

Maximal Independent Sets

Matteo Amatori 7024736



Theoretical introduction - Graphs and MIS

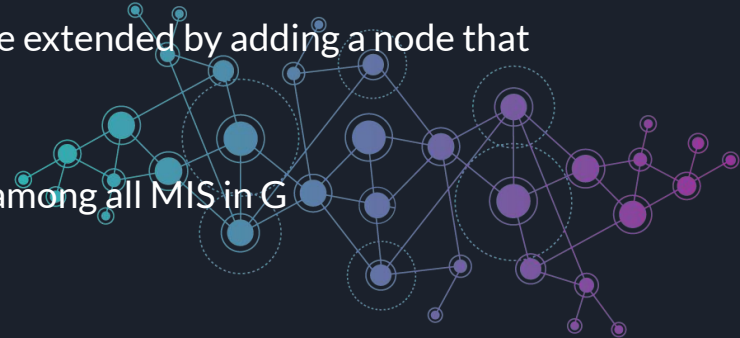
Undirected graph $G = (V, E)$ with:

- V = set of nodes (integers), we will call it's cardinality $n = |V|$
- E = set of edges (pairs of integers), we will call it's cardinality $m = |E|$

An Independent Set (IS) is a set of nodes in a graph G where no two nodes that belong to the IS are adjacent in G

A Maximal Independent Set (MIS) is an IS that cannot be extended by adding a node that is still not contained in the IS to the IS

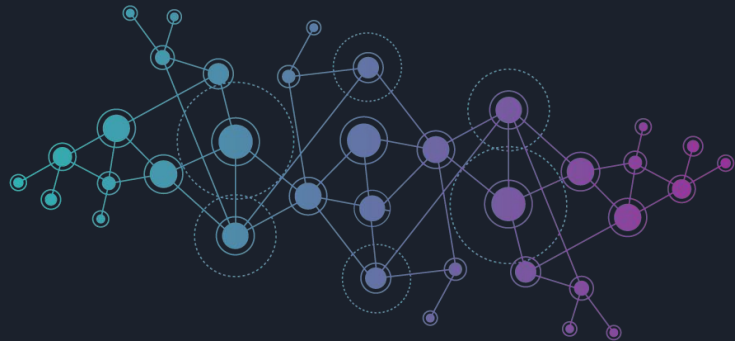
A Maximum Independent Set is the MIS of largest size among all MIS in G





Graph creation and data import

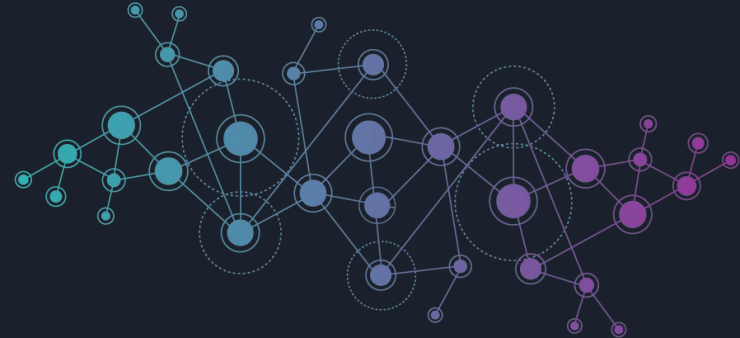
- Collecting all import files using `os.listdir(base_path)`
- For each file, we isolate every author name in each comment using the following RegEx: `'(_([a-zA-Z]+\.\?\s?)*_)'`
- For each author isolated, we create a node if the author is not yet present in G
- Then we create an edge that links every author that commented the same sequence





Problems to solve:

- find a Maximal Independent Set in G
- find all Maximal Independent Sets in G
- find the Maximum Independent Set in G

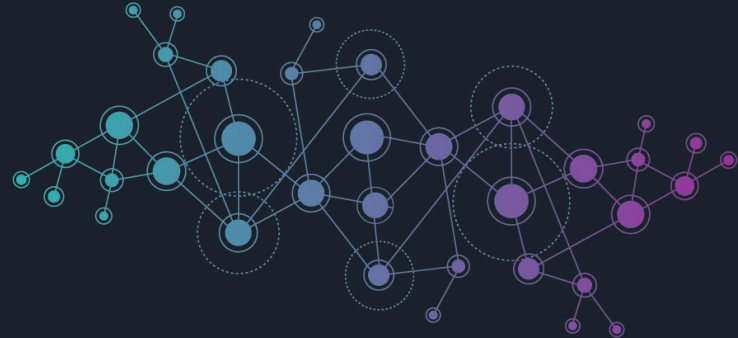




Find a MIS in G

In order to find a MIS in G , two approaches have been used:

- naive approach (greedy)
- implementation of the Luby's algorithm



Find a MIS in G - Greedy approach

Naive approach, main idea:

- select a random node v from the set of available nodes V_{set}
- add v to MIS
- remove v and its adjacency list from V_{set}
- loop until V_{set} is empty

Complexity: $O(m)$

Algorithm 1: Greedy algorithm for MIS

Result: a Maximal Independent Set

$V_{set} \leftarrow G.nodes$

$MIS \leftarrow emptylist$

while $V_{set} \neq \emptyset$ **do**

$v \leftarrow pick\ a\ random\ node\ in\ V_{set}$

$MIS \leftarrow MIS \cup v$

for node N in adjacency list of v **do**

$V_{set} \leftarrow V_{set} \setminus N$

end

end

return MIS

Find a MIS in G - Luby's Algorithm

More clever:

- doesn't pick nodes to add at random: it marks them with a probability which is inversely proportionate to its degree
- picks multiple nodes at the same time
- if two nodes that share an edge are chosen at the same time, one of them is unmarked
- removes unused edges so that following cycles are lighter

Complexity: $O(\log d * \log n)$

Algorithm 2: Luby's algorithm for MIS

Result: a Maximal Independent Set

$Vset \leftarrow G.nodes$

$MIS \leftarrow emptylist$

while $Vset \neq \emptyset$ **do**

$S \leftarrow \text{PICK-SET-S}(Vset)$

for edge E in edges **do**

if both nodes in E are in S **then**

 remove lowest degree node of E from S

end

end

for node N in S **do**

$MIS \leftarrow MIS \cup N$

 remove N and its adjacency list from $Vset$

end

for edge E in edges **do**

if E has missing endpoints **then**

 remove E from edges

end

end

end

function PICK-SET-S($Vset$)

$S \leftarrow emptyset$

for node N in $Vset$ **do**

 add N to S with probability $P_N = \frac{1}{2 \cdot \text{degree of } N}$

return S

Find all MIS in G

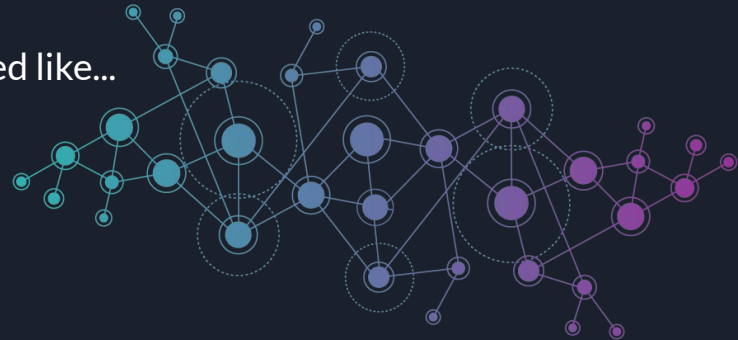
Just like finding all maximal Cliques, finding all MIS in a graph G using a naive approach is a hard problem

What's a 'naive approach' in this context?

A naive approach could be an algorithm that, for each node in the graph, recursively calls a function that searches for every possible node that can be added to the IS

Its complexity is of course exponential

An example of this naive approach could be implemented like...



Find all MIS in G - Naive approach

The algorithm first adds every node that has a degree of 0 to V_{set} , since they can't have neighbors, just by definition

Then, for each node v , it calls *find-all-MIS-rec*, that adds v to V_{set} , checks if the resulting set is a MIS. If it's not, then recursively calls itself on the remaining nodes

Complexity: exponential, quickly inefficient as n grows

```
function IS-ADDABLE( $V_{set}$ ,  $v$ )  
if  $v \in \bigcup_{w \in V_{set}} w.adj$  then  
  | return False  
else  
  | return True  
end function
```

```
function IS-MIS( $IS$ )  
if  $\exists v \mid IS-ADDABLE(v) = \textit{True}$  then  
  | return False  
else  
  | return True  
end function
```

Algorithm 3: Naive algorithm to list all MIS

```
 $foundMIS \leftarrow \textit{emptyset}$   
 $V_{set} \leftarrow \textit{emptyset}$   
 $RemainingNodes \leftarrow G.nodes$   
add all nodes of degree = 0 to  $V_{set}$   
remove all nodes of degree = 0 from  $RemainingNodes$   
for node  $v$  in  $RemainingNodes$  do  
  | if IS-ADDABLE( $V_{set}$ ,  $N$ ) then  
    | FIND-ALL-MIS-REC( $v$ ,  $V_{set}$ ,  $RemainingNodes$ ,  $foundMIS$ )  
return  $foundMIS$ 
```

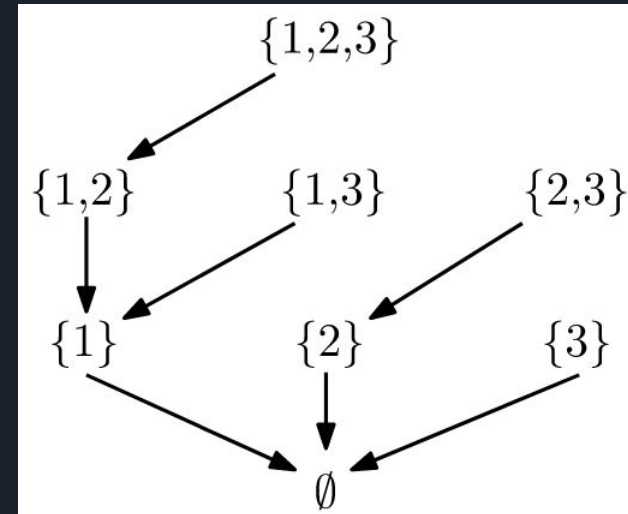
```
function FIND-ALL-MIS-REC( $v$ ,  $V_{set}$ ,  $RemainingNodes$ ,  $foundMIS$ )  
if  $RemainingNodes \neq \emptyset$  then  
  | return  
 $V_{set} \leftarrow V_{set} \cup \{v\}$   
 $RemainingNodes \leftarrow RemainingNodes \setminus \{N(v) \cup \{v\}\}$   
if IS-MIS( $V_{set}$ ) then  
  |  $foundMIS \leftarrow foundMIS \cup V_{set}$   
  |  $V_{set} \leftarrow V_{set} \setminus \{v\}$   
if  $\{V_{set} \cup RemainingNodes\} \notin foundMIS$  then  
  | for node  $w$  in  $RemainingNodes$  do  
    | FIND-ALL-MIS-REC( $w$ ,  $V_{set}$ ,  $RemainingNodes$ ,  $foundMIS$ )  
end function
```


Find all MIS in G - Reverse search approach

Reverse search is a technique that allows efficient enumeration of sets (or any other enumerable item that are governed by a natural ordering), by organizing them in a tree in which every distinct set appears only once

It is based on the idea of adopting a function to move from an item to another, using their natural ordering

This way, a tree is built, where each node in the tree is a distinct set, and we can enumerate all sets by performing a DFS on the tree






Listing Maximal Independent Sets with Minimal Space and Bounded Delay - Alessio Conte, Roberto Grossi, Andrea Marino, Takeaki Uno, Luca Versari

In the context of reverse search applied to the MIS problem, let's consider for example two MIS I_i and I_j

Here, each parent-child relation is generated by a rule that links each child I_j to exactly one parent I_i , such that $I_i < I_j$ for some given ordering (degeneracy ordering d)

The goal is to build a tree where each node is a MIS and we can go from I_i to I_j through a node v

The MISs that have no parent are called *roots* of the reverse search tree



Listing Maximal Independent Sets with Minimal Space and Bounded Delay - Alessio Conte, Roberto Grossi, Andrea Marino, Takeaki Uno, Luca Versari

The algorithm starts to run from the roots, which are MISs that have no parent, and performs a DFS from them


At each iteration, we if the current MIS has a child, we compute it using the formula

$$F(I, v) = \text{COMPLETE}(I'_{<v} \cup \{v\})$$

On the other hand, if the current MIS has no child, we restore the state of the parent in the DFS tree

Finally, if we reach a root going up in the tree, we return


During all the computation, if the discovered MIS is new, we store it in a list, to finally list them all at the end of the DFS traversal



Complexity of the referenced paper's algorithm based on reverse search

The complexity of the referenced algorithm is:

- Space: $O(n)$ additional memory (to save, for each node, the amount of neighbors that specific node has in the set $I_{<v}$)
- Time: $O(\min\{d\Delta n, mn\})$, where:
 - d is the graph's degeneracy
 - Δ is the maximum node's degree



Differences between the naive approach and the referenced paper's algorithm

Even if the naive approach, every time it adds a new node to the IS, removes each time the adjacency list of the node from the remaining nodes, due to its nature its complexity remains, in the worst case, exponential.

The referenced paper's algorithm instead, doesn't reach an exponential complexity.

This means that, already on ~30 nodes, the naive approach doesn't end its computation, while the other one terminates its execution in just a couple of seconds



References

- Luby's Alg. for Maximal Independent Sets using Pairwise Independence, Eric Vigoda - 02/02/2006
- Alessio Conte, Roberto Grossi, Andrea Marino, Takeaki Uno, Luca Versari. Listing Maximal Independent Sets with Minimal Space and Bounded Delay. International Symposium on String Processing and Information Retrieval (SPIRE), Sep 2017, Palermo, Italy. pp.144-160, ff10.1007/978-3-319-67428- 5_13ff. fffhal-01609012
- Avis, D., Fukuda, K.: Reverse search for enumeration. Discrete Appl. Math. 65(1– 3), 21–46 (1996)
- Reverse Search, Sebastian Nowozin - 07/08/2015
<http://www.nowozin.net/sebastian/blog/reverse-search.html>