On Maximum Independent Sets ©

Or S. Naim
Bar-Ilan University ISRAEL
Faculty of Exact Sciences
Founder of CUCUMBER - An OrSN Company
OrSN349@Yahoo.Com

Abstract
We consider the problem of finding a Maximum Independent Set in a graph. A recent paper by Xiao and Nagamochi (2017) shows an algorithm that can find a maximum independent set in O(1.1996^n) time and polynomial space. Even with restrictions on the degree of vertices, no polynomial-time algorithm discovered yet. In this paper we are showing a polynomial-time algorithm solving this problem.

1. Introduction

All graphs $G = (V, E)$ in this paper are simple, undirected, and unweighted.
A Maximum Independent Set of a graph is the largest subset of vertices that we can assemble without an edge $e \in E$ connecting any of its members.
A Maximal Independent Set (MIS) of a graph is a subset of vertices with no edge connecting any of its members. By adding any additional vertex from the vertices set V, we would violate this condition.
It is considerably easy to find a Maximal Independent Set using a greedy algorithm.

Lemma 1: Any Maximum Independent Set is a Maximal Independent Set.
Proof: Let us assume in contradiction, there is a Maximum Independent Set that is not a Maximal Independent Set. According to the definition of Maximal Independent Set, the Maximum Independent Set has two vertices that are neighbors therefore, the Maximum Independent Set is violating the term of an independent set and is not in contradiction.

2. Preliminaries

The easiest way of finding a Maximum Independent Set is to check all $O(2^{|V|})$ combinations of vertices and returning the largest subset that is independent. The runtime of this approach would be exponential. In their paper: "On Cliques in graphs" (1965), Moon, and Moser, show that the number of maximal independent sets in a graph is at most $3^{\frac{n}{3}}$. Therefore if we find a Maximal Independent Set using a greedy polynomial algorithm, we stand a chance of $\frac{1}{\left(3^{\frac{n}{3}}\right)}$ to be correct. A possible method to increase our chances of returning a correct output would be to repeat the process multiple times, start from a random vertex on each iteration, and output the largest subset among all iterations. To amplify the accuracy to be polynomial, we have to repeat the algorithm exponential number of times, hence losing the runtime benefit of a Monte-Carlo algorithm.

3. The Cucumber Method:

After gaining the intuition, a probabilistic approach isn't fruitful will try to solve the problem differently. For starters, we need to make some definitions:
Definition 2: A **degree** of a vertex v: $deg(v)$ is the number of neighbors vertex v has.

Definition 3: A **Social Degree** is a number reflecting the summary of all vertices' degrees in a set: $\Sigma_{i=1}^{n}\, deg(v_i)$.

Lemma 4: The Social Degree of a subset is lesser or equal to the Social Degree of V.
Proof: Since the degree of a vertex is a non-negative number, the more vertices you have in a set, the higher the social degree would become.

Lemma 5: The Social Degree of V equals $2 * |E|$.
Proof: Since each edge increases the degree of two vertices by one, the Social Degree of V increases by 2, so overall, the social degree of V equals to $|E| * 2$.

Lemma 6: The Social Degree of any independent set is lesser or equal to $|E|$.
Proof: Let us assume in a contradiction there is an independent set with a social degree of $|E| + 1$. As such, from the pigeonhole principle, at least two members of the Independent Set must be neighbors, in contradiction.

We are now ready to present the Cucumber method of finding a Maximum Independent Set in a graph. The general idea is to look for the vertices with the smallest degree and add them one at a time. The purpose of this is to shrink the input with each iteration while letting as many vertices as possible joining the set.

Now we will see a preparation polynomial algorithm to help get our input graph ready.
$CuPreperation(G(V,E))$:

1. $Initialize\ an\ empty\ array\ CuArray\ = \emptyset$
2. $Scan\ the\ graph\ and\ give\ each\ vertex\ a\ unique\ name\ using\ Breadth-First\ Search$
3. $Scan\ the\ graph\ using\ BFS\ and\ add\ to\ CuArray\ each\ vertex\ with\ the\ following\ fields$:
    3.1. $Degree$
    3.2. $Blacklist\ (Containing\ all\ neighbors\ of\ the\ vertex)$
4. $Sort\ CuArray\ by\ the\ degree\ of\ each\ vertex.$
5. $Return\ CuArray$

Runtime: Step 1: $O(1)$, Step 2: $O(|V| + |E|)$, Step 3: $O(|V| + |E|)$ for Breadth-First Search and $O(|V|)$ time on each vertex to assemble the local blacklist field for each vertex. The total runtime for this step is $O(|V|^2 + |E|)$.
Step 4: Efficient sorting algorithm takes: $O(|V| * log(|V|)$.
Overall: $O(|V|^2 + |E|)$.

Now we are ready to see the polynomial-time algorithm for finding a Maximum Independent Set.
$CuAlgorithm(CuArray)$:

1. $Initialize\ CuSet\ = \emptyset$
2. $Initialize\ a\ global\ Blacklist\ = \emptyset$
3. $Initialize\ Social\ Degree\ = 0$
4. $Loop\ through\ CuArray$

4.1. $If\ vertex\ IS\ NOT\ in\ global\ Blacklist$
    4.1.1. $Add\ vertex\ to\ CuSet$
    4.1.2. $Merge\ the\ vertex\ local\ blacklist\ with\ the\ global\ blacklist\ and\ add\ the\ vertex\ to\ the\ blacklist.$
    4.1.3. $Increase\ Social\ Degree\ by\ the\ degree\ of\ the\ vertex.$
    4.1.4. $If\ |global\ Blacklist| \geq |V|$
        4.1.4.1 $Return\ CuSet$
    4.1.5. $If\ Social\ Degree\ = |E|$
        4.1.5.1 $Return\ CuSet$

5. $Return\ CuSet$

Steps 4.1.4 and 4.1.5 are optional.
To prove the correctness of CuAlgorithm, we need to prove three lemmas.

Lemma 7: The independency property of CuSet. Steps 4.1 and 4.1.2 of CuAlgorithm, enforce that only vertices without neighbors in CuSet are permitted to join the set. As such, CuSet is an Independent Set.

Lemma 8: CuSet is a Maximal Independent Set: Let us recall the definition of a Maximal Independent Set. It is an independent set that an additional vertex will cease to keep this property. By Lemma 7, we know that CuSet is

independent. By the end of the run, the global blacklist contains the entire V. Therefore, all vertices are in CuSet or are neighbors of CuSet. As such, we cannot add a vertex to CuSet.

Lemma 9: CuSet is larger or equal in size to any other independent set.
To prove this, we will remove an arbitrary vertex v from CuSet, and show we can add up to a single vertex in its place. Since the neighborhood is a mutual property, we can be sure that no neighbor of the removed vertex is in the set and safely remove it from the blacklist. In regards to its neighbors, we will only remove them from the blacklist if they are not neighbors of other vertices in CuSet. Because the graph is connected, One of two scenarios can occur:

1. The degree of the neighbor of the removed vertex is 1 (That neighbor is the only neighbor of the removed vertex). In such a case, we can add the neighbor to CuSet.
2. The degree of the neighbor of the removed vertex is more than 1. In such a case, we have multiple scenarios:
   2.1. Both the neighbor and the removed vertex have share neighbors. In such a case, all shared neighbors "compete" on one available spot in CuSet.
   2.2. The neighbor has neighbors which it doesn't share with the removed vertex, and they are already in CuSet, preventing the neighbor from joining.
   2.3. The neighbor has neighbors which it doesn't share with the removed vertex, and they are not in CuSet. In such a case, we can add the neighbor.
3. The graph consists of only one vertex, resulting in no substitution.

Scenario 1 can only occur if the graph consists of two vertices. That's because we sort the vertices by their degree and add the vertex with the lower degree first if possible. Since the neighbor has a degree of one, and it wasn't qualified to CuSet, the removed vertex also has a degree of 1. Scenario 2.3 can also occur only once for a similar reason.

Lemmas 7 to 9 prove the correctness of our algorithm.

Run time: Besides constant time operations, our algorithm requires $O(|V|)$ time for the loop in step 4 and $O(|V|)$ to merge the blacklists on each iteration. For a total of $O(|V|^2)$.
Since the preparation took us longer, we are getting this algorithm for practically free.
Overall the runtime of our algorithm is $O(|V|^2 + |E|)$.

Space complexity: The length of CuArray, CuSet, and Blacklist are $O(|V|)$. CuArray also includes a field for the neighbors of each vertex for a total of $O(|V| * |V|) \in O(|V|^2)$.

4. Discussion
Our work is a huge step forward in solving a problem that was considered NP-Hard for decades. We now saw the Maximum Independent Set is in P.

References:
"Exact algorithms for maximum independent set" Xiao and Nagamochi (2017).

"On Cliques in graphs" Moon and Moser (1965).

Author: Or S. Naim ©, Bar Ilan University, ISRAEL.
Faculty of Exact Sciences.
Founder of CUCUMBER an OrSN Company.
January 24th, 2021.

...