SIAM J. COMPUT.
Vol. 16, No. 2, April 1987

A POLYNOMIAL-TIME ALGORITHM TO FIND THE SHORTEST CYCLE BASIS OF A GRAPH*

J. D. HORTON†

Abstract. Define the length of a basis of the cycle space of a graph to be the sum of the lengths of all cycles in the basis. An algorithm is given that finds a cycle basis with the shortest possible length in $O(m^3n)$ operations, where m is the number of edges and n is the number of vertices. This is the first known polynomial-time algorithm for this problem. Edges may be weighted or unweighted. Also, the shortest cycle basis is shown to have at most $3(n-1)(n-2)/2$ edges for the unweighted case. An $O(mn^2)$ algorithm to obtain a suboptimal cycle basis of length $O(n^2)$ for unweighted graphs is also given.

Key words. graph theory, cycle basis, minimization, polynomial-time algorithm, fundamental set of circuits

AMS(MOS) subject classifications. Primary 68Q25; secondary 05C38, 68R10, 05C35

1. Introduction. A cycle basis is used to examine the cyclic structure of a graph. For example, many algorithms use cycle bases to list all simple cycles in a graph (Mateti and Deo [11], Sysło [16]) or look for the longest cycle (Dixon and Goodman [6]). Cycle bases have been also used to solve electrical networks since the time of Kirchoff (see, for example, Chua and Chen [1]). Similar techniques have been used in the analysis of algorithms (Knuth [9]). Recently surveyors have also suggested using cycle bases to look for blunders in survey data (Steeves [13], Cribb et al. [2]). Some other uses are mentioned by Deo et al. [4].

In many of the above algorithms, the amount of work is dependent on the cycle basis chosen. A basis with shorter cycles in it may speed up the algorithm. Thus one would like to be able to calculate the shortest cycle basis possible, preferably quickly. The first references to this problem that I have found are Stepanec [14] and Zykov [17], which are in Russian. Hubicka and Sysło [8] showed that the Stepanec-Zykov algorithm does not work on all graphs, and proposed other algorithms. However, Kolasinska [10] gave an example on which the Hubicka-Sysło algorithm fails to obtain the shortest cycle basis. Both Cribb et al. [2] and Steeves [13] also asked the question of finding the shortest cycle basis. Steeves proposed a different algorithm, but examples have been found on which it did not find the shortest cycle basis either. Some other papers on the topic are Deo [3] and Sysło [15].

A second related problem first proposed in Hubicka and Sysło [8] has fared even worse. How can you find a spanning tree whose fundamental set of cycles has total shortest length? Deo, Prabhu and Krishnamoorthy [4] have shown that this question is *NP*-complete.

2. Definitions. In this paper, graphs are finite, undirected, without loops or multiple edges. Since the cycle space of a graph is the direct sum of the cycle spaces of its 2-connected components, we are concerned only with 2-connected graphs. An edge is denoted by the (unordered) pair of its endpoints. Each edge e of a graph is weighted with a real number $w(e)$, which extends to a weight function on all sets of edges. The unweighted case, or equivalently the case in which all weights are one, does not seem to lead to any simplification of the algorithm. Weights are not allowed to be negative.

* Received by the editors April 18, 1985; accepted for publication (in revised form) July 2, 1986. This research was supported by Natural Sciences and Engineering Research Council of Canada grant A5376.

† School of Computer Science, University of New Brunswick, Fredericton, New Brunswick, Canada E3B 5A3.

A *cycle* is any set of edges of a graph in which each vertex of the graph is incident with an even number of edges. Two cycles, C and D, can be added to form their sum, $C+D$, using the set symmetric difference operation, $(C \cup D)-(C \cap D)$. Equivalently, one can identify a cycle with its incidence vector, and cycle-addition as vector addition over the integers modulo 2. The set of all cycles is closed under addition, forming the *cycle space*, and can be generated by the set of simple cycles. Subgraphs are identified with their edge-sets so that a simple circuit is also a cycle. The *weight* or length of a cycle basis is the sum of the weights of its cycles.

In much of the following we will be dealing with an arbitrary graph. Let the number of vertices in the graph be n; let the number of edges in the graph be m; let ν be the dimension of the cycle space. Then $\nu = m-n+1$, since we are assuming the graph to be 2-connected.

3. Theoretical results. A cycle in a shortest cycle basis of a graph must have certain properties, as will be shown in this section. Only simple cycles that have these properties need to be considered by the algorithm. We start by citing some known results.

THEOREM 1. *If* $\mathbf{B}$ *is a cycle basis for a graph,* C *is a cycle in* $\mathbf{B}$, *and* $C = C_1 + C_2$, *then either* $\mathbf{B}-\{C\}\cup\{C_1\}$ *is a cycle basis or* $\mathbf{B}-\{C\}\cup\{C_2\}$ *is a cycle basis.*

Another result that is well known, and was used by Hubicka and Sysło [8], is that the shortest cycle through an edge is always in the cycle basis. Although I do not use this fact, I include it for completeness.

THEOREM 2. *If e is an edge, and C is a shortest cycle through e, then C is in some minimum cycle basis. Moreover, any minimum cycle basis must contain some shortest cycle through e.*

The major observation that I make is that in any cycle in a minimum cycle basis, any two vertices are connected by a shortest path.

THEOREM 3. *Let x and y be two vertices of G, and let* $\mathbf{B}$ *be a cycle basis of G. Let P be a path from x to y. Then any cycle C of* $\mathbf{B}$ *that contains x and y can be exchanged for either a cycle that includes the path P, or a cycle that excludes one of x and y.*

Proof. Let P_1 and P_2 be the two paths in C joining x and y. Assume $P \neq P_1$ and $P \neq P_2$, for otherwise C does not need to be exchanged. Define $C_1 = P + P_1$ and $C_2 = P + P_2$. Then C_1 and C_2 are cycles, and $C = C_1 + C_2$. By Theorem 1, C can be exchanged for either C_1 or C_2. If the exchanged C_i is not a simple cycle, then C_i can be exchanged for one of its simple cycles, and that simple cycle cannot include both x and y.

COROLLARY. *If* $\mathbf{B}$ *is a minimum cycle basis, and P is the unique shortest path from x to y, then every cycle of* $\mathbf{B}$ *that contains x and y must contain the path P.*

Proof. In the proof of Theorem 3, $w(C_i) \leqq w(P) + w(P_i) < w(P_1) + w(P_2) = w(C)$. Hence if C does not contain P, it can be exchanged for a shorter cycle, and so $\mathbf{B}$ would not be minimum.

Note that the corollary assumed that the shortest path from x to y was unique. In general, there can be several paths between any two vertices that are shortest. What Theorem 3 and its corollary imply is that in any minimum cycle basis, any two vertices of a cycle must be joined by a shortest path. Moreover, if there is a preferred shortest path from x to y, one can exchange all cycles in $\mathbf{B}$ for cycles that contain this preferred shortest path, and still retain a minimum cycle basis.

Define $d(x, y)$ to be the length of a shortest path from x to y.

THEOREM 4. *Let x be any vertex of any cycle C in a minimum cycle basis. Then there is an edge $\{y, z\}$ in C such that C consists of a shortest path from x to y, a shortest path from x to z, and the edge $\{y, z\}$.*

Proof. Let $C=(x_0, x_1, \cdots, x_{n-1}, x_n)$, where $x=x_0=x_n$. Define $P_i=(x, x_1, x_2, \cdots, x_i)$ and $Q_i=(x, x_{n-1}, \cdots, x_i)$. By the corollary to Theorem 3, $d(x, x_i)=w(P_i)$ or $w(Q_i)$. Let $y=x_i$, where i is the largest subscript such that $d(x, x_i)=w(P_i)$. Let $z=x_{i+1}$. Then $C=P_i+\{x_i, x_{i+1}\}+Q_{i+1}$.

Theorem 4 gives a strong condition on the cycles in a minimum cycle basis, especially if all the shortest paths are unique. In the latter case all basis cycles can be found by considering for each vertex-edge pair $(x, \{y, z\})$ the cycle $C(x, y, z)=P(x, y)+P(x, z)+\{y, z\}$, where $P(v, u)$ is the shortest path from v to u. Obviously there are at most nm such cycles. The set of all cycles forms a matroid, so that the greedy algorithm can be used to extract the minimum cycle basis.

Unfortunately, this last greedy step is necessary. A counterexample to the converse of Theorem 4 is given in Fig. 1. The shortest cycle basis consists of the inner faces. However, the outer face also satisfies the condition of Theorem 4. Thus one cannot hope to eliminate the check for independence of the cycles unless another condition is found that cycles in a minimum cycle basis must satisfy.

4. The algorithm.

The outline of the algorithm can be given as follows:

(1) Find a minimum path $P(x, y)$ between each pair of vertices x, y.

(2) For each vertex v and edge $\{x, y\}$ in the graph, create the cycle $C(v, x, y)=P(v, x)+P(v, y)+\{x, y\}$, and calculate its length. Degenerate cases in which $P(v, x)$ and $P(v, y)$ have vertices other than v in common can be omitted.

(3) Order the cycles by weight.

(4) Use the greedy algorithm to find the minimum cycle basis from this set of cycles.

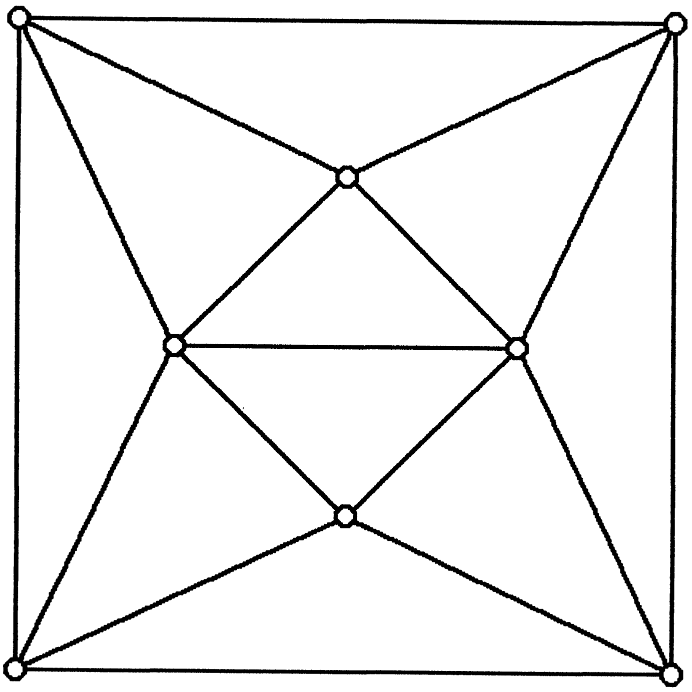

FIG. 1. *A counterexample to the converse of Theorem* 4.

That this algorithm works is not entirely obvious. It would be obvious if the shortest paths were all unique, or even if all subpaths of any chosen shortest path $P(x, y)$ were also a chosen subpath. The former could be guaranteed by perturbation or lexicographic techniques. The latter could be done by post-processing the output from the shortest path algorithm. However, these complications are unnecessary if all edges have positive weight.

THEOREM 5. *If all edges have nonnegative weight, the algorithm finds a minimum cycle basis no matter what shortest path $P(x, y)$ is chosen for each pair of vertices x and y.*

Proof. Let **B** be a minimum cycle basis. We shall change **B** by exchanging cycles in **B** with cycles containing the chosen shortest paths. First order the vertices $x_1, x_2, \cdots, x_n$. For each vertex x_i, order the $n-1$ paths $P_{ij} = P(x_i, y_{ij})$ in nondecreasing order of length. Now process all $n(n-1)$ paths in the lexicographic order of the subscripts, $P_{11}, P_{12}, P_{13}, \cdots, P_{21}, \cdots, P_{n\,n-1}$. For each P_{ij}, exchange each cycle in the basis that contains vertices x_i and y_{ij} with a cycle that contains the path P_{ij}. Note that only simple cycles can be formed, since otherwise we would obtain a basis of shorter length. By Theorem 3 and its corollary, a minimum cycle basis is obtained. We must show that the resultant basis **B**$'$ obtained after processing all $n(n-1)$ paths is a subset of the cycles obtained from step (2) of the algorithm.

Let C be any cycle in **B**$'$. Let x_i be the vertex of C with the largest subscript. Then C must have been in the basis at the end of the processing of $P_{i1}, P_{i2}, \cdots, P_{i\,n-1}$. By Theorem 4 there is an edge $\{y, z\}$ in C such that C consists of a shortest path from x_i to y, say P_1, a shortest path from x_i to z, say P_2, and the edge $\{y, z\}$. The last path P_{ij} that may have affected C must have been either $P(x_i, y)$ or $P(x_i, z)$, because all other vertices in C are closer to x. Assume that $P(x_i, y)$ was the last. Then either $P(x_i, y) = P_1$ or $P(x_1, y) = P_2 + \{y, z\}$.

First assume $P_1 = P(x_i, y)$. The only other paths that could have been processed after $P(x_i, z)$ was processed, and that changed the cycle C, are paths $P(x_i, w)$ where w is on a shortest path from y to x_i. But this could not have affected P_2, the path from z to x_i, since if it had, a nonsimple cycle would have been formed. Hence $P_2 = P(x_i, z)$. Thus C would have been found by the algorithm.

The alternative is that z is on $P(x_i, y)$. Thus C consists of two shortest paths from y to x_i. Let w be the other vertex neighboring y in the cycle C. Then the path from w to x_i is a shortest path, because it is a subpath of a shortest path. Thus w can replace z throughout the whole argument, and this second case reduces to the first case.

5. Analysis and implementation ideas.

A quick analysis indicates that (4) is the critical step:

Step (1) takes $O(n^3)$ operations,
Step (2) takes $O(mn^2)$ operations,
Step (3) takes $O(a \log(a))$ operations,
Step (4) takes $O(nk)$ operations,

where a is the number of cycles found in Step (2), and k is the number of operations to decide whether a cycle is independent of another given set of independent cycles or not.

Step (1) can use your favorite shortest path algorithm, e.g., Floyd [7] or Dijkstra [5]. Step (2) is straightforward, but can be improved. The first factor m counts the number of edges, the factor n counts the number of vertices, and a second n is a bound on the time to write out the cycle. However, the cycles do not have to be calculated immediately but only when needed. It is necessary to calculate the length of the cycle, which is the sum of the lengths of two shortest paths and an edge if the case is not degenerate. A degenerate case will never be needed in the processing later anyway. Thus Step (2) can be reduced to $O(nm)$ operations.

The bound of nm on the number of cycles can also be improved. If Dijkstra's shortest path algorithm is used [5], which requires each edge to have nonnegative length, then a shortest-path spanning tree is grown at each vertex. A vertex and an edge in its shortest-path spanning tree will never create a cycle. Hence $a \leqq n\nu$. This improvement is important only if the graph is very sparse. A larger practical gain may be made if the chosen shortest paths are such that all subpaths of a chosen path are also chosen. This will automatically occur if all shortest paths are unique. In this case, each cycle C will be generated once for each vertex, and hence the number of cycles is cut by at least a factor of three. Of course, a method of detecting duplicate circuits is then required, which at first glance seems to require calculating the circuits. One method to avoid calculating all the cycles is to first order the vertices, then remove any shortest path $P(x, y)$ that passes through a vertex z that precedes x in the ordering. Again, for this operation to allow the algorithm to work, it is sufficient that the set of chosen shortest paths satisfy the subpath property that all subpaths of chosen paths are also chosen paths. Each cycle will now be generated only once. The largest number of cycles that can be generated by the algorithm using this technique is $\sum_{k=2}^{n} k(k-1)/2-k+1 = n(n-1)(n-2)/6$. This bound is realized by K_n, the complete graph on n vertices with unweighted edges. The algorithm must generate all triangles, and $n(n-1)(n-2)/6$ is precisely the number of triangles in K_n.

Another method to reduce the number of cycles is to test whether the cycles produced by Step (2) satisfy Theorem 4, which can be done in $O(k)$ operations, where k is the number of vertices in the cycle.

Step (3) is a noncritical step. Conceivably it would be made to run faster by taking into account the order in which the cycles are produced. However, its bound of $O(n\nu \log n)$ is certainly insignificant in comparison to Step (2) or (4).

Step (4) is critical. One method to perform this step is to consider the cycles as the rows of a 0-1 matrix. The columns correspond to the edges of the graph; the rows are the incidence vectors of the cycles. Gaussian elimination using elementary row operations over the integers modulo 2 can be applied to the matrix. Each row should be processed in turn, in the order of the weights of the cycles. When enough independent cycles have been found, the process can stop. Each row can be processed in $O(mr)$ operations, where m is the number of columns, and r is the number of rows above the row being processed. Since r is bounded by ν, since only a complete set of independent cycles are needed to be kept, and since the total number of cycles is $O(n\nu)$, Step (4) takes $O(m\nu^2 n)$ operations.

As the bound for Step (4) dominates the functions for the other steps, the algorithm as a whole has worst case upper bound of $O(m\nu^2 n) = O(m^3 n) = O(n^7)$ operations.

6. The length of the shortest cycle basis.

6. The length of the shortest cycle basis. Deo et al. [4] raised another question concerning fundamental sets of circuits in an unweighted graph. What is the maximum possible total length (weight) of the shortest (lightest) fundamental set of circuits of a graph? They conjectured that the total length is at most $3n^2/2$. I am able to show that they were correct at least for minimum cycle bases.

THEOREM 6. *The length of a minimum cycle basis of an unweighted graph with n vertices is at most* $3(n-1)(n-2)/2$.

Proof. The theorem is certainly true for $n = 1$, 2, or 3. Assume $n \geqq 3$. Let G be a graph with $n+1$ vertices. If G is not two-connected, then let v be a cut vertex of G.

Let H be a component of $G-v$, and let $G-H$ be the rest the graph, including the vertex v. Let H have k vertices; then $G-H$ has $n-k+1$ vertices. By an induction hypothesis, the subgraph of G induced by H and v combined has a cycle of length less than or equal to $3k(k-1)/2$, while $G-H$ has a cycle basis of length $3(n-k)\times(n-k-1)/2$ or less. Since the union of these two cycle bases forms a cycle basis for G, and $k(k-1)+(n-k)(n-k-1)\leqq n(n-1)$, the result is true for G. Therefore, we may assume G is two-connected. Let x be a vertex of minimal degree in G. As an induction hypothesis, assume that $G-x$ satisfies the theorem. Thus, if **B** is a minimum cycle basis of $G-x$, $w(G)\leqq 3(n-1)(n-2)/2$. I intend to add simple cycles that pass through x to **B** until a cycle basis for G is found. The total length of the cycles will be less than or equal to $3n-3$. This will complete the proof to the theorem, since $\sum_{k=3}^{n} 3k-6=3(n-1)(n-2)/2$. Let the degree of x be k. Then $k-1$ cycles have to be added to B to form a cycle basis for G and all these cycles must pass through x.

What other constraints are there on the choice of these cycles? Consider another graph which I shall call the x-network, in which vertices are called *nodes* and edges are called *branches*, to distinguish the original graph from the x-network. The edges adjacent to x are the nodes of the x-network, and two nodes are connected by branches if and only if a chosen cycle uses both edges. The set of chosen cycles together with **B** forms an independent set if and only if the network does not contain a cycle. As the x-network has k nodes and $k-1$ branches, the x-network must be a tree. Conversely, if the x-network is a tree then the set of chosen cycles together with **B** forms a cycle basis for G.

Call the other end points of the edges incident with x: $x_1, x_2, \cdots, x_k$. A chosen cycle will consist of a path in $G-x$, from some x_i to some other x_j, and two edges adjacent to x. I choose these $k-1$ paths from a spanning tree T of $G-x$.

Consider the special case when $k=n$. Then it is easy to pair off the x_i's, which are all the vertices of the tree. Pair any vertex of degree one in the spanning tree with its only neighbor, and remove the vertex of degree one from the tree. Continue in this way until all vertices are paired. All paths are of length 1. Then the $n-1$ cycles are all triangles, so the total weight to be added to **B** is $3n-3$.

The next case to be considered is for $k<n/2+1$. The paths joining the x_i in the tree T can be chosen so that no edge of T is in more than two paths. Consider, for example, an embedding of T in the plane, and also consider the closed path consisting of the face of T. Each edge occurs twice. Take the subpaths of this path that go from one x_i to the next x_j. In the corresponding network, these paths correspond to branches. The network is of course connected. In fact it contains at least k branches; and hence at least one cycle. A spanning tree in this network corresponds to a required set of paths in T. At least one of the paths have to be removed, and the total length of the chosen paths is at most $2(n-1)-1=2n-3$. Then the total length of the circuits to be added to **B** is at most $2n-3+2(k-1)<3n-3$.

The last case to be considered is for k such that $v/2+1\leqq k<v$. We use induction on k, the minimum degree of G. Since x, a vertex, is of minimum degree, the degree of every vertex in $G-x$ is at least $n/2$. Since there are $k\geqq n/2+1$ vertices adjacent to x, any vertex x_i adjacent to x is also adjacent to some other vertex x_j adjacent to x. Remove x_i from G. By an induction hypothesis, there are $k-2$ independent cycles in $G-x_i$ that pass through x, of total length at most $3n-6$. Adding the triangle (x, x_i, x_j) to this set satisfies the condition of the theorem.

Deo et al. [4] showed that the bound of Theorem 6 cannot be improved, by noting that for the complete graph on n vertices, the star tree has a set of fundamental circuits of length $3(n-1)(n-2)/2$.

The ideas used in the proof to Theorem 6 have some interesting extra implications, which are important enough to include as another theorem. Define a set of simple cycles through a vertex x to be *locally-independent-at-x* if the corresponding network, as developed in the proof to Theorem 6, does not contain a cycle. Thus, locally-independent-at-x implies independent, but not conversely. A maximal locally-independent-at-x set of cycles consists of degree $(x)-1$ cycles in a two-connected graph. In the corresponding x-network, the set corresponds with a spanning tree. Each cycle in a minimum-weight maximal locally-independent-at-x set consists of two edges $\{x, y\}$, and $\{x, z\}$, and the shortest path from y to z in $G-x$. If we label all the branches in the x-network with the length of the corresponding cycle, then a minimum-weight maximal locally-independent-at-x set of cycles corresponds to, and has the weight of, a minimum spanning tree in the network.

THEOREM 7. *Given a vertex x in a graph G, a minimum-weight maximal locally-independent-at-x set of circuits is a subset of a minimum cycle basis.*

Proof. Let B be a basis, and let C be a cycle, not in B, that is in the minimum-weight maximal locally-independent-at-x set of cycles. Let C correspond to the branch a in the network. C is the sum of cycles in a subset of B, say **A**. The cycles in **A** that contain x, together with C, sum to zero on the edges at x, and so must correspond to a cycle in the x-network. Hence there must be a set of cycles in **A** that, together with C, correspond to a cycle through a in the network. That cycle must contain a branch b that is not in the minimum spanning tree of the x-network. The branch b corresponds to a cycle D in the original graph. Then $B-\{D\}\cup\{C\}$ is also a cycle basis. But $w(C)=w(a)\leqq w(b)\leqq w(D)$, because a is in the minimum spanning tree, and b is not.

7. A fast algorithm for a suboptimal cycle basis. The proof to Theorem 6 also leads to an algorithm faster than the exact algorithm, that is guaranteed to give a cycle basis with no more than $3(n-1)(n-2)/2$ edges in the unweighted case.

(1) Choose a vertex x of minimum degree.
(2) Find all shortest paths in $G-x$ between neighbors of x.
(3) Find a minimal spanning tree in the corresponding weighted network.
(4) Form the shortest path cycles corresponding to the branches in the tree.
(5) Add to the set of cycles from Step 4 a cycle basis for $G-x$ (obtained by recursion).

A quick analysis of this algorithm is: Step (1), $O(1)$; Step (2), $O(dn^2)$, where $d=$ degree of x; Step (3), $O(n^2)$; Step (4), $O(dn)$; Step (5), multiplies by $O(n)$. The dominating term comes from Step (2), and is $O(dn^2)$. Given Step (5), the whole algorithm is $O(n^2$ (sum of degrees)$)=O(n^2m)$.

To find a minimum cycle basis, it is not sufficient to calculate the minimum-weight maximal locally-independent-at-x set of circuits at all vertices. Figure 2 gives a counter-

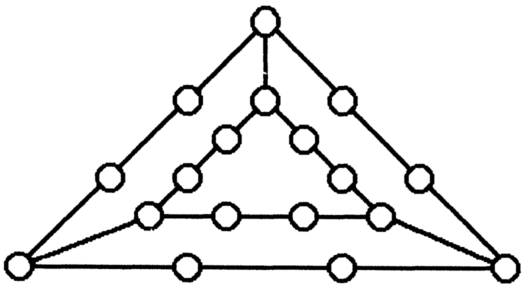

FIG. 2. *Shortest cycle basis with a circuit that is not locally independent.*

example. The three trapezoidal faces include all minimum-weight maximal locally-independent-at-x sets of circuits at any vertex x. But one of the triangular faces must be added to form a minimum-weight cycle basis.

Deo et al. [4] also suggest several heuristic algorithms to search for a spanning tree with a minimum fundamental set. In all cases, the algorithms may generate a basis of weight $O(n^3)$, as can be shown by graphs based on the graph used by Paton [12] for a similar purpose.

8. Conclusions.

The algorithm that is outlined in §§ 4 and 5 is the first polynomial-time algorithm that actually generates the shortest possible cycle basis. However, it is a very expensive algorithm, $O(m^3n)$. Now a cycle basis is generally not wanted for its own sake, but to be used as input for a later algorithm. The importance of a short cycle basis is to reduce the amount of work that has to be done in this later algorithm. Other published algorithms, although much faster than my algorithm, may not produce a cycle basis that is even close in length. Much of the published work has concentrated on trying to generate a tree with a short fundamental set of circuits. These algorithms, although fast, are not guaranteed to arrive at a good answer. Indeed, the exact solution to the shortest fundamental set of circuits has been shown to be *NP*-complete. If the follow-up algorithm is an expensive one, such as listing all simple cycles, an exact algorithm may provide a much better starting point for it than other algorithms.

There is considerable room for improvement on this problem. It is quite likely that some more constraints on cycles in the shortest cycle basis can be found. Nor does the algorithm use the fact that certain cycles must be in the basis, as pointed out in Theorems 2 and 7, for example. Such a marriage between this algorithm and the earlier ones could make for a fast algorithm by eliminating checks for independence. Even better would be to find an exact set of conditions that a member of the shortest cycle basis must satisfy.

REFERENCES

[1] L. O. CHUA AND L. K. CHEN, *On optimally sparce cycle and coboundary basis for a linear graph*, IEEE Trans. Circuit Theory, CT-20 (1973), pp. 495-503.

[2] D. W. CRIBB, R. D. RINGEISEN AND D. R. SHIER, *On cycle bases of a graph*, Congr. Numer., 32 (1981), pp. 221-229.

[3] N. DEO, *Minimum length fundamental cycle set*, IEEE Trans. Circuits and Systems, 26 (1979), pp. 894-895.

[4] N. DEO, G. M. PRABHU AND M. S. KRISHNAMOORTHY, *Algorithms for generating fundamental cycles in a graph*, ACM Trans. Math. Software, 8 (1982), pp. 26-42.

[5] E. W. DIJKSTRA, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269-271.

[6] E. T. DIXON AND S. E. GOODMAN, *An algorithm for the longest cycle problem*, Networks, 6 (1976), pp. 139-149.

[7] R. W. FLOYD, *Algorithm* 97: *shortest path*, Comm. ACM, 5 (1962), p. 345.

[8] E. HUBICKA AND M. M. SYSŁO, *Minimal bases of cycles of a graph*, in Recent Advances in Graph Theory, Proceedings of the symposium held in Prague, June 1974, M. Fiedler, ed., Academia Praha, Prague, 1975, pp. 283-293.

[9] D. E. KNUTH, *The Art of Computer Programming*, Vol. 1, Addison-Wesley, Reading, MA, 1968, pp. 363-370.

[10] E. KOLASINSKA, *On a minimum cycle basis of a graph*, Zastos. Mat., 16 (1980), pp. 631-639.

[11] P. MATETI AND N. DEO, *On algorithms for enumerating all circuits of a graph*, this Journal, 5 (1976), pp. 90-99.

[12] K. PATON, *An algorithm for finding a fundamental set of cycles of a graph*, Comm. ACM, 12 (1969), pp. 514-518.

[13] P. STEEVES, *Numerical processing of horizontal control data-economization by automation*, Ph.D. thesis, University of New Brunswick, 1984.

[14] G. F. STEPANEC, *Basis systems of vector cycles with extremal properties in graphs*, Uspekhi Mat. Nauk, 19 (1964), pp. 171-175. (In Russian.)

[15] M. M. SYSŁO, *On cycle bases of a graph*, Networks, 9 (1979), pp. 123-132.

[16] ———, *An efficient cycle vector space algorithm for listing all cycles of a planar graph*, this Journal, 10 (1981), pp. 797-808.

[17] A. A. ZYKOV, *Theory of Finite Graphs*, Nauka, Novosibirsk, 1969. (In Russian.)