OMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 47, 227–242 (1989)

NOTE

A Systolic Algorithm for Extracting Regions from a Planar Graph

ZEN-CHUNG SHIH

*Department and Institute of Information Science, National Chiao Tung University,
Hsinchu 30050, Taiwan, Republic of China*

R. C. T. LEE

*Institute of Computer and Decision Sciences, National Tsing Hua University, Hsinchu 30043,
Taiwan, and Academia Sinica, Nankang, Taipei, Republic of China*

AND

S. N. YANG

*Institute of Computer and Decision Sciences, National Tsing Hua University,
Hsinchu 30043, Taiwan, Republic of China*

Received January 6, 1988; accepted January 4, 1989

In this paper, we describe a systolic algorithm for extracting all of the fundamental regions in a planar graph. It takes $O(n)$ computation time and uses $O(n)$ processing elements, where n is the number of edges of the input planar graph.

1. INTRODUCTION

Let us consider Fig. 1. In Fig. 1, there are four regions, as indicated (R_4 epresents the entire space not occupied by the graph). Suppose we are given all of he edges of the graph and their geometrical orientation. Can we find all of the egions automatically? A similar problem was discussed in [Nie 82]. In their roblem, we are given a polygon which is not simple and we are asked to produce ll regions formed by the intersecting line segments of the polygon.

In this paper, we shall show that indeed we can. We have designed a systolic lgorithm to produce all of these regions.

This paper is organized as follows: Some definitions and notations are given in ection 2. An overall picture of our algorithm is given in Section 3. Sections 4 and 5 ntroduce the systolic algorithm. Concluding remarks and future research are resented in Section 6.

2. DEFINITIONS AND NOTATIONS

DEFINITION 1. An *edge* is a portion of a line having two endpoints.

Let v_a and v_b be the endpoints of an edge e. This edge e will be represented by $v_a, v_b)$ or (v_b, v_a) and v_a and v_b are called the *ending vertices* of e. An edge without irection is called an undirected edge. An undirected edge $e = (v_a, v_b)$ defines two irected edges $\mathbf{e}_i = \langle v_a, v_b \rangle$ and $\mathbf{e}_j = \langle v_b, v_a \rangle$. Let $\mathbf{e} = \langle v_a, v_b \rangle$ be a directed edge.

0734-189X/89 $3.00

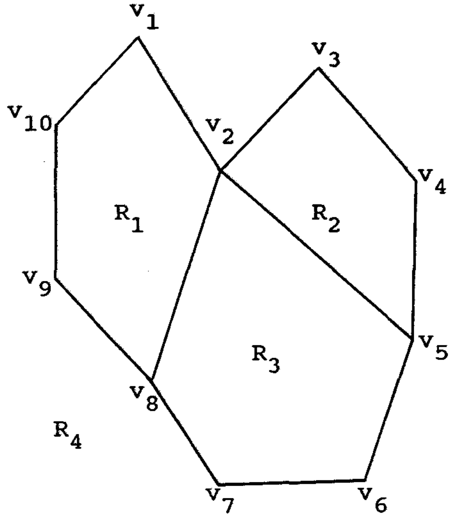

FIG. 1. There are four regions R_1, R_2, R_3, and R_4 in the planar graph.

We define v_a and v_b to be the *initial* and *terminal vertices* of **e**, respectively. Conversely, **e** is called the *outgoing* and *incoming edges* of v_a and v_b, denoted as OUTGO(v_a) and INCOM(v_b), respectively. For an edge, either undirected or directed, it is said to be *incident* on its ending vertices.

A *path* of directed edges is a sequence of directed edges which are linked at their ending vertices. Let $[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k]$ denote a path of directed edges where $\mathbf{e}_1 = \langle v_{i_1}, v_{i_2} \rangle, \mathbf{e}_2 = \langle v_{i_2}, v_{i_3} \rangle, \ldots, \mathbf{e}_k = \langle v_{i_k}, v_{i_{k+1}} \rangle$. A *cycle* of directed edges is a path of directed edges $[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m]$, where $\mathbf{e}_1 = \langle v_{j_1}, v_{j_2} \rangle, \mathbf{e}_2 = \langle v_{j_2}, v_{j_3} \rangle, \ldots, \mathbf{e}_{m-1} = \langle v_{j_{m-1}}, v_{j_m} \rangle$, and $\mathbf{e}_m = \langle v_{j_m}, v_{j_1} \rangle$, where $v_{j_a} \neq v_{j_b}$, for $1 \leq a < b \leq m$.

DEFINITION 2. A *polygon* is a closed plane figure formed by three or more edges.

Since a polygon is realized by a collection of edges, it can be specified by a cycle of directed edges. The cycle may begin with any edge and proceed in clockwise direction. That is, when we move along the cycle of directed edges, the interior of the polygon lies to the right of us.

DEFINITION 3. A polygon is *simple* [Sham 78] if and only if no non-consecutive edges intersect and consecutive edges intersect only at the ending vertices.

DEFINITION 4. The area which consists of a simple polygon and its interior is called a *region*.

Consider Fig. 1. The polygon $[\langle v_1, v_2 \rangle, \langle v_2, v_8 \rangle, \langle v_8, v_9 \rangle, \langle v_9, v_{10} \rangle, \langle v_{10}, v_1 \rangle]$ and its interior form region R_1. The polygon $[\langle v_2, v_3 \rangle, \langle v_3, v_4 \rangle, \langle v_4, v_5 \rangle, \langle v_5, v_6 \rangle, \langle v_6, v_7 \rangle, \langle v_7, v_8 \rangle, \langle v_8, v_2 \rangle]$ and its interior is also a region which is the union of regions R_2 and R_3. We shall say that the region which is not the union of other regions is a *fundamental region*, such as R_1, R_2 and R_3 in Fig. 1. We specify a region to be the same as specifying a polygon. The difference between a polygon and a region is that a polygon is a cycle of directed edges, but a region is an area.

DEFINITION 5. Let there be two edges $e_a = (v_{i_1}, v_{i_2})$ and $e_b = (v_{i_2}, v_{i_3})$. Let us assume that when we sweep from e_a to e_b in the counterclockwise direction, we encounter no other edges incident on v_{i_2}. Then the area comprised by e_a and e_b is a *wedge* between e_a and e_b, denoted as $(v_{i_1}, v_{i_2}, v_{i_3})$. The wedge $(v_{i_1}, v_{i_2}, v_{i_3})$ is said to be *associated* with v_{i_2}.

By the definition, if there are k edges incident on a vertex v, there are k wedges associated with v. Since an edge is incident on its two ending vertices, there are $2n$ wedges formed by an n-edge planar graph.

Let $w = (v_{i_1}, v_{i_2}, v_{i_3})$ be a wedge which is comprised by $\mathbf{e}_a = \langle v_{i_1}, v_{i_2} \rangle$ and $\mathbf{e}_b = \langle v_{i_2}, v_{i_3} \rangle$. We define $\mathbf{e}_a$ and $\mathbf{e}_b$ to be the *back* and *front edges* of w, denoted as $\mathrm{BE}(w)$ and $\mathrm{FE}(w)$, respectively. Two wedges w_1 and w_2 are said to be *contiguous* if $w_1 = (v_{i_1}, v_{i_2}, v_{i_3})$ and $w_2 = (v_{i_2}, v_{i_3}, v_{i_4})$. For two consecutive wedges w_1 and w_2, there is a common edge, $\mathbf{e} = \langle v_{i_2}, v_{i_3} \rangle$, shared by them, i.e., $\mathrm{FE}(w_1) = \mathrm{BE}(w_2)$. We shall define w_1 and w_2 to be the *back* and *front wedges* of $\mathbf{e}$, denoted as $\mathrm{BW}(\mathbf{e})$ and $\mathrm{FW}(\mathbf{e})$, respectively.

Since a fundamental region contains no other regions, it is more appropriate to describe a fundamental region by a sequence of contiguous wedges. Let $R = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k]$ be a fundamental region, where $\mathbf{e}_1 = \langle v_1, v_2 \rangle, \mathbf{e}_2 = \langle v_2, v_3 \rangle, \dots, \mathbf{e}_k = \langle v_k, v_1 \rangle$. Then R can also be represented as $[w_1, w_2, \dots, w_k]$, where $w_1 = (v_1, v_2, v_3)$, $w_2 = (v_2, v_3, v_4), \dots, w_k = (v_k, v_1, v_2)$. From the above discussion, a sequence of wedges $[w_1, w_2, \dots, w_m]$ defines a fundamental region if and only if w_i and w_{i+1} are contiguous, for $i = 1, 2, \dots, m-1$, and w_m and w_1 are also contiguous.

3. AN OVERALL PICTURE OF THE REGION EXTRACTION ALGORITHM

In this section, we shall try to present an overall picture of our region extraction algorithm.

The input of our algorithm is a set of undirected edges. To extract regions, we must perform two tasks:

(1) Find all of the wedges.

(2) Group the wedges into sequences such that each sequence of wedges corresponds to a region.

Let us explain this informally by an example. Consider Fig. 2. Since there are six edges in this planar graph, by the definition, there are twelve wedges formed. All the wedges will be found to be (v_5, v_1, v_4), (v_4, v_1, v_2), (v_2, v_1, v_5), (v_1, v_2, v_3), (v_3, v_2, v_1), (v_2, v_3, v_4), (v_4, v_3, v_2), (v_3, v_4, v_1), (v_1, v_4, v_5), (v_5, v_4, v_3), (v_1, v_5, v_4), and (v_4, v_5, v_1).

After finding all of the wedges, our algorithm will then group them into the following sequences:

$$S_1 : [(v_5, v_1, v_4), (v_1, v_4, v_5), (v_4, v_5, v_1)]$$

$$S_2 : [(v_4, v_1, v_2), (v_1, v_2, v_3), (v_2, v_3, v_4), (v_3, v_4, v_1)]$$

$$S_3 : [(v_2, v_1, v_5), (v_1, v_5, v_4), (v_5, v_4, v_3), (v_4, v_3, v_2), (v_3, v_2, v_2)].$$

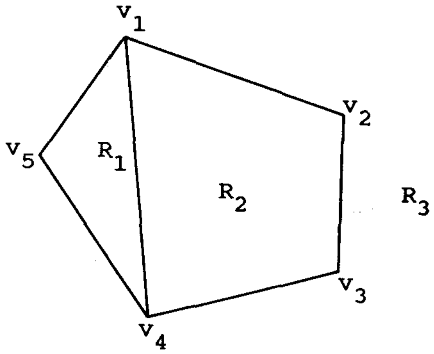

FIG. 2. There are twelve wedges in the planar graph.

It can be easily seen that each sequence of wedges corresponds to a fundamental region. The following table gives the one-to-one correspondence relationship:

$$S_1 : R_1$$
$$S_2 : R_2$$
$$S_3 : R_3.$$

In the following section, we shall explain how we can find all of the wedges by examining the input edges.

4. PHASE ONE: A SYSTOLIC ALGORITHM TO FIND WEDGES

The principle to construct wedges is quite simple. Consider Fig. 3. There are four edges incident on v_1. Note that the edges mentioned in this section are undirected. Let us sort these four edges according to their angles with respect to the horizontal line passing through v_1. We have $(v_1, v_2) < (v_1, v_3) < (v_1, v_4) < (v_1, v_5)$. After these edges are sorted, we can consider these edges as forming a cycle by logically linking the last edge to the first one. Then, each pair of consecutive edges in this cycle forms a wedge.

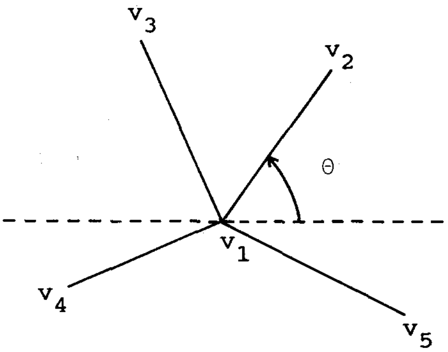

FIG. 3. There are four edges incident on v_1.

In general, let e_i and e_{i+1} be a pair of consecutive edges. Let v_b be the ending vertex common to both edges. Let v_a and v_c be the ending vertices other than v_b which appear in e_i and e_{i+1}, respectively. Then e_i and e_{i+1} are considered to form the wedge (v_a, v_b, v_c). For the above sorted sequence of edges, the following wedges are formed: (v_2, v_1, v_3), (v_3, v_1, v_4), (v_4, v_1, v_5), and (v_5, v_1, v_2). The last wedge is formed by combining the last and the first edges.

Given a planar graph, let $E(v)$ denote the set of edges incident on a vertex v. To find the wedges associated with v, we may sort $E(v)$ according to their angles with respect to the horizontal line passing through v. We then combine each consecutive pair of edges in the sequence to form wedges. The last edge is also combined with the first edge in the sequence. Let $|E(v)|$ denote the number of edges incident on v. Then the total number of wedges formed is $|E(v)|$.

Actually, we do not have to separate the edges into groups and sort edges within each group. As shown below, we may sort all of the edges once and for all by the following rules:

(1a) Duplicate every edge. If an edge is incident on v_i and v_j, then one edge is labeled with tag i and the other edge is labeled with tag j. Thus, each edge is now labeled with a tag.

(1b) The precedence-relation between two edges is defined as follows:
- (a) If two edges have the same tag, $e_i < e_j$ if the angle between e_i and the horizontal line is smaller than the angle between e_j and the horizontal line.
- (b) If two edges have two different tags, then $e_i < e_j$ if the tag associated with e_i is smaller than the tag associated with e_j.

Using the above rules, duplicated edges of a planar graph can be sorted into a sequence. Note that in the resulting sorted sequence, edges labeled with the same tag will be grouped together. A linear scanning over all of the edges will produce all of the wedges.

Let us consider Fig. 4 which contains the same planar graph in Fig. 2. The input edges after the duplication process will be as follows:

$$(v_1, v_2)^1, (v_1, v_2)^2, (v_2, v_3)^2, (v_2, v_3)^3, (v_3, v_4)^3, (v_3, v_4)^4, (v_1, v_4)^1,$$
$$(v_1, v_4)^4, (v_1, v_5)^1, (v_1, v_5)^5, (v_4, v_5)^4, \text{and } (v_4, v_5)^5.$$

The superscripts of edges in the above sequence are the tag values. The sorting process will produce the sequence:

$$(v_1, v_5)^1, (v_1, v_4)^1, (v_1, v_2)^1, (v_1, v_2)^2, (v_2, v_3)^2, (v_2, v_3)^3, (v_3, v_4)^3,$$
$$(v_3, v_4)^4, (v_1, v_4)^4, (v_4, v_5)^4, (v_1, v_5)^5, \text{and } (v_4, v_5)^5.$$

A linear scanning over the sorted sequence will now take place to produce all of the wedges by using the following rules:

(2a) Within every group of edges with the same tag, combine every pair of consecutive edges.

(2b) Within every group of edges with the same tag, combine the last edge with the first edge.

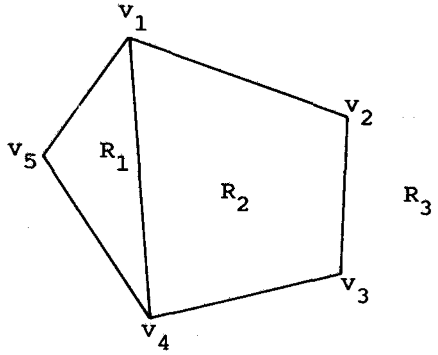

FIG. 4. There are 12 edges generated after the edge duplication process.

In our wedge-finding algorithm, we exploit the zero-time sorter [Lee 81; Mir 83] with a slight modification to perform the edge sorting process. There are n processing elements, which are controlled locally and named as $PE_1, PE_2, \ldots, PE_n$ from left to right.

In addition to the sorter, we shall attach a special cell S in front of the input/output cell of the sorter as shown in Fig. 5. In the input phase of sorting, each edge should pass through cell S and be duplicated. Then, the edges leave this cell and enter the sorter one by one. Therefore, there are $2n$ edges entering the sorter. In the output phase of sorting, the sorted sequence of edges should be scanned by this cell where the consecutive edges are merged to form wedges according to the rules discussed before. That is, this special cell performs two functions. It duplicates edges and attaches tags to them before they are fed into the sorter. It later examines edges coming out of the sorter and merges consecutive ones and also the last and the first in the same group to form wedges. Finally, there are $2n$ wedges constructed, where n is the number of input edges. Let these $2n$ wedges be denoted as $w_1, w_2, \ldots, w_{2n}$.

In this phase, there are $2n$ edges fed into the zero-time sorter. By [Mir 83], the time spent by the sorter is $O(n)$. The special cell S is operated parallel with the sorter. In the input phase of sorter, cell S scans and duplicates each edge in one time unit. Therefore, the total time complexity of phase one is $O(n)$ and takes $O(n)$ processing elements in the sorter.

5. PHASE TWO: A SYSTOLIC ALGORITHM TO EXTRACT REGIONS

In Section 4, we showed a systolic algorithm to find all of the wedges of a planar graph. In this section, we shall show how the wedges can be combined to form regions.

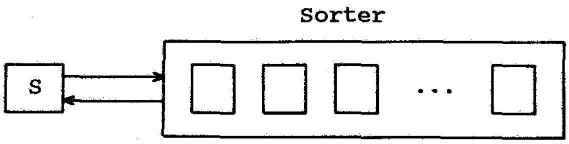

FIG. 5. The cell S is linked to the left of the sorter.

In our phase-two algorithm, we use a linear systolic array which consists of $2n + 1$ processing elements, where n is the total number of edges of the planar graph. The input of our phase-two algorithm is the set of wedges $w_1, w_2, \ldots, w_{2n}$, which is the output of our phase-one algorithm. This set of wedges define m regions $R_1, R_2, \ldots, R_m$. Let $|R_i|$, $i = 1, 2, \ldots, m$, denote the number of wedges of R_i. As explained in Section 3, a sequence of wedges $[w_{a_1}, w_{a_2}, \ldots, w_{a_k}]$ define a region if and only if w_{a_i} and $w_{a_{i+1}}$ are contiguous for $i = 1$ to $k - 1$ and w_{a_k} and w_{a_1} are also contiguous. Let $w_{i,j}$ denote a wedge of R_i. Let us assume that $w_{i,j}$ and $w_{i,j+1}$ are contiguous for $j = 1$ to $|R_i| - 1$ and $w_{i,|R_i|}$ and $w_{i,1}$ are also contiguous. Our algorithm will accept a set of wedges $w_1, w_2, \ldots, w_{2n}$ and output a sequence of wedges as follows: $w_{1,1}, w_{1,2}, \ldots, w_{1,|R_1|}, w_{2,1}, w_{2,2}, \ldots, w_{2,|R_2|}, \ldots, w_{m,1}, w_{m,2}, \ldots, w_{m,|R_m|}$. It should be obvious, that in the output sequence, each subsequence corresponds to a region.

Initially, the set of wedges $w_1, w_2, \ldots, w_{2n}$ are fed into the systolic array one by one from left to right, such that w_i will be stored in PE_i. Then, we shall arrange the wedges into the output sequence. That is, the wedges of each R_i are placed in continuous processing elements and the wedges in each pair of neighboring processing elements are contiguous if they belong to the same region.

Our Phase two algorithm consists of the following four major steps:

Step 1. Feed the set of $2n$ wedges $w_1, w_2, \ldots, w_{2n}$ into the systolic array one by one from left to right, such that w_i will be stored in PE_i.

Step 2. Find $w_{i,1}$, for $i = 1, 2, \ldots, m$. Then delete the back edge, $\text{BE}(w_{i,1})$, of $w_{i,1}$.

Step 3. Label all of the wedges in such a way that two wedges have the same label if and only if they are in the same region. And, assign a sequence number to each $w_{i,j}$ such that the sequence number of $w_{i,j}$ is j.

Step 4. Arrange the wedges so that they can be outputted in the following sequence: $w_{1,1}, w_{1,2}, \ldots, w_{1,|R_1|}, w_{2,1}, w_{2,2}, \ldots, w_{2,|R_2|}, \ldots, w_{m,1}, w_{m,2}, \ldots, w_{m,|R_m|}$, where $w_{i,j}$ and $w_{i,j+1}$ are contiguous, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, |R_i|$, and $w_{i,|R_i|}$ and $w_{i,1}$ are also contiguous, for $i = 1, 2, \ldots, m$.

Step 1 is rather simple. There are $2n + 1$ processing elements, labeled as $\text{PE}_1, \text{PE}_2, \ldots, \text{PE}_{2n+1}$ from left to right. We input the wedges $w_1, w_2, \ldots, w_{2n}$ one by one from left to right and w_i will be stored in PE_i, for $i = 1, 2, \ldots, 2n$. PE_{2n+1} is essentially a buffer area and its function will be explained and made clear later.

Inside each PE_i, $i = 1, 2, \ldots, 2n$ we shall store five pieces of data as shown in Fig. 6. Except w_i, which is stored in Step 1, the others are created by each PE_i. $\text{LABEL}(w_i)$ is the label of w_i. $\text{SEQNO}(w_i)$ is the sequence number of w_i. $\text{LABEL}(w_i)$

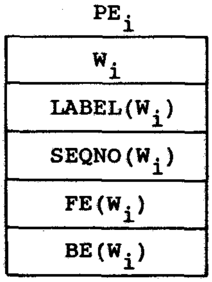

FIG. 6. There are five pieces of data stored in each PE_i, for $i = 1, 2, \ldots, 2n$.

and SEQNO(w_i) will be initialized to be i and 1, respectively, for all w_i's. FE(w_i) and BE(w_i) are the front edge and the back edge of w_i, respectively.

In the following paragraphs, we shall first introduce Step 2. Basically, we use a labeling scheme to identify each $w_{i,1}$ for $i = 1, 2, \ldots, m$. The basic principle of the labeling can be described below:

(3a) w_1 is labeled with 1.

(3b) For $1 < i \leq 2n$, if there exists a j, $j < i$, such that w_j and w_i are in the same region, then label w_i as w_j is labeled; otherwise, label w_i with i.

For example, imagine that we have wedges $w_1, w_2, \ldots, w_8$, where w_1, w_5, and w_6 belong to one region, w_2, w_3, and w_7 belong to one region, and the rest of wedges to another region. Then our labeling process will label the wedges as follows:

w_i	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
LABEL(w_i)	1	2	2	4	1	1	2	4

Step 2 consists of three basic steps. Step 2a labels each wedge in such a way that two wedges will be labeled the same if and only if they belong to the same region. Step 2b identifies the first wedge $w_{i,1}$ of region i for each i. Step 2c then eliminates the back edge of $w_{i,1}$.

Let $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{2n}$ be the set of directed edges of the input planar graph. In Step 2a, we feed each $\mathbf{e}_i$ to the linear systolic array. Each w_j is labeled j initially. After it enters PE_{2n+1}, it will have found the back wedge and the front wedge of $\mathbf{e}_i$. For instance, for edge $\langle 2,3 \rangle$, let its back wedge be $(1,2,3)$ and its front wedge be $(2,3,4)$. Let the label of $(2,3,4)$ be initially greater than that of $(1,2,3)$. Then, as edge $\langle 2,3 \rangle$ travels back through the systolic array, it will change the label of $(2,3,4)$ to the label of $(1,2,3)$.

But, we have another problem. Consider wedges $(1,2,3)$, $(2,3,4)$, and $(3,4,5)$. suppose edge $\langle 2,3 \rangle$ enters the systolic array before edge $\langle 3,4 \rangle$. Suppose $(1,2,3)$ is labeled as 1. $(2,3,4)$ will be labeled 1 as we discussed in the above paragraph. But, how can the label of $(3,4,5)$ be labeled to 1 also? This is accomplished as follows:

(a) *Case* 1. Edge $\langle 2,3 \rangle$ arrives at the processor holding $(2,3,4)$ as it travels back before edge $\langle 3,4 \rangle$ arrives as it travels forward. In this case, the label of the wedge $(2,3,4)$ will be changed by edge $\langle 2,3 \rangle$ to 1. Later, edge $\langle 3,4 \rangle$ arrives at this processor holding $(2,3,4)$. It will identify $(2,3,4)$ as its back wedge and also it will identify $(3,4,5)$ as its front wedge. Besides, it notes that the label of $(2,3,4)$ is 1. As it travels back, it will then change the label of $(3,4,5)$ to 1.

(b) *Case* 2. Edge $\langle 2,3 \rangle$ arrives at the processor holding $(2,3,4)$ as it travels back after edge $\langle 3,4 \rangle$ arrives as it travels forwardly. In this case, edge $\langle 3,4 \rangle$ will identify $(2,3,4)$ as its back wedge and it notes the label of $(2,3,4)$. Later, edge $\langle 3,4 \rangle$ will encounter edge $\langle 2,3 \rangle$ at a processor on the right of the processor holding $(2,3,4)$. The label of $(2,3,4)$ traveling with edge $\langle 3,4 \rangle$ will be changed by edge $\langle 2,3 \rangle$ to 1. As edge $\langle 3,4 \rangle$ travels back, it will then change the label of $(3,4,5)$ to 1.

Through the above mechanism, we can label all of the wedges such that they are labeled the same if and only if they are in the same region. This is the main task of Step 2a.

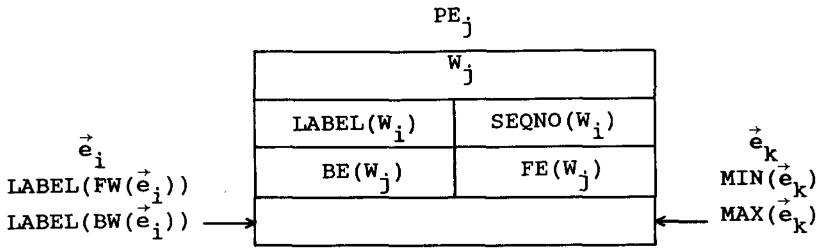

FIG. 7. Edges $\mathbf{e}_i$ and $\mathbf{e}_k$ may be entering from the left and right of PE_j, respectively.

In Step 2b, each processor PE_i checks the label of the wedge it holds. If the label of the wedge is also i, then this wedge is identified as the first wedge of a region. In Step 2c, each PE_k which holds $w_{i,1}$ eliminates the back edge of $w_{i,1}$.

To label each w_j, for $j = 1$ to $2n$, we have to input the set of directed edges $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{2n}$ one by one from left to right into the systolic array. $\mathbf{e}_i$ is fed into the systolic array at an interval of every two time steps. The reason will become clear later. For each $\mathbf{e}_i$, the following parameters travel together with $\mathbf{e}_i$: LABEL(FW($\mathbf{e}_i$)) and LABEL(BW($\mathbf{e}_i$)). Initially, LABEL(FW($\mathbf{e}_i$)) and LABEL(BW($\mathbf{e}_i$)) are set to null. As $\mathbf{e}_j$ enters PE_{2n+1}, we shall have two parameters attached to $\mathbf{e}_i$: MIN($\mathbf{e}_i$) and MAX($\mathbf{e}_i$). MIN($\mathbf{e}_i$) and MAX($\mathbf{e}_i$) will be explained later.

In the systolic array, $\mathbf{e}_i$ and the associating parameters travel from left to right and the movement takes place at an interval of every time step. As soon as $\mathbf{e}_i$ enters PE_{2n+1}, all of the parameters have been updated. Then, $\mathbf{e}_i$ will travel from right to left. Again, the movement takes place at an interval of every time step. Thus, at PE_j, $\mathbf{e}_i$ may be entering from the left and $\mathbf{e}_k$ may be entering from the right as shown in Fig. 7. In Step 2a, the value of SEQNO(w_j) is not updated.

The operations inside PE_j, $1 \le j \le 2n$, is summarized as follows:

(4a) If FE(w_j) = $\mathbf{e}_i$, then LABEL(BW($\mathbf{e}_i$)) := LABEL(w_j).

(4b) If BE(w_j) = $\mathbf{e}_i$, then LABEL(FW($\mathbf{e}_i$)) := LABEL(w_j).

(4c) If LABEL(BW($\mathbf{e}_i$)) = MAX($\mathbf{e}_k$), then LABEL(BW($\mathbf{e}_i$)) := MIN($\mathbf{e}_k$).

(4d) If LABEL(FW($\mathbf{e}_i$)) = MAX($\mathbf{e}_k$), then LABEL(FW($\mathbf{e}_i$)) := MIN($\mathbf{e}_k$).

(4e) If LABEL(w_j) = MAX($\mathbf{e}_k$), then LABEL(w_j) := MIN($\mathbf{e}_k$).

In PE_{2n+1}, if $\mathbf{e}_i$ arrives, then let MIN($\mathbf{e}_i$) be the minimum of LABEL(FW($\mathbf{e}_i$)) and LABEL(BW($\mathbf{e}_i$)) and MAX($\mathbf{e}_i$) be the maximum of LABEL(FW($\mathbf{e}_i$)) and LABEL(BW($\mathbf{e}_i$)).

Now, let us consider a simple example. There are five wedges in the systolic array. The whole process of entering $\langle 2,3\rangle$ and $\langle 3,4\rangle$ is shown in Fig. 8. In this example, when $\langle 2,3\rangle$ arrives at PE_1 and PE_3, the parameters LABEL(FW($\langle 2,3\rangle$)) and LABEL(BW($\langle 2,3\rangle$)) are set according to operations (4a) and (4b), respectively. When $\langle 3,4\rangle$ arrives at PE_3 and PE_4, the parameters LABEL(FW($\langle 3,4\rangle$)) and LABEL(BW($\langle 3,4\rangle$)) are set according to operations (4a) and (4b), respectively. At time step 7, $\langle 2,3\rangle$ and $\langle 3,4\rangle$ encounter in PE_5 and the parameter LABEL(BW($\langle 3,4\rangle$)) of $\langle 3,4\rangle$ is updated according to (4c). At time steps 9 and 10, the labels of $(2,3,4)$ and $(3,4,1)$ are updated according to (4e).

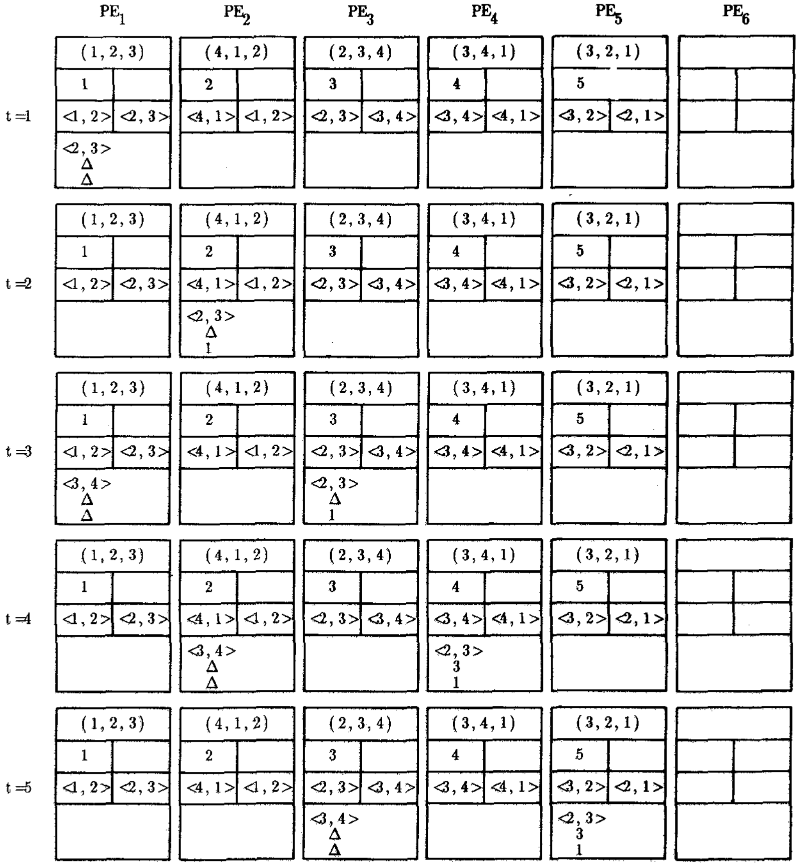

FIG. 8. An example of the labeling scheme to find all $w_{i,1}$'s.

From the above example, it is easy to see why $\mathbf{e}_i$ is fed into the systolic array every two time steps. In order to ensure operations (4c) and (4d) to be performed, $\mathbf{e}_i$ and $\mathbf{e}_k$ should encounter in some processing element. Only by feeding $\mathbf{e}_i$ every two time steps can achieve this purpose.

When the above labeling process is completed, we can easily identify $w_{i,1}$, for $i = 1, 2, \dots, m$. Then, we shall clear the value of BE($w_{i,1}$).

In the following paragraphs, we shall discuss Step 3. Before starting Step 3, each PE$_i$ should initialize LABEL(w_i) with the value i, again.

Basically, Step 3 consists of the labeling operation and the sequencing operation. The basic principle of labeling is the same as that of Step 2. The sequencing operation is to assign a sequence number to each w_i, for $i = 1, 2, \dots, 2n$. The basic principle of sequencing is described below:

(5a) The sequence number of w_1 is 1.

(5b) For $1 < i \le 2n$, if there exists a j, $j < i$, such that w_j and w_i are in the same region and $w_i = \overbrace{\mathrm{W}(\mathrm{FE}(\dots\mathrm{FW}(\mathrm{FE}}^{k\text{ levels}}(w_j))\dots))$, then SEQNO($w_i$) is set to SEQNO($w_j$) + k; otherwise SEQNO(w_i) is 1.

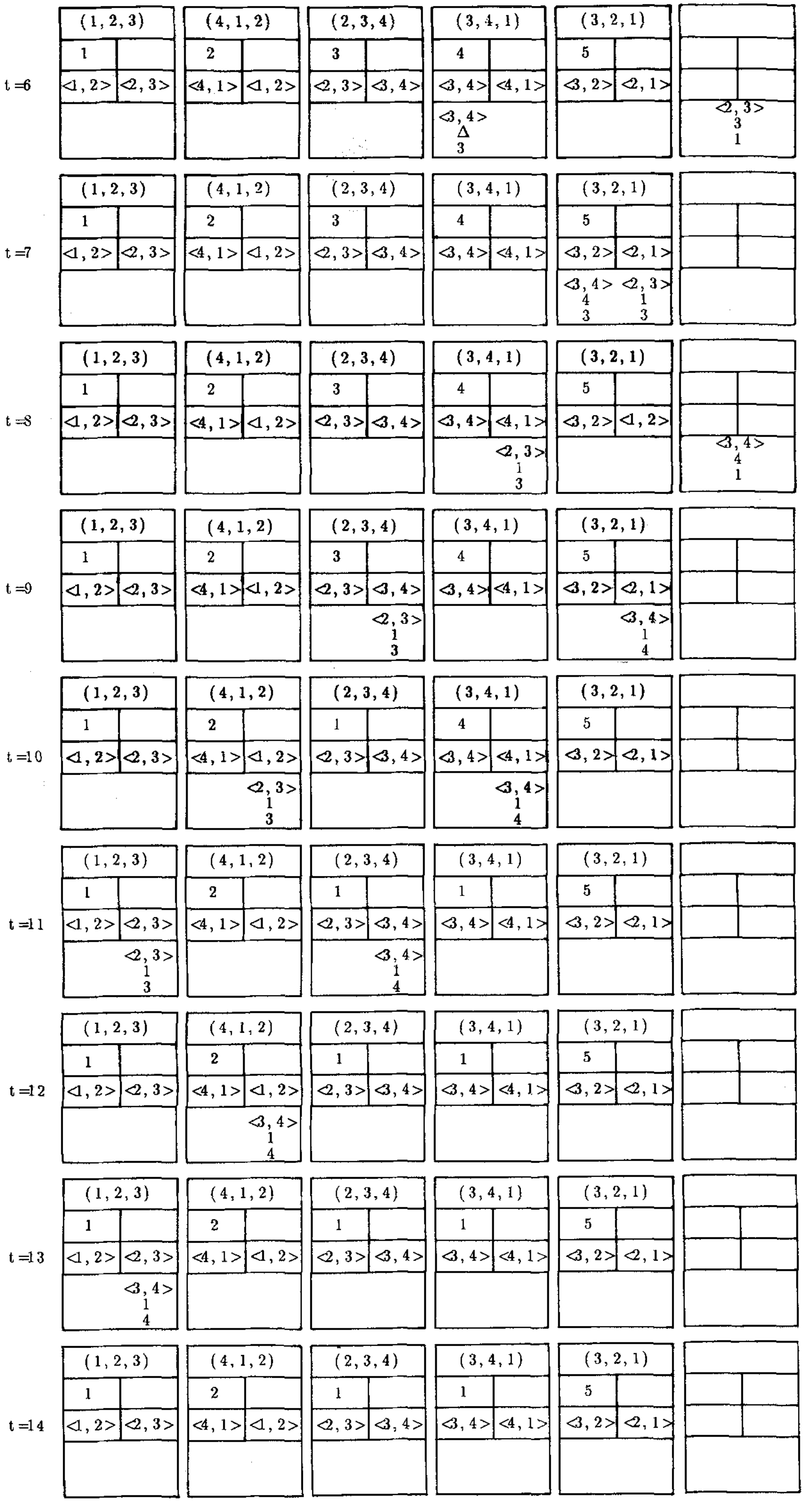

FIG. 8—*Continued*

For example, imagine that we have wedges $w_1, w_2, \ldots, w_8$, where w_1, w_5, and w_6 belong to one region, w_2, w_3, and w_7 belong to one region and the rest of wedges to another region. Moreover, $w_6 = \mathrm{FW}(\mathrm{FE}(w_1))$, $w_5 = \mathrm{FW}(\mathrm{FE}(\mathrm{FW}(\mathrm{FE}(w_1))))$, $w_3 = \mathrm{FW}(\mathrm{FE}(w_2))$, $w_7 = \mathrm{FW}(\mathrm{FE}(w_3))$ and $w_8 = \mathrm{FW}(\mathrm{FE}(w_4))$. Then, the result of the sequencing process is:

w_i	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
SEQNO(w_i)	1	1	2	1	3	2	3	2

Initially, each w_i is stored in PE$_i$. Then the set of directed edges $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{2n}$ are fed into the systolic array one by one from left to right again. Each $\mathbf{e}_i$ is fed into the systolic array at an interval of every two time steps. After it enters PE$_{2n+1}$, it will have found the back wedge and front wedge of $\mathbf{e}_i$. For instance, for edge $\langle 2,3\rangle$, let its back wedge be $(1,2,3)$ and its front wedge be $(2,3,4)$. Suppose that the label of $(2,3,4)$ is greater than that of $(1,2,3)$. Later, edge $\langle 2,3\rangle$ will travel back through the systolic array. We shall change the label of $(2,3,4)$ to the label of $(1,2,3)$ and change the sequence number of $(2,3,4)$ to the sequence number of $(1,2,3)$ plus one.

Consider wedges $(1,2,3)$, $(2,3,4)$, and $(3,4,5)$. Let edge $\langle 2,3\rangle$ enter the systolic array before edge $\langle 3,4\rangle$. Suppose $(1,2,3)$ is both labeled and sequenced as 1. $(2,3,4)$ will be labeled 1 and sequenced 2 as we discussed in the above paragraph. To label $(3,4,5)$ to 1 is the same as in Step 2. But, how can the sequence number of $(3,4,5)$ be changed to 3? This is accomplished as follows:

(a) *Case* 1. Edge $\langle 2,3\rangle$ arrives at the processor holding $(2,3,4)$ as it travels back before edge $\langle 3,4\rangle$ arrives as it travels forward. In this case, the sequence number of the wedge $(2,3,4)$ will be changed by edge $\langle 2,3\rangle$ to 2. Later, edge $\langle 3,4\rangle$ arrives at this processor holding $(2,3,4)$. It will identify $(2,3,4)$ as its back wedge and also it will identify $(3,4,5)$ as its front wedge. Besides, it notes that the sequence number of $(2,3,4)$ is 2. As it travels back, it will then change the sequence number of $(3,4,5)$ to 3.

(b) *Case* 2. Edge $\langle 2,3\rangle$ arrives at the processor holding $(2,3,4)$ as it travels back after edge $\langle 3,4\rangle$ arrives as it travels forward. In this case, edge $\langle 3,4\rangle$ will identify $(2,3,4)$ as its back wedge and it notes the sequence number of $(2,3,4)$. Later, edge $\langle 3,4\rangle$ will encounter edge $\langle 2,3\rangle$ at a processor on the right of the processor holding $(2,3,4)$. The sequence number of $(2,3,4)$ traveling with edge $\langle 3,4\rangle$ will be changed by edge $\langle 2,3\rangle$ to 2. As edge $\langle 3,4\rangle$ travels back, it will then change the sequence number of $(3,4,5)$ to 3.

Through the above mechanism, we can set the sequence number of each $w_{i,j}$, for $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,|R_i|$, to j.

For each $\mathbf{e}_i$, the following parameters travel together with $\mathbf{e}_i$: LABEL(FW($\mathbf{e}_i$)), LABEL(BW($\mathbf{e}_i$)), and SEQNO(BW($\mathbf{e}_i$)). Initially, LABEL(FW($\mathbf{e}_i$)), LABEL(BW($\mathbf{e}_i$)), and SEQNO(BW($\mathbf{e}_i$)) are set to null. As $\mathbf{e}_i$ enters PE$_{2n+1}$, we shall have four parameters attached to $\mathbf{e}_i$: MIN($\mathbf{e}_i$), MAX($\mathbf{e}_i$), LABEL(FW($\mathbf{e}_i$)), and BASE($\mathbf{e}_i$). BASE($\mathbf{e}_i$) will be explained and made clear later.

In the systolic array, $\mathbf{e}_i$ and its associating parameters travel from left to right and the movement takes place at an interval of every time step. As soon as $\mathbf{e}_i$ enters PE$_{2n+1}$, all of the parameters have been updated. Then, $\mathbf{e}_i$ will travel from right to

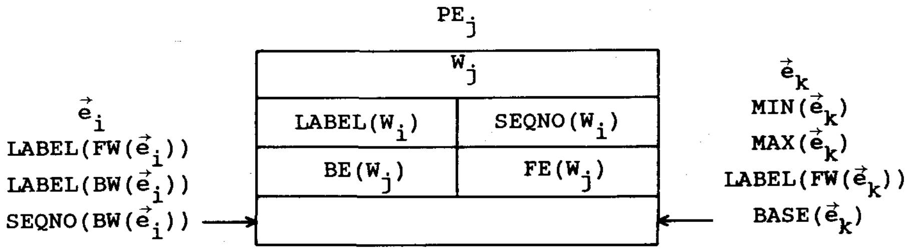

FIG. 9. Edges $\mathbf{e}_i$ and $\mathbf{e}_k$ may be entering from the left and right of PE$_j$, respectively.

left. Again the movement takes place at an interval of every time step. Thus, at PE$_j$, $\mathbf{e}_i$ may be entering from the left and $\mathbf{e}_k$ may be entering from the right as shown in Fig. 9.

The operations inside PE$_j$, $1 \le j \le 2n$, are summarized as follows:

(6a) If FE(w_j) = $\mathbf{e}_i$, then

$$\mathrm{LABEL}(\mathrm{BW}(\mathbf{e}_i)) := \mathrm{LABEL}(w_j)$$
$$\mathrm{SEQNO}(\mathrm{BW}(\mathbf{e}_i)) := \mathrm{SEQNO}(w_j).$$

(6b) If BE(w_j) = $\mathbf{e}_i$, then

$$\mathrm{LABEL}(\mathrm{FW}(\mathbf{e}_i)) := \mathrm{LABEL}(w_j).$$

(6c) If LABEL(BW($\mathbf{e}_i$)) = LABEL(FW($\mathbf{e}_k$)), then

$$\mathrm{SEQNO}(\mathrm{BW}(\mathbf{e}_i)) := \mathrm{SEQNO}(\mathrm{BW}(\mathbf{e}_i)) + \mathrm{BASE}(\mathbf{e}_k).$$

(6d) If LABEL(BW($\mathbf{e}_i$)) = MAX($\mathbf{e}_k$), then

$$\mathrm{LABEL}(\mathrm{BW}(\mathbf{e}_i)) := \mathrm{MIN}(\mathbf{e}_k).$$

(6e) If LABEL(FW($\mathbf{e}_i$)) = MAX($\mathbf{e}_k$), then

$$\mathrm{LABEL}(\mathrm{FW}(\mathbf{e}_i)) := \mathrm{MIN}(\mathbf{e}_k).$$

(6f) If LABEL(w_j) = LABEL(FW($\mathbf{e}_k$)), then

$$\mathrm{SEQNO}(w_j) := \mathrm{SEQNO}(w_j) + \mathrm{BASE}(\mathbf{e}_k).$$

(6g) If LABEL(w_j) = MAX($\mathbf{e}_i$), then

$$\mathrm{LABEL}(w_j) := \mathrm{MIN}(\mathbf{e}_k).$$

In PE$_{2n+1}$, if $\mathbf{e}_i$ arrives, then let MIN($\mathbf{e}_i$) be the minimum of LABEL(FW($\mathbf{e}_i$)) and LABEL(BW($\mathbf{e}_i$)), MAX($\mathbf{e}_i$) be the maximum of LABEL(FW($\mathbf{e}_i$)) and LABEL(BW($\mathbf{e}_i$)), and BASE($\mathbf{e}_i$) be SEQNO(BW($\mathbf{e}_i$)).

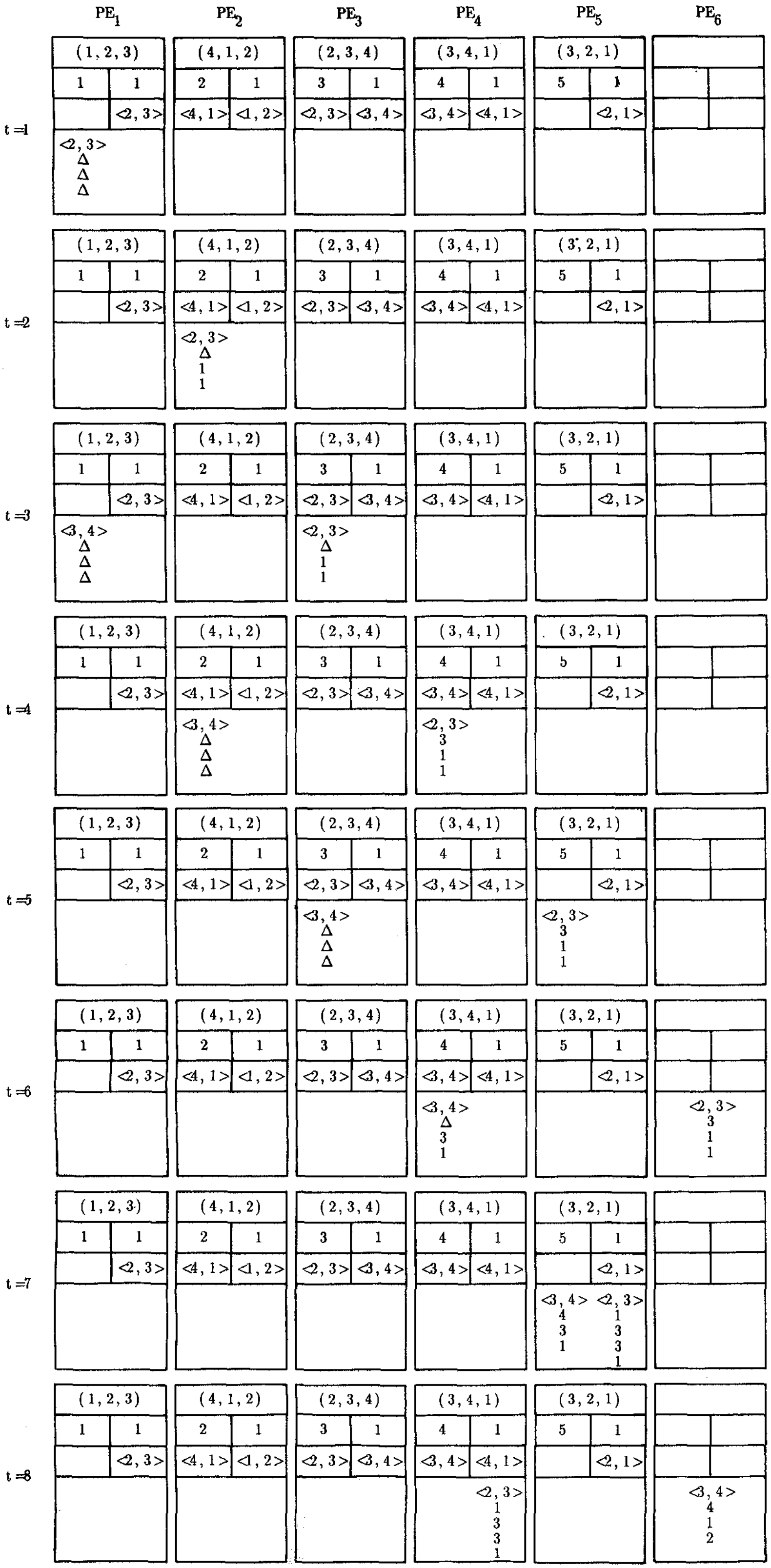

FIG. 10. An example of the sequencing process.

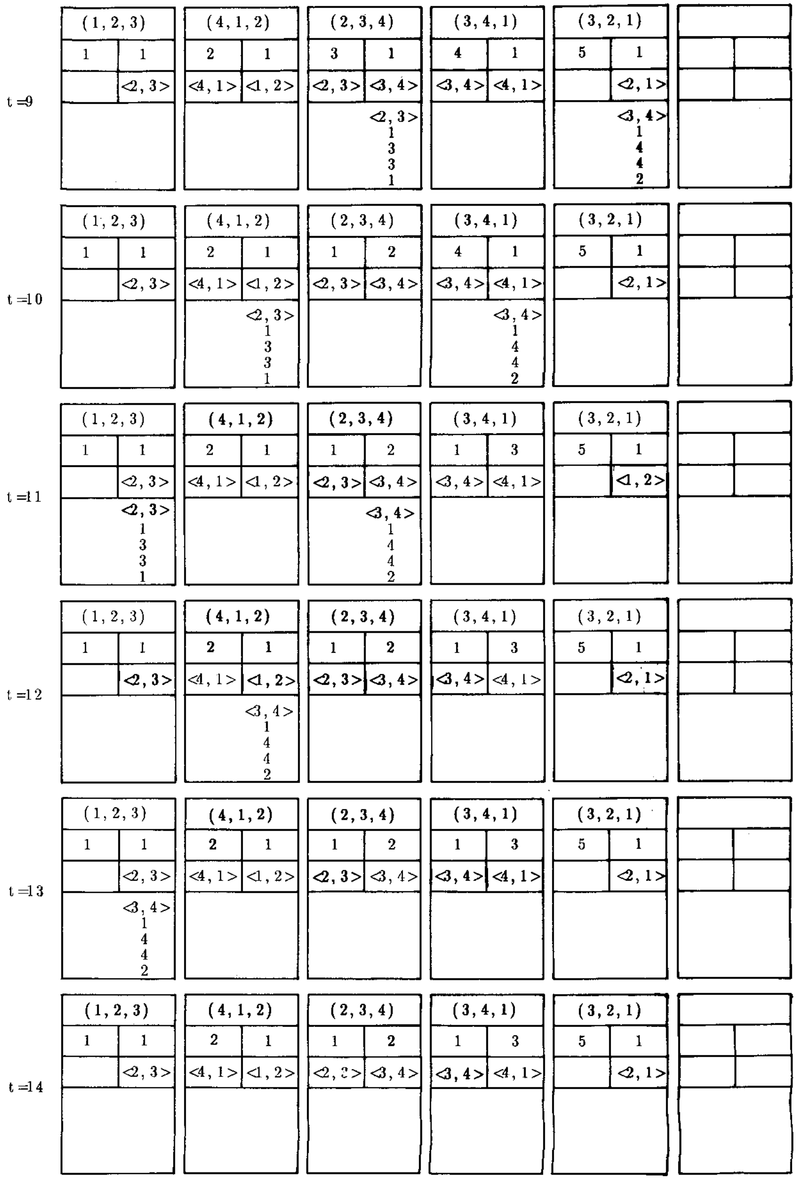

FIG. 10—*Continued*

Now, let us consider a simple example as shown in Fig. 10. There are five wedges in the systolic array. Initially, the label of the wedge in PE_i is i and the sequence number of each wedge is 1. Since $(1,2,3)$ and $(3,2,1)$ are the first wedges of regions, their back edges are set to null in Step 2.

When Step 3 is completed, the wedges have the same label if and only if they are in the same region. And, each $w_{i,j}$ for $i = 1$ to m and $j = 1$ to $|R_i|$, have sequence number j. Then we can apply a sorting process on the set of wedges $w_1, w_2, \ldots, w_{2n}$

in the systolic array according to the following rules:

(7a) $w_i < w_j$, if LABEL(w_i) < LABEL(w_j).

(7b) $w_i < w_j$, if LABEL(w_i) = LABEL(w_j) and

$$\text{SEQNO}(w_i) < \text{SEQNO}(w_j).$$

(7c) $w_i > w_j$, otherwise.

The above sorting process is the main task of Step 4. We can use the odd–even transposition sort [Baud 78, Knut 72] and zero-time sorter to accomplish the sorting process. When the sorting process completed, the wedges in the systolic array are in the following sequence: $w_{1,1}, w_{1,2}, \ldots, w_{1,|R_1|}, w_{2,1}, w_{2,2}, \ldots, w_{2,|R_2|}, \ldots, w_{m,1}, w_{m,2}, \ldots, w_{m,|R_m|}$, where $w_{i,j}$ and $w_{i,j+1}$ are contiguous, for $i = 1$ to m and $j = 1$ to $|R_i|$ and $w_{i,|R_i|}$ and $w_{i,1}$ are also continuous, for $i = 1$ to m.

Since there are $2n$ wedges output from phase one, the phase two algorithm needs $O(n)$ processing elements. In $O(n)$ time steps all of the directed edges can be fed into the systolic array, move forward and backward, and, finally, leave the systolic array. So, the time complexity of the phase two algorithm is $O(n)$.

6. CONCLUSIONS

In this paper, we showed how a systolic algorithm can be constructed to extract all regions of a planar graph. We believe that the techniques developed in this research can be used to develop a systolic algorithm to solve the polyhedra congruity problem discussed in [Sugi 84]. We are presently working on this research and we hope that we can report progress in the near future.

In the process of demonstrating the practical value of our region extraction algorithm we found another interesting research topic. That is, how are we going to put a label, say a 2-digit integer into a region appropriately. We should put this integer inside the region without touching the sides of the region and it should be close to the center as much as possible. We are also working on this research topic.

REFERENCES

[Baud 78] G. Baudet and D. Stevenson, Optimal sorting algorithms for parallel computers, *IEEE Trans. Comput.* **C-27**, No. 1, 1978, 84–87.

[Knut 72] D. E. Knuth, *The Art of Computer Programming*, Vol. 3, Addison-Wesley, Reading, MA, 1972.

[Lee 81] D. T. Lee, H. Chang, and C. K. Wong, An on-chip compare/steer bubble sorter, *IEEE Trans. Comput.* **C-30**, No. 6, 1981, 396–405.

[Mira 83] G. Miranker, L. Tang, and C. K. Wong, A zero-time VLSI sorter, *IBM J. Res. Develop.* **27**, No. 2, 1983, 140–148.

[Niev 82] J. Nievergelt and F. P. Preparata, Plane-sweep algorithms for intersecting geometric figures, *Commun. ACM* **25**, No. 10, 1982, 739–747.

[Sham 78] M. I. Shamos, *Computational Geometry*, Ph.D. dissertation, Computer Science Department, Yale Univ., New Haven, CT, 1978.

[Sugi 84] K. Sugihara, An $n \log n$ algorithm for determining the congruity of polyhedra, *J. Comput. System Sci.* **29**, No. 1, 1984, 36–49.