Experimental Study to Find Maximum Independent Sets:

Implementation of MIS-1, MIS-2 & Minimum Vertex Cover Approximation

Name: Gourav Ghoshal

UIN : 726006523

Dept: Industrial and System Engineering

Summary:

The study reports the implementation of 2 algorithms that have run time of O(1.619^n) and O(1.47^n) repectively, to find the Maximum Independent Set in a graph. For implementation and calculation purpose, 10 dataset with less than 500 nodes were selected from DIMACS. However, due to computational limitation, the result for only one dataset with 34 nodes was generated and 2 other test datasets were also created and checked to calculate the correctness of the implementation. Further, since the density (Number of existing arcs upon number of possible archs) is low, I tried to find the Minimum Vertex Cover for each of the graphs, since Maximum Independent Set is complementary to the Minimum Vertex Cover. This was also limited by the computational power available. Hence, I implemented the greedy Minimum Vertex Cover Approximation algorithm to find the lower bound of Minimum Vertex Cover and corresponding upper bound of Maximum Independent Sets. This study reports how the computation time varies for 2 Maximum Independent Set algorithms and increases with the number of nodes for the Minimum Vertex Cover Approximation Algorithm.

Keyword: Maximum Independent Set, Minimum Vertex Cover, DIMACS, Greedy Algorithm

Introduction and Background:

An Independent Set is in a graph G(V,E) is a set of nodes, which are not adjacent to each other. Therefore, it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two vertices. Equivalently, each edge in the graph has at most one endpoint in S. The size of an independent set is the number of vertices it contains. A Maximum A **maximum independent set** is an independent set of largest possible size for a given graph G. The problem of finding such a set is an NP-hard Optimization problem. As Such, it is unlikely that there exists an efficient algorithm for finding a Maximum Independent Set (MIS) of a graph.

In this project 2 Algorithms for finding the MIS is presented. Both of this algorithm use Depth First Serach (DFS) Methodology and search the whole solution space. Therefore, one of the most important characteristics of this problem is the number of nodes. Also, the complexity is O(1.619^n) and O(1.47^n) repectively for MIS1 and MIS2.

Linear Programming formulation:

The MIS problem can be formulated as a LP with binary decision variables denoting if a node is included or not in the MIS. This is a maximization problem where we try to maximize the number of nodes in the MIS. The constraint applied to the formulation makes sure that at most 1 vertex is selected for every edge in the graph:

$$\text{Maximize } \sum v_i$$

$$\text{s.t.} \quad v_i + v_j \leq 1 \;\; \forall (v_i, v_j) \in E$$

where, v_i = a binary variable : (0,1)

Since the MIS requires that all the possible independent sets are explored to finally get the MIS, it is a NP-Hard problem. This will result in a algorithms with complexity of order 2^n. Another way to solve the MIS problem is to implement intelligent Recursive Algorithms, such as MIS-1 and MIS-2 that were taught in the lecture in ISEN 661 course:

MIS-1:

The recursive algorithm is as follow:

- Let $N[v]$ denotes the closed neighborhood of v, e.g., $N[v] = N(v) \cup \{v\}$.

---

function $MIS1(G)$
if G has no edges **then**
  $MIS1 := |V(G)|$
else
  choose a non-isolated vertex v of G
  $$MIS1 := \max \begin{cases} MIS1(G[V \setminus \{v\}]) \\ 1 + MIS1(G[V \setminus N[v]]) \end{cases}$$
end if

The running time f(n) of the algorithm for the graph with n vertices satishfies the following:

$$f(n) \leq cn^2 + f(n-1) + f(n-2)$$

This results in O(1.619^n) time, since x=1.619... is a positive root of x^2= x+1

MIS-2:

The recursive algorithm is as follow:

function $MIS2(G)$
if G has no vertex of degree ≥ 2 **then**
$MIS1 := |V(G)| - |E(G)|$
else
choose a vertex v of degree ≥ 2
$$MIS1 := \max \begin{cases} MIS2(G[V \setminus \{v\}]) \\ 1 + MIS2(G[V \setminus N[v]]) \end{cases}$$
end if

The running time f(n) of the algorithm for the graph with n vertices satishfies the following:

$$f(n) \leq cn^2 + f(n-1) + f(n-3)$$

This results in O(1.466^n) time, since x=1.4657... is a positive root of x^3= x^2+1

Minimum Vertex Cover:

Another important combinatorial optimization problem in Graph theory is the Minimum Vertex Cover (MVC) which is the complementary case of the Maximum Independent Set, i.e. if G be the graph, the MVC = G \ MIS.

During the implementation of MIS in the dataset, only the “Karate” dataset with 34 nodes was solved using both MIS-1 and MIS-2 algorithms. Therefore, I tried to find out any alternative options to get to the results. Therefore, I tried to implement the Maximum Vertex Cover in the graph to finally, get the MIS.

Pseudo code for the Algorithm to implement the MVC that I used:

function MVC(G)

while there is an edge in the graph:

```
Select an edge (i,j)
G_residual_i = G \ i        (and also delete all the arc incident to i)
A = 1 + MVC(G_residual_i)
G_residual_j = G \ j        (and also delete all the arc incident to j)
B = 1 + MVC(G_residual_j)
Return minimum (A,B)
```

However, this approach also failed for the other dataset.

Minimum Vertex Cover Approximation:

The MVC Approximation algorithms generate solution that is no greated than 2 times the MVC, therefore, implementing this algorithm, I was able to find the Lower bound of MVC and hence, correspondingly the upper bound of MIS as follows:

MVC_approximation <= 2* MVC

MVC >= (MVC_approximation / 2)

MIS = |V| - MVC

<= |V| - (MVC_approximation / 2)

Where, |V| = number of nodes in the graph.

This way I was able to generate bounds of the optimal solutions.

The Pseudo code:

function MVC_approx(G)

while there is an edge in the graph:

```
Select an edge (i,j)
G_residual = G \ i          (and also delete all the arc incident to i)
G_residual = G \ j          (and also delete all the arc incident to j)
B = 1 + MVC(G_residual)
Return minimum (A,B)
```

This gave me results which helped me define the bounds for the optimal solutions.

Correction of the Codes:

The Algorithm run successfully on Chesapeake and Karate dataset, and gave output of MIS 17 and 20 respectively. 2 different test dataset was also generated and at every recursion, the output was used to create graphs for visualization the DFS tree search, one of which is represented below as the recursion tabular format:

Test-1:

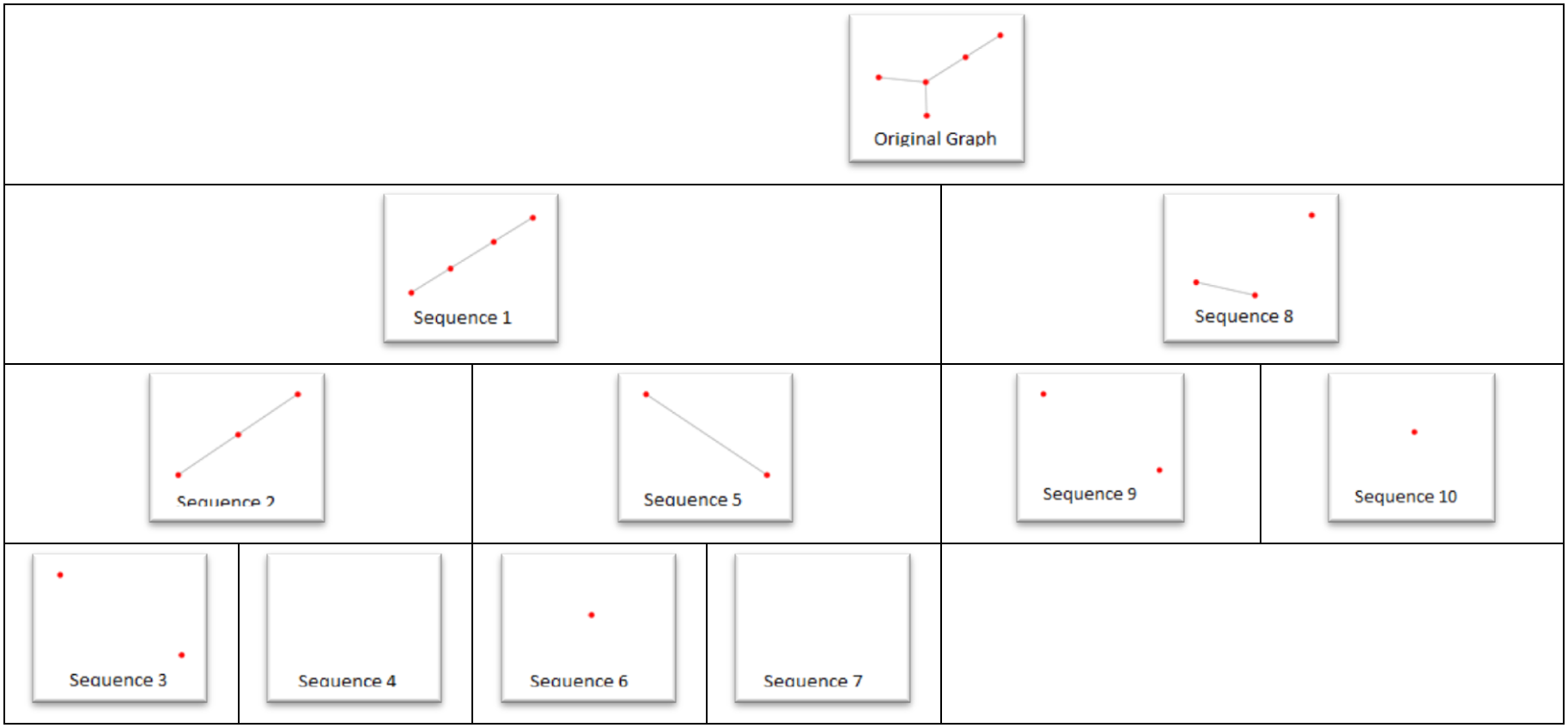
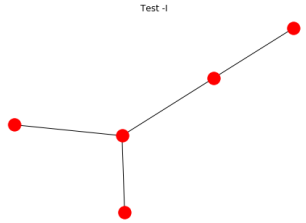


Figure 1Binary Recursion : DFS tree for Test- 1

The Algorithm follows this sequence: Original Graph ➡ Sequence 1 ➡ Sequence 2 ➡ Sequence 3 (Search finishes and it moves up by one step to Sequence 2) ➡ Sequence 4 (Agian, search finishes and moves up to Sequence 2, which is also completed now, so moves up another step to Sequence 1 and starts searching in Sequence 5) ➡ Sequence 5 ➡ Sequence 6 (Similar to as before, it proceeds) ➡ Sequence 7 ➡ Sequence 8 ➡ Sequence 9 ➡ Sequence 10.

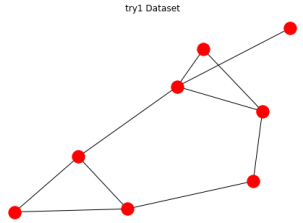
This confirms the correct implementation of the Algorithm through coding in Python.

For Test 1 :



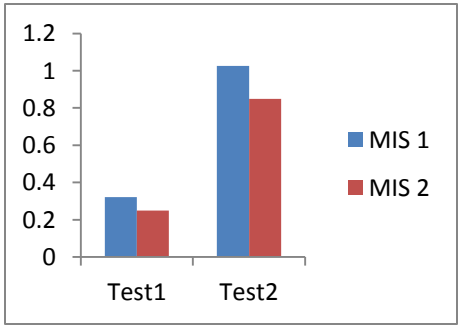
MIS = 3; Number of Nodes = 5;
Time required using MIS1= 0.322 mili seconds
Time required using MIS2= 0.249 mili seconds

For Test 2:



MIS = 4 ; Number of Nodes = 7;
Time required using MIS1= 1.025 mili seconds
Time required using MIS2= 0.8489 mili seconds

Run – Time Analysis on Test data:

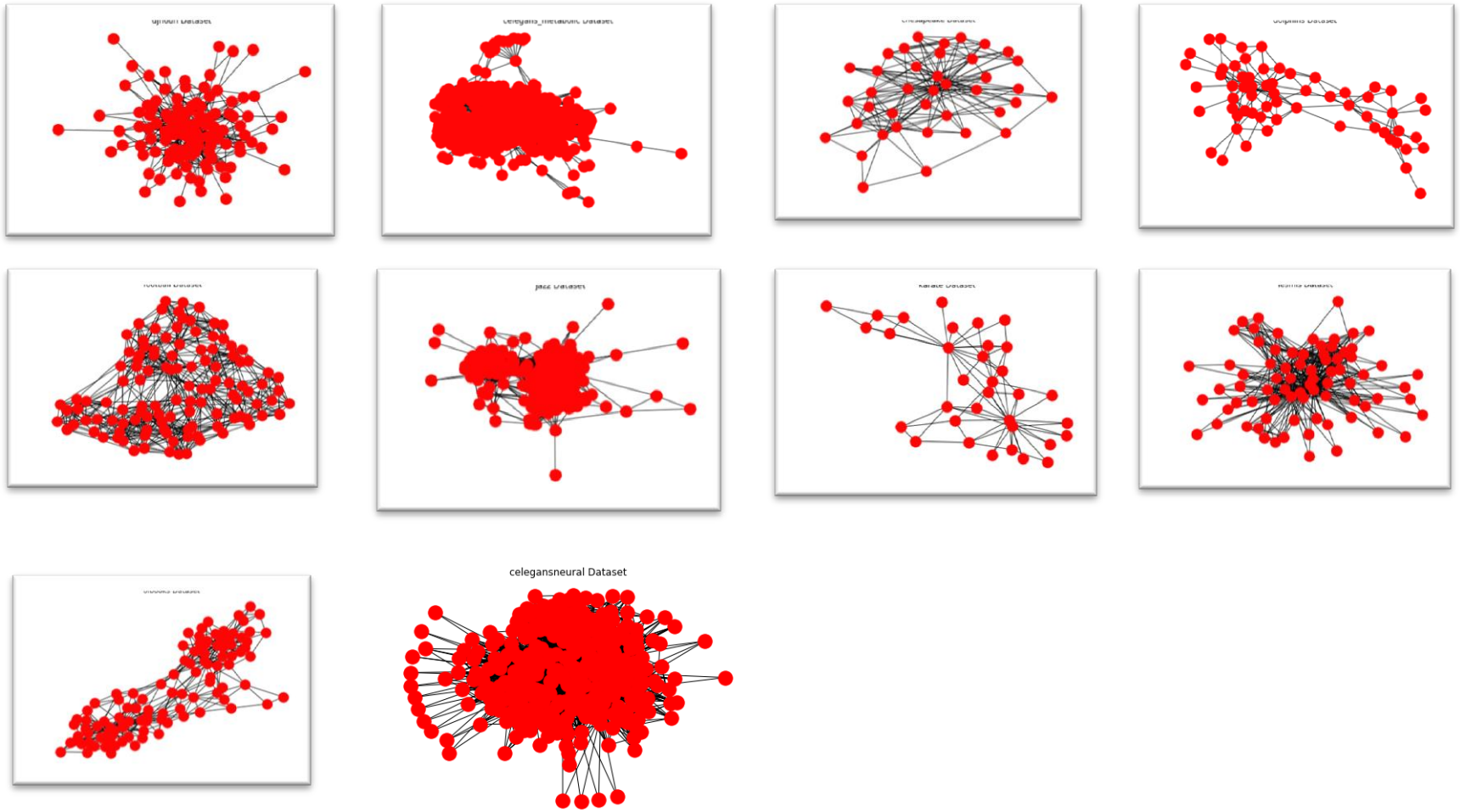


The Bar plot shows the Run-time for MIS-1 and MIS-2 algorithms on the 2 test dataset.

Since, it is already known the MIS-1 and MIS-2 has theoretical complexity as O(1.619^n) and O(1.47^n), this result just confirms it.

Data Exploration:

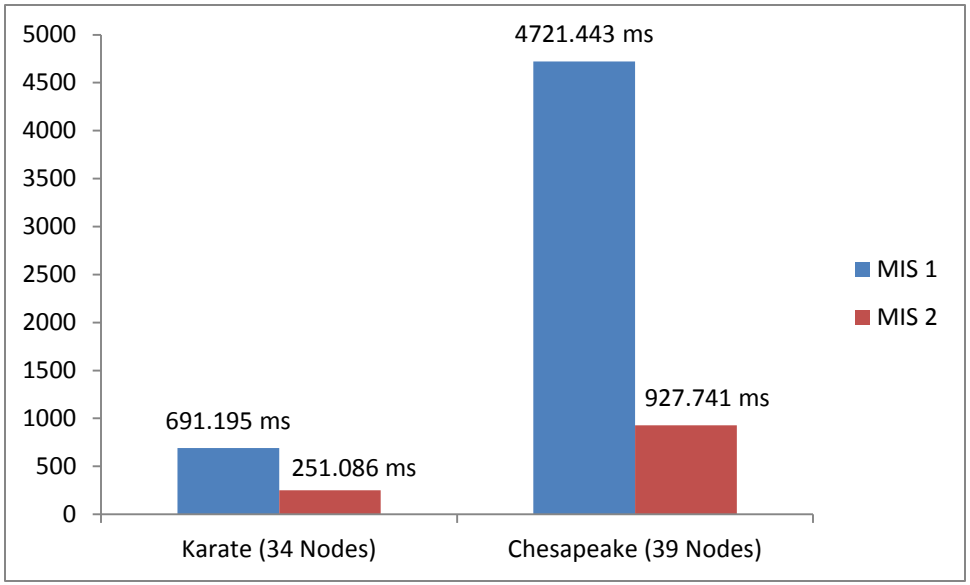
10 dataset was downloaded from the DIMACS website. I created the graph for visualization:



The number of nodes in the dataset are : 112, 453, 39, 62, 115, 198, 34, 77, 105 ,297

MIS-1 & MIS-2 Implementation:

For the 'Karate' and 'Chesapeake' dataset with 34 and 39 nodes, I was able to implement both the algorithms with MIS value 20 and 17 respectively and compare the results:



The bar plot clearly shows the MIS 2 algorithm take much less time as compared to MIS 1 algorithm. As the number of nodes increases, the difference is more prominent, as it is exponential order of algorithms.

Moreover, if we assume the complexity of MIS-1 is O(x^n), then we have:

$$\frac{x^{39}}{x^{34}} = \frac{4721.443}{691.195}$$

x^5 = 6.83 => x = 1.46

This is better than the theoretical value of O(1.619^n)

Similarly, for MIS-2:

$$\frac{x^{39}}{x^{34}} = \frac{927.741}{251.086}$$ => x^5 = 3.69 => x = 1.29, which is also better than the theoretical value of O(1.47^n)

This results proves that in Worst-Case scenario, the complexity order is as much as theoretical values, and for this example toy problems, the order of complexity is much less than the theoretical Worst Case scenario.

Minimum Vertex Cover Approximation:

Since, due to computational limitation, I could not calculate the MIS for other 8 graphs, I decided to approach the problem from the Complementary case of Minimum Vertex Cover(MVC). However, in a non-bipartite graph, finding MVC is again an NP-Hard problem. But, there exists an approximation greedy algorithm that generates solution within the ration of 2, i.e. if C is thesolution generated by Minimum Vertex Cover Approximation and C* be the Minimum Vertex Cover, then C <= 2C* , i.e. C* >= C/2.

Using this Fact I was able to calculate following results:

Dataset	Number of Nodes	Approx. MVC	Inference about MVC	Inference about MIS	Time Taken in miliseconds
adjnoun	112	82	>41	<71	4.316
celegans_metabolic	453	364	>182	<271	116.598
chesapeake	39	30	>15	<24 (17 to be exact)	0.756229
dolphins	62	46	>23	<39	1.4474
football	115	106	>53	<62	6.0534
jazz	198	180	>90	<108	44.378
karate	34	22	>11	<23 (20 to be exact)	0.505
polbooks	105	90	>45	<60	4.1988

Using this information, I used upper bounds to find the optimal Minimum Vertex Cover through pruning to speed up the search of MVC. However, I was able of find solutions for only 'Karate' and 'Chesapeake' dataset. But, using pruning reduced the computational time, as compared to the case where I implemented the exact recusrive algorithm without using any Bounds.

Dataset	MVC	Time for without Bounds	Time for with Bounds
chesapeake	NA	NA	NA
karate	14	206621.74	91857.65139

Conclusion:

The project tries to implement 2 recursive algorithms MIS-1 and MIS-2 to find the Minimum Independent Set, which is a NP-Hard problem. It was implemented successfully on 2 Dataset from DIMACS with 34 and 39 nodes in 691.195 mili seconds and 4721.443 miliseconds respectively using MIS-1 and in 251.086 mili seconds and 927.741 mili seconds using MIS-2 respectively. Also, the correctness of the algorithm was confirmed using 2 toy dataset and DFS search tree is recorded for visualization. Additionally, I tried to find Minimum Vertex Cover in the non-bipartite graph which is even harder than finding MIS, as the computational time suggests. For the same graph, finding MIS took less time than finding MVC using exact algorithms. However, using MVC approximation algorithm reduced the computational time significantly, giving us the lower bound on MVC and corresponding upper bound on MIS.

Future Scope:

It might be possible to prune the search tree to start with the upperbound of MIS found from the MVC approximation algorithm, which may be helpful for finding the solution in lesses computation using bounds. Also, instead of randomly choosing the node for removal, we can conduct research to identify which node should be selected to reduce the computation steps (probably using advanced Machine Learning techniques such Neural Networks/Decision Trees).