Is it possible to find the maximum clique in general graphs?

José Ignacio Alvarez-Hamelin

IS IT POSSIBLE TO FIND THE MAXIMUM CLIQUE IN GENERAL GRAPHS?

JOSÉ IGNACIO ALVAREZ-HAMELIN

Abstract. Find the maximum clique is a know NP-Complete problem and it is also hard to approximate. This work propose an efficient algorithm to obtain it. If the presented theorem holds, this algorithm run in polynomial time in terms of the size of the graph.

1. Introduction

Finding cliques in graphs is a very known problem, mainly the maximum clique was fond to be a NP-Complete problem [Karp, 1972]. Indeed, from any vertex, to discover the maximum clique to which it belongs, we should take all combinations of k neighbors and verify whenever they are mutually adjacent, which yields an exponential time as function of vertex degree. Moreover, [Johnson, 1973] shows that there exists no sublinear approximation algorithm. This problem was largely treated and an extensive survey can be found in [Bomze et al., 1999]. In this work, we present an algorithm to find maximum cliques based in a reduction of the problem complexity. We decompose the graph from the vertex of highest degree until no vertices remains, then we re-build the graph computing the maximum clique at each step, restoring each one of their vertices. Next section is devoted to present the algorithm and the theorem showing its correctness. A section showing the algorithm applied to real graphs is presented to illustrate how it works. The paper is concluded with a discussion about the complexity of the problem.

2. Algorithm

Let $G = (V, E)$ be a simple undirected graph with $n = |V|$ vertices and $m = |E| \leq |V| \times |V|$ edges. The neighborhood of a vertex is the the set composed by vertices directly connected to it, i.e., $w \in N(v)$ such as $\{v, w\} \in E$. Then, the degree of vertex v is denoted as $d(v) = |N(v)|$. Call $H = (V, E, A)$ the annotated graph G, where $|A| = |V|$ such that if $v \in V$ then a_v is a list of attributes of vertex v. In this work, each of these attributes is a set called a_v^i, and $i \in N(v)$. Abusing of the notation, we use $a_v = \emptyset$ when $N(v) = \emptyset$.

The attribute a_v^x have the properties presented in Definition 1, and it is computed by Algorithm 1.

Definition 1. *Let a vertex $v \in V$ and one neighbor $x \in N(v)$ the a_v^x is a set of vertices such that the three following properties holds:*

(i) *this set denotes a clique having v and x;*

Date: October 8, 2011.
Key words and phrases. algorithms, graphs, complexity.
Acknowledgements: Juan Ignacio Giribet, Pablo Jacovkis, Mario Valencia-Pabon.

(ii) *the maximum clique* $K_{\max}(v)$ *in which* v *participates is found as*

$$K_{\max}(v) = a_v^x : \max(|a_v^x|, \forall x \in N(v)) \ ; \tag{1}$$

(iii) *and, if* $K_{\max}(v) = a_v^x$ *and* $w \in K_{\max}(v)$*, then* $a_v^w \subseteq K_{\max}(v)$.

Notice that we call $a_v = \{a_v^x, \forall x \in N(v)\}$.

Finally, the maximum clique of the graph G according to Definition 1 is,

$$K_{\max}(G) = a_v^x : \max\left(|a_v^x| / v \in V, x \in N(v)\right) \ , \tag{2}$$

where its cost is $O(m)$ for a connected graph or $O(n+m)$ in general.

The following theorem proof the correctness of Algorithm 1, verifying how this algorithm accords with Definition 1.

Theorem 1. *Algorithm 1 ends if* $G = (V, E)$ *has finite size, computing correctly attributes* a_v^x *for all vertices* $v \in V$ *and their corresponding neighbors, according to Definition 1.*

Proof. Let us start with the winding phase of the recursion, that is, steps 1.2, 1.4 and 1.8 are executed until we reach an empty graph G' (see 1.7). At this step, the end of recursion (step 1.7) is found because each call to `find_cliques`(G') function is done with a reduced set of vertices $|V'| = |V - 1|$ (and its induced graph), and G has finite size; i.e., the recursion is done n times.

From there, we analyze the unwinding phase. Let's start when steps 1.19 and 1.20 are executed for the first time: the return of the function will carry a graph with

Algorithm 1: Function $H \leftarrow$`find_cliques`(G)

Input: a graph $G = (V, E)$
Output: a graph $H = (V, E, A)$

```
1.1  begin
1.2      find a vertex v such that d(v) is maximum ;
1.3      set M = N(v);
1.4      G' ← G \ v ;
1.5      a_v^x ← ∅ for all x ∈ M, or a_v ← ∅ if M ∈ ∅ ;
1.6      set H' ← {∅, ∅, ∅} ;
1.7      if G' ∉ ∅ then
1.8          set H' ← find_cliques(G') ;
1.9          for each x ∈ M do
1.10             for each set a_x^i or a_x ∈ ∅ do
1.11                 set L ← (N(v) ∩ a_x^i) ∪ v ∪ x ;
1.12                 if (|L| ≥ |a_v^x|) then
1.13                     a_x^v ← L ;
1.14                     a_v^x ← L ;
1.15                 end
1.16             end
1.17         end
1.18     end
1.19     set H ← H' ∪ (v, v × M, a_v) ;
1.20     return H ;
1.21 end
```

just the vertex v, no edges and $a_v = \emptyset$ (step 1.5), that is $H = (\{v\}, \emptyset, a_v)$. Then, the following instance(s) can add vertices of degree zero until one instance begins to add the first edges (edge), getting a star with leaves (one leaf), because the degree is an increasing function (the winding phase was carried out taking the maximum degree at step 1.2, so the unwinding one reconnects vertices with the same degree or greater one). At this instance, steps 1.9 and 1.10 are executed and step 1.11 yields $L = \{v, x\}$ because a_x is empty (the vertex x has no registered neighbors until now). The following conditional sentence is true (step 1.5 assures $a_v^x \in \emptyset$) setting each neighbor as a clique, on both sides, the neighbor vertex $a_x^v = \{v, x\}$ and the local one $a_v^x = \{v, x\}$ (see 1.12, 1.13 and 1.14, we consider objects a_w as mutable).

From this point of the algorithm execution any one of the next vertices could build a $K_s/s \in \mathbb{N}$ (e.g., $s = 3$) because a new vertex joining former vertices constituting a K_{s-1} could appear. It is worth remarking that it is not possible to build a K_{s+1} at this stage (e.g., $s + 1 = 4$). The reason why it is not possible is that there are only K_{s-1} and a new vertex just adds edges between this new vertex and the present vertices, although this new vertex will never add an edge between the present vertices. In this way, the size of new cliques is an increasing function (either the maximum clique remains at same size or it is increased by one vertex).

Now, we will show the property (i) for a_v^x in Definition 1. We have also shown that the first elements in a_v^x constitute a clique of two vertices: v and its neighbor x. Considering, at any instance in the unwinding phase of the Algorithm 1, a vertex v has a clique stored for each one of its neighbors x in a_v^x. Let's consider, without loss of generality, a new neighbor of v, called w, having as neighbors $B \subseteq a_v^x = K_t$, that is $N(w) \supseteq B \cup v$. When step 1.12 is executed, either $a_w^v \in \emptyset$ or $a_w^v = \{w, v\}$ (because it is possible that $a_v^x \cap N(w)$ is empty in 1.11), and then $a_w^v \leftarrow B \cup w$, which is also a clique because $B \subseteq a_v^x$ is a clique and w is a neighbor of all vertices in B by hypothesis.

The property (iii) in Definition 1 is verified by Algorithm 1 because, according to what was shown in the last example, we have that a_v^x remains without any modification, meanwhile if $B = a_v^x$ then $a_v^w = a_v^x \cup w$. Notice that the maximum clique of a vertex x is the a_x^y such that either x or y is the last vertex reconnected to the graph.

To conclude the proof is enough to determine if the property (iii) in Definition 1 is obtained by Algorithm 1. The initial case was already shown in the second paragraph of this proof. Then, considering a case where the maximum clique of vertices w and y is K_s, and there exists another clique K_t such that $w, y \in K_t$ and $t \leq s$. As seen in a previous paragraph, vertices can only build cliques that increase the previous one by just one vertex. Let's consider that the next vertex z is connected with all vertices in K_t and z is at most connected with $t - 1$ vertices in K_s. The minimum difference that is needed to distinguish between two cliques is one vertex. Taking into account, without loss of generality, that the maximum clique for z is $K_s \cup z$, steps 1.11 and 1.12 will select the clique $K_s \cup z$ because at least one neighbor of z, let's call it u, has $a_u^i = K_s/i \in N(u)$. At this point, the values a_z^w, a_z^y, a_w^z, and a_y^z will be updated because their size is greater or equal to the size of a clique found before (see 1.12, 1.13 and 1.14). Thus, if $t+1 < s$ then K_s remains the maximum clique for w and y; or else $t + 1 \geq s$ and the set of t vertices in K_t plus z is the maximum clique for w and y. It is worth remarking that it is

possible that a vertex v has several cliques with a neighbor x, and the condition in 1.12 assures that the maximum clique is taken. Notice that a_w^y and a_y^w have the other clique K_s stored, but still $K_{t+1} = a_w^x \max(|a_w^x|, \forall x \in N(v))$ is the maximum clique for w because $t+1 \geq s$ and the previous maximum clique was K_s; the same occurs to y. □

Time complexity. Step 1.11 of Algorithm 1 is an intersection of two sets of size $d_{\max}$ (maximum degree), if both are ordered[1] that takes $O(d_{\max})$. Next, we consider the loop 1.10, that takes an extra $d_{\max}$, which yields $O(d_{\max}^2)$. The loop on neighbors 1.9 takes an extra $d_{\max}$, giving $O(d_{\max}^3)$. Step 1.2 has a complexity of $O(n)$ if vertices are not ordered, but it is an additive cost that is expected smaller than $O(d_{\max}^3)$. Finally, the recursion is done in every vertex of the graph, producing a total time complexity of $O(n \cdot d_{\max}^3)$. For the case when neighbors are not ordered, we get $O(n \cdot d_{\max}^4)$. Considering connected graph, we can express the n recursions and the visit to all neighbors in the loop 1.9 as visiting all the edges, yielding a time complexity of $O(m \cdot d_{\max}^2)$.

For general graphs, where $d_{\max}$ could be bound by n, getting a time complexity of $O(n^4)$. However, for graphs having a heavy tailed distribution which can be bound by a power law[2]: $P(d) \propto d^{-\beta}$, it is possible to find a lower bound. Indeed, these graph has $n^{\frac{1}{\beta}}$ as a bound of $d_{\max}$, therefore the complexity yields $O(n^{1+\frac{3}{\beta}})$ for $\beta \leq 3$, and for $\beta > 3$ either the search in 1.2 or the elimination in 1.4 dominates, reaching the bound $O(n^2)$ or $O(n \cdot d_{\max} \cdot \log d_{\max})$ respectively.

Storage complexity. It can be computed as the space to storage the graph G, which is $O(n \cdot d_{\max})$, and the space occupied by all the a_v^x sets. This last quantity can be computed as the length of each set a_v^x, which is bound by d_{max} and the number of them by vertex, which is also bound by d_{max}, yielding $O(d_{\max}^2)$ per vertex. Thus, the total storage complexity is $O(n \cdot d_{\max}^2)$.

3. Applications

In this section we illustrate Algorithm 1 through an implementation made in `python` programming language [Alvarez-Hamelin, 2011], and its application to some graphs showing their maximum clique.

First, we apply our algorithm for finding cliques to some random graphs defined by [Erdös and Rényi, 1959]. It is shown in [Bollobás, 2001, Bollobás and Erdös, 1976] that an ER random graph has a high probability to contain a clique of size,

$$r = \frac{2 \cdot \log n}{\log 1/p} \quad , \tag{3}$$

where n is the number of vertices and p is the probability that exist an edges between any pair of vertices.

Table 1 show the results, where the columns are: the size of the graph, the probability p, the average degree $\bar{d}$, the computed r according to Equation 3, the size found by Equation 2, the number of different cliques (i.e., at least one vertex is different), and if an induced clique of size $r+1$ where found. The last column is obtained adding new edges to build a greater clique than the maximum, it shows 'yes' when this clique is found and 'not' if this is not found. Moreover, the 'yes' answer also means that we found just one clique of that size (see the number of

[1] This can be done at the beginning for all vertices in the graph G, taking $O(n \cdot d_{\max} \cdot \log(d_{\max}))$.

[2] Most of the real problems in Complex Systems field has $2 \leq \beta \leq 3$.

n	p	$\bar{d}$	r	$\lvert K_{\max}\rvert$	$\lvert\mathcal{K}\rvert : \mathcal{K} = \{K \in K_{\max}\}$	induced $r+1$
100	0.01	1	2	**2**	60	yes
1000	0.01	10	3	**3**	159	yes
10000	0.01	100	4	**4**	372	yes
10000	0.04642	464.2	6	**6**	5	yes

TABLE 1. Maximum cliques in Erdös Renyi graphs.

clique r found in the original graph). We test the algorithm for several graph of each one obtaining the same results (excluding $|\mathcal{K}|$ which changed some times), but we display just one result of each case.

The result is evident, we always found the predicted maximum clique, even when an artificially one is introduced.

In a second place, we apply our algorithm to a AS Internet graph. This graph has, as a main properties, a power law degree distribution and most of vertices of low degree are connected to the high degree ones. We used an exploration of [CAIDA, 1998] of September 2011. Figure 1 shows a visualization of this map

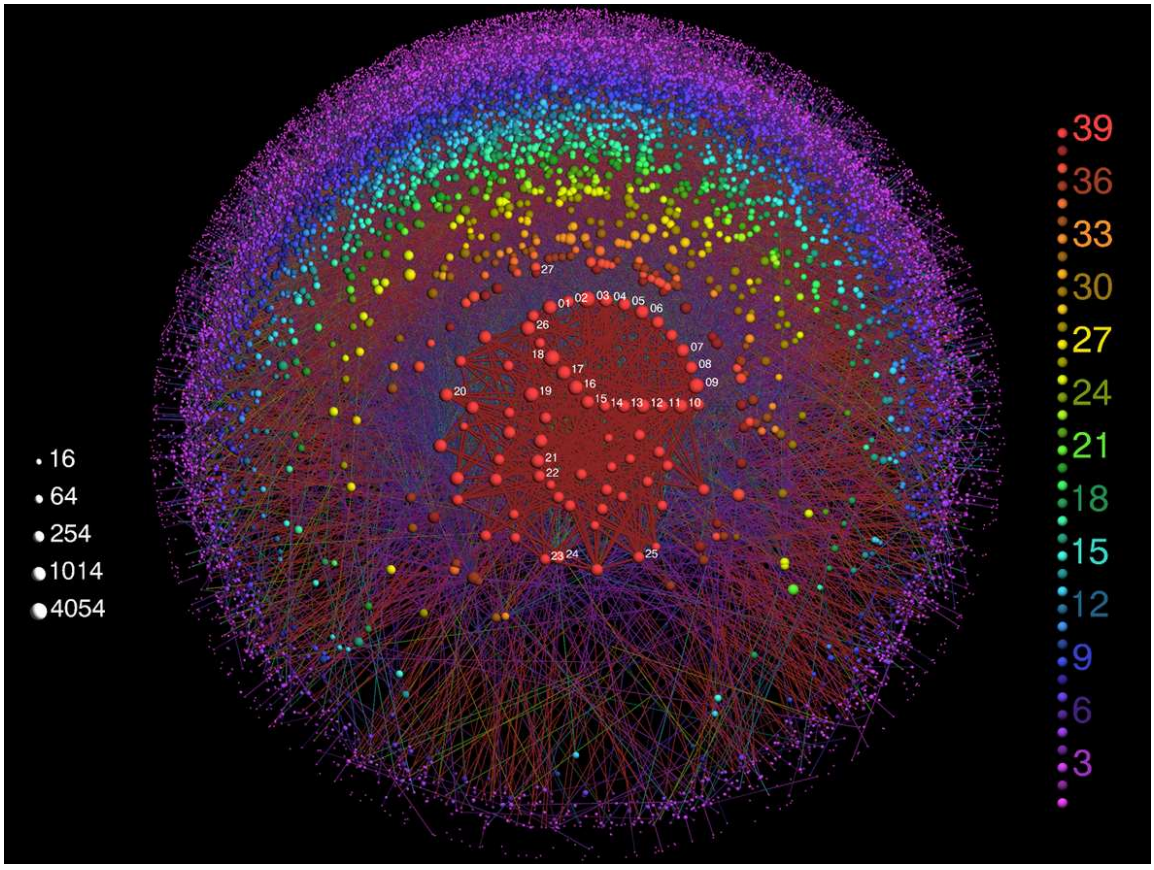

FIGURE 1. Visualization of AS Internet map by LaNet-vi. Vertices in the maximum clique are labeled with numbers.

obtained by LaNet-vi [Beiró et al., 2008]. This visualization is based on k-core decomposition. A k-core is a the maximum induced subgraph such that all vertices have at least degree k [Seidman, 1983, Bollobás, 1984]. LaNet-vi paints each vertex with the rain-bow colors according to its *shell index*, i.e., the maximum core that a vertex belongs to. It also makes a greedy clique decomposition of the top core, i.e., the core with maximum k, placing each clique in circular sector according to its size.

In this graph the Equation 2 found a K_{27} while LaNet-vi found a K_{24}, this is displayed as the largest circular sector of red vertices in Figure 1. Moreover, this figure shows vertices of the K_{27} as those enumerated from 01 to 27. It is possible to appreciate that vertex 27 is not at the top core, therefore it is impossible for LaNet-vi to find this clique.

Comparing the execution time of this graph and a ER graph of the same size, e.g., the same number of edges, we find that AS graph ends quicker than the ER graph, due to that its degree distribution follows a power law with $\beta \simeq 2.2$.

4. Discussion

As we have already remarked this problem is NP-Complete. If Theorem 1 is correct, this will lead to $P = NP$. Due to the importance of such problem, we hope that the scientific community confirms its correctness, or eventually find a counterexample.

References

[Alvarez-Hamelin, 2011] Alvarez-Hamelin, J. I. (2011). `find_cliques`: a library for graphs in phyton: `http://findcliques.sourceforge.net/`.

[Beiró et al., 2008] Beiró, M. G., Alvarez-Hamelin, J. I., and Busch, J. R. (2008). A low complexity visualization tool that helps to perform complex systems analysis. *New J. Phys*, 10(12):125003.

[Bollobás, 1984] Bollobás, B. (1984). The evolution of sparse graphs. *Graph Theory and Combinatorics*, pages 35–57.

[Bollobás, 2001] Bollobás, B. (2001). *Random Graphs*. Cambridge University Press.

[Bollobás and Erdös, 1976] Bollobás, B. and Erdös, P. (1976). Cliques in random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 80:419–427.

[Bomze et al., 1999] Bomze, I., Budinich, M., Pardalos, P., and Pelillo, M. (1999). The maximum clique problem. In Du, D.-Z. and Pardalos, P. M., editors, *Handbook of Combinatorial Optimization, volume 4*. Kluwer Academic Publishers.

[CAIDA, 1998] CAIDA (1998). Cooperative Association for Internet Data Analysis , Router-Level Topology Measurements. `http://www.caida.org/tools/measurement/skitter/`.

[Erdös and Rényi, 1959] Erdös, P. and Rényi, A. (1959). On random graphs I. *Publ. Math. (Debrecen)*, 6:290–297.

[Johnson, 1973] Johnson, D. S. (1973). Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, STOC '73, pages 38–49, New York, NY, USA. ACM.

[Karp, 1972] Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R. and Thatcher, J., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

[Seidman, 1983] Seidman, S. B. (1983). Network structure and minimum degree. *Social Networks*, 5:269–287.

INTECIN (UBA-CONICET), Facultad de Ingeniería, Paseo Colón 850, C1063ACV Buenos Aires – Argentina

E-mail address: `ignacio.alvarez-hamelin@cnet.fi.uba.ar`