CHAPTER - 4

ALL CIRCUITS OF A NON ORIENTED GRAPH G

4.1. Introduction:

Given an undirected graph, the problem of enumerating all its circuits and cutsets is an interesting problem. Given a graph $G(V, E)$ its circuit and cutset vector sub-spaces contain many members which are edge disjoint union of circuits and cutsets. The problem discussed in this chapter is essentially directed to sieve the sub-spaces as cheaply as possible so that we can only get circuits to play with. The circuit generation algorithms will only be discussed here as cutset generation is a complement problem which can be treated in the same token. The basis of the algorithm consists of generation of a superset of circuits and organise a testing sieve to eliminate unwanted elements. For undirected graphs it is observed that only vector space algorithms are interesting and we have concentrated on them only.

4.2. Preliminaries:

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 $G(V, E)$.

4.2.1. Vector space associated with a graph:

Vector space $V_G$ associated with a graph $G$ consists of the set of all subgraphs of $G$. $V_G$ consists of

1. A Galois field $F$ of modulo 2.
(2) $2^e$ vectors (e tuples).

(3) Addition operation between two vectors is the ring sum operation between the subgraphs.

(4) A scalar multiplication operation between a scalar (from the field $F$) and vectors from $2^e$ vectors.

(5) The coordinate system: Basis is a set of $e$ linearly independent vectors each representing a subgraph of one edge of $G$.

**Definition 4.1:**

The circuit matrix of a non-oriented graph $G$ is a matrix $B$ of order $c \times e$ with entries

$$b_{ij} = \begin{cases} 
1 & \text{if edge } e_j \text{ is in circuit } c_i \\
0 & \text{otherwise}
\end{cases}$$

$c$ being the total number of circuits in $G$. Each row in the matrix $B$ represents a circuit of $G$ and each column an edge of $G$.

**Definition 4.2:**

A circuit vector in $V_G$ represents either a circuit or a union of edge disjoint circuits of $G$.

**Theorem 4.3:**

The ring sum of two circuit vectors in a graph $G$ is either a circuit or an edge disjoint union of circuits.
Proof: Let \( c_1 \) and \( c_2 \) are any two circuits. The vertices in subgraph \( c_1 \oplus c_2 \) are of degrees 2 or 4 and hence it is an Euler graph. Therefore, it consists either a circuit or edge disjoint union of circuits.

Theorem 4.4:

The set of all circuit vectors in \( V_c \) forms a sub-space \( V_c \).

Definition 4.5:

The fundamental circuits (f-circuit) of a connected graph \( G \) for a tree \( T \) are \( e-n+1 \) circuits, each formed by a chord and its unique tree path.

Theorem 4.6:

The set of circuit vectors corresponding to the set of fundamental circuits with respect to any spanning tree, forms a basis for the circuit sub-space \( V_c \).

Proof: (N-1).

Let \( N = e-n+1 \) be the number of chords in a graph \( G \). Thus the circuit vector space contains \( 2^N \) circuit vector including the null vector.

Theorem 4.7:

The circuit sub-space and cutset sub-space in a graph are orthogonal to each other.

Proof: Since the number of edges common between a circuit vector and a cutset vector have even number of edges common, it is obvious.
Corollary:

\[ DC_f^T = 0 \]

where \( D \) is the reduced incidence matrix of dimension \( e \times (n-1) \) and \( C_f \) is the fundamental circuit matrix of dimension \( (e-n+1) \times e \) and the edges are arranged in the same order.

4.3. Algorithm to obtain Fundamental Circuit Matrix from Reduced Incidence Matrix:

4.3.1. Philosophy:

Let \( T = n-1 \) and \( N = e-n+1 \) for a graph \( G \) and the edges of \( D \) and \( C_f \) are arranged in the same order. We write \( D = [D_N : D_T] \) where chords first and branches last matrix partition is done with respect to a spanning tree.

Similarly \( C_f = [I_N : C_T] \).

Now

\[ DC_f^T = 0 = D_N + D_T C_T^T \]

or

\[ C_f = [I_N : (D_T \cdot D_T^T)] \]

The last relation gives a practical method for obtaining fundamental circuit matrix of a graph given a spanning tree and incidence matrix.

Illustration:

The reduced incidence matrix of the graph in Fig. 4.1 is
Obviously $C_f = [I_N : (I_T^{-1} B_N)^t] = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$

Algorithm 4.3.2:

This algorithm generates f-circuits of a graph $G$. Input to the algorithm is incidence matrix.

Step 1: Generate a spanning tree and permute the incidence matrix in chords first branches last order.

Step 2: By elementary transformation, the branch submatrix of incidence matrix is diagonalised. The operations are all mod 2.

Step 3: Transpose the chord submatrix and form f-circuit matrix with the Unity matrix to obtain the chord submatrix.

Alternatively, we can generate f-circuit as a by product in the DFS procedure for a spanning tree generation. However, in this chapter we are mainly concerned with the enumeration of all circuits of a graph $G$. 
4.4. Circuit Vector Space of Undirected Graph:

All circuit vector space algorithms known so far compute all of the $2^N$ vectors and then tests for containment to reject the edge disjoint union of circuits. But there are classes of graphs for which the ratio of the number of circuits $C$ to vectors approaches zero for large $N$. In fact there are only four graphs with $C = 2^N - 1$. These four graphs are shown in Fig. 4.2.

In general, a large fraction of the circuit vectors are edge disjoint union of circuits. The ratio of circuits to all vectors in the vector space for numerous classes of graphs tends to zero. Three such class are shown in Fig. 4.3 (P-5).

(a) A wheel graph:

$$N = e - n + 1$$

$$e = \frac{(n-1)^3 + (n-1)}{2} = 2(n-1)$$

$$N = 2n - 2 - n + 1 = n - 1$$

Number of circuit vectors $= 2^N - 1$

Number of circuits $C = N(N-1) + 1$

(b) A modified ladder graph:

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

$$C = \frac{n^2}{2} + \frac{5N}{2} - 3$$
(76)

(c) A complete graph:

\[ N = \frac{n^2}{2} - \frac{3n}{2} + 1 \]

\[ C = \frac{1}{2} \sum_{S=3}^{n} \binom{n}{S} (S-1)! \]

4.5. Circuit Vector Space Algorithms:

The reasonable attempt to improve upon all circuit generation must be composed of the following steps.

1) Store only the f-circuits.

2) Recognise apriori which vectors are edge disjoint union of circuits.
   If not possible, generate as small number as possible the non-solutions.

3) Testing sieve should be as simple as possible.

4) Circuit vector generation must take as little time as possible.

4.5.1. An algorithm for all circuit generation: (B-4, J-8)

All circuits can be found by brute-force techniques which will take \( 2^N - N - 1 \) linear combinations of \( N \) fundamental circuits, store them and compare pairwise for containment so that edge disjoint union of circuits are recognised. The storage requirement in these algorithms is proportional to \( e.2^N \) words. Apart from this, the linear combinations are obtained by ring sum of 2 to \( N \) f-circuit vectors which requires \( O(N.2^{N-1}) \) exclusive-or operations. The operation of exclusive-or is inefficient in high level programming languages.
In the algorithm presented here, linear combinations of \( f \)-circuit vectors are so generated that the exclusive-or operations are reduced to \( 2/3 \)th times with very little increase in computational overhead. The ring sum is always taken between two circuit vectors and a three level testing sieve rejects edge disjoint union of circuits in first or second level in majority of the cases. The third level of the testing sieve is most time consuming.

4.5.2. The algorithm:

The algorithm operates on the adjacency matrix of the graph and outputs circuits one at a time.

Step 1: \( f \)-circuit vectors are generated by Algorithm 4.3.2.

Step 2: Unit distance \( N \) (nullity of the graph) dimensional codes are generated one at a time and corresponding circuit vectors are generated by setting appropriate pointers to the \( f \)-circuit vectors (if the codeword contains two ones only). Since two consecutive codes differ by one distance, the next sequential circuit vector is generated by mod-2 sum of the \( f \)-circuit vector with the already generated one. (This \( f \)-circuit corresponds to the position where two consecutive codes differ).

Step 3: Test whether the parents of the circuit vectors are mutually exclusive or not. If yes, do not output the vector and go to Step 2.

Step 4: Test whether in the subgraph containing the edges of the circuit vector has any node of degree more than two. If yes, do not output the circuit vector and go to Step 2.
Step 5: Next, the circuit vector is tested for connectivity. It is finally identified in this step as a circuit or an edge disjoint union of circuits. In the former case go to Step 5 otherwise go to Step 2.

Step 6: Go to Step 2 after outputing the circuit vector until all circuits are generated, i.e., the codes recycle.

Elaboration of Step 2: (E.4)

The unit distance codes are known as Gray codes. These are $2^N$, N-bit strings such that successive code words differ by the complementation of a single bit. A Gray code can be conveniently represented by its transition sequence that is the ordered list of the bit positions numbered from right to left that change as we go from one code word to the next. The transition sequence can be conveniently generated by using a stack operated as follows.

The stack initially contains $N, N-1 ... 1$ (with 1 on the top). The algorithm pops off the top element $i$ and puts it into the sequence, the elements $i-1, i-2 ... 1$ are then pushed onto the stack. By slight modification, the transition sequence can be generated at constant time.

Elaboration of Step 5:

The circuit vector with all nodes of degree 2 needs to be tested for connectivity to ascertain whether it is a union of edge disjoint circuits or not. The testing is done by extracting the incidence matrix of the subgraph corresponding to a circuit vector. If starting from any vertex in the subgraph
we can return to it without traversing every edge, the circuit vector is an edge disjoint union of circuits.

4.5.3. Illustration:

The f-circuit matrix of the graph in Fig. 4.4 is

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{bmatrix}
\]

Here \( N = 4 \). The transition sequence generated is:

1 2 1 3 1 2 1 4 2 1 3 1 2 1

The gray codes are 0 0 0 0, 0 0 0 1, 0 0 1 1, 0 0 1 0, 0 1 1 0,

7 5 4 12 13
0 1 1 1, 0 1 0 1, 0 1 0 0, 1 1 0 0, 1 1 0 1,

15 14 10 11 9 8
1 1 1 1, 1 1 1 0, 1 0 1 0, 1 0 1 1, 1 0 0 1, 1 0 0 0,

where decimal numbers correspond to the decimal values of the codes.

Let us consider two circuit vectors

0 1 1 0 1 1 0 1 0
0 1 1 1 1 0 1 1

The second circuit vector is obtained by the ring sum of fourth f-circuit with the immediate previously generated circuit vector 0 1 1 0. The transition vector helps us to generate all circuit vectors by mod-2 sum of two circuit vectors only.
4.6. Analysis of the Algorithm:

Let us consider a graph of $e$ edges and nullity $N$. Generally we obtain all linear combinations of $N$ f-circuits and ring sum them. This will involve approximately

$$e \sum_{k=1}^{N} \binom{N}{k} = eN 2^{N-1}$$

exclusive or operations.

However, in the proposed method we ring sum always two vector at a time. Thus the number of exclusive or operations is roughly $e2^N$ only. Thus the time is reduced by $2/N$ times.

The storage requirement is that required to store the fundamental circuits only.

This algorithm is programmed in Fortran IV and tested in IBM 1130 for random graphs. The computation involved in connectivity testing is proportional to the number of vertices of the graph. The most interesting feature is that 50% of the nonsolutions were rejected before connectivity testing. The angle of attack in this algorithm may be fruitful in all the problems with exponential number of members in the solution space.
Computer Programs
and
Sample Results
***MAIN ROUTINE***
***A IS THE ADJACENCY MATRIX***
***NV = NO OF VERTICES, NE = NO OF EDGES***
***NC = NO OF CHORD BRANCHES***
***NT = NO OF TREE BRANCHES***
***KK = NO OF COMBINATIONS, L = NO OF TIMES***
***CONNECTIVITY TESTING HAS TO BE DONE***
***ICC = NO OF CIRCUITS (EXCEPT F-CI-CUITS)***
***IND = MEANS EDGE DISJOINT UNIG. OF CIRCUITS***

INTEGER A(20,50), BF(30,5D), P(30), C(30), S(5C)

DO 20 I=1, 5
READ(2,5) NV, NE, NC, NT, ((A(I,J), J=1, NE), I=1, NT)
READ(2,5) ((BF(I,J), J=1, NE), I=1, NC)

KK=0
L=:
ICC=:

WRITE(5,6) NV, NE, NC, NT
WRITE(5,44)
DO 10 I=1, NT
  WRITE(5,9) A(I,J), J=1, NE
10  WRITE(5,47)
DO 11 I=1, NC
11  WRITE(5,9) BF(I,J), J=1, NE
DO 21 KN=2,NC
DO 13 I=1,NC
C(I)=J
P(I)=I
I=I+1
C(I)=P(I)
IF(I-KN) 14,15,14
P(I+1)=P(I)+1
IF(P(I+1)-NC) 17,17,16
I=I+1
GO TO 12
CALL CIR(A,DF,S,IND,C,NN,NV,NF,NT,L)
KK=KK+1
IF(IND) 20,19,20
WRITE(5,9) (S(Il),Il=1,NE)
ICC=ICC+1
P(I)=P(I)+1
IF(P(I)-NC) 12,12,16
I=I-1
IF(I) 19,21,19
CONTINUE
WRITE(5,28) KK,L,ICC
FORMAT(1H1,///20X,' NV=',I4,' NE=',I4,' NN=',I4,' NV=',I4,' NF=',I4,' NT=',I4)
FORMAT(/20X,' CIRCUIT MATRIX IS/')
PAGE NO- 3

47   FORMAT(/20X, ' F-CIRCUIT MATRIX IS')
200  CONTINUE
44   FORMAT(/20X,' REDUCED INCIDENCE MATRIX IS')
9    FORMAT(20X,5(I1,1X))
28   FORMAT(/20X,'KK=',14,' L=',I4,' ICC=',I4)
5    FORMAT(4I2,7I1)
STOP
END

SUBROUTINE CI(A,EF,S,IND,CC,NN,NV,NE,NT,IC)

C ***NT=NV-1,S GIVES RING-SUM OF F-CIRCUITS***
C ***T GIVES SUCCESSOR LISTING ***
C ***OF THE SUBGRAPH FORMED BY EDGES IN S***
INTEGER T(2,3),A(20,50),BF(30,50),CC(30),S(50)
DO 11 K=1,NE
    S(K)=0
10   S(K)=S(K)+BF(I,K)
    S(K)=S(K)-(S(K)/2)*2
    DO 32 I=1,NV
        DO 32 J=1,3
            T(I,J)=0
        DO 50 1 = 1,NE
            S(K)=S(K)+BF(I,K)
            S(K)=S(K) -(S(K)/2)*2
            DO 32 I=1,NV
            DO 32 J=1,3
        T(I,J)=0
        DO 50 I=1,NE
STOP
END
IF(S(I)) 12,57,12

12    K=C

13    I=1,NT

14    IF(A(J,1)) 13,15,13

15    K = J

16    CONTINUE

17    IF(L) 17,16,17

18    L=J

19    IF(T(K,3)-2) 19,23,28

20    M=T(K,3)+1

21    CONTINUE

22    IC=IC+1

23    I=I+1

GO TO 21
PAGE NC-5

22   J=1
25   K=T(I, J)
30   T(I, 3)=5
124  IF(T(K, 3)) 124, 27, 124
124   J=1
124  IF(T(K, J)-1) 26, 30, 26
36   J=2
26   I=6
26   GO TO 25
27   DO 36 I=2, NY
36   CONTINUE
36   CONTINUE
30   IND=1
30   GO TO 131
28   IND=1
131  RETURN

NV = 8  NE = 12  NC = 5  AT = 7

REDUCED INCIDENCE MATRIX IS

```
1 0 1 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 1 1 1 0 1 1 1
0 0 0 0 0 1 1 1 0 1 0 0
0 0 1 0 0 0 0 0 0 0 1 0
0 0 0 1 0 1 0 0 0 0 0 0
```

F-CIRCUIT MATRIX IS

```
0 1 1 0 0 0 0 1 0 1 0 0
0 1 0 1 1 1 0 0 1 0 0 0
0 1 0 1 1 1 0 0 1 0 0 0
0 0 1 1 1 1 0 0 1 0 0 1
0 0 1 1 1 0 0 0 0 0 1 1
```

CIRCUIT MATRIX IS

```
0 0 1 1 1 1 0 1 1 0 0 0
0 1 1 1 1 1 1 0 1 0 0 0
0 1 0 1 1 0 1 0 1 0 0 0
0 0 0 0 0 1 1 0 0 1 0 0
0 1 1 0 0 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 1 0 0 1
0 1 1 0 1 1 0 0 0 0 0 1
0 1 0 1 1 0 0 1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0
0 1 0 0 0 0 0 1 0 1 1 1
0 0 1 1 1 1 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 0 1 1
```

KK = 26  L = 19  ICC = 15
Fig. 4.1 A Graph G with tree \((a, b, d, g)\)

Fig. 4.3 Graphs for which \(\lim_{n \to \infty} \left(\frac{C}{2^n}\right) = 0\)
Fig. 4.3 Graphs all of whose circuit vectors are circuits.

Fig. 4.4 Graph $G$ with tree $(a, b, d, e, h)$