Some remarks on the Clique problem

THỊNH NGUYỄN
Moscow State University
kosmofarmer@gmail.com
June 22, 2018

Abstract

We survey some information about the clique problem - one of the cornerstone of the field of research in theory of algorithms. A short note at the end of the paper should be read with care on the reader’s own.

I. Introduction

An Independent Set in a graph is a set of nodes no two of which have an edge. E.g., in a 7-cycle, the largest independent set has size 3, and in the graph coloring problem, the set of nodes colored red is an independent set. The Independent Set problem is: given a graph G and an integer k, does G have an independent set of size at least k?

We reduce from Max-Clique. Given an instance \((G,k)\) of the Max-Clique problem, we output the instance \((H,k)\) of the Independent Set problem where H is the complement of G. That is, H has edge \((u,v)\) if and only if \(u,v\) are not adjacent in G. Then H has an independent set of size k if and only if G has a k-clique.

II. More on cliques

As mentioned earlier, it is an open problem whether P \(\neq\) NP (though everyone believes they are different). It is also an open problem whether NP \(\neq\) co-NP (though again, everyone believes they are different). One can also define even more expressive classes. For instance, PSPACE is the class of all problems solvable by an algorithm that uses a polynomial amount of memory. Any problem in NP is also in PSPACE, because one way to solve the problem is to take the given instance I and then simply run the verifier V(I,X) on all possible proof strings X, halting with YES if any of the runs outputs YES, and halting with NO otherwise. (Remember, we have a polynomial upper bound on |X|, so this uses only a polynomial amount of space.) Similarly, any problem in co-NP is also in PSPACE. Unfortunately, it is not even known for certain that P \(\neq\) PSPACE (though all the classes mentioned above are believed to be different). One can also define classes that are provably larger than P and NP. For instance, EXPtime is the class of all problems solvable in time \(O(2^n)\) for some constant \(n\). This class is known to strictly contain P. The class NEXPtime is the class of all problems that have a verifier which runs in time \(O(2^n)\) for some constant \(n\). This class is known to strictly contain NP. The class of all Turing-computable problems is known to strictly contain all of the above, and some problems such as the Halting problem (given a program A and an input x, determine whether or not A(x) halts) are not even contained in that!

NP-complete problems have the dual property that they belong to NP and they capture the essence of the entire class in that a polynomial-time algorithm to solve one of them would let you solve anything in NP. We proved that 3-SAT is NP-complete by reduction from Circuit-SAT. Given a circuit C, we showed how to compile it into a 3-CNF formula by using extra variables for each gate, such that the formula produced is satisfiable
if and only if there exists $x$ such that $C(x) = 1$. This means that a polynomial-time algorithm for 3-SAT could solve any problem in NP in polynomial-time, even factoring. Moreover, 3-SAT is a simple-looking enough problem that we can use it to show that many other problems are NP-complete as well, including Max-Clique, Independent Set, and Vertex Cover.

III. Results

What do these two conditions mean? Well (1) implies that the literals $z_{ab}, z_{cd}$ corresponding to the vertices $v_{ab}, v_{cd}$ respectively belong to different clauses $C_a \neq C_c$ in $\mathcal{I}$. The second condition implies that both literals can be satisfied simultaneously. This step of the construction takes $O(m^2)$ time. The final step is to determine the value of $k$; we will set $k$ to be $m$, the number of clauses in $\mathcal{I}$. Now, I claim that $\mathcal{I}$ is satisfiable if and only if $G$ as constructed above has a clique of size at least $k = m$.

IV. Discussion

The set $S = v_1, v_2, \ldots, v_m$ must form an $m$-clique in $G$. Why? Well notice that $z_{ab}, z_{cd} \ldots, z_{i,j}$ all have the same truth assignment, since otherwise $z_{ab}, z_{cd} \ldots, z_{i,j}$ for some $i, j \in \{1, m\}$ thus implying that one of $z_{ab}, z_{cd} \ldots, z_{i,j}$ is not the satisfying literal of $C_i, C_j$, a contradiction to our choice of the $z_{ab}$ literals. Notice also that the $z_{ab}$ literals belong to different clauses, that is how we chose them. Therefore, by the construction of $G$, every pair of $v_1, v_2, \ldots, v_m$ must have a connecting edge and thus $S = v_1, v_2, \ldots, v_m$ forms an $m$-clique in $G$. Conversely, suppose $G$ has a clique of size at least $m = k$. Let $v_1, v_2, \ldots, v_q$ be a clique in $G$ of size $q \leq m$, then the $\mathcal{I}$ must vertices $v_1, \ldots, v_m$ must also form a clique in $G$. Since there are no edges connecting vertices from the same clause, every $v_i$ corresponds to a literal $z_i$ from exactly one clause $C_i$. Moreover, since $v_1, \ldots, v_m$ is a clique, the corresponding literals $z_{i,j}$ of any pair $v_i, v_j \in \{1, m\}$ can be satisfied simultaneously (by construction). Now, to construct a satisfying assignment $x_1, \ldots, x_n$ for $\mathcal{I}$, we just need to satisfy all of $z_1, \ldots, z_m$ and assign the remaining variables arbitrarily. Every $C_i$ contains one $z_i$, and every $z_i$ is satisfiable thus every $C_i$ is satisfiable and so $\mathcal{I}$ is satisfiable. To conclude, we've shown that CLIQUE is in NP and that it is NP-hard by giving a reduction from 3-SAT. Therefore CLIQUE is NP-complete.

V. Nothing

The Maximal Clique Problem

Although many theoretical papers have been published on DNA computing since Adleman’s first crude demonstration in 1994, it would be over two years, in 1997, until another NP complete problem would be solved. This problem was the Maximal Clique Problem: given a group of vertices some of which have edges in between them, the maximal clique is the largest subset of vertices in which each point is directly connected to every other vertex in the subset. Every time a new point is added, the number of total cliques that must be searched at least doubles; hence we have an exponentially growing problem. Once again, researchers sought to take advantage of DNA’s high level parallelism which would essentially allow all the possible paths to be calculated simultaneously. After all the possible cliques were constructed, the scientists would simply need to fish out the largest clique. However, like many DNA experiments, each possible choice needed a unique DNA strand. This is a problem because, as noted early, the number of cliques grows exponentially. In Adleman’s experiment, each city and every connecting path had to be hard-coded, in other words, specially made to order in a biology lab. In a problem with more cities, it would be virtually impossible to manually create every possible path. Ouyang and company designed an algorithm that would only require the manual creation a linearly growing number of DNA strands, 2$N$ to be exact, where $\mathcal{N}$ is the number of ver-
vertices in the graph.

VI. Conclusion

In a (possibly) finite universe, anyone can thrive if (s)he knows about the secret of nature. Whichever is of philosophical concerns is in another dimensional spacey.