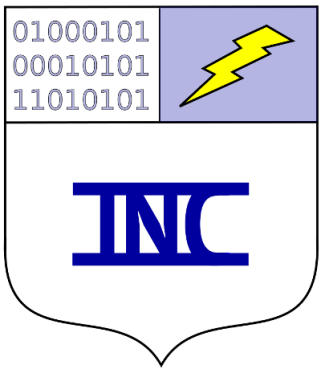
Cliques in Graphs

Brent Kirkpatrick

September 30, 2022

Intrepid Net Computing



www.intrepidnetcomputing.com

Document Revision History

August 19, 2022 Conceived.

August 28, 2022 Drafted.

September 30, 2022 Published original on www.intrepidnetcomputing.com

December 17, 2023 Typographic corrections to the text.

Abstract

The maximum clique problem is of interest precisely because of the symmetries of a clique. This paper presents a polynomial-time algorithm for finding all the maximal cliques of a graph.

In graph theory, many of the interesting problems have brief definitions. This is the case with the maximum clique problem. The maximum clique problem is to find the largest clique in a graph. The decision problem is phrased: does the maximum clique in a graph have k nodes?

The difficulty with the maximum clique problem is that it is easy to produce examples, it is easy to solve examples, and it is difficult to see the algorithm that works. For decades, the community of computer science has worked to write efficient algorithms for NP-hard problems, such as the maximum clique problem. This effort has occupied a generation of computer scientists.

Algorithm

When we look at the symmetries in a graph, we can begin to see a solution for the maximum clique problem. In particular, the breadth first search provides a method for exploring the symmetries of a graph in a useful order. We can use this useful order to begin detecting maximal cliques.

Let the graph $G = (V, E)$ be an undirected graph with edges E and nodes V. The set of nodes is a finite set, and the set of edges is tuples of nodes. Let the **d-splits** be defined as in the solution for graph isomorphism [1]. These will be represented in the sets D. Let $H = (W, F)$ be a new graph in which we will represent all the cliques. The edges in H will be labeled with a unique edge label for each clique and nodes in H will be connected when they are in the same clique. The graph H will contain only cliques. And, exactly the maximal cliques of graph G. The Algorithm 1 gives the breadth first search that creates the graph H.

Algorithm 1 Algorithm for all the maximal cliques in a graph.

Find the highest valence node $v \in V$.
Create new graph $H = (W, F)$
visit$[v] = 0 \; \forall \; v \in V$
Begin our recursion with node v.

SUBROUTINE isClique(v)
visit$[v] = 1$
$D = \emptyset$
$D' = V$
$u' = v$
for all $u =$ children(v) in G **do**
  if visit$[u] \neq 1$ **then**
    $D_u =$ isClique(u)
    $D = D \cup D_u \cup \{u\}$
    $I = D_u \cap D' \cap$ children(v)
    $D' = D_u$
    In H create edge between (u', u) if $I \neq \emptyset$ with $w(u', u) = I$
    $u' = u$
  end if
end for
RETURN $D \cup \{v\}$

Correctness

The proofs of correctness for the maximum clique algorithm rely on a straight forward discussion of breadth first search. The unique aspect of breadth first search is that all the children of a node are visited from that node first. This type of graph search allows us to explore the symmetries of the graph as in the algorithm for graph isomorphism. Proofs of correctness are used to illuminate the aspects of an algorithm that lead to it working properly. For this algorithm, we find that two lemmas are convincing.

Lemma 1. *A same edge label clique in graph H is a clique in graph G.*

Proof. Every clique has a source which is the first node visited in the breadth first search. All children are visited before any other descendants. Every clique has a sink. That sink labels the edge in H. □

Lemma 2. *There are polynomial cliques in a graph.*

Proof. The trivial clique is one edge. A larger clique is a union of edges. There are polynomial edges in a graph. □

With there being polynomial cliques in a graph, we can see that this algorithm runs in polynomial time. The graph H contains the cliques of graph G.

Conclusions

This paper gives a polynomial-time algorithm for finding all the maximal cliques of a graph. One of these maximal cliques is the maximum clique in the graph. This algorithm solves the maximum clique problem.

With a polynomial-time algorithm to a classic NP-complete problem like the maximum clique problem, we have a shift in how computer science is practiced. There are entire categories of proofs that assume that the problem class P does not equal the problem class NP. Very few proofs rely on the assumption that P equals NP, as the proof of such a result is straight forwardly given with an algorithm. This paper gives that algorithm.

References

[1] B. Kirkpatrick. A polynomial-time algorithm for graph isomorphism. *TIPS*, 2016.