An Optimal Algorithm to Detect a Line Graph and Output Its Root Graph

PHILIPPE G. H. LEHOT

Fireman's Fund American Insurance Companies, San Francisco, California

ABSTRACT. Given a graph H with E edges and N nodes, a graph G is sought such that H is the line graph of G, if G exists. The algorithm does this within the order of E steps, in fact in $E + O(N)$ steps. This algorithm is optimal in its complexity.

KEY WORDS AND PHRASES: algorithm, line graph, root graph of a line graph, Hamiltonian cycle, Hamiltonian-Eulerian duality, node-edge transformation

CR CATEGORIES: 5.32

Introduction

Only loopless undirected graphs without multiple edges are considered. We shall further limit attention to connected graphs without loss of generality.

The line graph H of a graph G is a graph where each node corresponds to a different edge of G, with two nodes of H adjacent if and only if the corresponding edges of G have a node in common. With G having e edges and n nodes, H will have e nodes and E edges where $E = \sum_{i=1}^{n} [d_i(d_i - 1)/2]$, d_i being the degree of node i in G. G is called the root graph of H, and H is the line graph of G. A star graph is a graph where all edges are incident to one common node. Each star subgraph of G defines a maximal complete subgraph of H, called clique of H. A node of G will be called "well defined" when the whole corresponding clique in H has been found and named by the algorithm. A node of H is "half-named" when it corresponds to an edge of G, having already one of its two nodes well defined. A node of H is fully named when it corresponds to an edge of G whose two nodes are well defined.

As an example, the nodes of G will be $1, 2, 3, 4, \ldots, n$, and the node of H corresponding to the edge of G joining node 1 and node 2 will be called "1–2."

A node of H will henceforth be a pair of numbers written in increasing order. The algorithm we present proceeds by first assuming H is a line graph, and defines in a consistent way a graph G such that H is necessarily the line graph of G if H is really a line graph.

Then to find out if indeed H is a line graph at all, we shall generate the line graph of G and compare it with H. The answer follows, for we know by [1] that except in a trivial case (triangle and star with three branches) the root graph of a line graph is unique up to isomorphism. (In fact it should not be very hard to deduce this result from a careful study of our own algorithm.) Beineke in [2] has proven that a graph H is a line graph if and only if it does not contain any of nine forbidden subgraphs, none of which has more

Author's address: Fireman's Fund American Insurance Companies, P.O. Box 3395, San Francisco, CA 94119.

than six nodes. This immediately guarantees the existence of an efficient[1] algorithm to find whether or not a graph is a line graph. Van Rooij and Wilf [3] have proven that H is a line graph if and only if two conditions are satisfied: (i) it does not have $K_{1,3}$ as an induced subgraph; and (ii) two odd triangles[2] having a common edge induce a subgraph which is K_4.

We provide a computer-oriented characterization of a line graph which is an "order-of-E-algorithm" where E is the number of edges of the given graph H. Furthermore, our algorithm will output the root graph G of H whenever the latter is a line graph.

First Step of Algorithm

Choose any two adjacent nodes of the graph H. Call these nodes "basic nodes" and name them 1–2 and 2–3. They define the clique 2, i.e. the set of edges of the root graph G which are incident to node 2 of the root graph G. List all the nodes adjacent to both basic nodes (and name them at the same time). A node of the form 1–3 is called a cross node. If we can determine which, if any, of the basic nodes is a cross node, the rest of the naming process is straightforward: all other nodes are named 2–4, 2–5, 2–6, ..., etc., so that all of them together with the basic nodes form the clique 2.

One essential problem is to discover which node, if any, is the cross node of the two basic nodes. We shall denote X the set of all nodes adjacent to both basic nodes. Any cross node is in X. In what follows, we shall consider three main cases, according to the value of the cardinality of set X.

Case 1. $|X| = 1$. We shall call x the unique node in X.

Case 1.1. There is a node y adjacent to one vertex of the triangle (x, 1–2, 2–3) which is not adjacent to any other, i.e. the triangle is odd. Then, necessarily, x = 2–4.

PROOF. If x were 1–3, then y would be adjacent to two nodes of the triangle.

Case 1.2. The triangle (x, 1–2, 2–3) is even, i.e. any node of H is either adjacent to zero or two nodes of the triangle.

Let Y be the set of nodes adjacent to both x and 1–2 excluding 2–3, and let Z be the set of nodes adjacent to both x and 2–3 excluding 1–2.

(1) $|Y| \geq 2$ or $|Z| \geq 2$; then x = 1–3.

PROOF. Indeed if x were 2–4, then x would not be the only node adjacent to both basic nodes, since at least one of two nodes adjacent to 1–2 and x or to 2–3 and x would have to contain 2 in its name i.e. to belong to clique 2. Then x being no longer unique, case 1 is contradicted.

(2) $|Y| = |Z| = 1$, say $Y = \{y\}$ and $Z = \{z\}$. Either (I) or (II) or (III) following is true.

(I) y and z are not adjacent, and then x = 1–3.

PROOF. If x were 2–4, then x would be 1–3 and y would be 3–4, and x and y would be adjacent.

(II) y and z are adjacent, and then either H has only five nodes (and then x may or may not be the cross node of the basic nodes, but it does not matter, as illustrated in Figure 1).

(III) H has more than five nodes, and necessarily x = 1–3.

PROOF. If a were 2–4, any node adjacent to both y and z would be adjacent to one of the nodes of the triangle (x, 1–2, 2–4) without being adjacent to the two others, and we ruled out the odd triangle case (1.1).

(3) $|Y| + |Z| = 1$. Say, for instance, $Y = \{y\}$ and $Z = \varnothing$, then if H has only four vertices, the name of x does not matter as seen in (2) (see Figure 1). If H has more than four vertices, the only connection with the rest of the graph is through adjacency to y = 1–4, and more specifically through node 4 of the root-graph G, because the triangle defined by x and the basic nodes is not odd. Hence x = 1–3.

[1] Efficient means where the order of complexity is bounded by a polynomial in n; here a polynomial of degree 6.

[2] An odd triangle is a triangle which is not even, and an even triangle is such that every node is adjacent to two or zero vertices of the triangle.

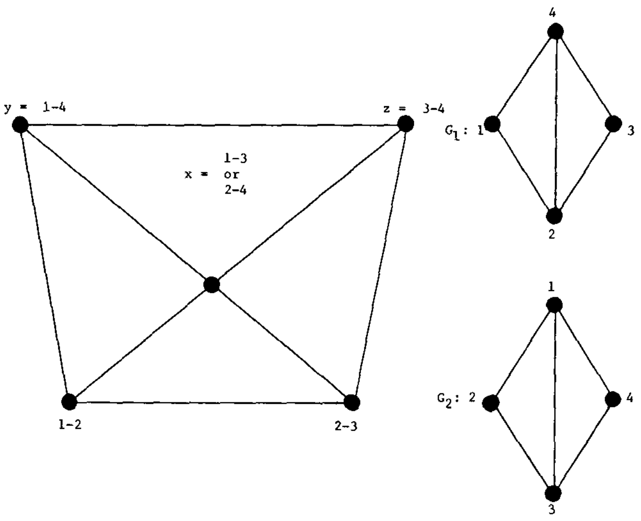

FIG. 1

(4) $|Y| = |Z| = 0$. Then H is a complete graph with three vertices because we assumed H to be connected and case 1.1 is already ruled out at this point. Hence we know this case to be the trivial case when two root graphs exist and exit with that answer.

Case 2. $|X| = 2$, say $X = \{x, y\}$.

(1) If x and y are adjacent, then there is no cross node.

(2) If x and y are not adjacent, then one of the two is the cross node:

(I) We deal with two triangles: (x, 1–2, 2–3) and (y, 1–2, 2–3). If we find one of them to be odd, the node of X (x or y) in this triangle must belong to clique 2. Hence, the other node of X (y or x) must be the cross node 1–3 of the two basic nodes.

(II) Both triangles are found even. Then we claim:

(i) There is at most one node adjacent to 1–2 and x.

(ii) There is at most one node adjacent to 2–3 and x.

(iii) There is at most one node adjacent to 1–2 and y.

(iv) There is at most one node adjacent to 2–3 and y.

PROOF. If there are, say, two nodes **a** and **b** adjacent to both 1–2 and x, then one of them is bound to be different from their cross node (unicity of the cross node). Then if x were 2–4, **a** would not be the cross node of 1–2 and 2–4. Hence **a** would belong to clique 2, a contradiction because **a** could not be y since x and y are not adjacent. If x were 1–3, then **a** or **b** would belong to clique 1 (since the cross node of 1–2 and x = 1–3 exists already: 2–3). These two nodes **a** and **b** would be named 1–5 and 1–6 (say). At least one of them could not be adjacent to y = 2–? and therefore the triangle (y, 1–2, 2–3) would be odd—contradiction.

The other three propositions are proven in a similar way.

Claim. Now both triangles being even the only remaining possibilities for H are the following (Figure 2):

PROOF. If there is a node adjacent to 1–2 and x, then it must be adjacent to 1–2 and y since the y-triangle is also even. If there is a node **a** adjacent to 1–2 (and x and y) and another node **b** adjacent to 2–3 (and x and y), then these two nodes must be adjacent as the situation is symmetric and they both belong to clique 4. (See Figure 3.)

Case 3. $|X| \geq 3$. All we need to do is to check for each node of X, whether or not it is adjacent to some other node of X. If we find such a node **a** to be nonadjacent to some node **b**, then either **a** is the first node of X to be investigated with respect to **b**, and we shall suspect both **a** and **b** as candidates to be cross nodes, or **a** is not the first node of X whose adjacency to **b** is investigated, in which case **a** must be the cross

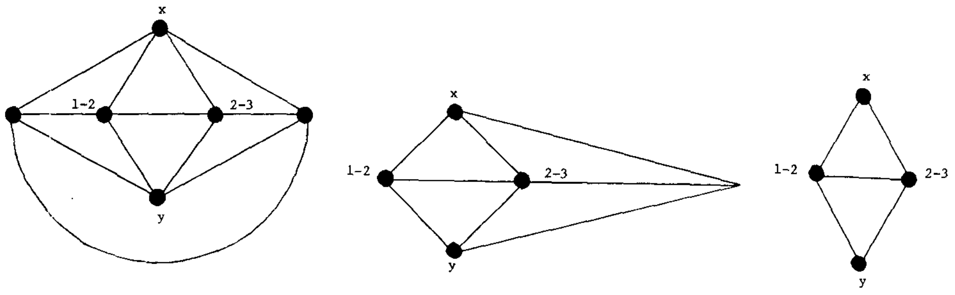

FIG. 2

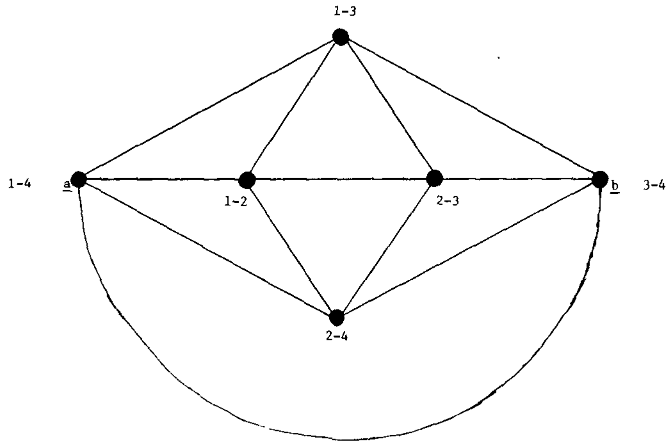

FIG. 3[3]

node. In the first case, where **a** and **b** are in a similar (symmetric) position, we need only to break the tie. We need only one more checking: the adjacency of **a** and some other node of the set X (which must exist since $|X| \geq 3$. In other words, **a** is the cross node if it is not adjacent to some third node, **b** is the cross node otherwise.

Main Step of Algorithm

The first step discovered the first clique and named all nodes of this clique, and in addition, half-named all adjacent nodes. It also named the cross node 1–3, if it exists, of this first clique. From then on, the two associated cliques are discovered. These associated cliques are defined (if 1–3 exists) by the following basic nodes: (2–3, 1–3) on the one hand, and (1–2, 1–3) on the other. Note that in the subsequent discovery of these two associated cliques, no node which is already fully named is considered; in particular, the cross nodes will not be considered. Hence the naming process is straightforward.

The main step consists in choosing two basic nodes, adjacent of course, one of which is fully named and belongs to an already discovered clique, the other only half-named.

THEOREM. *There is no cross node to be discovered and named in the main step clique.*

PROOF. Indeed, assume there is a cross node. Such a cross node will then belong to the already discovered clique to which one of the basic nodes belongs. Hence it is already fully named. Q.E.D.

As a conclusion, only once do we look for a cross node—in the first step of the algorithm.

THEOREM. *The choice of the two basic nodes in the main step is consistent.* (*a*) *If no such couple of basic nodes exists, the algorithm has already terminated, i.e. all nodes are already fully named.* (*b*) *There is no contradiction in the naming process.*

[3] Note: Underscored characters in the figures correspond to boldface ones in the text.

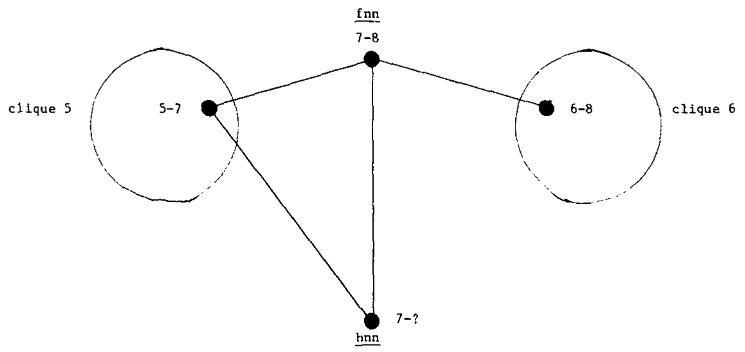

FIG. 4

PROOF OF (a). Assume not all nodes are fully named. Then, since the graph H is connected, there is somewhere a fully named node **fnn** adjacent to a half-named node **hnn**. Assume now that there was no such couple of basic nodes as described earlier. That is to say, the **fnn** does not belong to an already discovered clique. Then such an **fnn** could only have been adjacent to two different cliques, say cliques 5 and 6. The name of the **fnn** is therefore, say 7–8, as shown in Figure 4.

The name of **hnn** is, without loss of generality, say, 7–? That is to say that the **hnn** has been half-named because of adjacency to a clique which is one of the two cliques 5 and 6. Say, without loss of generality, that it was clique 5, and hence it had to be the node 5–7 which half-named **hnn**. We conclude the argument by noting the contradiction: 5–7 and 7–? are a contradiction to the assumption of the nonexistence of the pair of basic nodes as defined in the main step. Q.E.D.

PROOF OF (b). There is no contradiction in the naming process itself. Indeed, only "not-fully-named" nodes are considered. As soon as a node is fully named it is no longer considered by the naming process. Hence no contradiction can arise. Q.E.D.

The Algorithm

Step 1. Pick up two adjacent nodes. Name them 1–2 amd 2–3. They are the two basic nodes to start with.

Step 2. Find all nodes adjacent to both basic nodes. If there are three or more nodes, go immediately to step 5. If there is none, go to step 6.

Step 3. There is only one such node. Call it x. If there is a node adjacent to one node of the triangle without being adjacent to any other node of the triangle,[4] then x = 2–4. Then, go to step 6. Otherwise x = 1–3 and go to step 7.

Step 4. There are two nodes adjacent to both first step basic nodes. Call them x and y. If x and y are adjacent, there is no cross node and go to step 6. If they are not adjacent, they constitute two triangles with the basic nodes, and we find out if there is an odd triangle.[4] If there is one, the corresponding summit, say x, is 2–4, and then y will be named 1–3 and will be the cross node. Go to step 6. Otherwise, both triangles are even. The only three remaining possibilities are as shown in Figure 5.

Step 5. There is a group of three or more nodes adjacent to both basic nodes. If there is a node **a** of the group which is not adjacent to a certain node **b** (chosen at random but once and for all) then either **a** is the first node of the group under investigation, or it is not. In the first case, the tie is broken by examining the adjacency of **a** to a third node of the group. If it is adjacent, then **b** is declared the cross node. If it is not, then **a** is declared the cross node. If there is no cross node go to step 6. If there is one, go to step 7.

Step 6. All the nodes of the clique are named, and they half-name successively all the "not-yet-fully-named" nodes which are adjacent to them. Go to step 8.

Step 7. All the nodes of the clique are named and the cross node is fully named. Only the nodes of the clique will be used to half-name successively the "not-yet-fully-named" nodes which are adjacent. Then the two associated cliques are disposed of, and the naming process takes place as usual. Go to step 8.

[4] Note the naming process should be implemented simultaneously with the operation of finding out if there exists such a triangle called an "odd" triangle.

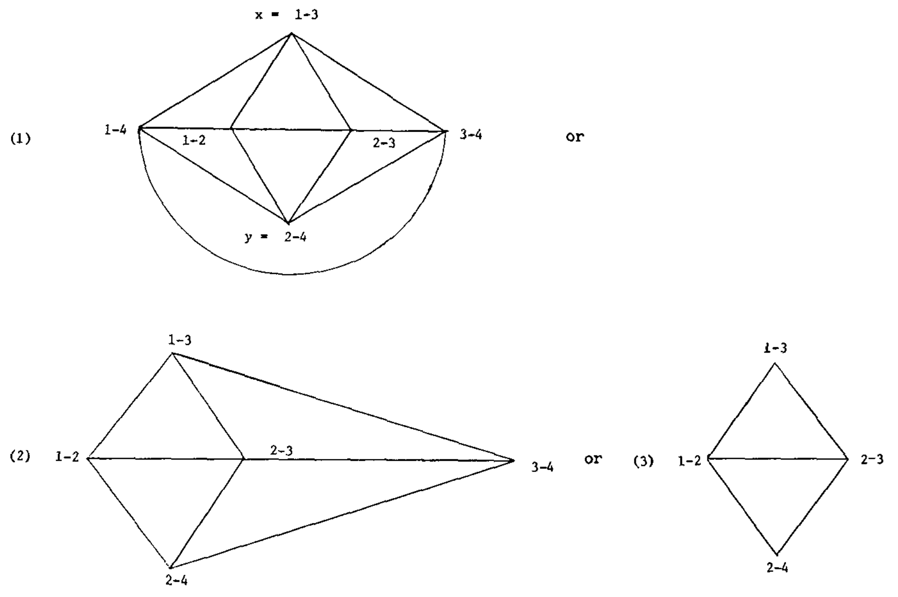

FIG. 5

Step 8. (Choice of a new couple of basic nodes.) If all nodes are fully named, go to step 9. Choose any half-named node. It is adjacent to a fully named node. If this fully named node does not belong to an already discovered clique, then there is a fully named node which does belong to an already discovered clique (which half-named the half-named node) and which is defined as the cross node of the two previous nodes. Hence, it is found in one or two steps. (Also the last couple of nodes involved in the half-naming process could be found on top of an ad hoc stack.) The couple of basic nodes name completely the basic node which was only half-named. Discover these nodes (among the LIST of not yet fully named nodes) which are adjacent to both basic nodes and half-name them with the clique number NCLQ, and complete their name. All nodes of the clique are now fully named. Write them off the LIST, and half-name all nodes adjacent to clique nodes with the number which is not the clique number and which is in their names. Then go to 8 again.

Step 9. Mark the edges of $L(G)$, the line graph of G, which are in H, until one edge of H is not in $L(G)$. If this happens: *Exit—H is not a line graph.* If this does not happen: *Exit—H is a line graph.*

Complexity of Computation

In the first part of the algorithm, the naming process easily indicates that the overall computation is of order N. Indeed any node is scanned at most twice: the first time to be half-named, and the second time to be fully named. It is no longer scanned afterward.

The second part of the algorithm which is the comparison between $L(G)$ and H takes at most E steps, where E is the number of edges of H. For the entire algorithm, we obtain $O(N) + \text{E}$.

Optimality

As far as the order of magnitude of the number of steps is concerned, this algorithm is optimal. Indeed, there is no way to find out whether or not N is a line graph and output its root graph G without looking at least once at all its edges. However, if we were not to output G, it is not clear that we should necessarily scan all edges.

Furthermore, the first part of the algorithm, that is the procedure to find the root graph of a given line graph, is itself optimal. Indeed, there is no way to accomplish this without scanning each node of H or, equivalently, each edge of G. This shows that the order of N is optimal.

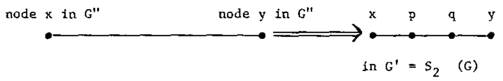

FIG. 6

Application

(1) *Hamiltonian graph.* If a graph H turns out to be a line graph, and if its root graph G is Eulerian (all nodes of G of even degree), then we can conclude that H is Hamiltonian. Furthermore, if H is a line graph such that all its cliques are disjoint—we mean all cliques of three nodes or more—then H is Hamiltonian *if and only if* G is Eulerian.

Here we mean two *cliques* are *disjoint* if no node of one clique is adjacent to any node of the other clique.

Indeed, such a graph H will be Hamiltonian if and only if a graph H', transformed from H by concatenating all paths of degree-2 nodes into only one degree-2 node, is Hamiltonian. This graph H' in turn will be Hamiltonian if and only if its root-graph G' is Eulerian, and the reason (see [4]) is that G' is indeed identical to $S_2(G'')$ where $S_2(G'')$ is the transform of some graph G'' by replacing each edge of G'' by three edges in series where the two middle nodes are degree-2 (see Figure 6).

It would be interesting to extend this method to cases when *almost* all the cliques of H are disjoint.

(2) *Clusters.* The clusters of a graph are often defined as their cliques. However, the discovery of cliques in an ordinary graph is not a very efficient process. On the other hand, we now have a very efficient tool to discover the cliques of a line graph. This leads us to consider a new definition of clusters for a class $C(L)$ of graphs. A graph belongs to $C(L)$ if and only if it is a subgraph of a line graph L, with the same nodes, but not containing edges which are not in L. The clusters of any graph of $C(L)$ may be defined as corresponding to the cliques of L. This leaves room for any cluster of a graph of $C(L)$ to grow, up to the clique of L. At this point, any more "growth" or addition of edges would necessarily destroy the superstructure L, and a new line graph L' should be defined, containing L and the last addition, thereby enlarging $C(L)$ into $C(L')$ which contains $C(L)$.

ACKNOWLEDGMENTS. My thanks are due to Dr. S. Kundu for helpful criticism, and to Professors R. Karp, E. Lawler, P. Spira, and E. Wong for their suggestions and encouragement. I owe the referee thanks for suggestions and corrections of errors.

REFERENCES

1. WHITNEY, H. Congruent graphs and the connectivity of graphs. *Amer. J. Math. 54* (1932), 150–168.
2. BEINEKE, L. W. Derived graphs of digraphs. *Beitrage zur graphentheorie*, H. Sachs, H. Voz, H. Walther, Eds., Teubner, Leipzig, 1968, pp. 17–33.
3. VAN ROOIJ, A., AND WILF, H. The interchange graphs of a finite graph. *Acta Math. Acad. Sci. Hungar. 16* (1965), 263–269.
4. HARARY, F. *Graph Theory*. Addison-Wesley, Reading, Mass., 1969, p. 81.
5. HARARY, F., AND NASH-WILLIAMS, C. ST. J. A. On Eulerian and Hamiltonian graphs and line graphs. *Can. Math. Bull. 8* (1965), 701–709.

RECEIVED AUGUST 1972; REVISED OCTOBER 1973