International Journal of Computational Geometry & Applications

A LINEAR-TIME ALGORITHM FOR TESTING THE INSCRIBABILITY OF TRIVALENT POLYHEDRA

MICHAEL B. DILLENCOURT
Information and Computer Science Department, University of California
Irvine, California 92717, USA

and

WARREN D. SMITH
NEC Research Institute, 4 Independence Way
Princeton, NJ 08540, USA

Received (October 15, 1992)
Revised (November 15, 1993)
Communicated by Franz Aurenhammer

ABSTRACT

We present an algorithm for testing the inscribability of a trivalent polyhedron or, equivalently, testing the circumscribability of a simplicial polyhedron. Our algorithm, which runs in linear time and uses only low-precision integer arithmetic, is based on a purely combinatorial characterization of inscribable trivalent polyhedra.

1. Introduction

A polyhedron is *inscribable* if it is combinatorially equivalent to the edges and vertices of the convex hull of a set of noncoplanar points on the surface of the sphere. The question of characterizing the combinatorial structure of inscribable polyhedra dates back to Descartes[1], and was formally posed as an open problem by Jakob Steiner in 1832[2]. A history of the problem and some related questions can be found in (Refs. 3,4,5). Recent interest in the problem has been stimulated by the close connection between inscribable polyhedra and Delaunay triangulations[6], and the importance of Delaunay triangulations as a fundamental structure in computational geometry[7,8].

Rivin has shown that a polyhedron is inscribable if and only if it is possible to assign weights to the edges so that certain conditions are satisfied[9] (also see Refs. 10,11,12,13). A precise statement of Rivin's characterization theorem is given below (Theorem 2.1). Smith has shown that inscribability of a polyhedron can be tested in polynomial time, using the ellipsoid method of linear programming to determine whether a suitable weighting exists[14]. The algorithm of (Ref. 14) tests inscribability of an n-vertex polyhedron using $O(n^{4.376})$ arithmetic operations, each of which operates on real numbers $O(n \log n)$ bits long.

In this paper, we consider the important special case of *trivalent* polyhedra. We present an algorithm for testing the inscribability of an n-vertex trivalent polyhedron in $O(n)$ time, using only low-precision integer arithmetic. In this special case, our results improve on the algorithm based on the ellipsoid method in two ways:

1. Our algorithm is much faster: it runs in $O(n)$ time, and is quite fast in practice.
2. Our algorithm is based on a *purely combinatorial* characterization of inscribable trivalent polyhedra. Rivin's characterization, in contrast, is *numerical.*

Thus we completely (and optimally) solve Steiner's problem for trivalent polyhedra.

The characterization theorem for inscribable trivalent polyhedra is stated and proved in Section 3. We actually establish two equivalent conditions. One is more natural to state, while the other lends itself more readily to efficient testing. The linear-time algorithm for testing inscribability is described in Section 4. In Section 5, we present several examples showing that the results of Section 3 do not hold in the general (nontrivalent) case.

2. Preliminaries

Except as noted, we use the notation of (Ref. 15). $V(G)$ and $E(G)$ denote, respectively, the vertices and the edges of a graph G. If $v \in V(G)$, $d(v)$ denotes its degree. A graph is *k-connected* if deletion of any set of $k-1$ vertices and their attached edges does not disconnect the graph. The planar dual of a planar graph G is denoted by G^*, and the image of $v \in V(G)$ under the dual map is denoted by v^*.

A *cutset* in a graph G is a minimal set of edges whose removal increases the number of components of G. A cutset is *coterminous* if all its edges have a common endpoint, noncoterminous otherwise. Cutsets in a plane graph dualize to cycles, and noncoterminous cutsets dualize to nonfacial cycles.

A *polyhedron* is the convex hull of a finite set of points in $\Re^3$, not all lying in the same plane. A classical theorem of Steinitz (see Ref. 4) asserts that a graph can be realized as the vertices and edges of a polyhedron if and only if it is 3-connected and planar. Thus a 3-connected, planar graph is called a *polyhedral graph.* A polyhedral graph has a topologically unique embedding in the sphere. A polyhedral graph is *trivalent* if all its vertices have degree 3, *simplicial* if all its faces are triangles. The dual of a polyhedral graph is polyhedral[4], and the dual of a trivalent polyhedral graph is simplicial. A polyhedron is *circumscribable* if it can be realized as a (combinatorially equivalent) polyhedron with every face tangent to the unit sphere. It is well known that a polyhedral graph is inscribable if and only if its dual is circumscribable[4].

The following numerical characterization theorem for inscribable polyhedral graphs is due to Rivin[9]. (Also see Refs. 10,11,12,13).

Theorem 2.1 *A polyhedral graph G is inscribable if and only if weights w can be assigned to its edges such that:*

(W1) *For each edge e, $0 < w(e) < 1/2$.*

(W2) *For each vertex v, the total weight of all edges incident on v is equal to 1.*

(W3) *For each noncoterminous cutset $C \subseteq E(G)$, the total weight of all edges in C is strictly greater than 1.*

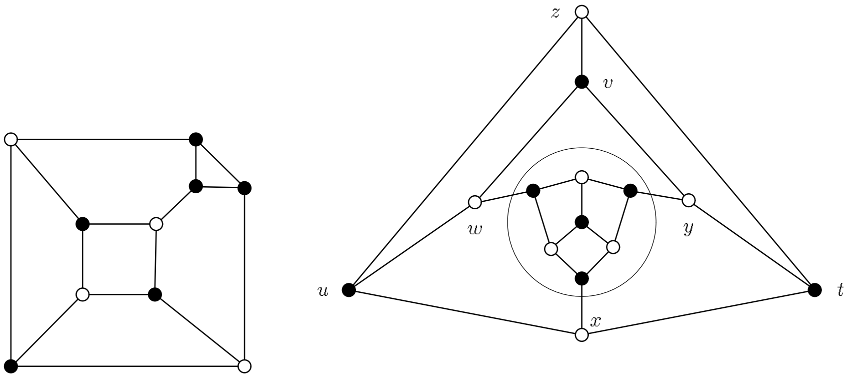
A graph is *1-tough* (Ref. 16) if for all nonempty $S \subseteq V(G)$, $c(G-S) \leq |S|$, where $|S|$ denotes the cardinality of S, $G-S$ is the graph obtained from G by deleting S and all attached edges, and $c(G-S)$ is the number of components of $G-S$. It was shown in (Ref. 17) that any inscribable graph is 1-tough. A graph is *1-supertough* if for all $S \subseteq V(G)$ with $|S| > 1$, $c(G-S) < |S|$. The following theorem is an easy consequence of Theorem 2.1; it may also be proved using the machinery developed in (Ref. 17). Since a bipartite graph cannot be 1-supertough, the two conditions in the following theorem are mutually exclusive.

Theorem 2.2 *Any inscribable graph is either bipartite or 1-supertough.*

Proof. Let G be a non-bipartite inscribable graph, $S \subseteq V(G)$. Weight the edges of G according to Theorem 2.1, and let W be the total weight of the edges connecting vertices of S with vertices of $G-S$. By properties (W1) and (W2), $W \leq |S|$, with equality holding if and only if there are no S-S edges in G. Properties (W1)–(W3) also imply that $W \geq c(G-S)$, with equality holding if and only if each component of $G-S$ consists of a single vertex. Since G is nonbipartite, at least one of these inequalities must be strict. Hence $c(G-S) < |S|$. □

3. Characterization of Inscribable Trivalent Polyhedra

Two examples of noninscribable trivalent polyhedral graphs are shown in Figure 1. The nonbipartite graph of Figure 1(a) fails to be 1-supertough (to see this, remove the four white vertices), and hence it is noninscribable by Theorem 2.2. The graph of Figure 1(b), which is bipartite, also fails to be inscribable. Indeed, suppose the graph could be weighted as in Theorem 2.1, and consider the cutset indicated by the ellipse. These three edges must have total weight > 1. It follows from (W2) that the six remaining edges that are incident on either w, x, or y have total weight < 2, which in turn implies (by the (W2) constraints at t, u, and v) that the three edges incident on z have total weight > 1. But this violates the (W2) constraint at z. The underlying reason why Figure 1(b) is noninscribable is that it is bipartite and has a noncoterminous 3-edge cutset. It follows from Theorem 3.1, below, that these two graphs illustrate the only ways that a trivalent polyhedral graph can fail to be inscribable. A precise statement of the theorem requires introducing the notion of "localization" of a polyhedral graph.



(a) (b)

Fig. 1. Two noninscribable trivalent polyhedra.

3.1. "Localization" of simplicial and trivalent polyhedra

A *separating triad* is a noncoterminous 3-edge cutset. A *separating triangle* in a simplicial polyhedral graph is a nonfacial triangle (e.g., triangle bcd in Figure 2(a)). A separating triad in a trivalent polyhedral graph corresponds, under the dual map, to a separating triangle in a simplicial polyhedral graph. Hence a trivalent polyhedral graph has no separating triad if and only if its dual is 4-connected. (A trivalent polyhedral graph with a 4-connected dual is sometimes called *cyclically 4-connected*[4].)

Let G be a simplicial polyhedral graph, and consider some fixed embedding of G in the plane. Let T be a triangle (not necessarily bounding a face) of G. The complement of T consists of two regions, r and r'. If r' contains at least one vertex, we define the *simplicial localization*, or *s-localization*, of G with respect to r to be the subgraph of G obtained by (1) removing all vertices in r, and (2) removing all vertices in r' that are separated from T by a triangle $U \neq T$. (We say that a vertex v is separated from T by U if any path from v to a vertex of T that is not also a vertex of U must contain a vertex of U.) An s-localization is always simplicial and 4-connected. Each face of an s-localization is either a face of G (called a *normal face*) or a union of faces of G (called a *superface*). Examples are shown in Figure 2(a)–(c). The outer face of an s-localization is called its *bounding face*, and the triangle forming its boundary is called the *bounding triangle*. The restriction that r' must contain a vertex ensures that any s-localization contains at least four vertices. In particular, the s-localization of G with respect to f is defined for any face f of G.

The concept of *trivalent localization*, or *t-localization* is dual to s-localization. Let G be a trivalent polyhedral graph. Choose some vertex v of G, and call it the "outer vertex." If G^* is embedded so that v^* is the outer face of G^*, then the set of t-localizations of G is the set of duals of s-localizations of G^*. Each t-localization of G is trivalent and has no separating triad. A vertex of a t-localization is a *normal vertex* if it corresponds (under the dual map) to a normal face of the corresponding s-localization, a *supervertex* if it corresponds to a superface. Notice

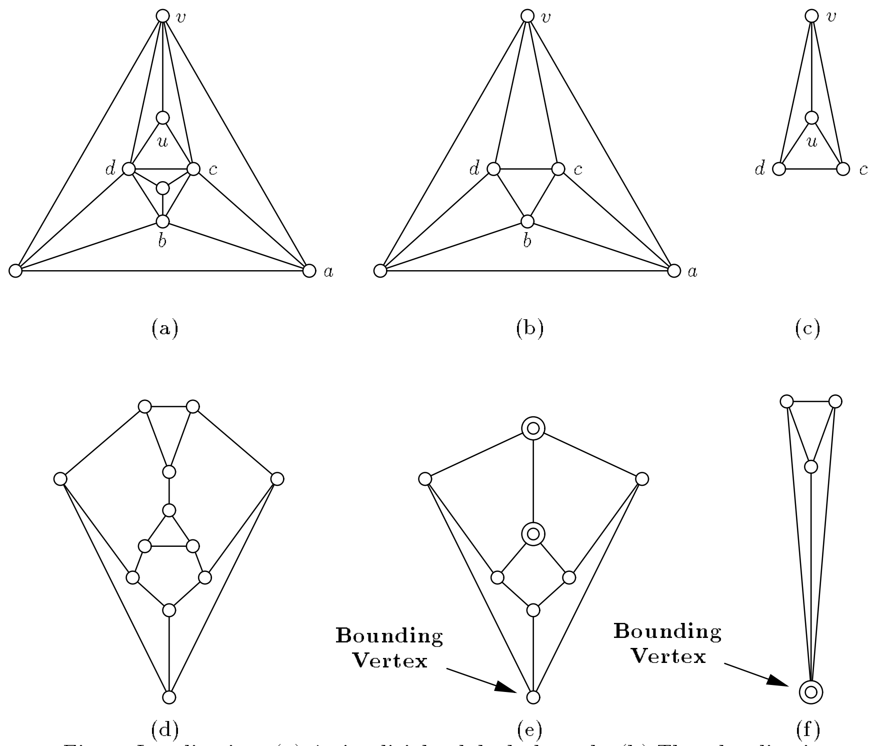


Fig. 2. Localization. (a) A simplicial polyhedral graph. (b) The s-localization of the graph in (a) with respect to the face bounded by triangle abc. (c) The s-localization of the graph in (a) with respect to the face bounded by triangle dcu. (d) The dual of the graph of (a). (e) The t-localization dualizing the s-localization of (b). (f) The t-localization dualizing the s-localization of (c). (In (e) and (f), double circles denote supervertices.)

that a normal vertex in a t-localization of G represents a vertex of G, while a supervertex corresponds to multiple vertices of G collapsed together. If L is a t-localization of G, and L^* is the corresponding s-localization of G^*, then the *bounding vertex* and the *bounding triad* of L are, respectively, the images under the dual map of the bounding face and the bounding triangle of L^*. Examples are shown in Figure 2(d)–(f).

Let L be a t-localization of a trivalent graph G. An *asymmetric 2-coloring* of L is a coloring of the vertices of L, using two colors, that satisfies the following two properties: (1) no two adjacent vertices have the same color (so, in particular, L is bipartite); and (2) all supervertices of L have the same color. A trivalent graph G is *locally non-2-colorable* if none of its t-localizations has an asymmetric 2-coloring.

3.2. The Characterization Theorem

The following theorem provides a purely combinatorial characterization of trivalent inscribable polyhedra.

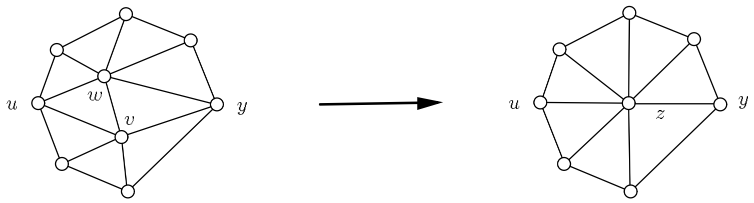


Fig. 3. Illustration of the proof of Lemma 3.2.

Theorem 3.1 *Let G be a trivalent polyhedron. The following are equivalent.*

(a) *G is inscribable.*

(b) *Either G is bipartite and has a 4-connected dual, or G is 1-supertough.*

(c) *Either G is bipartite and has a 4-connected dual, or G is nonbipartite and locally non-2-colorable.*

In both (b) and (c), the two cases are mutually exclusive. We first establish several lemmas about bipartite trivalent polyhedra.

Lemma 3.2 *Let G be a simplicial polyhedral graph. If G has exactly two vertices of odd degree, they cannot be adjacent.*

Proof. Call a simplicial polyhedron an *x-polyhedron* if it has exactly two vertices of odd degree and they are adjacent. We show, by induction on n, that there is no n-vertex x-polyhedron for any value of n. The base case is $n = 4$: the tetrahedron, which is not an x-polyhedron, is the only simplicial polyhedron with 4 vertices.

Let $n > 4$ be given, and assume that there are no x-polyhedra with fewer than n vertices. Let G be an n-vertex x-polyhedron. Let u and v be the two adjacent vertices of odd degree. Let w be a third vertex such that uvw is a face of G.

We first rule out the possibility that there is a vertex x such that vwx is a separating triangle. Suppose there were such a vertex x. Let C' be the component of $G - \{v, w, x\}$ containing u, C'' the other component. Let G' (respectively G'') be the subgraph of G induced by $C' \cup \{v, w, x\}$ (respectively, $C'' \cup \{v, w, x\}$). Let $d'(\cdot)$ and $d''(\cdot)$ denote, respectively, the degree of a vertex in G' and G''. If $d'(x)$ is even, then since $d'(u)$ is odd and all vertices of $G' - \{u, v, w, x\}$ have even degree in G', exactly one of $d'(v)$ and $d'(w)$ is odd. Hence, in this case, G' is an x-polyhedron. If $d'(x)$ is odd, then so is $d''(x)$. But then exactly one of $d''(v)$ and $d''(w)$ is odd, so G'' is an x-polyhedron. In either case, the inductive assumption is contradicted, so vertex x cannot exist.

Since vw is not an edge of any separating triangle, there is a unique vertex y such that $y \neq u$ and vwy is a triangle of G (see Figure 3). Let H be the graph obtained by shrinking edge vw to a vertex z, and connecting all neighbors of either v or w to z. H is a simplicial polyhedron, z and y have odd degree in H, and all other vertices of H have even degree. Thus H is an x-polyhedron with fewer vertices than G. This contradicts the inductive hypothesis, establishing the lemma. $\square$

Lemma 3.3 *Let G be a bipartite trivalent polyhedron. Suppose there is a separating triad in G whose removal separates G into two components, K_1 and K_2. Let H be the graph obtained from G by removing all vertices of K_1 and replacing it with a single supervertex. Then H is bipartite.*

Proof. In its dual form, the lemma says that if G is a simplicial polyhedral graph in which all vertices have even degree, uvw is a separating triangle of G, and H is obtained by removing all vertices "inside" triangle uvw, then all vertices of H have even degree. The preceding statement could fail only if exactly two of u, v, w had odd degree in H and the rest had even degree, which is impossible by Lemma 3.2. □

Lemma 3.4 *Let G be a nonbipartite trivalent polyhedron, v a vertex of G. Then G has a face with odd valence that is not incident on v.*

Proof. Let G be nonbipartite and trivalent. Let k be the number of faces of odd valence. k is positive and even. If $k \geq 4$, the result is obvious, since v is incident with at most 3 faces. If $k = 2$, the result follows from Lemma 3.2. □

Before proving Theorem 3.1, we note the following sufficient condition for inscribability.

Lemma 3.5 *A trivalent polyhedron with a 4-connected dual is inscribable.*

Proof. Let G be a trivalent polyhedron with a 4-connected dual. Assign each edge of G a weight of $1/3$. This weighting satisfies (W1)–(W3) of Theorem 2.1 since, as noted at the beginning of Section 3.1, G cannot have a separating triad. □

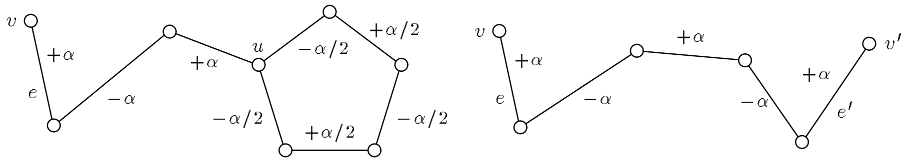
Proof of Theorem 3.1

(a) implies (b)

By Theorem 2.2, it suffices to prove that if G is bipartite and its dual is not 4-connected, then it cannot be inscribable. Suppose G is bipartite and has a separating triad. Let H be the graph obtained by identifying one of the components determined by the triad into a single supervertex, v. By Lemma 3.3, H is bipartite. Color H using two colors, red and blue, and assume that v is colored red. Now suppose that G can be weighted according to Theorem 2.1. Assign each edge of H the weight of the corresponding edge of G. In this edge-weighting of H, the sum of the weights of the edges incident on v is strictly between 1 and $\frac{3}{2}$, while the sum of the weights of the edges incident on any other vertex is 1. If r and b are the respective numbers of red and blue vertices in H, and if w_r and w_s represent the respective sums of the weights of all edges incident on red and blue vertices, then $w_b = b$ and $r < w_r < r + \frac{1}{2}$. Since H is bipartite, $w_b = w_r$, so $r < b < r + \frac{1}{2}$. But r and b are both integers, so this is impossible.

(b) implies (c)

Since the two conditions in (b) are mutually exclusive, it suffices to prove that if G is 1-supertough then G is locally non-2-colorable. We argue by contraposition. Assume G fails to be locally non-2-colorable. Then there is a t-localization H of G that can be 2-colored so that all its supervertices are blue. Since H is trivalent and bipartite, it has a perfect matching, so the blue and red vertex sets must have



(a) (b)

Fig. 4. (a) The odd-cycle construction. (b) The odd-path construction.

equal cardinalities. Removing the vertices in G corresponding to the red vertices of H establishes that G is not 1-supertough.

(c) implies (a)

We assume that G satisfies (c) and construct a weighting satisfying conditions (W1)–(W3). If G has a 4-connected dual, we can apply Lemma 3.5. Hence we may assume, for the remainder of the proof, that G is nonbipartite and locally non-2-colorable.

We will repeatedly use two constructions that permit us to modify edge weights in a controlled fashion. The first construction, the *odd-cycle* construction, is illustrated in Figure 4(a). Let v be a vertex, let e be an edge incident on v, and let α be a real number. Let Z be any odd face not containing v. (By Lemma 3.4, such a cycle Z exists for any choice of v). Let u be any vertex of Z, and let Φ be a simple path from v to u, chosen so (1) the first edge on Φ is e, and (2) Φ contains no vertex of Z other than the endpoint u. Label edges along Φ with $\pm\alpha$, alternating signs and labeling the first edge (e) with $+\alpha$. Label edges of Z with $\pm\alpha/2$, alternating signs, so that the two edges of Z incident on u have opposite sign from the last edge of Φ. Since Z has odd length, this labeling can be done consistently. If the weight of each edge is incremented by its label, then no edge weight changes by more than $|\alpha|$, the weight sums at each vertex of $G - \{v\}$ remain unchanged, the weight of edge e changes by α, and the weights of the other edges incident on v remain unchanged.

The second construction, the *odd-path* construction, is illustrated in Figure 4(b). Let Φ be an odd-length simple path from v to v', with first and last edge e and e'. Alternately label edges along Φ with $\pm\alpha$, starting (and hence ending) with $+\alpha$. If the weight of each edge is incremented by its label, then no edge weight changes by more than $|\alpha|$, the weight sums at each vertex of $G - \{v, v'\}$ remain unchanged, the weight of edge e and e' changes by α, and the weights of the other edges incident on v and v' remain unchanged.

To show that G is inscribable (and thus complete the proof of Theorem 3.1), we assign weights to edges in the t-localizations of G in such a way that the weight-assignments can be combined into a weighting for G with the required properties. We first define the order in which the t-localizations are processed. Then we describe how to assign weights to each t-localization so that the necessary constraints are satisfied and consistency is ensured.

The order in which t-localizations are processed seems more natural when formulated in terms of the dual, G^*. Choose some fixed embedding of G^*. Every s-localization of G^* has, as its bounding face, either the outer face of G^* or the exterior of a separating triangle of G^*. We build a tree of s-localizations as follows. The root is the s-localization having the outer face of G^* as its bounding face. For any other s-localization L, let T be the bounding face. L's father is the s-localization in which the interior of triangle T appears as a face. If we process the s-localizations according to a preorder walk of this tree, we are guaranteed that for any s-localization (other than the first one), the bounding triangle has already appeared as a triangle bounding an interior face of an s-localization previously processed. Moreover, for any s-localization, any edge that is not part of the bounding triangle has not appeared in a previous s-localization.

The preceding order dualizes to an order for processing t-localizations of G. The properties stated above imply that in all t-localizations except for the first one, the bounding vertex is a supervertex, and the bounding triad has already appeared in a previous t-localization. An edge that is not part of the bounding triad of a t-localization has not appeared in a previous t-localization.

It remains to describe how to assign weights. Let $\epsilon < 1/12$ be some fixed positive number. Let n be the number of vertices in G, and let m be the total number of t-localizations of G. Let $H_1, H_2, \ldots, H_m$ be the t-localizations of G, listed in the order described above. Let $\delta_i = (\epsilon/4^{m-i+1})$. Notice that $\delta_m < \epsilon$ and $\max\{\delta_j \,|\, j < i\} \leq \delta_i/4$. We process each H_i by first assigning each edge an initial weight of $1/3$. We then perturb the edge weights of the t-localization, in a manner which we will now describe.

Let b be the bounding vertex of H_i, and let $e_k, k = 1, 2, 3$ be the bounding triad. If $i > 1$, then as noted above, b is a supervertex and the bounding triad has appeared in a previous t-localization, H_j. In this case, let w_k be the weight assigned to e_k in H_j. If $i = 1$, each $w_k = 1/3$. The edge weights in H_i will be perturbed so that each edge e_k is assigned the weight w_k, and so that weight sums at all supervertices exceed 1. We assume, inductively, that for all $j < i$, no edge in H_j has a weight that differs from $1/3$ by more than δ_j.

If H_i is nonbipartite, we choose an odd cycle Z that is not incident on the bounding vertex. This is possible by Lemma 3.4. We first increment each edge e of the bounding triad from $1/3$ to w_k by $w_k - 1/3$, using the odd-cycle construction. (If $i = 1$, this step is unnecessary.) Next, for each supervertex v of H_i that does not lie on Z, we let Φ_v be a shortest path that starts at v, ends at some vertex on Z, and does not contain the bounding vertex (3-connectivity ensures Φ_v exists). We then apply the odd-cycle construction to the walk obtained by concatenating Φ_v and the cycle Z, incrementing the edge incident with v by $\delta_i/(4n)$. Finally, we process the supervertices on Z. For each such supervertex v, we make the following adjustment to the edge weights on Z: we alternately increase and decrease weights on the cycle by $\delta_i/(4n)$, so that both edges incident on v get incremented by $\delta_i/(4n)$.

If H_i is bipartite, assume that H_i has been two-colored red and blue, and that the bounding vertex b of H_i is blue. Because H_i has no asymmetric 2-coloring, there are both red and blue supervertices. We distinguish three subcases:

(i) b is not a supervertex (but some other blue vertex is a supervertex);

(ii) b is a supervertex, and there are other blue supervertices;

(iii) b is the only blue supervertex.

If subcase (ii) occurs, we choose a red supervertex r, and two sets of odd-length paths. The first set of paths consists of three vertex-disjoint paths from b to r (so each edge e_k is on one of the three paths). We apply the odd-path construction to each of these paths to increment the weight of e_k by $w_k - 1/3$. The second set of paths is a set of odd-length paths, each joining a red supervertex to a blue supervertex, such that each supervertex other than b is an endpoint of at least one path, and such that b is not on any of the paths. We apply the odd-path construction to each of these paths, incrementing the weights of the first and last edges by $\delta_i/(4n)$. Subcase (i) is a special case of subcase (ii); we only need the second set of paths. Note that subcase (i) can only occur when $i = 1$.

In subcase (iii), call an edge e_k "bad" if $w_k < 1/3$, "good" otherwise. Define

$$\begin{aligned} s_{\text{good}} &= \sum_{\{k: e_k \text{ is a good edge}\}} (w_k - 1/3), \text{ and} \\ s_{\text{bad}} &= \sum_{\{k: e_k \text{ is a bad edge}\}} (1/3 - w_k). \end{aligned}$$

Since $\sum w_k > 1$, there can be at most two bad edges, and $s_{\text{good}} > s_{\text{bad}}$. Let r be any red supervertex of H_i. For each bad edge e_k, construct a simple path from b to r that uses e_k. Use the odd-path construction to increment the weights of the bad edges e_k by $w_k - 1/3$. (Of course, since e_k is a bad edge, the increment is negative.) At this point, the weight of each bad edge e_k is w_k, but vertex r has weight sum $1 - s_{\text{bad}} < 1$. Now, for each good edge e_k and each red vertex, construct a simple path from b to the red vertex using the good edge. Use the odd-path construction along these paths to make the weight of each good edge e_k equal to w_k (increasing the weights of the good edges by a total of s_{good}), in such a way that the weight sum at each red supervertex (including r) becomes greater than 1. This can be done because $s_{\text{good}} > s_{\text{bad}}$.

After all m t-localizations have been processed, we obtain a well-defined weighting of G, since the weights assigned to edges shared by more than one t-localization are weighted consistently. An inductive argument shows that each edge appearing for the first time in H_i has its weight in the range $[1/3 - \delta_i, 1/3 + \delta_i]$. In particular, since $\delta_i < \epsilon < 1/12$, each weight is in the range $(1/4, 5/12)$, so (W1) holds. Since the weight sums at normal vertices are initially set to 1 and subsequently preserved at each step, (W2) holds. To establish (W3), consider any noncoterminous cutset. If it is a separating triad, then the edges appeared as the edges incident on a supervertex in some t-localization, so the total edge weight exceeds 1. If the cutset has four or more edges, it has total weight exceeding 1 because each edge weight exceeds 1/4. □

4. A linear-time algorithm for determining inscribability of trivalent polyhedra

Theorem 3.1 characterizes inscribable trivalent polyhedra in terms of graph-theoretical properties. In this section, we show that the theorem leads to a linear-time algorithm for determining whether a trivalent polyhedron is inscribable. It is more convenient to describe our algorithm in terms of the dual problem, namely determining whether a simplicial polyhedron is circumscribable. We define an *asymmetric 2-face coloring* of an s-localization of a simplicial polyhedron to be a 2-coloring of the faces such that all superfaces are the same color. A simplicial polyhedron is *locally non-2-face-colorable* if none of its s-localizations can be asymmetrically 2-face-colored. The following theorem follows from dualizing Theorem 3.1 and applying Lemma 3.5.

Theorem 4.1 *Let G be a simplicial polyhedron. The following are equivalent.*

(a) *G is circumscribable.*

(b) *Either G is 4-connected, or the dual of G is nonbipartite and G is locally non-2-face-colorable.*

Now let G be a simplicial polyhedron that we wish to test for circumscribability. If G is 4-connected, then we report that G is circumscribable and stop. If G is not 4-connected and its dual is bipartite, then we report that G is not circumscribable and stop. Since both these tests can be performed in linear time (see (Ref. 15) for a linear-time algorithm for testing simplicial polyhedral graphs for 4-connectedness), it suffices to show that the local non-2-face-colorability of G can be tested in linear time. Notice that it is only necessary for the algorithm to examine those s-localizations that have at least one normal face, as the s-localizations having only superfaces cannot be asymmetrically 2-face-colorable. By marking faces of G when we encounter them (as normal faces of L) during an examination of an s-localization L, we can ensure that we only test each s-localization once.

Let L be any s-localization of G. We can determine whether L is asymmetrically-2-face-colorable in $O(|L|)$ time, using a straightforward depth-first search of the faces of L, provided we can answer two types of queries. First, given any face f of L and an edge e on the boundary of f, we need to determine, in $O(1)$ time, the face of L on the other side of e from f. Second, given a face f, we need to determine in $O(1)$ time whether it is a superface (as opposed to a normal face). If we can answer all these queries in the indicated time bounds, then we can test all s-localizations of G in time proportional to the total number of faces in all s-localizations. Since each triangle of G (whether facial or not) appears at most twice as the boundary of some face in an s-localization of G, and since the total number of triangles in G is $O(|G|)$ (Ref. 15), it then follows that we can test for local non-2-face-colorability in $O(|G|)$ time.

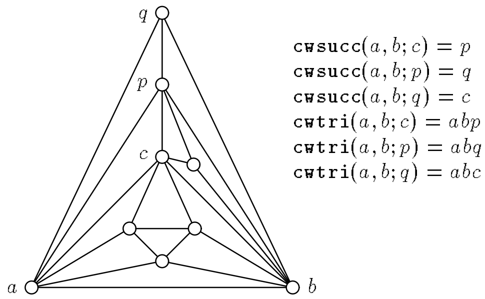


Fig. 5. Examples of `cwsucc`() and `cwtri`() functions.

Let x, y, and z be three vertices of a triangle in G (not necessarily a face). Then $\texttt{cwsucc}(x, y; z)$ is defined to be the vertex w of G such that (i) xyw is a triangle of G, and (ii) w is the first vertex with this property that can be reached by starting at z and following one or more edges, moving clockwise about y through the neighbors of y. $\texttt{cwtri}(x, y; z)$ is defined to be the triangle xyw. Some examples of these functions are shown in Figure 5. Knowledge of the `cwtri`() function permits efficient navigation through an s-localization of G. This notion is formalized in Proposition 4.3, below. We first establish a preliminary lemma.

Lemma 4.2 *Let G be a plane graph. Let $T = xyt$, $U = xyu$, and $V = xyv$ be three distinct triangles sharing a common edge xy. Assume that G is embedded so that a clockwise traversal (about x) of the neighbors of x from u to v does not contain y (see Figure 6). Then T separates U from V if and only if the clockwise traversal (about x) from u to v contains t.*

Proof. If t is not on the traversal, then the traversal represents a path from u to v in G that does not contain x, y, or t, so T cannot separate U from V.

For the converse, suppose t is on the traversal. Suppose there is a path Π from u to v that does not contain x, y, or t. Then Π, the portions of the traversal from u to t and from t to v, and the seven edges xy, xt, yt, xu, yu, xv, and yv are 10 paths which are pairwise vertex-disjoint except at their endpoints. But this means that G has a K_5 minor, and hence is not planar. □

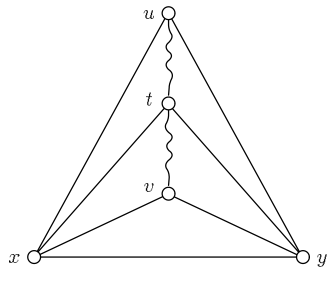


Fig. 6. Lemma 4.2 and its proof.

Proposition 4.3 *Let G be a simplicial graph, f a face. Let h be a face in the s-localization of G with respect to f, and let xyz be the boundary of h, with h to the left of the directed edge xy. Then the triangle* $\texttt{cwtri}(x, y; z)$ *is the face of the s-localization of G with respect to f lying to the right of edge xy.*

Proof. Let $p = \texttt{cwsucc}(x, y; z)$, so $xyp = \texttt{cwtri}(x, y; z)$. By Lemma 4.2, triangle xyp is not separated from face h by any triangle. Since h is in the s-localization of G with respect to f, h and f are not separated by any triangle. Hence triangle xyz and face f are not separated by any triangle. □

We precompute all values of $\texttt{cwtri}()$ and store them. This preprocessing is described below. The TRIANGLE structure contains one entry for each triangle of G. Each entry consists of the three vertices of the triangle (say x, y, z); a flag indicating whether the triangle is a nonfacial triangle of G; pointers to the three edges xy, yz, and xz; and pointers to the six triangles $\texttt{cwtri}(x, y; z)$, $\texttt{cwtri}(y, x; z)$, $\texttt{cwtri}(x, z; y)$, $\texttt{cwtri}(z, x; y)$, $\texttt{cwtri}(y, z; x)$, and $\texttt{cwtri}(z, y; x)$.

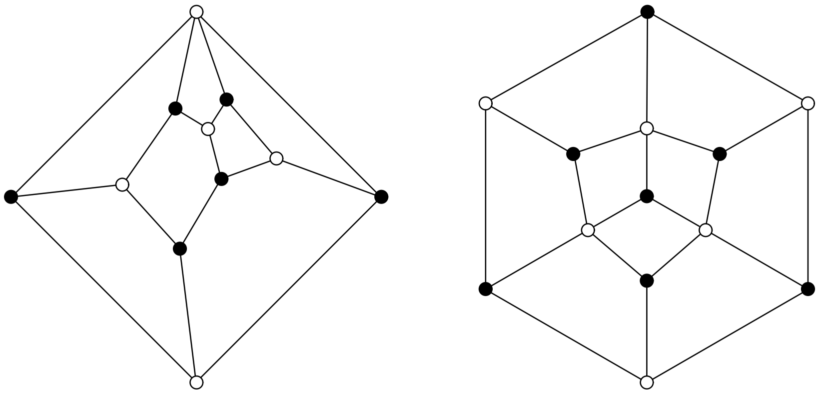
The TRIANGLE data structure allows us to answer the two queries discussed above in $O(1)$ time per query. Indeed, to find the next face in the s-localization currently being computed, Proposition 4.3 says that we need only find the appropriate $\texttt{cwtri}()$ value. We can do this in $O(1)$ time, provided we keep track of our position in the TRIANGLE structure as we visit faces of the s-localization in the depth-first search. We can tell whether a triangle represents a superface face by checking the nonfacial triangle flag.

It remains to show how to build the TRIANGLE structure in linear time. We first number the edges incident on each vertex consecutively according to a clockwise traversal about that vertex, starting from an arbitrary edge. Call the number associated with each edge its *index* with respect to that vertex. Next, we enumerate all triangles of G in linear time, using the algorithm in (Ref. 15, page 141). As we enumerate each triangle, we place it on three lists, one for each of its edges. We also set its vertex fields, its edge fields, and its nonfacial triangle flag at this time. Setting the nonfacial triangle flag requires a minor, but straightforward, modification of the algorithm in (Ref. 15).

After all triangles are enumerated, we process each list of triangles incident on an edge as follows. Let ab be an edge, with $d(a) \leq d(b)$. Assign each triangle abc having ab as an edge the position of c in the list of neighbors of a, clockwise about a, starting with b. This position can be computed in $O(1)$ time from the indices of ab and ac with respect to a. The triangles can then be sorted by these position numbers in $O(d(a))$ time, using bucket sort. Once this sorted list has been computed, it is easy to set all $\texttt{cwtri}(a, b; c)$ and $\texttt{cwtri}(b, a; c)$ fields in $O(d(a))$ time. The total time required to do all this processing for all edges is

$$O(n + \sum_{(u,v) \in G} \min(d(u), d(v))) \tag{4.1}$$

where n is the number of vertices in G. Since the sum in (4.1) is $O(n)$ for a planar graph (Ref. 15, Lemmas 8.1 and 8.2), the total preprocessing required is linear.



(a) (b)

Fig. 7. (a) A graph that is nonbipartite and not 1-supertough (and hence noninscribable), but has a 4-connected dual. (b) A graph that is bipartite and not 1-tough (and hence noninscribable), but has a 4-connected dual.

5. Some Counterexamples in the Nontrivalent Case

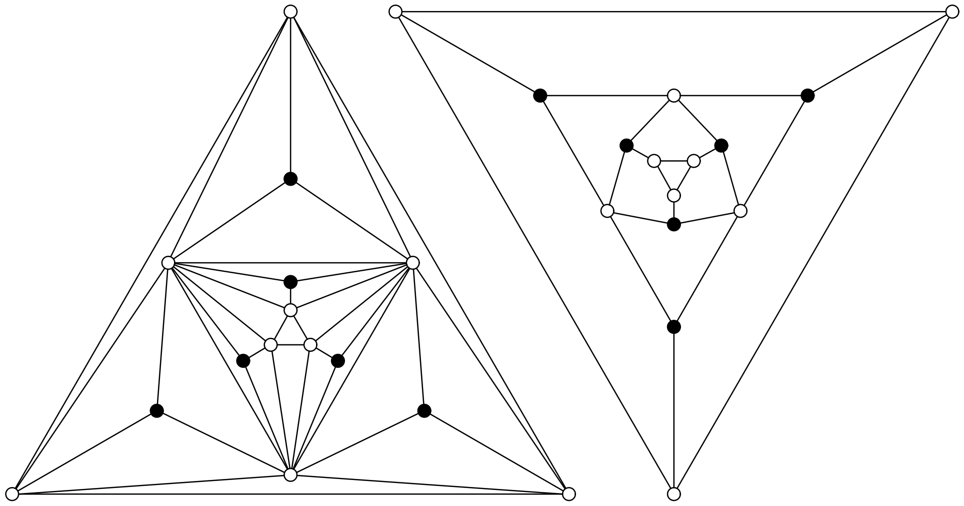
In this section, we present several examples to show that the results of Section 3 do not remain true if the assumption of trivalency is removed.

In the course of proving Theorem 3.1, we noted that any trivalent polyhedron with a 4-connected dual is inscribable (Lemma 3.5). This statement becomes false if the word "trivalent" is removed. Figure 7(a) shows a polyhedral graph that has a 4-connected dual. This graph is nonbipartite and fails to be 1-supertough (to see this, remove the five white vertices), so it is noninscribable by Theorem 2.2. The graph of Figure 7(b) is bipartite and has a 4-connected dual. However, it fails to be 1-tough (to see this, remove the six white vertices), and hence it is noninscribable. By taking the duals of these two examples, one obtains (nonsimplicial) polyhedral graphs that are 4-connected but not circumscribable.

Figure 8 shows two nontrivalent polyhedral graphs that are nonbipartite and 1-supertough, but not inscribable. In both Figure 8(a) and Figure 8(b), the noninscribability can be seen as follows. Suppose a weighting satisfying (W1)–(W3) exists. Let W be the total weight of the black-white edges. (W2) and the independence of the six black vertices imply that $W = 6$. But there are three white vertices such that all but six of the black-white edges are incident on these three vertices. Hence (W1) and (W2) imply that $W < 6$, a contradiction.

6. Final Remarks

In Sections 3 and 4, we developed a characterization theorem for trivalent inscribable graphs, and a linear-time algorithm for recognizing them. The linear-time algorithm is efficient in practice as well as in theory. On a Sun SparcStation, the algorithm takes about 25 seconds to examine all 7,595 12-face trivalent polyhedra, 3 minutes to examine all 49,566 13-face trivalent polyhedra, and about 22.5 minutes to examine all 339,722 14-face trivalent polyhedra. So the cost is somewhat less than 300 microseconds/face (or, equivalently, 300 microseconds/vertex to test



(a) (b)

Fig. 8. Two 1-supertough graphs that are not inscribable.

the circumscribability of simplicial polyhedra). More details on the results of this enumeration can be found in (Ref. 18).

An outstanding open problem is the problem of finding a simple, purely combinatorial characterization of inscribable polyhedra, analogous to Theorem 3.1, in the nontrivalent case. The simplicial case is of particular importance because of the connection with Delaunay triangulations. The examples of Section 5 illustrate some of the difficulties that arise when attempting to extend Theorem 3.1 to these cases.

Some partial results have been recently obtained by the authors. In particular, all 4-connected polyhedra are inscribable, as are all simplicial polyhedra with all vertex degrees between 4 and 6, inclusive[19]. The authors have also obtained an exact characterization of the conditions under which adding an edge to an inscribable polyhedron preserves inscribability[20]. One consequence of this last result is a simple method for resolving degeneracies in Delaunay triangulations that does not require symbolic perturbation of the data.

Acknowledgments

The first author gratefully acknowledges support from the University of California, Irvine through an allocation of computer resources and a UCI Faculty Research Grant.

References

1. P. J. Federico. *Descartes on Polyhedra: A study of the De Solidorum Elementis*, volume 4 of *Sources in the History of Mathematics and Physical Sciences.* Springer-Verlag, New York, NY, 1982.

2. J. Steiner. *Systematische Entwicklung der Abhängigkeit geometrischer Gestalten von einander.* Reimer, Berlin, 1832. Appeared in Jakob Steiner's Collected Works, 1881.
3. H. T. Croft, H. J. Falconer, and R. K. Guy. *Unsolved Problems in Geometry.* Springer-Verlag, New York, NY, 1991.
4. B. Grünbaum. *Convex Polytopes.* Wiley Interscience, New York, NY, 1967.
5. B. Grünbaum and G. C. Shephard. Some problems on polyhedra. *Journal of Geometry*, 29(2):182–190, August 1987.
6. K. Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5):223–228, December 1979.
7. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Berlin, 1987.
8. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.
9. I. Rivin. A characterization of ideal polyhedra in hyperbolic 3-space. Preprint, 1992.
10. C. D. Hodgson, I. Rivin, and W. D. Smith. A characterization of convex hyperbolic polyhedra and of convex polyhedra inscribed in the sphere. *Bulletin of the AMS*, 27(2):246–251, October 1992. See also Erratum, *Bulletin of the AMS*, 28(1):213, January 1993.
11. I. Rivin. On geometry of convex ideal polyhedra in hyperbolic 3-space. *Topology*, 32(1):87–92, January 1993.
12. I. Rivin and W. D. Smith. Inscribable graphs. Manuscript, NEC Research Institute, Princeton, NJ, 1991.
13. C. D. Hodgson and I. Rivin. A characterization of compact convex polyhedra in hyperbolic 3-space. *Inventiones Mathematicae*, 111(1):77–111, January 1993.
14. W. D. Smith. A polynomial-time algorithm for deciding the inscribability of graphs. Manuscript, NEC Research Institute, Princeton, NJ, 1992.
15. T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms*, volume 32 of *Annals of Discrete Mathematics.* North-Holland, Amsterdam, 1988.
16. V. Chvátal. Tough graphs and Hamiltonian circuits. *Discrete Mathematics*, 5(3):215–228, July 1973.
17. M. B. Dillencourt. Toughness and Delaunay triangulations. *Discrete & Computational Geometry*, 5(6):575–601, 1990.
18. M. B. Dillencourt. Polyhedra of small order and their Hamiltonian properties. Information and Computer Science Technical Report 92-91, University of California, Irvine, CA, September 1992.
19. M. B. Dillencourt and W. D. Smith. Graph-theoretical conditions for inscribability and Delaunay realizability. Information and Computer Science Technical Report 92-90, University of California, Irvine, CA, August 1992.
20. M. B. Dillencourt and W. D. Smith. A simple method for resolving degeneracies in Delaunay triangulations. In *Automata, Languages, and Programming: Proceedings of the 20th International Colloquium*, pages 177–188, Lund, Sweden, July 1993. Springer-Verlag Lecture Notes in Computer Science 700.