Chapter 16

Approximation schemes for Steiner problems

The *Steiner tree problem in networks* is as follows:

- *input:* an undirected graph G with edge-lengths, a subset of vertices called the *terminals*
- *output:* a minimum-weight connected subgraph of G that spans the terminals

We can consider two approaches for finding an approximation scheme. Using the framework described in Section 15.5 yields a linear-time approximation scheme but the "constant" is doubly exponential in a polynomial in ϵ^{-1}. All we have to do to obtain this approximation scheme is to provide an implementation of the *Spanner step*. A more sophisticated approach yields another linear-time approximation scheme, one in which the constant is *only* singly exponential in a polynomial in ϵ^{-1}.

Both approaches use a structure called a *brick decomposition* of the input graph. The brick decomposition can be used in obtaining approximation schemes for other problems as well.

16.1 Introducing the brick decomposition

Given a precision parameter $\epsilon > 0$, an embedded graph G with edge-weights, and a connected subgraph K of G, the *brick decomposition* construction selects a subgraph M of G that includes K and has weight at most $c_1\epsilon^{-1}\text{weight}(K)$ for a constant c_1. (For the construction we give, $\text{weight}(M) \leq (4\epsilon^{-1} + 1)\text{weight}(K)$.)

The starting subgraph K is called the *skeleton* of the brick decomposition. For each face f of M, the subgraph of G enclosed in f is called the *brick* corresponding to f, and the *boundary* of the brick consists of the edges of face

f. The brick decomposition is the decomposition of G into bricks. Note that bricks include their boundaries, and so the bricks are not disjoint.

The special properties of a brick decomposition arise from its construction, which we describe in Section 16.4. Here we describe how a brick decomposition is used in approximation schemes for Steiner tree, and we describe the special properties of brick decomposition that are relevant.

Let G be a planar embedded graph with edge-weights, and let Q be a set of terminals. We use $\mathrm{OPT}(G, Q)$ to denote an optimal Steiner tree. We assume for simplicity of presentation that G has degree at most three; a simple transformation using zero-weight edges can be used to achieve this. Fix a precision $\epsilon > 0$. As mentioned earlier, there are two approaches to obtaining a approximation scheme for Steiner tree, but they start out the same.

> **Step 16.1.** Let K be a Steiner tree whose weight is at most two times optimal.

There is a linear-time algorithm to find such a Steiner tree.

> **Step 16.2.** Construct a brick decomposition with mortar graph M.

The algorithm for constructing a brick decomposition runs in time that is linear when ϵ is considered to be a constant.

Since $\text{weight}(K) \leq 2\text{weight}(\mathrm{OPT}(G, Q))$ and $\text{weight}(M) \leq c_1\epsilon^{-1}\text{weight}(K)$, we infer that $\text{weight}(M) \leq 2c\epsilon^{-1}\text{weight}(\mathrm{OPT}(G, Q))$, where $\mathrm{OPT}(G, Q)$ is an optimal Steiner tree.

Before continuing with the description of the algorithms, we describe the special property underlying the correctness of the algorithms. The basic idea is that there is a nearly optimal Steiner tree that, for each brick B, "crosses" the boundary of B at most a constant number of times (where we consider ϵ to be constant).

To formalize the notion of *crossing* used here, we first describe a transformation of G called *shattering*. The shattered graph $\widehat{G}$ is obtained from G and M as follows.

Start from M.
For each face f of the embedded graph M,
  the vertices and edges of of f occur also on
  the boundary of the corresponding brick;
  embed a copy of the corresponding brick within f.
  For each vertex v on f,
    use an artificial zero-weight edge to connect v with the copy of v.

This process is illustrated in Figure 16.1. Each vertex v that lies on M occurs also in some number of bricks; therefore $\widehat{G}$ contains several copies of v. The copy that in $\widehat{G}$ lies on M is identified with the original vertex of G, and the others are said to be *duplicates*. In particular, the terminals are considered to belong to M, not to the bricks.

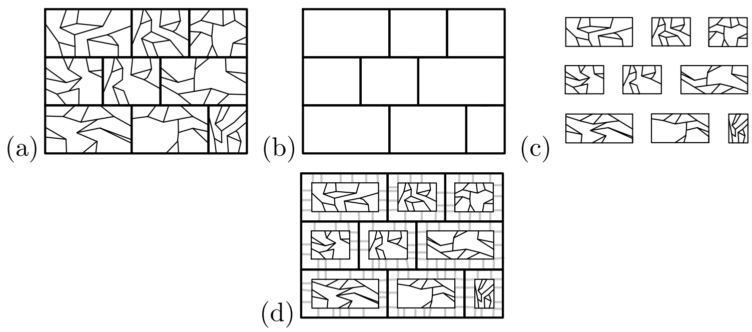


Figure 16.1: (a) An input graph G_{in} with bold edges forming the mortar graph M. (b) The mortar graph M . (c) The set of bricks corresponding to M (d) Bricks shattering the graph.

Now we can state the property underlying our use of the brick decomposition. Because of the artificial zero-weight edges, $\widehat{G}$ mimics G in that any tree T in G corresponds to a tree in $\widehat{G}$ of the same weight and spanning all the vertices spanned by G, and vice versa. We state the theorem in somewhat greater generality, in terms of a forest instead of a tree.

Theorem 16.1.1 (Steiner-Tree Structure Theorem)**.** *For any forest F in G, there exists a forest $\widehat{F}$ of $\widehat{G}$ such that*

- *$weight(\widehat{F}) \leq (1+\epsilon) weight(F)$*
- *$\widehat{F}$ spans all the vertices of M that F spans, and*
- *for each brick B, $\widehat{F}$ includes at most $c_2\epsilon^{-3.5}$ of the artificial edges incident to B.*

Since we are willing to accept approximately optimal solutions, therefore, the algorithm can restrict its attention to Steiner trees that, for each brick, connect between that brick and the rest of the graph only through a few artificial edges. This doesn't quite give us an efficient dynamic program since the algorithm doesn't know *which* artificial edges are used by the shortest such Steiner tree. However, we can get around this obstacle by using the fact that the mortar graph is longer than the optimal Steiner tree by only a constant factor. This fact allows us to select a constant number of artificial edges per brick, and to further restrict the Steiner tree to use no other artificial edges.

The number of such artificial edges is controlled by an integer parameter ρ. We use ∂B to denote the cycle formed by the boundary of brick B. The value of ρ is specified later.

Step 16.3. For each brick B, designate at most ρ of the vertices of the boundary ∂B as *portal* vertices, as follows:

Let v_0 be an arbitrary vertex of ∂B.

Designate v_0 as a portal vertex.
Set $i = 0$.
Repeat:
  Set $i = i + 1$.
  Let v_i be the first vertex of $\partial B[v_{i-1}, \cdot]$ such that
    $\text{weight}(\partial B[v_{i-1}, v_i]) > \text{weight}(\partial B)/\rho$.
  If $v_0 \in V(\partial B(v_{i-1}, v_i])$, stop.
  Otherwise, designate v_i as a portal vertex.

Lemma 16.1.2 (Portal Coverage Lemma)**.** *For any vertex x on ∂B, there is a portal vertex v_i such that*

$$\textit{weight}(\partial B[v_i, x]) \leq \textit{weight}(\partial B)/\rho$$

Proof. Let k be the number of iterations. For some i, the vertex x lies on the subpath $B[v_i, v_{(i+1) \bmod k})$. □

Lemma 16.1.3 (Portal Cardinality Lemma)**.** *Each brick has at most ρ portal vertices.*

Proof. Let k be the number of iterations. Each iteration selects a subpath of weight more than $\text{weight}(\partial B)/\theta$, so we have $\text{weight}\partial B \geq \sum_{i=1}^{p} \text{weight}\partial B[v_{i-1}, v_i] > k\,\text{weight}(\partial B)/\theta$, and so it follows that $k < \rho$. □

An artificial edge is called a *portal* edge if it is incident to a portal vertex. Let $\widehat{G}_\rho$ denote the subgraph of $\widehat{G}$ by excluding artificial edges that are not portal edges.

Now we combine the fact that $\text{weight}(M) \leq 2c_1\epsilon^{-1}\text{weight}(\text{OPT}(G, Q))$ with the fact that there is an approximately optimal forest using only $c_2\epsilon^{-3.5}$ artificial edges per brick.

Corollary 16.1.4. *Suppose the portal parameter ρ is assigned $8c_1c_2\epsilon^{-5.5}$. For any forest F in G, there exists a forest $\tilde{F}$ of $\widehat{G}_\rho$ such that*

- *$\textit{weight}(\tilde{F}) \leq (1 + 2\epsilon)\textit{weight}(F)$, and*
- *for any vertices u and v of M, if F connects u and v then so does $\tilde{F}$.*

Proof. By applying the Steiner-Tree Structure Theorem (Theorem 16.1.1), we obtain a forest $\widehat{F}$ of $\widehat{G}$ having weight at most $(1+\epsilon)\text{weight}(F)$. However, $\widehat{F}$ uses artificial edges that are not portal edges, so we must modify it.

Let B be a brick, and let xy be an artificial edge incident to B in $\widehat{G}$ that is used by $\widehat{F}$, where x lies on the boundary of B and y belongs to M. Note that x and y correspond to the same vertex of G. By the Portal Coverage Lemma (Lemma 16.1.2), there is a subpath P of ∂B from x to a portal vertex v_i, and $\text{weight}(P) \leq \text{weight}(\partial B)/\rho$. We replace the artificial edge xy with (i) the copy of P in $\widehat{G}$ that belongs to the brick copy B, (ii) the portal edge incident to v_i, and (iii) the copy of P in $\widehat{G}$ that belongs to M. The increase in weight is at most $2\text{weight}(\partial B)/\rho$.

We similarly replace each artificial edge incident to B with one copy of a path, a portal edge, and another copy of the path. Since $\widehat{F}$ used at most $c_2\epsilon^{-3.5}$ artificial edges incident to B, the total weight increase associated with brick B is at most

$$c_2\epsilon^{-3.5} \cdot \frac{2}{\rho}\text{weight}(\partial B)$$

We carry out the process on every brick B. The total weight increase is at most

$$c_2\epsilon^{-3.5} \cdot \frac{2}{\rho}\sum_B \text{weight}(\partial B)$$

where the sum is over all bricks in the brick decomposition. Each edge of M is on the boundary of two bricks, so

$$\sum_B \text{weight}(\partial B) \leq 2\text{weight}(M) \leq 4c_1\epsilon^{-1}\text{weight}(\text{OPT}(G, Q))$$

Therefore the total weight increase is at most

$$d\epsilon^{-3.5} \cdot \frac{2}{\rho} \cdot 4c_1\epsilon^{-1}\text{weight}(\text{OPT}(G, Q))$$

which is at most $\epsilon\,\text{weight}(\text{OPT}(G, Q))$ if we set $\rho = \rceil 8c_1c_2\epsilon^{-5.5}\rceil$. □

16.2 Spanner

Building on Corollary 16.1.4, we can complete the steps of the spanner construction.

Step 16.4.

Initialize G_1 to include M.
For each brick B,
  for each subset S of portal vertices of ∂B,
*   add to G_1 an optimal Steiner tree T of S in B

Can the line marked $*$ be implemented fast?

Lemma 16.2.1. *Given an n-vertex strict plane graph B with edge-weights or vertex-weights, and given a k-element set Q of terminals all on the boundary of a single face of B, there is an $O(k^3n)$ algorithm to find a minimum-weight Steiner tree.*

Problem 16.2.2. *Prove Lemma 16.2.1.*

We need to show that the resulting graph G_1 satisfies the requirements of a spanner for Steiner tree.

Theorem 16.2.3. *Let G_1 be the graph resulting from the algorithm described above applied to a plane graph G with edge-weights and with terminal set Q. Then*

1. *weight*$(G_1) \leq \alpha$ *weight*$(OPT(G,Q))$, *and*
2. *weight*$(OPT(G_1,Q)) \leq (1+2\epsilon)$*weight*$(OPT(G,Q))$

where α depends only on ϵ.

Proof. To show the first property, note the tree T added to the spanner in Line $*$ of Step 16.4 has weight at most weight(∂B). Since ∂B has at most ρ portal vertices, the number of trees added for B is at most 2^ρ. Therefore the total weight of all the trees added for B is at most 2^ρweight(∂B). Summing over all bricks B, the total weight is at most $2^\rho \cdot 2$weight(M). Since weight$(M) \leq 2c_1\epsilon^{-1}$weight(OPT$(G,Q)$) and $\rho = \lceil 8cd\epsilon^{-5.5} \rceil$, we can set $\alpha = 2c_1\epsilon^{-1}2^\rho$ to achieve the first property.

For the second property, suppose T is an optimal Steiner tree. By Corollary 16.1.4, there exists a tree $\tilde{T}$ of $\widehat{G}_\rho$ such that the vertices of M that are connected in T are also connected in $\tilde{T}$. In particular, since M contains all the terminals, $\tilde{T}$ is a Steiner treee of the terminals. Moreover, weight$(\tilde{T}) \leq (1+2\epsilon)$weight$(T)$.

Consider each brick B in turn, and consider the intersection of $\tilde{T}$ with B. The intersection consists of a collection of connected components, each of which includes some subset of portal vertices. For each connected component K, replace K with the minimum-weight Steiner tree in B connecting the same portal vertices, specifically the one included in Line $*$ of Step 16.4. The replacement does not increase the weight. After all the replacements are complete, the resulting tree includes only edges belonging to G_1. □

16.3 Beyond spanners: A more efficient PTAS for Steiner tree

16.4 Brick decomposition: the construction

In this section, we give the algorithm for constructing a *brick decomposition.* Let G be the plane graph, let K be the skeleton (a connected subgraph of G), and let $\epsilon > 0$ be the error parameter.

def BrickDecomposition(G, K, ϵ)
  input: plane graph G, skeleton K (connected subgraph of G), precision parameter $\epsilon > 0$

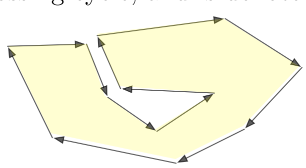
Initialize $M = K$
For each face f of K,
1 Decompose the subgraph of G embedded in f into *strips*,
2 and add strip boundaries to M

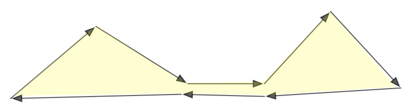
3 For each strip D,
4 identify *columns* $C_1, \ldots, C_t$
5 for $i = 0, \ldots, k-1$, let $w_i = \sum\{\text{weight}(C_p)\ : p \equiv i \pmod k\}$
6 let $i^* = \text{minarg}\ _i w_i$
7 add to M the columns C_p such that $p \equiv i^* \pmod k$

The lines marked 1 and 4 will be explained in more detail. The parameter k in Lines 5 and 7 is set to $\lceil \epsilon^{-1} \rceil$.

16.4.1 Strip decomposition

A *simple region* of a graph G is defined by a non-self-crossing cycle of darts. The edges,vertices, and faces that lie "to the right" of the cycle of darts are considered to belong to the region. The boundary of a region R is the non-self-crossing cycle, and is denoted ∂R. Here are two examples:





The figures illustrate that an edge can occur twice on the boundary of a simple region. A face of an edge subgraph of G defines a simple region of G; the boundary is the boundary of the face.

A $(1+\epsilon)$-*strip* of G is a simple region R of G whose boundary is $\partial R = P_1 \circ P_2$ where P_1 and P_2 are paths satisfying the following condition:

> For any path L whose only vertices in P_i are start(L) and end(L),
>
> $$\text{weight}(P_i[\text{start}(L), \text{end}(L)]) \leq (1+\epsilon)\text{weight}(L) \tag{16.1}$$
>
> Informally, P_i is a nearly shortest path between its endpoints among paths that stay inside the region.

We will typically refer to P_1 and P_2 as the southern and northern boundaries of the strip, but the distinction is arbitrary.

The basic strategy for decomposing a simple region R into strips is straightforward.

When there is a path L whose only vertices in ∂R are start(L) and end(L) such that

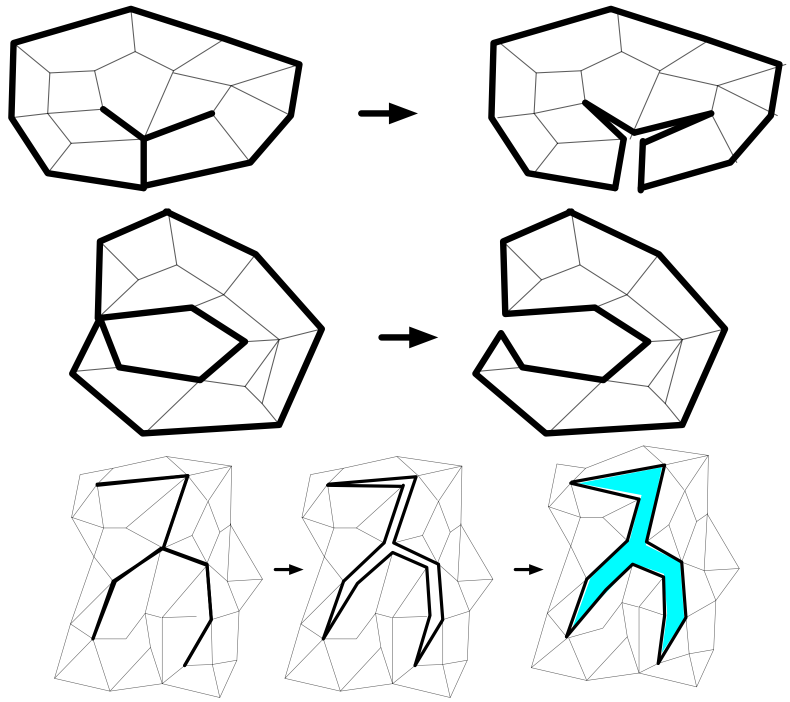
$$(1+\epsilon)\text{weight}(L) \leq \text{weight}(\partial R[\text{start}(L), \text{end}(L)]),$$

split R into subregions R_1 and R_2 by cutting along L, and then recursively subdivide each subregion. The first modification to this strategy is to ensure that for one of these regions, say R_1, there is no need for further splitting. To achieve this, we choose L to be in a sense as close as possible to the boundary on the R_1 side. This enables us to bound the total weight of all strip boundaries.

The second modification to the strategy involves relaxing the requirement that no internal vertices of L lie on ∂R. Instead, we require that L not *cross*

∂R. This modification is not necessary for the construction; it allows for a more convenient implementation.

Line 1 of the algorithm BRICKDECOMPOSITION(G, K, ϵ) tells us to take the subgraph Y of G embedded in a face f of K, and decompose it into *strips*. Note that the boundary of f in G might not be a simple cycle; edges and vertices might occur multiple times on the boundary. The first step in strip decomposition is to modify Y by duplicating edges and vertices on the boundary of f, getting a plane graph H_0 with a face f_0 that consists of duplicates of all the edges and vertices of f. Here are some examples of this transformation:



In the last example, the skeleton K is a tree. It has only one face f, and the entire graph is embedded in that face. Every edge occurs twice on the face. The number of occurences of each vertex v is $\deg_K(v)$. In the graph H_0 resulting from the transformation, every edge of f is represented by two duplicates in the face f_0, and each vertex v is represented by $\deg_K(v)$ duplicates.

To describe the rest of strip decomposition, we need a definition. Let H be a plane graph and let f be a face of H. An $(1+\epsilon)$-*shortcut of* H *with respect to* f is a shortest path P in H with the following properties:

- start(P) and end(P) belong to the boundary of f, and
- the start(P)-to-end(P) subpath of the boundary of f has weight that is greater by a $1+\epsilon$ factor than weight(P):

$$(1+\epsilon)\text{weight}(P) \leq \text{weight}(\delta(f)[\text{start}(P), \text{end}(P)])$$

The $(1+\epsilon)$-shortcut P is *minimally enclosing* if there is no $(1+\epsilon)$-shortcut whose start and end lie on the start(P)-to-end(P) subpath of the boundary of f (except for one whose start and end are the same as that of P).

We give a procedure, StripDecomposition(H, f), that is applied to a plane graph H with a face f. (The procedure is somewhat abstract; a fast implementation will be given later.) To complete the strip decomposition, we call StripDecomposition(H_0, f_0).

def StripDecomposition(H, f):
1 If there is no $(1+\epsilon)$-shortcut, return $\{H\}$.
Let P be a minimally enclosing $(1+\epsilon)$-shortcut.
2 Let D be the subgraph enclosed by the cycle $\delta(f)[\text{start}(P), \text{end}(P)] \circ \text{rev}(P)$
3 Let H' be the graph obtained from H by deleting $D - P$
4 Let f' be the face of H' that is obtained from f by replacing $\delta(f)[\text{start}(P), \text{end}(P)]$ with P.
5 return $\{D\} \cup$ StripDecomposition(H', f')

The subgraph D found in Line 2 is called a *strip*. The *boundary* of the strip is $\delta(f)[\text{start}(P), \text{end}(P)] \circ \text{rev}(P)$. We refer to P as the southern boundary and we refer to $\delta(f)[\text{start}(P), \text{end}(P)]$ as the northern boundary. (The distinction is not so important.)

Lemma 16.4.1.

- *Every subpath of the southern boundary of a strip is a $(1+\epsilon)$-shortest path.*
- *Every proper subpath of the northern boundary of a strip is a $(1+\epsilon)$-shortest path.*

Lemma 16.4.1 permits some leeway in Line 1 of StripDecomposition. The algorithm can terminate in Line 1 provided a weaker condition holds, that there are vertices u and v on f such that there is no $(1+\epsilon)$-shortcut whose endpoints are both in $\delta(f)[u, v]$ and none whose endpoints are both in $\delta(f)[v, u]$. If this condition holds, H is itself a strip; by designating

Note that the weight of the boundary of f' is less than that of f by at least $\epsilon\,\text{weight}(P)$. Therefore we can charge weight(P) to the reduction in boundary weight in going from H, f to H', f'. Therefore, when StripDecomposition(H_0, f_0) is executed, the total weight of all shortcuts found is at most ϵ^{-1} times the weight of the boundary of f_0.

Lemma 16.4.2. *Over all faces f of K, the total weight of all shortcuts found is at most $\epsilon^{-1} 2\,\text{weight}(K)$.*

Proof. Each edge of K occurs over all faces f of K, so the sum of weights of boundaries of faces of K is $2\,\text{weight}(K)$. Combining this with the above charging argument yields the lemma. □

16.4.2 Columns

For each strip D, Line 2 of BRICKDECOMPOSITION identifies *columns* within D. We now describe the algorithm to identify columns. We use S and N to denote the southern and northern boundaries of D.

```
def FindColumns(D, S, N):
   Let v_0, v_1, ..., v_k be the vertices of S in west-to-east order.
   Initialize v* = v_0.
   For i = 1, 2, ..., k,
      If weight(S[v*, v_i]) > ε dist_D(v_i, N):
*        Designate as a column a shortest v_i-to-N path in D.
         v* = v_i
```

The start v_i of a column is called the *base* of the column. Each column is a path from south to north. Note that the endpoints v_0 and v_k of the southern boundary are endpoints of the north boundary. We consider the one-vertex path consisting of v_0 to be a column. Similarly the one-vertex path consisting of v_k is a column.

Let $w_0, w_1, \ldots, w_r$ be the bases of the columns found by the algorithm. The condition in the algorithm ensures that

$$\text{weight}(S[w_{j-1}, w_j]) > \epsilon\, dist_D(w_j, N)$$

for $j = 1, \ldots, r$. We obtain the following lemma.

Lemma 16.4.3. *The sum of the weights of columns of a strip is at most ϵ^{-1} times the weight of the strip's southern boundary.*

In Line 7 of BRICKDECOMPOSITION, the weight of columns of strip D that are added to M is at most $\frac{1}{k}$ times the total weight of the columns of D. Since $k = \lceil \epsilon^{-1} \rceil$, the weight of columns of strip D that are added to M is at most the weight of the southern boundary of D.

Corollary 16.4.4. *The length of the mortar graph is $(4\epsilon^{-1}+1)weight(K)$ where K is the skeleton.*

16.5 Statement of subroutine lemmas for Steiner tree structure theorem

For the following three lemmas, G is a planar embedded graph, P is an $1+\epsilon$-short path forming part of the boundary of G, and T is a tree in G that intersects P only at leaves of T.

The first lemma is illustrated in Figure 16.2.

Lemma 16.5.1. *There is a procedure* SPAN0 *such that* SPAN0(P, T) *returns a subpath of P spanning $V(T) \cap V(P)$ whose total length is at most $(1+\epsilon)length(T)$.*

Figure 16.2: A subgraph is squished to a $1+\epsilon$-short path. The resulting subpath includes all vertices common to the subgraph and the path, and is not much longer.

Proof. Let P' be the shortest subpath of P that spans all the vertices of $T \cap P$. There is a path Q in T between the endpoints of T. Since P is $1+\epsilon$-short, $\text{length}(P') < (1+\epsilon)\text{length}(Q) \leq (1+\epsilon)\text{length}(T)$. □

Definition 16.5.2 (Joining vertex)**.** *Let H be a subgraph of G such that P is a path in H. A* joining vertex *of H with P is a vertex of P that is the endpoint of an edge of $H - P$.*

The second lemma is illustrated in Figure 16.3. The proof is given in Section 16.6.7.

Lemma 16.5.3. *There is a procedure* SPAN1(P, T, r) *that, for a vertex r of T, returns a subgraph of $P \cup T$ of length at most $(1+\epsilon)$length(T) that spans all the vertices of $\{r\} \cup (V(T) \cap V(P))$ and has at most $\epsilon^{-1.45}$ joining vertices with P.*

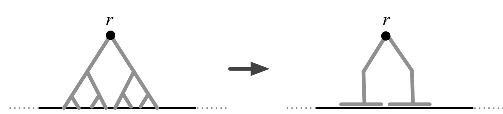


Figure 16.3: The output subgraph spans all vertices of P spanned by the input subgraph, and also spans x, but the output subgraph has fewer joining vertices with P.

The third lemma is illustrated in Figure 16.4. The proof is given in Section 16.6.8.

Lemma 16.5.4. *There is a procedure* SPAN2(P, T, x, y) *that, for x and y vertices of T, returns a subgraph of $P \cup T$ of length at most $(1+2\epsilon+\epsilon^2)$length(T) that spans all the vertices of $\{x, y\} \cup (T \cap P)$ and has at most $2\epsilon^{-2.5}$ joining vertices with P. are constants.*

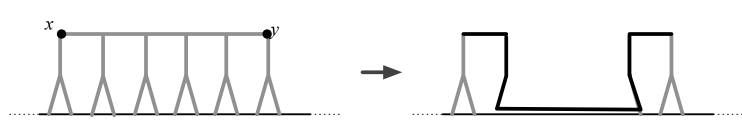


Figure 16.4: The output subgraph spans all vertices of P spanned by the input tree, and also spans x and y, but the output subgraph has fewer joining vertices with P.

16.6 Structure of Steiner tree within bricks

Lemma 16.6.1. *The counterclockwise boundary of a brick B equals $W_B \circ S_B \circ E_B \circ N_B$, where*

1. *S_B is 1-short in B, and every proper subpath of N_B is $(1+\epsilon)$-short in B.*
2. *$S_B = S_1 \circ S_1 \circ \cdots \circ S_\kappa$ where, for each vertex x of $S_i[\cdot,\cdot)$,*

$$\mathit{length}(S_i[\cdot,x]) \leq \epsilon\, \mathit{dist}(x, N_B) \tag{16.2}$$

Note that some of the paths S_i might be empty.

Theorem 16.6.2 (Structural Property of Bricks)**.** *Let B be a plane graph with boundary $N \cup E \cup S \cup W$, satisfying the brick properties of Lemma 16.6.1. Let F be a set of edges of B. There is a forest $\tilde{F}$ of B with the following properties:*

F1) *If two vertices of the boundary of B are connected in F then they are connected in $\tilde{F}$.*

F2) *The number of joining vertices of $\tilde{F}$ with N and with S is at most $4(\kappa+1)\epsilon^{-2.5}$.*

F3) *$\mathit{length}(\tilde{F}) \leq (1+5\epsilon)(\mathit{length}(F) + \mathit{length}(E) + \mathit{length}(W))$.*

16.6.1 Paths $\bar{P}_0, \ldots, \bar{P}_\kappa$

We now present the proof of the theorem. We refer to N as *north*, etc. We define P_κ to be the eastern boundary E of the brick. We define $\bar{P}_\kappa = P_\kappa$. We inductively define $\bar{P}_{\kappa-1}, \bar{P}_{\kappa-2}, \ldots, \bar{P}_0$ as follows. (The definition is illustrated in Figure 16.5.) For $i = \kappa-1, \kappa-2, \ldots, 0$, if $F \cup W$ has an S_i-to-north path that does not intersect $\bar{P}_{i+1}, \bar{P}_{i+2}, \ldots, \bar{P}_\kappa$, let P_i be the rightmost such path, and define $\bar{P}_i = S_i[\cdot, \text{start}(P_i)] \circ P_i$. If there is no such path, define $\bar{P}_i = \emptyset$.

Let P be a nontrivial $\bar{P}_i$-to-north path or $\bar{P}_i$-to-south path in F. We call P a *sprit* of $\bar{P}_I$ if $P - \text{start}(P)$ avoids $\bar{P}_i, \ldots, \bar{P}_\kappa$. It is a *northern* sprit if end(P) belongs to N and a *southern* sprit if end(P) belongs to S.

Because P_i is rightmost, we obtain the following lemma, whose proof is outlined in Figure 16.6.

Lemma 16.6.3 (Sprit Lemma)**.** *For $i = 0, 1, \ldots, \kappa$,*

- *if P is a northern sprit of $\bar{P}_i$ then end(P) is strictly left of end(P_i) on N, and*
- *if P is a southern sprit of $\bar{P}_i$ then end(P) is strictly left of start(P_i) on S, and*

Inequality 16.2 implies that, for $i = 0, \ldots, \kappa-1$,

$$\text{length}(\bar{P}_i) \leq (1+\epsilon)\text{length}(P_i) \tag{16.3}$$

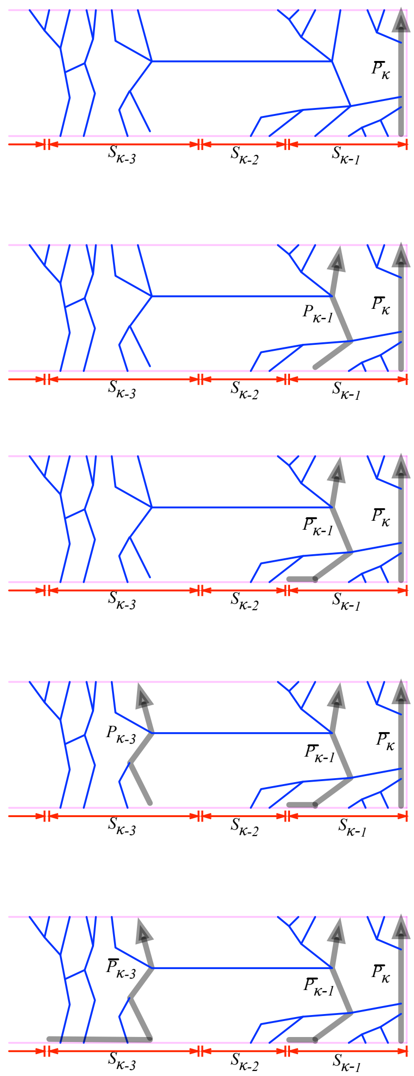


Figure 16.5: The top figure shows a fragment of a brick with $\bar{P}_\kappa$ defined as the eastern boundary. The second figure shows $P_{\kappa-1}$, defined as the rightmost south-to-north path that avoids P_κ. The third figure shows $\bar{P}_{\kappa-}$, which is obtained from $P_{\kappa-1}$ by prepending the to-start(P_κ) prefix of $S_{\kappa-1}$. There is *no* south-to-north path that originates in $S_{\kappa-2}$ and avoids $\bar{P}_{\kappa-1}$, so $\bar{P}_{\kappa-2}$ is empty. The fourth figure shows $P_{\kappa-3}$, defined as the rightmost south-to-north path that does not intersect $P_{\kappa-1}$ or P_κ. The fifth figure shows $\bar{P}_{\kappa-3}$, which is obtained from $P_{\kappa-3}$ by prepending a prefix of $S_{\kappa-3}$. Note that south-to-north paths originating in this prefix become $\bar{P}_{\kappa-3}$-to-north paths.

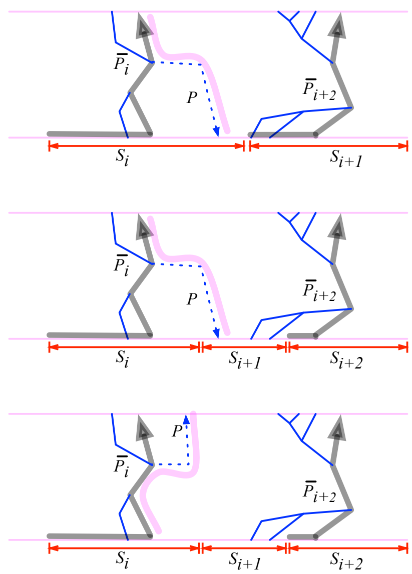


Figure 16.6: Suppose F contains a path P from P_i to a southern vertex to the right of start(P_i) (in the first and second figures) or from P_i to a northern vertex to the right of end(P_i) (in the third figure). In the first and third figure, the magenta contour indicates that P_i is not the rightmost S_i-to-north path avoiding $\bar{P}_{i+1}, \ldots, \bar{P}_\kappa$, a contradiction. In the second figure, end(P) belongs to S_{i+1}. Ordinarily end(P) would therefore belong to $\bar{P}_{i+1}$, but in this case $\bar{P}_{i+1} = P_{i+1} =$. However, the magenta contour indicates that there is an S_{i+1}-to-north path avoiding $\bar{P}_{i+2}, \ldots, \bar{P}_\kappa$, a contradiction.

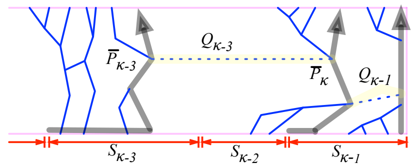


Figure 16.7: The paths $Q_{\kappa-1}$ and $Q_{\kappa-3}$ are signified by the dashed lines.

16.6.2 The forest F' and paths $Q_0, \ldots, Q_\kappa$

Let F' be a minimal subgraph of $F \cup W \cup \bigcup_{i=0}^{\kappa} \bar{P}_i$ that contains $\bigcup_i \bar{P}_i$ and that preserves connectivity among vertices of the boundary of B. Since F' is a subgraph of $F \cup W \cup \bigcup_{i=0}^{\kappa} \bar{P}_i$, Inequality 16.3 implies that

$$\begin{aligned} \text{length}(F') &\leq \text{length}(F \cup W \cup \bigcup_{i=0}^{\kappa} \bar{P}_i) \\ &\leq \text{length}(E) + \text{length}(W) + (1+\epsilon)\text{length}(F) \end{aligned}$$

For $i = 0, \ldots, \kappa-1$, if there is a path in F' from $\bar{P}_i$ to $\bar{P}_{i+1} \cup \bar{P}_{i+2} \cup \cdots \cup \bar{P}_\kappa$ whose internal vertices are not in $\bar{P}_i \cup \cdots \cup P_\kappa$, let Q_i be such a path, as shown in Figure 16.7. Otherwise let $Q_i = \emptyset$.

Claim 16.6.4. *Every internal vertex of Q_i has degree two in F'.*

Proof. Assume for a contradiction that some internal vertex u of Q_i has an incident edge e not on Q_i. By minimality of F', the edge e must be required to preserve connectivity among vertices of the boundary of B. Let v be a boundary vertex of B such that removing e separates u and v. Let P be the u-to-v path in F'.

If v is on $\bar{P}_j$ for some $j > i$ then u and v are connected via $\bar{P}_j$ and a suffix of Q_i, a contradiction. Otherwise, a prefix of Q_i together with P violates the Sprit Lemma (Lemma 16.6.3). □

16.6.3 The forest $\tilde{F}$

The construction of $\tilde{F}$ is as follows. Each connected component K of $F' - \bigcup_i Q_i$ is replaced with a component $\tilde{K}$ that achieves at least K's connectivity among boundary vertices of B and endpoints of paths Q_i. This ensures that $\tilde{K} = \bigcup_K \tilde{K} \cup \bigcup_i Q_i$ achieves the connectivity of F' among boundary vertices of B, which is property F1 of Theorem 16.6.2.

For each component K, moreover, $\text{length}(\tilde{K}) \leq (1 + 3\epsilon + \epsilon^2)\text{length}(K)$.

Therefore

$$\begin{aligned}
\mathrm{length}(\tilde{F}) &\leq \sum_i \mathrm{length}(Q_i) + \sum_K \mathrm{length}(\tilde{K}) \\
&\leq \sum_i \mathrm{length}(Q_i) + (1+3\epsilon+\epsilon^2)\sum_K \mathrm{length}(K) \\
&\leq (1+3\epsilon+\epsilon^2)\mathrm{length}(F') \\
&\leq (1+3\epsilon+\epsilon^2)((1+\epsilon)\mathrm{length}(F) + \mathrm{length}(E) + \mathrm{length}(W))
\end{aligned}$$

which proves part F3 of the theorem, assuming $\epsilon \leq 1/5$.

Our construction will ensure that there are at most $\kappa + 1$ components K for which $\tilde{K}$ has joining vertices with the boundary of B, and for each of these components, $\tilde{K}$ has at most $4\epsilon^{-2.5}$ joining vertices. Thus the total number of joining vertices is $4(\kappa+1)\epsilon^{-2.5}$.

16.6.4 Type-1 and type-2 components

For each connected component K of $F' - \bigcup_i Q_i$, the construction of $\tilde{K}$ depends on what kind of component K is. We say K is a *type-1* component if the boundary vertices in K are all internal vertices of S or all internal vertices of N, and is a *type-2* component otherwise.

For $i = 0, \ldots, \kappa$, let K_i be the connected component of $F' - \bigcup_j Q_j$ that contains $\bar{P}_i$ (if $\bar{P}_i \neq \emptyset$).

Lemma 16.6.5. *If K is a type-2 component then $K = K_i$ for some i.*

Proof. By minimality of F', every component of F' contains some boundary vertices.

- Suppose K contains a vertex of E. Since $\bar{P}_\kappa = E$ and $\bar{P}_\kappa$ belongs to $F - \bigcup_i Q_i$, K contains a vertex of S (namely start($\bar{P}_\kappa$)) and a vertex of N (namely end($\bar{P}_\kappa$)).

- Suppose K contains a vertex of W. Recall that F' is a subgraph of $F \cup W \cup \bigcup_i \bar{P}_i$ that preserves connectivity among vertices of the boundary. It follows that K contains a vertex of S and a vertex of W.

- Suppose K does not contain a vertex of E and a vertex of W. Since K is not a type-1 component, it must therefore contain a vertex of S and a vertex of N.

K, therefore, contains a vertex of S and a vertex of N. Let P be a south-to-north path in K, and let i be the integer such that start(P) belongs to $S_i[\cdot,\cdot)$. If start(P) belongs to $S_i[\cdot, \mathrm{start}(P_i)]$ then start(P) belongs to $\bar{P}_i$, so start(P) belongs to K_i. If not, then, by choice of P_i, the rightmost P intersects $\bar{P}_j$ for some $j > i$, so start(P) belongs to K_j. $\square$

16.6.5 Construction of $\tilde{K}$

First suppose K is of type 1. If its boundary vertices are in S, we let $\tilde{K} := \textsc{Span0}(S, K)$. If its boundary vertices are in N, we let $\tilde{K} := \textsc{Span0}(N, K)$. In either case, $\tilde{K}$ has no joining vertices, and $\text{length}(\tilde{K}) \leq (1+\epsilon)\text{length}(K)$.

Now we consider type-2 components. By Lemma 16.6.5, $K_0, \ldots, K_\kappa$ are the only type-2 components. For $i = 0, \ldots, \kappa$, we obtain $\tilde{K}_i$ from K_i by

(a) separating K_i into two parts, K_i^N and K_i^S,

(b) applying Span2 or Span1 to each part, and

(c) adding a subpath of $S_i[\cdot, \text{start}(P_i)]$.

By Lemmas 16.5.3 and 16.5.4, the total number of joining vertices is at most $4\epsilon^{-2.5}$, and the total length is at most $(1+2\epsilon+\epsilon^2)\text{length}(K_i)+\text{length}(S_i[\cdot, \text{start}(P_i)])$, which is in turn at most $(1+3\epsilon+\epsilon^2)\text{length}(K_i)$. These are the properties needed in the analysis in Section 16.6.3.

16.6.6 Decomposition of K_i into K_i^N and K_i^S

For $i = 0, \ldots, \kappa$, if $\bar{P}_i \neq \emptyset$, let x_i be the first vertex on $\bar{P}_i$ such that there is an x_i-to-north sprit P^N. If $\text{end}(P^N)$ were right of $\text{end}(\bar{P}_i)$ on N then it would violate the Sprit Lemma, so it is strictly left of $\text{end}(\bar{P}_i)$ on N.

Lemma 16.6.6. *For any vertex x of $\bar{P}_i(x_i, \cdot]$, there is no x-to-south sprit of P_i.*

Proof. Let P^N be an x_i-to-north sprit. Suppose P is an x-to-south sprit. By the Sprit Lemma, P^N and P are both left of $\bar{P}_i$, so they cross, forming a cycle with $\bar{P}_i$. This contradicts the minimality of F'. □

Lemma 16.6.7. *If there is an integer $j < i$ such that Q_j connects to $\bar{P}_i$ then end$(Q_j) = x_i$.*

Proof. The proof is illustrated in Figure 16.8. Combining Q_j with the to-start(Q_j) prefix of P_j yields a southern spar of $\bar{P}_i$. Therefore, by Lemma 16.6.6, $\text{end}(Q_j)$ is not strictly after x_i on $\bar{P}_i$. Combining Q_j with the from-start(Q_j) prefix of P_j yields a northern spar of $\bar{P}_i$. Therefore, by choice of x_i, $\text{end}(Q_j)$ is not strictly before x_i on $\bar{P}_i$. □

As illustrated in Figure 16.9, we decompose K_i into edge-disjoint subgraphs K_i^N and K_i^S as follows. K_i^N consists of the subpath $\bar{P}_i[x_i, \cdot]$ and paths between this subpath and N. K_i^S consists of the subpath $\bar{P}_i[\cdot, x_i]$ and paths between this subpath and S.

Our intention is to apply the procedure Span1 or Span2 to each of K_i^N and K_i^S, as shown in Figure 16.10, obtaining edge-disjoint trees $\tilde{K}_i^N$ and $\tilde{K}_i^S$, each having at most $2\epsilon^{-2.5}$ joining vertices with N and S, respectively. Because each of these two trees contains x_i, their union is connected.

There are two additional issues, however. First, consider the case, depicted in Figure 16.11, in which the vertex x_i is not on P_i but is on the subpath

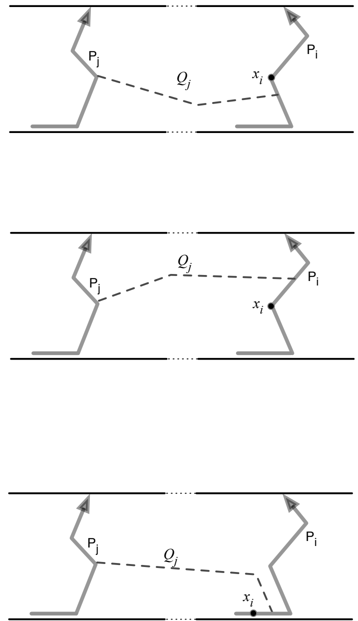


Figure 16.8: The first configuration is impossible since rev(Q_j) connects P_i to N (via P_j), which would contradict the choice of x_i. The second and third configurations are impossible since there is no way for an x_i-to-N path to avoid crossing Pj or Q_j.

$S_i[\cdot, \text{start}(P_i)]$ prepended to P_i to form $\bar{P}_i$. In this case, the tree $\tilde{T}_i^S$ is not required to include the vertices of $S_i[x_i, \text{start}(P_i)]$ other than x_i. In this case, therefore, we include this subpath in $\tilde{K}_i$.

The second issue is this: if $Q_i \neq \emptyset$, we need the new tree $\tilde{K}_i$ to include start(Q_i). Fortunately, the procedure Span2 allows us to specify *two* vertices to be spanned. The construction of $\tilde{K}_i$ is as follows. First we define $\tilde{K}_i^N$ and $\tilde{K}_i^S$:

- If $Q_i = \emptyset$, we define $\tilde{K}_i^N := Span1(N, K_i^N, x_i)$ and $\tilde{K}_i^S := Span1(S, K_i^S, x_i)$.
- If start(Q_i) belongs to K_i^N, we define $\tilde{K}_i^N := Span2(N, K_i^N, x_i, \text{start}(Q_i))$ and $\tilde{K}_i^S := Span1(S, K_i^S, x_i)$.
- Otherwise, we define $\tilde{K}_i^S := Span2(S, K_i^S, x_i, \text{start}(Q_i))$ and $\tilde{K}_i^N := Span1(N, K_i^N, x_i)$

We then define $\tilde{K}_i$ to be the union of $\tilde{K}_i^N$, $\tilde{K}_i^S$, and $S_i[x_i, \text{start}(P_i)]$. The analysis of length and number of joining vertices is as described in Section 16.6.5.

16.6.7 Span1

In this section, G is a planar embedded graph, P is an $1+\epsilon$-short path forming part of the boundary of G, r is a vertex of G, and T is an r-rooted tree of G

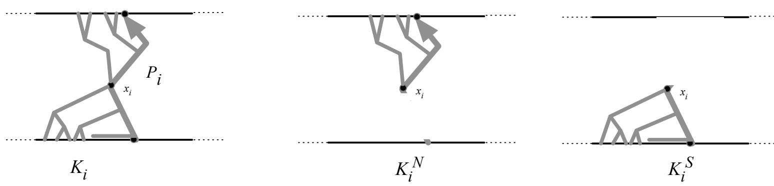


Figure 16.9: The component K_i is split at x_i into the northern part, K_i^N, and the southern part, K_i^S.

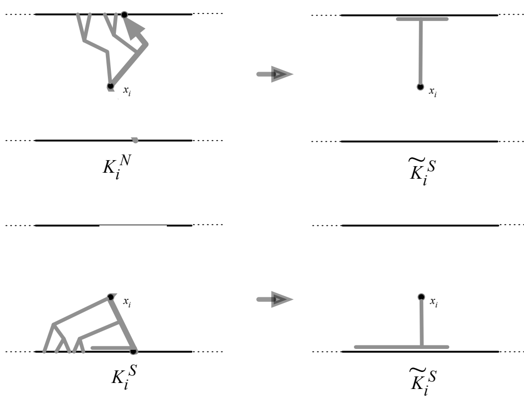


Figure 16.10: The northern subtree and the southern subtree are separately simplified (to reduce their number of joining vertices) using SPAN1.

that intersects P only at leaves of T.

For a rooted subtree T' of T, every root-to-leaf path of T' ends on P, so these paths are ordered according to the positions of the leaves along P. In this section and the next, we are particularly interested in the leftmost and rightmost root-to-leaf paths.

In this section, we give the proof of Lemma 16.5.3, which is paraphrased here:

> *There is a procedure* SPAN1(P, T, r) *that returns a subgraph of* $T \cup P$ *that (a) has length at most* $(1+\epsilon)$*length*(T)*, (b) spans all the vertices of* $\{r\} \cup (V(T) \cap V(P))$*, and (c) has at most* $\epsilon^{-1.45}$ *joining vertices with* P.

We start with a subprocedure.

Lemma 16.6.8. *There is a subprocedure* REDUCEDEGREE(P, T, r) *that, if the root* r *of* T *has more than two children, returns a subpath* P' *of* P *and an* r*-to-*P' *path* Q *consisting of edges of* T *such that that*

- $P' \cup Q$ *spans* $\{r\} \cup (V(T') \cap V(P))$*, and*

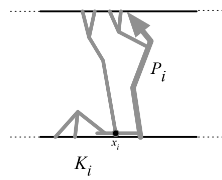


Figure 16.11: When x_i does not belong to P_i, the tree T_i^S does not include the vertices of $S_i(x_i, \text{start}(P_i)]$ so $\tilde{T}_i^S$ need not. In this case, therefore, we include the subpath $S[x_i, \text{start}(P_i)]$ in $\tilde{T}$.

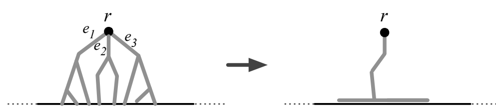


Figure 16.12: Replace the tree T with the minimal subpath of P that contains all leaves of T, together with the path to P that starts at a middle child edge of the root.

- *length*$(P') \leq (1+\epsilon)$*length*$(T - Q)$.

Proof. Let Q_1 and Q_3 denote, respectively, leftmost and rightmost root-to-leaf paths in T, and let e_1 and e_3 be the first edge in, respectively Q_1 and Q_2. Because G is planar, P is on the boundary of G, and r has at least two children, we have $e_1 \neq e_3$. Let e_2 be another child edge of r, and let Q_2 be the root-to-leaf path in T that starts with e_2.

The procedure returns the tree consisting of Q_2 and the minimal subpath of P that contains all leaves of T. The only joining vertex is the end of Q_2. Since $\text{rev}(Q_1) \circ Q_3$ is a start(P')-to-end(P') path and P is $1+\epsilon$-short, $\text{length}(P') \leq \text{length}(Q_1) + \text{length}(Q_3)$. ☐

By repeated application of ReduceDegree, we obtain

Lemma 16.6.9. *There is a subprocedure* ReduceDegrees(P, T, r) *that returns a subtree T' of T and a collection of subpaths $P_1, \ldots, P_k$ of P such that*

- $T' \cup \bigcup_i P_i$ *spans* $\{r\} \cup (V(T) \cap V(P))$ *and*
- *length*$(\bigcup_i P_i) \leq (1+\epsilon)$*length*$(T - T')$

Now we prove Lemma 16.5.3 by describing Span1(P, T, r). Let T' be the tree derived from T in Lemma 16.6.9. Every vertex of T' has at most two children. We will use an argument that requires that every nonleaf vertex has two children. We therefore modify T' by splicing out each nonroot vertex with exactly one child, merging the two incident edges into a single edge whose length is the sum

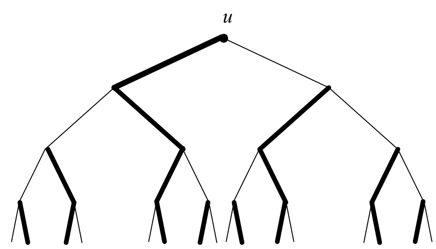
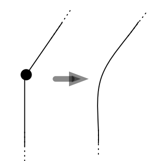


Figure 16.13: The edges in bold are the zig-zag edge.



of the lengths of the merged edges.
This will ensure that every nonleaf vertex (except possibly the root r) has two children.

- If r has two children, we define T'' to be the resulting modified tree. We show how to replace T'' with an r-rooted tree $\widehat{T}$ that satisfies properties (a) through (c) of Lemma 16.5.3.
- If r has only one child, r', we define T'' to be the r'-rooted subtree, and apply the argument of Case 1 to obtain a replacement r'-rooted tree $\widehat{T}$. Then $\widehat{T} \cup \{r\text{-to-}r' \text{ path}\}$ satisfies properties (a) through (c) of Lemma 16.5.3.

Say that an edge uv of T'' is a *zig-zag* edge if the two-step path from the parent $p(u)$ of u to u to v either goes from $p(u)$ to a left child and from u to a right child, or goes from $p(u)$ to a right child and from u to a left child. The above definition is inapplicable if u is the root of T'. Therefore we (rather arbitrarily) define the left edge of the root of T'' to be a zig-zag edge.

As in breadth-first search, the *level* of a vertex is equal to the number of edges traversed when going from the root of T' to the vertex. The level of an edge is equal to the level of its endpoint that is closer to the root. For each level i, let L_i denote the total length of the zig-zag edges at level i.

Let k be a level to be determined later. The procedure obtains a tree $\widehat{T}$ from T'' as follows (see Figure 16.14). For each level-k vertex u, the procedure applies a subprocedure similar to ReduceDegree: replace the u-rooted subtree of T'' (which we denote T''_u) with another u-rooted tree $\widehat{T}_u$ consisting of

- the minimal subpath P' of P spanning the vertices of $T''_u \cap P$, and
- the u-to-P' path that includes u's zig-zag child edge.

The construction ensures that $\widehat{T}$ spans all the vertices of $V(T'') \cap V(P)$. Moreover, the number of joining vertices is 2^k. We shall ensure that $k \leq \log_\phi \epsilon^{-1}$,

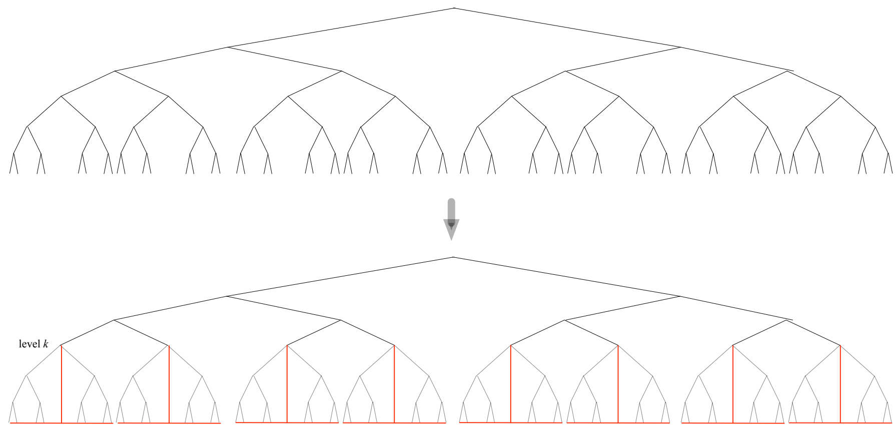


Figure 16.14: For each level-k vertex u, the subtree rooted at u is replaced with the minimal subpath P' of P containing the leaves of that subtree, together with a shortest u-to-P' path. The new subtrees are indicated in red.

where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. Hence the number of joining vertices is at most $\epsilon^{-1.45}$.

It remains to show that there is a choice of k for which length$(\widehat{T}) \leq (1+\epsilon)$length$(T'')$. The argument is illustrated in Figure 16.15. The length of the path P' is not much longer than the path Q_1 through T''_u between the endpoints of P'. The length of the shortest u-to-P' path is no longer than any u-to-P' path Q_2 in T''_u. Thus

$$\text{length}(\widehat{T}_u) \leq \text{length}(Q_1) + \text{length}(Q_2) \tag{16.4}$$

Since Q_1 and Q_2 are contained in T''_u, we would like to argue that length$(Q_1)+$ length$(Q_2) \leq$ length(T''_u). However, that might not be true because Q_1 and Q_2 overlap.

We address this difficulty by selecting the path Q_2 carefully and by selecting the level k carefully. As shown in Figure 16.15, we can select Q_2 so it shares only one edge e with Q_1. Moreover, in arguing that $\widehat{T}_u$ is not much longer than T''_u, we can use the fact that there are many edges that belong to T'_u but do *not* belong to $\widehat{T}_u$, including in particular the dashed edges in Figure 16.15.

For each level-k vertex u, we choose the path Q_2 to be the path starting at u that traverses the next two zig-zag edges and then continues to a leaf without taking any more zig-zag edges. For example, if, as in Figure 16.15, u is a right child of its parent, then Q_2 traverses u's left child edge, then goes right and continues going right until reaching a leaf.

The advantage of this choice of path is that, after the first two edges, Q_2 avoids all zig-zag edges. Note also that Q_1 also avoids all zig-zag edges except for the child zig-zag child edge of u. Let e denote this edge. Since e is the only edge common to Q_1 and Q_2, and none of the zig-zag edges at levels $k+2$ and

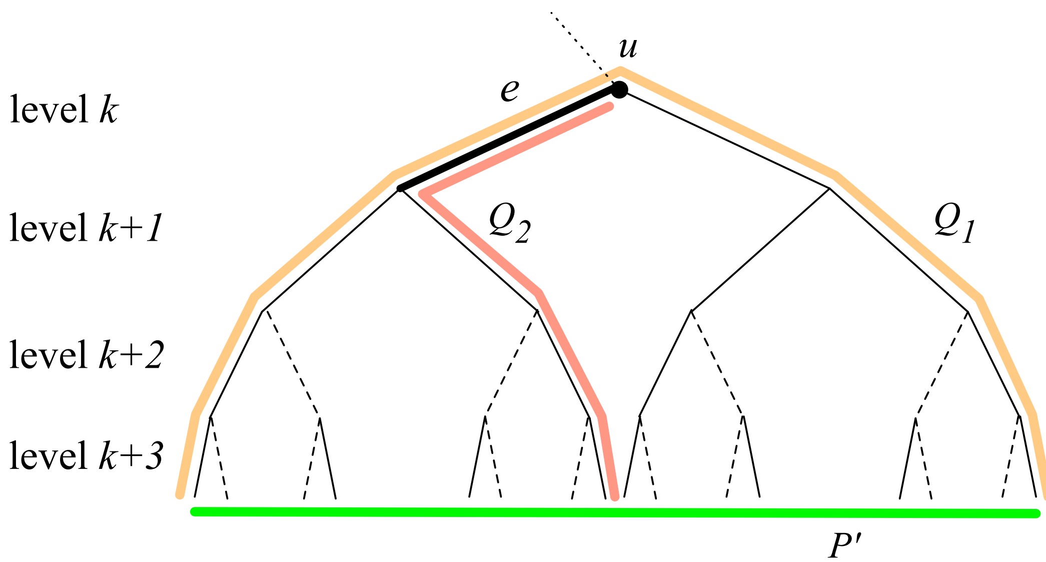


Figure 16.15: We bound the length of the replacement tree $\widehat{T}_u$ by the length of the path Q_1 through the tree from its leftmost leaf to its rightmost leaf, plus the length of the path Q_2 from the root to one of its leaves. We choose Q_2 to first traverse two zig-zag edges and subsequently not traverse any zig-zag edges. The dashed edges in the figure are zig-zag edges that are in neither Q_1 nor Q_2. The total length of these edges is a credit against the debit represented by the edge e that appears in both Q_1 and Q_2.

above belong to either Q_1 or Q_2,

$$\text{length}(Q_1) + \text{length}(Q_2) + \text{length}(\text{zig-zag edges at levels } k+2, k+3, \ldots) \\ \leq \text{length}(T''_u) + \text{length}(e)$$

where we include here only zig-zag edges belonging to T''_u.

We combine this inequality with Inequality 16.4, obtaining

$$\text{length}(\widehat{T}_u) + \text{length}(\text{zig-zag edges at levels } k+2, k+3, \ldots) \leq \text{length}(T''_u) + \text{length}(e) \tag{16.5}$$

Note that edge e is a level-k zig-zag edge. Now we sum 16.5 over all level-k vertices u, obtaining

$$\sum_u \text{length}(\widehat{T}_u) + L_{k+2} + L_{k+3} + \cdots \leq \sum_u \text{length}(T''_u) + L_k \tag{16.6}$$

We add the lengths of edges at levels less than k to both sides. These edges appear in both T'' and $\widehat{T}$, so we obtain

$$\text{length}(\widehat{T}) + L_{k+2} + L_{k+3} + \cdots \leq \text{length}(T'') + L_k \tag{16.7}$$

Combining this inequality with the following claim completes the proof of property (c).

Claim: *There exists $k \leq \log_\phi \epsilon^{-1}$ such that $L_k \leq \epsilon$ length$(T'') + L_{k+2} + L_{k+3} + \cdots$, where $|phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio*

Let $k_0 = \lfloor \log_\phi \epsilon^{-1} \rfloor$. Define $F_{-2}, F_{-1}, F_0, F_1, F_2, \ldots, F_{k_0}$ by the recurrence

$$\begin{aligned} F_{k_0} &= 1 \\ F_{k_0-1} &= 1 \\ F_k &= F_{k+2} + F_{k+3} + F_{k+4} + \cdots + F_{k_0} \end{aligned}$$

The recurrence implies that $F_k = F_{k+1} + F_{k+2}$ for $-2 \leq k \leq k_0 - 2$. Therefore $F_k \geq \phi^{k_0-k-1}$, so in particular $F_{-2} \geq \phi^{k_0+1} > \epsilon^{-1}$.

Assume the claim is false. Then, for each integer $0 \leq k \leq k_0$, $L_k > \epsilon\, \text{length}(T'')F_k$, so

$$\begin{aligned} L_0 + L_1 + L_2 + \cdots + L_{k_0} &> \epsilon\, \text{length}(T'')(F_0 + F_1 + F_2 + \cdots + F_{k_0}) \\ &= \epsilon\, \text{length}(T'')(F_{-2}) \\ &> \epsilon\, \text{length}(T'')(\epsilon^{-1}), \end{aligned}$$

which is a contradiction. Thus the claim is true.

16.6.8 Span2

Again G is a planar embedded graph, P is an $1+\epsilon$-short path forming part of the boundary of G, and T is an r-rooted tree of G that intersects P only at leaves of T.

In this section, we give the proof of Lemma 16.5.4, which is reproduced here:

> There is a procedure $\textsc{Span2}(\cdot,\cdot,\cdot,\cdot)$ such that, for vertices x, y of K, $\textsc{Span2}(P, K, x, y)$ returns a subgraph of $P \cup K$ of length at most $(1 + 2\epsilon + \epsilon^2)\text{length}(T)$ that spans all the vertices of $\{x, y\} \cup (K \cap P)$ and has at most $2\epsilon^{-2.5}$ joining vertices with P, where c is a constant.

Let Q be the unique x-to-y path in T. Removing the edges of Q from T breaks T into a forest consisting of trees rooted at vertices of Q with leaves on P. Let $r_1, \ldots, r_k$ be the roots in order along Q and let $T_1, \ldots, T_k$ be the trees.

If $k < 2\lceil\epsilon^{-1}\rceil$ then obtain a tree $\widehat{T}$ from T by applying $\textsc{Span1}$ to each tree T_i, replacing it with a tree $\widehat{T}_i$ that has at most $\epsilon{-1.45}$ joining vertices. It follows that $\widehat{T}$ has at most $2\epsilon^{-2.45}$ joining vertices.

Assume therefore that $k \geq 2\lceil\epsilon^{-1}\rceil$. For $j = 1, 2, \ldots, \lceil\epsilon^{-1}\rceil$, define $f(j) = k - \lceil\epsilon^{-1} + j$, and define $L_j = \text{length}(T_j) + \text{length}(T_{f(j)})$. Let $j^* = \text{minarg}\,_j L_j$. Then

$$L_{j^*} \leq \epsilon\,\text{length}(T_1 \cup T_2 \cup \cdots \cup T_k) \tag{16.8}$$

The transformations are illustrated in Figure 16.16. Write $Q = Q_1 \circ Q_2 \circ Q_3$ where $\text{start}(Q_2) = r_{j^*}$ and $\text{end}(Q_2) = r_{f(j^*)}$. Let Q_4 be the leftmost root-to-leaf path in T_{j^*} and let Q_5 be the rightmost root-to-leaf path in $T_{f(j^*)}$. Say a tree T_j is a *middle tree* if $j^* \leq j \leq f(j^*)$. As illustrated in Figure 16.16, we obtain $\widehat{T}$ from T as follows:

1. Remove Q_2 and the middle trees, and add Q_4, Q_5, and the $\text{end}(Q_4)$-to-$\text{end}(Q_5)$ subpath P' of P.

2. Apply $\textsc{Span1}$ to each of the non-middle trees.

Since there are at most ϵ^{-1} non-middle trees, and each is replaced with a tree with at most $\epsilon^{-1.45}$ joining vertices, there are at most $\epsilon^{-2.45} + 2$ joining vertices. (The two come from Q_4 and Q_5.)

The increase in length due to the second step is at most $1 + \epsilon$ times the length of the non-middle trees. Since P is $1 + \epsilon$-short and $\text{rev}(Q_4) \circ Q_2 \circ Q_5$ is a $\text{start}(P')$-to-$\text{end}(P')$ path,

$$\text{length}(P') \leq (1 + \epsilon)\text{length}(Q_4) + \text{length}(Q_2) + \text{length}(Q_5) \tag{16.9}$$

Since Q_4 is part of T_{j^*} and Q_5 is part of $T_{f(j^*)}$,

$$\text{length}(Q_4) + \text{length}(Q_5) \leq L_{j^*} \tag{16.10}$$

The increase in length due to the first step is

$$\begin{aligned}
&\text{length}(P') + \text{length}(Q_4) + \text{length}(Q_5) - \text{length}(Q_2) - \text{length}(\text{middle trees}) \\
&\quad \leq \quad \text{length}(P') - \text{length}(Q_2) \\
&\quad \leq \quad (1 + \epsilon)[\text{length}(Q_4) + \text{length}(Q_2) + \text{length}(Q_5)] - \text{length}(Q_2) \\
&\quad \leq \quad (1 + \epsilon)[\text{length}(Q_4) + \text{length}(Q_5)] + \epsilon\,\text{length}(Q_2) \\
&\quad \leq \quad (1 + \epsilon)\epsilon\,\text{length}(T_1 \cup \cdots \cup T_k) + \epsilon\,\text{length}(Q_2)
\end{aligned}$$

Hence the total increase is at most $(1 + \epsilon)\epsilon + \epsilon$ times the length of T.

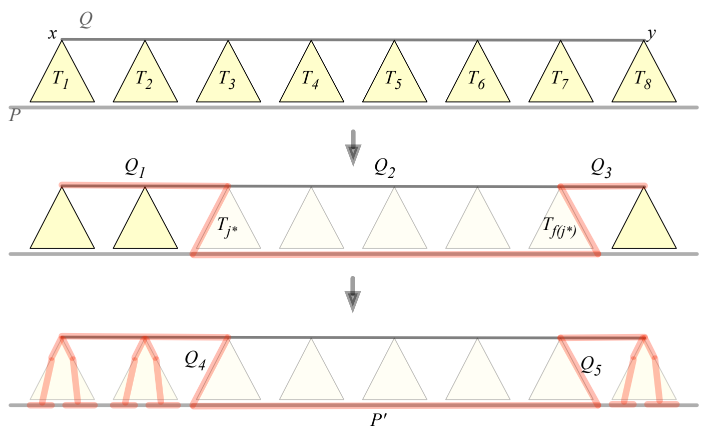


Figure 16.16: The original tree T is shown at the top. It consists of an x-to-y path Q and trees $T_1, \ldots, T_k$ rooted at vertices of Q. In the first step, the subpath Q_2 and the middle trees are replaced by the paths Q_4 and Q_5 and the subpath P' of P. In the second step, Span1 is applied to the non-middle trees.