_{i=1}^{f} g_{i-1}(\Delta_i) = w(T_f) - \sum_{i=1}^{f} g_{i-1}(\Delta_i) = w(T_0) + \sum_{i=1}^{f} g_{i-1}(\Delta_i) = \frac{1}{2}(w(T_0) + w(T_f))$. ■

3.4 A Trivial Lower Bound on the Weight of a Maximum Spanning Forest

The proof of the next lemma uses the concept of *arboricity*, defined below, and a theorem by Nash-Williams.

Definition 1 The *arboricity* of a graph is the minimum number of spanning forests into which its edge set can be partitioned.

We have this classic theorem:

Theorem 5 *(Nash-Williams; see [N64, CMWZZ94]) The arboricity of a graph G is the maximum, over all subgraphs H of G with at least two vertices, of*

$$\left\lceil \frac{|E(H)|}{|V(H)| - 1} \right\rceil.$$

Lemma 6 *Let $\mathcal{F}$ be a family of graphs closed under taking subgraphs, such that, for some positive integer c, $|E(G)| \leq c(|V(G)| - 1)$ for all G in $\mathcal{F}$. Let w be a nonnegative weight function on the edges of some G in $\mathcal{F}$, and F be a maximum spanning forest in G. Then $w(F) \geq \frac{1}{c}w(G)$.*

Proof. Let G be a graph in $\mathcal{F}$. We have $|E(H)| \leq c(|V(H)| - 1)$ for any subgraph H of G, because $\mathcal{F}$ is closed under taking subgraphs. Thus, by Theorem 5, the arboricity of G is at most c.

This means that the edge set of G can be partitioned into c forests $F_1, \ldots, F_c$. Clearly we have $w(F_1) + \cdots + w(F_c) = w(G)$. But $w(F) \geq w(F_i)$ for all i, and this implies that $w(G) = w(F_1) + \cdots + w(F_c) \leq c \cdot w(F)$. ■

The proof method of Lemma 17 gives an alternative way to prove Lemma 6 without relying on arboricity.

Corollary 7 *(See [DFF85, Theorem 3.3]) Let P be a simple planar graph with a nonnegative weight function w defined on its edges. If F is a maximum spanning forest in P, then $w(F) \geq \frac{1}{3}w(P)$.*

Proof. Apply Euler's formula ($|E(P)| \leq 3(|V(P)| - 1)$) and Lemma 6 with $c = 3$ to the family of simple planar graphs. ■

Recall that T_0 is a maximum spanning tree in G. As a consequence of Corollary 7, we have $w(T_0) \geq (1/3)w(P)$, for any planar subgraph P of G.

3.5 Some Lemmas

Definition 2 If T is a spanning tree in a connected multigraph H, and c and d are vertices of H, let $index_T(c, d)$ denote the weight of the lightest edge on the $c - d$ path in T.

Lemmas 8 through 12 are used solely in the proof of Lemma 13.

Lemma 8 *Let H be a connected multigraph and let T be a maximum spanning tree in H. Let $a \neq b$ be two vertices. Let e be a minimum weight edge on the $a - b$ path in T. Let e' be a new edge with endpoints a, b and of weight $w(e') \geq w(e) = index_T(a, b)$. Define $H' = H \cup \{e'\}$ and $T' = (T \cup \{e'\}) \setminus \{e\}$. Then T' is a spanning tree in H',*

1. *for all vertices u, v, $index_{T'}(u, v) \geq index_T(u, v)$, and*

2. T' is a maximum spanning tree in H'.

Proof. That T' is a spanning tree in H' is obvious.

Let C be the unique cycle in $T \cup \{e'\} = T' \cup \{e\}$. Let u and v be two vertices and let P and P' be the unique paths in T and T', respectively, between u and v. Let us prove part 1.

If P does not contain e, then $P = P'$ and therefore $index_{T'}(u,v) = index_T(u,v)$.

Assume now that P contains e. Then P is composed of three paths: P_1, from u to the first vertex of C on P; P_2, a segment of C containing e; and P_3, from the last vertex of C on P to v. P_1 or P_3 may be trivial. Similarly, P' is composed of three paths: P_1, P_2' and P_3, where P_2' is a segment of C containing e'. Let f be an edge of minimum weight on P', so that $w(f) = index_{T'}(u,v)$.

If f is on P_1 or P_3, then f is on P and therefore $index_{T'}(u,v) = w(f) \geq index_T(u,v)$.

If f is on P_2', then $w(f) \geq w(e)$ (e is the minimum weight edge on C since $w(e') \geq w(e)$). Since e is on P, $w(e) \geq index_T(u,v)$ and therefore $w(f) \geq index_T(u,v)$. Hence $index_{T'}(u,v) \geq index_T(u,v)$. The proof of part 1 is complete.

The statement that T' is a maximum spanning tree in H' is equivalent to the statement that T' is a spanning tree in H' and for any edge f of H' with, say, endpoints u and v, $w(f) \leq index_{T'}(u,v)$; an analogous statement holds for T and H [AMO93, Theorem 3.3].

Because T is a maximum spanning tree in H, for all edges f in H, with, say, endpoints u and v, $w(f) \leq index_T(u,v) \leq index_{T'}(u,v)$. Also, $w(e') = index_{T'}(a,b)$, because e' has endpoints a and b, and $e' \in T'$. It follows that for all $f \in H'$, with endpoints u, v, $w(f) \leq index_{T'}(u,v)$. This makes T' a maximum spanning tree in H'. ■

Lemma 9 *For $i = 0, 1, 2, \ldots, f$, T_i is a maximum spanning tree in G_i, and for all $i \geq 1$ and all vertices $u \neq v$, $index_{T_{i-1}}(u,v) \leq index_{T_i}(u,v)$.*

Proof. By induction on i, the case of $i = 0$ being trivial. Choose $i \geq 1$ and assume that T_{i-1} is a maximum spanning tree in G_{i-1}.

We can think of the augmentation of the tree T_{i-1} by a triangle Δ_i of positive gain $g_{i-1}(\Delta_i)$ as taking place in two steps. In the first step, a new artificial edge e_1' with endpoints x, z and weight $w(e_1') = w(e_1) + g_{i-1}(\Delta_i) > w(e_1)$ is added to the tree T_{i-1} and to G_{i-1}, and then e_1 is removed from the tree. Let $T_{i-1}' = (T_{i-1} \cup \{e_1'\}) \setminus \{e_1\}$ be the resulting tree and $G_{i-1}' = G_{i-1} \cup \{e_1'\}$ be the resulting graph. In the second step, an artificial edge e_2' with endpoints y, z and weight $w(e_2') = w(e_2) + g_{i-1}(\Delta_i) > w(e_2)$ is added to T_{i-1}' and to G_{i-1}'; next, e_2 is removed from the tree. $T_i = (T_{i-1}' \cup \{e_2'\}) \setminus \{e_2\}$ is the resulting tree and $G_i = G_{i-1}' \cup \{e_2'\}$ is the resulting graph.

We prove the lemma by applying Lemma 8 twice. The first time, we set $H = G_{i-1}$, $T = T_{i-1}$, $a = x$, $b = z$, $e = e_1$, $e' = e_1'$. We have $w(e') > w(e)$, as needed. $T' = (T \cup \{e_1'\}) \setminus \{e_1\} = T_{i-1}'$ and $H' = G_{i-1}'$. We conclude straightforwardly that

$$index_{T_{i-1}'}(u,v) \;=\; index_{T'}(u,v) \;\geq\; index_T(u,v) \;=\; index_{T_{i-1}}(u,v)$$

and that T_{i-1}' is a maximum spanning tree in G_{i-1}'.

The second time, we let $H = G_{i-1}'$, $T = T_{i-1}'$, $a = y$, $b = z$, and $e = e_2$. We need e_2 to be a minimum weight edge on the $y - z$ path in T_{i-1}', but this holds since the $y - z$ paths in T_{i-1} and T_{i-1}' are the same. We set $e' = e_2'$, with endpoints y, z. We have $w(e_2') > w(e_2)$, as needed. Conveniently, $T' = (T_{i-1}' \cup \{e_2'\}) \setminus \{e_2\} = T_i$ and $H' = G_{i-1}' \cup \{e_2'\} = G_i$. Now $index_{T_i}(u,v) \geq index_{T_{i-1}'}(u,v) \geq index_{T_{i-1}}(u,v)$ and T_i is a maximum spanning tree in G_i. ■

In the sequel, we will use $index_i(u,v)$ to denote $index_{T_i}(u,v)$.

Lemma 10 *Fix any triangle Δ of G. For all $i = 1, \ldots, f$, $g_i(\Delta) \leq g_{i-1}(\Delta)$.*

Proof. We use notation from the definition of the gain of a triangle $\Delta = xyz$. We have $index_{i-1}(x,y) = index_{i-1}(x,z) = w(e_1)$ and $index_{i-1}(y,z) = w(e_2) \geq w(e_1)$. Then

$$\begin{aligned} g_{i-1}(\Delta) &= w(\Delta) - index_{i-1}(x,y) - index_{i-1}(y,z) \\ &= w(\Delta) - \min\{index_{i-1}(x,y), index_{i-1}(x,z), index_{i-1}(y,z)\} \\ &\qquad\quad - \max\{index_{i-1}(x,y), index_{i-1}(x,z), index_{i-1}(y,z)\}. \end{aligned}$$

In other words, to obtain the gain of a triangle we subtract from the weight of the triangle the highest and lowest indices between two vertices of the triangle (the two lowest indices being equal). Since the index function cannot decrease as i increases (Lemma 9), the gain of a triangle cannot increase as i increases. ■

Lemma 11 *For any triangle Δ, there is some $i \in \{0, \ldots, f\}$ such that $g_i(\Delta) \leq 0$.*

Proof. This is certainly true if at the moment Δ was considered during the evaluation phase, $g_i(\Delta) \leq 0$, or in other words, Δ was ignored and not used for an augmentation. So in the following, assume $\Delta = xyz$ was used during the i^{th} augmentation. In other words, $\Delta = \Delta_i$. Therefore $g_{i-1}(\Delta) > 0$; and in this case, we prove that $g_i(\Delta) < 0$. One can easily verify that $N_i = \{e'_1, e'_2\}$, the set of artificial edges added during the i^{th} augmentation, is the only set of two edges of T_i with the property that their deletion pairwise disconnects x, y and z. Then, because in calculating the gain we subtract the weight of two edges whose deletion pairwise disconnects x, y, and z, and because only the pair $\{e'_1, e'_2\}$ has this property, $g_i(\Delta) = w(\Delta) - w(e'_1) - w(e'_2) = w(\Delta) - (w(e_1) + g_{i-1}(\Delta)) - (w(e_2) + g_{i-1}(\Delta)) = -g_{i-1}(\Delta) < 0$, since $g_{i-1}(\Delta) = w(\Delta) - w(e_1) - w(e_2)$. ■

The following lemma is an analogue of Lemma 3.10 of [BR94].

Lemma 12 *No triangle of the original graph G has positive gain with respect to T_f.*

Proof. The previous two lemmas prove that for all Δ, $g_f(\Delta) \leq 0$. ■

Let $P = (V, E_P)$ be any planar subgraph of $G = (V, E)$. Define a weight function w' on the edges of P by $w'(xy) = index_f(x, y)$ for all edges $xy \in E_P$. (Recall that $index_f(x, y)$ is the weight of the lightest edge in the unique $x - y$ path in T_f.)

If we refer to a specific triangle Δ of P, we denote by $a(\Delta) \geq b(\Delta) \geq c(\Delta)$ the weights, with respect to w, of the edges of the triangle. Similarly, $a'(\Delta) \geq b'(\Delta) \geq c'(\Delta)$ are the weights of the edges with respect to w' (where the edge of weight $a(\Delta)$ need not be the one of weight $a'(\Delta)$, etc.).

Lemma 13 *The weight function w' on the edges of P has the following three properties:*

1. *For any edge $e \in E_P$, $w'(e) \geq w(e)$.*
2. *For any triangle Δ of P, $a'(\Delta) + b'(\Delta) \geq a(\Delta) + b(\Delta) + c(\Delta)$. Also $c'(\Delta) = b'(\Delta)$.*
3. *$w'(\Delta) - w(\Delta) = [a'(\Delta) + b'(\Delta) + c'(\Delta)] - [a(\Delta) + b(\Delta) + c(\Delta)] \geq b(\Delta)$.*

Proof. Let $e = xy \in E_P \subseteq E \subseteq E(G_f)$. Since T_f is a maximum spanning tree in G_f, we have $w(e) \leq index_f(x, y) = w'(e)$.

Fix a triangle $\Delta = xyz$ of P. That $c'(\Delta) = b'(\Delta)$ follows immediately from the properties of the index function (see the proof of Lemma 10). Also, we know (by Lemma 12) that Δ has nonpositive gain with respect to T_f. By the method of computing the gain using the index function (again see the proof of Lemma 10), we have $0 \geq g_f(\Delta) = w(\Delta) - \max\{index_f(x, y), index_f(x, z), index_f(y, z)\} - \min\{index_f(x, y), index_f(x, z), index_f(y, z)\} = w(\Delta) - \max\{w'(xy), w'(xz), w'(yz)\} - \min\{w'(xy), w'(xz), w'(yz)\} = w(\Delta) - a'(\Delta) - c'(\Delta) = w(\Delta) - a'(\Delta) - b'(\Delta)$, and it follows that $a'(\Delta) + b'(\Delta) \geq w(\Delta) = a(\Delta) + b(\Delta) + c(\Delta)$.

The third statement is obvious. ■

Lemma 14 *The weight $w(T_f) \geq \frac{1}{3} w'(P)$.*

Proof. Consider the multigraph $H = (V, E_P \cup T_f)$, where we assign weight $w'(e)$ if $e \in E_P$ and $w(e)$ if $e \in T_f$; note that if $e \in E_P \cap T_f$ and e has endpoints x and y, then $w'(e) = index_f(x, y)$ because $e \in E_P$, and $index_f(x, y) = w(e)$, because $e \in T_f$.

We prove that T_f is a maximum spanning tree in multigraph H. Consider any edge e in H with endpoints, say, x and y. If $e \in E_P$, we have $w'(e) = index_f(x, y)$. If $e \in T_f$, we clearly have $w(e) = index_f(x, y)$. So in either case, the weight of an edge with endpoints x, y is at most $index_f(x, y)$; this makes T_f a maximum spanning tree in H.

Since P is a subgraph of H and T_f is a maximum spanning tree of H, the weight of T_f is at least the weight of a maximum spanning tree in P (with weights given by w'). Since Corollary 7 implies that any maximum spanning tree in P has weight at least $\frac{1}{3} w'(P)$, we have $w(T_f) \geq \frac{1}{3} w'(P)$. ■

3.6 Two Lower Bounds

All graphs considered in this section are simple. We will prove two bounds in this section. Both rely on the concept of a "good" set of triangles in the original graph G. The first bound, $w(T_f) \geq \frac{1}{3} w(P) + \frac{1}{12} [\sum_{\Delta \in \mathcal{S}} b(\Delta)]$ for any good set $\mathcal{S}$ of triangles in P (Corollary 16), is easy and follows directly from the definition of a good set of triangles. The second, $w(T_0) \geq \frac{1}{2} w(P) - \frac{1}{2} [\sum_{\Delta \in \mathcal{S}} b(\Delta)]$ (Corollary 20), is more difficult to prove and only holds for a carefully chosen good set $\mathcal{S}$ of triangles in P.

Definition 3 A *straight-line plane graph* is a plane graph whose edges are drawn as straight-line segments.

Definition 4 We call a set $\mathcal{S}$ of triangles of a straight-line plane graph P a *good family* if every edge of P participates in at most two triangles of $\mathcal{S}$ on each side of the line containing the edge.

3.6.1 The (Easy) Lower Bound on $w(T_f)$

We need the following lemma. Recall that P is a planar subgraph of G, and $w'(xy) = index_f(x, y)$ for all edges $xy \in E(P)$.

Lemma 15 *Let $\mathcal{S}$ be any good family of triangles in the straight-line plane graph P. Then $w'(P) \geq w(P) + \frac{1}{4}[\sum_{\Delta\in\mathcal{S}} b(\Delta)]$.*
Proof. Let $m(e)$ be the number of triangles of $\mathcal{S}$ in which e participates. Because $\mathcal{S}$ is good, we have $m(e) \leq 2\cdot 2 = 4$ for all e in G. Using Lemma 13, we have

$$\begin{aligned}\sum_{\Delta\in\mathcal{S}} b(\Delta) &\leq \sum_{\Delta\in\mathcal{S}} (w'(\Delta) - w(\Delta)) = \sum_{\Delta\in\mathcal{S}}\sum_{e\in\Delta} (w'(e) - w(e)) = \sum_{e\in E(P)} (w'(e) - w(e))m(e) \\ &\leq \sum_{e\in E(P)} 4(w'(e) - w(e)) = 4(w'(P) - w(P)). \blacksquare\end{aligned}$$

Corollary 16 *Let $\mathcal{S}$ be any good family of triangles in a straight-line plane graph P which is a subgraph of $G = G_0$. Then*

$$w(T_f) \geq \frac{1}{3}w(P) + \frac{1}{12}\left[\sum_{\Delta\in\mathcal{S}} b(\Delta)\right].$$

Proof. Combine Lemmas 14 and 15. $\blacksquare$

3.6.2 The (Difficult) Lower Bound on $w(T_0)$

We will need the following lemma, whose proof uses an idea from [KH78].
Lemma 17 *Let P be a triangle-free planar graph with a nonnegative weight function c defined on its edges, such that edges xz and yz in P have weight zero. Then there is a spanning forest F in P such that $c(F) \geq \frac{1}{2}c(P)$ and such that x, y, and z are in three different components of F.*
Proof. We use the greedy algorithm to construct F, first sorting the edges of P into nonincreasing order by weight, with edges xz and yz at the end. Let E_i be the set of the first i edges in this ordering, $1 \leq i \leq m$, $m = |E(P)|$. By c_i we denote the weight of the i^{th} edge in this ordering. Starting with $F = \emptyset$, the greedy algorithm scans the edges in the given order and adds an edge to F as long as it satisfies the following conditions:

1. It does not create any cycles.
2. It does not create a path connecting any two of x, y and z.

Let F be the set of edges chosen by the greedy algorithm and let $F_i = E_i \cap F$. Then

$$c(F) = \sum_{i=1}^{m-2} |F_i|(c_i - c_{i+1})$$

$$c(P) = \sum_{i=1}^{m-2} |E_i|(c_i - c_{i+1}).$$

Note that the index only goes to $m-2$. It is therefore enough to show that $|E_i| \leq 2|F_i|$ for $1 \leq i \leq m-2$.

Choose an $i \leq m-2$. Let us assume that the components of F_i have $p_1, p_2, \ldots, p_k$ vertices, where each of x, y and z belongs to one of the first three components. Of course, $|F_i| = \sum_{j=1}^{k}(p_j - 1)$.

We call the set of vertices in the first three components *supercomponent 123*. Note that no edge of E_i can have one endpoint in a component $j > 3$ and the other endpoint in another component $j' > 3$, since some such edge would have been selected by the greedy algorithm, thus merging two components. Similarly, no edge of E_i can have exactly one endpoint in the supercomponent 123. We associate each edge of E_i with the component $j > 3$ or with the supercomponent 123 containing both endpoints of the edge.

The edges of E_i associated with a component $j > 3$ are a subset of the edges of the graph induced in P by the vertices of the component. Thus, for $j > 3$, the number of edges associated with the component is at most $2(p_j - 1)$, because the induced graph is a triangle-free planar graph. (For all $l \geq 2$, the number of edges in a triangle-free planar graph with l vertices is at most $2(l-1)$; the number of edges is at most $2l - 4$ if $l \geq 3$ [B79, Theorem 12, page 18].

The number of edges in E_i associated with supercomponent 123 (which has at least three vertices) is bounded by $[2(p_1 + p_2 + p_3) - 4] - 2$. The "$-2$" appears because the edges xz and yz are not in E_i since $i \leq m-2$. Indeed, the graph induced in P by the vertices of supercomponent 123 is planar, triangle-free, and includes the edges xz and yz. It follows that the number of edges in E_i associated with supercomponent 123 is at most $2(p_1 + p_2 + p_3) - 6 = 2[(p_1 - 1) + (p_2 - 1) + (p_3 - 1)]$. But then $|E_i| \leq \sum_{j=1}^{3} 2(p_j - 1) + \sum_{j=4}^{k} 2(p_j - 1) = 2|F_i|$. $\blacksquare$

We continue with a straight-line plane subgraph P of a weighted graph G. We say a triangle xyz is *inside triangle* xyw (or xyw *contains* xyz) if z is embedded in the bounded plane face defined by xyw. If xyz and xyw are two triangles with z and w on the same side of the line containing xy, $w \neq z$, then one of the triangles has to be inside the other.

We will use the following topological fact, given without proof.

Fact 18 *Let xsz be a triangle containing a triangle xyz (with $s \neq y$). Then there is no triangle xyw (with $w \neq z$) which contains xyz.*

For a triangle Δ, we denote the median weight of the edges of the triangle by $b(\Delta)$, as above.

Definition 5 In a straight-line plane graph, a *separating triangle* is a triangle $\Delta = xyz$ such that some neighbor of x, y, or z is embedded inside Δ, and some neighbor of x, y, or z is embedded outside Δ. (This is slightly different from the standard definition of a separating triangle.)

We now prove the key result needed for the lower bound.

Lemma 19 *Let P be a straight-line weighted plane graph. There is a spanning forest $\overline{T}$ in P and a good family $\mathcal{S}$ such that*

$$w(\overline{T}) + \frac{1}{2}\sum_{\Delta \in \mathcal{S}} b(\Delta) \;\geq\; \frac{1}{2}w(P).$$

Proof. By induction on the number of separating triangles in P.

If P has no separating triangle, we choose $\mathcal{S}$ to be the set of all triangles in P. Now $\mathcal{S}$ is good, for if it were not, choose an edge xy with three triangles xya, xyb, and xyc of $\mathcal{S}$ on the same side of the line containing xy. Relabel a, b, and c, if necessary, so that a is inside $\Delta = xyb$ and c is outside. Then a is a neighbor of x embedded inside Δ and c is a neighbor of x embedded outside Δ, so that Δ is a separating triangle, a contradiction.

We set $\beta = \sum_{\Delta \in S} b(\Delta)$. Removing the median weight edge in each triangle of P leaves us with a triangle-free graph, whose weight is at least $w(P) - \beta$. Let $\overline{T}$ be a maximum spanning forest in the resulting triangle-free graph. Then

$$w(\overline{T}) \;\geq\; \frac{1}{2}(w(P) - \beta),$$

by a result similar to Corollary 7 for triangle-free planar graphs, using the same upper bound on the number of edges in triangle-free simple planar graphs as in the proof of Lemma 17.

For the inductive step, assume that P has a separating triangle. Choose a separating triangle, say xyz, of minimum area; there is no separating triangle inside it. Let P_{in} be the subgraph of P induced by the vertices x, y, z and the vertices inside the triangle xyz. P_{in} has no separating triangle. Each edge of xyz is in at most one triangle in P_{in} other than xyz. Consider the weight function w_{in} on the edges of P_{in} defined as follows: $w_{in}(e) = w(e)$ for all $e \in P_{in}$, except that $w_{in}(xy) = w_{in}(yz) = w_{in}(zx) = 0$. Let P_{out} be the subgraph of P induced by the vertices x, y, z and the vertices outside the triangle xyz. Now

$$w(P) \;=\; w(P_{out}) + w_{in}(P_{in}).$$

Our goal is to find a spanning forest F in P_{in} which does not connect any two of x, y and z, and a spanning forest in P_{out}. Together, they will form the desired spanning forest $\overline{T}$ in P.

Since every separating triangle in P_{out} is also a separating triangle in P, and since the separating triangle xyz of P is not separating in P_{out}, P_{out} has fewer separating triangles than P; by induction, we have a spanning forest T_{out} in P_{out} and a good family $\mathcal{S}_{out}$ of triangles of P_{out} such that

$$w(T_{out}) + \frac{1}{2}\sum_{\Delta \in \mathcal{S}_{out}} b(\Delta) \;\geq\; \frac{1}{2}w(P_{out}).$$

Since P_{in} has no separating triangles, we know, where $\mathcal{S}'_{in}$ is the set of *all* triangles of P_{in}, that $\mathcal{S}'_{in}$ is good for P_{in}. Unfortunately, however, we cannot guarantee that $\mathcal{S}'_{in} \cup \mathcal{S}_{out}$ is a good family for P. We will construct a slightly different good $\mathcal{S}$ for P.

In the following we will assume that, in P_{out}, on the side containing y of the line containing edge xz, there is only one triangle of the form xzw, namely, xzy, and on the side containing x of the line containing edge yz, there is only one triangle of the form yzw, namely, yzx.

We now justify this assumption. If the assumption is not true, then, in P_{out}, either (1) on the side containing y of the line containing edge xz, there are at least two triangles containing edge xz, or (2) on the side containing x of the line containing edge yz, there are at least two triangles containing yz. The two cases are symmetric, for we can interchange x and y. So we will only argue about the former, in which, in P_{out}, on the side containing y of the line containing edge xz, there are at least two triangles. One of them is triangle xyz and the other is triangle xzs

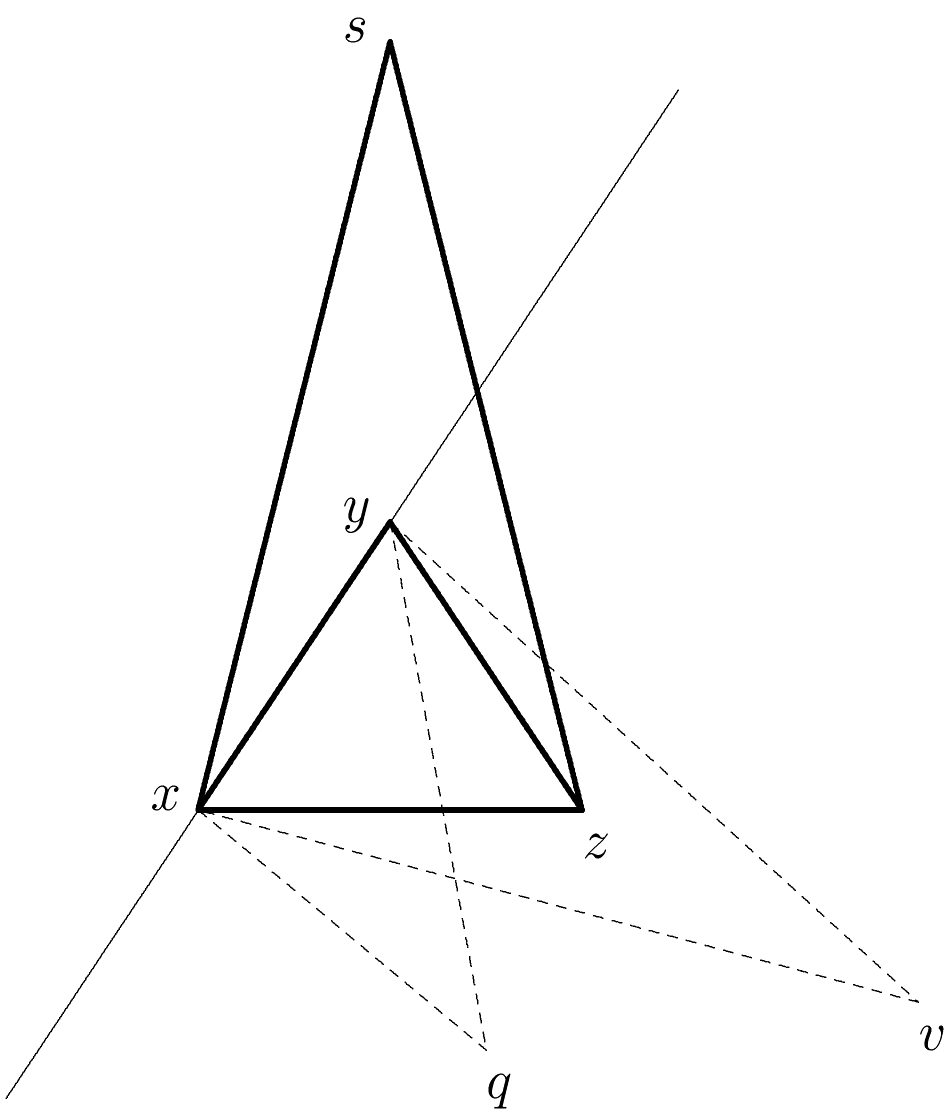

Figure 1: xyz is a triangular face (nothing embedded inside) of the straight-line plane graph P_{out}. If y is in the interior of triangle xzs, z cannot be in the interior of some triangle xyv of P_{out}. Also, a triangle like xyq cannot exist. In other words, on the side containing z of the line containing edge xy, there can be only one triangle xyw for some w, namely, xyz.

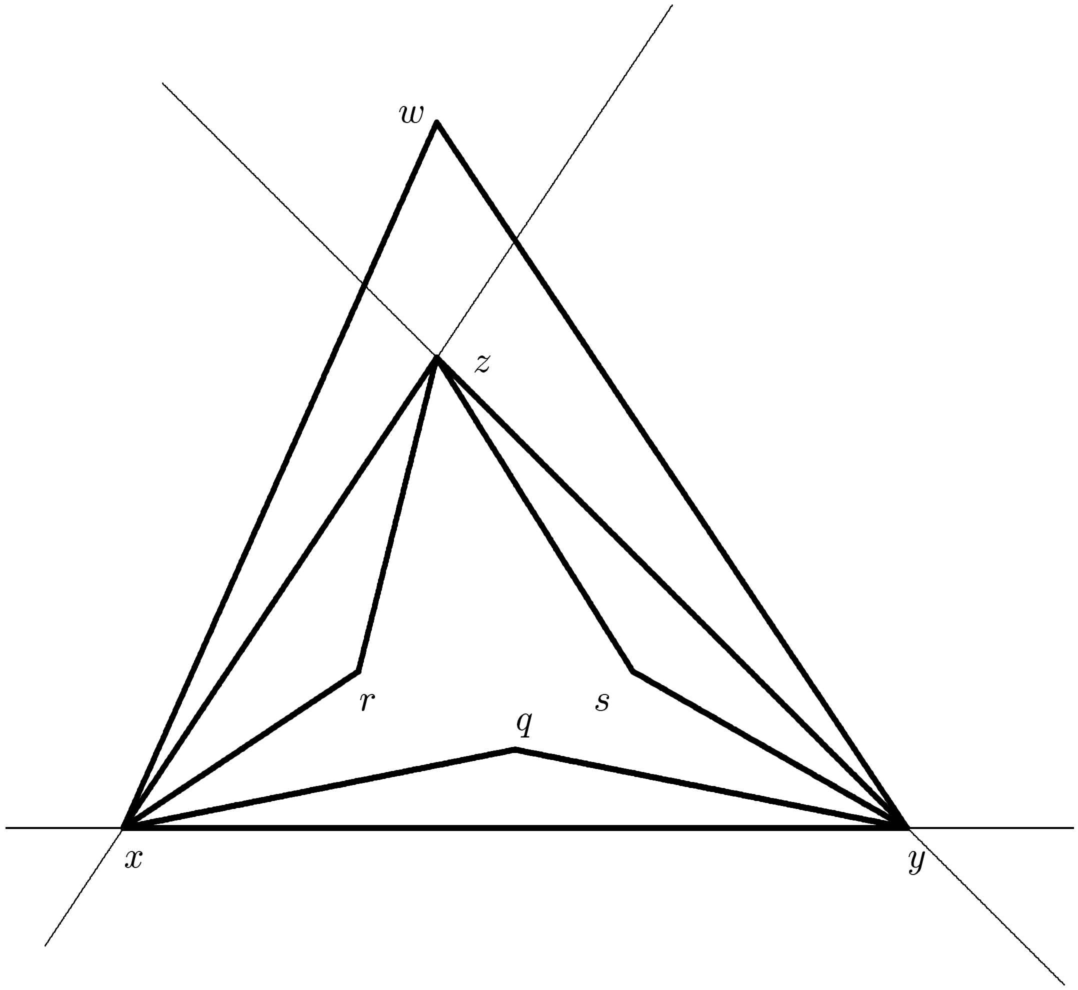

Figure 2: Since the triangles xyz and xyq have been excluded from $\mathcal{S}_{in}$, on the side containing z of the line containing x and y there are at most two triangles in $\mathcal{S}$, those from S_{out} (one of them being xyw in this picture). On the side containing x of the line containing yz, there can be only the triangles syz (from $\mathcal{S}_{in}$) and xyz (which might be in $\mathcal{S}_{out}$). No other triangle of this type can exist in $\mathcal{S}_{out}$ because of the assumption.

(containing y in its interior). For intuition, we refer to Figure 1. But in this case Fact 18 ensures that in P_{out}, on the side containing z of the line containing edge xy, there is only one triangle of the form xyw for some w, namely, the facial triangle xyz of P_{out}. (There is no triangle such as xyv or xyq in Figure 1.) Similarly, on the side containing x of the line containing edge yz, the only triangle wyz for some w is the facial triangle xyz of P_{out}. Since x, y, and z are symmetric, we can satisfy the assumption by interchanging y and z.

If possible, let q, r, and s be vertices, not necessarily distinct, (strictly) inside the triangle xyz, such that xyq, xzr, and yzs are facial triangles. Let $\mathcal{S}_{in}$ be the set of triangles of P_{in}, excluding xyz and xyq (if xyq exists). Clearly $\mathcal{S}_{in}$ is good for P_{in}. We will show that $\mathcal{S} = \mathcal{S}_{in} \cup \mathcal{S}_{out}$ is a good family for P. For intuition, we refer to Figure 2. Indeed, (1) any edge $e \neq xy, yz, xz$ participates in at most two triangles of $\mathcal{S}$ on each side of the line containing e (because $\mathcal{S}_{in}$ and $\mathcal{S}_{out}$ are good and the only edges shared by both are xy, yz and xz). (2) On the side containing z of the line containing xy, in $\mathcal{S}_{in}$ there is no triangle containing x and y. This is true since in P_{in} there is no separating triangle, and the triangles xyz and xyq have been excluded from $\mathcal{S}_{in}$. The other side clearly contains no triangle of $\mathcal{S}_{in}$. In $\mathcal{S}_{out}$, on each of the corresponding sides of the line containing xy, there are at most two triangles containing x and y. (3) On the side containing x of the line containing the edge yz, there is at most one triangle of $\mathcal{S}_{in}$ containing z and y, namely syz, since xyz has been excluded from $\mathcal{S}_{in}$. By assumption, in P_{out}, on the side containing x of the edge yz, there is only one triangle containing z and y, namely xyz. On the side not containing x of the line containing the edge yz, there is no triangle of $\mathcal{S}_{in}$ and there are at most two of $\mathcal{S}_{out}$. (4) For the edge xz, the reasoning is similar to that for yz. This completes the proof that $\mathcal{S} = \mathcal{S}_{in} \cup \mathcal{S}_{out}$ is good.

Let $\beta_{in} = \sum_{\Delta \in \mathcal{S}_{in}} b_{in}(\Delta)$, where b_{in}, the value of the median edge, is computed according to w_{in}. Then, using the fact that $w_{in} \leq w$,

$$\sum_{\Delta \in \mathcal{S}_{in}} b(\Delta) \;\geq\; \beta_{in}.$$

If we remove the median edge (according to w_{in}) from each triangle in P_{in}, never choosing xz or yz as a median-weight edge (so we remove xy from P_{in}), we get a triangle-free graph P'_{in} still having edges xz and yz of weight $w_{in} = 0$. By Lemma 17, there is a spanning forest F_{in} in P'_{in} not connecting any two of x, y, and z and having weight

$$w(F_{in}) \;=\; w_{in}(F_{in}) \;\geq\; \frac{1}{2} w_{in}(P'_{in}) \;\geq\; \frac{1}{2}(w_{in}(P_{in}) - \beta_{in}).$$

Then

$$w(F_{in}) + \frac{1}{2} \sum_{\Delta \in \mathcal{S}_{in}} b(\Delta) \;\geq\; \frac{1}{2} w_{in}(P_{in}).$$

Putting together T_{out} and F_{in} to get a forest $\overline{T}$ in P, we conclude with

$$\begin{aligned}
w(\overline{T}) + \frac{1}{2}\sum_{\Delta \in \mathcal{S}} b(\Delta) &= [w(F_{in}) + w(T_{out})] + \frac{1}{2}\sum_{\Delta \in \mathcal{S}_{in}} b(\Delta) + \frac{1}{2}\sum_{\Delta \in \mathcal{S}_{out}} b(\Delta) \\
&\geq \left[w(F_{in}) + \frac{1}{2}\sum_{\Delta \in \mathcal{S}_{in}} b(\Delta)\right] + \left[w(T_{out}) + \frac{1}{2}\sum_{\Delta \in \mathcal{S}_{out}} b(\Delta)\right] \\
&\geq \frac{1}{2}(w_{in}(P_{in}) + w(P_{out})) \\
&= \frac{1}{2} w(P). \blacksquare
\end{aligned}$$

Corollary 20 *There exists a good family $\mathcal{S}$ in P such that*

$$w(T_0) \;\geq\; \frac{1}{2} w(P) - \frac{1}{2}\left[\sum_{\Delta \in \mathcal{S}} b(\Delta)\right].$$

Proof. Choose $\mathcal{S}, \overline{T}$ as in Lemma 19. Since $w(T_0) \geq w(\overline{T})$, we have $w(T_0) \geq w(\overline{T}) \geq \frac{1}{2}w(P) - \frac{1}{2}[\sum_{\Delta \in \mathcal{S}} b(\Delta)]$. $\blacksquare$

3.7 A Bound on the Performance Ratio in the General Case

Theorem 21 *The performance ratio of the algorithm is at least $\frac{1}{3} + \frac{1}{72}$.*

Proof. Let P be a straight-line embedding of a maximum weight planar subgraph of G and F a maximum spanning forest in P. We know from Corollaries 2 and 4 that the algorithm produces a triangular structure S_0 of weight

$$w(S_0) \;\geq\; \frac{1}{2}\left[w(T_0) + w(T_f)\right].$$

Now $w(T_0) \geq w(F) \geq \frac{1}{3}w(P)$ (Corollary 7). By Corollary 20, there is a good family $\mathcal{S}$ in P such that $w(T_0) \geq \frac{1}{2}w(P) - \frac{1}{2}\beta$, where $\beta = \sum_{\Delta \in \mathcal{S}} b(\Delta)$. Therefore

$$w(T_0) \;\geq\; \max\left\{\frac{1}{3}w(P), \frac{1}{2}w(P) - \frac{1}{2}\beta\right\}.$$

(It is important that we are not using only the trivial first bound here.) Corollary 16 implies the bound

$$w(T_f) \;\geq\; \frac{1}{3}w(P) + \frac{1}{12}\beta$$

for the same $\mathcal{S}$. We therefore have

$$w(S_0) \;\geq\; \frac{1}{2}\left[\max\left\{\frac{1}{3}w(P), \frac{1}{2}w(P) - \frac{1}{2}\beta\right\} + \left(\frac{1}{3}w(P) + \frac{1}{12}\beta\right)\right].$$

A simple balancing argument shows that this lower bound is minimized when $\beta = \frac{1}{3}w(P)$, so that we obtain

$$w(S_0) \;\geq\; \left(\frac{1}{3} + \frac{1}{72}\right) w(P). \blacksquare$$

3.8 An Upper Bound on the Performance Ratio of the Algorithm

On the other hand, the performance ratio of the algorithm is at most $5/12$. In fact, we will exhibit *planar* graphs for which the weight of a maximum weight triangular structure is at most $5/12 + o(1)$ of the total weight. Running the algorithm on any of these planar graphs will result in a triangular structure and therefore the weight of the planar subgraph found by our algorithm is at most $5/12 + o(1)$ of the optimal weight. These planar graphs are obtained from arbitrary triangulated plane graphs through a construction we describe below.

Let P be any triangulated (simple) plane graph with $n \geq 3$ vertices. By Euler's formula, P has $3n - 6$ edges; let the weight of each edge of P be two. In each of the $2n - 4$ faces of P add a new vertex, and add three edges adjacent to this new vertex and the three vertices of P which define the face. These new edges have weight one. We denote by G the plane graph obtained this way. G has $3n - 4$ vertices, is triangulated, and has total weight $2 \cdot [3n - 6] + 1 \cdot [3(2n - 4)] = 12n - 24$. (See Figure 3.) We will prove that a maximum weight triangular structure in G has weight at most $5/12 + o(1)$ of the weight of G. To do so, we need a lemma (which will also be used later).

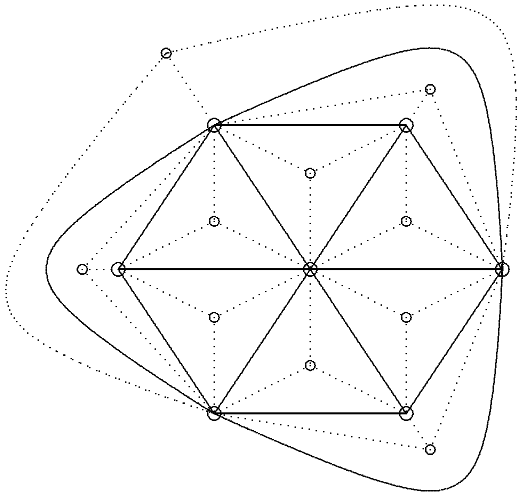

Figure 3: An example of graph G. The solid edges show graph P. Each of them has weight two. The dotted part shows the extra vertices and edges in G. The dotted edges have weight one.

Lemma 22 *Let $G = (V, E)$ be any graph and T be a set (possibly intersecting E) of edges on V. Let w be a nonnegative weight function on $E \cup T$. Assume that T is a maximum spanning tree in the multigraph $G \cup T = (V, E \cup T)$. For every triangle $\Delta = xyz$ in G, let*

$$\begin{aligned} g(\Delta) &= w(\Delta) - \max\{index_T(x,y), index_T(x,z), index_T(y,z)\} \\ &\quad - \min\{index_T(x,y), index_T(x,z), index_T(y,z)\}. \end{aligned}$$

If $g(\Delta) \leq 0$ for every triangle Δ in G, then $w(T) \geq w(A)$ for every triangular structure A in G.

Proof. Let A be an arbitrary triangular structure in G. Let us iteratively define edge set E_i, graph $G_i = (V, E_i)$, and triangular structure A_i in G_i. To start, let $E_0 = E$, $G_0 = G$, $A_0 = A$, and $i = 0$.

Let Δ be a triangle in A_i. Label the vertices of Δ as x, y, z so that $index_T(x, y) \leq index_T(x, z) \leq index_T(y, z)$. We modify A_i by removing the edges of Δ and adding two new edges e_1 and e_2 to get A_{i+1}. The endpoints of e_1 are x and y and $w(e_1) = index_T(x, y)$. The endpoints of e_2 are y and z and $w(e_2) = index_T(y, z)$. We set $E_{i+1} = E_i \cup \{e_1, e_2\}$, and $G_{i+1} = (V, E_{i+1})$.

Since $g(\Delta) \leq 0, w(\Delta) \leq index_T(x, y) + index_T(y, z) = w(e_1) + w(e_2)$. So $w(A_{i+1}) \geq w(A_i)$. Also, A_{i+1} is a triangular structure in G_{i+1} all of whose triangles are in G, having fewer triangles than A_i.

We iterate this process for all q triangles of A. At the end we have a triangular structure A_q in a graph G_q, with $w(A_q) \geq w(A)$. A_q has no triangles so it is a forest in G_q. We know that T is a maximum spanning tree in $G \cup T$. Each additional edge (with endpoints, say, u and v) we add has weight at most (in fact, exactly) $index_T(u, v)$. This guarantees that the one tree T is a maximum spanning tree in each $G_i \cup T$ and in particular in $G_q \cup T$. Since A_q is a forest in G_q, $w(T) \geq w(A_q)$. So we conclude that $w(T) \geq w(A_q) \geq w(A)$. ■

Now we are ready to prove that a maximum weight triangular structure in G has weight at most $5/12 + o(1)$ of the weight of G. Let A be a set of new weight-3 edges forming any spanning tree on $V(P)$, the vertex set of P.

Let B be a subset of the edges of G: for each of the $2n - 4$ vertices inserted in the faces of P, choose exactly one edge of G incident to it. Then we know that $T = (V(G), A \cup B)$ is a spanning tree of the graph $G \cup T$.

In fact, it is easy to verify that T is a maximum spanning tree in $G \cup T$. Consider an edge e of G. If e has weight one, the path in T between e's endpoints obviously cannot contain an edge of weight smaller than one. If e has weight two, then e is an edge of P. Let u and v be the endpoints of e; then $u, v \in V(P)$. The set of edges A is a spanning tree on $V(P)$. As there is a path in A from u to v, the unique path in T from u to v is contained in A and therefore the weight of any edge on the unique path in T from u to v is three.

Also, no triangle of G has positive gain with respect to T, for the following reason. Consider a triangle Δ in G. Label the vertices of Δ as x, y, z so that $index_T(x,y) \leq index_T(x,z) \leq index_T(y,z)$. If all the edges in Δ have weight two, then Δ is a triangle in P and the path in T between any pair of its vertices contains only edges of weight three. Thus we have $w(\Delta) = 6$, $index_T(x,y) = 3 = index_T(y,z)$. This makes $g(\Delta) = w(\Delta) - index_T(x,y) - index_T(y,z) = 0$. If Δ contains at least one edge of weight one, then it contains exactly two edges of weight one and one edge of weight two. This means $w(\Delta) = 4$. Since $index_T(x,y) \leq index_T(x,z) \leq index_T(y,z)$, we have $index_T(x,y) = 1$ and $index_T(y,z) = 3$. Thus again $g(\Delta) = w(\Delta) - index_T(x,y) - index_T(y,z) = 0$.

Therefore, we can apply Lemma 22 and conclude that $w(T) \geq w(S)$ for any triangular structure S in G. In conclusion, if S is a triangular structure in G,

$$w(S) \;\leq\; w(T) \;=\; w(A) + w(B) \;=\; 3(n-1) + 1(2n-4) \;=\; 5n - 7 \;=\; \left(\frac{5}{12} + \frac{1}{4n-8}\right) w(G).$$

In fact, we conjecture that $w(\mathcal{S}_{opt}) \geq (5/12) w(P)$, where S_{opt} is a heaviest triangular structure in planar graph P, for any weight function.

3.9 The Triangle Inequality

The next theorem gives a better performance ratio for the algorithm if the input graph is a weighted clique whose edge weights satisfy the triangle inequality.

Theorem 23 *Suppose that the edge weights in a weighted clique G satisfy the triangle inequality. Then $w(S_0) \geq \frac{3}{8} w(P)$, where P is a heaviest planar subgraph of G.*

Proof. Let P be a straight-line embedding of a heaviest planar subgraph of G. We assume, without loss of generality, that P is triangulated. By the triangle inequality, we have $b(\Delta) \geq \frac{1}{2} a(\Delta)$ for any (facial) triangle Δ of P. Recall (from Section 3.5, page 8) that $w'(xy) = index_f(x,y)$ for all edges $xy \in E(P)$. Using Lemma 13, it follows that for a triangle Δ of P we have

$$w'(\Delta) \;=\; a'(\Delta) + b'(\Delta) + c'(\Delta) \;\geq\; a(\Delta) + b(\Delta) + c(\Delta) + b(\Delta).$$

Using $b(\Delta) \geq c(\Delta)$ and $a(\Delta) \leq b(\Delta) + c(\Delta)$, we can deduce that

$$b(\Delta) \;\geq\; \frac{1}{3}\left[a(\Delta) + c(\Delta)\right]$$

and hence

$$w'(\Delta) \;\geq\; \frac{5}{4}[a(\Delta) + b(\Delta) + c(\Delta)].$$

Therefore

$$w'(\Delta) \;\geq\; \frac{5}{4} w(\Delta).$$

Then, because P is triangulated,

$$w'(P) \;=\; \frac{1}{2} \sum_{\text{facial triangles } \Delta \text{ of } P} w'(\Delta) \;\geq\; \frac{1}{2} \sum_{\text{facial triangles } \Delta \text{ of } P} \frac{5}{4} w(\Delta) \;=\; \frac{5}{4} w(P).$$

This bound and $w(T_f) \geq \frac{1}{3} w'(P)$ (Lemma 14) imply that $w(T_f) \geq \frac{5}{12} w(P)$ and hence, using Corollaries 4 and 7,

$$w(S_0) \;\geq\; \frac{1}{2}\left(w(T_0) + w(T_f)\right) \;\geq\; \frac{1}{2}\left(\frac{1}{3} + \frac{5}{12}\right) w(P) \;=\; \frac{3}{8} w(P). \blacksquare$$

3.10 Outerplanar Subgraphs

Consider the MAXIMUM WEIGHT OUTERPLANAR SUBGRAPH problem: given an edge-weighted graph G, find an outerplanar subgraph of G of maximum weight. This problem is known to be NP-hard [GJ79, p. 197].

We know of no published research on the problem. The naive approximation ratio is 1/2, which is obtained by any algorithm that produces a maximum spanning tree T_0 of G (assuming G is connected). To prove this, we need the following lemma.

Lemma 24 *Let P be an outerplanar graph with nonnegative weight function w on the edges. Then a maximum spanning forest F of P satisfies $w(F) \geq \frac{1}{2}w(P)$.*
Proof. Any subgraph of an outerplanar graph is outerplanar, and, by Euler's formula, $|E(H)| \leq 2(|V(H)| - 1)$ for any outerplanar graph H [Ha72, Cor. 11.9]. Thus, it is enough to apply Lemma 6 with $c = 2$ to the family of all outerplanar graphs. ■

Let P be any outerplanar subgraph of G. By Lemma 24, we deduce that $w(T_0) \geq \frac{1}{2}w(P)$, and the performance ratio of 1/2 follows.

A triangular structure is an outerplanar graph. In the unweighted case, a maximum triangular structure in a graph G (one with the maximum number of edges) has at least two thirds as many edges as a maximum outerplanar subgraph of G [CFFK98]. We will show that a maximum *weight* triangular structure in a weighted graph G has weight at least two thirds of the weight of a maximum weight outerplanar subgraph of G. Unfortunately, we do not know how, in polynomial time, to find a maximum weight triangular structure in a given weighted graph. The algorithm presented in this chapter produces a heavy triangular structure in a given weighted graph. We will prove that the algorithm improves the best known performance ratio for MAXIMUM WEIGHT OUTERPLANAR SUBGRAPH to $7/12 > 0.583$.

The proof of the next key lemma takes a few pages.

Lemma 25 *In any maximal outerplanar graph P, there are at most three (not necessarily distinct) triangular structures in P such that each edge of P appears in exactly two of them.*
Proof. If P has fewer than three vertices, then P and the empty graph are triangular structures. Let us assume P has at least three vertices. Embed P in the plane as a triangulation of a polygon. (Every maximal outerplanar graph with at least three vertices is a triangulation of a polygon. That is, the boundary of the exterior face of P is a Hamiltonian cycle H and each interior face is triangular [Ha72, p. 106].) Let E be the edge set of P, b be the exterior face of P, and F be the set of faces of P other than b.

Let D be the dual multigraph of P. Let us call the vertices of D (which are faces of P) *nodes*, and the edges of D, *arcs*. All nodes of D but b have degree three. Also, the edges in the Hamiltonian cycle H correspond to the arcs incident to b in D.

Let D' be the graph obtained from D by subdividing each arc incident to b, and then removing b. See Figure 4 for an example.

Lemma 26 *D' is a tree all of whose internal nodes have degree three.*
Proof. First, let us prove that D' has no cycle. It is enough to show that any cycle in D contains b. A cycle in D corresponds to a cut in P [BM76, p. 143, ex. 9.2.3]. Because H is a Hamiltonian cycle, any cut in P contains at least two edges of H, which correspond to arcs incident to b. Therefore, any cycle in D contains at least two arcs incident to b, so it contains b.

Second, let us prove that D' is connected. If D' were not connected, there would be two nodes u and v in different components of D'. Let us argue that we can choose u and v to be nodes in $V(D)$. If u were not a node in $V(D)$, then it would be a node which originated from the subdivision of an arc incident to b, and thus it would have degree one in D'. Change u to its unique neighbor in D'. Do the same for v. Note that the new u and v must still be in different components of D', since they are in the same component as the initial u and v, respectively. So we can assume u and v are nodes of D. And because D is connected, there is a path in D between u and v. For this path not to exist in D', it has to go through node b. But this implies that b would be a cut vertex in D. If b were a cut vertex in D, then there would be a *minimal* cut in D containing exactly a proper subset of the arcs incident to b (consider the set of edges going from b to one of the components of D after the removal of b). A minimal cut in D corresponds to a cycle in P [BM76, p. 143, ex. 9.2.3]. This implies that there would be a cycle in P whose edges are a proper subset of the edges of H, a contradiction (a proper subset of the edges of any cycle induces an acyclic graph, since it is enough to remove one edge of a cycle to be left with a path, which is acyclic).

Therefore D' is, in fact, a tree. Recall that all nodes of D but b have degree three. Before removing b, we subdivided all of the arcs incident to b. The new nodes have degree one in D', and all others have the same degree as in D, i.e., three. ■

A *special 3-coloring of D'* is a partition of the set of nodes of D' into three sets (X_1, X_2, X_3) (each set referred to as a *color class*) such that

1. adjacent nodes have different colors, and
2. for $i = 1, 2, 3$, if we remove all nodes of color i, in the resulting graph there is a path from any node to a leaf in D'.

Lemma 27 *There is a special 3-coloring of D'.*

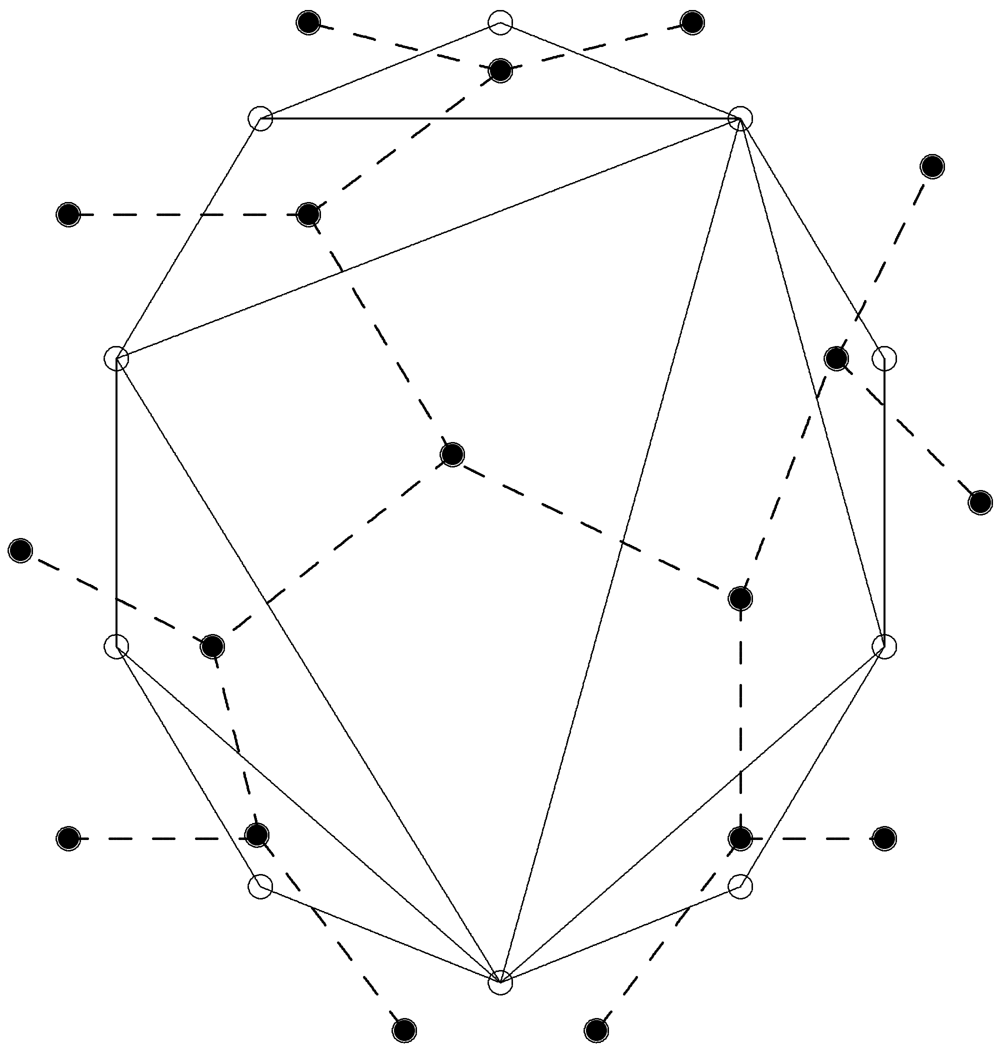

Figure 4: An outerplanar graph (solid lines) and the tree D' (filled vertices and dotted lines) obtained from its dual.

Proof. Root D' at one of its leaves. In the rooted D', all internal nodes except the root have two children. Color the root and its unique child with distinct colors. Now, start at level $i = 1$. If there are nodes in level $i+1$, for each node in level i with children, give its children distinct colors which differ from its own color. Proceed to level $i+1$.

Clearly, adjacent nodes get distinct colors.

Suppose we remove all nodes of color i, for some i; let j and k be the two remaining colors. Consider a remaining node. Either it is a leaf in D', and there is nothing to prove, or it is an internal node of the rooted tree D' different from the root (since the root is a leaf of D'). Any internal node colored j which is not the root has a child of color k, and *vice versa*. This means that there is a path from any node to a leaf in D'. ■

Given a special 3-coloring (X_1, X_2, X_3) of D', we now describe three triangular structures S_1, S_2 and S_3 as required by Lemma 25. Establish the natural one-to-one correspondence between edges of P and arcs of D': each edge of P corresponds to the unique arc of D' which "crosses" it.

Let S_i be the set of edges in P whose corresponding arc in D' has an endpoint of color i (i.e., in X_i). (An arc has either one or zero endpoints of a given color.)

Lemma 28 *S_i is a triangular structure in P.*

Proof. Suppose that there is a cycle C in S_i which is not a triangle. C partitions the set of faces of P into two sets $F_0 \ni b$ (outside C) and $F_1 \subseteq F$ (inside C), F being the set of faces of D other than b.

Because C is not a triangle, $|F_1| \geq 2$. (If $|F_1| = 1$, then C would be the boundary of the unique face in $F_1 \subseteq F$, which is a triangle.)

A cycle in P corresponds to a minimal cut in D [BM76, p. 143, ex. 9.2.3]. So F_1 must induce a connected subgraph of D. Since D' differs from D only in arcs incident to b, F_1 induces the same connected subgraph in D'. Also, F_1 consists only of internal nodes of D'.

Thus F_1 induces a connected subgraph of D' with at least two nodes. Hence, in the given special 3-coloring of D', not all nodes of F_1 can be of color i; there is a node $d \in F_1$ of color $j \neq i$.

All leaves of D' are outside C. Since C is in S_i, each edge of C corresponds to an arc in D' with one endpoint of color i. Removing the nodes of D' of color i hence eliminates all arcs with one endpoint inside C and one outside. Hence, after removing from D' the nodes (faces of P) of color i, there can be no path from node d to a leaf of D'. This is a contradiction to the fact that we have a special 3-coloring. ■

Clearly, S_1, S_2 and S_3 satisfy the statement of Lemma 25: each edge of P appears in exactly two of them. ■

Theorem 29 *Let G be a graph with a nonnegative weight function w on the edges and let S_{opt} be a maximum weight triangular structure in G. Then $w(S_{opt}) \geq \frac{2}{3}w(P)$ for any outerplanar subgraph P of G.*

Proof. By adding edges possibly not in G, extend P to a maximal outerplanar graph $\overline{P}$. For any edge e in $\overline{P}$ but not in G, let $w(e) = 0$. Clearly $w(\overline{P}) \geq w(P)$.

Let S_1, S_2, S_3 be three triangular structures in $\overline{P}$ as given by Lemma 25. Each edge of $\overline{P}$ appears in exactly two of these triangular structures.

Then $w(S_1) + w(S_2) + w(S_3) = 2w(\overline{P}) \geq 2w(P)$. Moreover, $w(S_{opt}) \geq w(S_i)$, $i = 1, 2, 3$. Therefore $2w(P) \leq 3w(S_{opt})$, implying that $w(S_{opt}) \geq \frac{2}{3}w(P)$. ■

Corollary 30 *The algorithm described in Section 2 has performance ratio at least $7/12$ for* MAXIMUM WEIGHT OUTERPLANAR SUBGRAPH.

Proof. Lemma 9 tells us that T_f is a maximum spanning tree in G_f; because G is a subgraph of G_f, T_f is a maximum spanning tree in $G \cup T_f$. By Lemma 12 we know that every triangle of G has nonpositive gain with respect to T_f. Thus, by Lemma 22, we infer that $w(T_f) \geq w(S_{opt})$, where S_{opt} is a maximum weight triangular structure in G.

Together with Corollary 4, this means that the algorithm produces a triangular structure S_0 such that

$$w(S_0) \;\geq\; \frac{w(T_0) + w(S_{opt})}{2},$$

where T_0 is a maximum spanning tree of G. So far, everything we have said applies to MAXIMUM WEIGHT PLANAR SUBGRAPH.

Let P be any maximum weight outerplanar subgraph of G. We know, by Lemma 24, that $w(T_0) \geq \frac{1}{2}w(P)$. From Theorem 29, we have $w(S_{opt}) \geq \frac{2}{3}w(P)$. Hence

$$w(S_0) \;\geq\; \frac{w(T_0) + w(S_{opt})}{2} \;\geq\; \frac{\frac{1}{2}w(P) + \frac{2}{3}w(P)}{2} \;=\; \frac{7}{12}w(P). \blacksquare$$

4 Acknowledgments

We thank Leonard Schulman and Robin Thomas for helpful discussions and suggestions. We are also very grateful to the referees for their very thorough readings and for their comments.

References

[AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows*, Prentice-Hall, Upper Saddle River, NJ, 1993.

[B79] B. Bollobas, *Graph Theory–An Introductory Course*, Springer-Verlag, New York, 1979.

[BETT94] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis, "Algorithms for Drawing Graphs: an Annotated Bibliography," *Computational Geometry: Theory and Applications* 4 (5), 1994, 235–282.

[BFP95] R.P.G. Barlow, G.N. Fisher, and S.N. Papantonopoulos, "The Application of a Knowledge-Based Engineering System to the Planning of Airport Facilities—a Case Study," in *IEE Colloquium on Knowledge-Based Approaches to Automation in Construction*, 1995, 1–4.

[BM76] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, MacMillan Press, 1976.

[BNTT84] C. Batini, E. Nardelli, M. Talamo, and R. Tamassia, "A Graph-Theoretic Approach to Aesthetic Layout of Information Systems Diagrams," in *Proc.* 10^{th} *International Workshop on Graph-Theoretic Concepts in Computer Science*, 1984, 9–18.

[BR94] P. Berman and V. Ramaiyer, "Improved Approximations for the Steiner Tree Problem," *Journal of Algorithms* 17 (1994), 381–408.

[CFFK98] G. Călinescu, C.G. Fernandes, U. Finkler, and H. Karloff, "A Better Approximation Algorithm for Finding Planar Subgraphs," *Journal of Algorithms* 27 (1998), 269–302.

[Ci97] R. Cimikowski, "An Analysis of Heuristics for Graph Planarization," *Journal of Information & Optimization Sciences* 18 (1), 1997, 49–73.

[CMWZZ94] B. Chen, M. Matsumoto, J. Wang, Z. Zhang, and J. Zhang, "A Short Proof of Nash-Williams' Theorem for the Arboricity of a Graph," *Graphs and Combinatorics* 10 (1994), 27–28.

[CN85] N. Chiba and T. Nishizeki, "Arboricity and Subgraph Listing Algorithms," *SIAM Journal on Computing* 14 (1985), 210–223.

[CNS79] N. Chiba, T. Nishizeki, and I. Shirakawa, "An Algorithm for Maximal Planarization of Graphs," in *Proc. IEEE International Symposium on Circuits and Systems*, 1979, 649–652.

[CON85] N. Chiba, K. Onogushi, and T. Nishizeki, "Drawing Planar Graphs Nicely," *Acta Informatica* 22 (1985), 187–201.

[CST95] P. Crescenzi, R. Silvestri, and L. Trevisan, "To Weight or Not to Weight: Where is the Question?", in *Proc.* 4^{th} *Israel Symposium on Theory of Computing and Systems*, 1996, 68–77.

[CYN85] N. Chiba, T. Yamanouchi, and T. Nishizeki, "Linear Algorithms for Convex Drawings of Planar Graphs," *Progress in Graph Theory*, J. A. Bondy and U. S. R. Murty (eds.), Academic Press, 1984, 153–173.

[DFF85] M.E. Dyer, L.R. Foulds, and A.M. Frieze, "Analysis of Heuristics for Finding a Maximum Weight Planar Subgraph," *European Journal of Operations Research* 20 (1985), 102–114.

[DH72] J.W. Dickey and J. Hopkins, "Campus Building Arrangements using TOPAZ," *Transportation Research* 6 (1972), 59–68.

[E77] A.N. Elshafei, "Hospital Layout as a Quadratic Assignment Problem," *Operational Research Quarterly* 28 (1977), 167–179.

[EFG82] P. Eades, L. Foulds, and J. Giffin, "An Efficient Heuristic for Identifying a Maximal Weight Planar Subgraph," *Combinatorial Mathematics IX*, *Lectures Notes in Mathematics* 952, 1982, 239–251.

[F92] L.R. Foulds, *Graph Theory Applications*, Springer-Verlag, New York, 1992.

[FG95] U. Feige and M.X. Goemans, "Approximating the Value of Two-Prover Proof Systems, with Applications to MAX 2SAT and MAX DICUT," in *Proc.* 3^{rd} *Israel Symposium on the Theory of Computing and Systems*, 1995, 182–189.

[FGG85] L.R. Foulds, P.B. Gibbons, and W.J. Giffin, "Graph-Theoretic Heuristics for the Facilities Layout Problem: an Experimental Comparison," *Operations Research* 33 (1985), 1091–1106.

[FR78] L.R. Foulds and D.F. Robinson, "Graph-Theoretic Heuristics for the Plant Layout Problem," *International Journal of Production Research* 12 (1978), 27–37.

[FT86] L.R. Foulds and H.V. Tran, "Library Layout via Graph Theory," *Computers and Industrial Engineering* 10 (3), 1986, 245–252.

[GJ79] M.R. Garey and D.S. Johnson, *Computers and Intractability*, W. H. Freeman and Co., New York, 1979.

[GS85] H.N. Gabow and M. Stallmann, "Efficient Algorithms for Graphic Matroid Intersection and Parity," *Automata, Language and Programming:* 12^{th} *Colloq., Lecture Notes in Computer Science* 194 (1985), 210–220.

[GVY94] N. Garg, V.V. Vazirani, and M. Yannakakis, "Multiway Cuts in Directed and Node Weighted Graphs," manuscript. Preliminary version appeared in *Proc.* 21^{st} *ICALP*, *Lecture Notes in Computer Science* 820, Springer-Verlag, 1994, 487–498.

[GW95] M.X. Goemans and D.P. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems using Semidefinite Programming," *Journal of the ACM* 42:6 (1995), 1115–1145.

[Ha72] F. Harary, *Graph Theory*, Addison-Wesley Publishing Company, 1972.

[Ho82] D. Hochbaum, "Approximation Algorithms for the Set Covering and Vertex Covering Problems," *SIAM Journal of Computing* 11 (1982), 555–556.

[Ho97] D. Hochbaum, *Approximation Algorithms for NP-Hard Problems*, PWS Publications, 1997.

[HW85] A. Hammouche and D.B. Webster, "Evaluation of an Application of Graph Theory to the Layout Problem," *International Journal of Production Research* 23 (5), 1985, 987–1000.

[JM93] M. Jünger and P. Mutzel, "Solving the Maximum Weight Planar Subgraph Problem," in *Proc. 3rd Integer Programming and Combinatorial Optimization Conf.*, 1993, 479–492.

[JM96] M. Jünger and P. Mutzel, "Maximum Planar Subgraphs and Nice Embeddings: Practical Layout Tools," *Algorithmica* 16 (1), 1996, 33–59 (special issue on graph drawing).

[JTS86] R. Jayakumar, K. Thulasiraman, and M.N.S. Swamy, "On Maximal Planarization of Nonplanar Graphs," *IEEE Trans. Circuits and Systems*, CAS-33, 1986, 843–854.

[KH78] B. Korte and D. Hausmann, "An Analysis of the Greedy Heuristic for Independence Systems," *Annals of Discrete Mathematics* 2 (1978), 65–74.

[Kr69] K. Krejcirik, "Computer Aided Plant Layout," *Computer Aided Design* 2 (1969), 7–17.

[LG77] P.C. Liu and R.C. Geldmacher, "On the Deletion of Nonplanar Edges of a Graph," in *Proc.* 10^{th} *Southeastern Conference on Combinatorics, Graph Theory, and Computing*, 1977, 727–738.

[Li01] A. Liebers, "Planarizing Graphs—A Survey and Annotated Bibliography," *J. Graph Algorithms and Applications* 5 (1), 2001, 1–74.

[LNS85] R. Lipton, S. North, and J. Sandberg, "A Method for Drawing Graphs," in *Proc. ACM Symposium on Computational Geometry*, 1985, 153–160.

[LP86] L. Lovász and M.D. Plummer, *Matching Theory*, Elsevier Science, Amsterdam, 1986.

[Ma78] M. Marek-Sadowska, "Planarization Algorithms for Integrated Circuits Engineering," in *Proc. IEEE International Symposium on Circuits and Systems*, 1978, 919–923.

[Mu94] P. Mutzel, "The Maximum Planar Subgraph Problem," PhD Thesis–Universität zu Köln, 1994.

[N64] C.St.J.A. Nash-Williams, "Decomposition of Finite Graphs into Forests," *Journal of London Mathematical Society* 39 (1964), 12.

[OT81] T. Ozawa and H. Takahashi, "A Graph-Planarization Algorithm and its Applications to Random Graphs," *Graph Theory and Algorithms*, *Lectures Notes in Computer Science* 108, Springer-Verlag, 1981, 95–107.

[S95] D.B. Shmoys, "Computing Near-Optimal Solutions to Combinatorial Optimization Problems," *Combinatorial Optimization, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 20, W. Cook and L. Lovász (eds.), AMS Publications (1995), 355–397.

[TDB88] R. Tamassia, G. Di Battista, and C. Batini, "Automatic Graph Drawing and Readability of Diagrams," *IEEE Transactions on Systems, Man and Cybernetics* 18 (1988), 61–79.

[Tu63] W.T. Tutte, "How to Draw a Graph," *Proc. London Math. Society* 13 (1963), 743–768.