Chapter 6

Planar Orientations

In this chapter we will focus on algorithms and techniques used for drawing planar graphs. The algorithms we will use are based on numbering the vertices and orienting the edges from the lower numbered vertices to high numbered vertices and they will help us to construct a kind of geometric representation of a planar graph, called tessellation representation.

We will first examine the method of numbering the vertices of a digraph, and focus especially on the st-numbering algorithm. We will then explore the properties of planar acyclic graphs. Understanding these properties is essential for understanding how tessellation representations work. At the end of this chapter we will provide the reader with the techniques used to construct a tessellation representation of a planar acyclic graph.

6.1 Numberings of Digraphs

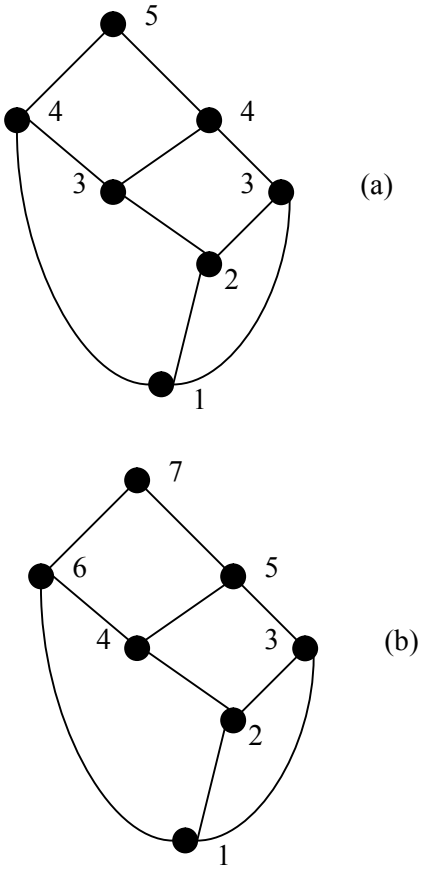

Figure 6.1**:** (a) Topological numbering of *G*; (b) Topological sorting of *G*

To be able to draw a planar graph, we must first number its vertices. This numbering of course is not arbitrary. Our goal is to number the vertices in such a way that we can have a sorting of the vertices, starting from the source vertex *s* and ending to the sink vertex *t.*

Let *G* be a digraph with *n* vertices and *m* edges. If we assign numbers to the vertices of *G* such that for every edge (*u, v*) the number assigned to *v* is greater than the number assigned to *u* (i.e. *number*(*v*) 〉 *number*(*u*)) then we have a *topological numbering* of *G.* If in a topological numbering of *G* we assign each vertex with a distinct integer between 1 and *n,* we will have a *topological sorting* of *G.* If *G* has a unique path visiting all its vertices, the topological sorting is unique (see Figure 6.1). The following statements are equivalent:

G is acyclic $\leftrightarrow$ *G* admits a topological numbering $\leftrightarrow$ *G* admits a topological numbering (see Figure 6.1)

We can apply a topological numbering to a graph *G* even if *G* has weights on its edges, provided that these weights are non negative. That is called a *weighted topological numbering* and is actually a topological numbering such that for every edge (*u, v*) of *G,* we assign a number to *v* that is greater than equal than the number we assigned to *u* plus the weight of (*u, v*) (i.e. *number*(*v*) ≥ *number*(*u*) + *weight*(*u, v*)). If we minimize the range of numbers assigned to the vertices, then the numbering is *optimal.*

6.1.1 st–Numbering

As we mentioned above, there are many different ways and techniques to number the vertices of a planar graph. In this section, we will focus on st-numbering and provide an algorithm to compute it.

Let *G*(*V, E*) be a non-separable graph. Given any edge *s* – *t* of *G,* a 1-1 function *g* such that g: V $\rightarrow \{1,2,3....|V|\}$, is called an *st-numbering* if all of the following are true:

(1) $g(s) = 1$
(2) $g(t) = |V|$ (=*n*)
(3) $\forall v \in V - \{s,t\}$ there are adjacent vertices *u* and *w*, such that $g(u)\langle g(v)\langle g(w)$

Although the definition of an st-numbering is quite simple, the algorithm that performs an st-numbering on a planar graph, may be a little confusing. Below, we provide an algorithm that performs an st-numbering in linear $O(n+m)$ time.

St-numbering algorithm
Input: A graph $G = (V, E)$, the source vertex $s \in V$ and the sink vertex $t \in V$
Output: An st-numbering for all the vertices of graph *G.*

St-numbering(*G, s, t*)

```
1.    do DFS(G, t) and ∀v ∈ V compute dfnumber(v), lowpoint(v), parent(v)
2.    i ← 1
3.    PUSH(s, STACK)
4.    PUSH(t, STACK)
5.    v ← POP(STACK)
6.    if (v = t) then
7.        g(v) ← i
8.        return
9.    else find-path(G, v)
```

10. **if** (path is empty) **then**
11. $g(v) \leftarrow i$
12. $i++$
13. **go to step 4**
14. **else let the path be** $v, v_1, v_2, ...v_{i-1}, v_i$
15. PUSH($v_i, v_{i-1}, ...v_1, v$, STACK)
16. **go to step 4**

In step 1 of the above algorithm we perform a DFS on G and while we do that we compute the $dfnumber(v)$ (the number the DFS algorithm assigns to each vertex $v \in V$), the *lowpoint*(v) (the least *dfnumber* of the vertices adjacent to v), and *parent*(v) (the parent of vertex v). Also, in step 8 of the above algorithm we use the *path finding algorithm*. This algorithm starts from a given vertex v and finds a path from it. This path may be directed from v or into v.

Path finding algorithm
Input: A graph $G = (V, E)$ and a vertex $v \in V$
Output: A path directed from v or into v.
Initialization: Mark vertices s, t and the edge connecting them as *old* and all the other vertices as *new*

Find-path(G, v)

1. **if** there is a *new* edge e ($v \xrightarrow{e} w$) **where** $dfnumber(w) \langle dfnumber(v)$ then
2. **Mark** $e \leftarrow old$
3. The path is $v \xrightarrow{e} w$
4. **return**
5. **if** there is a *new* edge e ($v \xrightarrow{e} w$) **where** $dfnumber(w) \rangle dfnumber(v)$ **then**
6. trace a path whose first edge is e and from there it follows a path which defined *lowpoint*(w), i.e. it goes up the tree and ends with a back edge to a vertex u such that $dfnumber(u) = lowpoint(w)$
7. **Mark** all vertices in the path as *old*
8. **return**
9. **if** there is a *new* edge e ($w \xrightarrow{e} v$) **where** $dfnumber(w) \rangle dfnumber(v)$ **then**
10. start a path with e and continue going backwards following tree edges until you find a vertex marked as *old*
11. **Mark** all vertices in the path as *old*
12. **return**
13. **if** all vertices incident to v are marked as *old* **then**
14. The path is empty
15. **return**

If we examine the st-numbering algorithm as a whole, we can see that we insert each node only once in the stack, and we visit each edge only one time. Thus, the algorithm performs its task in linear $O(n+m)$ time. An example of an st-numbering of a graph is shown in figure 6.2

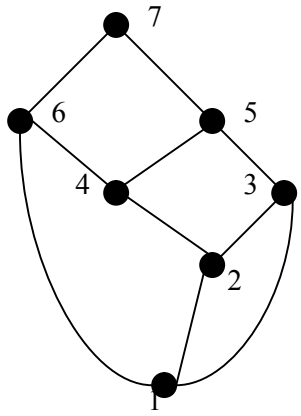

Figure 6.2: An st-numbering of graph *G*

6.2 Properties of Planar Acyclic Digraphs

We define an *st-graph*, as a planar acyclic digraph with one source vertex *s* and one single sink vertex *t*. If we apply a topological numbering on an st-graph *G*, we can see that the way the vertices are numbered, give a sense of direction, from a vertex with a low number to a vertex with a higher number, to the edges. Thus we can clearly state that:

- Given a topological numbering of an st-graph *G*, each directed path of *G* visits vertices with increasing numbers.

Given the way we defined an st-graph, we can also state that:

- For every vertex *v* of an st-graph *G*, there exists at least one directed path *P* from *s* to *t* that contains *v*

It is easy to see why this is true: if it wasn't, there would either be no path from *s* to *v*, or from *v* to *t*, thus *s* or *t* wouldn't be the source or tank vertices respectively.

An embedding of an st-graph that is planar where we position the source vertex *s* and the sink vertex *t* on the boundary of the external face is called a *planar st-graph.* We draw a planar st-graph by positioning *s* at the bottom and *t* at the top.

If *G* is a planar st-graph, and *F* its set of faces, then we will assume that the external face of *G* is divided in two "faces", the "left external face" *s**, that is incident with the edges on the left boundary of *G* and the "right external face" *t** that is incident with the edges on the right boundary of *G*. For each edge *e* between two vertices *u* and *v* ($u \xrightarrow{e} v$), we define $orig(e) = u$ and $dest(e) = v$. We also call the face to the left of e as *left*(*e*) and the face to the right of *e* as *right*(*e*) (see Figure 6.3).

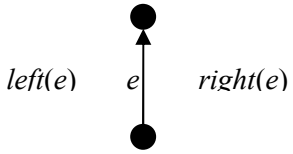

Figure 6.3 : The left (*left*(*e*)) and right (*right*(*e*)) faces of an edge *e*

The planar graph *G** of the planar st-graph *G* is defined as follows:

- The set of faces of *G* represent the set of vertices of *G**.

- For every edge e of G (where $e \neq (s,t)$), there is an edge $e^* = (f,g)$ in G^* where $f = left(e)$ and $g = right(e)$.

G^* is actually the dual graph of G, if in the dual graph of G we "break" the vertex associated with the external face into two different vertices, and associate one of them to the left external face that inherits the outgoing edges, and the other to the right external face that inherits the incoming edges (see Figure 6.4). If we rotate G^* 90 degrees, we can see that G^* is also a planar st-graph.

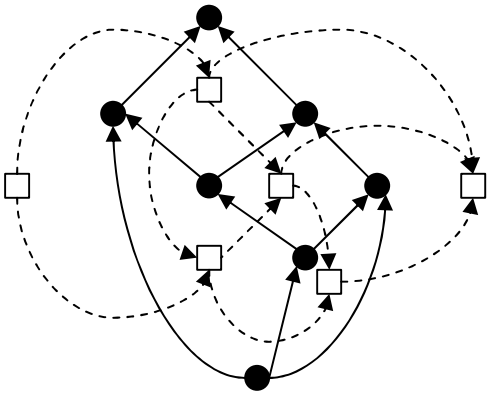

Figure 6.4: Graph G drawn with solid lines and G^* associated to it drawn with dashed lines

Given a vertex v of a planar st-graph, the face separating the incoming from the outgoing edges in the clockwise direction is called *left*(v), and the other separating face is called *right*(v) (see Figure 6.5).

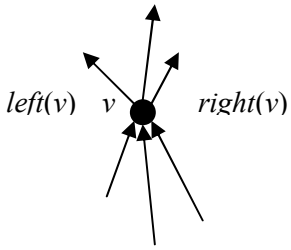

Figure 6.5**:** The left (*left*(v)) and right (*right*(v)) faces of a vertex v

Lemma 6.1 *Each face of a planar st-graph G, consists of two directed paths with common origin, called* $orig(f)$*, and common destination, called* $dest(f)$.

Proof: Lets suppose that the lemma is not true for a face f of the graph. Then there should be two vertices w and u on the boundary edges of f, and they create a path directed from *dest*(f) to *orig*(f). Then, there should also be a directed path P_1 from s to w and a path P_2 from u to t. But these two paths must intersect, and since G is planar, there should be a vertex x at their intersection point. But then, we can clearly see that G has a cycle, between vertices w, u and x which contradicts the fact that G is a planar st-graph (see Figure 6.6).

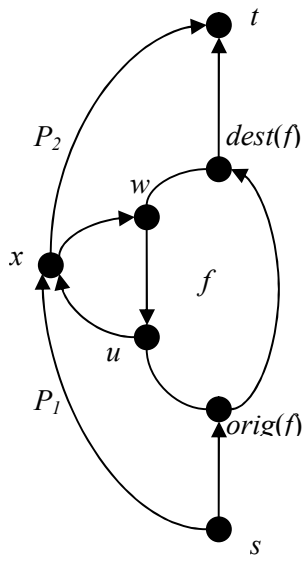

Figure 6.6: Graph *G* has a cycle between *x*, *w* and *u*

Lemma 6.2 *The incoming edges for each vertex v of a graph G appear consecutively around v, and so do the outgoing edges.*

Proof: The lemma is true for vertices *s* and *t.* Let's suppose that the lemma is not true for a vertex *v* of *G.* This means that there are edges $(v,a),(b,v),(v,c),(d,v)$, as they appear in Figure 6.7. Then there should be a directed path P_1 from *s* to *b*, and a directed path P_2 from *c* to *t*. But these two paths intersect, and since *G* is planar, there should be a vertex *x* at their intersection point. But then a cycle between vertices *v*, *c*, *x* and *b* is created, contradicting to the fact that *G* is a planar st-graph.

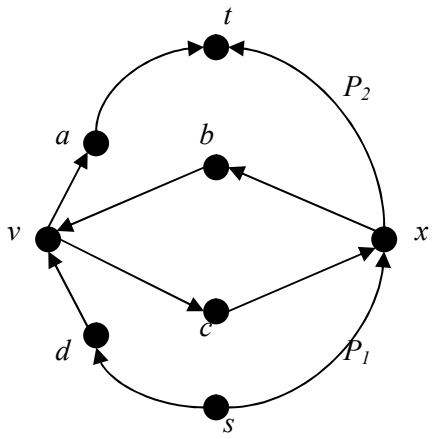

Figure 6.7: *G* has a cycle between *v*, *c*, *x* and *b*

Lemma 6.3 *For every two faces f and g of a planar st-graph, exactly one of the following holds:*

- *G has a directed path from dest(f) to orig(g)*
- *G has a directed path from dest(g) to orig(f)*
- *G* has a directed path from f to g*
- *G* has a directed path from g to f*

Proof: Before we prove lemma 6.3, we must give two more definitions that we will use in this proof. We call the *leftmost path* from a vertex *u* the path that always takes the leftmost outgoing edge. Similarly, the *rightmost path* of a vertex *u* is the path that always takes the rightmost edge. Now let's assume that *G* is topologically sorted and that the number of *dest*(*f*) is less than the number of *orig*(*g*). The rightmost and leftmost paths of from *dest*(*f*) to *t* are called P_1 and P_2 respectively. Similarly, the leftmost and rightmost paths from *orig*(*g*) are called P_3 and P_4 respectively. The lemma is obviously true if there is a directed path from *dest*(*f*) to *orig*(*g*). Otherwise P_2 and P_3 or P_1 and P_4 intersect. Let's assume that P_2 intersectsP_3 at a vertex *x*. Then from lemma 6.2, every edge incident to every vertex on P_2, from the right, is incoming, as it also happens with every edge incident to P_3 from the left. If we construct *G*,* we will see that there is a directed path in *G** from *f* to *g*. (see Figure 6.8)

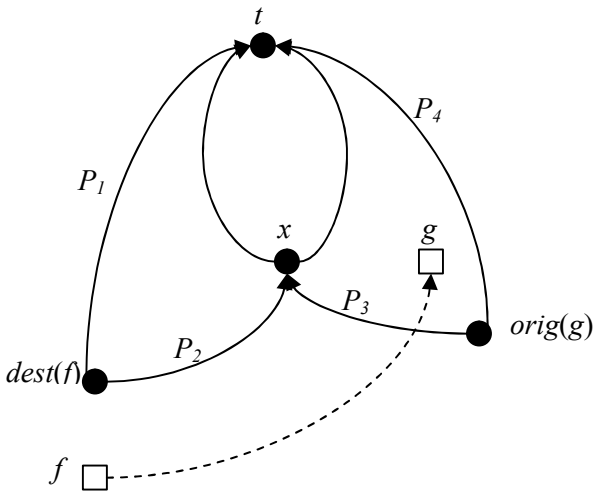

Figure 6.8: A directed path in *G** from face *f* to *g*

In the above lemma we deal with a special case of a more general property of planar st-graphs. We can make the abstraction and call an element of the set $V \cup E \cup F$ of a planar st-graph *G* an object. The definitions of $orig(\bullet), dest(\bullet), left(\bullet), right(\bullet)$ can then be extended as follows. For a vertex *v* we define $orig(v) = dest(v) = v$ and for a face *f* we define $left(f) = right(f) = f$. Lemma 6.3, can then be generalized as follows:

Lemma 6.4 *For any two objects* o_1 *and* o_2 *of a planar st-graph G, exactly one of the following holds:*

- *G has a directed path from* $dest(o_1)$ *to* $orig(o_2)$
- *G has a directed path from* $dest(o_2)$ *to* $orig(o_1)$
- *G* has a directed path from* $right(o_1)$ *to* $left(o_2)$
- *G* has a directed path from* $right(o_2)$ *to* $left(o_1)$

6.3 Tessellation Representation

6.3.1 Plane Tessellation Representation

A tessellation representation on the plane for a planar graph G is a partition of the plane into disjoint tiles, each associated with vertex, edge or face of G, such that the topological incidencies correspond to geometric adjacencies between tiles.

A tile is a rectangle with sides parallel to the coordinate axes. A tile can be unbounded or can degenerate to a segment or a point. Two tiles are horizontally or vertically adjacent if they share a portion of a vertical or horizontal side. The coordinates of a tile *?* will be denoted by $x_L(q)$, $x_R(q)$, $y_B(q)$, $y_T(q)$.

Let G be a planar st-graph. As usual, we denote the sets of vertices, edges and faces of G by V, E and F, respectively. A tessellation representation T for G maps each object (vertex, edge, or face) o of G into a tile $T(o)$, such that:

- The interiors of tiles $T(?_1)$ and $T(?_2)$ are disjoint whenever $?_1 \neq ?_2$.
- The union of all tiles T(o), $? \in V \cup E \cup F$ is a rectangle.
- Tiles $T(?_1)$ and $T(?_2)$ are horizontally adjacent if and only if

$$o_1 = left(o_2) \text{ or } o_1 = right(o_2) \text{ or } o_2 = left(o_2) \text{ or } o_2 = right(o_1)$$

- Tiles $T(?_1)$ and $T(?_2)$ are vertically adjacent if and only if

$$o_1 = orig(o_2) \text{ or } o_1 = dest(o_2) \text{ or } o_2 = orig(o_2) \text{ or } o_2 = dest(o_1)$$

The algorithm bellow constructs a tessellation representation Θ for a planar st-graph G.

Plane tessellation representation algorithm
Input: A planar st-graph G
Output: A tessellation representation T for G such that each vertex- and face-tile is a segment

Plane tessellation representation (G)

1. Construct planar st-graph G^*.
2. Compute a topological numbering Y of G.
3. Compute a topological numbering X of G^*.
4. For each object $? \in V \cup E \cup F$, let the coordinates of tile T(?) be

$$x_L(o) = X(left(o));$$
$$x_R(o) = X(right(o));$$
$$y_B(o) = Y(orig(o));$$
$$y_T(o) = Y(dest(o)).$$

Theorem 6.1 *Let G be a planar st-graph with n vertices. The plane tessellation representation algorithm constructs a tessellation representation of G in* O(n) *time.*

Proof: The tiles of any two distinct objects are separated either by a vertical or by horizontal line, according to Lemma 6.4. Each step of the algorithm takes linear time, so the above algorithm constructs the plane tessellation representation in O(n) time.

An example of a run of the plane tessellation representation algorithm is shown below in Figure 6.9.

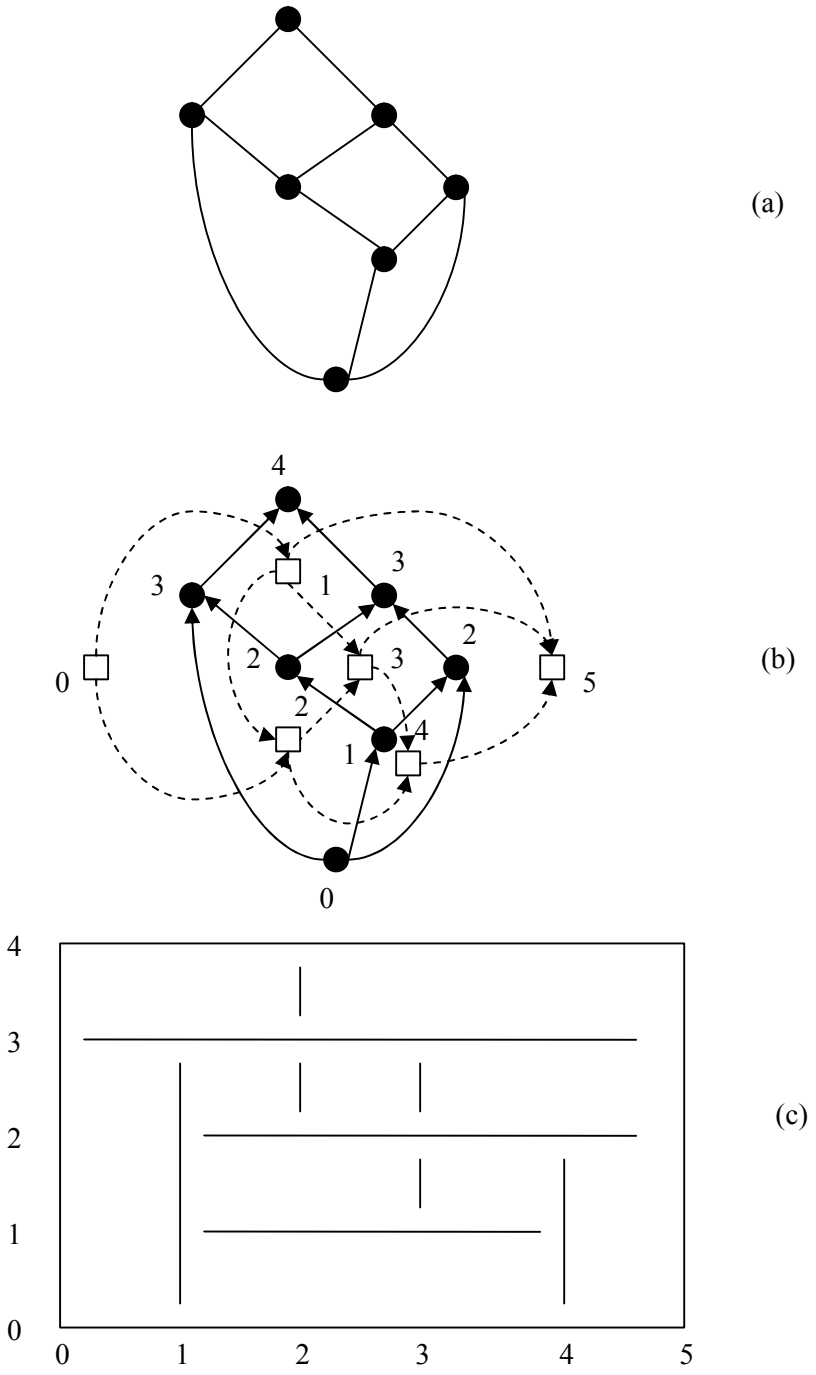

Figure 6.9: Example of a run of the plane tessellation representation algorithm: (a) a planar graph G; (b) planar st-graphs G and G^* labelled by topological numberings Y and X, respectively; (c) tessellation representation T of G constructed by the plane tessellation representation algorithm

Notice that the above algorithm constructs a tessellation representation on the plane, in which the tiles associated with the faces and the vertices (except s and t) are degenerate. In particular, the external face is associated with the *a* tile at infinity. It is possible to modify the construction so that only one tile is degenerate, namely the one associated with the external face.

We can modify the plane tessellation representation algorithm to support user-defined constraints on the size of the edge-tiles. Namely, let *h*(e) and *w*(e) be non-negative numbers associated with each edge *e* of *G*. By replacing the first two steps of the plane tessellation representation algorithm with the following ones, we obtain a tessellation representation of *G*, such that the tile of each edge *e* has height at least *h*(e) and width at least *w*(e):

1. Assign weight *h*(e) to each edge *e* of *G* and compute an optimal weighted topological numbering *Y* of *G*.

2. Assign weight *w*(e) to each edge e^* of G^* and compute an optimal weighted topological numbering *X* of G^*

Plane tessellation representation algorithm can also be further modified to support user-defined constraints on the size of the vertex- and face tiles. Namely, we construct from G a new planar st-graph G' as follows:

- Let $G' = G$.

- For each vertex v of G', we expand v into vertices v' and v'', joined by an edge e_u from v' to v'', such that v' contains the incoming edges of v and v'' contains the outgoing edges of v.

- For each face *f* of G', we add an edge e_f into face *f* from *orig*(*f*) to *dest*(*f*).

Every object of *G* is associated with an edge of G'. We then simply apply Plane tessellation representation algorithm to G' and represent each object of *G* with the tile of the associated edge of G'.

Theorem 6.2 *Given a planar st-graph G with n vertices and nonnegative numbers h*(o) *and w*(o) *for each object o of G, a minimum-area tessellation representation T for G, such that each tile T*(o) *has height at least h*(o) *and width at least w*(o) *can be constructed in time* O(n). *In particular, if h*(o) = *w*(o) = 1 *for each object o of G, then T has integer coordinates and area* O(n^2).

Proof: We construct a plane tessellation representation such that each tile *T*(*?*) has height and weight, by expanding each vertex into two new vertices. The only essential difference between the initial *G* graph and *G?* graph obtained by the changes we apply, is that the second one has more objects. So, according to theorem 6.1, we can construct a tessellation representation on the plane in O(n) time for a *G* graph. So, according to theorem 6.1 the above algorithm will take O(n) time. If *h*(*o*) = *w*(*o*) = 1, then the objects *o* of *G* graph will have integer dimensions. So *T* will have integer coordinates. Also *T* will have area O(n^2), because vertices and faces will be represented as rectangles and not as lines.

6.3.2 Sphere Tessellation Representation

A sphere S is the locus of points at the same distance from a point, called the centre of the sphere. The intersection of S with the horizontal plane that passes through the centre of S defines a circle, called the equator. Similarly, the intersection of S with planes parallel to the plane of the equator defines the parallels. The line that passes through the centre of S and is orthogonal to the plane of the equator, called the axis of the sphere, intersects the sphere into two points, the North Pole and the South Pole. Every plane that is orthogonal to the plane of the equator and passes through the two Poles defines a circle called a meridian. Every point p of S will be denoted by a pair (x, y) where x is the latitude measured with respect to the South Pole, and y is the longitude measured with respect to a reference meridian. The notion of horizontal and vertical is extended to the sphere by considering horizontal the parallels and vertical the meridians.

A spherical st-graph is an embedded planar acyclic digraph with exactly one source s and exactly one sink t. It is convenient to visualize a spherical st-graph as drawn on a sphere with s at the "South Pole" and t at the "North Pole". A tile on the sphere is the portion of the sphere delimited by two parallels and two meridians. On the sphere, we allow tiles containing one or both Poles.

Now we consider a spherical st-graph G. Let p be a path in G from s to t, p does not contain its extreme vertices s and t. We construct from G a new planar st-graph G_p by "cutting" G along path p and duplicating the vertices and edges of p. Note that the graph G_p is a planar st-graph. The two copies of p, denoted p' and p'' are the left and the right boundary of G_p, respectively. Also, we denote by o' and o'' the two copies of a vertex or edge o of p in p' and p'', respectively. For any other object o of G which is not in p, both o' and o'' denote the unique object of G_p associated with o.

The following algorithm constructs a tessellation representation on the sphere for a spherical st-graph G.

Sphere tessellation representation algorithm
Input: A spherical st-graph G
Output: A tessellation representation T for G_p on the plane such that each object o of p the tiles $\Theta(o')$ and $\Theta(o'')$ have the same y-coordinates.

Sphere tessellation representation (G)

1. Construct planar st-graph G_p.
2. Construct planar st-graph G_p*.
3. Compute a topological numbering Y of G_p.
4. Compute a topological numbering X of G_p*.
5. Set $x_0 = p / ?(t)$ and $y_0 = 2p / ?(t)$
6. For each object $? \in V \cup E \cup F$, let the coordinates of tile T(?) be

$$x_L(o) = Y(origin(o))x_0;$$
$$x_R(o) = Y(dest(o))x_0;$$
$$y_B(o) = X(left(o))y_0;$$
$$y_T(o) = X(right(o))y_0.$$

The above algorithm constructs a tessellation representation, in which the tiles associated with the faces and the vertices are degenerate. It is possible to modify the construction, so that no tile is degenerate.

Theorem 6.3 *Let G be a spherical st-graph with n vertices. The sphere tessellation representation algorithm constructs a tessellation representation of G on the sphere in* O(n) *time.*

Proof: The sphere tessellation representation, essentially, constructs a plane tessellation representation on the plane for G_p, with the property that for each object o of p tiles $T(o')$ and $T(o'')$ have the same y-coordinates. So, according to Theorem 6.1, the sphere tessellation representation algorithm constructs a tessellation representation on the sphere in O(n) time.

Let G be a planar undirected graph. Graph G is said to be spherically st-orientable if it can be oriented and embedded so that the resulting digraph is a spherical st-graph.

Theorem 6.4 *Let G be a planar undirected graph with n vertices.*

1. *G admits a tessellation representation on the plane if and only if it is possible to add an edge* (s, t) *to G such that the resulting graph is 2-connected and planar.*

2. *G admits a tessellation representation on the sphere if and only if it is possible to add an edge* (s, t) *to G such that the resulting graph is 2-connected.*