Journal of the Nigerian Association of Mathematical Physics
Volume 10 (November 2006), **97 -106**

An application of the maximal independent set algorithm to course allocation

V. V. N. Akwukwuma** and ***K. C. Ukaoha,
Department of Computer Science, University of Benin, Benin City.
***e-mail: vakwukwuma@yahoo.com**

Abstract

In this paper, we demonstrated one of the many applications of the Maximal Independent Set Algorithm in the area of course allocation. A program was developed in Pascal and used in implementing a modified version of the algorithm to assign teaching courses to available lecturers in any academic environment and it proved to be very effective.

***Keywords*:** maximal independent sets, graphs, course allocation, bipartite graphs.

1.0 Introduction

A graph is fundamentally a combinatorial object. That is, a set of points (or vertices) and a particular set of connecting lines (or edges) out of all possible sets of such lines. A directed graph is a finite nonempty set V and a set E of ordered pairs of distinct elements of V; the elements of V are called vertices and the elements of E are called directed edges. A bipartite graph is a graph in which the vertices can be divided into two disjoint nonempty sets A and B such that no two vertices in A are adjacent and no two vertices in B are adjacent. An independent set of a graph G is a subset of the vertices such that no two vertices in the subset are connected by an edge of G. That is, an independent set is a set of mutually non-adjacent vertices Halldorsson (2000). A Maximal Independent Set (MIS) in an undirected graph is a maximal collection of vertices I subject to the restriction that no pair of vertices in I are adjacent Luby (1985). Graphs are applied in a variety of simple, to complicated tasks and various studies have been carried out on their applications Ahuja et al (1993), Halperin (2000), Israeli (1984), Lev (1980) and Miller et al (2001). The MIS algorithm was proposed by Ford and Fulkerson (1962) and could be applied without any complications to bipartite graphs.
In this paper, we slightly modified the MIS algorithm and subsequently applied it to the allocation of teaching courses to available lecturers in an academic setting. The original MIS algorithm could not place a limit as to the number of courses a lecturer could teach while the modified version of the MIS algorithm places a limit to the number of courses which can be taught by any lecturer. We coded the modified version of the MIS algorithm Turbo Pascal which when executed, was faster, more efficient and more reliable than the traditional manual method of allocating courses in academic environments.

2.0 The Algorithm

Let G be a graph with vertices V and edges E. by a matching of G, we mean a subset M of E such that no vertex in V is incident with more than one edge in M. By a maximal matching of G, we mean a matching of G so that no other matching of G contains more edges. We say a graph with vertex set V and edge set E is bipartite in case V,E can be written as the union of two disjoint sets V_1 and V_2 such that each edge joins an element of V_1 with an element of V_2. By a covering C of a graph, we mean a set of vertices such that every edge is incident to at least one vertex in C. We say C is a minimal covering if no covering of the graph has fewer vertices.
The MIS algorithm is set out as follows;
:: Matrix representation of bipartite graph.
Given: an independent set of 1's in a matrix of 0's and 1's,

Begin:

Algorithm: Max_Indep_Set

1: **Label** with an ‘L’ all columns containing no starred 1’s (i.e. 1*)

2: **IF**<all labeled columns have been scanned> **THEN GOTO** Step 4
  OTHERWISE
  <Scan column and put ‘S’ under each scanned column>

3: **IF**<all labeled rows have been scanned> **THEN GOTO** Step 4
  OTHERWISE
  IF<some labeled but unscanned row contains no starred ‘1’>
  THEN GOTO Step 5
  OTHERWISE
  <Scan row and put ‘S’ after each scanned row> and **GOTO** Step 2

4: **STOP**

5: **DO** {a labeled row contains no starred 1}
  a. Circle the 1 in this row and in the column that row is labeled with
  b. Circle the starred 1 in this column and the row that the column is labeled with.
  UNTIL <a ‘1’ is circled in a column labeled with ‘L’>

6: **Reverse** the stars on all circled 1’s. {This gives an independent set of 1’s with one more element than the original set}.

In practical applications, the matrix for the MIS algorithm is derived by entering a ‘1’ whenever the vertices corresponding to the row and column are joined by an edge and a ‘0’ otherwise. A set of entries in such a matrix is independent if no two of them are in the same line while an independent set of 1’s in the matrix is a maximal independent set of 1’s if no independent set of 1’s in such a matrix contains more elements. In the above algorithm, we marked the 1’s in a particular independent set with stars.

Suppose a matrix of 0,s and 1’s has m rows and n columns. Step 1 involves looking at all mn entries in the matrix, which we count as mn operations. After this, the algorithm alternates between Steps 2 and 3, both of which involve scanning. In order to scan one of the n columns we need to look at the m entries in that column, so all column scanning will take at most mn operations. Similarly, row scanning will take at most nm operations. If we go to Step 4 we are done, so we analyze Steps 5 and 6. Backtracking will take at most $m + n$ operations, since each 1 we circle can be associated with a distinct row or column. Actually, we could combine Step 6 into Step 5 with no additional work, reversing the stars as we backtracked. Thus, one application of the algorithm will take at most $3m + m + n$ operations. To build up to a maximal independent set of 1’s the algorithm will have to be repeated at most $\min\{m,n\}$ times, even if we start with the empty set as our first independent set of 1’s. Thus, the complexity of the algorithm for finding a maximal independent set of 1’s in an m by n matrix is of order no more than $(3mn + m + n)\min\{m,n\}$. For the case of $m = n = 30$, fewer than 90,000 operations would be necessary, and a fast computer could do the problem in less than one second.

3.0 Application

In any academic environment, teaching courses are assigned to the available teaching staff based on their area of specialization or subject area. Since the available courses are equally grouped the same way as the subject areas of the lecturers; our MIS algorithm is applied as follows;

i. Select a particular subject area and bring out the list of lecturers available in that area
ii. For any match in the subjects and the lecturers, enter a ‘1’ and a ‘0’ otherwise
iii. Apply the MIS algorithm t o get the appropriate matching
iv. Move over to another subject area and again apply the algorithm provided no lecturer is assigned more than two courses per semester
iv. Continue applying the algorithm until all the courses have been allocated.
v. Stop

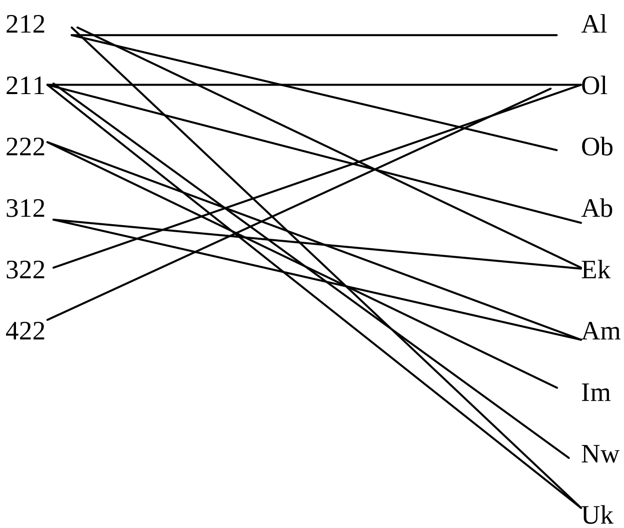
In one of the areas of specialization, PROGRAMMING LANGUAGES, we have the courses listed and a corresponding list of lecturers that can teach some of these courses as follows;

LIST OF COURSES
CSC212 Symbolic Programming in FORTRAN
CSC211 Programming in Pascal
CSC222 Assembly Language Programming
CSC312 Advanced Assembly Language and C++ Programming
CSC322 Data Structures
CSC422 Concepts of Programming Languages

LIST OF LECTURERS THAT CAN TEACH SOME OF THESE COURSES;

Al	CSC212
Ol	CSC211, CSC322 and CSC422
Ob	CSC212
Ab	CSC211
Ek	CSC212 and CSC312
Am	CSC222 and CSC312
Im	CSC222
Nw	CSC211
Uk	CSC211, CSC212, CSC312 and CSC322

Arranging the courses (using their numeric codes) side by side with the lecturers that can teach each course gives us the following bipartite graph;



By entering a '1' against any corresponding lecturer and course and a '0' otherwise, we have the following matrix representation of the graph;

	Al	**Ol**	**Ob**	**Ab**	**Ek**	**Am**	**Im**	**Nw**	**Uk**
211	0	1	0	1	0	0	0	1	1
212	1	0	1	0	1	0	0	0	1
222	0	0	0	0	0	1	1	0	0
312	0	0	0	0	1	1	0	0	1
322	0	1	0	0	0	0	0	0	1
422	0	1	0	0	0	0	0	0	0

The row headers are used to denote the courses to be taught while the columns header denote the available lecturers that will teach a course, after which we place an asterisk or star on the first available '1' on each column, provided such a '1' is the only one starred on each column and row where it appears. This gives us the following;

	Al	Ol	Ob	Ab	Ek	Am	Im	Nw	Uk	
211	0	1*	0	1	0	0	0	1	1	
212	1*	0	1	0	1	0	0	0	1	
222	0	0	0	0	0	1*	1	0	0	
312	0	0	0	0	1*	1	0	0	1	
322	0	1	0	0	0	0	0	0	1*	
422	0	1	0	0	0	0	0	0	0	
			L	L			L	L		

The following source code was developed in Turbo Pascal to handle the maximal independent set operation and the above matrix was entered as input.

{MAXIMAL INDEPENDENT SET ALGORITHM}

```
PROGRAM Algol(input, output, Infile);
CONST
  RowSize = 50;
  ColSize = 50;

TYPE
  ElementType = RECORD
                Value  :  char;
                Status :  char;
               END;
  RCStatus    = RECORD
                Labelled   : boolean;
                IValue     : integer;
                CValue     : char;
                Scanned    : boolean;
               END;
  BoolRowArry = ARRAY [1 .. RowSize] OF boolean;
  MatArray = ARRAY [1 .. RowSize, 1 .. ColSize] OF ElementType;
  BoolDim2 = ARRAY [1 .. RowSize, 1 .. ColSize] OF boolean;
  ColArray = ARRAY [1 .. ColSize] OF RCStatus;
  RowArray = ARRAY [1 .. RowSize] OF RCStatus;

VAR
  Matrix                 : MatArray;
  Circle                 : BoolDim2;
  ColStatus              : ColArray;
  RowStatus              : RowArray;
  UnstarredRow           : BoolRowArry;
  Starred, LabColStatus,
  LabRowStatus, Continue : boolean;
  M, N                   : integer;
  Infile, output         : text;

{THIS PROCEDURE READS THE ARRAY ELEMENTS }

PROCEDURE GetInput(VAR Mat: MatArray);
  VAR
    I, J : integer;
    Ch   : char;
  BEGIN
```

```
   write('Enter the number of rows: ');

   readln(M);
   write('Enter the number of columns: ');
   readln(N);
   FOR I := 1 TO M DO
     BEGIN
       FOR J := 1 TO N DO
         BEGIN
           WITH Mat[I,J] DO
              read(Infile, Value, Status);
           read(Infile, Ch)
         END;
       readln(Infile, Ch)
     END;
 END;

PROCEDURE Initialize(VAR Circ: BoolDim2; VAR CStatus: ColArray;
                    VAR RStatus: RowArray; VAR USRow: BoolRowArry);
 VAR
   I,J : integer;
 BEGIN
   FOR I := 1 TO M DO
     FOR J := 1 TO N DO
         Circ[i,j] := False;

  FOR J := 1 TO N DO
    WITH CStatus[J] DO
      BEGIN
        Labelled := False;
        IValue := 0;
        CValue := ' ';
        Scanned := False
      END;

  FOR I := 1 TO M DO
   WITH RStatus[I] DO
     BEGIN
       Labelled := False;
       IValue := 0;
       CValue := ' ';
       Scanned := False
     END;
   FOR I := 1 TO M DO
     USRow[i] := False;
   Starred := False
 END;

{STEP1 LABEL WITH AN "L" ALL COLUMNS CONTAINING NO STARRED 1}
PROCEDURE Step1(VAR CStatus: ColArray; Mat: MatArray);
 VAR
   I,J  : integer;
   LStatus : boolean;

 BEGIN
   FOR I := 1 TO N DO
```

```
    BEGIN

      LStatus := True;
      FOR J := 1 TO M DO
        IF (Mat[J,I].Value = '1') AND (Mat[J,I].Status = '*') THEN
          LStatus := False;
      IF LStatus THEN
        BEGIN
          CStatus[I].Labelled := True;
          CStatus[I].CValue := 'L'
        END
      ELSE
        CStatus[I].Labelled := False
    END
  END;
```

```
{STEP2: SCANNING COLUMNS - IF ALL LABELED COLUMNS HAVE BEEN SCANNED,
THEN GOTO STEP4 ELSE, FOR EACH COLUMN THAT IS LABELED BUT NOT SCANNED LOOK AT
ANY UNSTARRED 1'S IN THAT COLUMN. IF SUCH A 1 IS IN AN UNLABELLED ROW, THEN LABEL
THAT ROW WITH THE NAME OF THE COLUMN BEING SCANNED. PUT THE LETTER "S" UNDER
EACH COLUMN AFTER IT HAS BEEN SCANNED. }

PROCEDURE Step2(VAR CStatus : ColArray; VAR RStatus: RowArray;
          VAR LCStatus : boolean; Mat: MatArray);
  VAR
    I,J : integer;

  BEGIN
    LCStatus := True;
    FOR I := 1 TO N DO
      IF (CStatus[I].Labelled = True) AND (CStatus[I].Scanned = False) THEN
        BEGIN
          LCStatus := False;
          FOR J := 1 TO M DO
            IF (Mat[J,I].Value = '1') AND (Mat[J,I].Status <> '*') THEN
              IF (RStatus[J].Labelled = False) THEN
                BEGIN
                  RStatus[J].Labelled := True;
                  RStatus[J].IValue := I
                END;
          CStatus[I].Scanned := True
        END
  END;

{SCANNING ROWS -  IF ALL LABELED ROWS HAVE BEEN SCANNED, THEN GOTO STEP4.
IF SOME LABELED BUT UNSCANNED ROW CONTAINS NO STARRED 1, THEN GOTO STEP5
ELSE FOR EACH ROW THAT IS LABELED BUT NOT SCANNED, LOOK FOR THE STARRED 1 IN THAT
ROW. LABEL THE COLUMN CONTAINING THE STARRED 1 WITH THE NAME OF THE ROW BEING
SCANNED. PUT THE LETTER "S" AFTER EACH ROW WHEN IT HAS BEEN SCANNED.
GOTO STEP 2  }

PROCEDURE Step3(VAR CStatus : ColArray; VAR RStatus: RowArray;
          VAR LRStatus, Strred: boolean; VAR USRow: BoolRowArry;
            Mat: MatArray);
  VAR
    I,J        : integer;
```

```
  TempStarred : boolean;
 BEGIN
  LRStatus := True;
  Strred := True;
  FOR I := 1 TO M DO
   IF (RStatus[I].Labelled = True) AND (RStatus[I].Scanned = False) THEN
     BEGIN
      LRStatus := False;
      TempStarred := False;
      FOR J := 1 TO N DO
       IF (Mat[I,J].Value = '1') AND (Mat[I,J].Status = '*') THEN
        TempStarred := True;
      IF NOT TempStarred THEN
       BEGIN
        Strred := False;
        USRow[I] := True;
       END
     END;
   IF Strred THEN
    BEGIN
     FOR I := 1 TO M DO
      IF (RStatus[I].Labelled = True) AND (RStatus[I].Scanned = False) THEN
       BEGIN
        FOR J := 1 TO N DO
         IF (Mat[I,J].Value = '1') AND (Mat[I,J].Status = '*') THEN
          BEGIN
           CStatus[J].Labelled := True;
           CStatus[J].IValue := I
          END;
        RStatus[I].Scanned := True
       END
    END
 END;

{STEP4 NO IMPROVEMENT - STOP. THE GIVEN INDEPENDENT SET IS MAXIMAL.}

PROCEDURE Step4;
 BEGIN
  writeln(output,'The given independent set is maximal.')
 END;

{STEP5 - BACKTRACKING - A LABELED ROW CONTAINS NO STARRED 1. CIRCLE THE 1 IN THIS
ROW AND IN THE COLUMN THAT THE ROW IS LABELED WITH. CIRCLE THE STARRED 1 IN THIS
COLUMN AND THE ROW THAT THIS COLUMN IS LABELED WITH. CONTINUE IN THIS WAY UNTIL A
1 IS CIRCLED IN A COLUMN LABELED WITH THE LETTER "L"}

PROCEDURE Step5(VAR Circ: BoolDim2; USRow: BoolRowArry; RStatus: RowArray;
          CStatus: ColArray);
 VAR
  I,J,K,R,C : integer;
 BEGIN
  FOR I := 1 TO M DO
   IF USRow[I] THEN
    BEGIN
     K := I;
```

```
    REPEAT
     J := K;
     C := RStatus[J].IValue;
     Circ[J,C] := True;
     R := CStatus[RStatus[J].IValue].IValue;
     IF R <> 0 THEN
       Circ[R,C] := True;
     K := R;
    UNTIL (CStatus[C].CValue = 'L')
   END
 END;

{STEP6 LARGER INDEPENDENT SET - REVERSE THE STARS ON ALL CIRCLED 1'S
THIS GIVES AN INDEPENDENT SET OF 1'S WITH ONE MORE ELEMENT THATN THE
ORIGINAL SET.}

PROCEDURE Step6(VAR Mat: MatArray; Circ: BoolDim2);
 VAR
  I, J : integer;
 BEGIN
  FOR I := 1 TO M DO
   FOR J := 1 TO N DO
    IF Circ[I,J] = True THEN
     IF Mat[I,J].Status = '*' THEN
      Mat[I,J].Status := ' '
     ELSE
      Mat[I,J].Status := '*'
 END;

PROCEDURE Output1(Mat: MatArray);
 VAR
  I, J : integer;

 BEGIN

  FOR I := 1 TO M DO
   BEGIN
    FOR J := 1 TO N-1 DO
     write(output,Mat[I,J].Value:1, Mat[I,J].Status:1, ' ');
     writeln(output);
    writeln(output,Mat[I,N].Value:1, Mat[I,N].Status:1)
   END
 END;

BEGIN {MAIN PROGRAM STARTS HERE}
 assign(Infile, 'KINGS.INP');
 reset(Infile);
 writeln; writeln;
 GetInput(Matrix);
 assign(output, 'KINGS.OUT');
 rewrite(output);
 writeln(output); writeln(output);
 writeln(output,'The initial matrix set is: ');
 Output1(Matrix);
 Initialize(Circle,ColStatus,RowStatus,UnstarredRow);
```

```
  Step1(ColStatus, Matrix);
  LabColStatus := False;
  LabRowStatus := False;
  Continue := True;
  WHILE Continue DO
    BEGIN
      Step2(ColStatus, RowStatus, LabColStatus, Matrix);
      IF LabColStatus THEN
        Continue := False
      ELSE
        BEGIN
          Step3(ColStatus,RowStatus,LabRowStatus,Starred,UnstarredRow,Matrix);
          IF (LabRowStatus) OR (NOT Starred) THEN
            Continue := False
        END
    END;
  IF NOT Starred THEN
    BEGIN
      Step5(Circle,UnstarredRow,RowStatus,ColStatus);
      Step6(Matrix,Circle);
      writeln(output); writeln(output);
      writeln(output,'The larger independent set is: ');
      writeln(output);
      writeln(output);
      Output1(Matrix);
    END;
  IF LabColStatus OR LabRowStatus THEN
    Step4;
  close(Infile);
  close(output)
END.
```

The output produced from the above program when our original matrix was entered as input is given below;

	Al	**Ol**	**Ob**	**Ab**	**Ek**	**Am**	**Im**	**Nw**	**Uk**
211	0	1	0	1*	0	0	0	1	1
212	1*	0	1	0	1	0	0	0	1
222	0	0	0	0	0	1*	1	0	0
312	0	0	0	0	1*	1	0	0	1
322	0	1	0	0	0	0	0	0	1*
422	0	1*	0	0	0	0	0	0	0

From the result matrix above, we observed that the course allocations on execution of the source codes were;

CSC211: Ab
CSC212: Al
CSC222: Am
CSC312: Ek
CSC322: Uk
CSC422: Ol

We could then proceed to apply the matrix to other subject areas in order to produce an independent set for each area, but the condition that no Lecturer is involved in more than two courses per semester still holds.

2.0 Conclusion

As could be seen from our case study, the MIS algorithm can be applied to a variety of tasks which involves scheduling and planning. Our modified version of the algorithm was applied to course scheduling and course allocation in an academic environment. Its overall advantages amongst others were that; it made the task of allocation of courses faster, easier and convenient and more importantly, it ensures that no lecturer is allocated more than the maximum number of courses to be allocated i.e. if the maximum number of courses already agreed to be allocated per semester or term is 2 to any lecturer, the algorithm ensures that such rule is not violated.

We equally developed a source code in Turbo Pascal which, apart from being fast and efficient, could handle extremely larger input lists like allocation of courses to lecturers in a Faculty, a Polytechnic, a Secondary or High School, or even a University. Another noticeable advantage of the source code is that, it is impartial and can handle more-complex cases. We do hope that more research work would be carried out on other application areas of the maximal independent set algorithm.

References

[1] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. (1993). Network Flows: Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs, NJ.

[2] Dossey, J.A., Oho, A.D., Spence, L.E. and Eynden, C.V. (1987). Discrete Mathematics. Scott Foresman and Company, Illinois.

[3] Ford, L.R. and Fulkerson, D.R. (1962). Flows in Networks. University Press, Princeton, NJ.

[4] Halldorsson, M.M. (2000). Approximations of Weighted Independent Set and Hereditary Subset Problems. Journal of Graph Algorithms and Applications, vol.4, no.1, pp.1-16.

[5] Halperin, E. (2000). Improved Approximation Algorithm for the Vertex Cover Problem in Graphs and Hypergraphs. In Proc. Eleventh ACM-SIAM Symp. On Discrete Algorithms, pp. 329-337.

[6] Israeli, A. and Shiloach, Y. (1984). An Improved Maximal Matching Parallel Algorithm. Tech. Rep. 333, Computer Science Department, Technion, Haifa, Israel.

[7] Lev, G. (1980). Size Bounds and Parallel Algorithms for Networks. Report CSCT-8-80, Department of Computer Science, University of Edinburgh.

[8] Luby, M. (1985). Simple Parallel Algorithm for the Maximal Independent Set Problem. Journal of ACM.

[9] Miller, H.J. and Shaw, S.L. (2001). Geographic Information Systems for Transportation: Principles and Applications. Oxford University Press, Oxford.

[10] Karp, R.M. and Widgerson, A. (1984). A Fast parallel Algorithm for the Maximal Independent Set Problem. Proceedings of 16th ACM STOC, pp. 266-272