An Algorithm for Finding Maximum Independent Set in a Graph

European Journal of Scientific Research
ISSN 1450-216X Vol.23 No.4 (2008), pp.586-596
© EuroJournals Publishing, Inc. 2008
http://www.eurojournals.com/ejsr.htm

An Algorithm for Finding Maximum Independent Set in a Graph

Ahmad Sharieh
King Abdullah II School for IT, Computer Science Department
The University of Jordan, Amman 11942 Jordan
E-mail: sharieh@ju.edu.jo

Wagdi Al_Rawagepfeh
Al Isra Private University, Amman-Jordan
E-mail: alrawagfeh@yahoo.com

Mohammed H. Mahafzah
Faculty of Information Technology, Philadelphia University
Amman, Jordan
E-mail: mohammedmahafzah@yahoo.com

Ayman Al Dahamsheh
Ministry of Education in Jordan, Amman Jordan
E-mail: ayman_dahamsheh@yahoo.com

Abstract

The Maximum Independent Set (MIS) in a graph has important applications and needs exact algorithm to find it. The execution time complexity of the available exact algorithms to find the MIS tend to be an exponential growth function. One algorithm of the exact algorithms is the Modified Wilf's (MW) algorithm. It can not handle graphs with large sizes. This paper introduces a new algorithm, named FMIS, to find a MIS set with less run time complexity and close to $O(n^4)$, for a graph of n nodes. The run time complexity of the MW is $O(1.39^n)$, while the complexity of the FMIS algorithm is $O(1.0052^n)$. The Complexity of an algorithm proposed by R. Beigel is $O(2^{0.29n})$, for general graphs, while the complexity of the FMIS algorithm tends to be $O(2^{0.0076n})$. The FMIS algorithm was implemented and tested on graphs with different sizes and densities. It was compared with the MW algorithm in terms of CPU run time and size of graphs with different densities. The results indicate that the proposed algorithm is better than other known exact algorithms and can handle graphs with larger sizes.

Keywords: Graph Density, Independence Number, Maximum Clique, Maximum Independent Set, Modified Wilf Algorithm.

1. Introduction

An undirected graph $G = (V, E)$ is a pair of finite sets, where the set V contains the vertices of the graph and the set E contains its distinct unordered edges [5, 12, 13]. Two vertices of a graph are considered adjacent (connected by an edge) if an edge exits between them. An Independent Set (*I)* is a subset of vertices in *G* such that no two vertices in *I* are adjacent. The Maximum Independent Set (MIS) of *G* is an *I* with maximum cardinality among all *I* sets of *G*. The cardinality of a MIS is called independence number of *G* and is denoted as $\alpha(G)$ [1]. The MIS problem is to find an *I* with the largest number of vertices in a given *G*.

For a graph *G*, a clique is a subset *C* of vertices in *G* such that each pair of vertices in *C* is connected by an edge. A clique is an independent set in the complementary graph. The Maximum Clique (*MC*) problem is one of the first shown to be NP-hard. The MIS problem and the MC problem are essentially equivalent and have been extensively studied in graph theory and combinational optimization [13].

The problem of finding a MIS of a graph is important for applications in Computer Science, Operation Research, and Engineering [2]. There are many applications of the MIS such as graph coloring, scheduling final exam in a university, assigning channels to radio station, register allocation in a compiler, and the reader collision problem [5].

The problem of finding the MIS of a graph has received much attention [2, 9]. In [1], they presented two continuous multivariable polynomial formulations to find an independent set. Buuce discussed how to find the maximum number of MIS in graphs with *n* vertices with at most *r* cycles and connected graphs [3]. Timothy discussed the problem of finding the MIS in the intersection graph of an axis-parallel rectangles [10].

In a graph *G*, the number of nodes (vertices) is denoted as $|V|$ and the number of un-ordered edges are $|E|$. Note that $|E| < |V|$. For a graph of n $= |V|$, the running-time bound for the Modified Wilf's (MW) algorithm is $O(1.39^{n})$ [8]. Beigel proposed an algorithm to find the MIS in time $O(2^{0.29n})$ for general graph [9]. Those exact algorithms to find a MIS- with similar exponential order- will not be able to find the MIS for graphs of large sizes. In this paper, we will introduce an algorithm and investigate its run time complexity or growth in the forms: $O(r^{n})$, $O(2^{cn})$, and $O(n^{4})$, where *r* and *c* are constants.

Section 2 discusses the new algorithm in details. It will be refered to as an algorithm for Finding Maximum Independent Set (FMIS) in a graph. In Section 3, we present the experimental performance of the FMIS algorithms. The performance of the FMIS will be investigated more on graphs with different sizes and densities. Finally, concluding remarks are made in Section 4.

2. The FMIS Algorithm

2.1. Finding the MIS

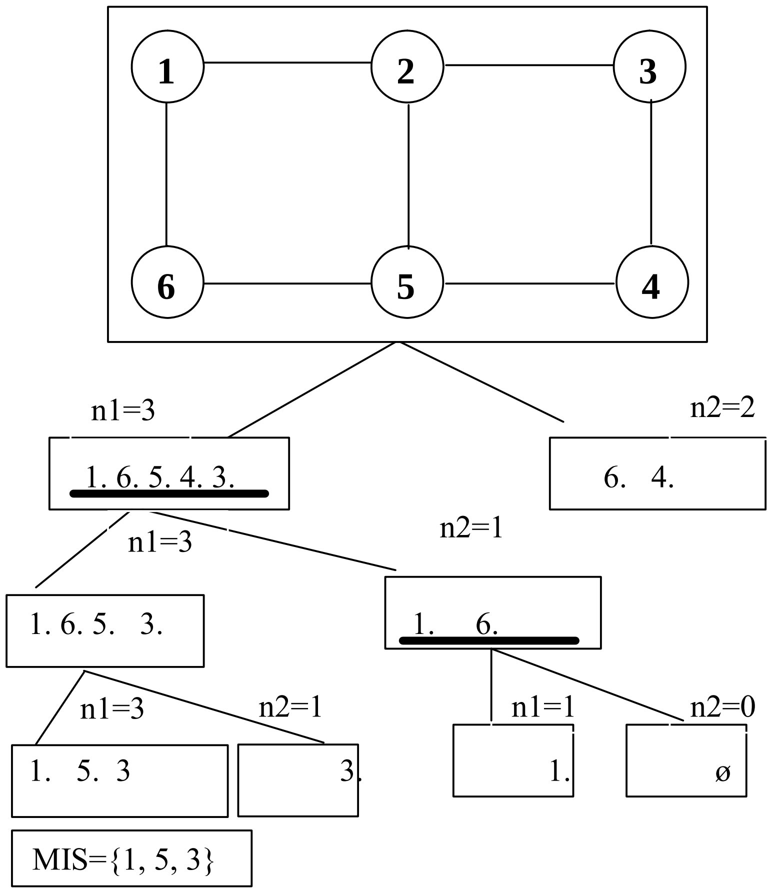
The modified Wilf's algorithm generates all possible independent sets and selects the one which has maximum cardinality [8]. It starts by choosing a vertex v^* in the graph. This vertex has the highest degree. Then, it produces two lists: one contains v^* and one does not. The set Nb(v^*) contains all vertices that are adjacent to v^*. Fig. 1 shows a recursive function which returns the size of a MIS for a given graph *G*.

Figure 1: A recursive function to find the MIS for a given graph G.

```
Maxset(G);
Begin /* to return the size of the largest Independent Set of G*/
        If degree(G) = 0 then
                Maxset = all vertices in G
        Else
                Choose a vertex v*;
        n1 = Maxset(G - {v*});
        n2 = Maxset(G – {v*} – Nb(v*));
        Maxset = maximum_of(n1, n2);
end;
```

Fig. 2 shows a graph, where V ={1, 2, 3, 4, 5, 6} and E = {(1,2), (1,6), (2,3), (2,5), (3,4), (4,5), (5,6)}. The vertices 2 and 3 are adjacent, but 3 and 5 are not. The set {1, 3} is an independent set, but the set {1, 4, 5} is not. The set {1, 3, 5} is a MIS. The $\alpha(G)$ = 3. The Fig. shows how the MIS is generated by the MW algorithm. In each level in the tree, the sub-graph to the left is generated by calling Maxset(G-{v*}). The sub-graph to the right is generated by calling Maxset(G –{v*}-Nb(v*)). The {2, 4, 6} is another MIS.

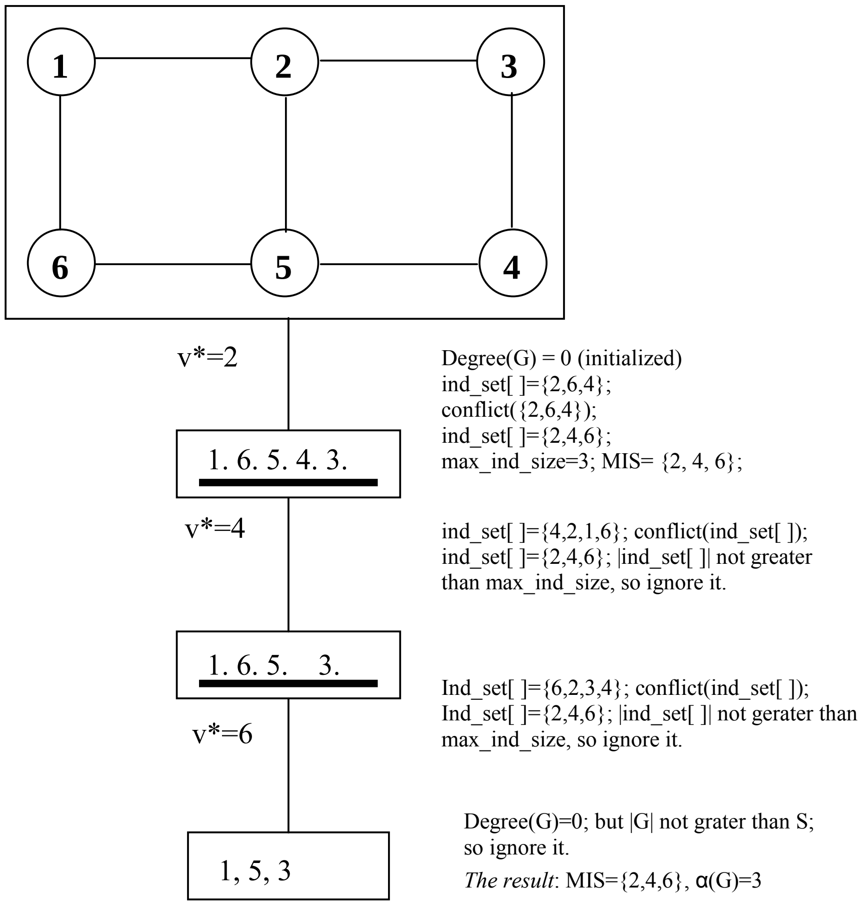
Figure 2: A graph G = {1, 2, 3, 4, 5, 6} and one of its maximum independent sets. The started v* was 2 with degree 3. The MIS ={1,3,5}, generated by Maxset(G) function in Fig. 1.



2.2. The FMIS Algorithm

The main function of this algorithm is to find the maximum independent set of a given *G*. The algorithm starts by computing the degree of the *G*. If it equals zero (there is no edges between vertices), then the MIS will be all vertices of G and the *α(G)* will be the cardinality of *G*. If the cardinality of *G* is not equal to zero, then a vertex which has highest degree-say *v* - is chosen. All vertices which are not adjacent to v^* are found and put with v^* in a candidate MIS. If the cardinality of this candidate set is greater than two and greater than the size of the MIS, then the algorithm continues to remove, recursively, the vertex which has the highest degree vertices in the candidate *I*.

Figure 3: Applying the FMIS on G= {1, 2, 3, 4, 5, 6}. The MIS, here, is {2, 4, 6}, where α(G)=3.


1
2
3
6
5
4
v*=2
Degree(G) = 0 (initialized)
ind_set[ ]={2,6,4};
conflict({2,6,4});
ind_set[ ]={2,4,6};
max_ind_size=3; MIS= {2, 4, 6};
1. 6. 5. 4. 3.
v*=4
ind_set[ ]={4,2,1,6}; conflict(ind_set[ ]);
ind_set[ ]={2,4,6}; |ind_set[ ]| not greater than max_ind_size, so ignore it.
1. 6. 5. 3.
v*=6
Ind_set[ ]={6,2,3,4}; conflict(ind_set[ ]);
Ind_set[ ]={2,4,6}; |ind_set[ ]| not gerater than max_ind_size, so ignore it.
Degree(G)=0; but |G| not grater than S; so ignore it.
The result: MIS={2,4,6}, α(G)=3
1, 5, 3

The cardinality of the sub-graph *G-{*v^**}* is tested. If it equals to zero, the cardinality of the tested graph is compared with variable max_ind_size, denoted *Z*, as shown in Fig. 2. If the cardinality is greater than *Z*, then *Z* will be the cardinality of the new graph *G*-{v^*} and the MIS will be all nodes of *G-{*v^**}*. If the cardinality is less than *Z*, then another node is chosen; and all vertices not adjacent to the new chosen vertex must be found. Keeping in mind that the previously chosen nodes may be interrelated to the new candidate MIS. If the cardinality of the new candidate set is greater than 2 and greater than *S*, then this set is passed to the conflict function.

Fig. 3 explains the FMIS algorithm to find the MIS = {2, 4, 6}. This is produced because the algorithm starts with $v^* = 2$, since the degree of 2 is 3. If the algorithm starts with $v^* = 5$, then the MIS will be {1, 3, 5} and *α(G)*=3. The latest MIS is the same MIS produced by the MW algorithm in Fig. 2.

Figure 4.a: The main function to find the MIS in FMIS algorithm.

```
void Gen_ind_set( )
{
    int v, counter=0,x=0;
  if(degree( )=0)
        {
            for(int i=0;i<num_node;i++)
                if(visited_node[i]='F') counter++;
            if(counter>max_size)
                {
                  for(int i=0;i<num_node;i++)
                   if(visited_node[i]='F')
                        max_ind_set[x++]=G_node[i];
                        max_size=x;
                        size1=x;
                }
        }
    else
          {
            v = select_node( );
            size1=0;
            for(int i=0;i<num_node;i++)
                {
                  if(graph[v][i]=0 AND visited_node[i]!='X')
                    ind_set[size1++]=i;
                }
            if(size1>2 AND size1>max_size)
                  conflict( );
            if(size1>max_size)
                  {
                    max_size=size1;
                    for(i=0;i<max_size;i++)
                        max_ind_set[i]=ind_set[i];
                  }
            visited_node[v]='T';
            Gen_ind_set( );
          }
}
```

Figure 4.b: The degree function to find the degree of a given graph.

```
int degree( )
{
  int deg=0;
    for(int i=0;i<num_node;i++)
    { if(deg>0) break;
        if(visited_node[i]!='F') continue;
        for(int j=0;j<num_node;j++)
          {
                if(graph[i][j]=1 AND visited_node[j]='F')
                deg++;
                if (deg>0) break;
          }
    }
  return deg;
}
```

Figure 4.c: The **select_node**( ) function to select a vertex with the highest degree in a graph to be a candidate in MIS.

```
int select_node( )
{
   int max1,max2=0,r=0;
    for(int i=0;i<num_node;i++)
        {
          max1=0;
          if(visited_node[i]!='F') continue;
          for(int j=0;j<num_node;j++)
            if(graph[i][j]=1 AND visited_node[j]='F')
                max1++;
          if(max1>max2)
            {
                r =i; max2=max1;
            }
        }
  return r;
}
```

Figure 4.d: The conflict() function in which to find an independent set.

```
void conflict( )
    {
        int c=0,del=0,rm,J=0;
        for(int i=0;i<size1;i++)
          {
            c=0;
            for(int j=0;j<size1;j++)
                if( graph[ind_set[i]][ind_set[j]]=1)
                  c++;
            if(c>del)
                {
                    del=c;
                  rm=ind_set[i];
                }
        }
        if(del>0)
          {
            size1--;
            J=0;
            for(int i=0;i<num_node;i++)
                if(rm!=ind_set[i] AND J<size1)
                           G1[J++]=ind_set[i];
            for(i=0;i<J;i++)
                  ind_set[i]=G1[i];
            size1=J;
            conflict( );
          }
      }
```

The essential code used by the FMIS for finding the MIS is shown in Fig. 4. The **Gen_ind_set( )** generates the nodes of the MIS for a given graph. The graph was represented in a square matrix (array of size *n* by *n*), where *n* is the number of vertices in the given graph *G*. Our main contribution is the **conflict()** function.

2.3. The FMIS Complexity

The function **Gen_ind_set( )** in Fig. 4.a is a recursive one and calls the **degree( ), select_node()**, and **conflict()** functions. Each of **degree()** and **select_node( )** has the running time complexity of $O(n^2)$. The running time complexity of one pass through **conflict()** function is $O(n^2 + n)$. In every phase, at least one node is removed. At every next phase, n will be decreased by at least one. As a result, we have the recurrence relation (1) that estimates the time complexity of the **conflict()** function. This can be simplified to O(n^3).

$$F(n) = O(n^2 + 2n) + F(n-1)\ldots\ldots\ldots\ldots\ldots \quad (1)$$

The main function **Gen_ind_set()** has four loops. Thus, the worst case complexity for one pass is $O(n^3 + 2n^2 + 3n)$. In every pass, we chose one node. At worst case, we need to choose all nodes. This means there is (n-1) passes. In next pass, the number of vertices will decremented by one. Thus, the complexity of **Gen_ind_set()** will be as shown in relation (2). The relation can be simplified to be $O(n^4)$.

$$T(n) = O(n^3 + 2n^2 + 3n) + T(n-1)\ldots\ldots. \quad (2)$$

3. Computational Results

In this section, the complexity of FMIS algorithm will be estimated from the experiment computational results. The results were gathered when the MW and the FMIS algorithms were programmed and tested on an Intel Pentium IV. The results are used to derive a run time complexity in the form *$O(r^n)$* and *$O(2^{cn})$*. The tested graphs are generated randomly by choosing the order (*n*) of the graph and the density (*D*).

A measurement of the density *D* of a graph can be computed by finding the degrees for all nodes divided into *n*(*n-1*). A graph *G* with *n* vertices and $|E| = O(n)$ is sparse, while a *G* with $|E| = O(n^2)$ is dense [13].

We tested the MW algorithm with graphs of orders 20, 30, 40,..., 200, and the FMIS algorithm with graphs of orders: 20, 30, …, 2000. Different graphs of densities: 0.1, 0.2, 0.3, …, 0.9 were generated for those orders. Ten graphs were generated for each configuration, and their average results were computed.

Table 1: Execution time in seconds, resulted by the FMIS algorithm and MW algorithm, for graph with sizes = (100, 120, 140,…,200) and densities = (0.1,0.2, 0.3, …, 0.6). The entry # means the used machine was not able to compute it.

Number of nodes	Modified Wilf			Proposed Algorithm	
	density	AVG time	AVG Size of MIS	AVG time	AVG Size of MIS
100	**0.1**	2642	33	0.1	33
	0.2	54	21	0.1	21
	0.3	3	17	0.1	17
	0.4	0.5	13	0.1	13
120	**0.1**	64269	35	0.2	35
	0.2	381	23	0.2	23
	0.3	14	17	0.2	17
	0.4	1.5	14	0.2	14
140	**0.1**	#	#	0.4	38
	0.2	2332	24	0.4	24
	0.3	49.5	18	0.4	18
	0.4	3.5	14	0.4	14
	0.5	0.5	12	0.2	12
	0.6	0.5	10	0.1	10
160	**0.1**	#	#	0.6	40
	0.2	13091.5	26	0.8	26
	0.3	172	19	0.7	19
	0.4	10	14	0.5	14
	0.5	1	12	0.4	12
	0.6	0.5	9	0.2	9
180	**0.1**	#	#	1.1	42
	0.2	55858.5	27	1.2	27
	0.3	508.5	19	1.1	19
	0.4	24.5	15	0.8	15
	0.5	2.5	12	0.6	12
	0.6	0.5	10	0.4	10
200	**0.1**	#	#	1.7	45
	0.2	#	#	1.9	28
	0.3	1488.5	20	1.7	20
	0.4	56.5	15	1.3	15
	0.5	4.5	13	0.9	13
	0.6	0.5	10	0.4	10

Table 1 shows orders (sizes) of graphs, sizes of the MISs, and the CPU run time for finding the MIS of graphs with densities: 0.1, 0.2, 0.3, …, 0.9. The value # entry means that the used computer was not able to compute the CPU run time. The configuration with CPU run times less than 0.001seconds were not reported in the table. The CPU times were rounded. The experiment CPU run times for both algorithms are increasing as the densities are decreasing-for a graphs of the same order. Fig. 5 shows the relation between the run time and the density. For the same graph, the FMIS algorithm produced the MIS with the same size of the MIS produced by the MW algorithm. This verifies the correct output of the FMIS algorithm. The MW algorithm runs out of space for graphs of sizes 140 and the density 0.1.

The run times of the MW algorithm are increasing very fast as the sizes of the graphs are increasing, and are getting worse for graphs with low densities. The results indicate that both algorithm's CPU time is not large vary in case of small size of graphs up to a graph of size 60 nodes. However, the MW algorithm run time is rapidly increasing after applying it on graphs of size greater than 60. All these results assure that the FMIS is more efficient than the MW algorithm.

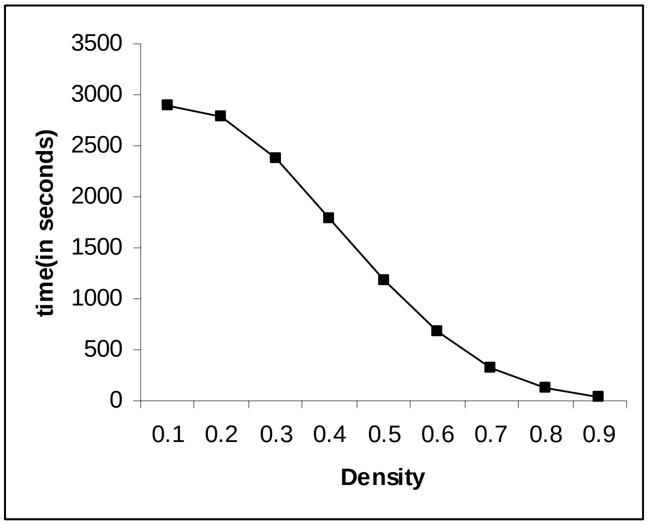
Table 2 shows the results when testing the FMIS algorithm on graphs, with orders: 300, 400, …, 2000 nodes. The results include the run time it took to find the MIS of the original graph.

Table 2: The CPU run times in seconds resulted of running the FMIS algorithm when graph size =, 300, 400, 500, …, 2000 and density = 0.1, 0.2, 0.3, …, 0.9.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
300	9.6	9.0	8.6	6.6	4.6	2.6	1.2	0.6	0.2
400	33.4	33.0	28.8	22.0	14.2	8.0	4.0	1.6	0.4
500	89.0	87.4	75.0	56.4	37.0	20.6	9.6	3.4	1.0
600	189.2	185.6	159.8	120.2	78.8	44.2	20.6	7.0	2.0
700	356.0	346.2	298.0	224.2	147.6	83.6	38.8	13.4	3.4
800	611.8	595.2	509.6	384.4	252.8	144.4	67.6	23.4	5.0
900	985.0	956.2	821.6	615.0	404.6	230.6	108.8	38.2	8.4
1000	1498.4	1456.4	1249.4	937.2	617.2	353.0	167.2	58.8	12.8
1100	2208.5	2131.5	1827.0	1371.5	900.0	514.0	243.5	87.0	19.0
1200	3109.0	3015.0	2592.5	1937.5	1270.5	722.0	342.0	121.0	26.0
1300	4295.0	4146.0	3561.5	2666.5	1753.0	991.5	467.0	166.5	36.0
1400	5786.0	5564.5	4790.5	3582.0	2349.5	1330.0	627.5	224.0	48.5
1500	7592.0	7340.0	6278.5	4714.0	3089.5	1752.5	831.5	295.0	63.0
1600	9836.0	9539.0	8132.0	6093.0	3979.0	2257.5	1063.5	377.5	81.5
1700	12562.0	12115.5	10379.5	7757.0	5057.5	2858.0	1339.5	474.0	102.0
1800	15884.5	15392.0	13125.5	9803.5	6380.5	3599.5	1684.0	594.0	126.5
1900	20188.5	19318.5	16448.5	12262.0	7972.0	4495.0	2106.0	743.5	158.0
2000	23828.5	23044.0	19706.0	14701.0	9576.0	5391.0	2525.0	894.0	192.0

Fig. 5 shows the performance of the proposed algorithm at fixed size of graph of N = 1000 nodes with densities: 0.1, 0.2, 0.3, …., 0.9. The results indicate that the run time to find the MIS is inversely proportional to the density for a graph with a fixed size. The results in Table 2 indicate that the size of the MIS decreases as the density increases, for a graph of fixed size.

Figure 5: The CPU run times of the FMIS for graphs of a fixed number of nodes (1000) and different densities.



Assume that the run time complexity is expressed as in Equation (3), where: *r*, *k*, and *c* are constants. If we let *k* =1 and *c* = 1, then from the experimental results, we can compute *r*. Table 3 shows the values of *r* for different sizes and densities. The average values of *r* vary from 1.0016 to 1.0069. If we let *r* to be the average of these values, then the complexity of the FMIS algorithm can be claimed to be $O(1.0052^n)$. This is better than $O(1.39^n)$ reported in [8]. If we let *k* =1 and *r* = 1.39 in Equation (3), we will have an average run time complexity of order $1.39^{0.0159n}$ for the FMIS compared to order of 1.39^n reported in [8].

$$C(\mathbf{n}) = k(r^{cn}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

Table 3: Average values of *r* and *c* in equations (3) and (4) for *n* = 300, 400, …,2000 with densities: 0.1, 0.2, …, 0.9

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**Avg**
r in r^n	1.0069	1.0069	1.0067	1.0064	1.0059	1.0052	1.0044	1.0032	1.0016	1.0052
c in 2^{cn}	0.0099	0.0099	0.0097	0.0092	0.0085	0.0076	0.0063	0.0046	0.0024	0.0076
c in 1.39^{cn}	0.021	0.0209	0.0204	0.0194	0.0179	0.0159	0.0133	0.0098	0.0048	0.0159

Now, assume that the run time complexity is expressed as in Equation (4), where *a* and *b* are constants. If we let each of *a* and *b* to be 1, then the value of *E*(*n*) can be computed. So, the complexity of the FMIS can be in the form $O(2^{0.0076n})$. Thus, the performance of the FMIS is better, in run time, than the best performance reported in [9].

$$E(n) = a2^{\mathbf{bn}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

4. Conclusions

It is important to find the Maximum Independent Set (MIS) in a graph in an acceptable run time. We have presented an algorithm to find the MIS in a graph; and analyzed its performance. The algorithm has been implemented; and experiments were conducted on a PC, using graphs of order up to 2000 nodes in which to validate the theoretical complexity of the FMIS algorithm. The experiment results indicate that the algorithm performs better than other known exact algorithms for finding MIS. It is more efficient than the Modified Wilf algorithm in terms of the number of nodes in a graph and the time complexity.

For future research, we plan to compare the FMIS algorithm with other approximation algorithms. Plans are also recommended to verify that the selection node is always better when it has the highest degree. The algorithm will be applied to solve problems such as coloring graphs.

References

[1] Abello J., S. Butenko, P.M. Pardalos, and M.G.C Resende, "Finding Independent Sets in a Graph Using Continues Multivariable Polynomial Formulation," *AT&T Labs Research Technical Report*, 2000.

[2] Al-Jaber, A. and Sharieh, A. "Algorithms Based on Weight Factors for Maximum Independent Set," *DIRASAT*, Vol. 27, No1., (1999) 74-90.

[3] Buuce E. Sagan, Vincent R.Vatter. Maximal and Maximum Independent Sets in Graphs With At Most Cycles, Michigan State University, 2003.

[4] Feo T.A.R.E, and M.G.C Resende, "A Probabilistic Heuristic for Computationally Difficult Set Covering Problem," *Operations Research Letters*,8,(1989) 67-71

[5] Garey, M. R. and Johnson, D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman, 1983

[6] Narendra Karmarkar, Mauricio G.C Resende and K.G Ramakrishnan. An Interior Point Approach to the Maximum Independent Set Problem in Dense Random Graphs, AT&T Bell Laborattories,1989

[7] Timothy M. Chan. A Note on Maximum Independent Sets in Rectangle Intersection Graphs, University of Waterloo, Canada,2003. visited in May 12, 2004

[8] Wilf, H.S. 1986. Algorithms and Complexity, Prentice Hall Inc, London, UK.

[9] Beigel, Richard, "Finding Independent Sets in Sparse and General Graphs," http://www.cis.temple.edu/~beigel/papers/mis-soda.html, visited 23/07/2006.

[10] http://www.worldhistory.com/wiki/G/Graph-coloring.htm, visited in June 6 2004

[11] http://en.wikipedia.org/wiki/Glossary_of_graph_theory#Independence, visited May 12, 2004

[12] http://www.nlsde.buaa.edu.cn/~kexn/benchmarks/graph-benchmarks.htm " Maximum Independent Set (MIS) and Minimum Vertex Cover (MVC)", visited 23/07/2006.

[13] http://www.cs.utk.edu/~doucet/graph/definitions.html "Definitions: What's a Graph", visited 2??/10/2006.