FINDING A MAXIMUM CLIQUE

Robert Tarjan

TR 72 - 123

March 1972

Department of Computer Science
Cornell University
Ithaca, New York 14850
Abstract

An algorithm for finding a maximum clique in an arbitrary graph is described. The algorithm has a worst-case time bound of $k(1.286)^n$ for some constant $k$, where $n$ is the number of vertices in the graph. Within a fixed time, the algorithm can analyze a graph with 2 3/4 as many vertices as the largest graph which the obvious algorithm (examining all subsets of vertices) can analyze.

Keywords

Algorithm, Clique, Graph, Independent Set

*This research was supported by The Office of Naval Research, Contract No. N00014-67-A-0077-0021.*
Finding a Maximum Clique

Robert Tarjan
Department of Computer Science
Cornell University

Introduction

Let $G = (V, E)$ be a graph with $|V| = n$ vertices. Consider the problem of discovering a clique (a set of vertices which determine a complete subgraph) of maximum size in $G$. Cook [1] has shown that if the clique problem has an algorithm with a time bound polynomial in $n$, then any algorithmically solvable problem has an algorithm with a time bound polynomial in the size of the problem data. Thus any improvement over the obvious clique-finding algorithm is an interesting forward step.

Suppose we examine every subset $S \subseteq V$ to see if $S$ determines a clique, and then we choose the largest clique found. This is the obvious algorithm. Since $V$ has $\mathcal{P}(V) = 2^n$ subsets, the algorithm has a time bound of $O(n2^n)$. However, the algorithm may be improved.

A Fast Algorithm for Finding a Maximum Clique

If we do not examine all subsets of $V$, but only a sufficiently large number of them, we may get a faster method for determining a maximum clique. The basic idea is partition $V$ into two sets, $S$ and $V - S$. Let $G_S$ and $G_{V-S}$ be the subgraphs of $G$ determined by these two vertex sets. Then any clique in $G$ determines a clique in $G_S$ and a clique in $G_{V-S}$. Further, any clique $C$ in $G_S$
may be combined with any clique in $G_{A(C)-S}$ to give a clique in $G$, if $A(C)$ is the set of vertices adjacent to one or more vertices in $C$. By finding each clique in $G_S$, solving a corresponding clique problem in $G_{V-S}$, and combining all these solutions, we may find a maximum clique in $G$.

A lemma will state this result more precisely. If $S \subseteq V$, let $G_S$ be the subgraph of $G$ with vertex set $S$. Let $A(S)$ be the set of vertices adjacent to one or more vertices in $S$. Finally, let $||G||$ be the size of a maximum clique in $G$.

**Lemma 1:** Let $G = (V, E)$ be a graph. Let $S \subseteq V$. Then:

\[
(I) \quad ||G|| = \max \left\{ |C| + ||G_{A(C)-S}|| \right\}
\]

C a clique in $G_S$

**Proof:** If $X$ is a clique in $G$, $X \cap S$ is a clique in $G_S$, and $X \cap (V-S)$ is a clique in $G_{A(X)\cap S}$. Expression (I) is then immediate.

In fact, the maximum in (I) need not be taken over all cliques in $G_S$ but only over a subset of them. Let $S \subseteq G$ and let $X, Y$ be cliques in $G_S$. Suppose that $||G_Y \cup (A(Y)-S)|| \leq ||G_X \cup (A(X)-S)||$. Then $X$ is said to dominate $Y$. A set of cliques $\mathcal{C} \subseteq \mathcal{G}(S)$ is said to be dominant in $G_S$ if every clique in $G_S$ is dominated by at least one clique in $\mathcal{C}$. Dominance is a transitive relation. A clique $X$ may be shown to dominate a clique $Y$ by giving a simple method of transforming any clique in $G_Y \cup (A(Y)-S)$ into a clique of equal size in $G_X \cup (A(X)-S)$. For instance, suppose that if $C$ is a clique in $G_{\lambda(X)-S}$, then
\((C \cap (\Lambda(X) - S)) \cup X \) is always a clique as large as \(C\). Then \(X\) dominates \(Y\).

**Lemma 2**: Let \(S \subseteq V\). Let \(\mathcal{C}\) be a dominant set of cliques in \(G_S\). Then:

\[
\|G\| = \max_{C \in \mathcal{C}} \{ |C| + \| G_{A(C) - S} \| \}
\]

**Proof**: Let \(Y\) be a clique in \(G_S\). Then some clique \(X \in \mathcal{C}\) dominates \(Y\) in \(G_S\). If \(Z\) is a clique in \(G_{Y \cup (\Lambda(Y) - S)}\), there is a clique at least as big as \(Z\) in \(G_X \cup (\Lambda(X) - S)\). Thus the maximum in (I) need only be taken over the dominant set of cliques \(\mathcal{C}\).

Thus to find a maximum clique in \(G\), we carefully choose a subset \(S\) of vertices, and we solve one smaller clique problem for each clique in a dominant set of cliques for \(G_S\). The procedure is applied recursively to solve the subproblems. The set \(S\) depends on the nature of \(G\); thus the algorithm has several cases. (In one case, the clique problem is solved directly.) Exposition of the cases is tedious; we shall skip details in a few places.

The entire algorithm has a time bound \(t(n) = kb^R\) for some constant \(b\) and \(k\). We shall calculate \(b\) separately for each case; the maximum of these values will give a bound for the complete algorithm.
The Possible Subproblems

The function $t_1(n)$ is a time bound for the algorithm if case (i) always applies.

1) If $G$ contains a vertex $v$ of degree $n-1$ or $n-2$, let $S = \{v\} \cup (V-A(v))$. Clique $\{v\}$ dominates all cliques in $G_s$. Thus $|G| = 1 + |G_{V-S}|$ and only one subproblem must be solved. If this case applies, $t_1(n) = t_1(n-1) + p(n)$ for some polynomial $p(n)$.

2) Suppose $G$ contains only vertices of degree $n-3$. Then $\overline{G}$, the complement graph of $G$, consists exclusively of cycles. We may easily find a maximum set of independent (pairwise non-adjacent) vertices in $\overline{G}$. Such a set is a maximum clique in $G$. If this case applies, $t_2(n) = p(n)$ for some polynomial $p(n)$.

3) If $G$ contains a vertex $v$ of degree $n-3$ and a non-adjacent vertex of degree $n-4$ or less, let $S = \{v\} \cup (V-A(\{v\})) = \{v_1, w_1, w_2\}$. If $(w_1, w_2) \notin E$, there is one subproblem of size $n-3$. If $(w_1, w_2) \in E$, there are two subproblems, one of size $n-3$ and one of size $|A(\{w_1, w_2\}) - S| \leq n-5$. In the worst case $t_3(n) = t_3(n-3) + t(n-5) + p(n)$ for some polynomial $p(n)$, and $t_3(n) = (1.17)^n$, ignoring constants and polynomial terms.
(4) If $G$ contains a vertex $v$ of degree $n-4$, let $S = \{v\} \cup (V - A(v)) = \{v, w_1, w_2, w_3\}$. Let $A_i = A(\{w_i\}) - S$, for $i = 1, 2, 3$. The subproblems depend on the subgraph $G_S$ and the $A_i$.

(4a) $G_S = \ldots \quad |G| = 1 + |G_{V-S}|$. There is one subproblem of size $n-4$. $t_{4a}(n) = t_{4a}(n-4) + p(n)$ for some polynomial $p(n)$.

(4b) $G_S = \begin{array}{c} v \\ w_1 & w_2 & w_3 \end{array}$. If $|A_2 \cap A_3| = n-5$, there is one subproblem of size $n-5$. If $|A_2 \cap A_3| \leq n-6$, there are two subproblems, one of size $n-4$ and one of size $|A_2 \cap A_3|$. In this case $t_{4b}(n) = t_{4b}(n-4) + t_{4b}(n-6) + p(n)$ for some polynomial $p(n)$, and $t_{4b}(n-4) = (1.15)^n$, ignoring constants and polynomial terms.

(4c) $G_S = \begin{array}{c} v \\ w_1 & w_2 & w_3 \end{array}$. If $|A_1 \cap A_2| \leq |A_2 \cap A_3| = n-6$, there are two subproblems, one of size $n-4$ and one of size $n-6$. In this case $t_{4c}(n) = t_{4c}(n-4) + t_{4c}(n-6) + p(n)$ and $t_{4c}(n) = (1.15)^n$.

If $|A_1 \cap A_2| \leq |A_2 \cap A_3| \leq n-7$, there are three subproblems. In this case $t_{4c}(n-4) + 2t_{4c}(n-7) + p(n)$ and $t_{4c}(n) = (1.22)^n$.

(4d) $G_S = \begin{array}{c} v \\ w_1 & w_2 & w_3 \end{array}$ There are several cases, depending upon $|A_1 \cap A_2 \cap A_3|$.

If $|A_1 \cap A_2 \cap A_3| \geq n-7$, there are two subproblems, one of size $n-4$ and one of size $n-7$. $t_{4d}(n) = t_{4d}(n-4) + t_{4d}(n-7) + p(n)$ for some polynomials $p(n)$, and $t(n) = (1.14)^n$. 
If \( |A_1 \cap A_2 \cap A_3| = n-8 \), there are at most three subproblems, of sizes \( n-4 \), \( n-6 \), and \( n-8 \).

\[
t_{4d}(n) = t_{4d}(n-4) + t_{4d}(n-6) + t_{4d}(n-8) + p(n),
\]
and \( t_{4d}(n) = (1.215)^n \).

If \( |A_1 \cap A_2 \cap A_3| = n-9 \), there are at most three subproblems, of sizes \( n-4 \), \( n-6 \), and \( n-9 \). This case is better than the one just above.

If \( |A_1 \cap A_2 \cap A_3| \geq n-10 \), there may be five subproblems, one of size \( n-4 \), three of size \( n-8 \), and one of size \( n-10 \). In this case

\[
t_{4d}(n) = t_{4d}(n-4) + 3t_{4d}(n-8) + t_{4d}(n-10) + p(n)
\]

for some polynomial \( p(n) \), and \( t_{4d}(n) = (1.26)^n \).

(5) If \( G \) contains a vertex \( v \) of degree \( n-6 \), let \( S = \{v\} \). There are two subproblems, one of size \( n-1 \) and one of size \( n-6 \).

\[
t_5(n) = t_5(n-1) + t_5(n-6) + p(n) \quad t_5(n) = (1.286)^n
\]

(6) If \( G \) contains a vertex of degree \( n-5 \), let

\[
S = \{v\} \cup (V - A(v)) = \{v, w_1, w_2, w_3, w_4\}
\]

Let \( A_i = A(w_i) - S \), for \( i = 1, 2, 3, 4 \). The subproblems depend upon the subgraph \( G_S \) and the \( A_i \).

(6a) \( G_S = \begin{array}{c} v \end{array} \begin{array}{c} w_1 \end{array} \begin{array}{c} w_2, w_3, w_4 \end{array} \) . Any possible set of subproblems is better than some set of subproblems which arises in case (4).

(6b) \( G_S = \begin{array}{c} v \end{array} \begin{array}{c} w_1 \end{array} \begin{array}{c} w_2, w_3, w_4 \end{array} \) \text{ If } |A_2 \cap A_3| = n-7, \text{ there are two subproblems, of sizes } n-5 \text{ and } n-7. \text{ If } |A_2 \cap A_3| \leq n-8 \text{ but } |A_1 \cap A_2| = n-7, \text{ there may be three subproblems, one of size}
n-5 and two of size n-7. In this case
\[ t_{6b}(n) = t_{6b}(n-5) + 2t_{6b}(n-7) + p(n) \quad \text{and} \quad t_{6b}(n) = (1.21)^n. \]

If \( |A_1 \cap A_2|, |A_2 \cap A_3|, |A_3 \cap A_4| \leq n-8 \), there may be four subproblems. In this case, \( t_{6b}(n) = t_{6b}(n-5) + 3t_{6b}(n-8) + p(n) \), and \( t_{6b}(n) = (1.22)^n \).

(6c) \( G_S = \)

\[ \begin{array}{c}
\text{If } |A_1 \cap A_2| = n-8, \text{ there may be at most four subproblems. A recursive bound on } t(n) \text{ in all cases is:}
\end{array} \]
\[ t_{6c}(n) = t_{6c}(n-5) + t_{6c}(n-7) + t_{6c}(n-8) + t_{6c}(n-10) + p(n), \quad \text{and} \quad t_{6c}(n) = (1.21)^n. \]

If \( |A_2 \cap A_3 \cap A_4| \geq n-9 \), there are at most three subproblems, and the bound above works in all cases.

If \( |A_2 \cap A_3 \cap A_4| = n-10 \), there are at most four subproblems, and the bound above works in all cases.

Suppose \( |A_2 \cap A_3 \cap A_4| = n-11 \). In the worst case there are five subproblems. A Venn diagram illustrates the situation.

\[ t_{6c}(n) = t_{6c}(n-5) + 3t_{6c}(n-9) + t_{6c}(n-11) + p(n) \]
\[ t_{6c}(n) = (1.22)^n \]

Suppose \( |A_2 \cap A_3 \cap A_4| = n-12 \). In the worst case there are six subproblems. A Venn diagram illustrates the situation.

\[ t_{6c}(n) = t_{6c}(n-5) + 2t_{6c}(n-9) + 2t_{6c}(n-10) + t_{6c}(n-12) + p(n) \]
\[ t_{6c}(n) = (1.24)^n \]
(6d) $G_S = \begin{array}{c} w_1 \\

\quad w_2 \\

\quad \quad w_3 \\

\quad \quad \quad w_4 \end{array}$ If $|A_1 \cap A_4| = n-7$, there are at most three subproblems. $t_{6d}(n-5) + 2t_{6d}(n-7) + p(n)$, and $t_{6d}(n) = (1.20)^n$.

If $|A_1 \cap A_2| \leq |A_2 \cap A_3| \leq |A_3 \cap A_4| \leq |A_4 \cap A_1| \leq n-8$, there may be five subproblems. $t_{6d}(n) = t_{6d}(n-5) + 4t_{6d}(n-8) + p(n)$, and $t_{6d}(n) = (1.25)^n$.

(6e) $G_S = \begin{array}{c} w_1 \\

\quad w_2 \\

\quad \quad w_3 \\

\quad \quad \quad w_4 \end{array}$ If $|A_2 \cap A_4| = n-8$, there are at most four subproblems, of sizes $n-5$, $n-8$, $n-10$, $n-10$.

If $|A_1 \cap A_2| = n-8$, there are at most four subproblems, of sizes $n-5$, $n-8$, $n-8$, $n-10$.

If $|A_1 \cap A_2 \cap A_4| = n-9$, there are at most two subproblems.

If $|A_1 \cap A_2 \cap A_4| = n-10$, there are at most five subproblems.

$t_{6e}(n) = t_{6e}(n-5) + 2t_{6e}(n-9) + t_{6e}(n-10) + t_{6e}(n-11) + p(n)$.

$t_{6e}(n) = (1.22)^n$.

If $|A_1 \cap A_2 \cap A_4| = |A_2 \cap A_3 \cap A_4| = n-11$, there are at most seven subproblems. $t_{6e}(n) = t_{6e}(n-5) + 4t_{6e}(n-9) + 2t_{6e}(n-11) + p(n)$.

$t_{6e}(n) = (1.26)^n$.

If $|A_1 \cap A_2 \cap A_4| = n-11$, $|A_2 \cap A_3 \cap A_4| \leq n-12$, there are at most seven subproblems. The bound above applies in this case.

If $|A_1 \cap A_2 \cap A_4| \leq |A_2 \cap A_3 \cap A_4| \leq n-12$, there are at most eight subproblems. A Venn diagram illustrates the situation, which is symmetric for $w_1$, $w_2$, $w_4$, and for $w_2$, $w_3$, $w_4$. 
\[ t_{6e}(n) = t_{6e}(n-5) + 5t_{6e}(n-10) + 2t_{6e}(n-12) + F(n). \]
\[ t_{6e}(n) = (1.26)^n \]

\[(6f) \quad G_S = V \quad w_1 \quad w_2 \quad w_3 \quad w_4 \quad \text{The situation is now really complicated.}\]

Cases (6f)-(6k) handle the possibilities. Suppose some vertex \(w_5\) is non-adjacent to \(w_1, w_2, w_3,\) and \(w_4\). Then let \(S = \{v, w_1, w_2, w_3, w_4, w_5\}\).

We now use the fact that \(G_S = V\) since case (5) does not apply, all vertices are of degree \(n-5\). Thus clique \(\{v, w_5\}\) dominates all cliques in \(G_S\) except those containing three or more vertices.

If \(|A_1 \cap A_2 \cap A_3 \cap A_4| \geq n-13\), there can be at most five subproblems. Case (4d) has a worse bound than this case.

If \(|A_1 \cap A_2 \cap A_3 \cap A_4| \leq n-14\), there can be six subproblems. In the worst case, \(t_{6f}(n) = t_{6f}(n-6) + 4t_{6f}(n-12) + t_{6f}(n-14) + p\).

Several cases are worse than this one.

\[(6g) \text{ No vertex except } v \text{ is non-adjacent to } w_1, w_2, w_3, \text{ and } w_4, \text{ and } |A_1 \cap A_2| = n-8. \text{ If } |A_1 \cap A_3 \cap A_4| \geq n-11, \text{ there are at most four subproblems, and } t_{6g}(n) = t_{6g}(n-5) + t_{6g}(n-8) + t_{6g}(n-9) + t_{6g}(n-11) + p(n). \text{ Case(6d) above is worse.}\]
If $|A_1 \cap A_3 \cap A_4| \leq n-12$, there may be six subproblems, and
\[ t_{6g}(n) = t_{6g}(n-5) + t_{6g}(n-9) + 3t_{6g}(n-10) + t_{6g}(n-12) + p(n). \]
Case (6e) above is worse.

(6h) $|A_1 \cap A_2| = n-9$. In this case cliques $\{w_1, w_3\}$ and $\{w_2, w_3\}$
are dominated by $\{w_1, w_2, w_3\}$. Similarly $\{w_1, w_4\}$ and $\{w_2, w_4\}$
are dominated by $\{w_1, w_2, w_4\}$. There are at most eight subproblems,
and $t_{6h}(n) = t_{6h}(n-5) + 2t_{6h}(n-9) + 4t_{6h}(n-11) + t_{6h}(n-13) + p(n)$.
$t_{6h}(n) = (1.25)^n$. All cases with fewer than eight subproblems
are better than case (6e).

(6i) We may now assume that $|A_i \cap A_j| \leq n-10$ for all $i \neq j$.
Suppose $|A_1 \cap A_2 \cap A_3| \geq n-11$. Then there are at most nine subproblems.
$t_{6i}(n) = t_{6i}(n-5) + 4t_{6i}(n-10) + 3t_{6i}(n-12) + t_{6i}(n-14) + p(n)$. $t_{6i}(n) = (1.26)^n$.

If $|A_1 \cap A_2 \cap A_3 \cap A_4| \geq n-13$, then there are at most eight subproblems.
$t_{6i}(n) = t_{6i}(n-5) + 6t_{6i}(n-10) + t_{6i}(n-12) + p(n)$.
$t_{6i}(n) = (1.26)^n$.

(6j) $|A_1 \cap A_2 \cap A_3 \cap A_4| = n-14$. Consider the Venn diagram below.

![Venn Diagram]

This situation is impossible, since
every vertex in $V-S$ is adjacent to
$w_1$, $w_2$, $w_3$, or $w_4$. Thus at least one
3-clique in $S$ in non-dominant, and
at least one 2-clique as well. If
only one of the 3-cliques is non-
dominant, the worst situation is:
If two of the 3-cliques are non-dominant, the worst situation is:

\[ t_{6j}(n) = t_{6j}(n-5) - 4t_{6j}(n-10) + 3t_{6j}(n-12) + t_{6j}(n-14) + p(n). \]

\[ t_{6j}(n) = (1.25)^n \]

\[ t_{6j}(n) = t_{6j}(n-5) + 3t_{6j}(n-10) + 2t_{6j}(n-12) + t_{6j}(n-14) + p(n), \] which gives a better bound than above.
\[(6k) \quad |A_1 A_2 A_3 A_4| \leq n-15. \quad \text{The worst case is:}\]

\[
t_{6k}(n) = t_{6k}(n-5) + 3t_{6k}(n-10) + 3t_{6k}(n-11) + t_{6k}(n-12) + 3t_{6k}(n-13) + t_{6k}(n-15) + p(n).
\]

\[
t_{6k}(n) = (1.28)^n
\]

These are the only possible cases we need to consider. Whatever the form of \( G \), the clique problem must be reducible in one of the ways described above. By applying the reductions recursively, we may find a maximum clique in \( G \). The cases may look complicated, but the algorithm can be implemented as a straightforward backtracking procedure; deciding between cases does not require too deep a decision tree, or too much extra work. (The polynomial \( p(n) \leq kn^2 \) in all cases.)

**A Time Bound**

Let \( t(n) \) be the time required to find a maximal clique in a graph with \( n \) vertices using the algorithm outlined above. \( \max t_i(n) \) gives an upper bound for \( t(n) \). The maximum occurs in case (5). Thus \( t(n) \leq (1.286)^n \), ignoring polynomial terms. This bound is asymptoti-
cally correct; if we multiply by a constant the bound is correct for all \( n \). Thus for some \( k \), \( t(n) \leq k(1.286)^n \). Since \( \log_2(1.286) = .364 \), \( t(n) \leq k2^{.364n} \). Within a fixed time, the recursive algorithm can handle a graph with about 2 3/4 as many vertices as the obvious algorithm can handle.

**Conclusions**

A recursive algorithm for finding a maximal clique in a graph has been described. The algorithm has a worst-case time bound of \( k(1.286)^n \) for some constant \( k \), if \( n \) is the number of vertices in the graph. This algorithm is a substantial improvement over the obvious algorithm. It is not clear whether the algorithm can be improved much more, or whether there is a non-exponential time algorithm for finding a maximal clique.
REFERENCES