AN APPROACH TO TWO-DIMENSIONAL CHANNEL ROUTING

by

Y. K. Chen

Memorandum No. UCB/ERL M81/83

6 November 1981

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

AN APPROACH TO TWO-DIMENSIONAL CHANNEL ROUTING*

Y. K. Chen**

ABSTRACT

In the layout of LSI chips, channel routing is one of the key problems. The so-called "one-dimensional channel routing" means that only the terminals on the upper and lower sides of the channel are specified. But sometimes, we have a rectangular space with terminals specified on all four sides. This occurs, for example, when regular channel meets.

Two specific problems are given in this paper. One is called the fixed position problem, and the other the fixed sequence problem. Attempts are made to extend the merging algorithm to two-dimensional problem provided that extra tracks are available at the two ends of a channel.

The algorithm were coded in PASCAL and implemented on VAX 11/780 computer.

---

*Research supported by the national Science Foundation Grant ENG-78-24425.

**On leave from Tsinghua University, Beijing, China.

AN APPROACH TO TWO-DIMENSIONAL CHANNEL ROUTING

Y. K. CHEN

1. Introduction

In the layout of LSI chips, channel routing is one of the key problems. Two rows of cells are placed on two sides of a channel. Along the channel, every terminal of cells has a certain number, and terminals with the same number must be connected by a net. Usually there are horizontal output leads which go sidewards from the channel.

So-called "one-dimensional channel routing" means that the positions of the terminals on the upper and lower sides of the channel are specified while the position of the horizontal output leads are arbitrary (Fig. 1(a)).

Generally, in a LSI chip there are several channels in the routing area as shown in Fig. 1(b). In Fig. 1(b) the routing of channels A, B, C, and D are all one-dimensional problems. It can be routed individually without considering the others. However when we consider channel E, all the terminals, both on the upper and lower sides, and on the left and right ends, have already been specified. So the problem is no longer one dimensional. It leaves us a two-dimensional problem.

Two kinds of two-dimensional problem will be discussed in this paper. The following are the two different specifications:

(1) The fixed position problem: both the vertical and horizontal output leads are specified in fixed position. An example is shown in Fig(2).

(2) The fixed sequence problem: the vertical output leads are specified in fixed position while the horizontal output leads are specified in fixed sequence. An example is shown in Fig. (3). Compare the horizontal output leads in Fig. (2) and Fig. (3), we found that the two are not in a same position but have the same sequence. Obviously, the constraint in the fixed sequence problem is less tight than that in the fixed position problem.

Several efficient algorithms [1], [2] are available for one-dimensional channel routing problem. Now attempts are made to extend the merging algorithm [2] to two-dimensional problem. It must be pointed out that when we use this approach both the height and width of the channel must be adjustable.

In this paper we stress emphasis on the approach for fixed sequence problem. In section 3, two different approaches for the fixed sequence problem are proposed. In addition, an effective algorithm for finding the longest common subsequence of two strings used in section 3 is introduced in section 4 and section 5 .

2. Approach for the fixed position problem

The basic idea of this approach is to transfer the two-dimensional problem to three one-dimensional problems. Suppose the example in Fig.2 is the problem to be solved. This approach includes the following two steps:

(1) First Step:

First, without regard to the fixed position of the horizontal output leads, we treat the problem just as an one-dimensional problem with horizontal output leads in an arbitrary position. The result obtained after the first step guarantees the vertical output leads placed at the specified position while the position of the horizontal outputs are arbitrary (Fig.4(a)).

(2) Second Step:

Make a permutation to left and right horizontal output leads respectively. After the permutation both the left and right output leads obtained from the first step are permuted to the specified fixed positions. Such a two layer permutation can be directly realized by means of a one-dimensional channel routing (Fig.4(b)).

Combining the results from the above two steps the final result is obtained (Fig.4(c)). In Fig.4(c) , there are two and three extra columns which occur at the left and right end respectively. If the total length is still within the limit of the routing area the routing realization in Fig.4(c) is acceptable. When we use merging algorithm, if we start from a different zone the result may be different. So by selecting the starting zone at both the first and second steps, it is possible to obtain a qualified routing realization in which the width and the length do not exeed the limit of the routing area.

3. Approaches for fixed sequence problem

Just as in the fixed position approach the two-dimensional problem can also be reduced to a one-dimensional problem. Because the constraint in fixed sequence is less stringent than that in the fixed position we can accomplish more in the approach. Suppose the example in Fig.3 is the problem to be solved.

3.1 Approach with additional vertical constraint

Let us first introduce a simpler approach. The fixed sequence problem can be treated equivalently as a problem with no horizontal outputs. For the sake of simplicity we will only consider the left output leads. In Fig.3 the specified sequence of the left output leads is 4,1,2. It means that net 4 should be laid on a track above net 1 and net 1 should be laid on a track above net 2. In Fig.5(a) we extend two extra columns at the left end of the channel. And we assign "4" and "1" to the upper and lower terminals on the one extra column, and "1" and "2" on the other extra column. The purpose of doing so is not only to extend the output leads leftwards beyond the original end but more importantly to add some additional vertical constraints to the Vertical Constraint Graph (in Fig.5(b)). Because of these additional vertical constraints the specified sequence of the horizontal outputs is surely guaranteed. By using this approach we get Fig.5(a). If we delete the extra columns in Fig.5(a) the final result of Fig.3 is obtained (in Fig.5(c)). Obviously, Fig.5(c) is equivalent to Fig.5(a).

3.2 Approach with additional permutation

The approach used in the fixed position problem can also be used here. But now we only know the sequence of the horizontal output leads. So between the two steps used in the fixed position problem there is another step to be considered. That is to specify the position of the horizontal output leads. As illustrated in Fig.6, this approach includes three steps:

(1) First Step:
It is the same as the firt step in the fixed position problem (Fig.6(a)).

(2) Second Step:
After the first step two sets of horizontal outputs with arbitrary position are obtained. One set goes to left and the other goes to right (Fig.6(a)). For example, let us discuss the right side only. From the top to the bottom the number of these outputs are

1, 10, 7, 0, 8

These are the numbers assigned to the terminals placed on the left side of the right channel. According to the specification, the sequence of the terminals on the right side are specified as

1, 10, 8, 7

In this step we use the following criterion to specify the position of such terminals. As shown in Fig. 6(b), the result is

1, 10, 8, 7, 0

As shown in reference [3], the lower bound of the track number is related with dmax and vmax (dmax means the maximum density, vmax means the maximum level in V.C.G). So the basic idea of specifying the position is to reduce the "dmax" and "vmax" in the right channel. The criterion accepted in this step is

(a) Match terminals with the same number in the same column (i.e. to get straight connection) as much as possible, because straight connection does not require a track (i.e. not to increase dmax). In Fig. 6(b), 1 and 10 are matched with the same number. Here we use an efficient aigorithm, as shown in section 4 and 5.

(b) Because there is no vertical constraint between a nonzero terminal and a zero in a same column, then we match nonzero terminals with zero in a same column as much as possible in order not to increase "vmax". In Fig. 6(b), 7 is matched with zero.

(3) Third Step: After the second step the position of the horizontal output leads has been specified. In this step the only thing we have to do is to make the permutation, just as we did in the second step in the fixed position problem (Fig. 6(c)).

Combining Fig. 6 (a) and Fig. 6(c), we get the final result as shown in Fig. 6(d). Because in the fixed sequence problem the terminals of the horizontal output leads have matched in such a way , the extra columns on both ends of the channel may be less than that in the fixed position problem.

Computational examples for both fixed position and fixed sequence are shown in Fig. 7 and Fig. 8

respectively.

Fig 9 is the same example as Fig.8, but the other approach (with additional vertical constraint) is used. From Fig.8 and Fig.9 we can see that there is no extra row occurring in Fig.8 while no extra column occurred in Fig.9. We can select one of the two approaches depending on how many extra rows and columns there are in the routing area.

4. An effective algorithm for finding the longest common subsequence of two strings with certain constraints

4.1 The problem we have mentioned in section 3.2 (2)(a) can be summed up as follows:

We have two specified sets ,Set P and Set Q. In Set P there is no duplicated element except null. In Set Q there are no null elements but all the nonnull elements that occur in Set P. However, the nonnull elements generally do not appear in the same sequence as in Set P. As shown in Fig.10, now we match each element in Set Q with an element in Set P. But after matching, the sequence of Set Q must remain the same. All the elements matched with the same element are called the common subsequence. The problem is how to get the longest common subsequence.

In order to match each element in Set Q to an element in Set P, there are two constraints to be considered:

Constraint(1)------end constraint

When Set Q is matched with Set P no element in Set Q is allowed to go beyond either end of Set P. As an example in Fig.11(a), element M can not be matched in pair; otherwise, element A in Set Q has to go beyond the right end of Set P.

Constraint(2)------internal constraint

when any two elements in Set Q are matched with Set P all the elements between these two elements in Set Q can find their corresponding spacing in Set P. As an example in Fig.11(b), if element C has been matched in pair then element E can not be matched in pair, oherwise element B can not find a spacing in Set P.

Fig.11(c) shows the longest common subsequence obtained by matching Set P and Set Q.

4.2 The main algorithm

This algorithm includes the following three steps:

(1) First Step: We first delete the elements which do not satisfy constraint(1).

Check each of the elements in Set Q (also Set P) respectively by matching that element with the same in Set P. If any element in Set Q go beyond either end of Set P then the checked element does not satisfy constraint(1). We just delete the checked element from both Set Q and Set P, instead put in a null element "O". "O" is also treated as null element "O". After all the elements have been checked, the remaining nonnull elements satisfy constraint(1). As an example

Set P={A,O,C,E,O,B,O,F,D,O,M}
Set Q={C,B,E,F,D,M,A}

By checking in this way, except for A and M, all the other elements satisfy constraint(1). The result obtained is

Set P1={O,O,C,E,O,B,O,F,D,O,O}
Set Q1={C,B,E,F,D,O,O}

(2) Second Step: We disregard to constraint(2) and find the longest common subsequence

Construct a bipartite graph by connecting the same elements which are separated in SetP1 and Set Q1 , as shown in Fig.12(a). Count the intersections of each net with the other nets. According to the number of intersections, delete the net with the largest number of intersections from both SetP1 and SetQ1. After the deletion of edges, a null element "O" is assigned to the vertices which define the deleted edges. Iterations are made step by step until there is no intersection between nets.

Fig.12 illustrates the above procedure. First, net A, then net D is deleted. Finally, the final bipartite graph without edge intersection is obtained (Fig.12(c)).

If there are two or more nets to be deleted and these nets have the same number of intersection then we will retain net k for which the condition

left(kp)/left(kq) = right(kp)/right(kq)

is satisfied or nearly satisfied.

This is as shown in Fig. 13, where
k is number of the net to be considered,
kp is one terminal of net k in Set P1,
kp is the other terminal of net k in Set Q1,
left(kp) is the distance from the left end to kp in Set P1, and
right(kp) is the distance from the right end to kp in Set P1,
left(kq) is the distance from the left end to kq in Set Q1, and
right(kq) is the distance from the right end to kq in Set Q1.

In place of the above formula we would rather use the following criterion to retain net k* which maximizes g(k)

g(k)=sqr(left(kp)*left(kq)) + sqr(right(kp)*right(kq))

From the final bipartite graph in Fig. 12(c) we have

Set P2 ={O, O, O, O, B, C, O, E, F, O, O}
Set Q2 ={B, O, C, O, E, O, F}

It must be pointed out that after deleting nets in the first and second steps Set Q2 also includes null elements "O". For convenience, in the next two sections we will only use the symbol "O" uniformly to express the null elements in Set P2 and Set Q2. Nonnull elements in Set Q2 have the same sequence with that in Set P2. But not all these nonnull elements can be matched in pairs because constrain(2) has not yet been considered .

(3) Third Step: We now consider constraint(2) and find the longest common subsequence from Set P2 and Set Q2

This step is the key part of the whole algorithm. An attempt is made to find an optimum algorithm. Before we introduce the algorithm in section 5, an example will be shown first in Fig. 14. Fig 14(a) is the result from using the simply algorithm by just matching nets from left to right. Fig 14(b) is the result by using an optimum algorithm which yields better result than that in Fig. 14(a).

5. The optimum algorithm for finding the longest common subsequence between two sets in which the nonnull elements are in a same sequence

Let us review this problem as follows: Except for null elements there is no duplicated element in both Set P2 and Set Q2. Set Q2 includes null elements and all the nonnull elements in Set P2. The nonnull elements in Set Q2 have the same sequence as that in Set P2. The problem is how to match Set P2 and Set Q2 to get the longest common subsequence.

5.1 To make the situation precise, let us first introduce several definitions and theorems.

DEFINITIONS

DEF(1) "the matchable point" of an original element

As shown in Fig.15, choose any element in Set Q2 as an original point and match that element with the same element in Set P2. Check the other elements whether they can be matched with the same elements or not. If one can be matched then this element is called "the matchable point" of the original element. In Fig.15, A and F are the matchable points of B; C,D, and E are not matchable points of B. If B is the matchable point of A then the necessary and sufficient condition can be expressed as:

numb{A o B}>= numb{A'o B'}

where
A,B belong to Set P2
A'B' belong to Set Q2
numb{A o B} means the number of null element "O" between A and B

Obviously, if B is the matchable point of A then A is also the matchable point of B (the reciprocal property of matchable point).

DEF(2) "the matchable point matrix M"

As shown in Fig.16(b), a square matrix M is defined such as to express all the information about the matchable points in Fig.16(a). The number of column of matrix M is equal to the number of nonnull elements in set P2 (or Set Q2).

Let M[i, i] = I, which means we use i as the original point.

$$\text{Let } M[i,j] = \begin{cases} 1, & j \text{ is the matchable point of } i \\ 0, & j \text{ is not the matchable point of } i \end{cases}$$

From the first row of M in Fig.16(b) B is the nearest matchable point of A. We call B the first matchable point of A, and D the second , E the third respectively. Obviously, because of the reciprocal property of the matchable point, matrix M is symmetric.

<u>LEMMA</u> (<u>1</u>)

If B and C are two successive matchable points of A as shown in Fig.17(b) and

suppose B is the i th matchable point of A
C is the (i+1) th matchable point of A
D is the first matchable point of B,

then D is also the matchable point of A. In addition either D coincides with C, or D is located at the right side of C.

[Proof]

Because B is the matchable point of A, and D is the matchable point of B, then we have

numb{A o B} >= numb{A'o B'} ;
numb{B o D} >= numb{B'o D'} ;
numb{A o B}+numb{B o D} >=
numb{A'o B'}+numb{B'o D'} ;
i.e. numb{A o D} >= numb{A'o D'} .

Therefore D is the matchable point of A. Because C is the next matchable point after B, D can not be located at the left side of C; otherwise, C can not be the next matchable point.

Therefore either D coincides with C, or D is located at the right side of C (QED).

According to the location of D, B and C can be divided into the following two cases:

Case(1) : C is also the first matchable point of B (Fig.17(a)).
Case(2) : C is not the matchable point of B (Fig.17(b)).

Let us introduce some basic intuitive ideas by means of an example. In Fig. 18(a), Set P2 and Set Q2 are the two sets to be matched. Fig. 18(b) is the matrix M of the sets. A is the starting point. B and C are the first and second matchable points of A. If B and C belong to Case(1), it means that C is also the first matchable point of B. Of course, we just match B as the matching point. Otherwise, if we match C, one matching point B is missed.

But now, matrix M tells us that B and C belong to Case(2). By checking the elements in matrix M, as shown in Fig. 18(b), we find that M[A,B]=1, M[A,C]=1, and M[B,C]=0. It means that B, C are the first and second matchable points of A, but C is not the matchable point of B. It seems to us that now we have two choices, either B or C can be matched as the matching point. Fig. 18(c) and Fig. 18(d) show the two different results. In Fig. 18(d), C is selected as a matching point which leads to a better result than that in Fig. 18(c).

The reason is that, in this case, C has more matchable points than that of B. First, we start from M[B,B] move right, and check the elements in row B. We find that B has only one matchable point F (M[B,F]=1). Then we start from M[C,C], by checking the same way, we find that C has two matchable points D and F (M[C,D]=1, M[C,F]=1). In addition, D, the first matchable point of C is nearer to the original point than that of F which is the first matchable point of B.

The above basic ideas can be summed up as a theorem.

<u>THEOREM(1)</u>

If B and C are two successive matchable points of A, and if B and C belong to Case(2) as shown in Fig. 19, and

suppose B is the ith matchable point of A
C is the (i+1)th matchable point of A
D is the first matchable point of B
E is the first matchable point of C; then
we have

(1) Any matchable point of B is also the matchable point of C, it means that the set of matchable points of B is the subset of that of C.
(only those matchable points which are at the right side of the original point are included in

the set)

(2) Either E coincides with D, or E is located at the left side of D.

[Proof]

(1) First, let D be any matchable point of B. Because D is the matchable point of B and C is not thus we have

numb{B o D} >= numb{B'o D'} ;
numb{B o C} < numb{B'o C'} ;
numb{B o D}-numb{B o C} >
numb{B'o D'}-numb{B'o C'} ;
i.e. numb{C o D} > numb{C'o D'} .

Therefore D is also the matchable point of C.

(2) Now, let D be the first matchable point of B. From (1), D is also the matchable point of C. Because E is the first matchable point of C, then either E coincides with D, or E is located at the left side of D. (GED)

5.2 The optimum algorithm

According to Theorem(1) we have our optimum algorithum as follows:

If an original point has at least two matchable points, then the matching point can be found at either of the following locations:

Location(1)------If the first and second matchable points B and C of an original point A belong to Case(1), as shown in Fig.20(a), then select B as the matching point.

Location(2)------If B,C,D....J,K,L are the first,second..(j-1)th, jth, (j+1)th matchable points of A; and suppose any two successive points from B to K (i.e. BC,CD...JK) all belong to Case(2), only KL belong to Case(1), then select K as the matching point.

Lamma(1) and Theorem(1) are applicable to any two successive matchable points of an original point. Because BC,CD....JK all belong to Case(2), according to Theorem(1) we have

the set of matchable points of B is the subset of matchable points of C,
the set of matchable points of C is the subset of matchable points of D,

.

.

the set of matchable points of J is the
subset of matchable points of K.

So sets of matchable points of X (X=B, C, D....J) are all the subset of matchable points of K. Therefore, when we select K as the matching point, it yields the better result than that of the others.

If we use this algorithm step by step, we get the whole algorithm:

a1 Start from mp1, the starting point, and find mp2, the matching point from either the location(1) or location(2) of mp1;

a2 consider mp2 as the original point, and find mp3, the matching point from either the location(1) or location(2) of mp2 ;

.

.

.

repeat this procedure step by step until the end of the set is reached. It gurantees our getting the largest number of matching points.

5.3 It is easy to sum up the procedure of the optimum algoriyhm by using matrix M :

a11 origin:=starting point;

(Start from starting point.)

a12 i:=origin, j:=origin;

(Start from M[origin,origin]=I.)

a13 repeat j:=j+1
until M[i,j]=1;

(Move right until the first matchable point j is found, where we have M[origin,j]=1.)

a14 i:=j;

(Move downward until reaching the diagonal element, where we have M[i,j]=I. Now i is the first matchable point of origin.)

a15 j:=j+1;

(Move right and attempt to find the first matchable point of i and to check the following three cases.)

a16 if (M[i,j+1]=0) and (M[origin,j+1]=0)
  then goto a15;

(In this case, j+1 is neither the first matchable point of i, nor the second matchable point of origin, Then keep on moving right.)

if (M[i,j+1]=0) and (M[origin,j+1]=1)
  then i:=j+1
       goto a15;

(In this case, j+1 is not the first matchable point of i, but the second matchable point of origin. It means that i and j+1 belong to Case(2), Then keep on checking the second and the third matchable points of origin, to see whether they belong to Case(2) or Case(1).).

if M[i,j+1]=1
  then j:=i
       select i as the matching point
   (the end of one step)

(In this case, j+1 is the first matchable point of i and also the second matchable point of origin. It means that i and j+1 belong to Case(1). Move back to M[i,i]=I, select i as the matching point.)

a17 origin:=i
  goto a12;
  (the begining of the next step.)

An example is shown in Fig.21,which is the matrix M of Fig.14. Fig.21 illustrated how the matching result, as shown in Fig.14(b) is obtained.

At first step, let A be the "origin". Now, B and C belong to Case(1). According to the procedure mentioned above we obtain B as the first matching point.

At the second step, let B be the "origin". Now, CD as well as DE belong to Case(2), and EF belong to Case(1). According to the procedure mentioned above we obtain E as the second matching point.

By using the same way, we obtain F and G as the third and fourth matching points.

5.4 An example is shown in Fig.22. In Fig.22(a), the simple algorithm is used. In the simple algorithm, we just select the first matchable point as the matching point. In Fig.22(b),(c) the optimum algorithm is used starting from E and R respectively.

Table(1)

	algorithm used	starting point	matching points
Fig.22(a)	simple algorithm	E	E, S, F, T, V, W, Y
Fig.22(b)	optimum algorithm	E	E, S, M, N, L, T, V, W, Y
Fig.22(c)	optimum aigorithm	R	E, S, R, N, L, T, V, W, Y

Table(1) compares the results of Fig.22. It is clearly shown that the results of the optimum algorithm is better than that of the simple algorithm. Moreover starting from the different points the results may be different. Therefore by using this algorithm and starting from the different points we can obtain the longest common subsequence.

6. Conclusion

The merging algorithm can be used to solve the two-dimensional problem provided that extra tracks are available at the two ends of a channel.

Three algorithms were proposed. One is for the fixed position problem. The other two are for the fixed sequence problem. The algorithm were coded in PASCAL and implemented on VAX 11/780 computer.

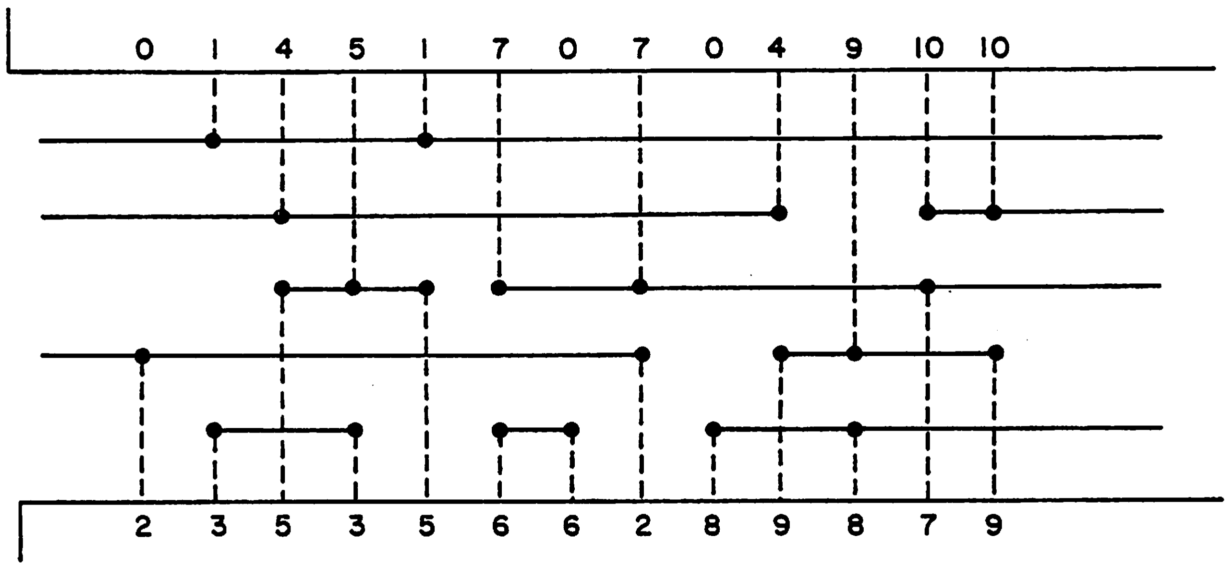
REFERENCES

[1] D. N. Deutsch, C. Persky and D. G. Schwiekert. "LTX-A System for the Directed Automatic Design of LSI Circuits" Proc. 13th Design Auto. Conf. 1976, pp.399-

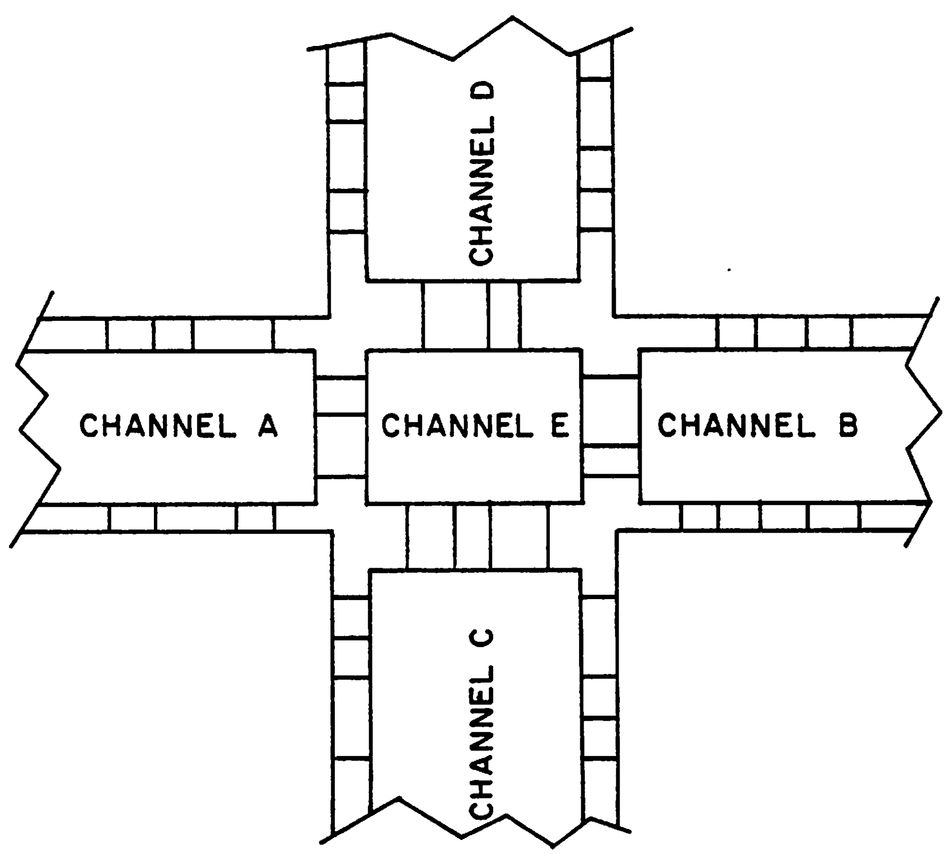
407.

[2] T. Yoshimura and E. S. Kuh "Efficient Algorithms for Channel Routing" UCB/ERL M80/43, Aug. 1980.

[3] Y. K. Chen and M. L. Liu "Three-layer Channel Routing" UCB/ERL M81/84, Nov. 1981.

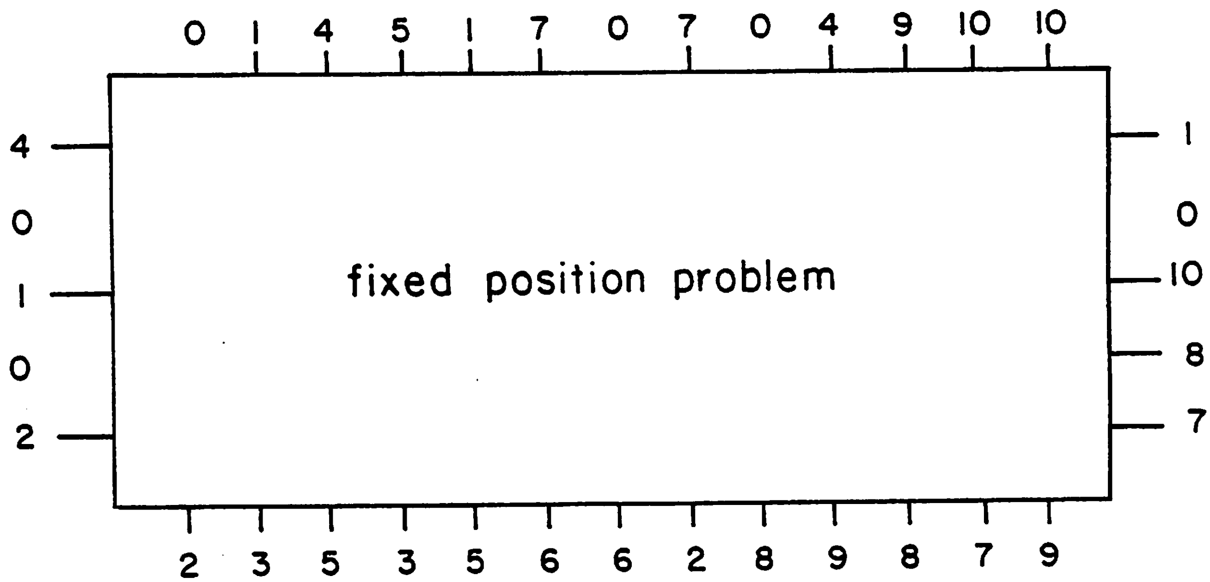

0 1 4 5 1 7 0 7 0 4 9 10 10
2 3 5 3 5 6 6 2 8 9 8 7 9

(a)


CHANNEL D
CHANNEL A
CHANNEL E
CHANNEL B
CHANNEL C

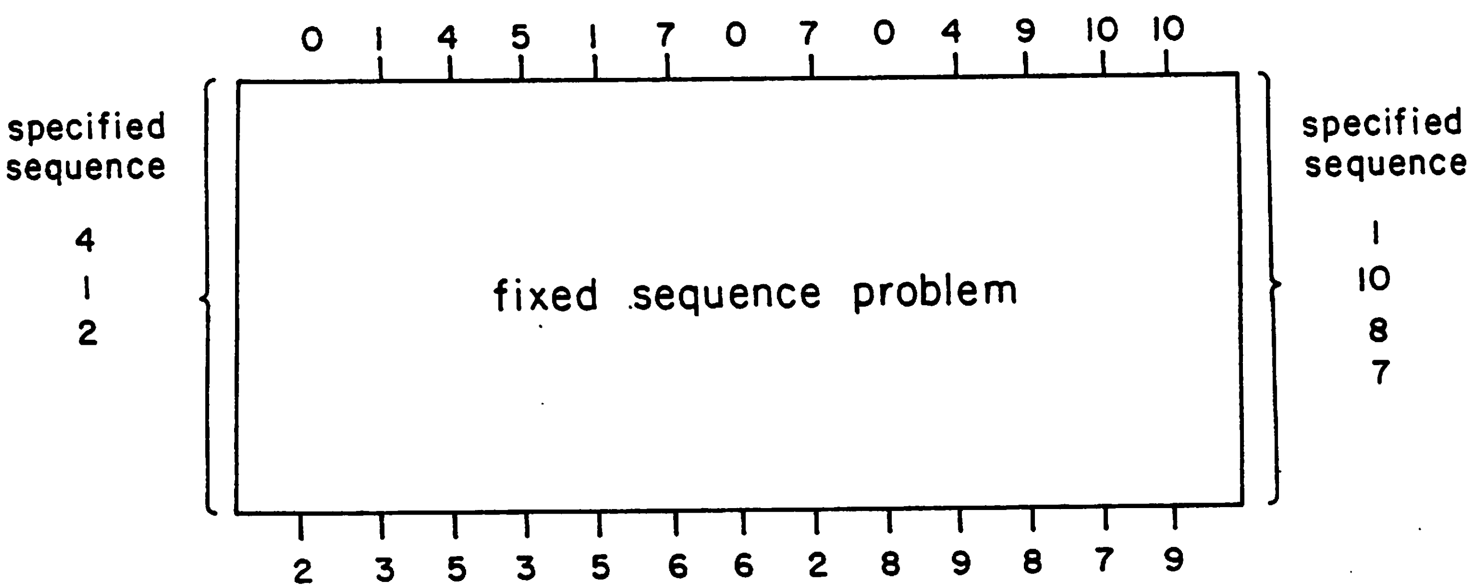
(b)

Fig. 1



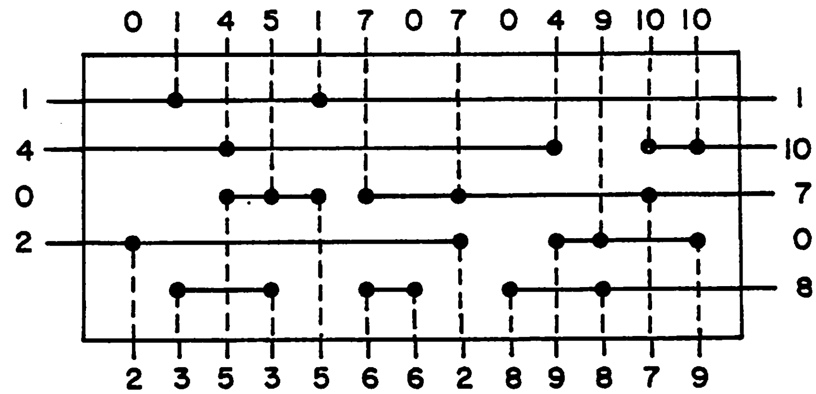
0 1 4 5 1 7 0 7 0 4 9 10 10
4 0 1 0 2
fixed position problem
1 0 10 8 7
2 3 5 3 5 6 6 2 8 9 8 7 9

Fig. 2

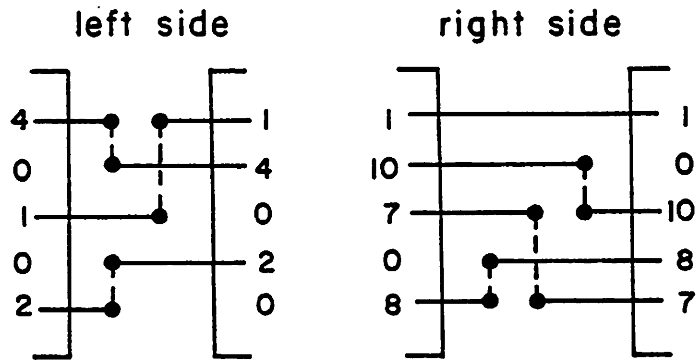


0 1 4 5 1 7 0 7 0 4 9 10 10
specified sequence
4 1 2
fixed sequence problem
specified sequence
1 10 8 7
2 3 5 3 5 6 6 2 8 9 8 7 9

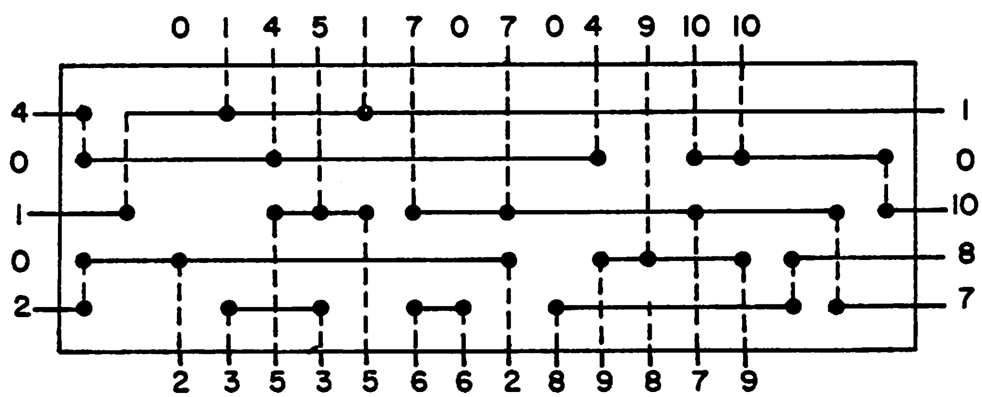
Fig. 3


0 1 4 5 1 7 0 7 0 4 9 10 10
1 4 0 2
1 10 7 0 8
2 3 5 3 5 6 6 2 8 9 8 7 9

(a)


left side
4 0 1 0 2
1 4 0 2 0
right side
1 10 7 0 8
1 0 10 8 7

(b)


0 1 4 5 1 7 0 7 0 4 9 10 10
4 0 1 0 2
1 0 10 8 7
2 3 5 3 5 6 6 2 8 9 8 7 9

(c)

Fig. 4

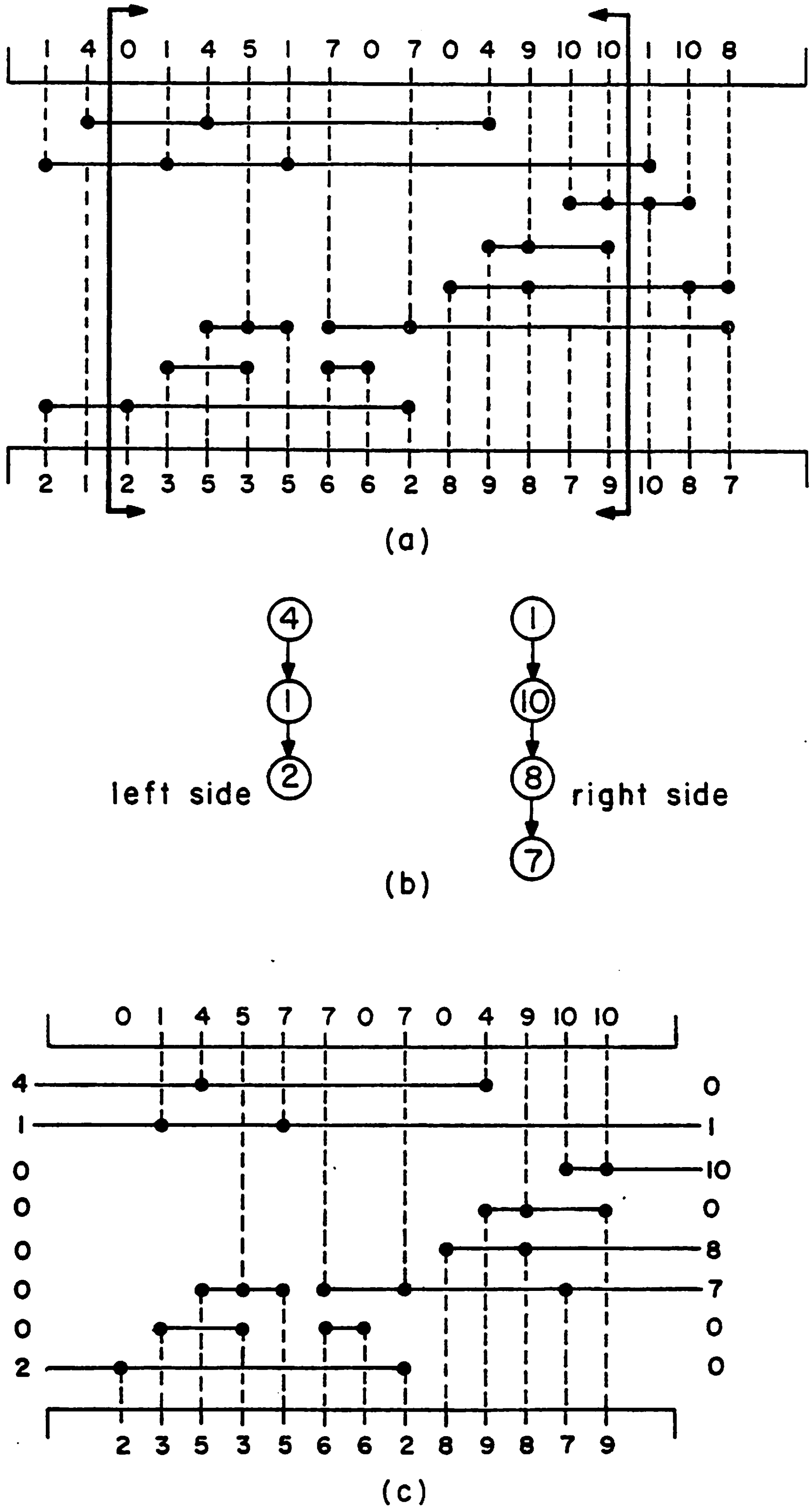

1 4 0 1 4 5 1 7 0 7 0 4 9 10 10 1 10 8
2 1 2 3 5 3 5 6 6 2 8 9 8 7 9 10 8 7
(a)
4
1
2
left side
1
10
8
right side
7
(b)
0 1 4 5 7 7 0 7 0 4 9 10 10
4 1 0 0 0 0 0 2
0 1 10 0 8 7 0 0
2 3 5 3 5 6 6 2 8 9 8 7 9
(c)

Fig. 5

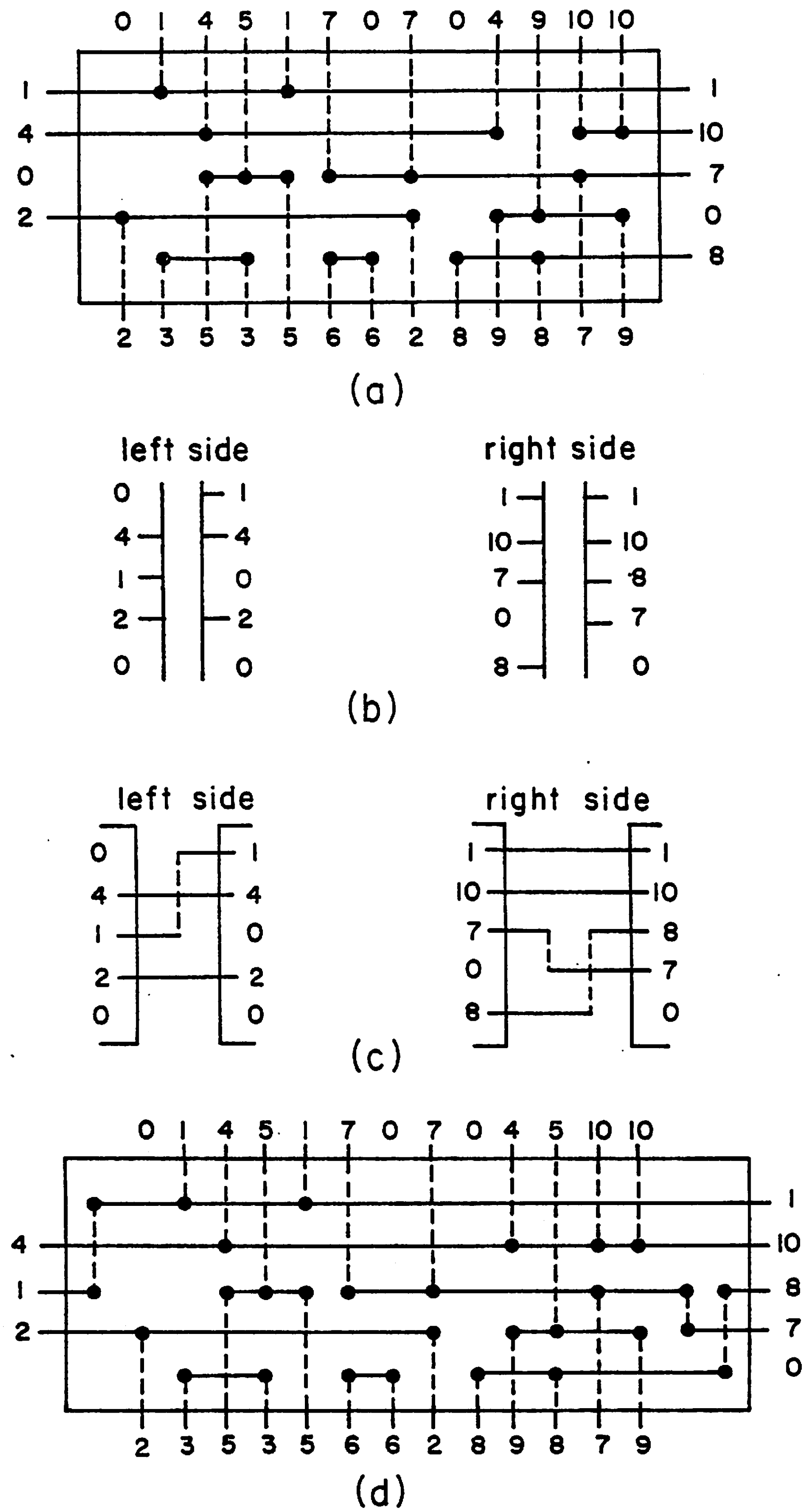
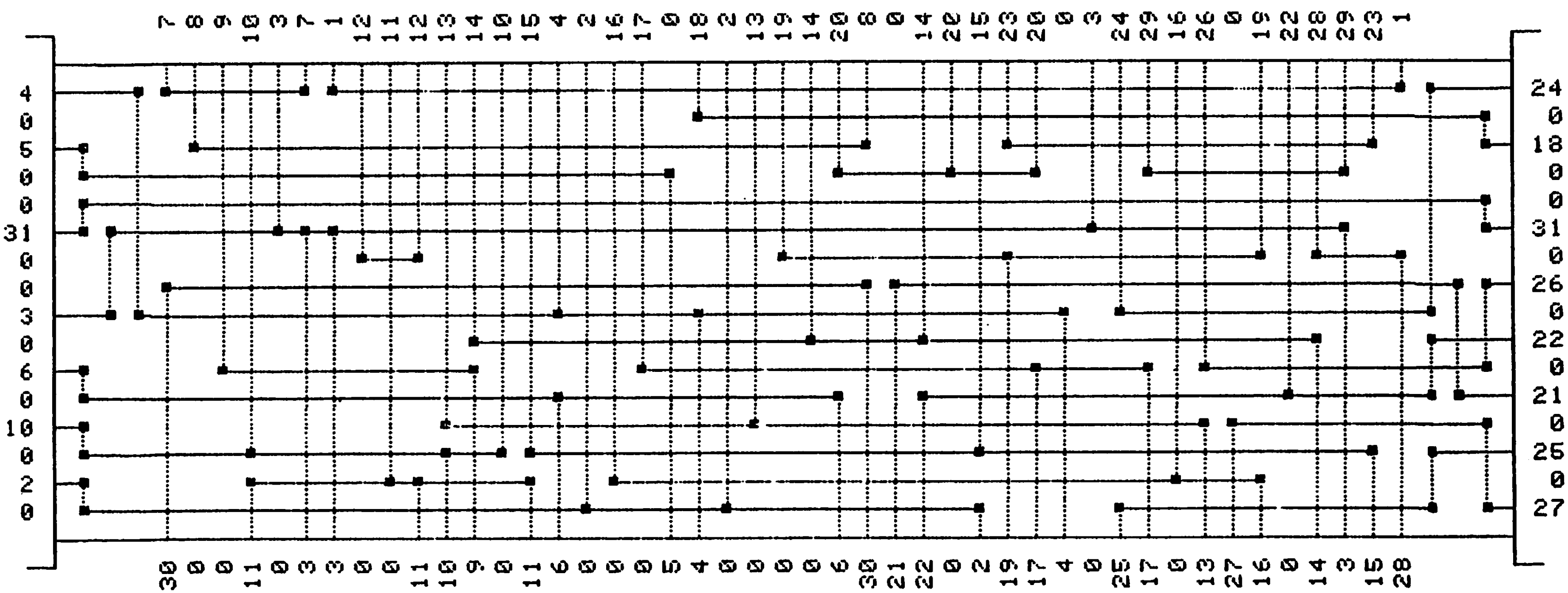

0 1 4 5 1 7 0 7 0 4 9 10 10
2 3 5 3 5 6 6 2 8 9 8 7 9
(a)
left side
right side
(b)
left side
right side
(c)
0 1 4 5 1 7 0 7 0 4 5 10 10
2 3 5 3 5 6 6 2 8 9 8 7 9
(d)

Fig. 6

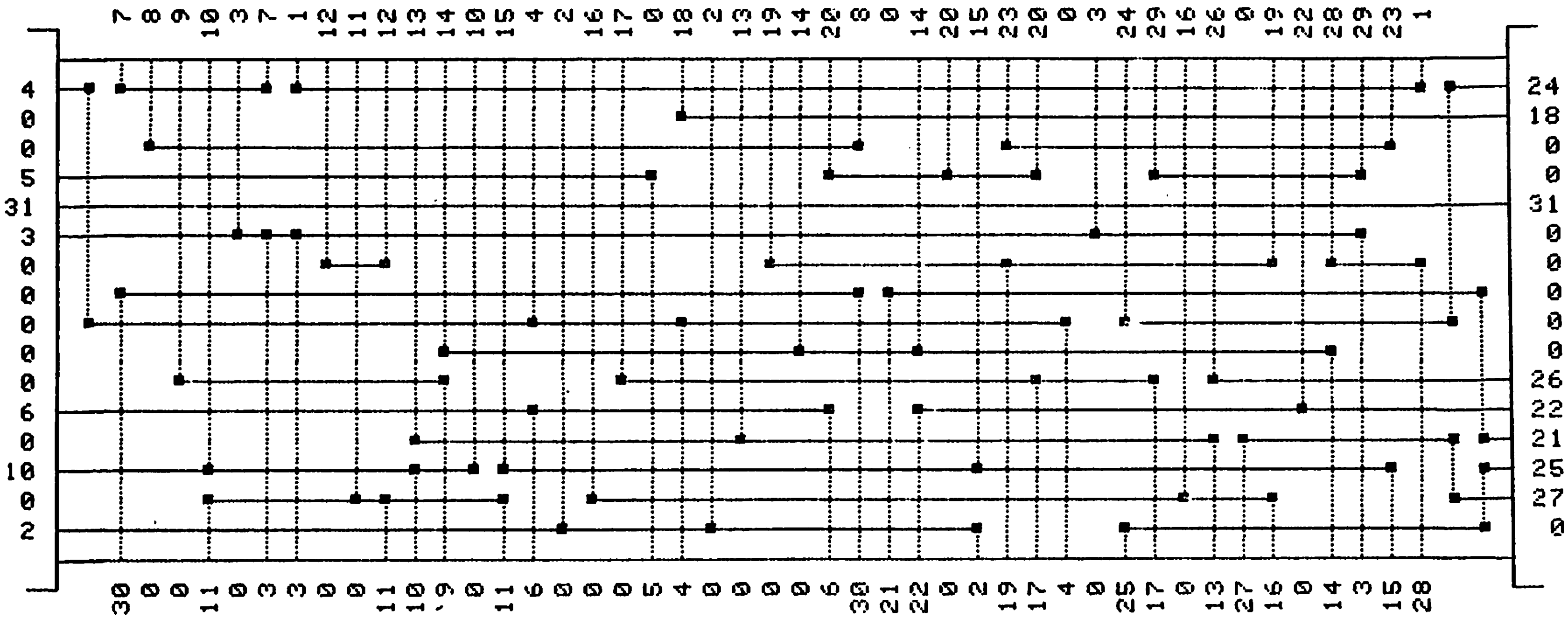
****** example 7 ******



\# column= 60 # zone= 12 before merging vmax= 4 maximum density= 16
\# net= 31 start zone= 4 after merging vmax= 7 number of tracks= 16

Fig. 7

****** example 8 ******

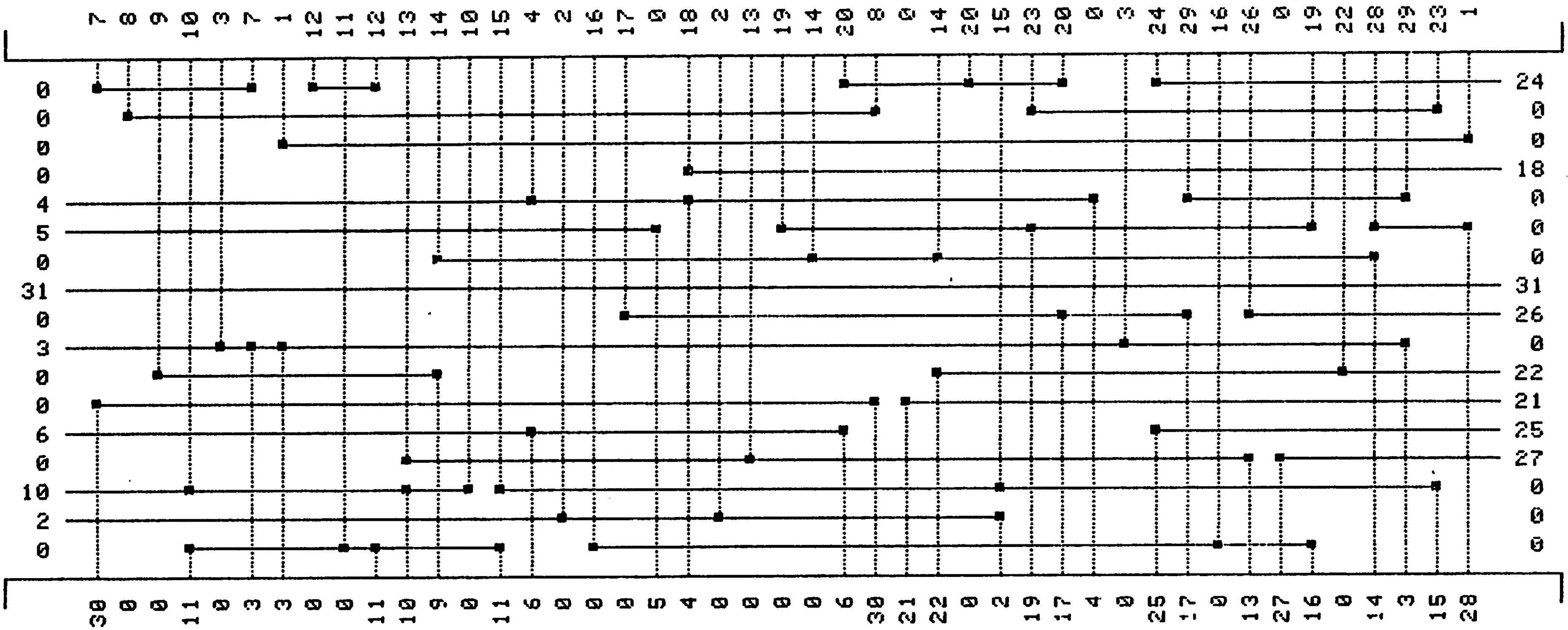


\# column= 60 # zone= 12 before merging vmax= 4 maximum density= 16

\# net= 31 start zone= 5 after merging vmax= 7 number of tracks= 16

Fig. 8

****** example 9 ******



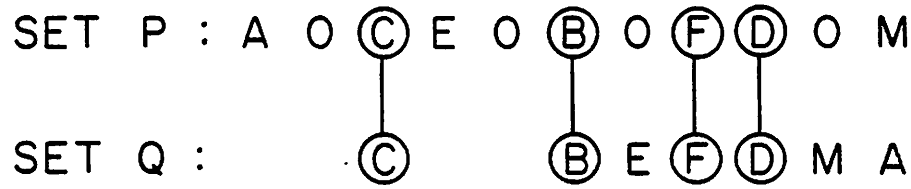
\# column= 60 # zone= 12 before merging vmax= 10 maximum density= 16
\# net= 31 start zone= 11 after merging vmax= 12 number of tracks= 17

Fig. 9

SET P = {A, O, C, E, O, B, O, F, D, O, M}

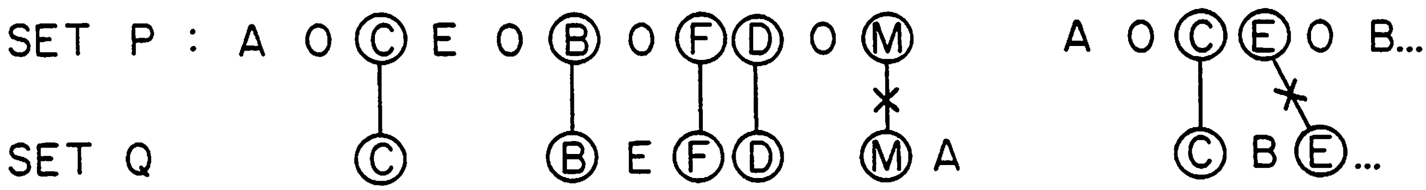
SET Q = {C, B, E, F, D, M, A}

(a)



(b)

Fig. 10



(a) (b)

Fig. 11

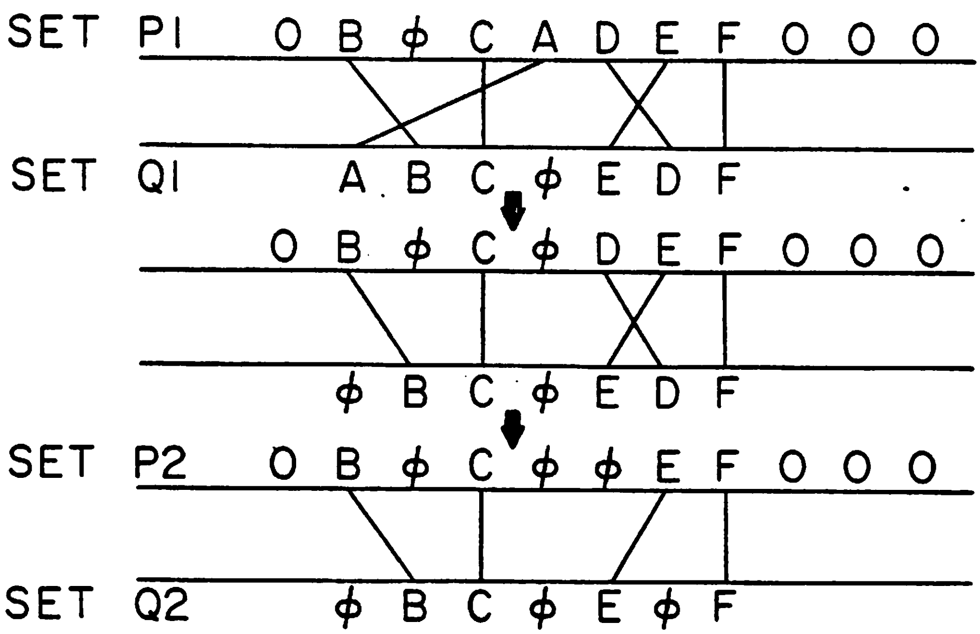


Fig. 12

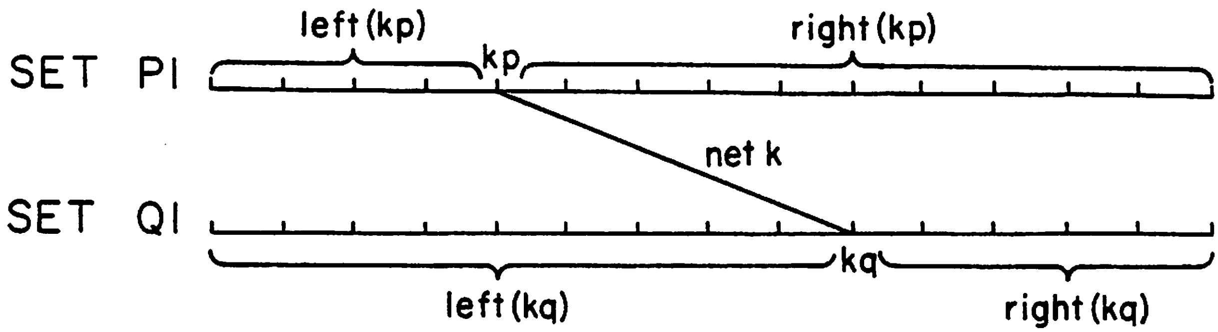
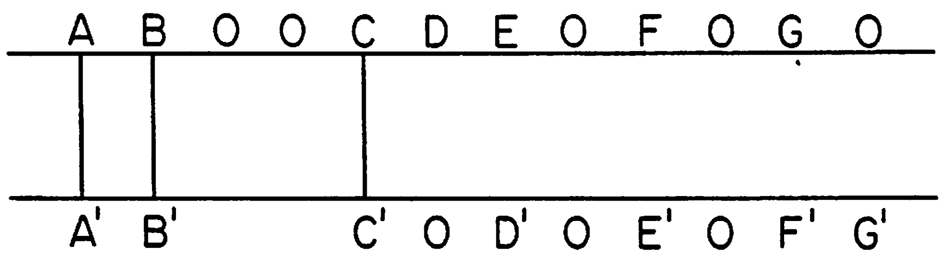
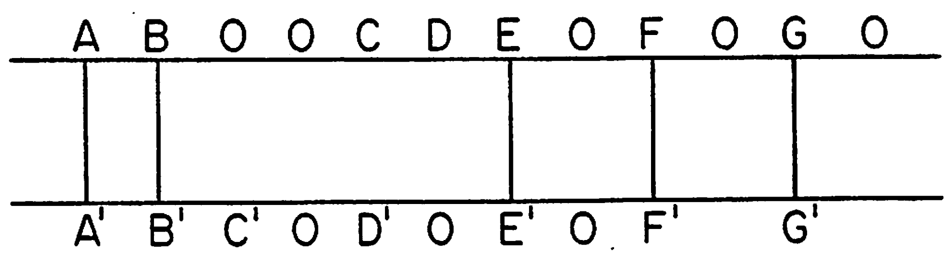


Fig. 13



(a)



(b)

Fig. 14

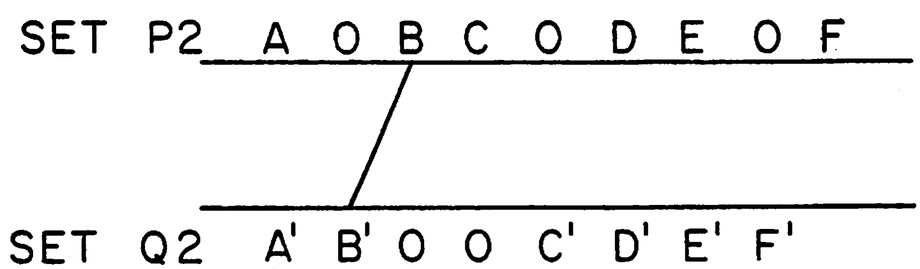
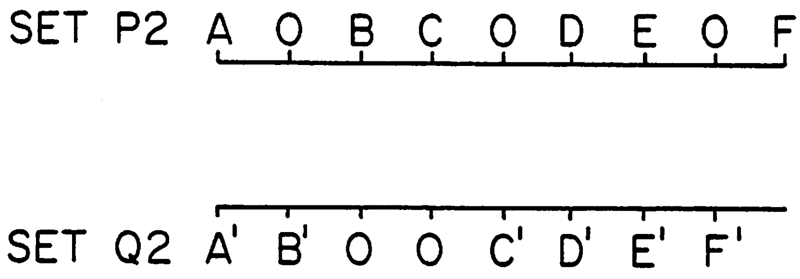


Fig. 15

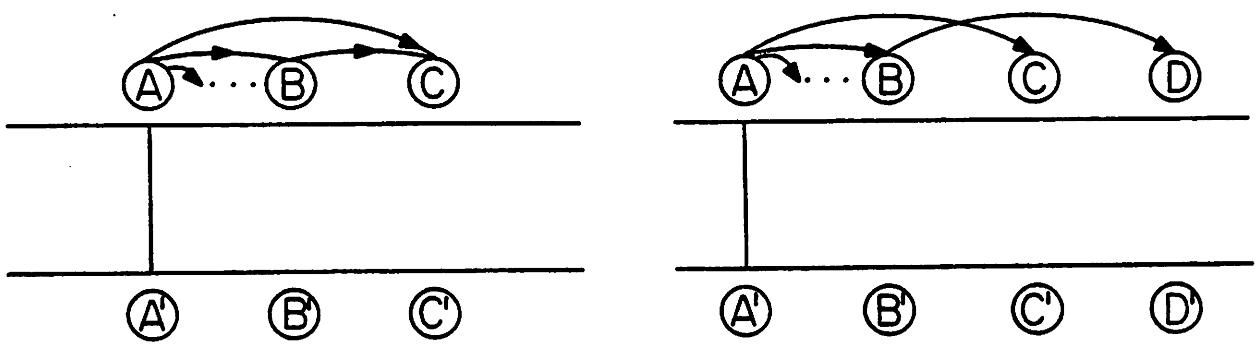


(a)

	A	B	C	D	E	F
A	I	1	O	1	1	1
B	1	I	O	O	O	1
C	O	O	I	1	1	1
D	1	O	1	I	1	1
E	1	O	1	1	I	1
F	1	1	1	1	1	I

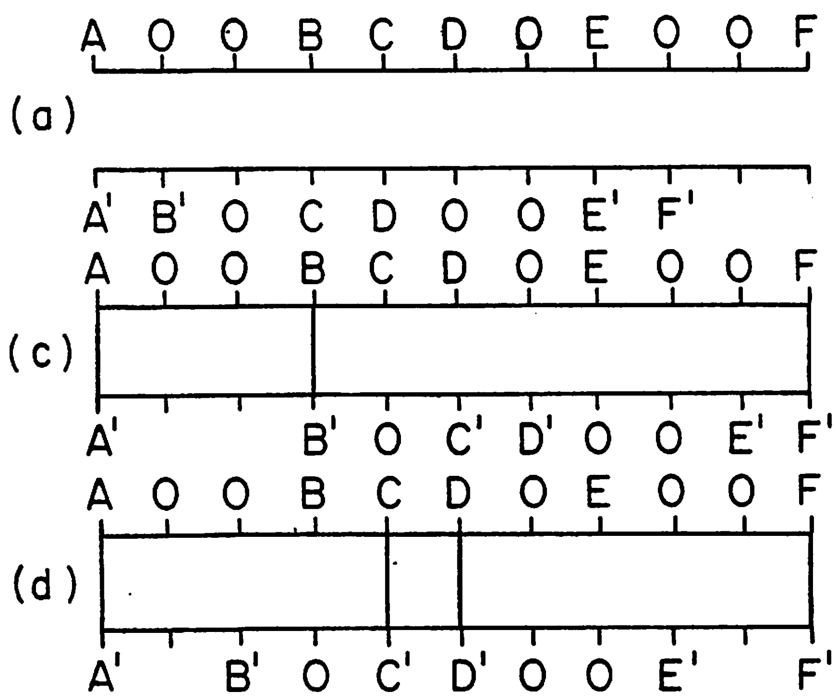
(b)

Fig. 16



(a) (b)

Fig. 17



	A	B	C	D	E	F
A	I	1	1	1	1	1
B	1	I	O	O	O	1
C	1	O	I	1	O	1
D	1	O	1	I	O	1
E	1	O	O	O	I	1
F	1	1	1	1	1	I

(b)

Fig. 18

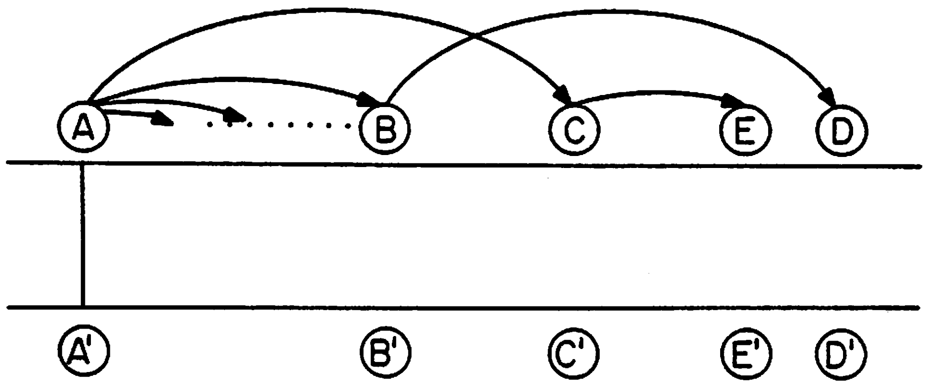
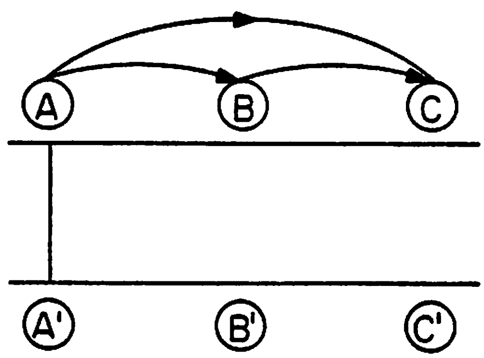
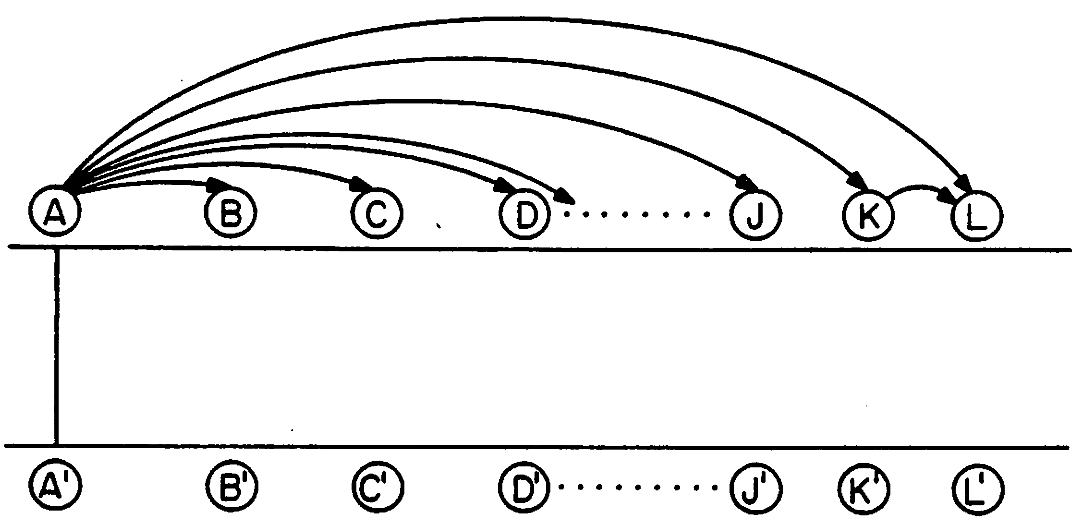


Fig. 19



(a)

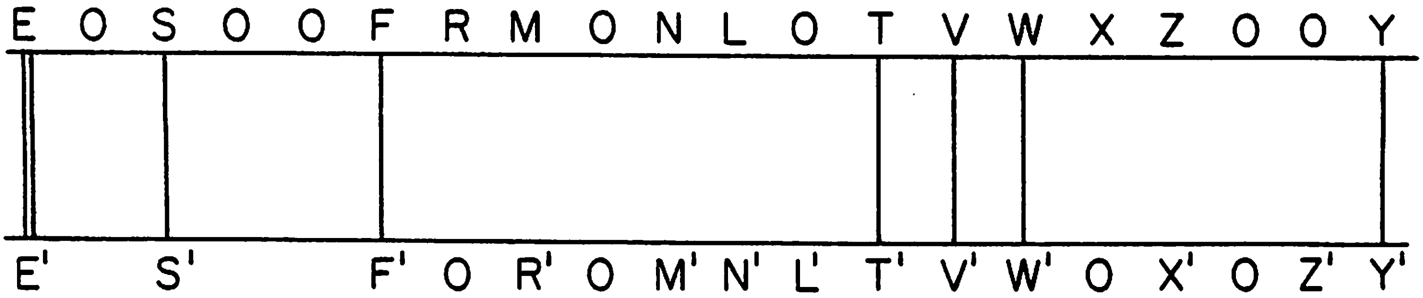


(b)

Fig. 20

	A	B	C	D	E	F	G
A	I	1	1	1	1	1	1
B	1	I	1	1	1	1	1
C	1	1	I	0	0	0	0
D	1	1	0	I	0	0	1
E	1	1	0	0	I	1	1
F	1	1	0	0	1	I	1
G	1	1	0	1	1	1	I

Fig. 21

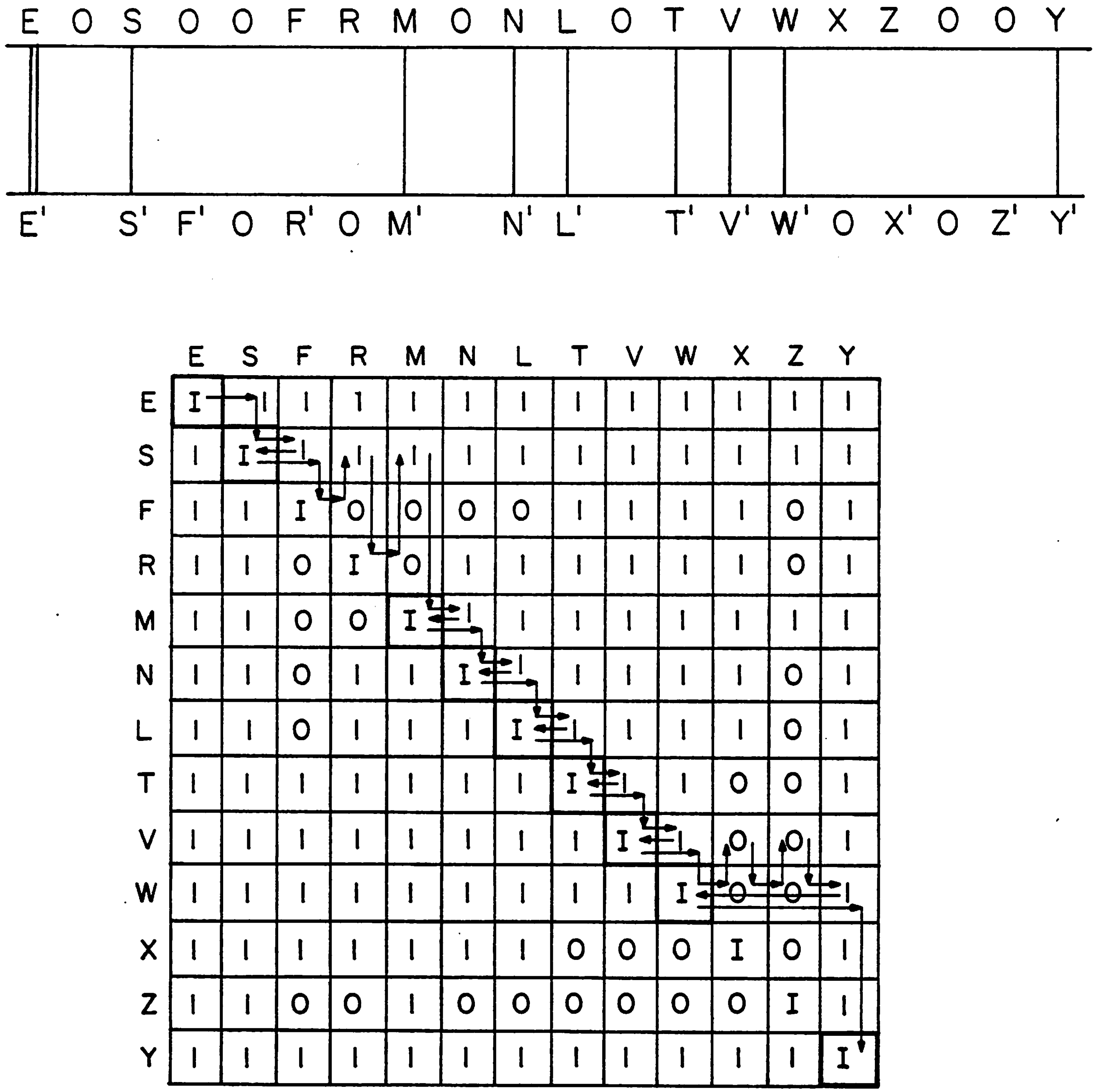


	E	S	F	R	M	N	L	T	V	W	X	Z	Y
E	I	1	1	1	1	1	1	1	1	1	1	1	1
S	1	I	1	1	1	1	1	1	1	1	1	1	1
F	1	1	I	O	O	O	O	1	1	1	1	O	1
R	1	1	O	I	O	1	1	1	1	1	1	O	1
M	1	1	O	O	I	1	1	1	1	1	1	1	1
N	1	1	O	1	1	I	1	1	1	1	1	O	1
L	1	1	O	1	1	1	I	1	1	1	1	O	1
T	1	1	1	1	1	1	1	I	1	1	O	O	1
V	1	1	1	1	1	1	1	1	I	1	O	O	1
W	1	1	1	1	1	1	1	1	1	I	O	O	1
X	1	1	1	1	1	1	1	O	O	O	I	O	1
Z	1	1	O	O	1	O	O	O	O	O	O	I	1
Y	1	1	1	1	1	1	1	1	1	1	1	1	I

Matching points $\longrightarrow$ E, S, F, T, V, W, Y

(a)

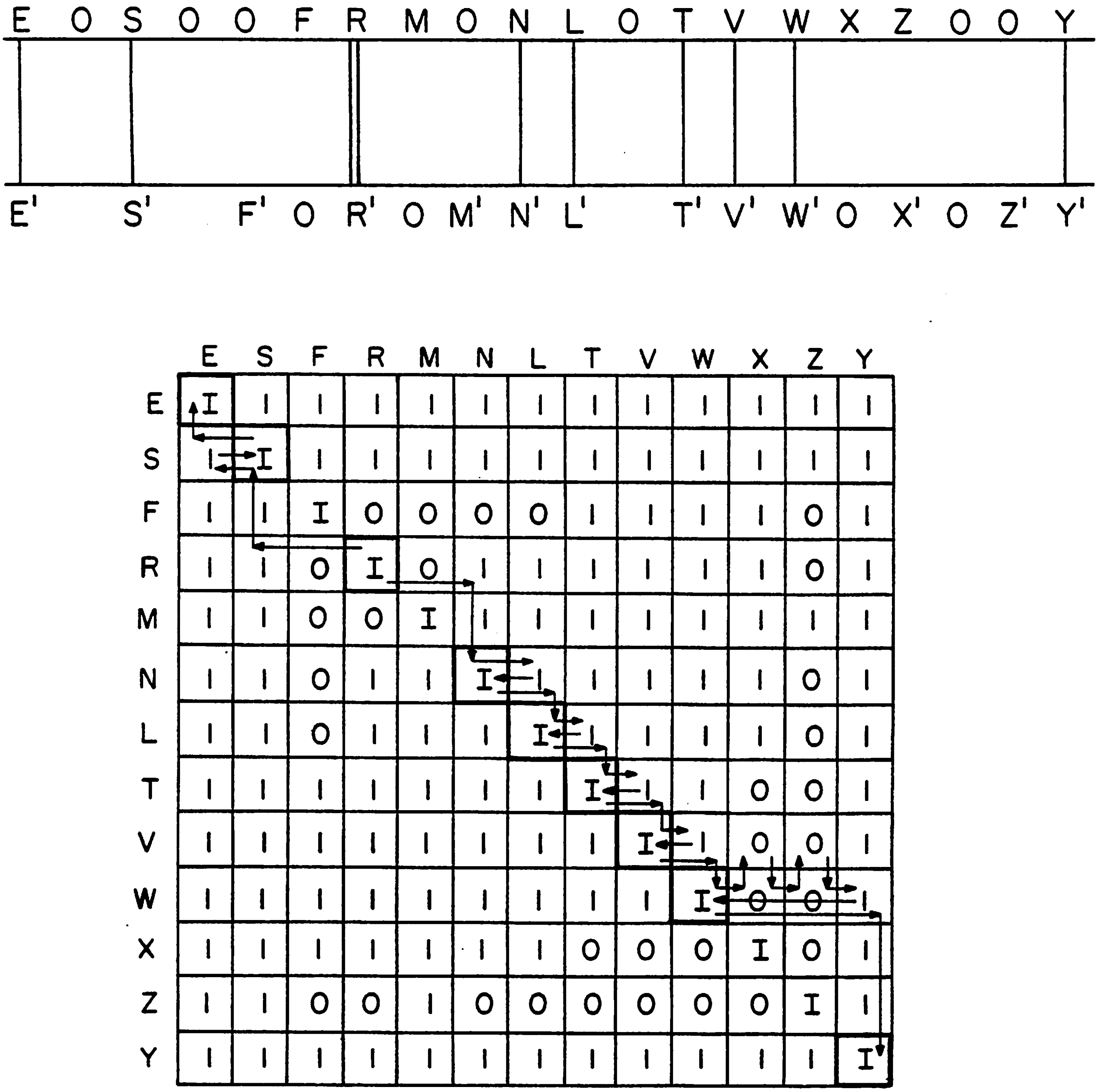
Fig. 22



matching points —— E, S, M, N, L, T, V, W, Y

(b)

Fig. 22



Matching points → E, S, R, N, L, T, V, W, Y

(c)

Fig. 22