Cliques via Cuts: Refuting the Exponential Time Hypothesis with a Subexponential Algorithm for Clique

Alexander Bistagne
Independent Research 2023

August 1, 2023

Abstract

This paper presents a short algorithm which finds a maximum clique in time $2^{O(\sqrt[3]{n^2 log^2(n)})}$. Rather than try to improve the average runtime of the algorithm, this paper attempts to present the simplest algorithm with this runtime, so as to minimize risk of analysis error.

1 Introduction

Off and on now for a while, I have been researching clique problems and PvsNP more generally. My most recent endeavor into clique research was focused on the idea that if one could find good algorithms for clique in both the dense case and sparse case, one might be able to combine them into a good general algorithm for clique. The following algorithm combines one sparse strategy —the inclusion-exclusion principle— with a well known dense strategy —co-disconnected graphs— and a new dense strategy: rough balanced cuts in the complement. The rough balanced cuts are made multiple minimum st cuts.

2 Preliminaries

This paper assumes a basic knowledge of graph theory for sake of brevity. (edges, neighborhoods, vertex degrees, connectedness, graph complements, trees and cuts) This paper uses the notation from [CITATION]

Definition 2.1. *An algorithm has subexponential time if it runs in time $2^{o(n)}$.*

Conjecture 2.2. *[6, abstract and theorem 1] The exponential time hypothesis: The best algorithm for 3-SAT takes $O(2^{\delta n})$ time for some $\delta > 0$*

Theorem 2.3. *[2, Theorem 5.5] The exponential time hypothesis holds if and only if there is no subexponential algorithm for clique*

Theorem 2.4. *If the exponential time hypothesis is false: then NEXPTIME can not be simulated in Nonuniform Polynomial Time.*

Definition 2.5. *[8] [4] A Gomory-Hu Tree is a weighted spanning tree of a graph where the weight of minimum st cuts in T match the minimum weight of st cuts in the original graph*

Theorem 2.6. *A Gomory-Hu Tree always exists [4] and in an unweighted graph, it can be constructed in almost quadratic time [1].*

Theorem 2.7. *Every tree has a balanced cut vertex[7] and it can be found in quadratic time(see appendix A).*

Definition 2.8. *Clique: a clique is a complete graph (the simple graph with every possible edge) which is an induced subgraph of some other graph. The complement of a clique is an independent set. A clique is maximal if it is not the induced subgraph of some other clique in the original graph. First studied by Erdos and Szekeres in 1935 [3]; first named by Luce and Perry in 1949.[9]*

Theorem 2.9. *[10] All Maximal cliques (or Independent sets) can be listed in time polynomial per clique (or independent set)*

Definition 2.10. *The Maximum Clique Search Problem is the problem of finding the largest clique contained in a finite simple undirected graph. First studied by Haray & Ross (1957)[5]*

Remark 2.11. *For sake of brevity, we will assume all basic data manipulations on sets over a finite universe: union, intersection, size take linear time; and, we will assume the readers knowledge of asymptotic analysis.*

3 Algorithm

Theorem 3.1. *Algorithm 1 Cliques via Cuts is a subexponential $2^{O(\sqrt[3]{n^2 \log^2 n})}$ algorithm for the maximum clique search problem.*

Lemma 3.2. *Lines 3-6 of algorithm 1 are recursively correct, and have worst case recursive bound $T(n) \leq T(n-1) + O(n^2)$*

Proof. We will begin by proving the correctness of lines 2-6 of the algorithm. The maximum independent set on a disconnected graph is clearly the disjoint union of the maximum independent set on each of its components. Thus, the maximum clique is the disjoint union of the maximum clique on each of the co-components. The runtime for this case is $T(n) \leq T(a) + T(b) + O(n^2)$ where a + b = n if there are only two components. A similiar bound occurs if there are more than two components. The $O(n^2)$ work bounds the connectivity test and O(n) set operations. In the worst case, b= 1 and this is the chip and conquer recursive bound stated above. □

Lemma 3.3. *Lines 7-11 of algorithm 1 are recursively correct and have worst case recursive bound $T(n) \leq T(n-1) + T(n - \sqrt[3]{n \log n}) + O(n^2)$*

Algorithm 1 Cliques via Cuts

```
procedure CLIQUE ( $G$ )
    max_clique , current_clique $\leftarrow \emptyset, \emptyset$
    if $\bar{G}$ is disconnected then
        for C connected component of $\bar{G}$ do
            max_clique $\leftarrow$ max_clique $\cup$ CLIQUE(C)
        return max_clique
    if $\delta(G) < n - \sqrt[3]{n \log n}$ then
        Find v with degree = $\delta(G)$
        max_clique $\leftarrow$ CLIQUE( $G \setminus v$)
        current_clique $\leftarrow$ CLIQUE( $G \cap N(v)$) $\cup v$
        if $\|max\_clique\| > \|current\_clique\|$ then return max_clique
        return current_clique
    $T \leftarrow$ Gomory_Hu_Tree( $\bar{G}$ )
    $v \leftarrow$ BalancedCutVertex($T$)
    C $\leftarrow \emptyset$
    for edge $uv \in T$ do
        Let cut be the edges represented by $uv$
        $C \leftarrow C \cup cut$
    Let H be ( V(G) , C )
    for maximal independent set I in H do
        current_clique $\leftarrow$ CLIQUE(G[I])
        if size(current_clique) > size(max_clique) then max_clique $\leftarrow$ current_clique
    return max_clique
```

Proof. Here we use the inclusion exclusion principle. Either the vertex v is in the maximum clique and we can restrict our search to v's neighborhood or v is not in the maximum clique and we can safely remove v without effecting the size of the maximum clique. Clearly if we explore both cases, the correct clique is the larger one. Now the degree bound gets us a runtime guarantee of at most $T(n) \leq T(n-1) + T(n - \sqrt[3]{n \log n}) + O(n^2)$ because the connectivity test and degree checks take at most quadratic time. □

Lemma 3.4. *In line 20 of Algorithm 1, G[I] always has co-components of have at most* $\frac{n}{2}+1$ *verticies*

Proof. $G[I]$ is an induced subgraph of $G \cup H$ proof: $G[I] = G[I] \cup (I, \emptyset) = G[I] \cup H[I] = (G \cup H)[I]$ The cuts which isloate v in T must also isolate v in $\bar{G}$ by the gomory hu tree definition2.5, and furthermore the size of the graph on the not v side of a cut in T is the same size as the not v side of a cut in $\bar{G}$ because an st cut in T is an st cut in $\bar{G}$2.5. Thus since the cuts isolate v in T and by theorem 2.7 make each component have at most $\frac{n}{2}+1$ verticies. These edges make $\bar{G} \setminus H$'s components have at most $\frac{n}{2}+1$ verticies. Therefore $G \cup H$'s co-components have at most $\frac{n}{2}+1$ verticies which makes the co-components of G[I] have at most $\frac{n}{2}+1$ verticies. □

Lemma 3.5. *In line 19 of algorithm 1, there are at most* $2^{\sqrt[3]{n^2 \log^2 n}}$ *such maximal independent sets, and does* $O(n^3)$ *work per independent set.*

Proof. Because to reach line 19, line 7 must be false, We know that when line 19 is reached, the minimum degree of G is at least $n - \sqrt[3]{n \log n}$ Thus the maximum degree of $\bar{G}$ is at most $\sqrt[3]{n \log n}$. Because the maximum degree bounds the degree of each vertex and the degree of each vertex bounds its minimum cuts, we know that the edge weight of T is at most $\sqrt[3]{n \log n}$ and because T is a spanning tree of $\bar{G}$, the degree of v is at most $\sqrt[3]{n \log n}$ in T. Therefore the number of edges in H is at most $(n \log n)^{\frac{2}{3}}$. Thus, by choosing only one vertex per edge we can get an upper bound on the number of maximal independent sets as $2^{\sqrt[3]{n^2 \log^2 n}}$. □

Lemma 3.6. *Lines 12-21 of algorithm 1 are recursively correct and have recuresive bound* $T(n) \leq 2^{\sqrt[3]{n^2 \log^2 n}}(T(n/2+1) + O(n^2)) + poly(n)$

Proof. Because H is a cut of $\bar{G}$ it is a subgraph; thus, G is a subgraph of $\bar{H}$. So the maximum clique of G is a subclique of a maximal clique of $\bar{H}$ which is identical to a maximal independent set of H. Thus by checking every maximal independent set of H we must find the maximum clique of G in at least one maximal independent set of H. Now combining this correctness with Lemmas 3.4 and 3.5, and the nature of this third case always feeding back into the first case because of co-disconnectedness. Theorems 2.6 and 2.7 bound our work per case on line 12 and 13, Remark 2.11 bounds 14-18 & 21-22 and we get the third recursive bound of the algorithm. □

Proof. Now we can finish theorem 3.1. The recursive correctness of each section means that by induction the algorithms correctness follows from its correctness on small cases. Clearly the bound in Lemma 3.2 is dominated by the bound in Lemma 3.3. Furthermore, the bound in 3.3 can be made nonrecursive $T(n) \leq T(n-1) + T(n - \sqrt[3]{n \log n}) + O(n^2) \leq \sqrt[3]{n \log n} T(n-$

$\sqrt[3]{n \log n}) + poly(n) \leq 2^{\frac{n \log n}{\sqrt[3]{n \log n}} + O(\log n)} \leq 2^{O(\sqrt[3]{n^2 \log^2 n})}$. Lemma 3.6 has the same upper bound $T(n) \leq 2^{\sqrt[3]{n^2 \log^2 n}}(T(n/2+1) + O(n^2)) + poly(n) \leq 2^{O(\sqrt[3]{n^2 \log^2 n}) \sum_{i=0}^{logn} (\frac{1}{\sqrt[3]{4}})^i} \leq 2^{O(\sqrt[3]{n^2 \log^2 n})}$. Now to bound a recursion tree with mixed cases: Let R be the recursion tree with c,s,d nodes corresponding to the three recursion relations when the graph is co-connected, sparse or dense. Clearly a recursion tree with c,s,d nodes can be upper bounded in size by a recursion tree with only s,d nodes and replace each c node an s node whose new subtree is a pure s recursion tree. Then, a recursion tree with only s, d nodes can be upper bounded by a recursion tree with only s nodes by replacing each d node with s nodes of the appropriate size since the bounds on pure s trees match the bounds on pure d trees. Therefore a recursion tree with all three cases occuring is bounded above by a recursion tree where only the sparse case occurs which has the stated subexponential bound; thus, the theorem follows. □

Corollary 3.7. *The Exponential Time Hypothesis is False; and NEXPTIME is not a subset of Nonuniform Polynomial time*

Proof. Take theorem 3.1 and combine with theorem 2.3 and 2.4 [missing citation about Williams 2010] □

4 Conclusion

I know that claiming this disproof is an very high stakes claim; therefore, if this paper is wrong, please politely refute. If each step in a proof has failure probability $1-p$ and the proof has n steps; then, the proof has validity probability p^n. Thus shorter human written proofs are more convincing. Regardless, I think this paper presents a simple interesting algorithm. I think a lot of interesting heuristics could fit in between lines 6 and 7 of the algorithm, but that was not the goal with this paper. The goal of this paper was to demonstrate my competency in computational complexity and graph theory.

References

[1] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. APMF < apsp? gomory-hu tree for unweighted graphs in almost-quadratic time. *CoRR*, abs/2106.02981, 2021.

[2] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.

[3] George Erds, Paul; Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

[4] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[5] I. C. Harary, F.; Ross. A procedure for clique detection using the group matrix. *Sociometry*, 20(3):205–215, 1957.

[6] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[7] Camille Jordan. Sur les assemblages de lignes. *Journal fr die reine und angewandte Mathematik*, 70(2):185–190, 1869.

[8] Vladimir Kolmogorov. A computational study of gomory-hu construction tree algorithms, 2022.

[9] Albert D. Luce, R. Duncan; Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[10] S. Tsukiyama; M. Ide; I. Ariyoshi; I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.