Algorithms for Generating Fundamental Cycles in a Graph

NARSINGH DEO and G. M. PRABHU
Washington State University
and
M. S. KRISHNAMOORTHY
Rensselaer Polytechnic Institute

The following problem is considered: Given an undirected, connected graph G, find a spanning tree in G such that the sum of the lengths of the fundamental cycles (with respect to this tree) is minimum. This problem, besides being interesting in its own right, is useful in a variety of situations It is shown that this problem is NP-complete. A number of polynomial-time, heuristic algorithms which yield "good" suboptimal solutions are presented and their performances are discussed. Finally, it is shown that for regular graphs of order n the expected value of the total length of a minimum fundamental-cycle set does not exceed $O(n^2)$.

Categories and Subject Descriptors: F.2.2 **[Analysis of Algorithms and Problem Complexity]**: Nonnumerical Algorithms and Problems—*computations on discrete structures*; G 2.2 **[Discrete Mathematics]**: Graph Theory

General Terms: Algorithms, Design

Additional Keywords and Phrases: fundamental-cycle set, spanning tree, NP-complete

1. INTRODUCTION

In graph analysis, it is often desirable to examine the cyclic structure of the given graph. The most commonly used method is to generate a set of fundamental cycles. A fundamental-cycle set is used by an organic chemist interested in the coding of ring compounds [16, 20]. A fundamental-cycle set can also be used in determining the isomorphism of graphs [2] and in the frequency analysis of computer programs [14]. In the literature, a number of algorithms have been proposed and implemented for generating a set of fundamental cycles [3, 7, 8, 11, 13, 15, 17, 18, 21, 22, 25].

If the data structure is chosen carefully, the computational time complexity of the best of these algorithms turns out to be

$$O\left(\sum_{i=1}^{\mu} l_i\right)$$

This work was supported in part by National Science Foundation Grant MCS-78-25851.
Authors' addresses: N. Deo and G. M. Prabhu, Computer Science Department, Washington State University, Pullman, WA 99164, M. S Krishnamoorthy, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12181.

[17, 18], where l_i is the length of the ith fundamental cycle (in the generated set) and μ is the nullity of the given graph. (The nullity of a connected graph G with n vertices and e edges is $\mu = e - n + 1$.) A set of fundamental cycles of a graph with respect to a spanning tree is a set of those μ cycles that contain exactly one nontree edge each. The total length of the fundamental-cycle set with respect to a spanning tree T in G, denoted as

$$L(T) = \left(\sum_{i=1}^{\mu} l_i \right),$$

is, in general, dependent on T. For example, in a complete graph K_n of n vertices, the total length of fundamental cycles with respect to a Hamiltonian tree T_h is

$$L(T_h) = \sum_{i=1}^{n-2} i(n - i + 1) = \frac{1}{6} n^3 + O(n^2),$$

because there is 1 cycle of length n, 2 cycles of length $(n - 1)$, 3 cycles of length $(n - 2), \ldots,$ and $(n - 2)$ cycles of length 3, in K_n. On the other hand, with respect to a star tree T_s, in the same graph K_n, the total length of fundamental cycles is

$$L(T_s) = 3 \cdot \mu = 3 \cdot \frac{(n-1)(n-2)}{2} \simeq \frac{3}{2} \cdot n^2,$$

because each fundamental cycle is of length 3. Since the computation time for generating a set of fundamental cycles is of the order of the total length L, which in turn is dependent on T, it is interesting and useful to explore the possibility of obtaining an optimal spanning tree T_{min} in a given graph G such that $L(T_{min}) \leq L(T_i)$ for every spanning tree T_i in G. If an algorithm to obtain such an optimal spanning tree is fast enough, it could be utilized in generating a set of fundamental cycles.

This problem was first posed by Hubicka and Syslo [10]. (A related problem of determining a minimum-length cycle basis of a graph was first posed by Stepanec [19] and discussed further by Zykov [26]. It may be noted that every basis of the cycle subspace need not correspond to a spanning tree. That is, every fundamental-cycle set forms a cycle basis, but not every cycle basis is a fundamental-cycle set.)

It has been recently conjectured [5] that generating such an optimal spanning tree T_{min} may be NP-hard. In Section 2 we prove that generating T_{min} is indeed NP-complete. In Section 3 we describe fast heuristic algorithms that generate suboptimal spanning trees (i.e., spanning trees T for which $L(T)$ may not be minimum). In Section 4 we discuss the implementation of these heuristic algorithms, and in Section 5 we compare their performances. In Section 6 we derive upper bounds on the expected length of the minimum-length fundamental-cycle set $L(T_{min})$, for certain "bad" classes of graphs. It is shown that this expected length is bounded by $O(n^2)$ for regular graphs of order n. This bound is consistent with the empirical observations made in Section 4 on a large number of randomly generated graphs. Section 7 consists of the concluding discussions and mention of further problems.

The graph terminology used here is fairly standard and can be found in most textbooks on graph theory [3, 9]. We denote an undirected graph $G = (V, E)$,

where V is the set of vertices and E, the set of edges. The cardinalities of sets V and E are denoted by n and e, respectively. For terminology related to NP-completeness we refer the reader to [1, 6, or 18].

2. GENERATING T_{min} IS NP-COMPLETE

The problem of finding a fundamental-cycle set with minimum total length may be formally stated as

Instance: Graph $G = (V, E)$, positive integer L.
Question: Is there a spanning tree T of G such that the sum over the lengths of all fundamental cycles of G with respect to T is no more than L?

We transform in polynomial time an already known NP-complete problem, namely, the shortest total-path-length spanning tree problem (STPLS), to the minimum-length fundamental-cycle-set problem. The STPLS problem may be formally stated following Garey and Johnson [6] as

Instance: Graph $G = (V, E)$, positive integer K.
Question: Is there a spanning tree T of G such that the sum, over all pairs of vertices $u, v \in V$, of the path length in T from u to v is no more than K?

The STPLS problem has been shown to be NP-complete by Johnson et al. [12]. We use this result in proving that generating T_{min} is NP-complete.

For this purpose, we introduce the following definitions.

Complete Chain. A complete chain between a pair of vertices u, v such that edge $(u, v) \notin E$ is defined as a set of vertices $\{(u, v, 1), (u, v, 2), \ldots, (u, v, n^4 - 1)\}$ and a set of edges $\{(u, (u, v, 1)), ((u, v, 1), (u, v, 2)), \ldots, ((u, v, n^4 - 1), v)\}$ (see Figure 1).

1-off Chain. A 1-off chain is obtained by deleting exactly one edge from a complete chain.

LEMMA. *The shortest total-path-length spanning tree problem is polynomially transformable to the minimum-length fundamental-cycle-set problem.*

PROOF. Let $G = (V, E)$ be a given undirected graph for which we wish to find the shortest total-path-length spanning tree. Construct another graph $H = (V_1, E_1)$ as follows:

$$V_1 = V \cup \{(u, v, i) \mid (u, v) \notin E, i = 1, 2, \ldots, n^4 - 1\}$$

$$E_1 = E \cup \{(u, (u, v, 1))\} \cup \{((u, v, i), (u, v, i + 1)) \mid i = 1, 2, \ldots, n^4 - 2\} \cup \{((u, v, n^4 - 1), v)\}$$

for pairs u, v, not in E. Let G have n vertices and e edges and let $r = [n(n - 1)/2] - e$, denote the number of vertex pairs (unordered) in G that are not joined by an edge. It is easy to see that the number of vertices in H is $n + r \cdot (n^4 - 1)$ and the number of edges in H is $e + r \cdot n^4$, since H is constructed from G by adding a complete chain between every pair of vertices u, v not joined by an edge in G.

We will now consider two different ways of constructing a spanning tree in graph H: one is to pick an arbitrary spanning tree T in G and add 1-off chains to

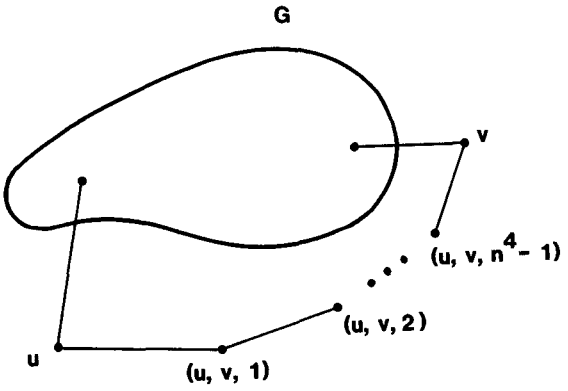

Fig 1. Complete chain added between u and v of G.

all pairs of vertices u, v of G such that $(u, v) \notin E$. Denote this spanning tree of H by T_1. The other way is to directly find a spanning tree of H, say T_2.

Consider spanning tree T_1. Considering the chords (i.e., nontree edges) of G with respect to T_1 and the one missing edge of each 1-off chain that forms fundamental cycles with respect to T_1 in the newly constructed graph H, the total length of the fundamental-cycle set in H with respect to T_1 is

$$L(T_1) = \sum_{\substack{(u,v)\in E \\ (u,v)\notin T}} (d_{uv} + 1) + \sum_{(u,v)\notin E} (n^4 + d_{uv}),$$

where d_{uv} is the distance in T between u and v.

Since the number of chords in the original graph G is $e - n + 1$ and the number of 1-off chains added to it is r, the above expression can be rewritten as

$$L(T_1) = \sum_{\substack{(u,v)\in E \\ (u,v)\notin T}} d_{uv} + (e - n + 1) + r \cdot n^4 + \sum_{(u,v)\notin E} d_{uv}. \tag{1}$$

Now, the total path length P in T (from which T_1 is derived) between all pairs of vertices of G is

$$P = 2\left[\sum_{(u,v)\notin E} d_{uv} + \sum_{\substack{(u,v)\in E \\ (u,v)\notin T}} d_{uv} + \sum_{(u,v)\in T} d_{uv}\right]. \tag{2}$$

Noting that $\sum_{(u,v)\in T} d_{uv} = n - 1$ and substituting eq. (2) in eq. (1), we get

$$L(T_1) = \frac{P}{2} + r \cdot n^4 + (e - n + 1) - (n - 1).$$

Now, in graph H let T_2 be a spanning tree that yields a set of fundamental cycles with minimum total length. Let us assume that T_2 has q complete chains. We will show presently that q must be equal to zero. Let S be the total path length in T_2 between all pairs of vertices of G. Proceeding in the same manner that was used to compute $L(T_1)$, we obtain $L(T_2)$, the length of the fundamental-cycle set with respect to T_2.

$$L(T_2) = \frac{S}{2} + r' \cdot n^4 + (e - (n - 1 - q)) - (n - 1 - q) - q \cdot n^4$$

where

$$r' = \frac{n(n-1)}{2} - e - q.$$

Since by definition $L(T_2) \leq L(T_1)$, we have

$$\frac{S}{2} + r' \cdot n^4 + (e - (n-1-q)) - (n-1-q) - q \cdot n^4 \leq \frac{P}{2} + r \cdot n^4 + (e - n + 1) - (n-1).$$

On simplifying we obtain

$$S \leq P + 4 \cdot q \cdot n^4 - 4 \cdot q. \tag{3}$$

It can be verified—by computing a lower bound for S and by observing that P is $O(n^3)$—that the inequality (3) holds only if $q = 0$. (The lower bound for S is computed by assuming that the spanning tree giving rise to the lower bound has the q complete chains connected in star fashion in H. With this assumption it follows that S is at least $2 \cdot q^2 \cdot n^4$. For the case where $q = 1$, S is in fact at least $2 \cdot n^4 + 2 \cdot (n-2)(n^4 + 1)$, and for the case where $q = 2$

$$S \geq 8 \cdot n^4 + 2 \cdot (n-3)(n^4 + 1).$$

For $q > 2$, $S \geq 2 \cdot q^2 \cdot n^4$, thereby proving our contention about inequality (3).)

Thus T_2 cannot have any complete chains, that is, $q = 0$. Hence eq. (3) yields $S \leq P$. From the construction of H it is easily seen that this transformation can be done in polynomial time. Thus the STPLS problem is polynomially transformable to the minimum-length fundamental-cycle-set problem. □

This implies that if we can find in polynomial time a spanning tree that yields a set of fundamental cycles in H with minimum total length $L(T_2)$, then we can find a shortest total-path-length spanning tree in G in polynomial time as well. Moreover, it is easy to see that the minimum-length fundamental-cycle-set problem is indeed in NP. Since the STPLS problem is known to be NP-complete [12], we have the following result.

THEOREM. *Finding a spanning tree T_{min} in a graph G that yields a fundamental-cycle set with minimum total length is NP-complete.*

3. ALGORITHMS FOR GENERATING A SET OF FUNDAMENTAL CYCLES

Since the computational time complexity of the best algorithms for generating a set of fundamental cycles of a graph is of the order of the total length L of the fundamental-cycle set [17, 18], an algorithm for a spanning tree corresponding to a minimum value for L could be used to generate a set of fundamental cycles (provided, of course, the cost of generating such a spanning tree itself is not too high). But we have just proved that generating $T_{\min}$ is an NP-complete problem. Therefore, we must look for fast heuristic algorithms for generating a suboptimal spanning tree T, for which $L(T) - L(T_{\min})$ will hopefully be small. None of the algorithms proposed in the literature has dealt with the problem of generating a spanning tree T with a reasonably small value for $L(T) - L(T_{\min})$.

As is the case with many of the polynomially bounded graph-theoretic algorithms, existing procedures for generating fundamental cycles can be classified into three groups, those utilizing (1) depth first search (DFS), (2) breadth first search (BFS), or (3) mixed search (MS) [4]. Tarjan's DFS method [23] inherently generates very long fundamental cycles. It is easy to verify that when applied to complete graph K_n the total length of the fundamental-cycle set will be $O(n^3)$ [18]. Paton's mixed search method [7, 17] does better than the DFS method on the average, but for a worst case input it would also produce total length of $O(n^3)$. An example of such a graph is given by Paton [17]. Since, in general, a BFS through a graph produces spanning trees of short diameters, the BFS method on the average generates fundamental cycles of shorter total length than either of the other two methods [5, 11]. Hence, all the heuristic algorithms considered in this paper employ a breadth first search through the given graph. These heuristic algorithms deviate from general breadth first search—the deviation lies in the criteria used to select a new vertex to explore from. Whenever the partial tree is to be extended, the new vertex to be explored from is not the "oldest" unexplored vertex as in a straightforward BFS but is selected according to some function of the degrees of the vertices. To the best of our knowledge, none of the heuristic algorithms proposed here have been implemented or tested earlier. The central idea of each of our heuristic algorithms now follows.

3.1 Static Degree Sort (SDS)

The given graph G is represented as an adjacency matrix. First the rows and columns of the adjacency matrix are reordered in descending order of the degrees of the corresponding vertices. Then the spanning tree is generated in a straightforward breadth first fashion. The first vertex explored from is vertex 1, which is the vertex with highest degree. Whenever a new vertex is considered, it is the one with the highest degree among the successors of the oldest vertex explored from (and not necessarily the vertex with the highest degree among those that are already present in the partial tree).

3.2 Dynamic Degree Selection (DDS)

The difference between DDS and SDS lies in the criteria used for selecting a new vertex to explore from. Whenever a new vertex is considered, it is a vertex of highest degree among all the vertices that are already present in the partial tree.

3.3 "Unexplored" Edges (UE)

As vertices are explored from, their degrees are thought of as getting depleted; therefore every vertex in the partial tree has a degree with respect to "unexplored" edges that is smaller than its degree in the original graph. In this heuristic, a new vertex considered is a vertex of highest degree in the partial tree, the degree being measured with respect to "unexplored" edges.

3.4 Multipoint Breadth First Search (MBFS)

Unlike the preceding three heuristic algorithms, where the new vertex is selected from vertices already in the partial tree, here we always explore from the vertex of highest degree, building a forest as we go along. We maintain the vertices

belonging to different subtrees of the forest as sets. As in any spanning tree generation algorithm, if both end vertices of a new edge belong to the same set, we treat that edge as a chord; otherwise we treat that edge as a tree edge and merge two subtrees. Edges of G are chosen in descending order of degrees of their end vertices.

4. IMPLEMENTATION OF THE ALGORITHMS

The four heuristic algorithms, namely, SDS, DDS, UE, and MBFS, were implemented in PASCAL, and a performance study was conducted on a PDP-11/60. For comparison we also implemented an algorithm (BF) which generates a spanning tree by doing a straightforward breadth first search of the graph and uses this BFS spanning tree to generate a set of fundamental cycles. Results of the performance of all five of these algorithms are summarized in Tables I-V. The data in these tables are also shown in graphical form in Figures 2 and 3. An entry in a table produced for an algorithm was generated as follows: for each specified n (number of vertices) and edge density ρ, 20 random graphs were generated. The values of L and t shown are the total length of the fundamental-cycle set, and the running time (in seconds), respectively, averaged over the 20 graphs for each case.

A plot of log L versus log n is shown in Figure 2. To avoid cluttering the plot, we have shown the plot for UE and MBFS for edge densities of 0.5 and 1.0 only. This plot suggests that L increases monotonically with edge density. The variation of log L against log n is found to be a straight line with a slope that is slightly larger than 2, and this leads us to suspect that L is $O(n^2)$. Since our algorithms are suboptimal (as discussed in Section 5), we have reason to believe that $L(T_{\min})$ is no larger than $O(n^2)$. Our suspicion is strengthened by the fact that in each case the value of L reported for a complete graph (i.e., a graph with edge density = 1.0) turns out to be exactly $\frac{3}{2}(n-1)(n-2)$, which is $O(n^2)$ and corresponds to a star spanning tree. The reason the slopes of the straight lines for log L against log n are slightly larger than 2 and not exactly 2 could be due to a multiplicative constant. It might turn out that $L = O(k \cdot n^2)$ for some k, and this factor would explain why the slopes are larger than 2.

Our empirical results seem to indicate that for randomly generated graphs UE yields the lowest value for L and BF yields the highest value for L. Hence one might want to conclude that UE will always yield a lower value for L than the other algorithms. However, an example in Section 5 illustrates that UE cannot be considered "best" (for computing the lowest value for L) for all graphs.

A plot of log t versus log n (with t expressed in milliseconds) is given in Figure 3. Here again to avoid cluttering we only consider algorithms UE and MBFS for an edge density of 0.5. In this plot also the variation of log t against log n is a straight line with a slope slightly larger than 2 (for the same reason as given above) implying that the time complexity of our algorithms is $O(n^2)$.

5. PERFORMANCE OF THE ALGORITHMS

From the empirical results obtained in Section 4 for the mean value of the fundamental-cycle-set length L, one might conclude that UE always yields a lower value for L than the other four. This is not always the case, however. There

Table I Performance of BF

Edge density ρ	Mean fundamental-cycle length L when number of vertices n = 10	20	30	40	50	Mean running time t (in seconds) when number of vertices n = 10	20	30	40	50
0.1	0	0	72.60	207.30	385.70	0.21	0.72	1.58	2.80	4.33
0.2	0	87 90	266.80	518.00	876.40	0 22	0.77	1.71	3.02	4.68
0 3	16.50	156 60	428.80	817.40	1316.55	0.23	0.83	1.82	3.19	4.95
0.4	35.20	271 80	569.40	1091.60	1729.05	0.24	0.86	1.89	3.35	5.20
0.5	48.05	283.85	704.30	1327 30	2113.45	0.25	0.90	1.98	3.53	5.42
0.6	63.05	338.65	837.05	1556.90	2482.25	0.26	0.93	2.06	3.63	5.68
0.7	75.65	394.50	940.70	1772.85	2779.65	0.26	0.97	2.11	3.75	5.86
0.8	88.15	440.20	1068.50	1925.25	3079.30	0.27	1.00	2.21	3.85	5.99
0.9	97.90	479.20	1148.55	2093.95	3333.75	0.28	1.01	2 22	3.92	6.10
1 0	108.00	513.00	1218.00	2223 00	3528.00	0.28	1.01	2.22	3.92	6.10

Table II. Performance of SDS

Edge density ρ	Mean fundamental-cycle length L when number of vertices n = 10	20	30	40	50	Mean running time t (in seconds) when number of vertices n = 10	20	30	40	50
0.1	0	0	71.65	195 35	362.75	0.20	0.64	1.43	2.53	3.93
0 2	0	83.75	247.35	499.75	830.60	0.20	0.72	1.56	2.75	4.27
0.3	16.60	147.20	397.10	766.60	1262.45	0.21	0.75	1.65	2.93	4.55
0.4	33.30	206.25	537.85	1019.35	1663.20	0.22	0 80	1.74	3.06	4.80
0.5	44.10	264.15	660.85	1251 10	2006.75	0.23	0.84	1.83	3.23	5.01
0 6	59.50	315.15	789.25	1456.90	2344.30	0.24	0.87	1.90	3.35	5.23
0.7	70.75	365.70	892.75	1656.45	2665.30	0.25	0.90	1.96	3.46	5.47
0.8	82.80	412.70	999.75	1838.45	2932.85	0.26	0.93	2.03	3.58	5.58
0.9	93.00	456.00	1093.15	2007.40	3198.45	0.26	0.94	2.08	3.68	5.71
1.0	108.00	513.00	1218.00	2223.00	3528.00	0.26	0.95	2.10	3.72	5.77

Table III. Performance of DDS

Edge density ρ	Mean fundamental-cycle length L when number of vertices n = 10	20	30	40	50	Mean running time t (in seconds) when number of vertices n = 10	20	30	40	50
0.1	0	0	59.60	172.10	319.30	0.23	0.89	2.26	4.89	9.07
0.2	0	73.40	228.00	470.85	784.65	0.22	1.02	2.83	6.08	10.98
0.3	14.05	137.95	380.40	730.25	1217.70	0.25	1.16	3.15	6.62	11.81
0.4	31.05	198.90	519.30	987.75	1613.00	0.27	1.24	3.32	6.98	12.48
0.5	42.10	255.25	640.40	1218.75	1959.85	0.28	1.30	3.47	7.24	12.94
0.6	57.60	308.35	773.40	1430.40	2314.45	0.29	1.36	3.58	7.45	13.28
0.7	69.10	360.30	881.85	1636.80	2632.95	0.30	1.41	3.70	7.67	13.58
0.8	82.55	410.80	992.25	1826.70	2916.10	0.31	1.44	3.80	7.84	13.95
0.9	93.00	456.00	1092.40	2005.15	3194.20	0.32	1.46	3.86	7.77	14.06
1.0	108.00	513.00	1218.00	2223.00	3528.00	0.33	1.47	3.86	7.95	14.08

Table IV. Performance of UE

Edge density ρ	Mean fundamental-cycle length L when number of vertices n = 10	20	30	40	50	Mean running time t (in seconds) when number of vertices n = 10	20	30	40	50
0.1	0	0	57.60	163.95	307.10	0.25	0.94	2.38	4.95	9.36
0.2	0	69.55	220.05	458.05	766.30	0.26	1.06	2.97	6.16	10.42
0.3	13.80	133.85	369.55	719.90	1198.35	0.26	1.34	3.32	6.83	12.16
0.4	29.75	194.25	506.10	968.65	1587.70	0.29	1.41	3.59	7.12	12.98
0.5	41.10	249.85	632.70	1201.90	1938.70	0.31	1.49	3.63	7.35	13.58
0.6	55.95	305.60	764.60	1417.65	2298.45	0.33	1.58	3.81	7.52	14.06
0.7	67.75	357.25	877.00	1627.10	2620.65	0.35	1.63	3.92	7.80	14.89
0.8	81.95	408.25	988.00	1819.05	2909.25	0.37	1.66	4.03	7.94	15 32
0.9	93.00	456.00	1091.65	2003.25	3190.20	0.39	1.69	4.16	8.12	15.85
1.0	108.00	513.00	1218.00	2223.00	3528.00	0.39	1.71	4.22	8.28	16.42

Table V. Performance of MBFS

Edge density ρ	Mean fundamental-cycle length L when number of vertices n =					Mean running time t (in seconds) when number of vertices n =				
	10	20	30	40	50	10	20	30	40	50
0.1	0	0	72.50	209.60	401.00	0.04	0.24	0.55	0.99	1.53
0.2	0	85.65	261.40	538.00	915.50	0.08	0.36	0.83	1.48	2.33
0.3	16.20	150.35	420.10	815.25	1308.20	0.11	0.47	1.09	1.96	3.06
0.4	33.45	211.20	548.70	1064.05	1716.80	0.14	0.59	1.33	2.42	3.81
0.5	44.40	268.30	675.30	1286.75	2090.90	0.16	0.70	1.59	2.93	4.54
0.6	59.70	318.95	804.40	1478.20	2374.45	0.19	0.80	1.84	3.30	5.20
0.7	71.10	366.65	893.30	1673.50	2676.75	0.21	0.92	2.09	3.75	5.89
0.8	82.80	412.70	1001.80	1838.45	2932.85	0.24	1.01	2.34	4.16	6.55
0.9	93.00	456.00	1093.15	2007.40	3198.45	0.27	1.11	2.55	4.57	7.20
1.0	108.00	513.00	1218.00	2223.00	3528.00	0.29	1.22	2.80	5.03	7.89

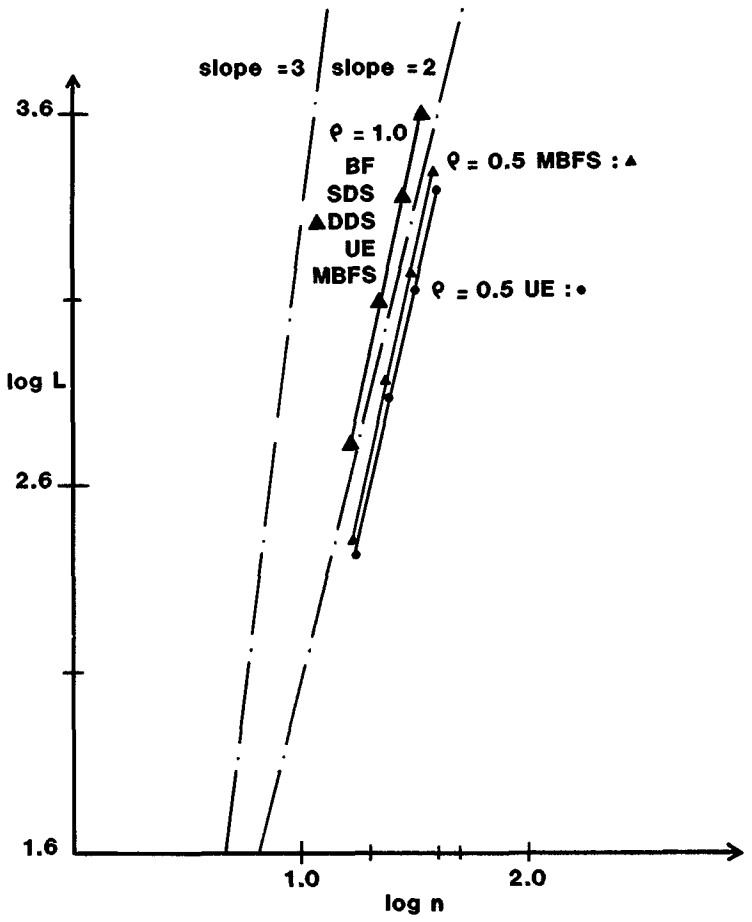

Fig. 2. Mean fundamental-cycle length L.

are classes of graphs for which MBFS yields a lower value for L than the others. One such graph is shown in Figure 4.

It may also be observed from the empirical results that SDS runs faster than BF, and MBFS runs faster than BF for graphs with edge densities less than 0.7. This is partly due to the fact that the computation time in generating a set of fundamental cycles is dependent on L, and algorithms that achieve a lower value for L will also run faster than algorithms that do not. However, our empirical results indicate that both DDS and UE achieve lower values for L than SDS and MBFS, but they spend more time in doing this (because of the overhead involved in more complex operations). Hence, we recommend the following:

(a) For graphs with edge densities ≤ 0.6, MBFS should be used, since it takes the least time among all the algorithms that were tried.
(b) For the same reason, on graphs with edge densities >0.6, SDS should be used.

We have proved in Section 2 that generating $T_{\min}$ is NP-complete. For the sake of completeness, we now give an example to show that none of our heuristic

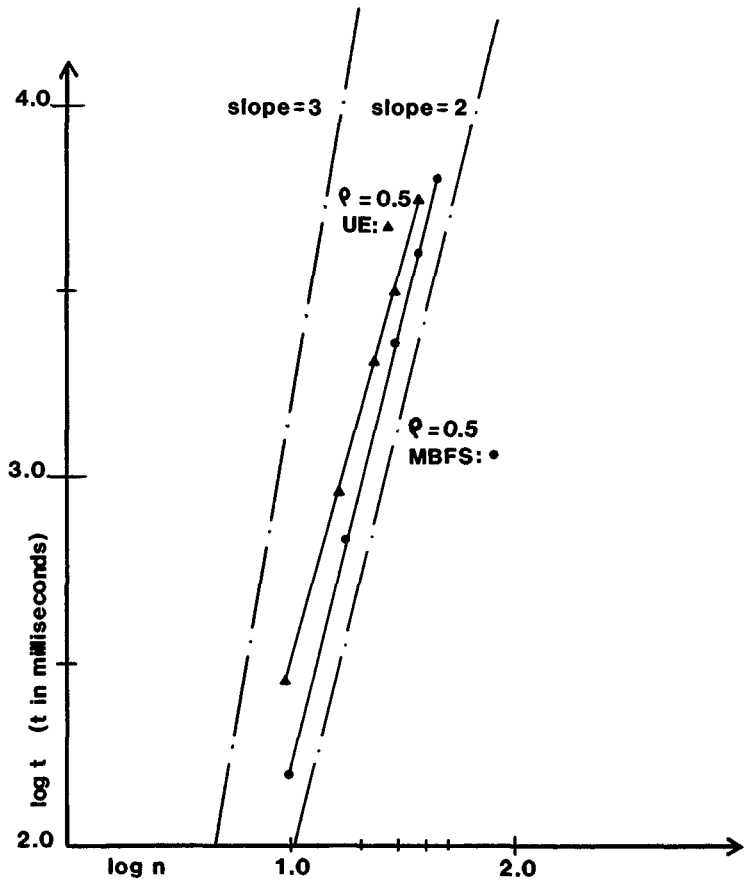

Fig. 3. Mean running time t.

algorithms yields an optimal solution. For the graph in Figure 5, the minimum value for L is 19, corresponding to the T_{min} shown in bold lines. Using UE and exploring from the vertices in the order a, e, b, we find L to be 20. Using DDS, SDS, and MBFS, and exploring from the vertices in the order a, b, d, e, we find L to be 21.

Therefore, all four algorithms are only suboptimal.

All the algorithms we have discussed use some form of breadth first search in which a new vertex to be explored from is selected by some criteria involving the degrees of the vertices and not any other property of the graph. Our empirical results show that for graphs with up to 50 vertices, these heuristics yield a value for L that is $O(n^2)$. It would be interesting to explore whether this is also the case for larger graphs. Such an analysis is carried out in the next section.

6. BOUND ON THE EXPECTED VALUE OF $L(T_{min})$

Though generating an arbitrary spanning tree of a given graph takes only linear time in the number of vertices and edges, generating a fundamental set of cycles requires time proportional to the total length L of the fundamental-cycle set

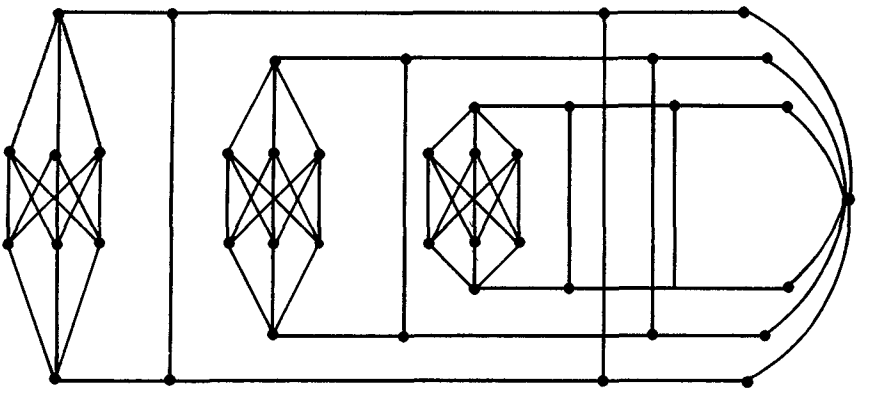

Figure 4.

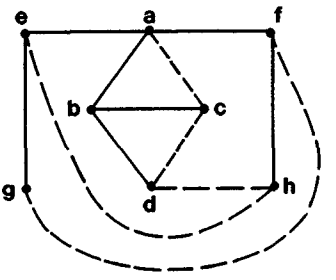

Figure 5.

[5]. Therefore, it is of considerable interest to establish some tight bound on the length $L(T_{\min})$ of a minimal fundamental-cycle set for an arbitrary connected graph.

In a graph of order n the length of a cycle is at most n. The total number of fundamental cycles in a connected graph (with n vertices and e edges) is its nullity $\mu = e - n + 1$. Therefore an obvious worst case bound on L is $n(e - n + 1)$ or $O(n^3)$. What is interesting, however (as Section 5 indicates), is that this bound on $L(T_{\min})$ may actually be $O(n^2)$. This is because (loosely speaking) graphs that have fundamental cycles with length $O(n)$ seem to have fewer of them (sparse graphs), whereas graphs in which the number of fundamental cycles is $O(n^2)$ seem to have the length of these cycles bounded by a constant. Although we were unable to prove this bound of $O(n^2)$ for $L(T_{\min})$ in general, the following probabilistic analysis shows that the expected bound on L generated by a BFS is $O(n^2)$, at least for certain "bad" classes of graphs. Since in all our heuristic algorithms the next vertex to explore from is selected on the basis of the degree, regular graphs prevent us from exploiting the degree distribution. Thus regular graphs are, in some sense, "bad" inputs to these algorithms. Therefore we are justified in considering only the class of regular graphs for analysis in this section.

Consider a large graph G with n vertices and e edges. Let the graph be random in the sense that the occurrence of an edge between any distinct vertex pair is equally likely. Let p be the probability of occurrence of an edge (i, j) for all $1 \leq i \leq n$, $1 \leq j \leq n$ and $i \neq j$. Then $q = (1 - p)$ is the probability that there is no edge

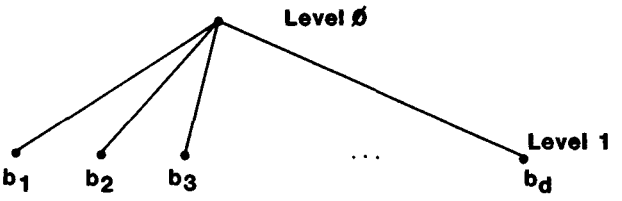

Figure 6.

between vertex i and vertex j. For more detailed properties of such graphs and BFS on them one should refer to Tinhofer [24, chap. 1].

Since all vertices have identical degrees, any of n vertices in G can be chosen as the starting vertex, that is, the root of the BFS tree. So there is one vertex at level 0 of the BFS tree. After exploring from this vertex, the expected number of vertices at level 1 is $p(n-1)$. The BFS tree at this stage is as shown in Figure 6.

Since G is regular of degree $d = p(n-1)$, we can choose any vertex at level 1 as the next vertex to explore from. Let us arbitrarily choose b_1 as the next vertex to explore from. Vertex b_1 has $(d-1)$ vertices to choose from as adjacent ones at level 1, so the expected number of edges incident on b_1 and at level 1 is $p(d-1)$. These edges are all chords, since they connect vertices already in the tree. The remaining $(1-p)(d-1)$ edges are tree edges and must lead to new vertices at level 2.

We choose b_2 as the next vertex to be explored from. The expected number of chords from b_2 at level 1 is also

$$p(d-1).$$

The expected number of chords from b_2 to the descendants of b_1 at level 2 is

$$p(1-p)(d-1).$$

Since the degree of b_2 is d, the expected number of tree edges from b_2 leading to new vertices at level 2 is

$$(1-p)^2(d-1).$$

We choose b_3 as the next vertex to be explored from. The expected number of chords from b_3 at level 1 is

$$p(d-1).$$

The expected number of chords from b_3 to the descendants of b_1 and b_2 at level 2 is

$$p(1-p)(d-1) + p(1-p)^2(d-1).$$

Since the degree of b_3 is d, the expected number of tree edges from b_3 leading to new vertices at level 2 is

$$(1-p)^3(d-1).$$

Proceeding in this fashion, the expected number of vertices at level 2,

$$n_2 = (d-1)\sum_{k=1}^{d}(1-p)^k = \frac{(d-1)\,(1-p)\,(1-(1-p)^d)}{p}. \tag{4}$$

The chords at level 1 form fundamental cycles of length 3, and the expected number of such cycles is

$$\frac{p \cdot d \cdot (d-1)}{2}. \tag{5}$$

The chords from level 1 to level 2 form fundamental cycles of length 4, and the expected number of such cycles is

$$\sum_{k=1}^{d-1} (d-k) \cdot p \cdot (1-p)^k \cdot (d-1). \tag{6}$$

Exploring from vertices at level 2, we find that the expected number of fundamental cycles of length 3 at level 2 is

$$\frac{p \cdot n_2 \cdot (n_2 - 1)}{2} \cdot p^2 \tag{7}$$

(the p^2 factor arises because a fundamental cycle is of length 3 at level 2 only if both the end vertices of the chord at level 2 have a common parent at level 1).

Similarly, the expected number of fundamental cycles of length 5 at level 2 is

$$\frac{p \cdot n_2 \cdot (n_2 - 1)}{2} \cdot (1 - p^2). \tag{8}$$

We proceed in this fashion exploring from vertices level by level until there are no more vertices to be explored. We wish to determine the expected value of L generated by the heuristic algorithms described earlier.

For graphs with $p = 1$ the tree generated is the star tree with all fundamental cycles of length 3; so

$$L_{p=1} = 3 \cdot \frac{(n-1)\,(n-2)}{2} \simeq \frac{3}{2} \cdot n^2.$$

As the analysis of the expected length becomes complicated for an arbitrary value of p, we compute it for two more values of p: $p = \frac{1}{2}$ and $p = \frac{1}{4}$. These turn out to be

$$L_{p=1/2} \leq \frac{9}{8} n^2 + O(n) \qquad \text{and} \qquad L_{p=1/4} \leq \frac{3}{4} n^2 + O(n).$$

Thus for regular random graphs of densities 1, $\frac{1}{2}$, and $\frac{1}{4}$, the expected value for $L(T)$, where T is an arbitrary BFS spanning tree, is kn^2, where $0 < k \leq \frac{3}{2}$. Our empirical observation on L (Tables I–V and Figure 2) shows that the average value of L increases monotonically with density, for random graphs of all sizes for each one of our algorithms. We therefore venture the conjecture that $L(T_{\min})$ is bounded by $O(n^2)$. More specifically, $L(T_{\min}) \leq \frac{3}{2} \cdot n^2$. It seems that although the nullity μ of an n-vertex graph can be as high as $O(n^2)$, only a fixed number of fundamental cycles (with respect to $T_{\min}$) can have lengths that are proportional to n, thus making the bound for $L(T_{\min})$ $O(n^2)$ and not $O(n^3)$.

7. CONCLUSIONS

The computational time complexity of the best known algorithms for generating a set of fundamental cycles is of the order of the total length L of the set of

fundamental cycles. Hence algorithms that strive to minimize this length could prove more efficient in generating a set of fundamental cycles, provided, of course, that the overhead involved in doing this is not too high. We have shown here that it is not worth the effort to obtain the absolute minimum fundamental-cycle set, because it turns out to be an NP-complete problem.

We have, however, proposed some fast heuristic algorithms to obtain the suboptimal solutions. Two of these, SDS and MBFS, are recommended in particular, on the basis of our empirical studies. Both of these are faster than the straightforward BFS generation and in general produce smaller values for L.

All our algorithms are based on breadth first search, where a new vertex to explore from is selected on some criteria based on the degrees of the vertices. It would be useful to consider other heuristic approaches for efficiently generating a set of fundamental cycles based on some other property of the graph.

An analysis to establish a tight upper bound on $L(T_{\min})$ for a worst case input for our algorithms was attempted. Although we were unable to establish such a bound on $L(T_{\min})$, we did show the expected value of $L(T')$ (T' being a suboptimal spanning tree) to be $O(n^2)$—analytically for regular graphs and empirically for random graphs. This suggests, but in no way proves, that $O(n^2)$ is the lower bound for generating a set of fundamental cycles for a worst case graph. The proof of our conjecture that $L(T_{\min}) \leq \frac{3}{2} \cdot n^2$ is suggested for further investigation in this area. An analytic proof of the conjecture that $L(T_{\min})$ increases monotonically with edge density of the graph will be an even stronger result.

ACKNOWLEDGMENTS

We wish to thank Richard Wong, Robert McNaughton, and Karl Winklmann for going over an earlier draft of the paper and providing valuable suggestions. We are also grateful to one of the referees for suggesting many improvements and clarifications in the final version.

REFERENCES

1. AHO, A.V., HOPCROFT, J.E., AND ULLMAN, J.D. *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass., 1974.
2. CORNEIL, D B., AND READ, R C. The graph isomorphism disease *J Gr. Theory 1* (1977), 339–363.
3. DEO, N. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall, Englewood Cliffs, N J., 1974.
4. DEO, N. Breadth and depth first searches in graph theoretic algorithms *J. Comput. Soc India 4* (1974), 1–8.
5. DEO, N. Minimum-length fundamental cycle set. *IEEE Trans. Circuits Systems CAS-26* (Oct 1979), 894–895.
6. GAREY, M.R., AND JOHNSON, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco, 1979.
7. GIBBS, N.W. Algorithm 491: Basic cycle generation *Commun ACM 18,* 5 (May 1975), 275–276.
8. GOTLIEB, C.C., AND CORNEIL, D.G. Algorithms for finding a fundamental set of cycles for an undirected linear graph. *Commun. ACM 10,* 12 (Dec. 1967), 780–783.
9. HARARY, F. *Graph Theory.* Addison-Wesley, Reading, Mass , 1969.
10. HUBICKA, E., AND SYSLO, M.M. Minimal bases of cycles of a graph. In *Proc 2nd Czechoslovak Symp. Graph Theory.* Academia (Prague, 1975), pp. 283–293.
11. ITO, T., AND KIZAWA, M. The matrix rearrangement procedure for graph-theoretical algorithms and its application to the generation of fundamental cycles. *ACM Trans. Math. Softw. 3,* 3 (Sept 1977), 227–231.

12. JOHNSON, D.S., LENSTRA, J.D., AND RINNOOY KAN, A.H.G. The complexity of the network design problem. *Networks 8* (1978).
13. JOVANOVICH, A.D. Note on a modification of the fundamental cycle finding algorithm. *Inf. Process. Lett. 3* (July 1974).
14. KNUTH, D.E. *The Art of Computer Programming,* vol. 1. Addison-Wesley, Reading, Mass., 1968, pp. 363–368.
15. KOLASINSKA, E. On a minimum cycle basis of a graph. *Zastosow Matem 16* (1980), 631–639.
16. LEDERBERG, J. *Topology of Molecules. Mathematical Sciences,* M.I.T. Press, Cambridge, Mass., 1968, 37–51.
17. PATON, K. An algorithm for finding a fundamental set of cycles of a graph. *Commun. ACM 12,* 9 (Sept. 1969), 514–518.
18. REINGOLD, E.M., NIEVERGELT, J., AND DEO, N. *Combinatorial Algorithms: Theory and Practice.* Prentice-Hall, Englewood Cliffs, N.J., 1977.
19. STEPANEC, G.F. Basis systems of vector cycles with extremal properties in graphs (in Russian), *Usp. Mat. Nauk 19,* 2 (116) (1964), 171–175.
20. SUSSENGUTH, E., JR. A graph theoretical algorithm for matching chemical structures. *J. Chem. Doc. 5,* 1 (Feb. 1965), 36–43.
21. SYSLO, M.M. Fundamental set of cycles of a graph. *Zastosow Matem 13* (1973), 399–409.
22. SYSLO, M.M. Minimum-length cycle bases of a graph (Extended Abstract). *Methods Oper. Res. 37* (1980), 385–390.
23. TARJAN, R.E. Depth-first search and linear graph algorithms. *SIAM J. Comput. 1* (1972), 146–160.
24. TINHOFER, G. *Zufallsgraphen.* Hanser, Munich, 1980.
25. WELCH, J.T., JR. A mechanical analysis of the cyclic structure of undirected linear graphs. *J. ACM 13,* 2 (1966), 205–210.
26. ZYKOV, A.A. *Theory of Finite Graphs* (in Russian). Nauka, Novosibirsk, 1969.

Received January 1981; revised August 1981; accepted September 1981