SIAM J. Comput.
Vol. 5, No. 1, March 1976

ON ALGORITHMS FOR ENUMERATING ALL CIRCUITS OF A GRAPH*

PRABHAKER MATETI† AND NARSINGH DEO‡

Abstract. A brief description and comparison of all known algorithms for enumerating all circuits of a graph is provided, and upper bounds on computation time of many algorithms are derived. The vector space method of circuit enumeration is discussed. It is proved that K_3, K_4, $K_4 - x$ and $K_{3,3}$ are the only undirected and reduced graphs which do not have any edge-disjoint unions of circuits.

Key words. algorithms, graph theory, circuits, dicircuits, cycles, circuit vector space, circuit-graph, adjacency matrix, graph search, backtrack

1. Introduction. Given an undirected graph, the problem of enumerating all its circuits has received much attention lately. The analogous problem for a directed graph is to enumerate all its directed circuits. The two problems have much in common,, and we consider them in parallel.

In this paper, we survey all known algorithms for circuit enumeration and compare their efficiencies. In § 2, we classify and describe the basic ideas behind these algorithms. It is shown, in § 3, that only four graphs exist that have circuit vector spaces consisting of all circuits. The algorithms are compared in § 4, and conclusions are presented. To avoid misunderstanding, we define our terminology.

A *graph* G is a triple $\langle V, E, f\rangle$ where V is a finite set of *vertices*, E a finite set of *edges*, and f is a function. A graph is either *directed* (a *digraph*) when $f: E \to V \times V$, or is *undirected* when $f: E \to \{\{u, v\} | u, v \in V\}$. Observe that this definition permits the graph to have parallel edges (two edges e_1 and e_2 are *parallel* if $e_1 \neq e_2$ and $f(e_1) = f(e_2)$) and self-loops (an edge e is a *self-loop* if for some vertex $v, f(e) = (v, v)$ for digraph or $f(e) = \{v\}$ for undirected graph). A graph G is *simple* if it has neither self-loops nor parallel edges. An edge e of a digraph is said to be *incident out* of vertex v_1, and *incident into* a vertex v_2 if $f(e) = (v_1, v_2)$. The edge e is *incident with* both vertices v_1 and v_2. The *in-degree* of a vertex v of a digraph is the number of edges incident into v, and the *out-degree* of v is the number of edges incident out of v. An edge e of an undirected graph is *incident with* vertices v_1 and v_2 if $f(e) = \{v_1, v_2\}$. The *degree* of a vertex v in an undirected graph is the number of edges incident with this vertex, self-loops being counted twice.

Two vertices v_1 and v_2 of a directed graph G are *adjacent* if there is some edge e so that $f(e) = (v_1, v_2)$ or (v_2, v_1). An *edge-sequence* from vertex v_0 to v_k in a digraph is an alternating sequence of vertices and edges, $v_0e_1v_1e_2v_2 \cdots v_{k-1}e_kv_k$, such that for $1 \leqq i \leqq k$, $f(e_i) = (v_{i-1}, v_i)$. This edge sequence is said to *pass through* the vertices v_i, $0 < i < k$. An edge sequence is *simple* if it does not pass through a vertex more than once. A *dipath* from v_0 to v_k is an edge-sequence from v_0 to v_k where all vertices are distinct, except possibly that $v_0 = v_k$. When $v_0 = v_k$,

* Received by the editors August 13, 1973, and in final revised form April 2, 1975. This work was supported in part by the National Science Foundation under Grant GJ-31222.

† Department of Computer Science, University of Illinois, Urbana, Illinois 61801.

‡ Department of Electrical Engineering, Indian Institute of Technology, Kanpur, India. Now at Computer Science Department, Washington State University, Pullman, Washington, 99163.

the dipath is called a *rooted dicircuit* rooted at v_0. The rooted dicircuit considered as a subgraph of G is called a *dicircuit*. The *length* of an edge-sequence is the number of edges in it. By *deleting an edge e* we obtain a new graph $G' = \langle V, E', f' \rangle$ where $E' = E - \{e\}$ and f' is a restriction of f to E'. By *deleting a vertex v* from a graph G we obtain a new graph $G' = \langle V', E', f' \rangle$, where $V' = V - \{v\}$ and $E' = \{e \in E | f(e) \neq (v, u) \text{ or } (u, v) \text{ for } u \in V\}$ and $f' : E' \to V' \times V'$ is a restriction of f to E'. By *ignoring* the direction of an edge e we mean that the ordered pair $f(e) = (u, v)$ be considered as a set $\{u, v\}$. The corresponding definitions for an undirected graph follow if the directions of the edges are ignored.

An undirected graph is *connected* if there exists a path between every pair of vertices. A vertex is a *cut-vertex* if it lies on every path between a pair of vertices. A connected undirected graph is *separable* if it has at least one cut-vertex. A maximal, connected subgraph is called a *component* of the graph. A maximal, nonseparable subgraph of a graph is called a *block* of the graph. A directed graph is *strongly-connected* if there exists a dipath between every pair of vertices. A maximal, strongly-connected subgraph of a digraph is called a *fragment*. The *undirected version* G of a digraph D is obtained by ignoring the direction of the edges of D. The *directed version* $D = \langle V, E_D, f_D \rangle$ of an undirected graph $G = \langle V, E, f \rangle$ is obtained by introducing two edges e' and e'' into E_D such that $f_D(e') = (u, v)$ and $f_D(e'') = (v, u)$ iff $e \in E$ and $f(e) = \{u, v\}$. A *spanning tree* of a connected undirected graph is a maximal subgraph which has no circuits. The unique circuit obtained by adding a nontree edge to the spanning tree is called a *fundamental circuit* (with respect to that spanning tree). The *ring-sum* of two circuits $C_1 = \langle V_1, E_1, f_1 \rangle$ and $C_2 = \langle V_2, E_2, f_2 \rangle$ of an undirected graph G is an undirected graph $S = C_1 \oplus C_2 = \langle V', E', f' \rangle$ where $E' = E_1 \cup E_2 - E_1 \cap E_2$, and for all $e \in E'$, $f'(e) = f(e)$, and $V' = \{u, v \in V | \text{ for some } e \in E', f'(e) = \{u, v\}\}$. A *variable adjacency matrix* A of a digraph $D = \langle V, E, f \rangle$ is an $n \times n$ symbolic matrix in which the (i, j)-element $A_{ij} = \sum_{\text{all } e_k} e_k$, where $e_k \in E$ such that $f(e_k) = (v_i, v_j)$. If there is no such e_k, then $A_{ij} = 0$. The powers of A, $p \geqq 1$, are defined as follows: $A^1 \equiv A$, and for $p > 1$, $A^p \equiv A^{p-1} \cdot A = A \cdot A^{p-1}$, employing the usual symbolic multiplication. To obtain a binary adjacency matrix, replace all nonzero entries A_{ij} by 1's.

We shall use n to denote the number of vertices, e the number of edges, and c the number of circuits of the given graph G. We shall refer to an edge-sequence of length k as a k-sequence. Similarly we use k-dipath and k-dicircuit, etc.

2. Outline of circuit enumeration algorithms.

In principle, any algorithm enumerating all dicircuits of a digraph can be used to enumerate all circuits of an undirected graph, and vice versa. (However, see § 4 for computational complexity considerations.) The algorithms are therefore considered together. Every circuit enumeration algorithm proposed can be put into one of the following four classes, depending on the underlying approach:

1. circuit vector space for undirected graphs;
2. search algorithms;
3. powers of adjacency matrix;
4. edge-digraph.

In the following subsections, we attempt to describe the basic ideas behind various algorithms belonging to each class. More details may be found in Mateti and Deo [19].

2.1. Circuit vector space algorithms. It is well known that the set of all circuits and edge-disjoint unions of circuits of an undirected graph is a vector space over $GF(2)$, with the vector addition corresponding to the ring-sum operation on subgraphs. To obtain all circuits, one can start from a basis for the circuit vector space. The dimension of this space is $\mu = e - n + 1$, for a connected graph of n vertices and e edges. All the $2^\mu - \mu - 1$ vectors (excluding the basic circuits themselves, and the null element) are computed. Since not every vector is a circuit, a test is necessary to determine if the vector generated is a circuit. Generally only a small fraction of the $2^\mu - \mu - 1$ vectors are circuits, the rest being edge-disjoint unions of circuits. In fact, we will prove in § 3 that there are only four undirected graphs having $2^\mu - 1$ circuits, and we will exhibit classes of graphs for which the ratio of number of circuits to the number of vectors goes asymptotically to zero as the number of vertices increases.

An attempt to compute only a subset of all vectors and yet enumerate all circuits was made by Welch [36], and later by Hsu and Honkanen [15]. Gibbs [13] showed that Welch's attempt led to incorrect results, and Mateti and Deo [19] give a class of graphs for which Hsu and Honkanen's algorithms end up computing all $2^\mu - \mu - 1$ vectors.

Algorithms belonging to the vector space class are: Maxwell and Reed [20], Welch [36], Gibbs [13], Rao and Murti [26], Hsu and Honkanen [15], Mateti and Deo [19].

2.2. Search algorithms. These algorithms search for circuits in an appropriate search space which is a super set containing all circuits. The efficiency of such an algorithm depends on (i) the size of this super set, (ii) the effort it takes to compute an element in the search space, and (iii) a test to find if the element is indeed a circuit. The algorithms of Char [6], and Chan and Chang [5] have the set of all permutations of vertices of the graph as the search space.

Several algorithms since Floyd [12] have used backtracking to generate dipaths of the graphs, and then identified if it is a dicircuit. The algorithm of Tarjan [32] uses improved pruning methods to decrease the size of the subset of dipaths generated considerably. Further improvements in this algorithm are made by Johnson [16], Read and Tarjan [28] and Szwarcfiter and Lauer [31].

The algorithms of Roberts and Flores [29], Floyd [12], Tiernan [33], Berztiss [2], [3], Weinblatt [34], and Ehrenfeucht et al. [11] are also basically backtrack algorithms.

2.3. Algorithms using powers of adjacency matrix. Let A be the variable adjacency matrix of the given digraph. Then a nonzero element A_{ij}^p, $p \geqq 1$, is the set of all p-sequences from vertex v_i to vertex v_j. Since no dicircuit can be of length greater than n, we need compute only up to A^n. Observe that an edge-sequence need neither be a dipath nor a dicircuit. The crux of the problem is to avoid such nonsimple edge-sequences. It is in this avoidance that different algorithms in this class vary. Note that all algorithms using this method find rooted dicircuits.

Algorithms belonging to this class are: Ponstein [25], Yau [37], Kamae [17], Danielson [9], Ardon and Malik [1].

2.4. Algorithm using edge-digraph of a digraph. Consider a digraph $E(D)$ derived from the given digraph D as follows: the vertex-set of $E(D)$ is the edge-set of D, and $E(D)$ contains an edge q between its vertices e_1 and e_2 iff e_1 and e_2 (which are edges of D) are adjacent in D.

The edge-digraph of a p-dipath is a $(p - 1)$-dipath; the edge-digraph of a p-dicircuit is also a p-dicircuit. Thus we see that there is a one-to-one correspondence between the dicircuits of D and $E(D)$. An edge of $E(D)$ will correspond to a rooted 2-dicircuit. Enumerate and delete all 2-dicircuits of $E(D)$, and call the resulting graph D^1. Now, take the edge-digraph of D^1. $E(D^1)$ has only dicircuits of length $p \geqq 3$. Enumerate and delete all dicircuits of length 3; call the resulting digraph D^2, and so on until a D^p, for some p, is empty.

An algorithm using this approach is due to Cartwright and Gleason [4].

3. Circuit vector space of undirected graph. All circuit vector space algorithms, known so far, compute all of the 2^μ vectors, in the worst case. But there are classes of graphs for which the ratio of number of circuits c to vectors approaches zero for large μ. In fact, there are only four graphs with $c = 2^\mu - 1$. These four graphs are shown in Fig. 1. To prove this, it is useful to have the following definition.

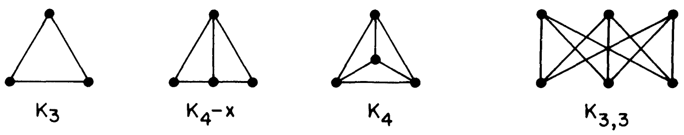


FIG. 1. *Graphs all of whose vectors are circuits*

DEFINITION 1. A graph G is said to be *reduced* if

(i) G is simple,

(ii) G has no vertex of degree 0 or 1, and,

(iii) for every vertex v of degree 2, the two vertices adjacent to v are themselves adjacent.

LEMMA. *Let G be a reduced graph with the number of vertices $n \geqq 6$. If G has no edge-disjoint union of circuits, then the number of edges $e \geqq n + 3$. The converse is not true.*

Proof. Since G is reduced, and has no edge disjoint union of circuits, G must be connected. No vertex in G can be of degree less than 3; for otherwise, G is either not reduced or has an edge-disjoint union of circuits. Therefore $e \geqq 3n/2 = n + n/2 \geqq n + 3$, since $n \geqq 6$.

The graphs in Fig. 2 have $e = n + 3$ and yet have edge-disjoint unions of circuits.

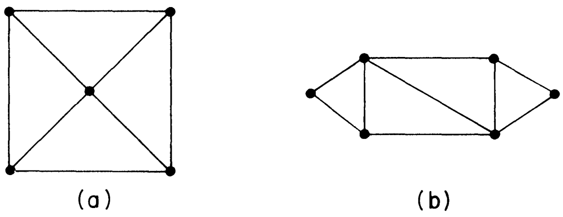


FIG. 2. *Graphs with $e = n + 3$ and yet having edge-disjoint unions of circuits*

THEOREM 1. *Only the four reduced graphs K_3, K_4, $K_4 - x$ and $K_{3,3}$ have all vectors as circuits.*

Proof. That the theorem is true for graphs with at most five vertices and seven edges can be seen by enumerating reduced graphs. It is also true for graphs with n vertices $n \geqq 6$, and e edges, $e \leqq n + 2$ by the lemma. For graphs with $n \geqq 5$, and $e \geqq n + 3$, the theorem will be proved by contradiction. We will first assume that there exists a graph G which has $n \geqq 5$, is not $K_{3,3}$, and yet has all vectors as circuits, and then show a contradiction. If G is disconnected or separable, we are done because every reduced graph which is separable or disconnected will contain one or more edge-disjoint union of circuits. Therefore G is connected and is nonseparable. Graph G is either planar or nonplanar.

Case 1. G is planar, $n \geqq 5$, $e \geqq n + 3$; that is, nullity $\mu = e - n + 1$ is at least 4.

There are at least 4 finite regions and one infinite region defined by a plane representation of the graph G (see Deo [10, Chap. 5]). If any two finite regions are not adjacent, the circuits defining these regions are edge-disjoint. Likewise, every finite region must also be adjacent to the infinite region; otherwise the circuit defining the finite region is disjoint with the circuit defining the infinite region. The dual of G is therefore a complete graph $K_{\mu+1}$ of $\mu + 1$ vertices, and $\mu + 1 \geqq 5$ which is impossible. Thus there must exist a pair of regions which are not adjacent, and hence an edge-disjoint union of circuits.

Case 2. G is nonplanar, and therefore G either has a subgraph homeomorphic to $K_{3,3}$, or has a subgraph homeomorphic to K_5.

Case 2.1. Let G have a subgraph homeomorphic to K_5. An edge-disjoint union of circuits of K_5 shown in Fig. 3 is homeomorphic to a subgraph of G.

Case 2.2. G has a proper subgraph g homeomorphic to $K_{3,3}$. Since g is a proper subgraph, there exists an additional path between two vertices u and v. If u and v correspond to vertices in the same independent set of vertices of $K_{3,3}$,

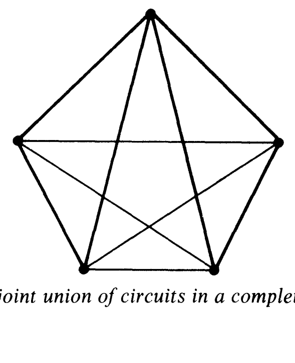


FIG. 3. *An edge-disjoint union of circuits in a complete graph of five vertices*

there exists an edge-disjoint union of circuits (see Figure 4(a)). If u and v correspond to vertex u' in one independent set of vertices of $K_{3,3}$ and to v' in the other independent set, respectively, an edge-disjoint union of circuits exists (see Figure 4(b)).

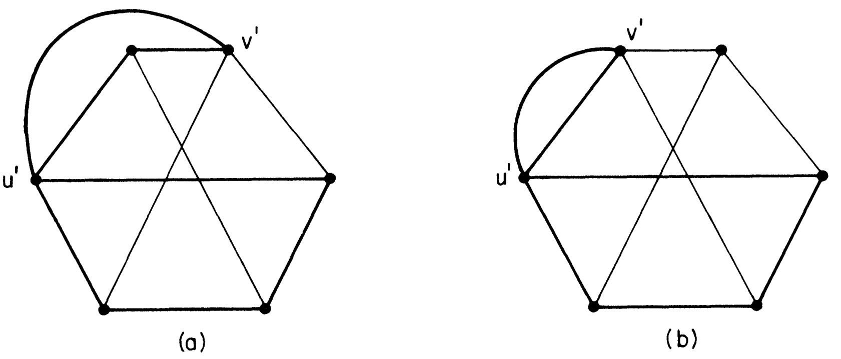
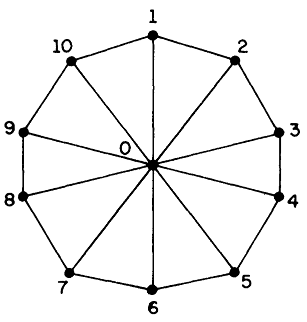


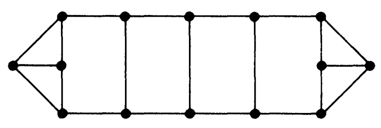
FIG. 4. *Edge-disjoint unions of circuits in a nonplanar graph having a proper subgraph homeomorphic to* $K_{3,3}$

Observe that the Theorem 1 does not give us information as to how many edge-disjoint unions of circuits a given graph can have. In general, a large fraction of the vectors are edge-disjoint unions of circuits. The ratio of circuits to all vectors in the vector space for numerous classes of graphs tends to zero. Three such classes are shown in Fig. 5.



$\mu = n - 1$
$c = \mu(\mu - 1) + 1$

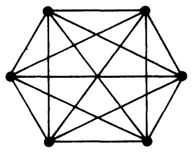
(a) *A wheel graph*



$\mu = \frac{1}{2}(n + 2)$

$$c = \frac{\mu^2}{2} + \frac{5\mu}{2} - 3$$

(b) *A modified ladder graph*



$$\mu = \frac{n^2}{2} - \frac{3n}{2} + 1$$

$$c = \frac{1}{2}\sum_{s=3}^{n} \binom{n}{s}(s-1)!$$

(c) *A complete graph*

FIG. 5. *Graphs for which* $\lim_{n\to\infty}(c/2^{\mu}) \to 0$

4. Analyses of algorithms. We take as our data unit a single vertex, or an edge. Other data objects like paths, circuits, or graphs are composed of these basic data units. The space bounds for various algorithms are given in terms of these data units. Any operation performed on an edge, between two edges, or between two vertices takes a unit of time. Thus adding an edge to a partly constructed path, testing if the label of a vertex is greater than that of another vertex, given e finding $f(e)$, etc., all take a unit of time. All other operations performed in the actual execution of an algorithm are assumed to take no time. The time bounds of the algorithms are given in terms of these time units.

In what follows we give summaries of analyses performed on the algorithms. Space bounds are rather obvious and are given in Table 1 at the end of this section. Since the number of time units consumed by some of these algorithms depends intricately on the structure of the graph whose circuits are being enumerated, the worst-case time bounds given are in general pessimistic bounds. Thus if T_A and T_B are upper bounds on the time required by two algorithms A and B, respectively, and if $T_A < T_B$, it cannot be said with certainty that algorithm A is faster than B. The algorithms A and B are often directly compared with each other to make a statement to the effect that A is faster than B on any given graph. In general, the worst-case graphs for two algorithms are not the same. For a discussion on related matters, see Chase [8].

It will be computationally advantageous to perform some initial simplifications on the given graph which do not disturb the circuit structure of the graph before proceeding to enumerate all circuits. Self-loops can be enumerated initially. Then if the given graph is a directed graph, take its fragments, take the undirected version of each fragment and decompose it into blocks, and enumerate the dicircuits of the subdigraph corresponding to each of these blocks. A set of p parallel edges can be replaced by a single new edge. For each dicircuit containing this new edge, enumerate p dicircuits, each containing an edge from the set of edges replaced. Vertices v of in-degree and out-degree 1 may be deleted by adding an edge connecting the two adjacent vertices of v. Note, however, that this might result in parallel edges. Similar editing operations may be performed on undirected graphs. This is advantageous because all these editing operations have upper bounds of $O(n + e)$ (see Hopcroft and Tarjan [14]). And even the fastest known algorithm (Johnson's) has an upper time bound[1] of $O((n + e)(c + 1))$. For the purpose of time analysis, we will therefore assume that the graph given is an "edited" graph.

[1] For some positive constant K, and given function $\Psi(\cdot)$, $O(\Psi(\cdot))$ means that the number being represented by $O(\Psi(\cdot))$ is at most $K \cdot \Psi(\cdot)$; $T = O(\Psi(\cdot))$ means that $T \leqq K \cdot \Psi(\cdot)$; and $\Psi(\cdot) = O(T)$ means that $T \geqq K \cdot \Psi(\cdot)$.

If the size of the input is a and the size of the output is b, then any upper bound cannot be less than $O(a + b)$. We insist that each circuit be enumerated as a sequence of vertices or edges. Thus $b = O(n \cdot c)$. Johnson's algorithm approaches this bound.

It may be pointed out that an algorithm A enumerating all the dicircuits of a digraph may be used on an undirected graph G by taking the directed version D of G. The digraph D is obtained from G by replacing each edge e of G by two oppositely directed edges e' and e''. For each circuit of G, we now have two dicircuits in D, and, in addition, e 2-dicircuits are created, where e is the number of edges in G. Thus if c_D is the number of dicircuits in D, and c_G is the number of circuits in G, $c_D = 2c_G + e$. (However, an algorithm for undirected graphs, such as a circuit vector space algorithm, should not be applied on an undirected version G of a digraph D to enumerate the dicircuits because the number of circuits in G could be exponential in the number of dicircuits of D.) Thus circuit vector space algorithms should not be used even to enumerate the circuits of an undirected graph until progress has been made in the pruning methods used to avoid fruitless computations of noncircuit vectors giving a circuit vector space algorithm having an upper time bound of $O((n + e)c)$. When this is achieved, the multiplying constants, which depend on the implementation, may make one algorithm better to use than the other. Such a progress appears promising if it is remembered that the backtrack algorithms for digraphs have been improved from exponential (Tiernan [33]) to polynomial in output (Johnson [16]) in three years.

TABLE 1
Upper bounds on time and space

Algorithm of	Time bound	Space bound	Method used
Ardon and Malik	$n(\text{const.})^n$	n^2	Powers of adjacency matrix
Berztiss	$n(\text{const.})^n$	$n + e$	Backtrack
Cartwright and Gleason	$n(\text{const.})^n$	$n(\text{const.})^n$	Edge-digraph
Chan and Chang	$\sum_{i=1}^{n-1} i!i$	$n + e$	Searches permutations
Char	$\sum_{i=3}^{n} n!/(n-i)!$	$n + e$	Searches permutations
Danielson	$n(\text{const.})^n$	$n(\text{const.})^n$	Powers of adjacency matrix
Ehrenfeucht *et al.*	$n^3 + n \cdot e \cdot c$	$n + e$	Backtrack
Hsu and Honkanen	$n \cdot 2^{2\mu}$	$e \cdot 2^{\mu}$	Circuit vector space
Johnson	$(n + e)c$	$n + e$	Backtrack
Kamae	—	$n(\text{const.})^n$	Powers of adjacency matrix
Mateti and Deo	$\mu^2 \cdot 2^{\mu}$	μ^2	Circuit vector space
Maxwell and Reed	$n \cdot 2^{2\mu}$	$e \cdot 2^{\mu}$	Circuit vector space
Ponstein	—	$n(\text{const.})^n$	Powers of adjacency matrix
Rao and Murti	$e \cdot \mu^2 \cdot 2^{\mu}$	$\mu \cdot n$	Circuit vector space
Read and Tarjan	$(n + e)c$	$n + e$	Backtrack
Szwarcfiter and Lauer	$(n + e)c$	$n + e$	Backtrack
Tarjan	$n \cdot e \cdot c$	$n + e$	Backtrack
Tiernan (RFFT)	$n(\text{const.})^n$	$n + e$	Backtrack
Weinblatt	$n(\text{const.})^n$	$n \cdot c$	Backtrack
Welch-Gibbs	$n \cdot 2^{2\mu}$	$e \cdot 2^{\mu}$	Circuit vector space
Yau	—	$n(\text{const.})^n$	Powers of adjacency matrix

5. Conclusions. Of all the algorithms analyzed, the Johnson's algorithm [16] having an upper bound of $O((n + e)c)$ on the time is the asymptotically fastest algorithm. (Szwarcfiter and Lauer [31] give a similar algorithm with the same upper bound.) The success of this backtrack algorithm is due to a very effective pruning technique which avoids much of the fruitless search present in earlier algorithms.

Little has been done in circuit vector space algorithms by way of pruning unnecessary computations. The algorithms of Hsu and Honkanen [15], and Mateti and Deo [19] are the fastest among circuit vector space algorithms. It cannot be said of these two algorithms that one is faster than the other.

Acknowledgments. The authors wish to thank the referees for making valuable suggestions. The authors are grateful to Professor Jurg Nievergelt for many insightful discussions.

REFERENCES

[1] M. T. Ardon and N. R. Malik, *A recursive algorithm for generating circuits and related subgraphs*, 5th Asilomar Conf. on Circuits and Systems 5, Pacific Grove, Calif., Nov. 1971, pp. 279–284.
[2] A. T. Berztiss, *Data Structures: Theory and Practice*, Academic Press, New York, 1971.
[3] ———, *A K-tree algorithm for simple cycles of a directed graph*, Tech. Rep. 73-6, Dept. of Computer Sci., Univ. of Pittsburgh, 1973.
[4] D. Cartwright and T. C. Gleason, *The number of paths and cycles in a digraph*, Psychometrika, 31 (1966), pp. 179–199.
[5] S. G. Chan and W. T. Chang, *On the determination of dicircuits and dipaths*, Proc. 2nd Hawaii Internat. Conf. System Sci., Honolulu, Hawaii, 1969, pp. 155–158.
[6] J. P. Char, *Master circuit matrix*, Proc. IEE (London), 115 (1968), pp. 762–770.
[7] ———, *Master circuit matrix*, Ibid., 117 (1970), pp. 1655–1656.
[8] S. M. Chase, *Analysis of algorithms for finding all spanning trees of a graph*, Ph.D. thesis, Rep. 401, Dept. of Computer Sci., Univ. of Illinois, Urbana, Ill., 1970.
[9] G. H. Danielson, *On finding the simple paths and circuits in a graph*, IEEE Trans. Circuit Theory, CT-15 (1968), pp. 294–295.
[10] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice-Hall, Englewood Cliffs, N.J., 1974.
[11] A. Ehrenfeucht, L. D. Fosdick and L. J. Osterweil, *An algorithm for finding the elementary circuits of a directed graph*, Tech. Rep. CU-CS-024-73, Dept. of Computer Sci., Univ. of Colorado, Boulder, 1973.
[12] R. W. Floyd, *Nondeterministic algorithms*, J. Assoc. Comput. Mach., 14 (1967), pp. 636–644.
[13] N. E. Gibbs, *A cycle generation algorithm for finite undirected linear graphs*, Ibid., 16 (1969), pp. 564-568.
[14] J. Hopcroft and R. Tarjan, *Efficient algorithms for graph manipulation*, Comm. ACM, 16 (1973), pp. 372–378.
[15] H. T. Hsu and P. A. Honkanen, *A fast minimal storage algorithm for determining all the elementary cycles of a graph*, Computer Sci. Dept., Pennsylvania State Univ., University Park, 1972.
[16] D. B. Johnson, *Finding all the elementary circuits of a directed graph*, this Journal, 4 (1975), pp. 77–84.
[17] T. Kamae, *A systematic method of finding all directed circuits and enumerating all directed paths*, IEEE Trans. Circuit Theory, CT-14 2 (1967), pp. 166–171.
[18] J. W. LaPatra and B. R. Meyers, *Algorithms for circuit enumeration*, IEEE Internat. Conv. Record, part 1, 12 (1964), pp. 368–373.
[19] P. Mateti and N. Deo, *On algorithms for enumerating all circuits of a graph*, UIUCDCD-R-73-585 (revised), Dept. of Computer Sci., University of Illinois, Urbana, Ill., 1973.

[20] L. M. MAXWELL AND G. B. REED, *Subgraph identification—Segs, circuits and paths*, 8th Midwest Symp. on Circuit Theory, Colorado State Univ., Fort Collins, Colo., June 1965, pp. 13-0–13-10.

[21] R. L. NORMAN, *A matrix method for location of cycles of a directed graph*, Amer. Inst. Chem. Engrs. J., 11 (1965), pp. 450–452.

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

[23] ———, *An algorithm for the blocks and cut-nodes of a graph*, Ibid., 14 (1971), pp. 468–476.

[24] M. PRABHAKER, *Analysis of algorithms for finding all circuits of a graph*, Master's tech. thesis, Dept. of Electrical Engrg., Indian Inst. of Tech., Kanpur, India, 1972.

[25] J. PONSTEIN, *Self-avoiding paths and adjacency matrix of a graph*, SIAM J. Appl. Math., 14 (1966), pp. 600–609.

[26] V. V. B. RAO AND V. G. K. MURTI, *Enumeration of all circuits of a graph*, Proc. IEEE, 57 (1969), pp. 700–701.

[27] V. V. B. RAO, K. SANKARA RAO, R. SANKARAN AND V. G. K. MURTI, *Planar graphs and circuits* Matrix Tensor Quart., 18 (1968), pp. 88–91.

[28] R. C. READ AND R. E. TARJAN, *Bounds on backtrack algorithms for listing cycles, paths, and spanning trees*, *Networks* (to appear); ERL Memo. M 433, Electronics Research Laboratory, Univ. of California, 'Berkeley, 1973.

[29] S. M. ROBERTS AND B. FLORES, *Systematic generation of Hamiltonian circuits*, Comm. ACM, 9 (1966), pp. 690–694.

[30] M. M. SYSLO, *The elementary circuits of a graph*, *Algorithm 459*, Comm. ACM, 16 (1973), pp. 632–633; Errata: Ibid., 18 (1975), pp. 119.

[31] J. L. SZWARCFITER AND P. E. LAUER, *Finding the elementary cycles of a directed graph in $O(n + m)$ per cycle*, no. 60, Univ. of Newcastle upon Tyne, Newcastle upon Tyne, England, May 1974.

[32] R. TARJAN, *Enumeration of the elementary circuits of a directed graph*, this Journal, 2 (1973), pp. 211–216.

[33] J. C. TIERNAN, *An efficient search algorithm to find the elementary circuits of a graph*, Comm. ACM, 13 (1970), pp. 722–726.

[34] H. WEINBLATT, *A new search algorithm for finding the simple cycles of a finite directed graph*, J. Assoc. Comput. Mach., 19 (1972), pp. 43–56.

[35] J. T. WELCH, *Cycle algorithms for undirected linear graphs and some immediate applications*, Proc. 1965 ACM Nat. Conf., P-65, pp. 296–301.

[36] ———, *A mechanical analysis of the cyclic structure of undirected linear graphs*, J. Assoc. Comput. Mach., 13 (1966), pp. 205–210.

[37] S. S. YAU, *Generation of all Hamiltonian circuits, paths, and centers of a graph, and related problems*, IEEE Trans. Circuit Theory, CT-14 (1967), pp. 79–81.