AN ALGORITHM FOR DETERMINING WHETHER THE CONNECTIVITY OF A GRAPH IS AT LEAST k

Shimon Even*

TR 73-184

September 1973

Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot, Israel

---

* This work was done while the author was with the Department of Computer Science, Cornell University, during the summer of 1973.

AN ALGORITHM FOR DETERMINING WHETHER THE CONNECTIVITY OF A GRAPH IS AT LEAST k

Shimon Even
Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot, Israel

__Abstract__:

The algorithm presented in this paper is for testing whether the connectivity, of a large graph of n vertices, is at least k. First the case of undirected graphs is discussed, and then it is shown that a variation of this algorithm works for directed graphs. The number of steps the algorithm requires, in case $k < \sqrt{n}$, is bounded by $O(kn^3)$.

AN ALGORITHM FOR DETERMINING WHETHER THE CONNECTIVITY OF A GRAPH IS AT LEAST k

Shimon Even

I. INTRODUCTION

Let G be a finite undirected graph with n vertices and e edges. We assume that G has no self-loops and no parallel edges. A set of vertices, S, is called a <u>separating set</u> if there exist two vertices $a,b \notin S$ such that all paths between a and b pass through at least one vertex of S. The <u>connectivity</u>, c, of G is defined in the following way

(i) If G is completely connected* then $c = n-1$.

(ii) If G is not completely connected then c is the least number of vertices in a separating set.

Menger's theorem [1] states that if the connectivity of G is c then for every two vertices a,b there exist c vertex-disjoint paths connecting a and b†; and conversely, if for every two vertices a,b there exist c vertex-disjoint connecting paths then the connectivity of G is at least c. Dantzig and Fulkerson [2] introduced the relation between connectivity and network flow. Thus, the Ford and Fulkerson [3] algorithm can be used to determine the connectivity of a graph. In fact the max-flow min-cut theorem (and algorithm) immediately translates to

* Each pair of vertices is connected by an edge. In this case G has no separating sets.

† Clearly, the vertices a,b are shared by all c paths, but no other vertex, and therefore no edge, is shared.

the following: The maximum number of vertex-disjoint paths connecting vertices a and b is equal to the minimum cardinality separating set between a and b, in case there is no edge between a and b; otherwise the number of paths is one more than the minimum cardinality of a set separating a from b after the edge between them has been deleted.

Thus, one can find the connectivity of a graph in the following way: For each pair of vertices find the maximum number of vertex-disjoint paths. The minimum value over all pairs is the connectivity.

For each pair of vertices we construct a flow-network whose number of vertices is $2n$ and the number of edges is $2e+n$. The capacities are all one* . Each labeling and augmenting path realization costs $O(e)$ steps, and it corresponds to one path between the two vertices. Since the connectivity can be as high as $n-1$, the whole procedure for finding the maximum number of vertex-disjoint paths connecting this pair of vertices is at most of cost $O(ne)$, or $O(n^3)$ if $O(e) = O(n^2)$.

Repeating this for all pairs will cost, then, at most $O(n^5)$.

Gomory and Hu [5] showed that the multi-terminal network-flow problem can be solved by considering only $n-1$ pairs of vertices instead of $n(n-1)/2$. (Their result is for undirected networks.) This means that the connectivity can be determined in at most $O(n^4)$ steps.

---

* For more details, see, for example, [4]. There, some of the capacities are infinite and some are unit. Changing them all to one unit does not change anything.

Assume now that we are not interested in the connectivity itself, but rather would like to check whether the connectivity is at least k, where k is much smaller than n.

It seems like the Gomory and Hu technique fails here because one has to find a minimum-cut through which one splits the sets of vertices created in their procedure. Thus, no reduction over the $O(n^4)$ is provided.

Kleitman [6] has shown a method which takes at most $O(k^2n^3)$ steps. Clearly Kleitman's algorithm is better than that implied by the Gomory and Hu method if $k < \sqrt{n}$.

In Section II I shall present a method which takes at most $O(kn^3)$ steps. Directed graphs are discussed in Section III.

It is proper to comment here that the case of $k=1$ is trivially solvable in $O(e)$ steps. The case $k=2$ was solved in $O(e)$ steps by Hopcroft and Tarjan [7] who proceeded to solve the case $k=3$ in $O(e)$ steps too [8]. Their methods are different from the ones described above. They use the powerful technique of Depth First Search (which was already known in the 19th century as a maze threading technique. See for example Lucas' [9] report on Trémaux's work). I do not believe that their methods will extend for higher k's.

II. THE ALGORITHM FOR UNDIRECTED GRAPHS

Let G be an undirected graph with n vertices and e edges. Let $L = \{v_1, v_2, \ldots, v_\ell\}$ be a set of vertices of G and u be a vertex of G not in L. Let k be a positive integer such that $k \leq \ell$.

Let us add to G a new vertex a and connect it by an edge to each of the vertices in L. The new graph, $\tilde{G}$, will be called the <u>augmented graph</u>.

<u>Lemma 1</u>: If in G each vertex $v_i (1 \leq i \leq \ell)$ can be connected to u via k vertex-disjoint paths then in $\tilde{G}$ there are k vertex-disjoint paths between a and u .

<u>Proof</u>: Assume not. Then there is a separating set S, $|S| < k$, such that all paths from u to a pass through at least one vertex of S . Consider the set of vertices, U, such that, like for u , all the paths from them to a pass through at least one vertex of S . None of the vertices in L can be in U , since each vertex of L is connected by an edge to a . Thus, there exists a vertex v in L which is not in U and not in S . Every path from u to v must pass through at least one vertex of S . Thus there cannot be k vertex-disjoint paths between u and v . A contradiction. Q.E.D.

Assume the set of G's vertices is $\{1,2,\ldots,n\}$. Let j be the least vertex such that for some $i < j$ there are no k vertex-disjoint paths connecting i and j in G .

<u>Lemma 2</u>: Let j be as defined above and $\tilde{G}$ be the augmented graph where $L = \{1,2,\ldots,j-1\}$. There are no k vertex-disjoint paths connecting a and j in $\tilde{G}$.

<u>Proof</u>: Consider a minimum separating set S, such that all paths between i and j pass through at least one vertex of S. It follows that $|S| < k$. Let U be the set of all vertices such

that all paths from them to i must pass through at least one vertex of S. Clearly $j \in U$. If a vertex $j' < j$ is in U then there are no k vertex-disjoint paths from j' to i, and j-th choice was erroneous. Thus, j is the least vertex in U, or $L \cap U = \phi$. Namely, all paths from j to vertices in L must pass through, or end in, a vertex in S. It follows that in $\tilde{G}$ there are no k vertex-disjoint paths between a and j. Q.E.D.

We are now ready for the algorithm for determining whether the connectivity of G is at least k.

<u>Algorithm 1</u>:

(1) For every i and j such that $1 \leq i < j \leq k$ check whether there are k vertex-disjoint paths between them. If for some i and j the test fails, then G's connectivity is less than k.

(2) For every j, $k+1 \leq j \leq n$, form $\tilde{G}$ and check whether there are k vertex-disjoint paths between a and j. If for some j the test fails, then G's connectivity is less than k.

(3) The connectivity of G is at least k.

The proof of validity of this algorithm is as follows: If G's connectivity is at least k then, by Lemma 1, Step (2) will detect no failure and the algorithm will halt with the correct answer. If G's connectivity is less than k then by Lemma 2, failure will occur and again the algorithm will halt with the correct answer.

Step (1) of the algorithm requires at most $O(k^3n^2)$ elementary steps and Step (2) requires at most $O(kn^3)$. Clearly, for $k < \sqrt{n}$ the whole algorithm requires at most $O(kn^3)$ (or $O(kne)$ if $O(e) < O(n^2)$) . In fact one could use the method of Gomory and Hu to perform Step (1) and that takes at most $O(kn^3)$ for any k .

III. THE ALGORITHM FOR DIRECTED GRAPHS

Let G be a directed graph whose set of vertices is $\{1,2,...,n\}$ and with e edges. An (i,j)-separating set, S, is a set of vertices such that every directed path from i to j passes through at least one vertex in S . The connectivity of G is defined as follows:

(1) If the graph is completely connected (namely, $e = n(n-1)$) then $c = n-1$.

(2) If the graph is not completely connected then c is the least cardinality of a separating set.

Menger's theorem holds in this case too, and the network flow technique applies. The straightforward technique of checking if there are k vertex-disjoint directed paths between every ordered pair of vertices, takes at most $O(kn^4)$ steps.

At first it seems like the case of directed graphs is harder. The Gomory and Hu technique has not been generalized to directed graphs and Kleitman did not discuss directed graphs at all.*

* I believe that his method can be applied to directed graphs too, with minor necessary changes.

However, this is not the case.

Let $\vec{G}$ be an augmented graph constructed for j as follows: Add a new vertex a to the graph and connect a by an edge to each of the vertices in $L = \{1,2,\ldots,j-1\}$. Similarly $\overleftarrow{G}$ is constructed by adding a new vertex a and edges from each of the vertices in L to a . Assume now that $j > k$.

<u>Lemma 3</u>: If in G each $i \in L$ can be connected to j via k vertex-disjoint directed paths then in $\vec{G}$ there are k vertex-disjoint directed paths from a to j .

<u>Proof</u>: Assume not. Then there is a separating set S , $|S| < k$, such that all paths from a to j pass through at least one vertex of S . Consider the set of vertices, J , such that, like for j , all paths from a to them pass through at least one vertex of S . None of the vertices in L can be in J , since a is connected to each of them by an edge. Thus, there exists a vertex v in L which is not in J and not in S . Every path from v to j must pass through at least one vertex of S . Thus, there cannot be k vertex-disjoint directed paths from v to j . A contradiction. Q.E.D.

Symmetrically, the following Lemma can be proved.

<u>Lemma 4</u>: If in G , j can be connected to each $i \in L$ via k vertex-disjoint directed paths, then in $\overleftarrow{G}$, there are k vertex-disjoint directed paths from j to a .

Let j be the least vertex such that for some $i < j$ either there are no k vertex-disjoint directed paths from i to j or there are no k vertex-disjoint directed paths from j to i.

Lemma 5: Assume j is as defined above and that there are no k vertex-disjoint directed paths from i to j (from j to i). There are no k vertex-disjoint paths from a to j in $\vec{G}$ (from j to a in $\overleftarrow{G}$).

Proof: Consider a minimum separating set, S, such that all paths from i to j pass through at least one vertex of S. It follows that $|S| < k$. Let U be the set of all vertices such that all directed paths from i to them pass through at least one vertex of S. Clearly $j \in U$. If $j' < j$ is in U then there are no k vertex-disjoint paths from i to j', and j-th choice was erroneous. Thus, j is the least vertex in U, or $L \cap U = \phi$. Namely, all paths from vertices in L to j must pass through, or start in, a vertex in S. It follows that in G there are no k vertex-disjoint directed paths from a to j. Q.E.D.

Algorithm 2:

(1) For every i and j such that $1 \leq i < j \leq k$ check whether there are k vertex-disjoint directed paths from i to j and also if there are k such paths from j to i. If one of these tests fails then G's connectivity is less than k.

(2) For every j, $k+1 \leq j \leq n$, form $\vec{G}$ and check whether there are k vertex-disjoint directed paths from a to j; also

form $\overleftarrow{G}$ and check whether there are k such paths from j to a. If for some j one of these tests fails then G's connectivity is less than k.

(3) The connectivity of G is at least k.

The proof of validity is similar to that of Algorithm 1. Again, Step (1) takes at most $O(k^3n^2)$ and Step (2), $O(kn^3)$. If $k < \sqrt{n}$ then the whole algorithm takes at most $O(kn^3)$ steps.

REFERENCES

[1] Menger, K., "Zur allgemeinen Kurventheorie," Fund. Math. Vol. 10, 1927, pp. 96-115.

[2] Dantzig, G.B., and Fulkerson, D.R., "On the max-flow min-cut theorem of networks, Linear Inequalities and Related Systems, Annals of Math. Study, 38, Princeton Univ. Press, 1956, pp. 215-221.

[3] Ford, L.R., and Fulkerson, D.R., "Flows in Networks," Princeton Univ. Press, 1962.

[4] Even, S., "Algorithmic Combinatorics," Macmillan, 1973, page 226.

[5] Gomory, R.E., and Hu, T.C., "Multi-terminal network flows," J. SIAM, Vol. 9, No. 4, 1961, pp. 551-570.

[6] Kleitman, D.J., "Methods for investigating connectivity of large graphs," IEEE Trans. on Circuit Th., May 1969, Vol. CT-16, pp. 232-233.

[7] Hopcroft, J., and Tarjan, R., "Algorithm 447: Efficient algorithms for graph manipulation," Comm. of the ACM, June 1973, Vol. 16, No. 6, pp. 372-378.

[8] Hopcroft, J.E., and Tarjan, R.E., "Finding the triconnected components of a graph", Tech. Rep. of the Dept. of Comp. Sci., Cornell University, Aug. 1072.

[9] Lucas, E., Récreations Mathématiques, Paris 1882.