Chapter 4

Planar embedded graphs

4.1 Planar embeddings

We say that an embedding π of a graph $G = (V, E)$ is *planar* if it satisfies Euler's formula: $n - m + \phi = 2\kappa$, where n=number of nodes, m=number of arcs, ϕ=number of faces, and κ=number of connected components. In this case, we say (π, E) is a *planar embedded graph* or *plane graph.*

Problem 4.1.1. *Specify formally a smallest embedded graph that is not a planar embedded graph. (This is not the assume as giving the smallest graph that has no planar embedding.) You should give the embedding π and a drawing in which the darts are labeled. (You will have to find some way of drawing the embedding even though it is not planar.) Then give the dual in the same way, using a permutation and a drawing.*

The definition of planarity immediately implies the following lemma.

Lemma 4.1.2. *π is a planar embedding of G iff, for each connected component G' of G, the restriction π' of π to darts of G' is a planar embedding of G.*

Lemma 4.1.3. *The dual of a planar embedded graph is planar.*

Problem 4.1.4. *Prove Lemma 4.1.3.*

4.2 Contraction preserves planarity

Our goal for this section is to show that contracting an edge preserves planarity.

Lemma 4.2.1. *Let G be a planar embedded graph, and let e be an edge that is not a self-loop. Then G/e is planar.*

Proof. Let n, m, ϕ, κ be the number of vertices, edges, faces, and connected components of G. By planarity, $n - m + \phi = 2\kappa$. Let $G' = \text{dual}(G^* - e)$. Let

n', m', ϕ', κ' be the number of vertices, edges, faces, and connected components of G'. Clearly $m' = m - 1$. By Lemma 3.4.8, $n' = n - 1$. By Lemma 3.4.6, e is not a cut-edge in G^*. It follows from the Cut-Edge Lemma (Lemma 3.2.6) that $\kappa' = \kappa$. Therefore $n' - m' + \phi' = 2\kappa'$. □

4.3 Sparsity of planar embedded graphs

Lemma 4.3.1 (Sparsity Lemma)**.** *For a planar embedded graph in which every face has size at least three, $m \leq 3n - 6$, where m is the number of edges and n is the number of vertices.*

Problem 4.3.2. *Prove the Sparsity Lemma, and show that the upper bound is tight by showing that, for every integer $n \geq 3$, there is an n-vertex planar embedded graph whose number of edges achieves the bound.*

Problem 4.3.3. *Prove a lemma analogous to the Sparsity Lemma in which faces of size one are permitted.*

4.3.1 Strict graphs and strict problems

A face of size two consists of two *parallel edges*, edges with the same endpoints. A face of size one consists of a self-loop. A graph with neither parallel edges nor self-loops is a *strict* graph. [1]

For many optimization problems, it is sufficient to consider strict graphs. Consider a optimization problem whose input includes a graph G and a dart vector $\boldsymbol{c}$. We say a graph optimization problem is *strict* if there is a constant-time procedure that, given an instance $\mathcal{I}$ and a pair of parallel edges or a self-loop, modifies the instance to eliminate one of the parallel edges or the self-loop, such that, given an optimal solution for the modified instance, an optimal solution for the original instance can be obtained in constant time.

Consider, for example, the problem of finding a *minimum-weight spanning tree* . A self-loop can simply be eliminated because it will never appear in any spanning tree. Given a pair of parallel edges, the one with greatest weight can be eliminated since it will not appear in the minimum-weight spanning tree. Therefore finding a minimum-weight spanning tree is a strict problem. Many other problems discussed in this book, such as shortest paths, maximum flow, the traveling salesman problem, and the Steiner tree problem, can similarly be shown to be strict. For a strict problem, we generally assume that the input graph is strict and therefore has at most three times as many edges as vertices.

We also discuss a problem, two-edge-connected spanning subgraph, that is not, strictly speaking, strict. However, by using a similar technique we can ensure that there are no *triples* of parallel edges. It follows that we can assume for this problem that there are at most six times as many edges as vertices.

4.3.2 Semi-strictness

Strictness is too strict. A weaker property, *semi-strictness*, can be more easily established and maintained. We say an embedded graph is *semi-strict* if every face has size at least three. The Sparsity Lemma applies to such graphs.

The strictness of a problem can be exploited more thoroughly to obtain an algorithm.

Theorem 4.3.4. *There is a linear-time algorithm to compute a minimum-weight spanning tree in a planar embedded graph.*

Here is a (not fully specified) algorithm for computing a minimum-weight spanning tree.

```
def MST(G):
1 if G has no edges, return ∅
2 let ê be an edge of G contained in some MST of G
3 contract ê
4 eliminate some parallel edges
  return {ê} ∪ MST(G)
```

The choice of $\hat{e}$ in Line 2 is guided by the following observation.

Lemma 4.3.5. *Let G be a connected undirected graph with edge-weights, let v be a vertex of G, and let e be a minimum-weight edge incident to v. Then there is a minimum-weight spanning tree of G that contains e.*

Problem 4.3.6. *Prove Lemma 4.3.5, and then prove Theorem 4.3.4 by showing how to implement* MST *for semi-strict planar embedded graphs in such a way that each iteration takes constant time.*

4.3.3 Orientations with bounded outdegree

An *orientation* of a graph is a set $\mathcal{O}$ of darts consisting of exactly one dart of each edge. We say it is an α-orientation if each each vertex is the tail of at most α darts.

Corollary 4.3.7 (Orientation Corollary)**.** *Every semi-strict planar embedded graph has a 5-orientation.*

Problem 4.3.8. *Prove the Orientation Corollary.*

One simple application of the Orientation Corollary is maintaining a representation of a planar embedded graph to support queries of the form

"Is there an edge whose endpoints are u and v?"

Here is the representation. For each vertex u, maintain a list of u's outgoing darts. To check whether there is an edge with endpoints u and v, search in the list of u and the list of v. Since each list has at most five darts, answering the query takes constant time.

4.3.4 Maintaining a bounded-outdegree orientation for a dynamically changing graph

For an unchanging graph, the same bound can be obtained for all graphs simply by using a hash function. However, the orientation-based approach can be used even when the graph undergoes edge deletions and contractions, and we will see how this can be used in efficient implementations of other algorithms.

An orientation $\mathcal{O}$ is represented by an array adj[·] indexed by vertices. For vertex v, adj[v] is a list consisting of the darts in $\mathcal{O}$ that are outgoing from v.

Let G be a semi-strict planar embedded graph, and let $\mathcal{O}$ be a 14-orientation of G. Suppose G' is obtained from G by deleting an edge e. Then $\mathcal{O} - \{\text{darts of } e\}$ is a 14-orientation for G', and the representation adj[·] can be updated as follows:

```
def DELETE(e):
  let v be the endpoint of e such that adj[v] contains a dart d of e
  remove d from adj[v]
```

Suppose instead that G' is obtained from G by contracting e, and that G' remains semi-strict. The vertex resulting from coalescing the endpoints of e might have more than fourteen outgoing darts in $\mathcal{O} - \{\text{darts of } e\}$. However, a 14-orientation of G' can be found as follows:

```
def CONTRACT(e):
  let u and v be the endpoints of e
  let w be the vertex obtained by coalescing u and v
  adj[w] := adj[u] ∪ adj[v] − {darts of e}
  if |adj[w]| > 14,
    S := {w}
    while S ≠ ∅,
      remove a vertex x from S
      for each dart xy ∈ adj[x],
        add yx to adj[y]
        if |adj[y]| > 14, add y to S
      adj[x] := ∅
```

4.3.5 Analysis of the algorithm for maintaining a bounded-outdegree orientation

The key to analyzing the algorithm is the following lemma.

Lemma 4.3.9. *For any semi-strict plane graph G, any orientation $\mathcal{O}$ of G, and any vertex v, there is a path of size at most $\lceil \log_{4/3} |V(G)| \rceil - 1$ from v to a vertex whose outdegree is at most 3.*

Proof. Let V_i denote the set of vertices reachable from v via paths of size at most i consisting of darts in $\mathcal{O}$. We prove that, for each $i \geq 1$, if V_i contains no vertex of outdegree at most 3, then $|V_{i+1}| > \frac{4}{3}|V_i|$. If $V_{\lceil \log_{4/3} n(G) \rceil - 1}$ contains no vertex of outdegree at most 3, it would follow that $|V_{\lceil \log_{4/3} |V(G)| \rceil}| > (4/3)^{\lceil \log_{4/3} |V(G)| \rceil} \geq |V(G)|$, a contradiction.

Let E_i be the set of darts in $\mathcal{O}$ whose tails are in V_i. Suppose that each vertex in V_i has outdegree at least four, so $|E_i| \geq 4|V_i|$. The graph induced by E_i obeys the Sparsity Lemma, so $|E_i| \leq 3|V_{i+1}| - 6$. Combining these inequalities yields $|V_{i+1}| > \frac{4}{3}|V_i|$. □

We show a bound of $O((k+n)\log n)$ on the total time for maintaining a 14-orientation using DELETE and CONTRACT for k operations on an n-vertex graph.

Consider a sequence of semi-strict planar embedded graphs

$$G_0, \ldots, G_k$$

such that, for $i = 1, \ldots, k$, G_i is obtained from G_{i-1} by a deletion or a contraction. Let $n = \max_i n(G_i)$.

Lemma 4.3.10. *There exist 5-orientations $\mathcal{O}_0, \ldots, \mathcal{O}_k$ of $G_0, \ldots, G_k$ respectively, such that, for $i = 1, \ldots, k$, there are at most $\log_{4/3} n$ edges that whose orientations in O_i and $\mathcal{O}_{i-1}$ differ.*

Proof. We construct the sequence $\mathcal{O}_0, \ldots, \mathcal{O}_k$ backwards. Since G_k is semi-strict, it has a 5-orientation. Let $\mathcal{O}_k$ be this 5-orientation.

Suppose we have constructed a 5-orientation $\mathcal{O}_i$ of G_i for some $i \geq 1$. We show how to construct a 5-orientation $\mathcal{O}_{i-1}$ of G_{i-1}. First suppose G_i was obtained from G_{i-1} by contraction of an edge uv, and let w be the vertex of G_i resulting from coalescing u and v. The number of darts in $\mathcal{O}_i$ outgoing from u and v in G_{i-1} is the number outgoing from w in G_i, so is at most five. Hence in G_{i-1} at least one of u and v has fewer than five outgoing darts in $\mathcal{O}_i$. Let d be the dart of uv oriented out of whichever of u and v has fewer outgoing darts in $\mathcal{O}_i$, and let $\mathcal{O}_{i-1} := \mathcal{O}_i \cup \{d\}$. Then $\mathcal{O}_{i-1}$ is a 5-orientation of G_{i-1}.

Now suppose G_i was obtained from G_{i-1} by deletion of an edge or contraction of a leaf edge. Let uv be one of the darts of the edge in G_{i-1} and not in G_i. Let $\mathcal{O}$ be the orientation $\mathcal{O}_i \cup \{uv\}$. Note that $\mathcal{O}$ might not be a 5-orientation because u might have outdegree 6. However, by Lemma 4.3.9, there is a path P of size at most $\lceil \log n \rceil - 1$ consisting of darts in $\mathcal{O}$ from u to a vertex of outdegree at most 3. Let $\mathcal{O}_{i-1}$ be the orientation obtained from $\mathcal{O}$ by replacing the darts of P with their reverses. This replacement reduces the outdegree of u by one and increases the outdegree of the end of P, so $\mathcal{O}_{i-1}$ is a 5-orientation of G_{i-1}. □

Now we can analyze the use of DELETE and CONTRACT in maintaining an 14-orientation of a changing semi-strict plane graph G. The time for a delete operation is $O(1)$. The time for a contract operation is $O(1)$ not including the

time spent in the while-loop of CONTRACT. The time spent in the while-loop is proportional to the number of changes to orientations of edges. The next theorem proves that the number of such changes is $O(m + k \log n)$, which shows that the total time is also $O(m + k \log n)$.

Theorem 4.3.11. *As G is transformed from G_0 to G_1 to $\cdots$ to G_k, the total number of changes to the orientation is $O(n + k \log n)$.*

Proof. Lemma 4.3.10 showed that there are 5-orientations $\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_k$ of $G_0, G_1, \ldots, G_k$ such that each consecutive pair of orientations differ in at most $\lceil \log n \rceil$ edges. When G is one of $G_0, G_1, \ldots, G_k$, we denote by $\mathcal{O}[G]$ the corresponding 5-orientation.

We use $\tilde{\mathcal{O}}$ to denote the orientation maintained by the algorithm (and represented by adj[·]).

For the purpose of amortized analysis, we define the potential function

$$\Phi(G, \widehat{\mathcal{O}}) = |\widehat{\mathcal{O}} - \mathcal{O}[G]|$$

We say an edge of G is *good* if $\widehat{\mathcal{O}}$ and $\mathcal{O}[G]$ agree on its orientation, and *bad* otherwise. Then Φ is the number of bad edges. The value of the potential is always nonnegative and is always at most m, the number of edges in G_0.

Consider the effect on Φ of a delete or contract operation. Since the operation is accompanied by a change in $\mathcal{O}[G]$ in the orientations of at most $\log n$ edges, the potential Φ goes up by at most $\log n$. Since there are k operations, the total increase due to these changes is at most $k \log n$.

The loop in CONTRACT also has an effect on the value of the potential. In each iteration, a vertex x with outdegree greater than 14 is removed from S and the outgoing darts of x are replaced in $\widehat{\mathcal{O}}$ with their reverses. The replacement turns good edges into bad edges and bad edges into good edges. Before the replacement, at most five of x's outgoing darts were in $\mathcal{O}[G]$ so at most five edges were good. Thus at most five good edges turn to bad, and at least $15 - 5 = 10$ bad edges turn to good. The net reduction in Φ is therefore at least $10 - 5 = 5$.

Since the initial value of Φ is at most m and the increase due to operations (not counting the loop) is at most $k \log n$, the total reduction in Φ throughout is at most $m + k \log n$. Since each iteration of the loop reduces Φ by at least 5, the number of iterations is at most $\frac{m + k \log n}{5}$.

Each iteration changes the orientations of many edges; we next analyze the total number of orientation changes. Since each iteration changes at most five edges from good to bad, the total number of edges changed from good to bad is as at most $m + k \log n$. Initially the number of bad edges is at most m, so there are at most $2m + k \log n$ changes of edges from bad to good. Thus the total number of orientation changes is $3m + 2k \log n$. □

4.4 Cycle-space basis for planar graphs

Lemma 4.4.1. *Let G be a planar embedded graph. For each vertex v of G and each vertex f of G^*,*

$$\boldsymbol{\eta}(v) \cdot \boldsymbol{\eta}(f) = 0$$

Proof. Let D^- be the set of darts in f having v as head. Let D^+ be the set of darts in f having v as tail. Since $\boldsymbol{\eta}(v)$ assigns 1 to darts in D^+ and -1 to darts in D^-, the dot product is $|D^+| - |D^-|$. Note that, for each $d \in D^-$, $\pi^*(d)$ belongs to D^+, and, for each $d \in D^+$, $(\pi^*)^{-1}(d)$ belongs to D^-. Hence $|D^+| = |D^-|$. This shows the dot product is zero. □

Corollary 4.4.2 (Cut-Space/Cycle-Space Duality)**.** *The cut space of G^* is the cycle space of G.*

Proof. For simplicity, assume G is connected. Let v_∞ and f_∞ be vertices of G and G^*, respectively. A basis for the cut space of G is

$$\{\boldsymbol{\eta}(v) \ : v \in V(G) - \{v_\infty\}\}$$

and a basis for the cut space of G^* is

$$\{\boldsymbol{\eta}(f) \ : f \in V(G^*) - \{f_\infty\}\}$$

By Lemma 4.4.1, the vectors in the basis for the cut space of G^* are orthogonal to the vectors in the basis for the cut space of G, so the former belong to the orthogonal complement of the cut space of G, i.e. to the cycle space of G. Moreover, the former basis has cardinality exactly one less than the number of faces in G, which equals $|E(G)| - |V(G)| + 1$, which is the dimension of the cycle space of G. This proves the corollary. □

MacLane [S. MacLane, "A combinatorial condition for planar graphs," *Fund. Math.* 28 (1937), p.22–32.] in fact formulated a criterion for planarity based on cycle-cut duality.

4.4.1 Representing a circulation in terms of face potentials

Recall from Section 3.3.5 that a vector in the cycle space of G is called a *circulation* in G. It follows from Cut-Space/Cycle-Space duality (Corollary 4.4.2) that an arc vector $\boldsymbol{\theta}$ is a circulation iff it can be written as a linear combination of basis vectors

$$\boldsymbol{\theta} = \sum \{\rho_f \boldsymbol{\eta}(f) \ : \ f \in V(G^*) - \{f_\infty\}\}$$

The sum does not include a term corresponding to f_∞. It is convenient to adopt the convention that $\rho_{f_\infty} = 0$ and include the term $\rho_{f_\infty}\boldsymbol{\rho}(f_\infty)$ in the sum. We can then represent the coefficients by a face vector $\boldsymbol{\rho}$, and so write

$$\boldsymbol{\theta} = \sum \{\boldsymbol{\rho}[f]\boldsymbol{\eta}(f) \ : \ f \in V(G^*)\} \tag{4.1}$$

Even if $\boldsymbol{\rho}[f_\infty] \neq 0$, since $\boldsymbol{\eta}(f_\infty)$ is a circulation, the sum 4.1 is a circulation. In this context, we refer to the coefficients $\boldsymbol{\rho}[f]$ as *face potentials*.

We can write the relation between a circulation and face potentials more concisely using the dart-vertex incidence matrix A_{G^*} of the dual G^*. (We could call this the *dart-face incidence matrix* of G.)

$$\boldsymbol{\theta} = A_{G^*}\ \boldsymbol{\rho}$$

4.5 Interdigitating trees

Lemma 4.5.1. *Suppose G is a connected plane graph with a spanning tree T. Every cycle in G^* has an edge in T.*

Proof. Let C^* be a cycle in G^*. Since $\boldsymbol{\eta}(C^*)$ is in the cycle space of G^*, it is in the cut space of G by Cut-Space/Cycle-Space duality, so it can be written in terms of the fundamental cut basis of G with respect to T:

$$\boldsymbol{\eta}(C^*) = \sum_{e \in T} \alpha_e \boldsymbol{\eta}(\text{fundamental cut of } e)$$

Since the left-hand side is nonzero, there is at least one edge $\hat{e} \in T$ such that $\alpha_{\hat{e}} \neq 0$. Since different edges of T are not in each other's fundamental cuts, it follows that the sum assigns a nonzero value to a dart of $\hat{e}$. This proves the lemma. □

Corollary 4.5.2. *Let G be a plane graph. If T is a spanning tree of G then the edges $E(G) - E(T)$ form a spanning tree of G^*.*

Problem 4.5.3. *Prove Corollary 4.5.2.*

If T is a spanning tree of a plane graph G_π, we use T^* to denote the spanning tree of G^* whose edges are $E(G) - E(T)$. We refer to T^* as the dual spanning tree with respect to T in G_π. The trees T and T^* are called *interdigitating trees*.

Interdigitating trees combined with rootward computations give rise to simple algorithms for some problems in planar graphs, as illustrated in the following problems. Beware, however, that the choice of the root of T^* might be significant.

Problem 4.5.4. *Using rootward computation (Section 1.1) on the dual tree, give a simple linear-time algorithm for the following problem.*

- input: *a planar embedded graph G, a spanning tree T, and a vertex r*
- output: *a table that, for each nontree edge uv of G, gives the least common ancestor of u and v in T rooted at r*

Problem 4.5.5. *Using the result of Problem 4.5.4, give a simple linear-time algorithm for the following problem.*

- input: *a planar embedded graph G with edge-weights and a spanning tree T*
- output: *a table that, for each nontree edge e of G, gives the total weight of the fundamental cycle of e with respect to T.*

Problem 4.5.6. *Show that a connected planar graph G with edge-weights can be represented so as to support the following operations in $O(\log n)$ amortized time:*

- *Given an edge e of G, determine whether e is in a minimum-weight spanning tree of G.*
- *Given an edge e of G and a number λ, set the weight of e to λ.*

4.6 Simple-cut/simple-cycle duality

Lemma 4.6.1 (Fundamental-Cut/Fundamental-Cycle Duality)**.** *Let G be a connected planar embedded graph with spanning tree T and let $\hat{e}$ be an edge of T. Then*

$$\{\textit{darts of the fundamental cut of } \hat{e} \textit{ in } G \textit{ with respect to } T\} = \{\textit{darts of the fundamental cycle of } \hat{e} \textit{ in } G^* \textit{ with respect to } T^*\}$$

Proof. Let C^* be the fundamental cycle of $\hat{e}$ in G^* with respect to T^*. As in the proof of Lemma 4.5.1, we write $\boldsymbol{\eta}(C^*)$ in terms of the fundamental cut basis of G with respect to T:

$$\boldsymbol{\eta}(C^*) = \sum_{e \in T} \alpha_e \boldsymbol{\eta}(\text{fundamental cut of } e)$$

Since different edges are not in each other's fundamental cuts, for each edge $e \in T$, if $\alpha_e \neq 0$ then $\boldsymbol{\eta}(C^*)$ assigns nonzero values to the darts of e. However, the only edge of T with darts in C^* is $\hat{e}$, so the sum in the right-hand side is just $\alpha_{\hat{e}}\boldsymbol{\eta}(\text{fundamental cut of } \hat{e})$. Furthermore, since the primary dart of $\hat{e}$ is assigned 1 by both $\boldsymbol{\eta}(C^*)$ and $\boldsymbol{\eta}(\text{fundamental cut of } \hat{e})$, we conclude that $\alpha_{\hat{e}} = 1$. Thus $\boldsymbol{\eta}(C^*) = \boldsymbol{\eta}(\text{fundamental cut of } \hat{e})$, which proves the lemma. □

Theorem 4.6.2 (Simple-Cycle/Simple-Cut Theorem)**.** *Let G be a planar embedded graph. A nonempty set of darts forms a simple cycle in G^* iff the set forms a simple cut in G.*

Proof. We prove the theorem for the case in which G is connected. The result immediately follows for disconnected graphs as well.

(*only if*) Let C^* be a simple cycle in G^*. Let $\hat{e}$ be an edge of C^*, and let P^* be the simple path in G^* such that $C^* = P^* \circ \hat{e}$. By the matroid property of forests

(Corollary 3.1.3), there exists a spanning tree T^* of G^* containing the edges of P^*. Note that C^* is the fundamental cycle of $\hat{e}$ with respect to T^*. Therefore, by Fundamental-Cut/Fundamental-Cycle Duality (Lemma 4.6.1), the darts forming C^* are the darts forming a fundamental cut in G, and such a cut is a simple cut by the Fundamental-Cut Lemma (Lemma 3.2.2).

(*if*) Let S_1 be a set of vertices of G such that $\vec{\delta}_G(S_1)$ is a simple cut. Let $S_2 = V(G) - S_1$. By definition of simple cut in a connected graph, for $i = 1, 2$, the vertices of S_i are connected; let T_i be a tree connecting exactly the vertices of S_i. Let d be a primary dart such that d is in $\vec{\delta}(S_1)$ or $\vec{\delta}(S_2)$, and let e be the edge of d. Let $T = T_1 \cup T_2 \cup \{e\}$. Then $\vec{\delta}(S_1)$ or $\vec{\delta}(S_2)$ is a fundamental cut with respect to T, and so by Fundamental-Cut/Fundamental-Cycle Duality (Lemma 4.6.1), its darts form a simple cycle in G^*. □

4.6.1 Compressing self-loops

Figure 4.1 shows some examples of compressing edges.

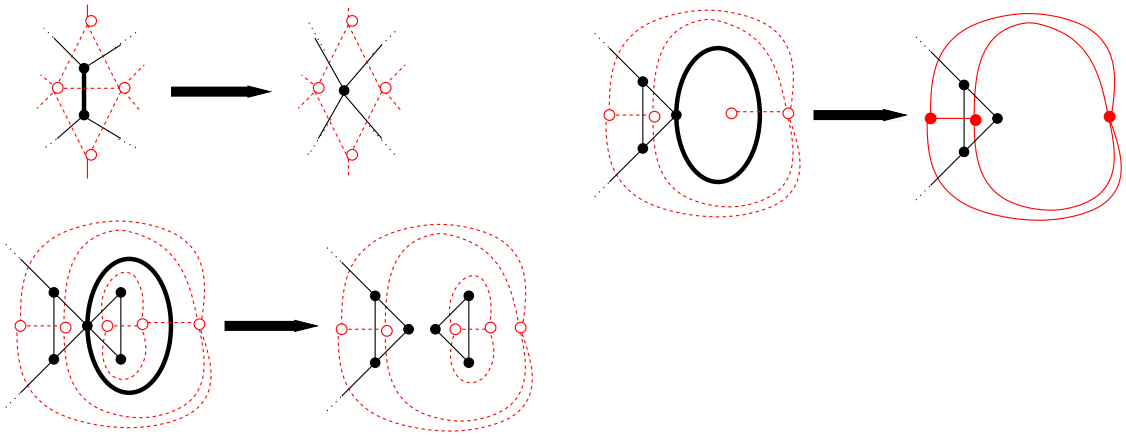

Figure 4.1: Examples of compressing an edge $\hat{e}$ in G (solid lines and filled vertices), i.e. deleting $\hat{e}$ from G^* (dashed lines and open vertices).

Compressing a self-loop in a planar embedded graph is an interesting operation. The graph can be divided into two parts, the part enclosed by the self-loop and the part not enclosed. These parts have only one vertex in common, namely the endpoint of the self-loop. Compression has the effect of duplicating the common vertex, and attaching each part to its own copy.

The Simple-Cycle/Simple-Cut Theorem immediately yields the following.

Corollary 4.6.3. *If e is a self-loop in a planar embedded graph G then e is a cut-edge in G^*.*

We use the corollary to help analyze the effect of compressing a self-loop in a planar graph.

Lemma 4.6.4. *If G is a planar embedded graph and e is a self-loop then G/e is planar.*

Proof. Let n, m, ϕ, κ be the number of vertices, edges, faces, and connected components of G. By planarity, $n - m + \phi - 2\kappa = 0$. Let n', m', ϕ', κ' be the numbers for $G' = \text{dual}(G^* - e)$. In order to prove that G' is planar, it suffices to show that $n' - m' + \phi' - 2\kappa' = n - m + \phi - 2\kappa$.

Clearly $m' = m - 1$. By Corollary 4.6.3, e is a cut-edge in G^*.

First suppose each endpoint of e in G^* has degree greater than one. In this case, deletion of e does not cause the elimination of its endpoints in G^*. Therefore $\phi' = \phi$. Since e is a cut-edge, deleting it increases the number of connected components, so $\kappa' = \kappa + 1$ (using the Connectivity Corollary, which is Corollary 3.4.5). Let v be the common endpoint of the self-loop e in G, and let the corresponding permutation cycle be $(d_0\ d_1\ \cdots\ d_k\ \cdots\ d_\ell)$, where d_0 and d_k are the darts corresponding to e. In G^*, v is a face. Deletion of e in G^* breaks the face up into $(d_0\ d_1\ \cdots\ d_{k-1})$ and $(d_{k+1}\ \cdots\ d_\ell)$ and leaves all other faces alone. Since faces of G^* are vertices of G, we infer $n' = n + 1$. Thus

$$n' - m' + \phi' - 2\kappa' = (n+1) - (m-1) - \phi - 2(\kappa+1) = n - m - \phi - 2\kappa$$

If exactly one of the endpoints of e in G^* has degree one, that endpoint will disappear when e is deleted, so $\phi' = \phi - 1$, and there is no change to the number of connected components. In this case,

$$n' - m' + \phi' - 2\kappa' = n - (m-1) + (\phi - 1) - 2\kappa = n - m + \phi - 2\kappa$$

If both endpoints of e in G^* have degree one, deleting e eliminates both endpoints (vertices of G^*), a connected component, and a face of G^*, so $\phi' = \phi - 2$, $\kappa' = \kappa - 1$, and $n' = n - 1$.

$$n' - m' + \phi' - 2\kappa' = (n-1) - (m-1) + (\phi - 2) - 2(\kappa - 1) = n - m + \phi - 2\kappa$$

□

4.6.2 Compression and deletion preserve planarity

Combining Lemma 4.6.4 with Lemma 4.2.1 shows that compression preserves planarity. Since compression in the dual is deletion in the primal, it follows that deletion preserves planarity. We state these results as follows

Theorem 4.6.5. *For a planar embedded graph G and an edge e, $G - e$ and G/e are planar embedded graphs.*

4.7 Faces, edges, and vertices enclosed by a simple cycle

Let C be a simple cycle of darts in a connected plane graph G_π. Let f_∞ be an arbitrary face, designated the *infinite face*. We say the cycle C *encloses* a face f with respect to f_∞ if $E(C) = \delta(S)$ where $f \in S, f_\infty \notin S$.

Using the Path/Cut Lemma, we immediately obtain

Proposition 4.7.1. *Let G_π be a connected plane graph, and let C be a cycle of G_π. Every path in G^* from a face enclosed by C to a face not enclosed by C goes through an edge of C.*

We say C encloses an edge or vertex if C encloses a face whose boundary contains the edge or vertex, and *strictly* encloses the edge or vertex if in addition the edge or vertex is not on C.

Using Proposition 4.7.1 and Corollary 3.4.4, we obtain

Proposition 4.7.2. *Let G_π be a connected plane graph, and let C be a cycle of G_π. Every path in G from a vertex enclosed by C to a face not enclosed by C goes through a vertex of C.*

The *interior* of a cycle C is the subgraph consisting of edges and vertices enclosed by C. The *exterior* is the subgraph consisting of edges and vertices not strictly enclosed by C. The *strict* interior and exterior are similarly defined.

4.8 Crossing

Two kinds of crossing: sets that cross (violate laminarity) and paths/cycles that cross in a graph sense.

4.8.1 Crossing walks

Let W be a walk, and let $P = a\ W\ b$ and $Q = c\ W\ d$ be walks that are identical except for their first and last darts. Let c' be the successor of c in Q and let d' be the predecessor of d in Q. We say Q forms a *crossing configuration* with P (see Figure 15.2) if the permutation cycle at head(c) induces the cycle ($c\ c'$ rev(a)) and the permutation cycle at tail(d) induces the cycle (rev(d') $b\ d$).

We say a walk P *crosses* a walk Q if a subwalk of P and a subwalk of Q form a crossing configuration.

4.8.2 Non-self-crossing paths and cycles

4.9 Representing embedded graphs in implementations

It makes sense to base our computer representation of embedded graphs on the mathematical representation. We will even use this representation when we

don't care about the embedding.

For the purpose of specifying algorithms, our finite set E will consist of positive integers. For example, if $|E| = m$ then we can use the integers $1 \ldots m$. We also need a way to represent darts, remembering that each element of E corresponds to two darts. We use some convention to represent each dart as an integer. (Two ways: use +/- or use a low-order bit). A permutation π of darts is represented by a pair of arrays, one for the forward direction and one for the backward direction. That way, it takes $O(1)$ time to go from a dart d to the darts $\pi[d]$ and $\pi^{-1}[d]$.

We also have to discuss the implementation of arc deletion. It will be necessary to delete arcs in constant time. The key is to allow some integers to become unused.

Deletion of an arc consists of deletion of its two darts from the representation of the permutation π.