N72 101,79

Technical Report TR-167 August, 1971

NSF GJ-754

AN APPLICATION OF CLUSTER DETECTION TO SCENE ANALYSIS

Azriel Rosenfeld
Yung H. Lee

ABSTRACT

Certain arrangements of local features in a scene tend to group together and to be seen as units. It is suggested that in some instances, this phenomenon might be interpretable as a process of cluster detection in a graph-structured space derived from the scene. This idea is illustrated using a class of scenes that contain only horizontal and vertical line segments.

Table of Contents

1. Introduction

2. Clusters

3. Combinations of clusters

4. Possible alternatives and extensions

References

Figures

Appendix: Description of the data structure and programs

A1. Data structure

A2. Programs

A3. Sample output

1. Introduction

The Gestalt psychologists have pointed out [1] that certain arrangements of parts or local features in a scene tend to group together and to be seen as units. In [2], it was suggested that some types of these groupings can be interpreted in terms of spot and streak detectors of various sizes applied to the scene. For example,

a) If there are two or more types of such local features present -- spots and streaks, small spots and large spots, horizontal streaks and vertical streaks, etc. -- then each type may constitute a grouping; this illustrates the Gestaltists' "Law of Similarity".

b) A dense cluster of objects on a sparser background constitutes a grouping, perhaps because it is detected by a single coarse spot (or streak) detector; this illustrates the "Law of Proximity".

c) Objects that lie along a line group together, perhaps because they are detected by a single streak detector; this illustrates the "Law of Good Continuation".

Other types of groupings discussed by the Gestaltists, however, cannot readily be interpreted in this way. Consider, for example, Figure 1. The drawing can be "seen" as two squares sharing a common side, or as a rectangle with a horizontal line across it; but it is considerably harder, if not impossible, to see it as (e.g.) an E with a vertical line down its right side, even though E's are very familiar figures. In any event, none of these groupings readily lends itself to an explanation in terms of (a-c). They all involve both horizontals and verticals, ruling out (a); they are not cluster-like, so that (b) does not apply;

nor can they be accounted for solely in terms of collinearity, so that (c) is insufficient.

The purpose of this report is to suggest an interpretation of certain types of grouping in a scene in terms of a process of clustering in a graph-structured space derived from the scene. This process has been implemented for a class of scenes that contain only horizontal and vertical line segments. The possibility of generalizing it to wider classes of scenes is also discussed.

The clustering procedure to be described below resembles in certain respects the procedure developed by Guzman [3] to "group" regions in a scene that belong to the same polyhedral "body". Guzman makes use of local features such as collinear or parallel segments, L's, T's, etc., to establish "links" between pairs of regions. If a set of regions is linked to a sufficiently strong degree, it constitutes a "body". In the present scheme too, line segments will be linked on the basis of local information (L's, T's, etc.), and sets of segments that have strong mutual linkages will be taken to constitute natural "groupings". An important difference between the two schemes is that Guzman's groupings are essentially connected components (under the transitive closure of the relation "is strongly linked to"), so that distinct groupings cannot have regions in common; whereas in the present scheme, the groupings are clusters, and two clusters can share a segment (e.g., the two squares in Figure 1 have a common side).

It should be stressed that the present procedure is not the only one that can be devised to produce reasonable groupings of parts in a line drawing. The procedure is undoubtedly also much simpler than would be required to account for the wide range of grouping phenomena that can be observed in such drawings. It is presented here only as simple illustration of a class of possible procedures in which groupings in a scene correspond to clusters in a data structure. On the idea that cluster detection processes may play an important role in the functioning of the brain, see [4].

2. Clusters

In this section we describe how graph structures will be derived from line drawings. The nodes of the graph will represent the segments in the drawing; the edges, representing links between pairs of segments, will be constructed as described in the following paragraphs. By a cluster in the graph we shall mean a maximal complete subgraph (MCSG); for a discussion of this and other graph-theoretic cluster concepts, see e.g., [5]. Our objective is to define the graph in such a way that clusters will correspond to natural groupings of segments in the drawing. A more complicated question is that of determining which combinations of these groupings are seen simultaneously in the drawing (e.g., for Figure 1, the rectangle and line should be one such combination, the two squares should be another); discussion of this question will be deferred to Section 3.

Two (noncollinear) line segments will be said to form an L if they have a common endpoint.

Rule 1: Two segments are linked if they form an L

Rule 2: Two segments are linked if they form L's with a third segment, on the same side of it.

For example, by Rule 1, segments x and y would be linked in the case shown in Figure 2a, but not in the case of Figure 2b (since in the latter case the segments from a T, not an L; on the treatment of T's, see below). By rule 2, segments u and v would be linked in Figure 2c, but not in Figure 2d.

In Figure 1, the segment pairs (1,2), (1,4), (2,5) and (4,5) are linked by virtue of Rule 1, and the pairs (1,5) and (2,4) are linked by virtue of Rule 2; but segment 3 is not linked to any other segment. Thus the graph for Figure 1, according to Rules 1-2, is as shown in Figure 3. It is seen that the subgraphs {1,2,4,5} and {3}, corresponding to the rectangle and the central line segment, are indeed MCSG's. However, there is no obvious way of obtaining the two squares ({1,2a,3,4a} and {2b,3,4b,5}) as clusters from this graph; indeed, the segments 2a,2b, 4a,4b are not even represented by nodes. To obtain the squares, we must introduce an additional rule.

Two (noncollinear) segments will be said to form a T if an endpoint of one of them (the "leg" of the T) conincides with a non-endpoint of the other one (the "crossbar" of the T). The two segments will be said to cross (or form a C) if they have a point in common and it is not an endpoint of either of them. The idea underlying our final rule is that T's and C's are ambiguous configurations. One can regard a T as made up of two L's; in other words, one can regard its crossbar as consisting of two (collinear) segments, so that, by Rule 1, the leg is linked to both of these segments. On the other hand, if one regards the crossbar as a single segment, the leg should presumably not be linked to it, since it does not split at the point where the leg meets it. Similarly, a C can be treated in two ways; its two segments can be regarded as split into four at their intersection point, so that they form four L's (and are linked pairwise), or they can be regarded as two unsplit (and hence unlinked) segments.

The ambiguity of T's and C's implies that if a drawing contains them, it can give rise to at least two graphs --

one in which T's and C's are split, the other in which they are not. (Conceivably, one could also consider more complicated possibilities, e.g., all T's split but no C's do, or some individual T splits but no others do, etc.; but it has not been found necessary to introduce such possibilities in the present scheme). In the case of Figure 1, if the T's split, segments 2 and 4 break up into 2a and 2b, 4a and 4b. Since segment 3 now forms L's with these segements, Rules 1 and 2 introduce six new links, and the graph for Figure 1 becomes as shown in Figure 4. Here the MCSG's evidently correspond to the two squares. Thus the ambiguity of Figure 1 can be attributed to the two ways of interpreting the T's that occur in it.

Before summarizing these remarks about T's and C's in the form of a third rule, we must point out one additional complication regarding the treatment of T's. In the foregoing, it has been assumed that when a T splits, the crossbar is replaced by its two parts. Consider, however, Figure 5. Here, if the T's are not split, by Rules 1-2, there are links between the setment pairs (1,2), (1,4), (2,6), (5,6), (3,5), (1,6), (3,6), (2,4), and (2,5). This yields a rather complex graph (Figure 6a; the MCSG's are {1,2,4}, {1,2,6}, {2,5,6} and {3,5,6}). If we split the T's, the graph becomes as shown in Figure 6b; here the upper rectangle corresponds to the MCSG {1,2a,3a,4}, but the lower rectangle is not an MCSG, since its upper side has been split into {3a,3b}.

This problem can be avoided by stipulating that when a T is split, the crossbar is retained in addition to, rather than replaced by, its two parts. When this is done, the parts are linked to the leg of the T, as already indicated, and if the whole crossbar was linked to any segments lying on the same side of it as the leg, the appropriate parts (i.e., the ones that form the L's) are linked to these segments instead. If there were links to segments on the side opposite the leg, however, such segments do not get linked to the parts, but remain linked to the whole crossbar. In Figure 5, for example, when the T crossbar segment 3 is split, its parts 3a and 3b are linked to the leg (4), but the parts are not linked to segment 5, since it lies on the side of 3 opposite the leg; 5 thus remains linked to 3 itself. When segment 2 is split, its parts 2a and 2b become linked to segments 1 and 6, respectively; moreover, 2a is linked to 3a (since it lies on the same side of 3 as the leg 4), but 2b is linked to 3*. The resulting graph is as shown in Figure 6c; in it, both the upper and lower rectangles now correspond to MCSG's ({1,2a,3a,4} and {2b,3,5,6}). [Segment 2 is no longer linked to anything in this graph and so is a one-node MCSG; and {3b,4}, corresponding to the exterior L in Figure 5, is also an MCSG. The significance of these MCSG's to the interpretation of the drawing will be discussed in the next section.]

In Figure 1, when the T's are split, the crossbars are no longer linked to anything, since all of their previously existing links were to segments on the same side of them as the leg. Thus the "split graph" of Figure 4 should

*If 2 is split before 3 is split, both 2a and 2b become linked to 3; but when 3 is then split, the link to 2a is transferred to 3a, since 2a lies on the same side of 3 as the leg. Thus the order in which splitting is performed is irrelevant; in fact, it should be regarded as performed in parallel.

also contain nodes corresponding to segments 2 and 4 themselves, but these are now isolated nodes (and, as will be indicated in the next section, are unimportant to the interpretation of the drawing.) Analogously, whenever a cross splits, the two original line segments are no longer linked to anything, since there are "legs" on all possible sides, so that all possible links get transferred to the parts. In the case of the cross, the original segments can thus be ignored. Similarly, if a segment is the crossbar of T's having legs on both sides of it, the original segment will be linked to nothing after the links have been transferred to the parts. In the examples to be given below, isolated nodes corresponding to crossbars that are linked to nothing will not be shown.

In summary, we can now state

Rule 3: If two segments form a cross, they can either be split into four segments or left unsplit. Similarly, the crossbar of a T can either be split into two segments or left unsplit. In the latter case, links to segments on the side of the crossbar opposite the leg remain attached to the crossbar; all other links are transferred to the appropriate parts.

An alternative idea might be to keep all previously existing links attached to the crossbar as well as transferring them to the appropriate parts. However, this would yield graphs that were more complicated and that had many additional MCSG's. For example, in the case of Figure 1, {2,1,4a}, {2,5,4b}, {2a,1,4} and {2b,5,4} would all be MCSG's. These would presumably be unimportant by virtue of the discussion in the next section, but the added complexity still seems undesirable. One way of suppressing such MCSG's entirely would be to regard a complete subgraph as a cluster only if it is maximal in the drawing; the four MCSG's just listed would be rejected on these grounds, since they correspond to parts of the drawing that are all contained in the rectangle (corresponding to the MCSG {1,2,4,5}), which is thus "bigger" in the drawing than each of the four. However, it seems preferable to work with a definition of cluster that makes reference only to the graph, rather than having to refer back to the drawing. (Moreover, in Figure 9 below, the alternative definition would not reject the four rectangles {1a,2,3,6a}, {1b,3,5,6b}, {1,2a,4,5a} and {2b,4,5b,6}, since they are maximal in the drawing; but these do not seem to be as "good" as the large square or the four small squares. Worse yet, in Figure 11, MCSG's such as {2,1a,3a} would not be rejected.)

Some additional examples of graphs obtained from line drawings using Rules 1-3 are shown in Figures 7-12.

An example of a case that may not be adequately handled by Rule 3 is shown in Figure 13. Here, if no segments split, the MCSG's correspond to the outer square and the two line segments; if all segments split, they correspond to the three rectangles. There is no way to get the two larger rectangles and the small line segment without splitting some T's but not others. However, it is not clear whether this last interpretation of the drawing is as "good" as the first two.

3. Combinations of clusters

In all of the examples given in Section 2, the MCSG's seem to correspond to reasonably good groupings. If a subset of a drawing does not correspond to an MCSG, or union of MCSG's, it is hard or impossible to see this subset as a grouping. The "E" hidden in Figure 1 is an example of this. (In Figure 14, on the other hand, the "E" is a union of MCSG's ({2,3,5} and {4}) in the unsplit graph, and indeed is much less well hidden.)

However, not all of the clusters obtained by the method of Section 2 correspond to equally good groupings. For example, in Figure 6c one of the MCSG's ({3b,4}) corresponds to an expterior "L" in Figure 5; but this L is certainly not as good a grouping as are the two rectangles ({1,2a,3a,4} and {2b,3,5,6}). A similar remark applies to Figure 10; here the split graph has four MCSG's ({2a,4a}, {3a,4c}, {2c,7a}, and {3c,7c}) that correspond to exterior L's, but these are certainly not as good as the five squares ({1,2a,3a,4b}, {2b,4a,5,7a}, {2b,3b,4b,7b}, {3b,4c,6,7c}, and {2c,3c,7b,8}).

No attempt will be made here to formulate precise rules for determining the relative goodness of groupings. However, a few general guidelines and illustrative examples can be given.

It seems reasonable to assume -- e.g., on grounds of information compression -- that if each of two collections of groupings completely covers the figure, the smaller of the two collections should be preferable to the larger one. Let us suppose that in Figure 1, the two interpretations (rectangle + line, square + square) are equally good, since each involves just two MCSG's. Now consider Figure 15. Here the rectangle and line (which are still the MCSG's of the unsplit graph) still cover the drawing; but the two squares (which are MCSG's of the split graph, but not the only ones) no longer cover the drqwing completely. Thus one might expect that the rectangle and line should be a better combination of groupings for Figure 15 than would be combinations involving the two squares (since the latter combinations require more parts), and this does indeed seem to be the case.

On similar grounds, it seems reasonable to assume that large groupings (i.e., large MCSG's) should be preferable to small ones. A class of examples which may be related to this idea is provided by Figures 6c, 10, etc.; here the L's are less noticeable than the squares, perhaps because they correspond to smaller graphs. Note that in each of these cases the L's are also not needed to cover the drawing, since the squares already cover it. However, this alone would not necessarily suffice to make the L's less conspicuous; in Figure 10, the center square is not needed to cover the drawing either (and in fact, the drawing <u>can</u> be seen as four touching squares surrounding a hole, rather than as five squares), yet it is still far easier to see the center square than it is to see the L's.

A small grouping can apparently be "overcome" by a large one even if it is not completely covered by large ones. As an example, consider Figure 16. Here the split graph yields the rectangle and an L ({3,4b}) as clusters; but one sees the drawing as consisting of a rectangle and a line, not a rectangle and an L. Apparently, since one segment of the L belongs to the rectangle, the larger cluster "overcomes" the smaller one, i.e., breaks it apart, so that only the line segment (4b) remains unaccounted for.*

As a final example, consider Figure 17. Here the arches that are open upward can be seen as "stalactites", or the arches that open downward can be seen as "stalagmites"; but only one of these interpretations is possible at a time. This suggests that the clusters for this drawing (the three-node MCSG's shown in the figure) cannot all be seen at once; only alternating ones ({1,2,3}, {5,6,7}, and {9,10,11}, *or* {3,4,5}, {7,8,9}, and {11,12,13}, but not both) can be seen at any given time. In the split graph of Figure 8, on the other hand, the four-node MCSG's corresponding to the squares *can* all be seen at once. In other words, the segments in Figure 8 can have links in opposite directions simultaneously, but the segments in Figure 17 cannot. A possible explanation of this phenomenon might be that four-node MCSG's are so "strong" that they cannot be "overcome" even by others of their own kind; compare the case of the central square in Figure 10.

*In Figure 7, similarly, one can see the drawing as consisting of the two long vertical lines (present, but not shown, in the split graph) and the two rectangles, perhaps because the incomplete rectangles are pulled apart by the complete ones; the long lines are then preferable to the four short segments because fewer of them are needed to account for the remainder of the drawing.

4. Possible alternatives and extensions

The method used in this report to define the graphs of line drawings is certainly not the only possible one. For example, one could consider a scheme in which pairs of line segments are regarded as "surrounding" areas (e.g., the areas obtained by joining the endpoints of the segments), and pairs of these areas are linked if they overlap. However, such schemes do not seem to lead to simpler rules than the scheme used above.

The definition of cluster used in this report is also not the only alternative. For example, there is another standard class of definitions based on the notion of k-connectivity: a subgraph is k-connected if it cannot be disconnected by deleting fewer than k of its nodes. Suppose, in fact, that we took clusters to be maximal 2-connected subgraphs rather than maximal complete subgraphs. At first glance, this would seem to make it possible to simplify the rules used to define the graph; indeed, if Rule 2 were eliminated, a rectangle would no longer have a complete graph, but its graph would still be 2-connected. However, consider Figure 9; here the entire split graph (with the edges due to Rule 2 deleted) is 2-connected, so that the squares are not maximal 2-connected subgraphs.

Another possibility for modifying the present scheme is with regard to the types of links used. For example, one could consider linking two segments if they are parallel and aligned at (one or both of) their ends. The strength of such links would presumably have to vary with the lengths of the segments and the distance between them; in Figure 18a, for example, the close pairs of parallels seem to be much more strongly linked than the distant pairs, since it is almost impossible to group the parallels in any other way than into close pairs. Note, however, than in Figure 18b, it is almost as easy to group the more distant pairs as it is to group the close pairs [1]. This suggests that the links introduced when L's (and T's) are present may be, in some sense, much "stronger" than the links due to parallelism, since the linkages between the close pairs of parallels are"broken"as soon as the connecting lines are added. It may therefore not have been unreasonable to have ignored parallelism links in constructing the scheme described in this report.

In principle, it should be possible to generalize the present scheme to wider classes of scenes, involving line segments at arbitrary orientations, or even arcs and curves. For example, one might segment a curvilinear drawing at curvature maxima ("angles"), and link pairs of segments that meet at an angle (Rule 1), or that meet a third segment, on the same side of it, at angles (Rule 2). T's and crosses in the present scheme would correspond to branch points ("Y-junctions", "X-junctions", etc.) in the general case. Such generalizations are currently being investigated, and will be the subject of a future report.

References

1. M. Wertheimer, Investigations on the Gestalt theory (in German), Psychol. Forschung 4, 1923, 301-350.

2. A. Rosenfeld and M. Thurston, Edge and curve detection for visual scene analysis, IEEE Trans. C-20, 1971, 562-569.

3. A. Guzman, Decomposition of a visual scene into three-dimensional bodies, Proc. Fall Joint Computer Conf., December 1968, 291-304.

4. D. Marr, A theory for cerebral neocortex, Proc. Royal Soc. B176, 1970, 161-234.

5. J. G. Augustson and J. Minker, An analysis of some graph theoretical cluster techniques, J. ACM 17, 1970, 571-588.

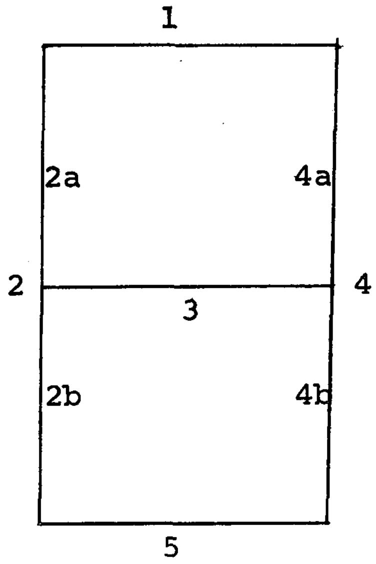


Figure 1. Ambiguous figure.

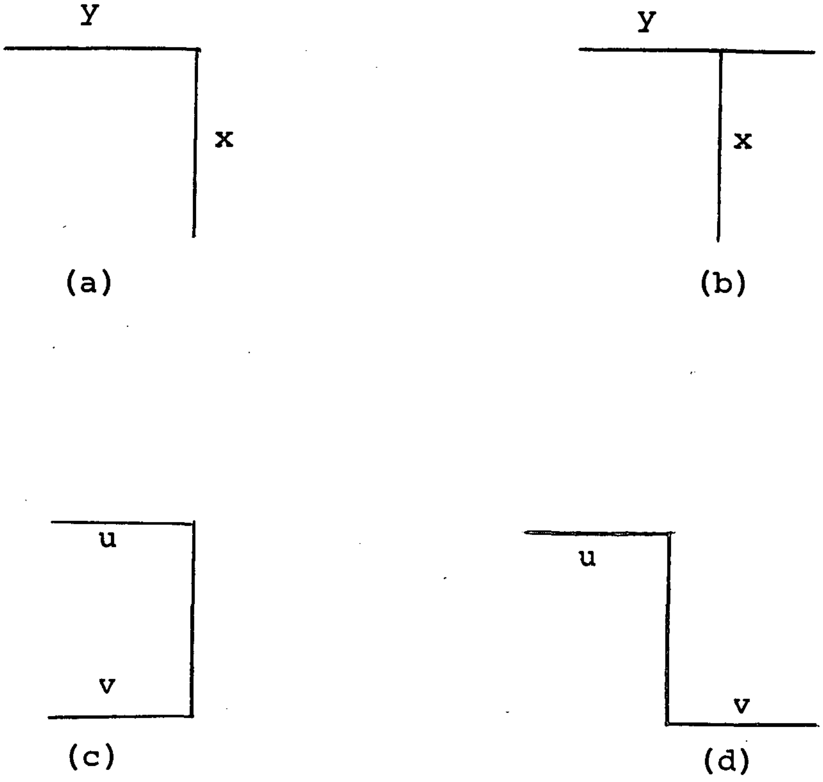


Figure 2. Illustrations of Rules 1-2.

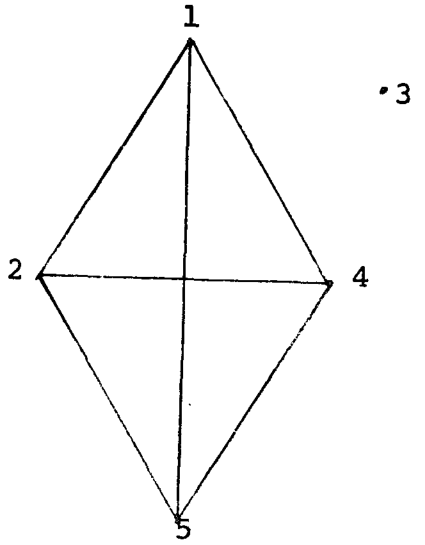


Figure 3. Graph corresponding to Figure 1 according to Rules 1-2.

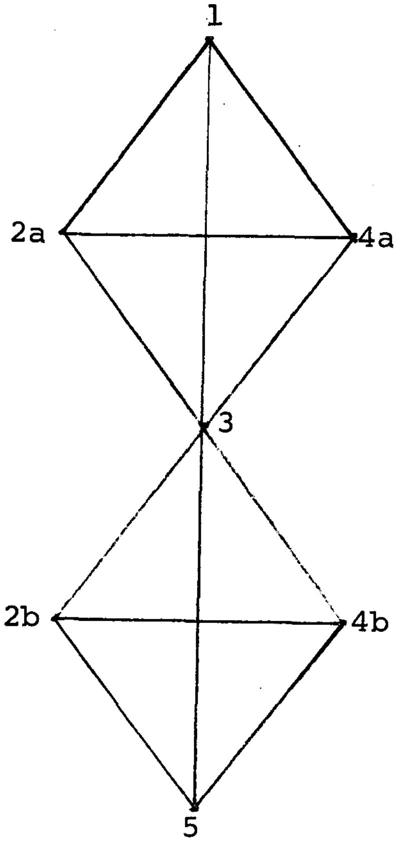


Figure 4. Graph corresponding to Figure 1 when the T's are split.

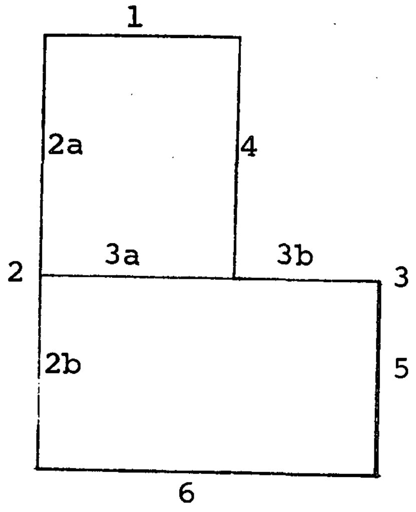
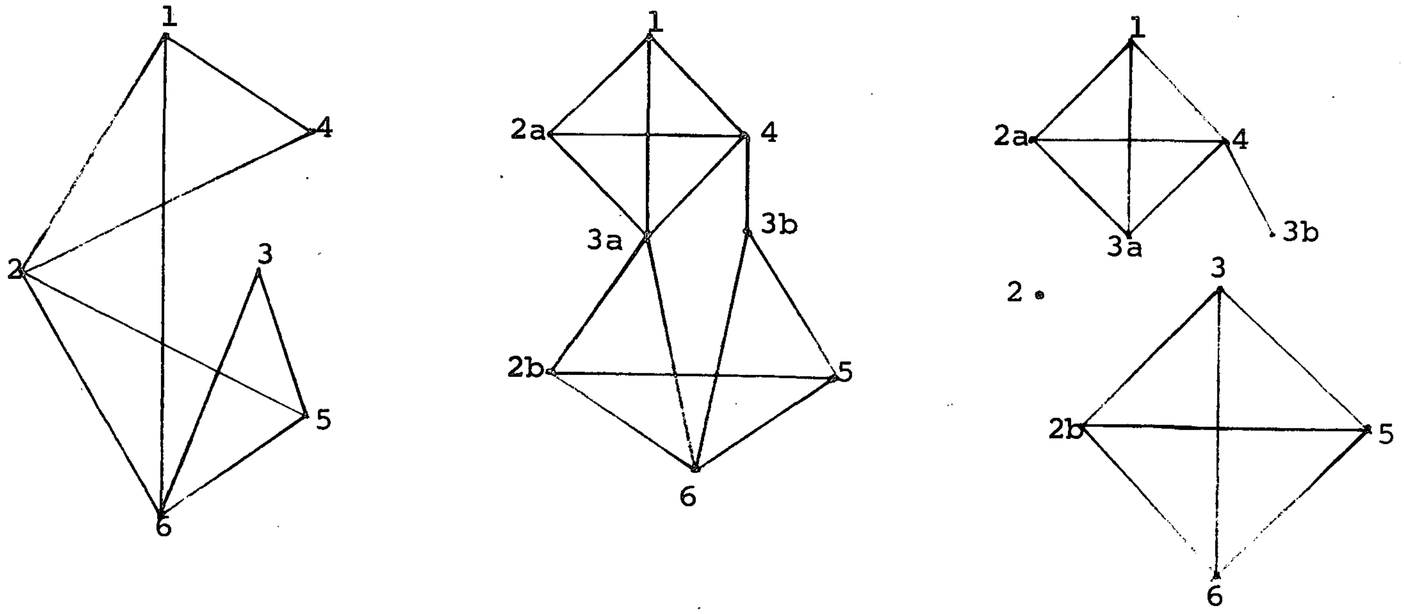


Figure 5. Example illustrating the need for "one-sided" splitting.



(a) T's unsplit

(b) T's split

(c) T's split "one-sidedly"

Figure 6. Graphs corresponding to Figure 5.

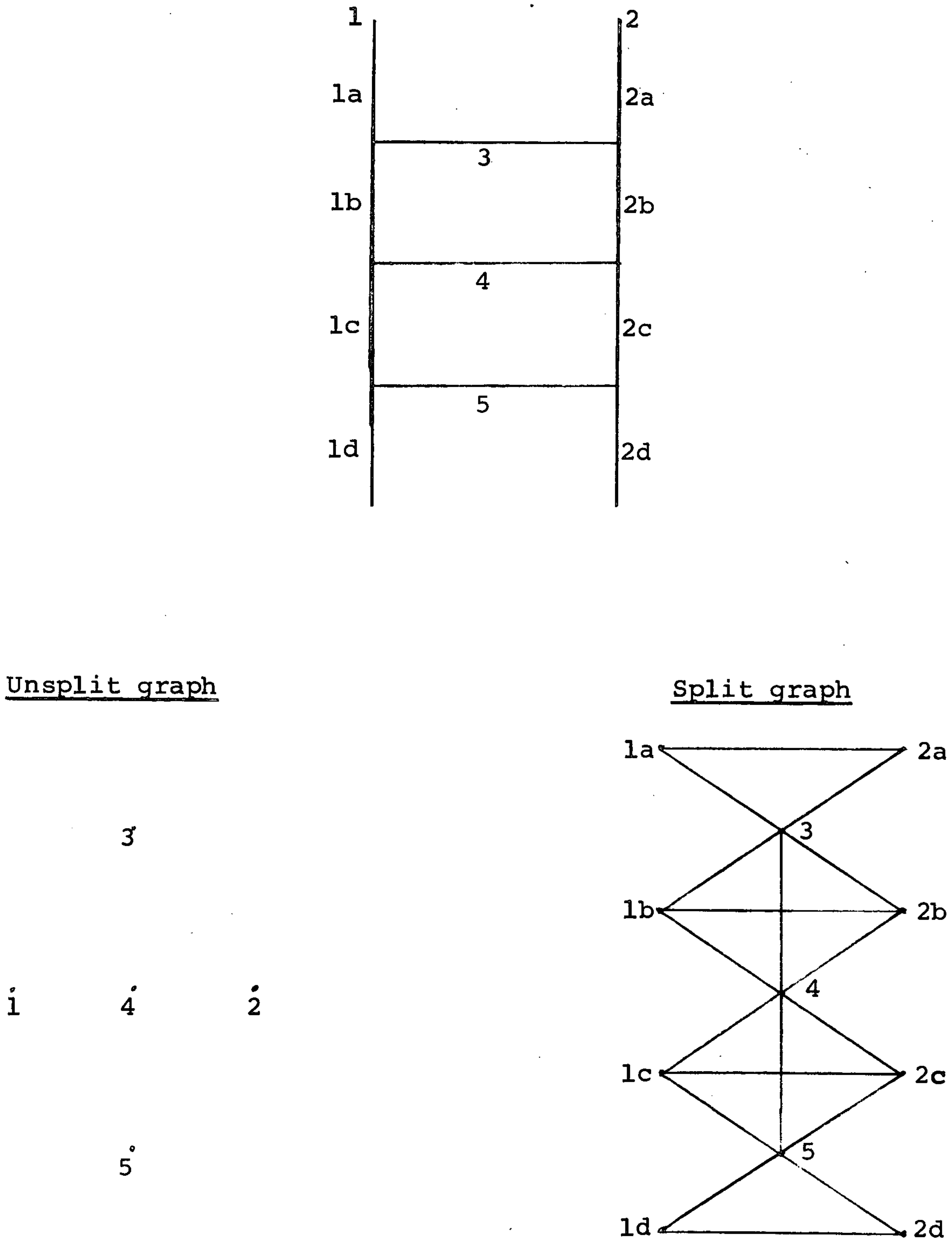


Figure 7. The unsplit graph yields disjoint lines; the split graph yields two squares and two incomplete squares.

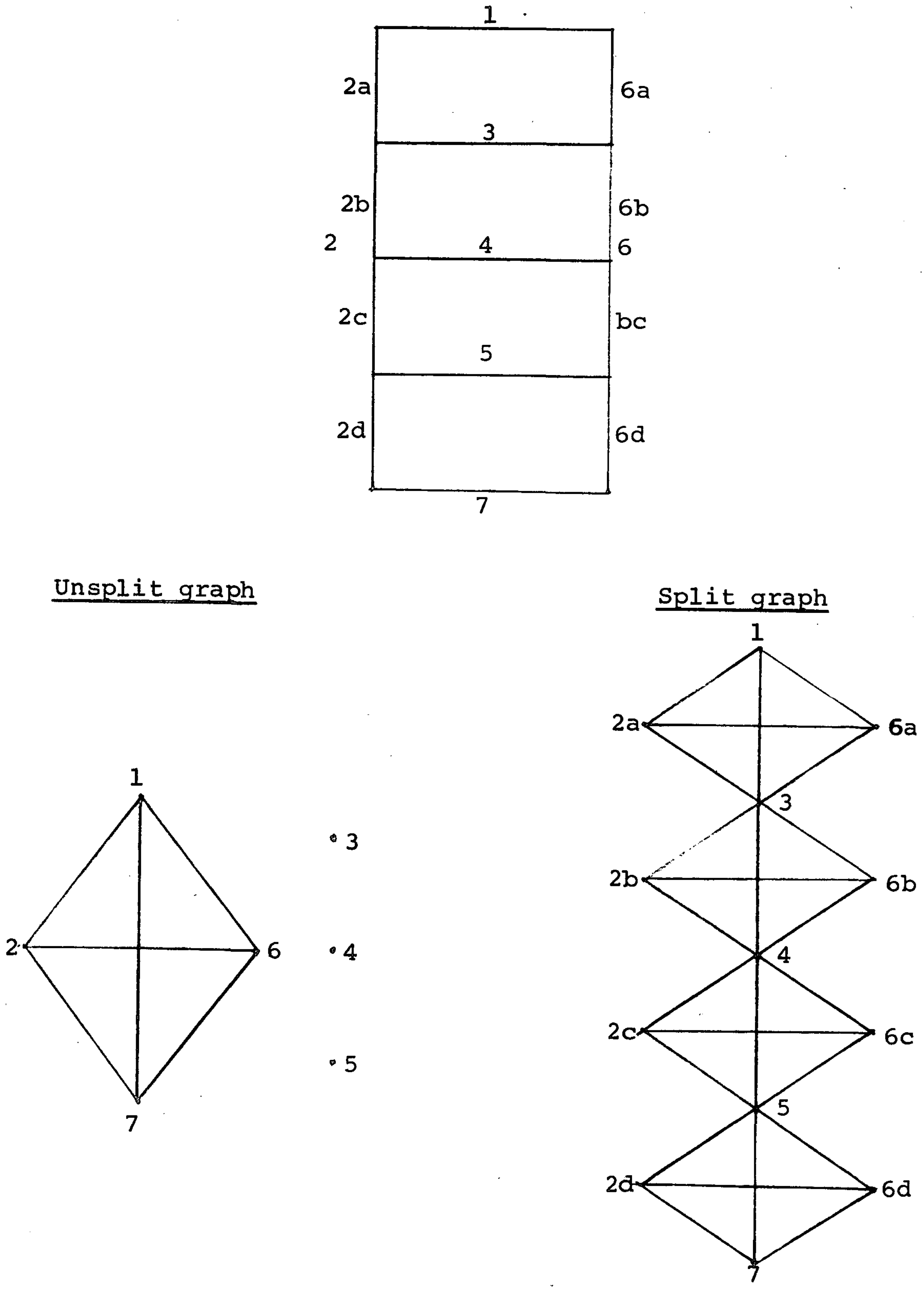


Figure 8. The unsplit graph yields the rectangle and the parallel lines crossing it; the split graph yields the four squares.

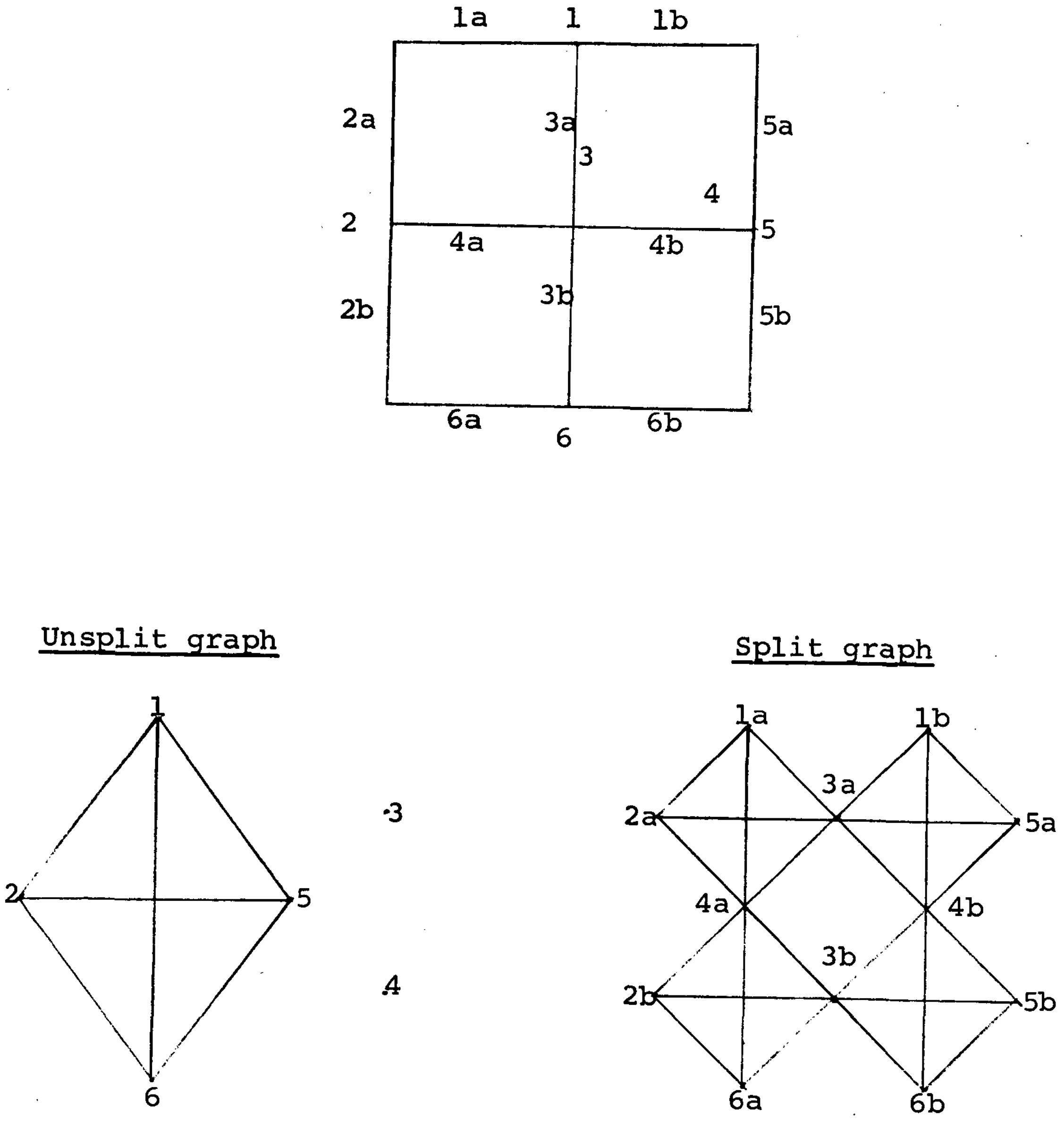


Figure 9. The unsplit graph yields the outer square and the cross; the split graph yields the four small squares.

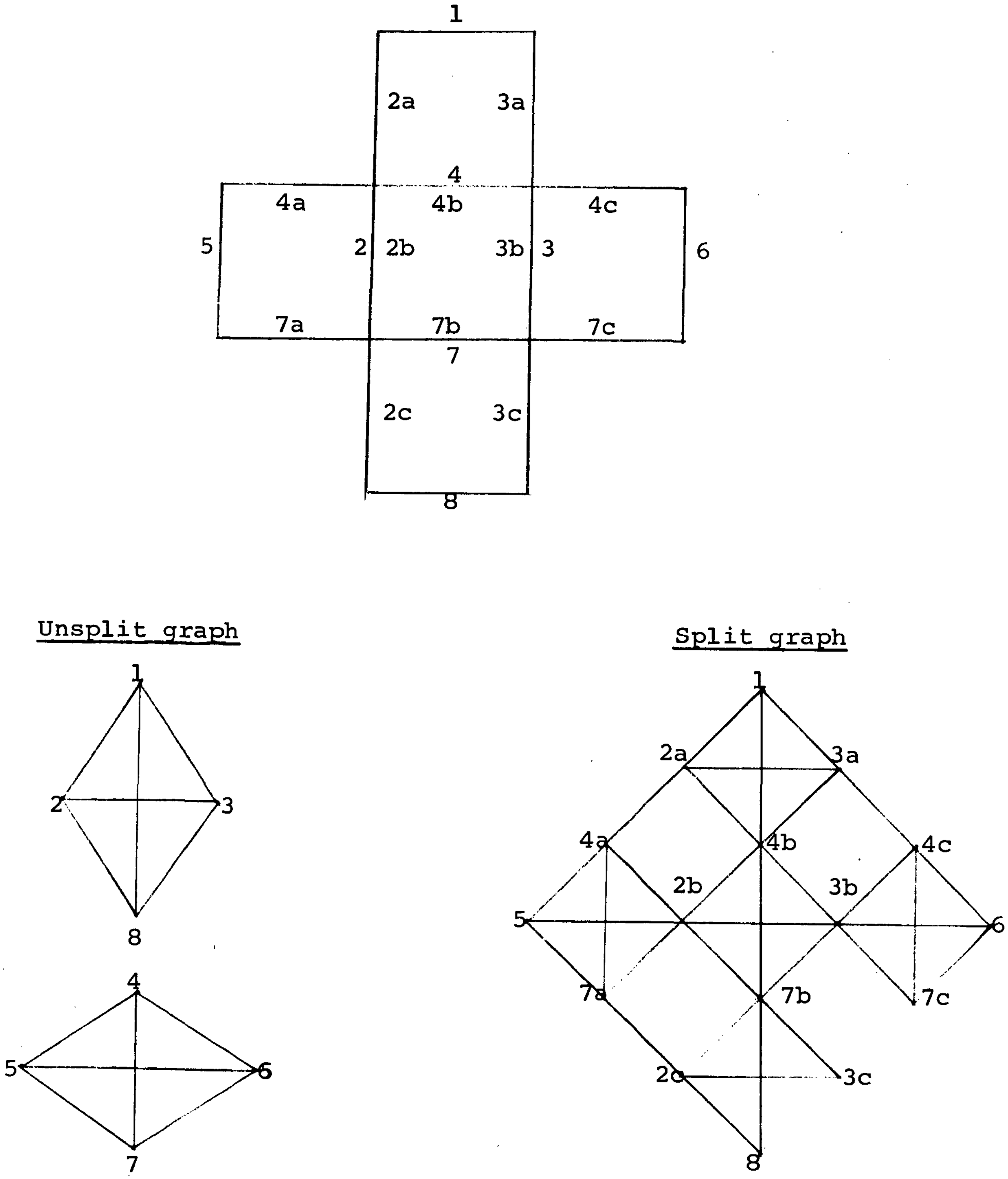


Figure 10. The unsplit graph yields the two rectangles; the split graph yields five squares (together with four L's).

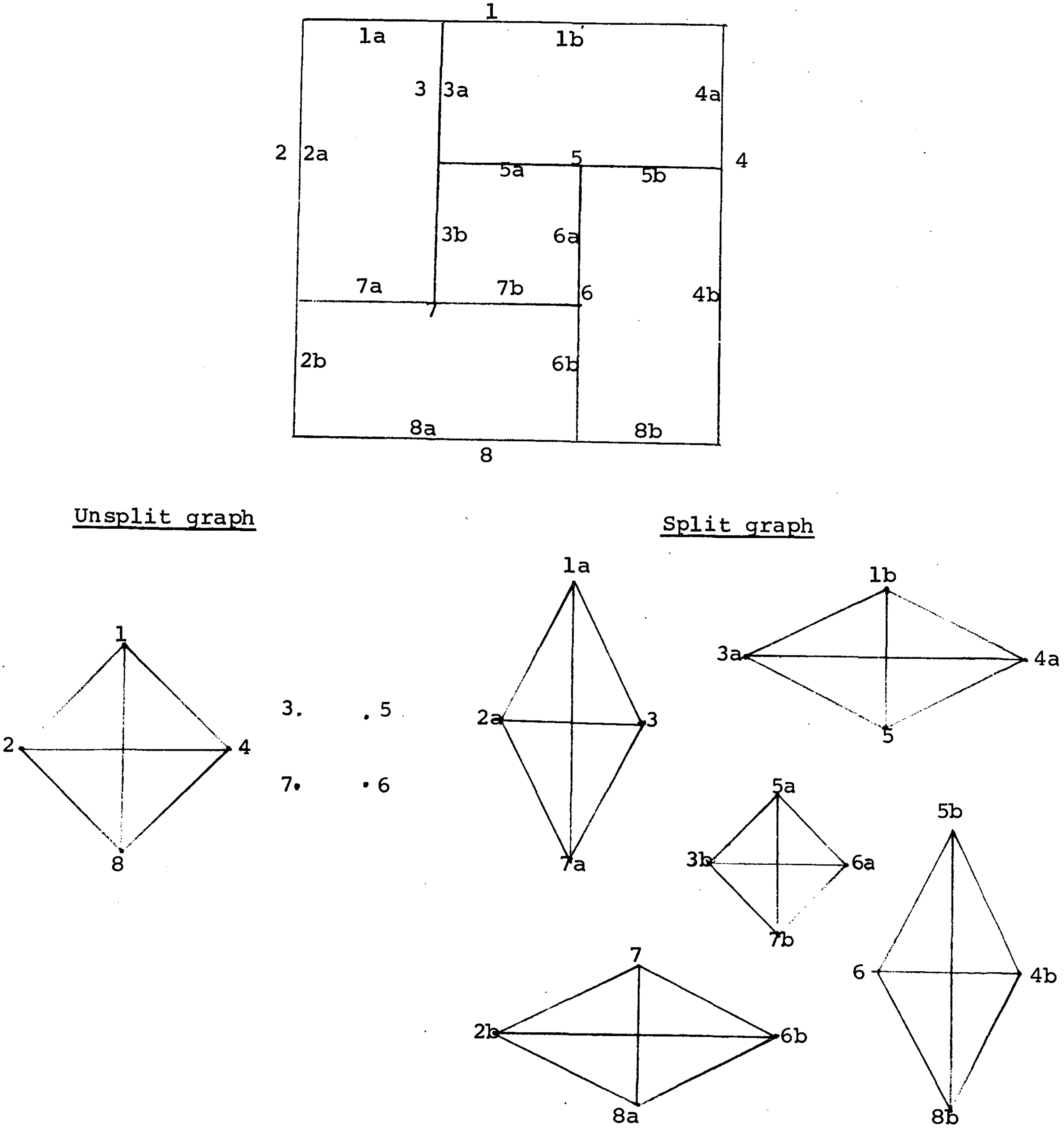


Figure 11. The unsplit graph yields the outer square and the T's inside it; the split graph yields the four rectangles and inner square.

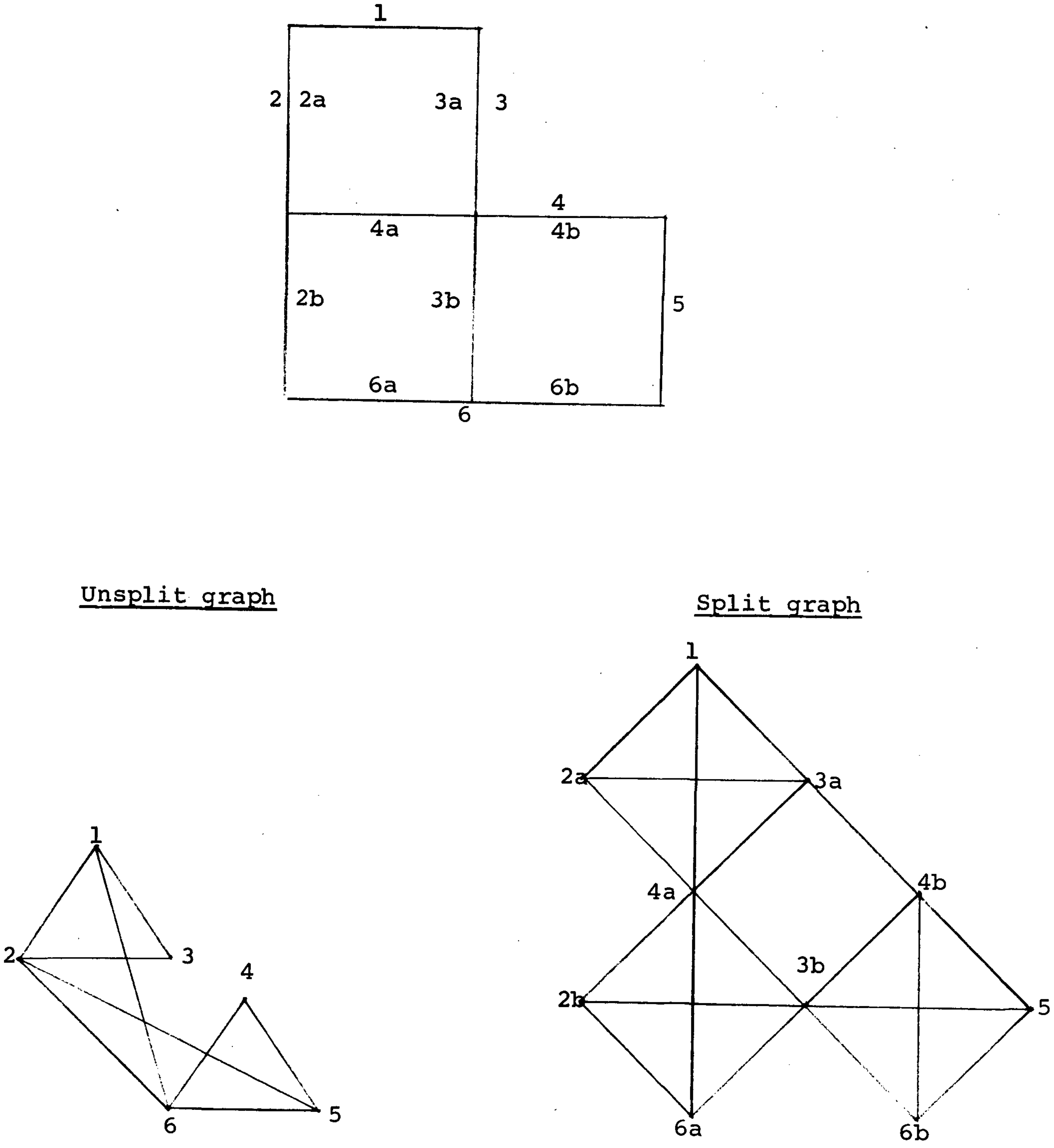


Figure 12. The unsplit graph has many MCSG's (compare Figure 6a); the split graph yields the three squares (and an L). Here the rectangles cannot be obtained except one at a time, by preferential splitting.

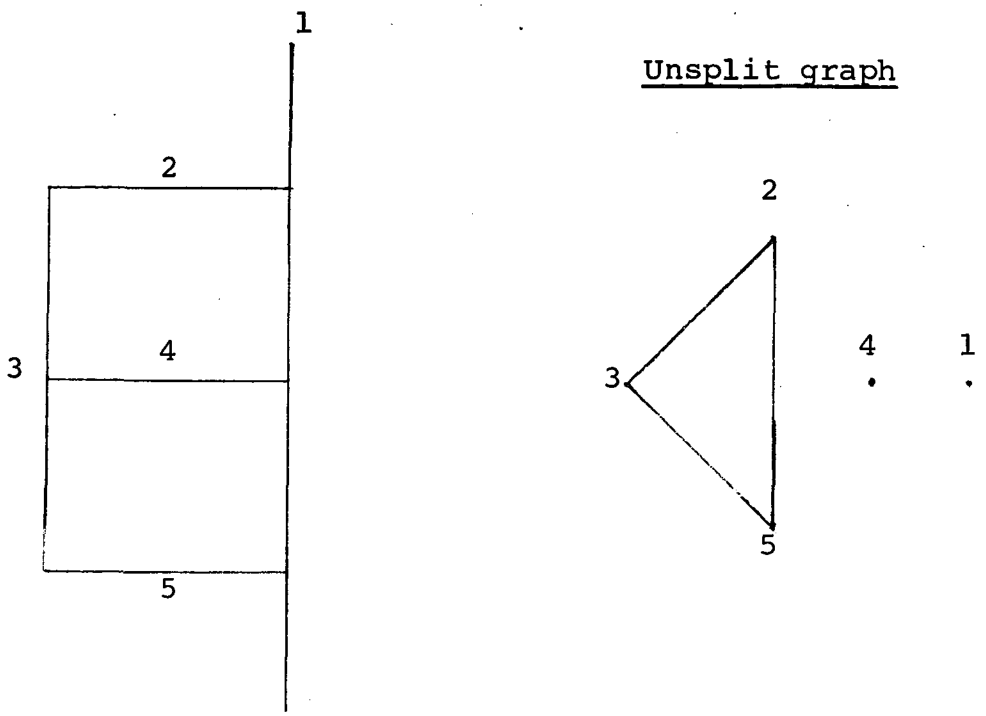


Figure 14. The E is much more poorly hidden than in Figure 1.

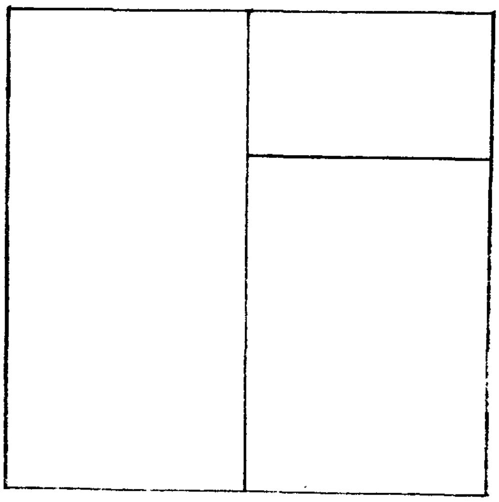


Figure 13. The unsplit graph yields the square and two segments; the split graph yields three rectangles. To get the two large rectangles and the small line segment, it would be necessary to split just two of the four T's.

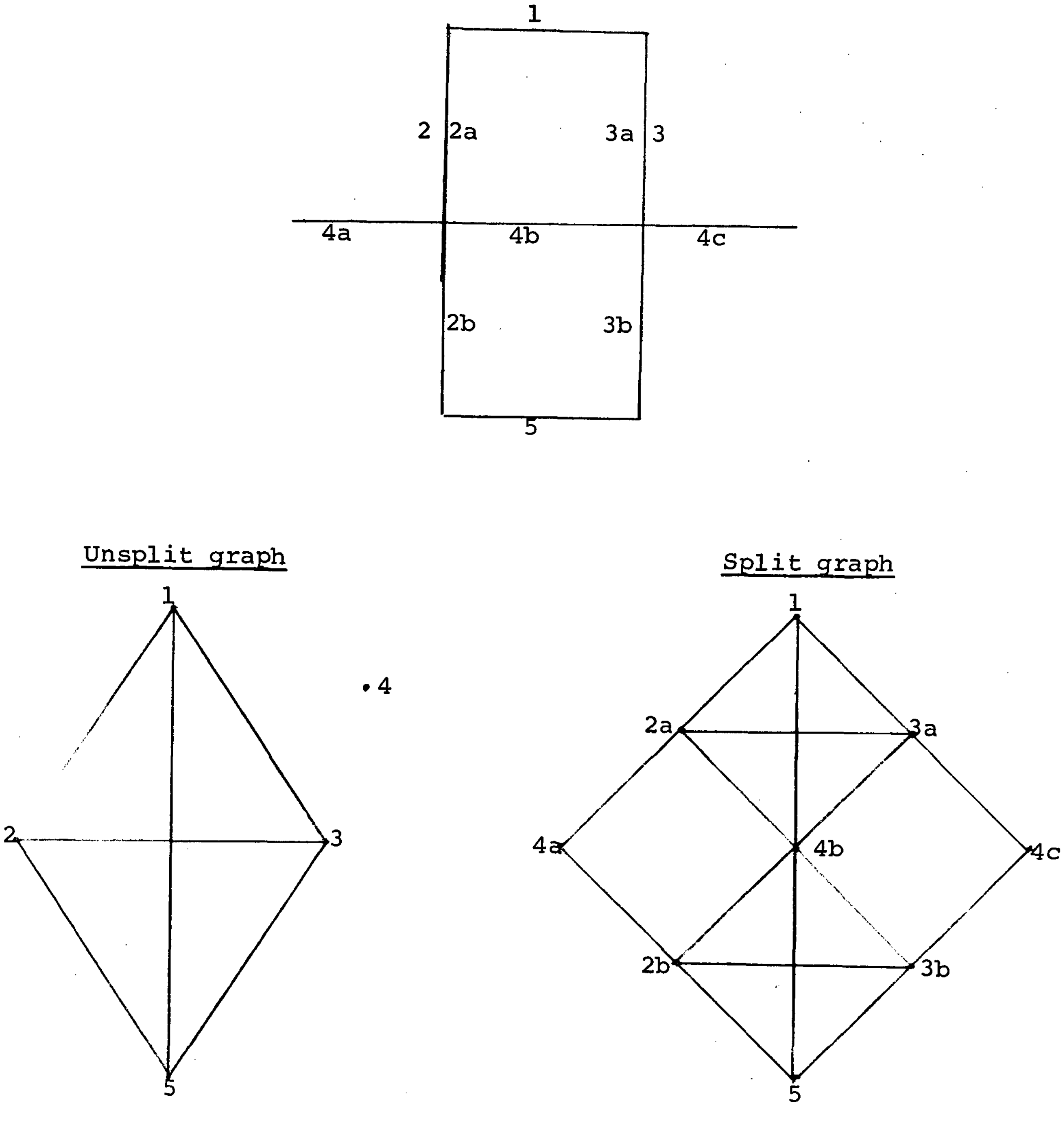
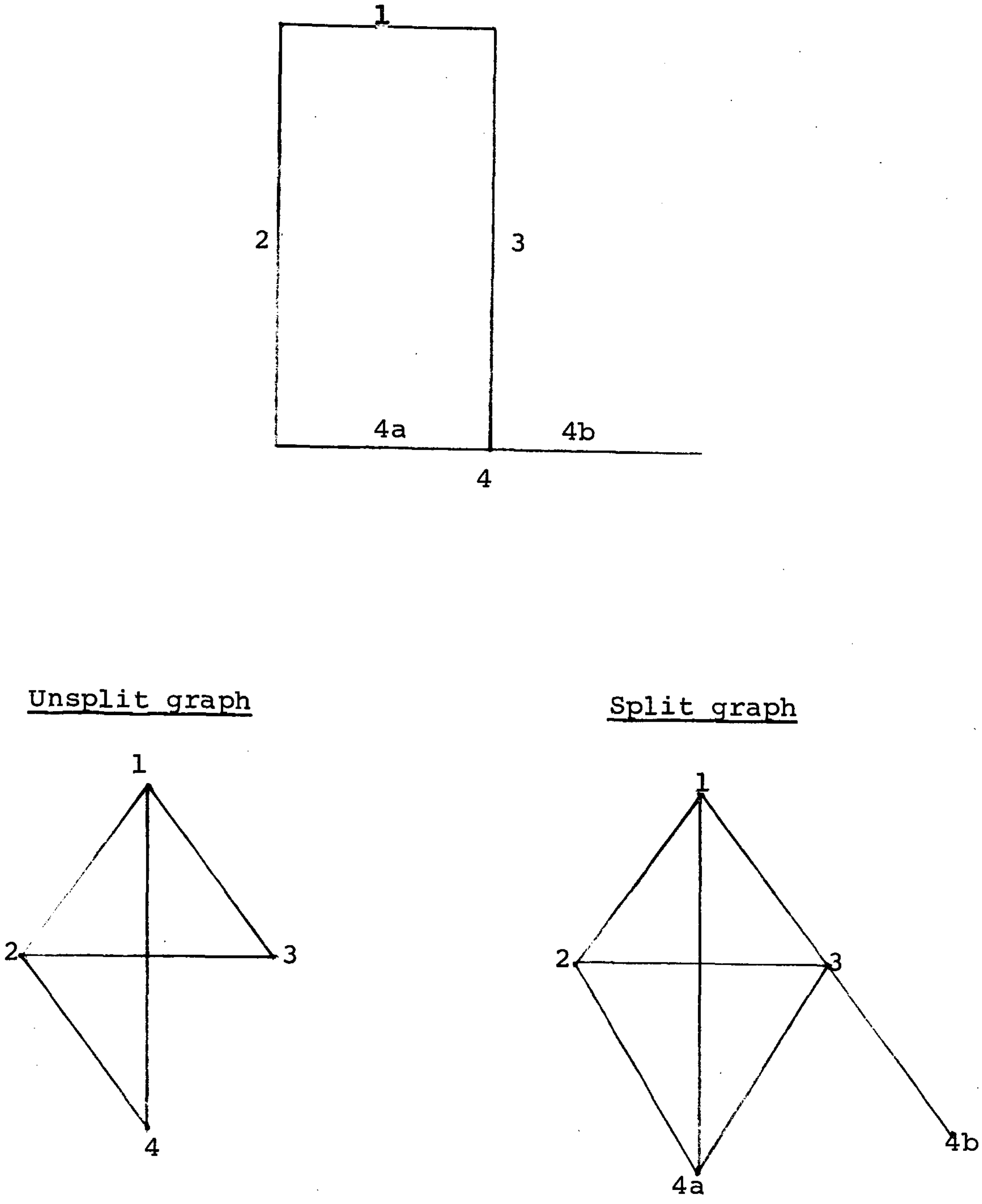


Figure 15. Here the rectangle and line "cover" the drawing with fewer pieces.



Unsplit graph

Split graph

Figure 16. This is seen as a rectangle and line, not as rectangle and L.

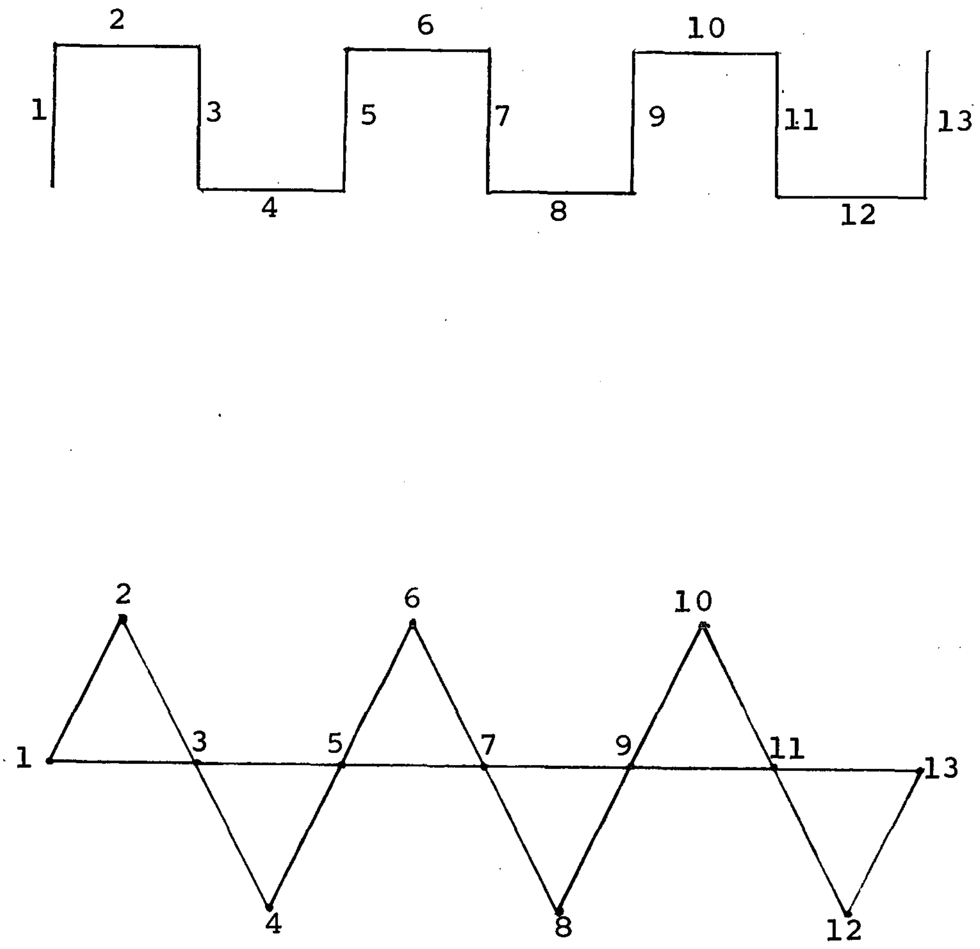


Figure 17. This can be seen as bounding the region above it or the region below it, but not both at the same time.

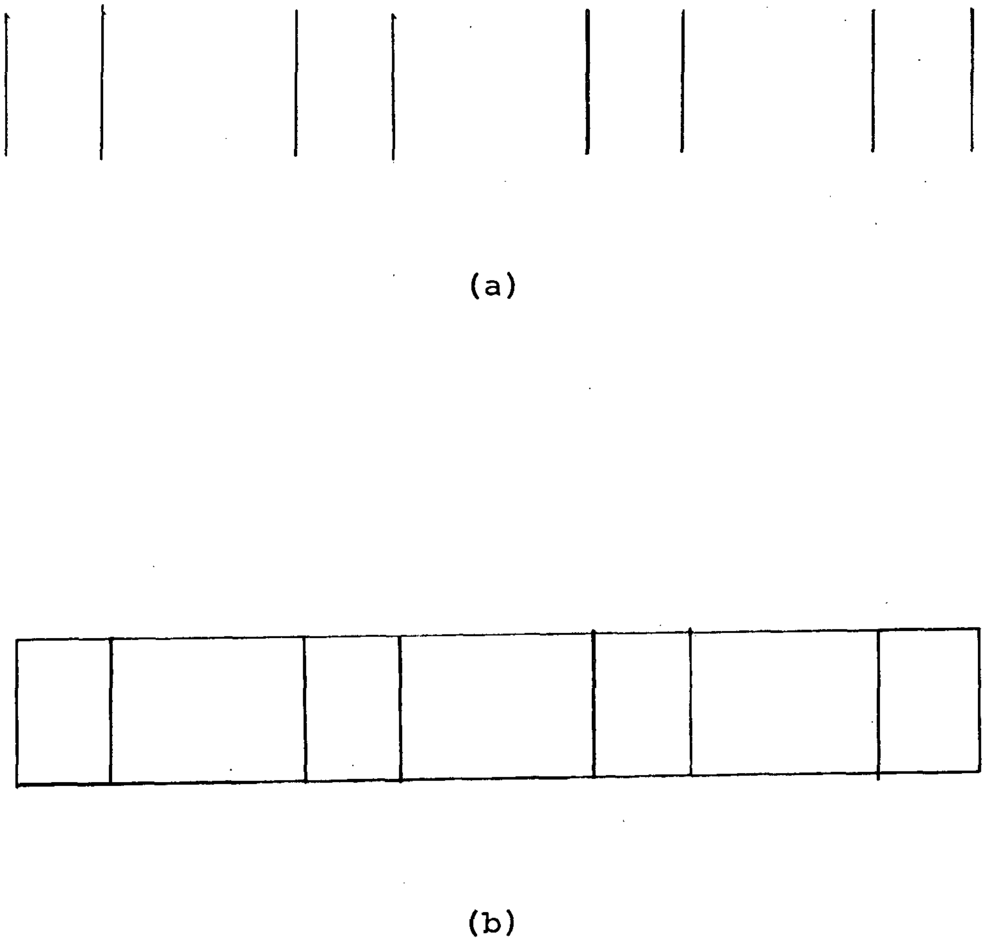


Figure 18. Links between pairs of parallels.

Appendix: Description of the data structure and programs

A1. Data structure

The data structure provides for storing information about each input line segment and its relationships with other segments. Not all of this information is used by the present cluster-finding programs; some of it was incorporated to allow for the possibility that alternatives to the present programs might prove desirable.

For each line segment, the structure provides for eight types of pointers to other segments:

No.	Type	Relationship between the segments
1	T	T-junction
2	L	L-junction
3	C	C-junction
4	P	Parallel aligned at both ends
5	W	Weak parallel (aligned at only one end)
6	V	Very weak parallel (aligned at neither end)
7	CL	Collinear
8	LL	Making L's on same side of another segment

The information about each line segment is stored in a block of N words, where $N = \Sigma_{i=1}^{8} N_i$ (N_i being the number of pointers of type i).

The first word contains the length of the line segment (bits 0-11), and 12 other bits (24-35) that provide the information indicated in the following table; the remaining bits of this word are reserved for possible future use.

Bit No.	Significance if equal to 1
35	Line is nondegenerate (i.e., has length > 0)
34	Line is vertical (if = 0: horizontal)
33	Line has been split
32-25	Line stands in relationship T (L,C,P,W,V,CL,LL) to at least one other line
24	Line has been erased

The second word consists of three 12-bit fields. If the line segment is horizontal, its y-coordinate is stored in bits 24-35, and the x-coordinates of its endpoints in bits 0-11 and 12-23. If it is vertical, the stored information is analogous, but with x and y interchanged. Words 1-2 of the block are referred to as "LINFO" subblock.

In the third word, bits 0-11 (the "FATHER" field) point to a line segment, if any, from which the given one was derived by splitting. Bits 12-23 (the "LREF" field) have value 1 for input segments, 2 for splits, 3 for splits of splits, etc. Bit 35 (the "LNKBIT") has the value 1 if the line segment is part of another line, 0 otherwise.

In the fourth word, bits 0-11 (the "NEXT" field) point to the line segment, if any, that is collinear with the given one and immediately on its right (for horizontal segments; below it, for vertical segments). Bits 12-23 (the "PREV" field) point to the collinear segment (if any) immediately to the left of (or above) the given one. Bits 24-35 (the "SON" field) point to the leftmost (or topmost) segment obtained by splitting the given one, if it does in fact split. Words 3-4 of the block are referred to as the "SEG" subblock.

In the next group of N_1 words, information about T-relationships involving the given line segment is stored. Each of these words consists of four fields. Bit 35 (the "FLAG") indicate whether or not the T-junction in question has been split. Bits 24-29 (the "RELPOS" field) are used to indicate the orientation of the T:

Value of RELPOS	Orientation of T
0	T
1	⊥
2	⊢
3	⊣

Bits 12-23 (the "LINKWT" field) allow for storage of a weight associated with the given T-link (see below). Bits 0-11 (the "LINK" field) contain the number of the line segment (i.e., the position of its block in the array of line segment blocks) that forms the T-junction with the given one.

The succeeding groups of $N_2, N_3, \ldots, N_7, N_8$ words store analogous information about L,C,P,W,V,CL,LL relationships involving the given line segment. In each of these groups, the format of each word is the same as for the T-words. For L,P,CL,LL relationships, the FLAG field is not used, since splitting is not applicable in these cases. For L relationships, the values of RELPOS correspond to the following orientations:

Value	Orientation
0	L
1	┌
2	┘
3	┐

The method of assigning weights to the various types of links is as follows:

1) The weights assigned to C-, T-, L-, LL-, and CL-links are constant. Specifically, the link weight associated with each C-, T-, L- (or LL-), and CL-link is equal to 0,1,2, and 4, respectively.

2) The weights assigned to P-links, W-links, and V-links vary as the ratio of the length of the shorter of the two parallel lines and the distance between the lines.

For example, let the lines have lengths L_1 and L_2, where $L_1 \le L_2$, and let D be the distance between them. Let

WT_o = 4 for a P-link

2 for a W-link

1 for a V-link

Then the link weight WT is assigned as follows:

If $d = L_1$ then $WT = WT_o$

If $d < L_1$ then $WT = 2WT_0$

If $d > L_1$ then $WT = \frac{1}{2}WT_o$

A2. Programs

Input: Each line segment is input as a quadruple (NVH,NZ, N1,N2), where

NVH denotes the orientation of the line

NVH = 2: horizontal

NVH = 3: vertical

NZ denotes the x-coordinate of a vertical line,
the y-coordinate of a horizontal line

N_1,N_2 denote the y-coordinates of the endpoints of a vertical line
the x-coordinates of the endpoints of a horizontal line

Subroutine READIN reads in and counts the input segments. The quadruples are stored in array ARRAY, one segment per block, in the first two words of the block; the format for each block is as described in Section A1. The segments are counted, and their number is stored in variable NLINE.

Link generation: The conditions for generating links between segments are as follows:

Suppose $L_i = (NVH_i, NZ_i, N1_i, N2_i)$ and
$L_j = (NVH_j, NZ_j, N1_j, N2_j)$,
where $i \neq j$ and $i \leq i,j \leq NBLK$

(1) Generation of a T-link requires

(a) $NVH_i \neq NVH_j$ and

(b) $NZ_i = N1_j$ or $NZ_i = N2_j$

(c) $N1_i < NZ_j < N2_i$

or (b') $NZ_j = N1_i$ or $NZ_j = N2_i$;

(c') $N1_j < NZ_i < N2_j$

(2) Generation of an L-link requires

(a) $NVH_i \neq NVH_j$ and

(b) $N1_i = NZ_j$

or $N2_i = NZ_j$

or $N1_j = NZ_i$

or $N2_j = NZ_i$

(3) Generation of C-link requires

(a) $NVH_i \neq NVH_j$ and

(b) $N1_j < NZ_i < N2_j$ and

(c) $N1_i < NZ_j < N2_i$

(4) Generation of a P-link requires

(a) $NVH_i = NVH_j$ and

(b) $N1_i = N1_j$ and

(c) $|N2_i - N1_i| = |N2_j - N1_j|$

(5) Generation of a W-link requires

(a) $NVH_i = NVH_j$ and

(b) $N1_i = N1_j$ (or $N2_i = N2_j$) and

(c) $|N2_i - N1_i| = t|N2_j - N1_j|$, where $t \neq 1$

(6) Generation of a V-link requires

(a) $NVH_i = NVH_j$ and

(b) $N1_i < N1_j < N2_j < N2_i$

or $N1_j < N1_i < N2_i < N2_j$

(7) Generation of a CL-link requires

(a) $NVH_i = NVH_j$ and

(b) $NZ_i = NZ_j$ and

(c) $N2_i < N1_j$ or $N2_j < N1_i$.

(8) Generation of an LL-link requires that there exist at least one $L_k = (NVH_k, NZ_k, N1_k, N2_k)$, where $k \neq i,j$ and $1 \leq k \leq NBLK$, such that

a) L_i and L_j are both related to L_k by L-links

b) $|RELPOS_{ik} - RELPOS_{jk}| = 2$ if L_k is horizontal; $= 1$* if L_k is vertical (where $RELPOS_{xy}$ denotes the orientation of the L formed by L_k and L_y)

Subroutines LNKGN and GLINKL determine the links that exist between the pairs of segments in ARRAY, and store the information about these links in ARRAY. LINKGN is used for the first seven types of links (one call per type), and GLINKL for the LL-links.

Line splitting: For each input segment S the segments related to it by the C relationship are found by a call to subroutine LINKGN. The coordinate of the split point corresponding to each of these is then determined. These coordinates are put into order of increasing x (or y). (If there are no split points, bit 33 in the first word of S's block is set to 0; otherwise, to 1.) The subsegments between successive split points are regarded as new line segments, and are added to ARRAY using subroutine READIN. Further information about these segments (see LREF, NEXT, PREV, etc. in section A1) is also stored in ARRAY. The total number of segments is stored in variable NOLIN.

For each line segment S which splits, bit 24 in the first word of S's block is set to 1 to indicate that this line segment is deleted (i.e., the line segment S will not be used for subsequent line splitting). The T-links between pairs of segments are then determined and stored in ARRAY by one call to subroutine LINKGN. For each line segment S,

*and the condition 1) $RELPOS_{ik} = 1$ and $RELPOS_{jk} = 2$ or 2) $RELPOS_{ij} = 2$ and $RELPOS_{jk} = 1$ is not true.

two sets of split points are determined, corresponding to legs of T's that lie on the two sides of S. The resulting subsegments are then regarded as new line segments. If both sets of split points are nonempty, segment S is marked as deleted. Again, the total number of line segments is stored in variable NOLIN.

Generation of matrices and cluster detection: In order to apply the cluster detection procedure described in the body of the report, two types of links between pairs of line segments, L and LL, are determined and are stored in ARRAY by calls to subroutines LINKGN and GLINKL, respectively. This is done both for the set of input line segments and for the set of segments that results after line splitting.

Two matrices $(a_{ij})_{mxn}$ and $(b_{ij})_{nxn}$ are then generated; m and n are the numbers stored in variables NLINE and NOLIN, respectively. Element (i,j) of each matrix is the link weight between line segment S_i and line segment S_j. These matrices can be regarded as defining graphs (two nodes are joined by an edge if the link weight is 2). All maximal complete subgraphs of these graphs may then be obtained*. The line segments corresponding to the nodes of these subgraphs can then be printed out.

*The subroutine that accomplishes this was made available by Professor Jack Minker, whose help is hereby gratefully acknowledged.

A3. Sample output

Input segments

```
1 400000000016 000000160034
2 400000000034 001600000034
3 400000000034 003400000034
4 600000000016 000000160034
5 600000000034 001600000034
6 600000000034 003400000034
```

Display of input drawing

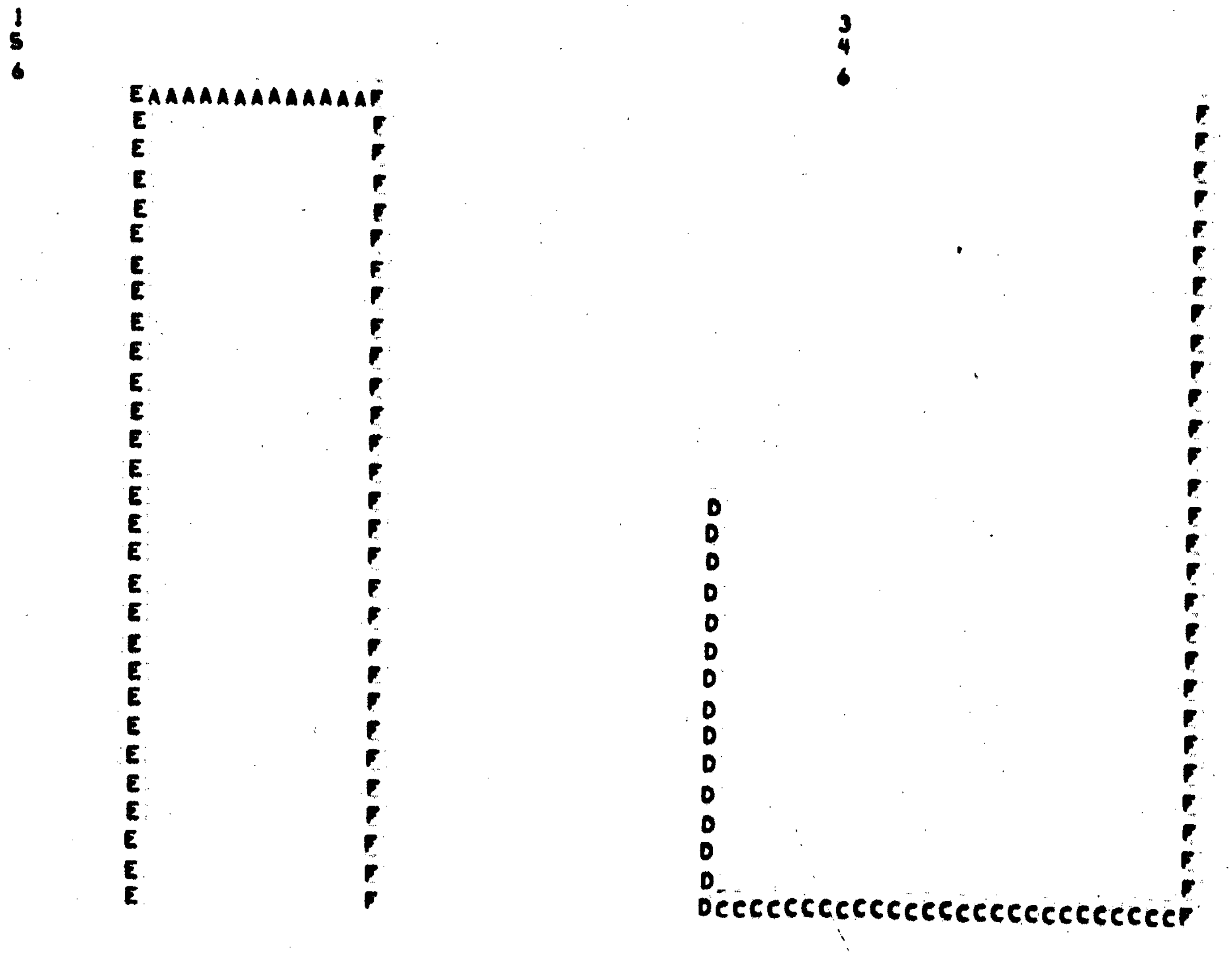
```
              EAAAAAAAAAAAAAF
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
              E             F
DBBBBBBBBBBBBBEBBBBBBBBBBBBBF
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
D             E             F
DCCCCCCCCCCCCCECCCCCCCCCCCCCF
```

Incidence matrix for unsplit graph

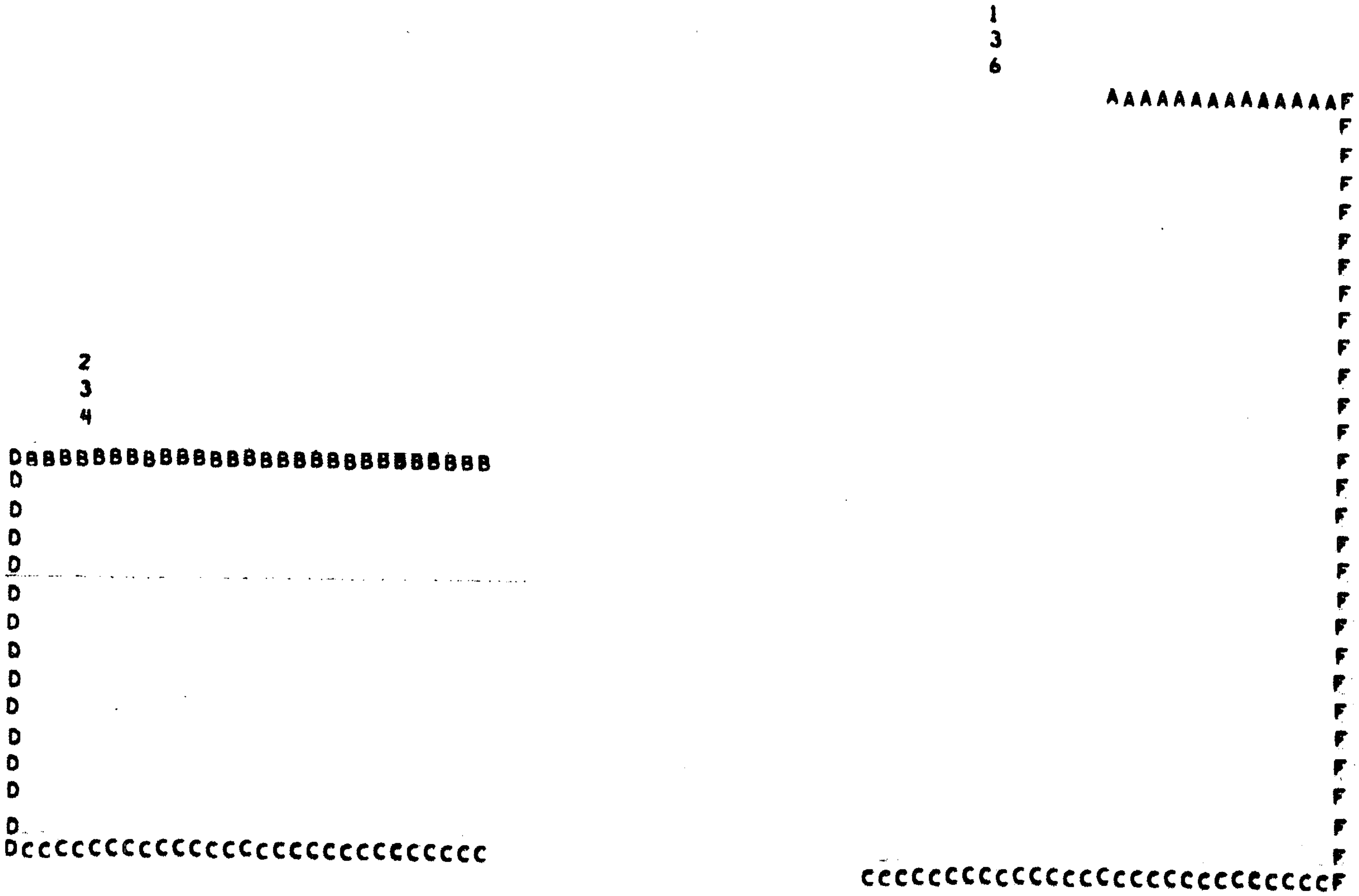
$$\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 0 & 2 & 0 & 2 & 2 \\
2 & 0 & 0 & 2 & 2 & 0 & 0 \\
3 & 2 & 2 & 0 & 2 & 0 & 2 \\
4 & 0 & 2 & 2 & 0 & 0 & 2 \\
5 & 2 & 0 & 0 & 0 & 0 & 2 \\
6 & 2 & 0 & 2 & 2 & 2 & 0
\end{array}$$

Incidence matrix for split graph (original crossbars kept for T's but not for C's)

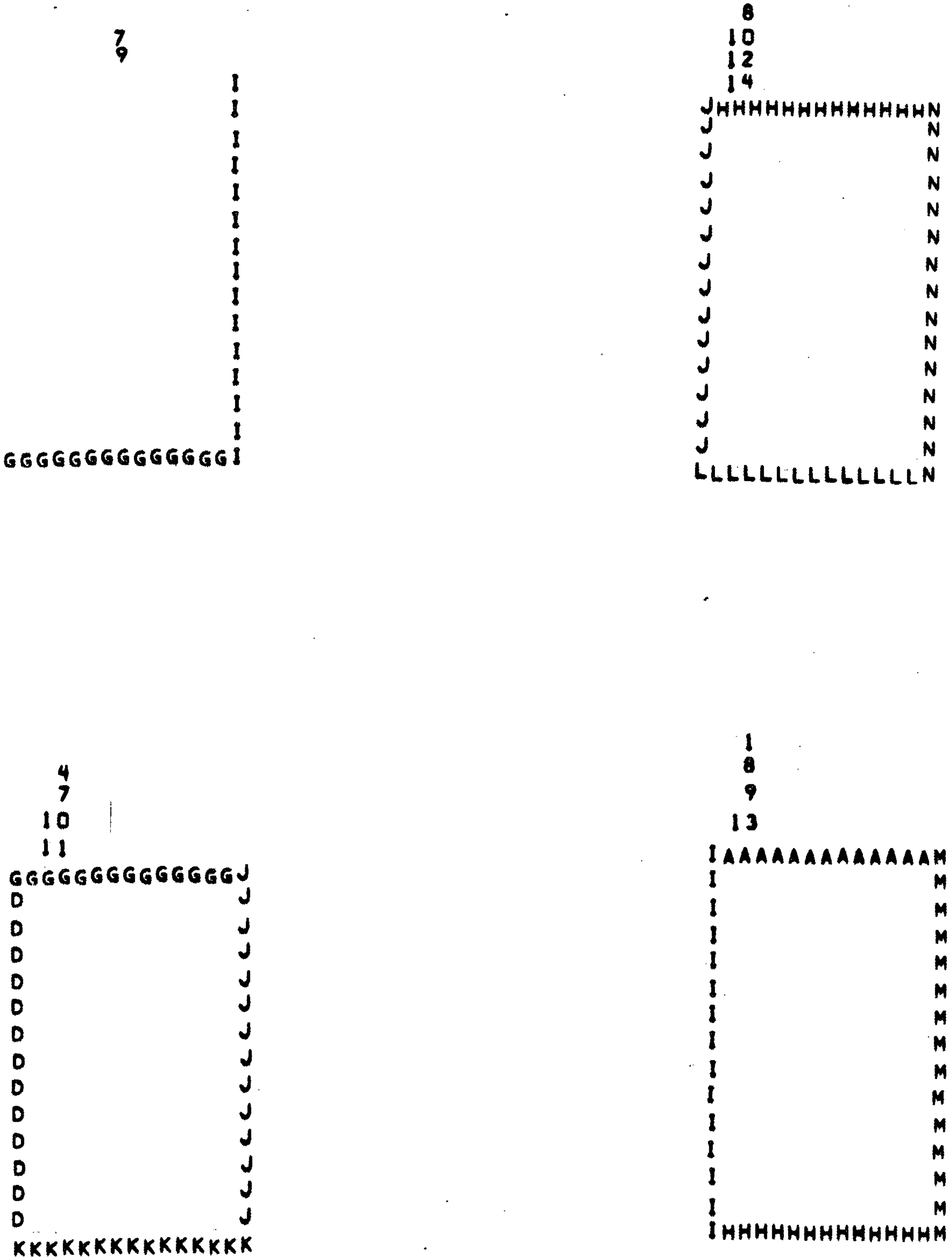
$$\begin{array}{c|cccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
\hline
1 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 2 & 2 & 0 & 0 & 2 & 2 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 2 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 2 \\
4 & 0 & 0 & 2 & 0 & 0 & 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 & 2 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 2 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 2 & 0 & 0 \\
7 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 \\
8 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 2 & 2 & 2 \\
9 & 2 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 2 & 0 \\
10 & 0 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 2 & 2 & 0 & 2 \\
11 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
12 & 2 & 0 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 2 \\
13 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\
14 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 0
\end{array}$$



Clusters in unsplit graph



Clusters in unsplit graph (continued)



Clusters in split graph