数理解析研究所講究録
第 427 巻 1981 年 1-14

ON THE MAPPING DATA OF PLANAR GRAPHS

Takao Ozawa

Department of Electrical Engineering
Kyoto University, Kyoto, Japan 606

1. Introduction

A planar graph is defined to be a graph which can be mapped on a plane. The definition itself indicates only a gloval nature of a graph, and in order to map a planar graph on a plane a set of data concerning the local structure of the graph in terms of graph elements such as vertices, edges, and/or faces, is necessary. There can be varieties of such sets of data for planar mapping. In view of a graph algorithm which is applied to the graph, a set may be more convenient than the other, or an algorithm can be more efficient if it can utilize more than one set of data. Then it becomes necessary to generate sets of data from the given set.

In this paper the relations among these sets are investigated and generation or conversion algorithms from a set to others are presented. We consider sets of data involving vertices, edges, and/or faces only, although there can be data involving paths, etc. Special cares are taken to make the time complexity of the algorithms linear.

2. Sets of Mapping Data

Let G be an undirected planar graph with V vertices, E edges and F faces. We assume G is nonseparable. From a planar map of G we can obtain the following sets of mapping data. In these data an undirected edge in G is represented by two directed edges with opposit direction. For an directed edge e, e* denotes the other directed edge in the pair representing an undirected edge. The edge e* is called the reverse edge of e. Note (e*)*=e.

INEV(e): vertex from which edge e is coming out.
SVE(v): circular sequence of edges around vertex v.
SVV(v): circular sequence of vertices around vertex v.
SVF(v): circular sequence of faces around vertex v.
INEF(e): face which lies on the right side of edge e.
SFE(f): circular sequence of edges around face f.
SFV(f): circular sequence of vertices around face f.
SFF(f): circular sequence of faces around face f.

The vertices, edges and faces of G are numbered from 1 to V, 2E and F respectively.

If we denote the corresponding mapping data for the geometrical dual of G by the same notations as those for G but with bars above them, we have the following relations.

$$\mathrm{INEV}=\overline{\mathrm{INEF}},\quad \mathrm{SVE}=\overline{\mathrm{SFE}},\quad \mathrm{SVV}=\overline{\mathrm{SFF}},\quad \mathrm{SVF}=\overline{\mathrm{SFV}}. \qquad (1)$$

<u>Example 1</u>. For the graph shown in Fig. 1, we have INEV, INEF,

SVE, SVV and SVF as shown in Tables 1 and 2. V=5, E=7, F=4 and $e^*=(e+7)_{\text{mod } 14}$ for this graph.

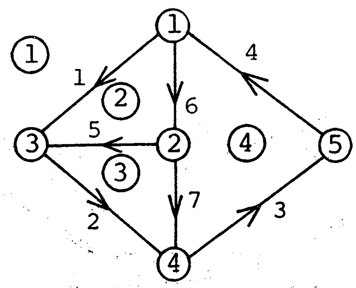

Fig. 1 Example 1

Table 1

e	1	2	3	4	5	6	7	8	9	10	11	12	13	14
INEV(e)	1	3	4	5	2	1	2	3	4	5	1	3	2	4
INEF(e)	1	1	1	1	2	2	3	2	3	4	4	3	4	4

Table 2

v	SVE(v)	SVV(v)	SVF(v)
1	1, 6, 11	3, 2, 5	1, 2, 4
2	5, 7, 13	3, 4, 1	2, 3, 4
3	2, 12, 8	4, 2, 1	1, 3, 2
4	3, 14, 9	5, 2, 3	1, 4, 3
5	4, 10	1, 4	1, 4

3. Conversion Algorithms

Conversion of a set of data to other sets is depicted in Fig. 2. Since the data in a set involve only one or two of the three element kinds(vertices, edges and faces), graph elements which do not appear in a data set must be introduced when it is converted to another. For example SVE says nothing about faces. Thus to obtain SVF from SVE faces must be introduced. The introduced elements are properly numbered to identify them.

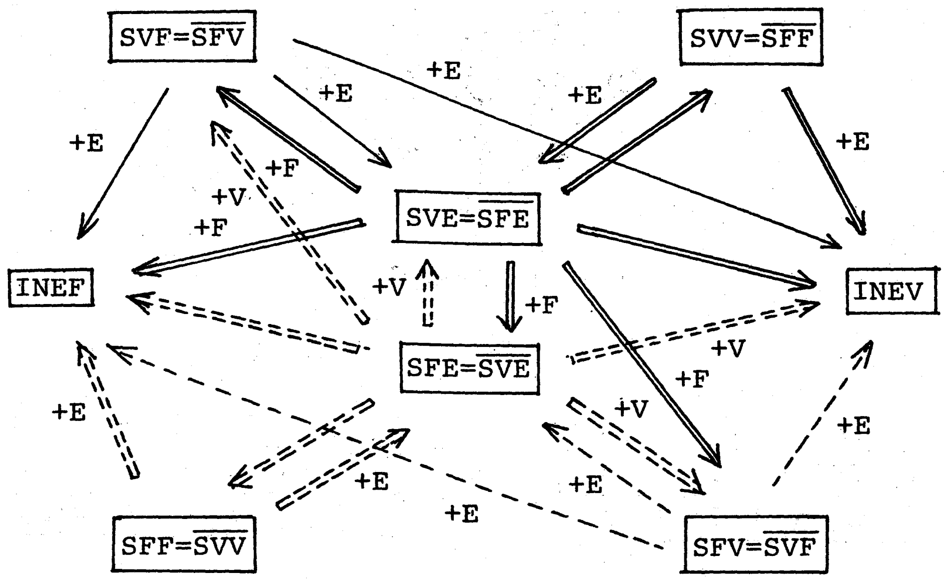

Fig. 2 Conversion of data sets

In Fig. 2 +V, +E and +F mean introducing vertices, edges and faces respectively. The dotted arrows indicate the conversions concerning the dual graph. The thin arrow from SVF to SVE indicates that this conversion is not necessarily unique.

(1.1) From SVE to INEV.

[Algorithm 1]

<u>step 1</u>. Set $v \leftarrow 1$.

<u>step 2</u>. For each edge e in SVE(v) set $INEV(e) \leftarrow v$.

<u>step 3</u>. If $v=V$, stop. Otherwise set $v \leftarrow v+1$ and go to step 2.

(1.2) From SVE to SVV.

To obtain SVV from SVE, INEV(e) is used. To each edge in

SVE(v) corresponds a vertex in SVV(v).

[Algorithm 2]

<u>step 1</u>. Set $v \leftarrow 1$.

<u>step 2</u>. For each edge e in SVE(v) set the vertex in SVV(v) corresponding to e, equal to INEV(e*).

<u>step 3</u>. If v=V stop. Otherwise set $v \leftarrow v+1$ and go to step 2.

(1.3) From SVE to INEF, SVF, SFV or SFE.

INEV is used here, too. To each edge in SVE(v) corresponds a face in SVF(v).

[Algorithm 3]

<u>step 1</u>. Set $v \leftarrow 1$ and $r \leftarrow 0$.

<u>step 2</u>. If all the edges in SVE(v) are scanned, go to step 4.

<u>step 3</u>. Let e be the first unscanned edge in SVE(v).

<u>step 3.1</u>. Set $r \leftarrow r+1$. Introduce a face f_r. Set $v_J \leftarrow v$ and $e_J \leftarrow e$. (v_J and e_J are variables indicating a vertex and an edge respectively.)

<u>step 3.2</u>. Set the face in SVF(v_J) corresponding to e_J, equal to f_r. Set INEF(e_J)$\leftarrow f_r$. Add v_J and e_J to SFV(f_r) and SFE(f_r) respectively. Mark e_J scanned.

<u>step 3.3</u>. Let e_K be the edge following e_J in SVE(v_J) and let v_K=INEV(e_K*).

<u>step 3.4</u>. If e_K*=e, go to step 2. Otherwise set $v_J \leftarrow v_K$ and $e_J \leftarrow e_K$*, and go to step 3.2.

<u>step 4</u>. If v=V, stop. Otherwise set $v \leftarrow v+1$ and go to step 2.

At step 3 a face f_r is introduced. Then the vertices and edges around this face are sought in steps 3.2, 3.3 and 3.5, as is illustrated in Fig. 3.

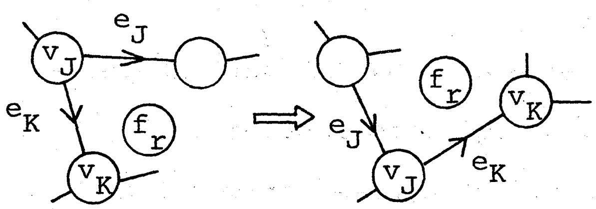

Fig. 3 Step 3.2 to step 3.4.

(2) From SVV to INEV, SVE.

To obtain INEV from SVV edges must be introduce. Some cares must be taken for parallel edges. Once INEV is obtained, SVV is essentially the same as SVE, and conversion algorithms from SVV to others can be derived similarly to those from SVE.

(3) From INEV and INEF to SVE.

By sorting the edges on the keys(vertices) given by INEV, we get a set of edges around each vertex. The order of edges in SVE, however, cannot be determined from INEV only. As is shown in Fig. 4(a), the set of edges(undirected) around a vertex and the set of edges around a face have two edges in common, if they have any common edges at all. These two edges

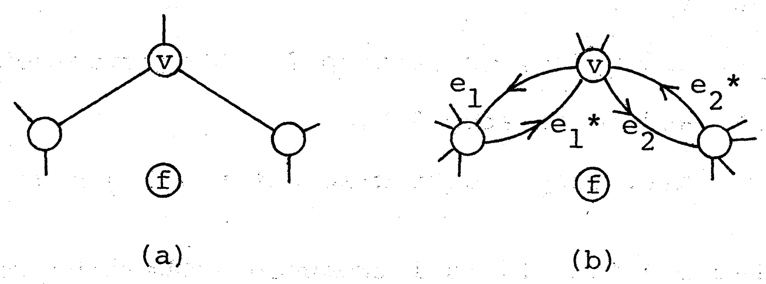

Fig. 4 Common edges of SVE(v) and a face.

must be consecutive in SVE(v), and by finding these consecutive edges we can determine the order of edges in SVE.

To find such edges we sort all the edges again on the keys (faces) given by INEF this time. Now, INEV and INEF are data on directed edges. If undirected edges are replaced by pairs of directed edges, the consecutive edges in SVE and the edges in INEF are not the same. As is shown in Fig. 4(b), if the consecutive edges around v are e_1 and e_2, e_1* and e_2 give INEF(e_1*)=INEF(e_2)=f. Considering this fact we have the following algorithm, in which we can use the bucket sort to obtain linear time complexity.

[Algorithm 4]

step 1. Sort all the edges on the keys(vertices) given by INEV to obtain a sequence of edges which is partitioned into sets of edges around vertices. The sequence is denoted by S_A and the sets of edges around vertex v is denoted by AVE(v).

step. 2. Behind each edge in S_A insert its reverse edge to obtain a new sequence S_B. The set of edges thus obtained from AVE(v) is denoted by BVE(v).

step 3. Sort the edges of S_B on the keys(faces) given by INEF. By this sorting BVE(v) is partitioned into subsets each of which consists of two adjacent edges around a face. The set of two edges obtained from BVE(v) for face f is denoted by CVE(v,f).

Now we are ready for the algorithm to obtain SVE from INEV and INEF.

[Algorithm 5]

step 1. Set $v \leftarrow 1$.

step 2. Let e be the first edge in AVE(v). Set $e_J \leftarrow e$.

step 3. Add e_J to SVE(v).

step 4. Let $f = INEF(e_J{}^*)$ and $\{e_J{}^*, e_K\} = CVE(v,f)$. If $e_K = e$, go to step 5. Otherwise set $e_J \leftarrow e_K$, and go to step 3.

step 5. If $v = V$ stop. Otherwise set $v \leftarrow v+1$ and go to step 2.

Example 1(continued). For the graph in Fig. 1 we get AVE and BVE as shown in Table 3 by sorting the edges on the keys given by INEV, and then inserting reverse edges. By sorting again

Table 3

v	1	2	3	4	5
AVE(v)	1, 6, 11	5, 7, 13	2, 8, 12	3 9, 14	4, 10
BVE(v)	1,8,6,13,11,4	5,12,7,14,13,6	2,9,8,1,12,5	3,10,9,2,14,7	4,11,10,3

Table 4

v	1			2			3			4			5	
CVE(v,f)	1,4	8,6	13,11	5,6	12,7	14,13	2,1	8,5	9,12	3,2	9,7	10,14	4,3	11,10
f	1	2	4	2	3	4	1	2	3	1	3	4	1	4

BVE on the keys given by INEF, we get CVE in Table 4. The first steps of application of Algorithm 5 are as follows.

step 1: $v=1$. step 2: $e=1$, $e_J=1$. step 3: $SVE(1)=\{1\}$.

step 4: $e_J{}^*=8$, $f=2$, $e_K=6$. $e_K \neq e$. $e_J=6$. step 3: $SVE(1)=\{1,6\}$.

step 4: $e_J{}^*=13$, $f=4$, $e_K=11$. $e_K \neq e$. $e_J=11$.

step 3: $SVE(1)=\{1,6,11\}$. step 4: $e_J{}^*=4$, $f=1$, $e_K=1$. $e_K=e$.

step 5: $v \neq V$. $v=1+1=2$. step 2: $e=5$, $e_J=5$. ...

(4.1) From SVF to INEV and INEF.

To obtain INEV from SVF edges must be introduced. A face and its subsequent face in SVF(v) define an edge, which, however, may not be unique. For example we get $SVF(v_J) = SVF(v_K) = \{f_R, f_L\}$ from Fig. 5. From these we get two pairs of faces (f_R, f_L), (f_R, f_L), which must define different two edges.

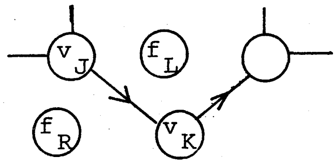

Fig. 5 Example 2.

For the simplicity of discussion we first assume that there are no two edges having the same pair of faces on their both sides, and therefore the pairs obtained from SVE are all unique.

Another problem in introducing edges is to identify the directed edge pair e and e* representing an undirected edge. If a pair of faces (f_R, f_L) is replaced by an edge e, (f_L, f_R) must be replaced by e*. This problem can be solved in linear time by bucket sorting all the pairs of faces twice. In the following algorithm the number of faces in SVF(v) is denoted by NF(v), and p is the number to identify a pair of faces.

[Algorithm 6]

step 1. Set $v \leftarrow 1$ and $p \leftarrow 1$.

step 2. From SVF(v) construct a sequence of pairs of faces, each of which are consecutive in SVF(v). Assign numbers p, p+1,.., and p+NF(v)-1 to the pairs. Denote the pair assigned number p by $P(p) = (f_{Rp}, f_{Lp})$. Set $MV(p) \leftarrow v$. (MV(p) is a memory to record the relation between v and p.)

step 3. If $v<V$, set $p \leftarrow p+NF(v)$, $v \leftarrow v+1$, and go to step 2.

step 4. Sort numbers p on the keys given by P(p) twice to obtain two sequences of numbers, which are denoted by S_R and S_L. The keys for the first sorting are the pairs of faces in the order obtained from SVF, and those for the second are the pairs of faces in the reverse order to the above(If the order obtained from SVF is f_R, f_L, its reverse order is f_L, f_R.). Denote the k-th($k=1,2,\ldots,2E$) number p in S_R and S_L by $S_R(k)$ and $S_L(k)$ respectively.

Now we can construct INEV and INEF.

[Algorithm 7]

step 1. Set $k \leftarrow 1$ and $e \leftarrow 1$.

step 2. Suppose $S_R(k)=p$. If $f_{Rp}<f_{Lp}$ for P(p), introduce a new edge e. Set $INEV(e) \leftarrow MV(p)$, $INEF(e) \leftarrow f_{Rp}$, $ME(p) \leftarrow e$(ME(p) is a memory to record the relation between e and p.), and $e \leftarrow e+1$, and go to step 4.

step 3. Now $f_{Rp}>f_{Lp}$ for P(p), and an edge has already been introduced for (f_{Lp}, f_{Rp}). This edge is given by $ME(p_L)$ where $p_L=S_L(k)$. Suppose $ME(p_L)=e_L$ and $e_L{}^*=(e_L+E)_{\text{mod } 2E}$. Then set $INEV(e_L{}^*) \leftarrow V(p)$, $INEF(e_L{}^*) \leftarrow f_{Rp}$ and $ME(p) \leftarrow e_L{}^*$.

step 4. If $k=2E$, stop. Otherwise set $k \leftarrow k+1$, and go to step 2.

Next, if some of the pairs of faces obtained from SVF are same, we have to find some information from SVF to distinguish them. In case of a series connection of exactly two edges(undirected), the vertex to which they are incident can be identified, since SVF for it has only two faces. In introducing edges, we must take care so that different edges must be incident to the vertex.

Example 1(continued). From SVF in Table 2 for the graph in Fig. 1, we construct pairs of faces as given in Table 5. Then they are sorted to result in S_R and S_L in Table 6. By Algorithm 7 we get ME(p) as given in Table 5, and INEV and INEF as in Table 1.

Table 5

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P(p)	1,2	2,4	4,1	2,3	3,4	4,2	1,3	3,2	2,1	1,4	4,3	3,1	1,4	4,1
MV(p)	1	1	1	2	2	2	3	3	3	4	4	4	5	5
ME(p)	1	6	11	5	7	13	2	12	8	3	14	9	4	10

Table 6

k	1	2	3	4	5	6	7	8	9	10	11	12	14	14
$S_R(k)$	1	7	10	13	9	4	2	12	8	5	3	14	6	11
P	1,2	1,3	1,4	1,4	2,1	2,3	2,4	3,1	3,2	3,4	4,1	4,1	4,2	4,3
$S_L(k)$	9	12	14	3	1	8	6	7	4	11	13	10	2	5
P	2,1	3,1	4,1	4,1	1,2	3,2	4,2	1,3	2,3	4,3	1,4	1,4	2,4	3,4

If there is a series connection of more than two edges(undirected) in G, SVF does not give a unique planar mapping. For the cases other than the above we need more detailed discussion which is omitted here.

(4.2) From SVF to SVE.

Once INEV is obtained, SVE can easily be constructed from

SVF. The pairs of faces obtained from SVF are replaced by edges by use of number p and ME(p), that is, the pair numbered p is replaced by ME(p).

4. Planar Mapping of G.

From SVE, SVV or SFE a planar map of G can be obtained as discussed in references [1]-[5]. INEV or INEF alone is not sufficient for giving a planar map, but SVE can be constructed by use of both INEV and INEF, and then a planar map of G can be derived.

Suppose G is not triconnected. Then there can be more than one planar map of G, and mapping data may have to be modified to allow possible mappings. Let (a,a') be a separation pair of G.[4] If there are s_a separation classes(including classes consisting of one edge) with respect to (a,a'), there can be $(s_a-1)!$ ways of mapping the classes for the separation pair. A separation class can be split again, if it contains another separation pair. A split component is a separation class which contains no separation pair other than the relevant one. If a split component has more than one vertex besides the separation pair, it can be mapped in two ways on the plane. Let n_s be the number of such split components. Then the total number of planar maps of G is :

$$\prod_{(a,a')}^{\text{all separation pairs}} (s_a-1)!\, 2^{n_s} \qquad (2)$$

For a vertex v in a separation pair, SVE(v) is partitioned according to its separation classes. The order of subsequences obtained by the partitioning are changed to give different planar maps of G.

5. Concluding Remarks.

In the conversion algorithms given in this paper no searching for vertices, edges or faces is performed, since a searching in SVE etc. makes the time complexity of an algorithm nonlinear. A searching may be more efficient if only a part of mapping data is to be derived.

Actual drawing of a graph on a plane from a set of mapping data manually or by use of a computer is another problem. If the numbers of vertices, edges and faces are very large, this problem becomes very difficult to solve. An algorithm for actual drawing of such a graph will depend on what kind of drawing is needed.

Acknowledgements. This work was partly supported by the Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grant Cooperative Research (A) 435013(1979-1980).

References.

[1] Edmonds' Theorem: Busacker, R. G. and Saaty, T. L., "Finite Graphs and Networks," pp. 95-96, McGraw Hill, N. Y., 1965.

[2] Iri, M.: "On the method to draw a planar graph on a plane," Tech. Rep. Circuit & System Theory, CT68-28, Inst. Elec. Comm. Eng. Japan, 1968.

[3] Yoshida, K. and Ohta, T.: "Topological layout of monolithic integrated circuits," Trans. (C), Inst. Elec. Comm. Eng. Japan, vol.52-C, No.12, pp.796-803, 1969.

[4] Hopcroft, J. E. and Tarjan, R.: "Dividing a graph into triconnected components," SIAM J. Comup. vol.2, No.3, pp.135-159, 1973.

[5] Tutte, W. T.,: "How to draw a graph," Proc. London Math. Soc.(Series 3), vol.13, pp.743-767, 1963.