ESD-TR-67-61

Technical Note 1967-1

R. M. Baecker

Planar Representations of Complex Graphs

6 February 1967

Prepared for the Advanced Research Projects Agency
under Electronic Systems Division Contract AF 19(628)-5167 by

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lexington, Massachusetts

ESVH

AD648133

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology; this work was supported by the U.S. Advanced Research Projects Agency of the Department of Defense under Air Force Contract AF 19(628)-5167 (ARPA Order 691).

ESD-TR-67-61

ERRATA SHEET

for Technical Note No. 1967-1

The author has detected the following errors in Technical Note No. 1967-1 (R. M. Baecker, "Planar Representations of Complex Graphs," 6 February 1967). Kindly insert this page into your copy of that report.

Page 1	In Line 1, change the reference to the Bibliography from 1 to 2,5
Page 6	In Line 19, change the reference to the Bibliography from 2 to 1,2,7
Page 21	In Line 21, change the reference to the Bibliography from 3 to 5
Page 28	In Line 17, change the reference to the Bibliography from 4 to 3
Page 29	In Line 2, delete the reference to the Bibliography given as 4
Page 29	In Line 3, change the reference to the Bibliography from 5 to 4

17 March 1967

Publications
M.I.T. Lincoln Laboratory
P.O. Box 73
Lexington, Massachusetts 02173

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

PLANAR REPRESENTATIONS OF COMPLEX GRAPHS

R. M. BAECKER

Consultant
Group 23

TECHNICAL NOTE 1967-1

6 FEBRUARY 1967

LEXINGTON MASSACHUSETTS

PLANAR REPRESENTATIONS OF COMPLEX GRAPHS

ABSTRACT

This paper's major topic is the problem of constructing good two-dimensional representations, using straight lines as edges, of complex, generally non-planar graphs. Two precise measures that correlate well with the subjective "clarity" of planar representations of randomly generated graphs are proposed. One is the number of edge intersections that do not correspond to graph vertices. The second measure, called infidelity, quantifies the degree to which the distance between pairs of vertices (in a particular metric in the plane) corresponds to their graph-theoretical distance from one another.

Some experiments in graph representation, using the TX-2 computer on-line display, are described. Heavy emphasis is given the definition of the infidelity measure, the nature of its correlation with graph "clarity", and some theorems which add to its characterization. Included is a discussion of heuristics for the simplification of a graph representation.

In conclusion, there is presented a sketch of a model of a graph theory computer package that should be useful to a researcher or student of graph theory.

Accepted for the Air Force
Franklin C. Hudson
Chief, Lincoln Laboratory Office

TABLE OF CONTENTS

INTRODUCTION

This paper is a report on research conducted by the author during the summer of 1966 in the Digital Computers Group of MIT's Lincoln Laboratory. In its broadest sense, the goal of the research is to demonstrate useful and interesting interconnections among three distinct, yet seemingly related, disciplines:

1. The purely mathematical study known as graph theory;
2. the as-yet unformalized body of programming know-how involved in the creation and manipulation of complex "data structures"; and
3. the field of computer graphics, which seeks to facilitate graphical man-machine communication.

More specifically, a major goal is to delimit issues involved in the design of a class of potential "graph theory packages", a term we use to denote any programming system allowing effective man-machine partnership in the teaching of, or research in, graph theory. In Chapter II we sketch a model for a graph theory package based upon the work of this summer.

To gain some experience with such system design, we have assembled a small package tailored for exploration of the problem of finding "good" representations, in a plane, of complex (in general, non-planar) graphs. Chapter I discusses both the problem and the package in detail. We shall describe the nature of the man-machine communication achieved with this rather simple system, coded in a five-week assembly language programming effort.

Of theoretical interest is a lengthy discussion of a class of precise measures that correlate well with intuitive notions of the "simplicity" and "clarity" of a representation of a complex graph in a plane. We include both a critical discussion of various features of the measure as well as several theorems characterizing them.

We conclude Chapter I with a discussion of proposed extensions of this work, which include questions relevant to both mathematical graph theory and computer graphics.

I. REPRESENTATION OF COMPLEX GRAPHS IN A PLANE[1]

A. PROBLEM STATEMENT

We shall investigate representations of a graph in a plane, assignments of (x, y) coordinant pairs to each vertex, and straight line segments between vertices corresponding to each edge. These shall be termed <u>planar representations</u>. The graphs may be directed or undirected, <u>planar</u> or <u>nonplanar</u>. The restriction to straight line representations of the edges results in great analytic simplification and appears to be appropriate for some applications.

In such a representation of either a planar or a non-planar graph, there may appear apparent intersections which do not correspond to vertices and are therefore artifacts of the representation. The graph of Figure 1 possesses such an intersection; an alternate representation in Figure 2 demonstrates that it is, in fact, a planar graph.

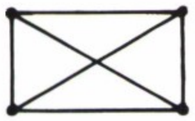


FIGURE 1

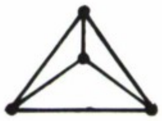


FIGURE 2

The mathematical question of the minimum number of such intersections for all representations of a graph on a particular topological surface (with or without the straight line restriction) is the natural extension of the planar-nonplanar distinction in graph theory (if nonplanar, "how much so"?) It will be discussed further in Section C.

The problem may be simply but not precisely stated: Given a graph, find a planar representation which "best illustrates the structure" of the graph. How can one organize a complicated interconnection of vertices and edges so that an observer can "best see and understand the relationships thereby represented".

The problem will occur whenever one wishes to see a pictorial representation of a complex graph. If this be "computer generated" onto a display console, the issue arises in doing computer graphics. Potential applications to computer science, such as the display of complex data structures, and flowcharting or flowmapping for debugging purposes, suggest themselves immediately.

We shall therefore present techniques that reorganize a planar representation of a graph in order to "simplify" its appearance. We shall discuss precise measures that appear to correlate well with subjective notions of graph "clarity". As computer graphics was a natural partner in this investigation, we must first describe, by example, the simple system developed to aid in the effort.

B. A SMALL GRAPH THEORY PACKAGE TAILORED FOR INVESTIGATION OF THIS PROBLEM — AN EXAMPLE ILLUSTRATING ITS USE

The user specifies a TOTAL number of potential graph vertices and a desired DENSITY of edges. The program generates a random graph with DENSITY (TOTAL) (TOTAL - 1)/2 edges connecting some (but not necessarily all) of the TOTAL vertices. As a typical example, consider a case, pictured in Figure 3, in which TOTAL = 16 (decimal), and the DENSITY of connections is 20 percent of the theoretical maximum. Notice that the vertices are initially assigned randomly to positions on a grid of points. These are spaced equidistantly with a slight random offset to reduce overlap of lines.

By pointing with a light pen at pairs of points on the grid (some of which are graph vertices), he may interchange the point positions and cause appropriate relocation of the adjacent edges. Special symbols appear on the display to mark the two chosen points. The hand in Figure 4, for example, points to the vertex in the lower right corner which is to be relocated. Figure 5 shows the new representation achieved after that vertex has been moved to a position on the left boundary.

The user may initiate a series of such requests, without interruptions for recalculation of the display, by preceding and following the series with hits on the appropriate light pen targets, which appear at the bottom of the display. Three such interchanges relocate a section on the right side, resulting in the representation of Figure 6. Figure 7 is achieved after two more interchanges. Figure 8 shows another representation of the same graph achieved after fifteen minutes of console experimentation at another sitting.

With the aid of the light pen tracking cross, the user may generate a new grid point if the given set proves inadequate.

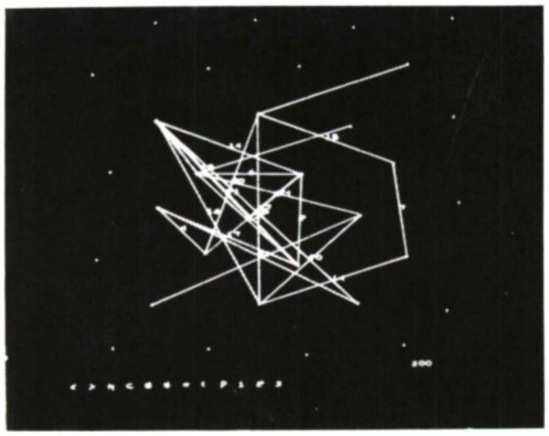


FIGURE 3
Infidelity = 300 (octal)

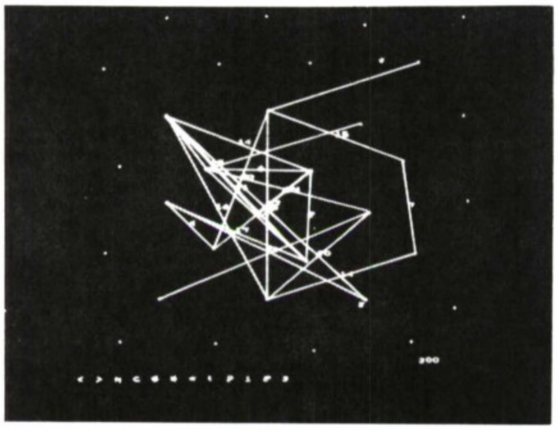
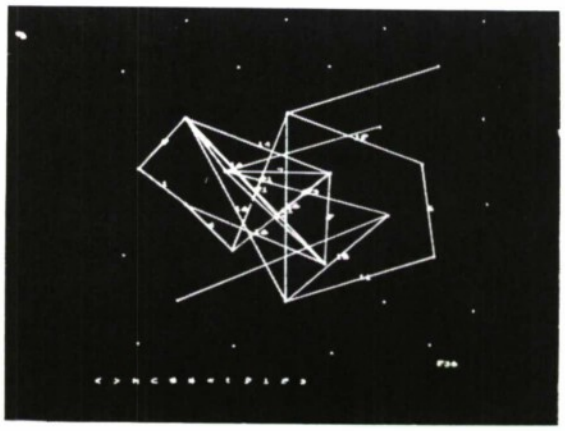


FIGURE 4
Infidelity = 300

FIGURE 5
Infidelity = 236



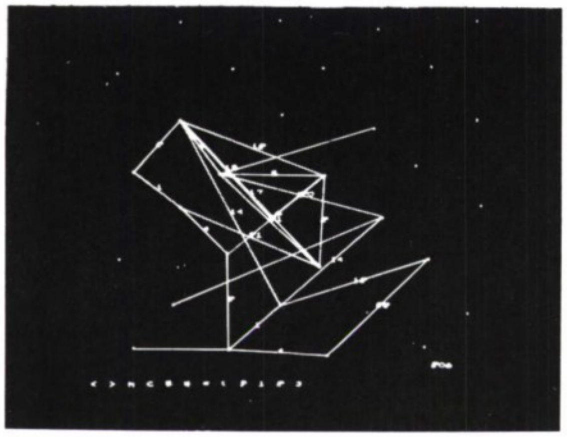


FIGURE 6

Infidelity = 206 (octal)

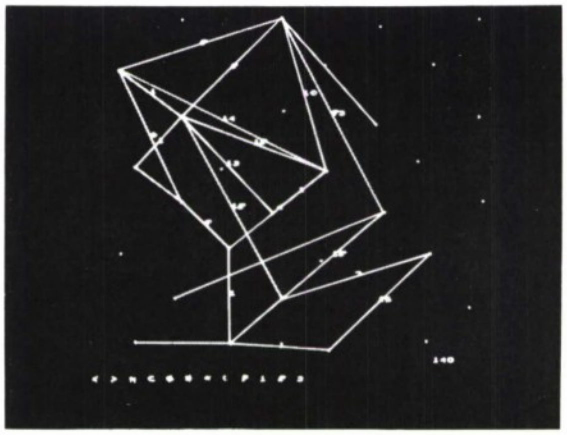


FIGURE 7

Infidelity = 140

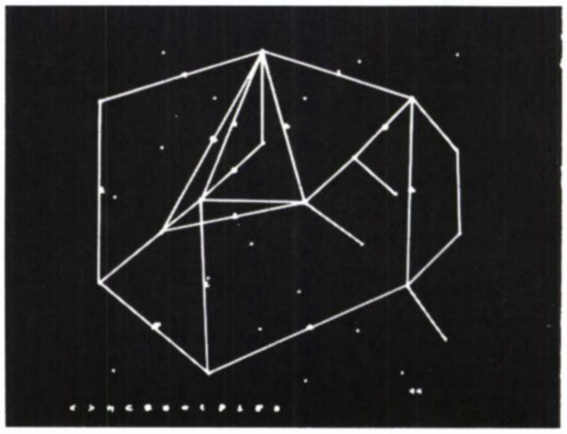


FIGURE 8

Infidelity = 44

As we shall later see, these are the simplest examples of a large class of user-controlled reorganization features that could be included in such a system.

A second set of light pen targets may be used to initiate routines that calculate interesting parameters of the given graph. In the present system, one target displays the degree of each vertex; a second calculates and displays all centers of a particular connected component of the graph, after the user chooses a component by pointing at any of its vertices.

C. MEASURES ON A GRAPH REPRESENTATION

The intersection criterion and the notions of graph infidelity, to be discussed below, were formulated to meet the need for precise and simple measures that correlate well with the subjective clarity of a planar representation. Any automatic or partially automatic simplification scheme would require measures to evaluate its performance.

1. Intersections

The first measure that correlates well with intuitive notions of graph simplicity is an obvious one, the number of edge intersections not corresponding to graph vertices. Attempts to find or characterize the theoretical minimum of this quantity for a given graph have had little success.[2] The only known results apply to very special cases, the complete graph and the complete bipartite graph. In the latter case, the minimum representation is known and may be proved to be minimum.

Such special cases are of little help in attacking the problem for graphs in general. It seems unlikely that techniques for constructing the minimum representations in more general situation will be found. A more promising approach would be to search for simplification heuristics and search algorithms that may be shown, either theoretically or experimentally, to reduce significantly the number of such intersections for a large class of graphs.

2. Infidelity

We have developed a second kind of measure on the graph representation, to be called infidelity, one that quantifies the degree to which the distances (in a particular metric in the plane) between vertices correspond to the graph-theoretical distances between vertices. We recall that the graph-distance between two vertices of a connected graph component may be defined as the minimum number of edges in all paths from one vertex to the other. One infidelity measure may be defined as follows:

(a) Consider an edge e_j and one of its endpoints v_{j1} in some planar representation r. Construct the locus of all points at a distance (in the chosen metric) equal to the length of e_j from v_{j1}. Let the measure u^r_{j1} be the number of vertices within the locus that are at a finite-graph distance strictly greater than one unit from v_{j1}. This quantity u^r_{j1} shall be known as <u>the infidelity of the edge e_j with respect to its endpoint v_{j1}</u> (in the representation r).

(b) _The infidelity of the representation r of the (connected) graph_, u^r, is defined as

$$\sum_{j \in J} (u^r_{j1} + u^r_{j2}),$$

where J is the set of all edges of the graph.

(c) _The infidelity of the (connected) graph_ is the minimum value of u^r for all representations r. It is, therefore, a measure of the degree to which the "best" possible planar representation can "correspond" to the graph.

The example in Figure 9 (see next page) should clarify the definitions. Notice how each one endpoint pair is considered separately. Another representation of the same graph in Figure 10 (see next page) shows that the infidelity of this graph is actually zero.

Let another name for v_d be v_{12} and for v_f be v_{11}. The infidelity of the pair (e_1, v_{11}), to be called u^r_{11}, is obtained by constructing the locus of all points at a distance from v_{11} equal to that of v_{12} (the length of e_1). In the usual Euclidean metric, this is a circle with center v_{11} passing through v_{12}. Four vertices, v_a, v_b, v_c, and v_g lie within this locus. Because v_c and v_g are one graph-distance unit from (directly connected to) v_f, they contribute nothing to the infidelity u^r_{11}. The constraint violations of v_a and v_b,

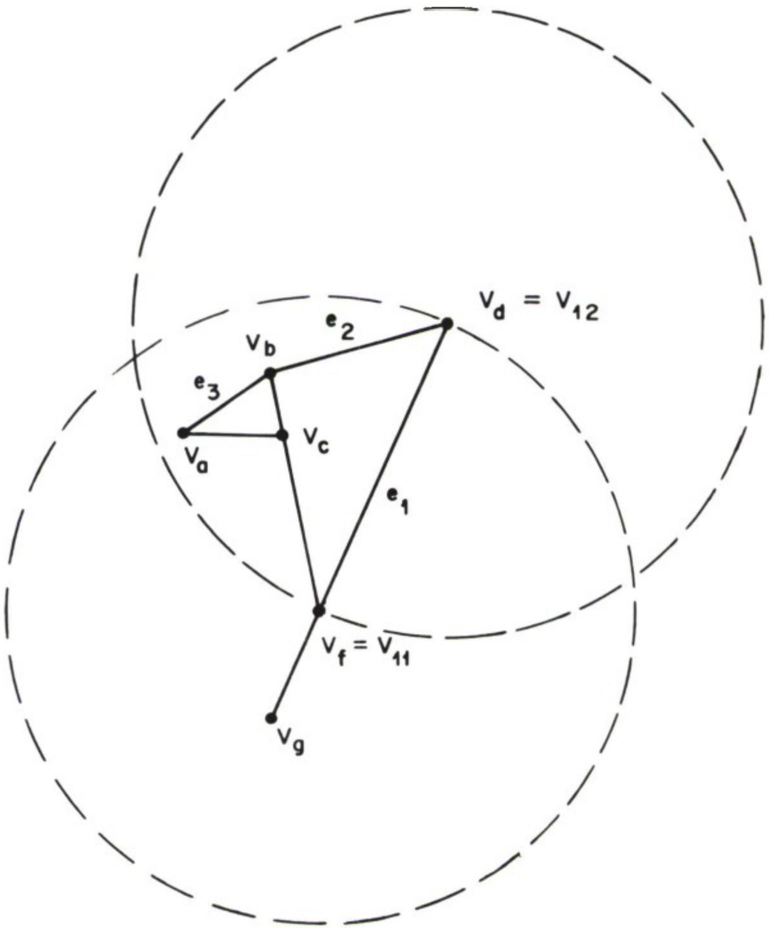


FIGURE 9

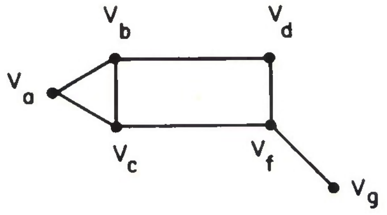


FIGURE 10

however, cause the infidelity u^r_{11} to be two.

Because v_a and v_c are found inside the corresponding locus surrounding v_{12}, the infidelity u^r_{12} is two. It should be clear by inspection that u^r_{21}, u^r_{22}, u^r_{31}, u^r_{32} are all zero.

The reader is now encouraged to review the original simplification series of Figure 3 through 8. The infidelity of each representation (in octal) is displayed in the lower right corner. On each edge is displayed the sum of the infidelities of that edge with respect to its two endpoints. It should be noted that in the existent system the metric used is the rectangular one, in which the distance between the two points (x_1, y_1) and (x_2, y_2) is given by the formula $|x_2-x_1| + |y_2-y_1|$. Furthermore, vertices of degree one, which posses only one adjacent edge, have been totally excluded from every aspect of the computation. (See Section C.5. for the justification of this.)

3. A Critical Discussion of Possible Measures of Graph Infidelity

We may view planar fidelity in a "distance" sense as follows: Given a set of vertices, and a set of distance constraints between certain pairs of vertices (the edges), assign (x, y) locations to the vertices that "best" satisfy the constraints. Although we have assumed that each distance constraint is unity (graph-distance), the generalization to weighted edges is obvious.

The simplest calculation scheme would compare the distance (in a metric in the plane) between constrained vertices to the graph-distance constraints. Yet, such a measure yields the same quantity for the two planar representations shown in Figure 11 and 12.

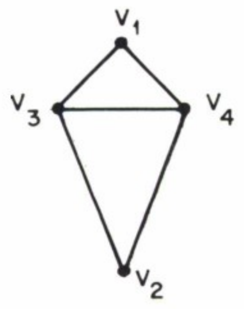


FIGURE 11

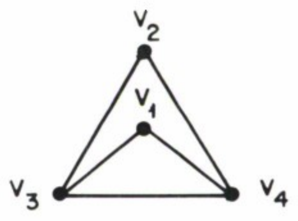


FIGURE 12

The representation in Figure 11 is considered to be the better one, since if it were embedded in a larger graph it would more often yield the intuitively "clearer" representation of the total graph structure.

Experimentation at the TX-2 console has verified this assumption. We have calculated the average length of an edge in the plane, found the difference between each edge length and the average, taken the absolute value of the result, and summed this quantity over all edges of the graph. After dividing the sum by the average length for normalization, and scaling appropriately, we have displayed this quantity after each reorganization. Modifications that yield a much clearer graph are rarely accompanied by a significant reduction in the measure.

The trouble with this calculation is that it ignores an indirect constraint on vertices v_1 and v_2, that they are two units graph-distance apart. One might, therefore, propose to modify the measure by considering the constraints (graph-distance) between all pairs of points. The above calculation has included only those pairs separated by one graph-distance unit, those pairs that define an edge.

We have repeated the above calculation for all pairs of points, considering in each case a normalized distance equal to the distance in the planar metric divided by the graph-distance. As before, we found the average of these quantities and calculated a normalized measure of the dispersion around the average value.

This measure correlates with intuitive simplicity fairly well. The difficulties with it are subtler than in the previous case, in which the measure was clearly insensitive to certain "significant" changes in a planar representation. In one sense, the new measure is oversensitive. Consider, for example, in the preceding Figure 11, movements of vertex v_1 above the horizontal line $v_3 - v_4$. Although the (essentially continuous) measure proposed will vary with movement of v_1, the representation of the graph has hardly changed. The "relative positions" of the vertices, for example, are the same.

Finally, a class of infidelity measures, essentially discontinuous with movement of a vertex, is proposed. As defined in Section 2, the infidelity of an edge e_j with respect to one of its endpoints v_{j1} does not vary directly with its length in a planar metric, but is in fact zero if that edge is drawn "short enough" (the other endpoint v_{j2} is "close enough"). v_{j2} is considered to be close enough if it is nearer to v_{j1} than vertices at a greater graph-distance (two or more) from v_{j1}.

It has been observed experimentally that this measure is somewhat better than the previous one in its correlation with intuitive criteria of representation clarity.

4. An Interpretation of Representation Infidelity in Terms of Constraint Violations

With each edge-endpoint pair of a graph, we may associate a set of fidelity constraints on certain vertex positions. If there is associated with each vertex at finite graph-distance greater than or equal to two units from the endpoint a constraint that the vertex remain outside of the locus defined by the edge and the endpoint, then the infidelity of the representation is equal to the number of constraint violations.

5. Possible Modifications to the Measure of Representation Infidelity

One minor modification is a weighting of constraint violations as a function of the graph-distance of the offending vertices from the edge endpoint in question. We have experimented with two such possibilities, of which the most interesting one assigns an infidelity penalty that increases monotonically with the graph-distance between the vertex and the endpoint. Comparison of these variations while experimenting at the TX-2 console has revealed no significant reason to modify the (conceptually simplest) definition in this way.

A second minor modification may best be explained in terms of the constraint interpretation. The present scheme assigns a set of constraints to each pair of vertices connected by an edge (graph-distance one unit apart). In the generalization, a set of constraints might be assigned to each pair of vertices. For example, consider two vertices, a and b, two graph-distance units apart. One might consider the constraint that vertices at graph distance $d \geq 3$ from vertex a be further from vertex a than vertex b (as measured in a particular planar metric). Figure 13 demonstrates with a trivial example that there exist graph representations for which the infidelity measure is zero but for which such an extended measure would not be zero. Notice that v_4 is closer to v_1 than v_3 is.

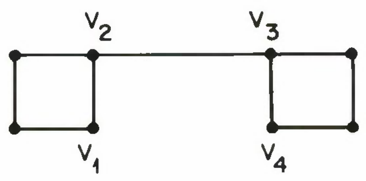


FIGURE 13

The most interesting modification is the selective elimination of certain subparts (subpatterns) of a graph from inclusion in the calculation. There are two chief reasons for such a procedure. A particular pattern may be ignored in the infidelity calculation if its occurrence in the graph allows a trivial solution to the problem of positioning it to obtain a clear representation.

A vertex of degree one, for example, may be positioned arbitrarily close to the vertex to which it is connected. Certain chains of vertices of degree two may be identified and positioned quite easily and thus could also be ignored in the infidelity calculation. In our work, we have always omitted vertices of degree one from the computation.

It is hoped that ignoring certain subpatterns would also allow more interesting theorems on infidelity to be proved. One possible example is a decomposition of a graph structure into portions containing cycles and those that are "tree-like" (see Section C.8). It is clear that the interesting questions of representation occur in those graph portions containing cycles. It would be most remarkable if such selective elimination of patterns from the infidelity calculation would result in a measure that could be related somehow to the minimum intersection criterion.

6. Summary of the Advantages of the Infidelity Measure

(a) Clear intuitive simplifications of a graph representation (by hand at the TX-2 console) usually cause a decrease in infidelity. Among the measures that we have considered, infidelity as defined and the criterion of the number of intersections show the effect best.

(b) The measure is interesting mathematically. It may go to zero for a wide class of representations. In some cases the infidelity of a graph may be proved to be zero or non-zero (see Section C.8). Even if the minimum value is non-zero it may, if calculable, be useful as a standard to evaluate the degree to which a representation has been simplified.

(c) Its calculation is easily mechanized; given an appropriate data structure, it is easily updated after minor changes in the representation.

7. Summary of Those Features of the Infidelity Measure That are Curious or Incompletely Understood

(a) It appears unlikely that there exist "bad" representations with low infidelity. Experimenters at the TX-2 console who have tried to ruin the clarity of a representation have usually caused the measure to increase. As a final test, however, one must program mechanical infidelity-reducing algorithms and examine the representations they produce.

(b) Step-by-step simplification of many graphs appears to attain a plateau in which further rearrangement does not lower the infidelity of the representation or improve its clarity. It is unknown whether or not such apparent plateaus are near the infidelity of the graph.

(c) One can sometimes lower the infidelity while adding apparent intersections. This phenomenon may occur when we move a vertex away from the loci of influence of several others.

(d) The infidelity of any representation of any complete graph is zero.

8. Theorems Characterizing the Infidelity Measure

(a) The infidelity of a representation is independent of the total scale. Proof: Obvious.

(b) Any representation of any complete graph has infidelity zero. Proof: In a complete graph, every two vertices are separated by one graph-distance unit. Therefore, there can be no constraint violations.

(c) Any graph with four or fewer vertices has infidelity zero. Proof: By exhaustion.

(d) There exist simple planar graphs with non-zero infidelity. One with five vertices is the "two house, three utility" graph shown in Figure 14.

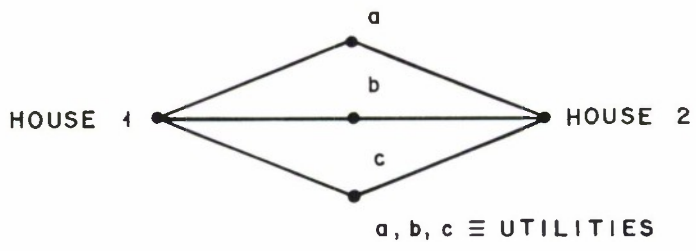


FIGURE 14

Proof: With no loss in generality, we may assign the two houses positions anywhere in the plane. If the infidelity of a representation is to be zero, then utilities a, b, and c must all lie within the shaded sector of Figure 15. There are three cases, corresponding to the location of utility a above the line between the two houses (l_h), on the line, or below the line.

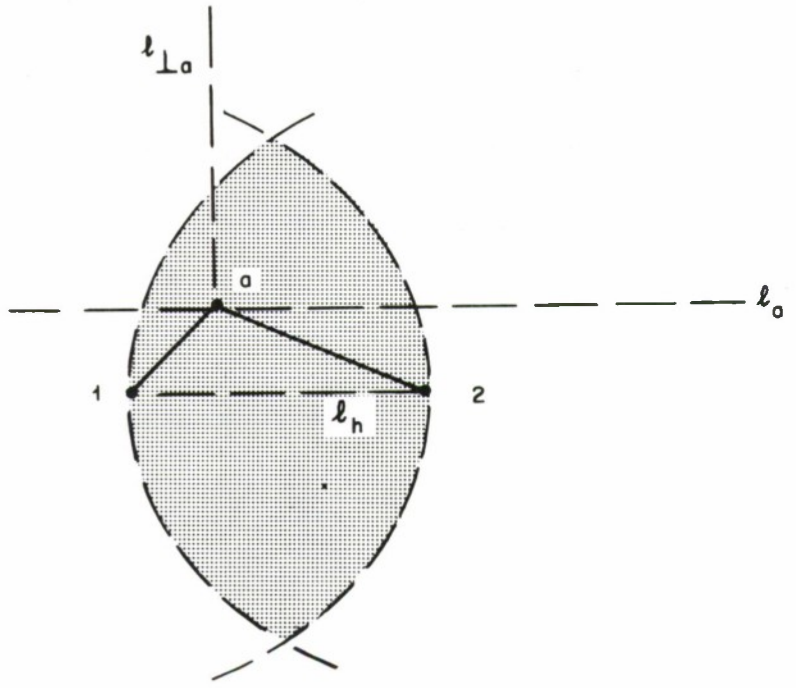


FIGURE 15

We consider the first case, in which a is positioned arbitrarily above l_h within the sector. Obviously utility b cannot be located either between l_a and l_h or on l_a without introducing a constraint violation. If utility b is located on line $l_{\perp a}$ (the normal to l_a through a) above l_a, there is a violation. Finally, if b is located above l_a to the right (left) of $l_{\perp a}$, there are violations involving the edge from vertex b to vertex 1(2).

Thus, b must be located in the half-sector below l_h. By analogous reasoning, c cannot now be located anywhere in the sector without violating a constraint.

Similar reasoning may be applied to the other two cases. Note: For other examples, consider why the theorem (f) clearly fails with very dense trees.

(e) The smallest non-planar graph with non-zero infidelity is the Kuratowski, three houses, three utilities graph shown in Figure 16.

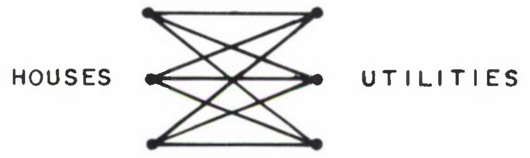


FIGURE 16

Proof: The proof follows from theorems (b) and (d), Kuratowski's theorem, and the fact that the addition of a new vertex and six new edges does not <u>in this case</u> reduce the infidelity of any of the edge-endpoint pairs already included in the two houses, three utilities graph.

(f) A finite chain in which every vertex has degree less than or equal to two has zero infidelity. Proof: Obvious.

D. HEURISTICS FOR SIMPLIFICATION OF A GRAPH REPRESENTATION

After considerable experience with graph simplification by single-step vertex movement, we have developed a collection of powerful heuristics for this task. As these may be of interest to students of cognitive processes and of artificial intelligence, we shall present each heuristic with a short discussion and any relevant theoretical results.

1. Isolate maximal dangling trees.

Consider a connected graph containing at least one cycle. We define a maximal dangling tree (m.d.t.) by the following procedure: Choose a non-empty set of vertices V with the properties that:

(a) No one of the vertices may be included in a cycle in the graph;

(b) the addition of any other vertex adjacent to one of the vertices in the set V violates condition a.

If such a set exists, the m.d.t. is formed from all the edges incident to any of the vertices in V. So that it be a subgraph, there must be included at least one vertex not in the set V, but adjacent to a vertex in V. Such vertices shall be called attachment points, and the existence of at least one corresponding to each m.d.t. is guaranteed.

In Figure 17, the sets of vertices {a,b}, {e}, and {h,i,j} may be used to form the only maximal dangling trees. c, d, g, and k are attachement points. The m.d.t.'s are {(a,c), (b,c)}, {(e,d)}, and {(g,h), (i,h), (j,h), (k,h)}.

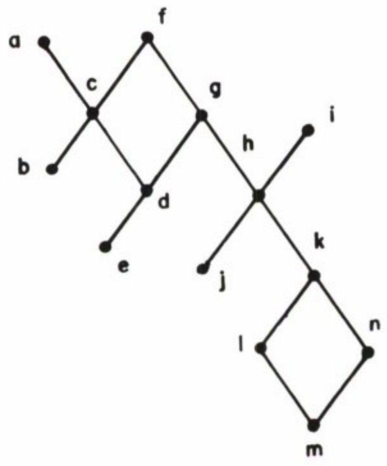


FIGURE 17

Theorem: If a connected graph contains at least one cycle and at least one maximal dangling tree, it possesses an articulation point (a point whose removal disconnects the graph). In fact, every attachment point is an articulation point.

Proof: Choose any attachment point p, and assume that it is not an articulation point. It is possible to choose two vertices adjacent to p, one included in a m.d.t. but not in any cycle, which we shall denote by q, and one not included in a m.d.t., which we shall denote by r. (Why?) If p is not an articulation point, then there exists a path from q to r not passing through p. This implies the existence of a cycle including this path and the edges (q, p) and (p, r). (Why?) Thus, the assumption that q is not included in any cycle is contradicted, and p must be an articulation point.

The above result and experimentation at the TX-2 display console justify an early isolation of m.d.t.'s, and their temporary "removal" from further simplification considerations. At the conclusion of the procedure, it is easy to merge the m.d.t.'s with the other components of the graph in a manner consistent with one's intuitive notions of representation clarity.

It is interesting to note that this "factoring" of the graph into its "cyclic" components and its "cycle-free" components is closely related to the notions of the homomorphic "leaf" and "lobe" images of a graph, discussed by Ore in his text.[3]

2. Find a vertex with a large connection vector, and move it along that path.

An example of a "connection vector" would be the vector sum of each edge incident to a vertex, in which each vector is directed along the edge with magnitude equal to the length of the edge in the planar metrix. A simple-minded heuristic that has been used with considerable success at the console is the identification of a vertex with a large connection vector, and subsequent movement of the vertex along the vector in order to reduce its magnitude. The identification of an appropriate vertex is accomplished instantaneously and accurately by eye; it would likewise be easy to automate.

Care must be exercised in the application of this heuristic to avoid vertex congestion at the center of the display. This may occur if too many vertices are moved into the center to reduce their connection vectors. In terms of the infidelity measure, these vertices have been brought too close to many others to which they are not directly connected. That it rises sharply in such cases of extreme congestion is a good feature of the measure.

3. Reduce intersections by local vertex traversal of boundaries.

Figures 18 and 19 illustrate the application of this heuristic, a very powerful one in certain special cases. Notice the intersections of the edge labeled e.

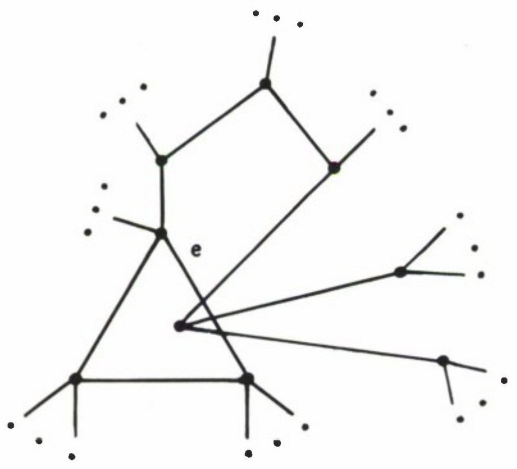


FIGURE 18

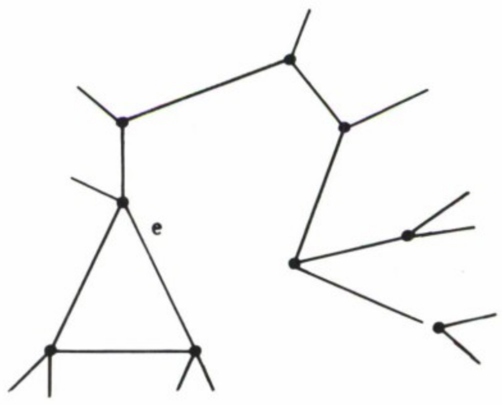


FIGURE 19

4. Identify interesting subpatterns and manipulate them in three dimensions.

Although the display is two-dimensional, experienced observers often begin to visualize patterns in three dimensions. This seems to facilitate the perception of certain advantageous, large-scale, reorganization of the display. Although it is difficult to describe any precise heuristics, a few examples may illustrate the phenomenon.

One occasionally recognizes that an entire subpattern is poorly positioned and should be moved. Its destination may be defined by relationship with another subpattern. For example, a certain subgraph may be moved "inside" a boundary (cycle) elsewhere in the graph. A deformation of a representation may sometimes take on a clear intuitive significance, such as pulling a subpattern, (for instance, a chain) through some loop or weaving it around a boundary, twisting a subpattern around an axis, or reflecting one with respect to another axis.

Such cognitions are three-dimensional in two senses. They are based on simultaneous perceptions of several distinct subpatterns, which must in some sense be visually separated. In addition, they involve transformations of a representation which are conceived as a continuous mapping of the graph from the plane into three-space and finally back into the plane.

E. PROPOSED EXTENSIONS TO THIS WORK; MORE UNSOLVED PROBLEMS

1. The design of mechanical algorithms that simplify a graph representation by attempting to reduce its infidelity or the number of edge intersections, or both, should be investigated. This is in a sense a classical "hill-climbing" problem; graph theory should be used to investigate the mathematical properties of the hills. How can the absolute minima be located? Mechanical simplification would reveal whether or not minimizing only infidelity or intersections yields intuitively clear representations. A technique we should like to explore is initial reduction of the infidelity, then reduction of the number of intersections, and then further reduction of the infidelity.

2. It would be desirable to characterize the relationship between infidelity and intersections in a more interesting mathematical manner than has been accomplished to date.

3. Can one develop an axiomatic approach to measures of representation clarity in such a way that the axioms define a unique measure? How would this compare to the infidelity criterion? (An example of this procedure is the derivation of Shannon's measure of information as the unique function satisfying certain axioms.)

4. Of great mathematical interest would be some results characterizing the noted relationship between infidelity measures on graph representations, inequalities, and the satisfaction of constraints.

5. Simple data structures can be adequately used to represent a graph as a set of vertices and interconnecting edges. Such a structure can be updated easily after a representation is modified. If, however, one attempts to store explicit information about the structure of the "faces" defined by graph edges and their interconnections, a more complicated data structure is needed. Updating the information also becomes more difficult. In more general form, the topic to be studied is that of data structures for computer manipulation of graphs imbedded in surfaces.

6. Our investigations have dealt entirely with random graphs. If graph representations were used to display complex data structures, such as appear in list or ring processing languages or in certain

debugging applications, then one may not wish to ignore features of the graph's structure in generating the display. The graph may possess symmetry properties; there may be application-dependent constraints on the characteristics of the display. For example, a display may be required to possess symmetry with respect to a certain axis (set of edges); alternatively, such a set may be required to be horizontal. The generation of clear planar representations of complex graphs upon which certain symmetries and constraints are imposed is an important topic for future study.

7. Finally, one must investigate the design of a graph-theoretic processor and linquistic facility to enable a mathematician to converse easily with a computer about graph representations. Ideas relevant to this goal are developed in a broader context in Chapter II.

II. THE USE AND DESIGN OF A GRAPH THEORY PACKAGE

Experience with the graph representation package has demonstrated at least three distinct ways in which a graph theory package may be useful to a student or researcher:

1. The construction of examples of graphs satisfying certain constraints (with specific properties);
2. experimentation with these examples, in which the computer could display graphs, calculate interesting functions on them, and modify them to produce new examples; and,
3. the investigation of graph-theoretical algorithms ("constructive" graph theory).

A. THE LEVEL OF GRAPH STRUCTURE

At this level the mathematician, who need not necessarily be a programmer, may speak in a natural language of vertices, edges, and graphs, may state their graph-theoretical properties, and may specify certain primitive iterative or recursive (non-graph-theoretic) procedures that operate upon the graphs.

As primitive graph-theoretical terms, <u>vertex</u>, <u>edge</u>, <u>set of edges</u>, <u>graph</u>, <u>subgraph</u>, <u>value associated with a vertex</u>, <u>connected to</u>, <u>face</u>, and perhaps a few others would suffice. It is true that sophisticated graph theory is not readily expressed in terminology that simple. We have, however, adopted the philosophy that the success or failure of a graph theory package lies in its expandability and flexibility, and not in the size of its pre-programmed library of graph-theoretic primitives. This implies that the major effort should be expended in increasing the power and flexibility of the primitive operations which act on primitive structures (edges, sets of vertices, etc.), and the variety of computational "data structures" that represent them in the implementation. The goal is that the user be able to synthesize easily his own desired set of graph-theoretical algorithms.

The primitive capabilities must include at least the following:

1. Mechanization of the logical connectives and logical quantifiers over finite domains (for example, "for every edge of one set incident with some vertex in another");

2. mechanization of set-theoretic operations such as finding the union of two sets and testing if one set is a subset of another;

3. iterative procedures on sets of values associated with vertices or edges (for example, searching and sorting);

4. a flexible class of loop-control and branching capabilities; and,

5. recursive use of all subroutines.

Graph theory becomes interesting when one considers certain properties of, for example, a vertex, a set of edges, a subgraph, or an edge with respect to a subgraph. A vertex may have degree K; a set of edges may form a cycle; a subgraph may be bipartite; an edge may be terminal with respect to a subgraph. The study of a particular graph becomes interesting when it is constructed to satisfy certain constraints or is found to possess certain properties, for it may then shed light upon the characteristics of some general class of graphs.

As a mathematician explores a particular graph, the computer system should develop a store of properties relevant to that graph and its subparts. The concepts underlying the "associative language" now being developed at Lincoln Laboratory and Stanford University are relevant here.[4] Such a language may form an appropriate base in which to embed the store of graph properties.

B. THE LEVEL OF IMPLEMENTATION

Because the computational demands on a graph theory package will be great, questions of efficiency are critical to effective system design. Thus, the programmer maintaining the level of graph structure should base the implementation on work quite close to the machine. We tentatively suggest for consideration , the Bell Telephone Laboratories' Low-Level Linked List Language, L^6.[4] It consists of a set of short subroutines with at least three highly desirable features:[5]

1. A wide class of field-handling capabilities,
2. user choice of the locations and sizes of his data and pointer fields, and
3. run-time redefinition of fields which causes run-time recompilation of the corresponding field-manipulating routines.

C. THE LEVEL OF GRAPH-THEORETICAL ALGORITHMS

Finally, we shall discuss the synthesis of features at the level of graph structure into a powerful descriptive and algorithmic graph-theoretical language. The most difficult and profound issue is simply stated -- how to name subgraphs and subparts of a graph.

It is true that, in order for any subgraph name to have (semantic) meaning, there must exist an algorithm that evaluates or finds the named subgraph. Yet, an explicit name, such as "vertex A" or "edge #17", requires only a trivial mapping from the "name space" into the "location space". Such explicit names are rare in graph theory; more natural are expressions such as "the vertex (vertices) of degree three within two graph-distance units of vertex #14", "the section graph formed from the edges in a maximal cycle", or "the Hamiltonian path of minimum length".

In the first example, a set of vertices is defined by the simultaneous satisfaction of two predicates applicable to vertices. The second case requires the construction of all cycles, the evaluation of (search for) the maximal one, and the formation of a section graph from a set of edges. The third case possesses similar complexity.

Thus, the mathematician must speak to a processor that can decompose such statements into a sufficient set of search routines and calls of already-existent, graph-theoretic functions, logically combined and nested appropriately. The language must allow not only these natural descriptive features, but also all the procedural features necessary to name (and save) intermediate results of any kind. One must be able to mix descriptive and procedural modes with no artificial restrictions. The system should also free the mathematician from much of the usual programming burden of error-recovery by informing him when a search has failed or several maxima are found, and allowing him to reformulate how to proceed further. On-line interaction is naturally a prerequisite.

These capabilities should allow the mathematician easy expression of algorithms that evaluate functions of a graph (maximal circuit, all centers, colorings with properties P and Q), and also of algorithms that construct simple examples of graphs satisfying certain constraints. Investigations in the latter domain would unveil particularly challenging and undoubtedly new classes of graph-theoretical problems.

D. THE ROLE OF COMPUTER GRAPHICS

In conclusion, we emphasize the obvious importance of graphical communication to the utility of a graph theory package. A valuable design philosophy is that of the Graphical Service System[6] under construction at Lincoln Laboratory, that the graphical conventions of a package be under user control, and that they may be established without changing the syntax and the semantics of the language.

Several desirable features may be noted here:

1. Automatic or partially automatic simplification of planar graph representations,
2. the inclusion of elementary constraints on such representations,
3. graphical subroutine conventions under user control,
4. graphical subroutine output conventions under user control, and
5. visual graph-probes ("bugs") to trace graph-theoretical algorithms.

An example will clarify items 3, 4, and 5. Assume that a subroutine exists which accepts a cut-set and forms two disconnected subgraphs. The user may define a pair of light-targets which, when sandwiched around a set of light pen bits on graph edges, accepts these edges as input. It checks that they form a cut-set, or requests a cut-set routine if none exists, and eventually feeds them to the appropriate subroutine. As an output convention the user may specify that the cut-set flash and that the <u>vertices</u> of one subgraph be circled. If the algorithm itself is under study, he may request that changes in interesting sets of vertices or edges be displayed in slow motion.

ACKNOWLEDGEMENT

Special thanks are due Dr. Jerome A. Feldman, under whose valuable guidance and encouragement the research was conducted. I am also indebted to the Digital Computers Group of the MIT Lincoln Laboratory for its support and to its members for their enthusiastic as well as critical interest in this work.

BIBLIOGRAPHY

1. Blazek, J., and Koman, M., "A Minimal Problem Concerning Complete Plane Graphs", Theory of Graphs and Its Applications, Proceedings of the Symposium held in Smolence, Academic Press, New York, (1963) p. 113-117.
2. Busacker, R. G., and Saaty, T. L., Finite Graphs and Networks, McGraw Hill, New York, (1965).
3. Feldman, J. A., "Aspects of Associative Processing", MIT Lincoln Laboratory Technical Note 1965-13, (21 April 1965).
4. Knowlton, K., "A Programmer's Description of L^6", Communications of the ACM, 9, Number 8, (August 1966).
5. Ore, Oystein, Theory of Graphs, American Mathematical Society, Providence, Rhode Island, (1962).
6. Roberts, L. G., "A Graphical Service System with Variable Syntax," Communications of the ACM, 9, Number 3, (March 1966).
7. Zarankiewicz, K., "On a Problem of P. Turan Concerning Graphs", Fundamenta Mathematica, XLI (1955), p. 137-145.

DOCUMENT CONTROL DATA – R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Lincoln Laboratory, M.I.T.	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP None

3. REPORT TITLE

Planar Representations of Complex Graphs

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Note

5. AUTHOR(S) *(Last name, first name, initial)*

Baecker, Ronald M.

6. REPORT DATE 6 February 1967	7a. TOTAL NO. OF PAGES 44	7b. NO. OF REFS 7

8a. CONTRACT OR GRANT NO. AF 19(628)-5167	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Note 1967-1
b. PROJECT NO. ARPA Order 691 c. d.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* ESD-TR-67-61

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency, Department of Defense

13. ABSTRACT

This paper's major topic is the problem of constructing good two-dimensional representations, using straight lines as edges, of complex, generally non-planar graphs. Two precise measures that correlate well with the subjective "clarity" of planar representations of randomly generated graphs are proposed. One is the number of edge intersections that do not correspond to graph vertices. The second measure, called infidelity, quantifies the degree to which the distance between pairs of vertices (in a particular metric in the plane) corresponds to their graph-theoretical distance from one another.

Some experiments in graph representation, using the TX-2 computer on-line display, are described. Heavy emphasis is given the definition of the infidelity measure, the nature of its correlation with graph "clarity," and some theorems which add to its characterization. Included is a discussion of heuristics for the simplification of a graph representation.

In conclusion, there is presented a sketch of a model of a graph theory computer package that should be useful to a researcher or student of graph theory.

14. KEY WORDS

planar representations
graphs
TX-2