SIAM J. DISCRETE MATH.
Vol. 18, No. 2, pp. 219–225

APPROXIMATING MAXIMUM CLIQUE BY REMOVING SUBGRAPHS*

URIEL FEIGE†

Abstract. We show an algorithm that finds cliques of size $(\log n/\log\log n)^2$ whenever a graph has a clique of size at least $n/(\log n)^b$ for an arbitrary constant b. This leads to an algorithm that approximates max clique within a factor of $O(n(\log\log n)^2/(\log n)^3)$, which matches the best approximation ratio known for the chromatic number. The previously best approximation ratio known for max clique was $O(n/(\log n)^2)$.

Key words. approximation algorithm, clique, independent set

AMS subject classifications. 05C69, 05C85, 68W25

DOI. 10.1137/S089548010240415X

1. Introduction.

Max clique is the problem of finding a clique of maximum size in an input graph. This problem is NP-hard. An algorithm is said to have approximation ratio ρ for max clique if, on every graph, it is guaranteed to find a clique whose size is at most a factor of ρ smaller than that of the maximum clique. We allow ρ to grow as a function of n (the number of vertices in the input graph). Håstad [8] shows that for every $\epsilon > 0$ there is no polynomial algorithm that approximates max clique within a ratio of $n^{1-\epsilon}$, unless NP has expected polynomial time algorithms. (See [10] for additional information in this respect.) The best approximation ratio known for max clique was $O(n/(\log n)^2)$ by Boppana and Halldorsson [4].

The problem of max independent set is strongly related to the max clique problem (by complementing the graph), and hence shares the same approximation ratio. The chromatic number of a graph is the smallest number of independent sets that cover all vertices of the graph. It shares essentially the same hardness of approximation results as max clique [5, 10] (this is an empirical observation rather than a theorem). In terms of approximation algorithms, Halldorsson [6] shows that the chromatic number can be approximated within a ratio of $O(n(\log\log n)^2/(\log n)^3)$, which is better than the best approximation ratio known for max clique.

In this paper we show an algorithm that approximates max clique within a ratio of $O(n(\log\log n)^2/(\log n)^3)$, matching the known approximation ratio for the chromatic number. The technically new ingredient in our result is an algorithm that finds cliques of size $(\log n/\log\log n)^2$ whenever a graph has a clique of size at least $n/(\log n)^b$ for an arbitrary constant b. This algorithm is based on ideas which can be viewed as natural extensions of ideas used by Boppana and Halldorsson [4] and by Berger and Rompel [2].

In section 2 we describe our new algorithm. In section 3 we explain how (combined with ideas from [6]) it leads to an $O(n(\log\log n)^2/(\log n)^3)$ approximation ratio for max clique. In section 4 we discuss possible future research directions.

*Received by the editors March 18, 2002; accepted for publication (in revised form) February 2, 2004; published electronically October 1, 2004. This research was supported by Israel Science Foundation grant 263/02.

http://www.siam.org/journals/sidma/18-2/40415.html

†Department of Computer Science and Applied Mathematics, The Weizmann Institute, Rehovot 76100, Israel (uriel.feige@weizmann.ac.il).

2. The new algorithm. Let $G(V,E)$ be an input graph with n vertices which contains a clique of size n/k. In this graph we wish to find a large as possible clique. For a parameter $t \ll n/k$, we shall give an algorithm that finds a clique of size at least $t(\log_{3k}(n/t)-3)$. The running time of the algorithm is $O(\binom{2kt}{t}n^c)$, where c is some universal constant. For every $b>0$, whenever $k<(\log n)^b$, we can choose $t=\Theta(\log n/\log\log n)$, and then our algorithm finds a clique of size $\Omega((\log n/\log\log n)^2)$ in polynomial time.

In the course of our algorithm, we shall consider vertex induced subgraphs of G.

DEFINITION 2.1. *Let G be a graph with a clique of size n/k. A vertex induced subgraph S is called* poor *if it does not contain a clique of size $|S|/2k$.*

LEMMA 2.2. *Let G be a graph with a clique of size n/k. Let $S_1,S_2,\ldots$ be arbitrary disjoint poor subgraphs of G (with no clique of size $|S_i|/2k$, respectively). Let $G'(V',E')$ be the vertex induced subgraph of G that remains after removing the poor subgraphs. Then $|V'|\geq n/2k$, and G' contains a clique of size at least $|V'|/k$.*

Proof. The union of disjoint poor subgraphs is itself a poor subgraph. Any subgraph of G contains at most n vertices. Hence the poor subgraph cannot contain a clique larger than $n/2k$. As G has a clique of size n/k, at least $n/2k$ of the clique vertices must remain in G', proving $|V'|\geq n/2k$.

Removing a poor subgraph from G increases the relative density of the maximum clique in the remaining graph. Hence G' contains a clique of size at least $|V'|/k$. ∎

Our algorithm works in phases. The input to a phase is a vertex induced subgraph $G'(V',E')$ of G. (The input to the first phase is the graph G itself.) This subgraph contains a clique of size $|V'|/k$. A phase is completed when one of the following two conditions hold:

1. A clique of size $t\log_{3k}(|V'|/6kt)$ is found.
2. A poor subgraph is found.

If upon finishing a phase the first condition holds, then the algorithm terminates. If upon finishing a phase the second condition holds, then the poor subgraph is removed from G' and a new phase begins with the resulting graph. From Lemma 2.2 it follows that the invariant that the input graph contains a clique of size $|V'|/k$ is maintained when moving from phase to phase. Moreover, by removing poor subgraphs, $|V'|$ cannot drop below $n/2k$, and hence eventually the algorithm must terminate and output a clique of size at least $t\log_{3k}(n/12k^2t)>t(\log_{3k}(n/t)-3)$.

It remains to show how a single phase is performed. Recall that the input to a phase is a graph $G'(V',E')$ which has a clique of size at least $|V'|/k$. The location of the clique is unknown to the algorithm, but the value of k is known. The algorithm has a parameter t. The larger the t, the larger the size of the clique eventually found. However, the running time of the algorithm also increases with t, and eventually we shall choose $t=\log n/\log\log n$ to balance these two factors.

Each phase has several iterations. The input to an iteration is a subgraph $G''(V'',E'')$ of G' and a set of vertices C that form a clique in $V'\setminus V''$. When the iteration ends, either the set C grows (and V'' shrinks), or V'' is declared as poor. In the first iteration $G''=G'$ and C is empty. We now describe a single iteration:

1. If $|V''|<6kt$, end the phase and output C.
2. Partition V'' into disjoint parts, each with $2kt$ vertices. (For simplicity we assume that $2kt$ divides $|V''|$. The algorithm can easily be modified to handle the case that this is not so with negligible effect on the size of the final clique output by the algorithm.)

3. In each part P_i, consider all possible subsets S_{ij} of vertices of cardinality t. (Namely, for every part P_i, for every subset j, $|S_{ij}| = t$.)
4. Let $N(S_{ij})$ be the set of vertices in $V'' \setminus S_{ij}$ that are neighbors in G'' to every vertex in S_{ij}. (Hence $S_{ij}, N(S_{ij})$ form the two sides of a complete bipartite graph in G''.) Call S_{ij} *good* if the subgraph of G'' induced on S_{ij} is a clique and $|N(S_{ij})| \geq |V''|/2k - t$.
5. If some set S_{ij} described above is good, then $C = C \bigcup S_{ij}$, and go to the next iteration with the subgraph induced on $N(S_{ij})$ serving as the new G''.
6. If no set S_{ij} is good, then declare V'' poor, and end the phase.

We first analyze the running time of a phase. The number of iterations in a phase is clearly bounded by $|V'|/t$, as each iteration removes at least t vertices from V'. The number of parts considered in an iteration is $|V''|/2kt$. In each part there are $\binom{2kt}{t}$ subsets to consider. For each subset, the test of whether it is good or not takes polynomial time. Hence the whole phase takes polynomial time if $\binom{2kt}{t}$ is polynomial in n. This condition governs the choice of t. We shall be interested in the case where $k \leq (\log n)^b$ for some constant $b > 0$, in which case we can take $t = \log n / \log\log n$, ensuring a polynomial running time.

We now analyze the output of a phase.

LEMMA 2.3. *If a phase declares a set V'' poor, then indeed the subgraph of G induced on V'' does not contain a clique of size $|V''|/2k$.*

Proof. Assume that the subgraph induced on V'' contains a clique of size $|V''|/2k$. Then by the pigeon-hole principle, at least one of the parts P_i will contain at least t vertices from this clique. The subset that corresponds to these t vertices must be good (it is a clique and has the rest of the clique vertices as its neighbors), and hence V'' will not be declared poor. ∎

Note that Lemma 2.3 does not claim an if and only if relation. Step 5 of an iteration may succeed even if the subgraph induced on V'' does not contain a clique of size $|V''|/2k$, and then the algorithm does not declare V'' poor.

LEMMA 2.4. *If a phase ends by outputting the set C, then this set contains at least $t \log_{3k}(n'/6kt)$ vertices, and these vertices form a clique in G'.*

Proof. Each iteration of the phase adds t vertices to C. To lower bound the number of iterations in a phase, let n'' denote the number of vertices in the beginning of an iteration. Then the next iteration starts with at least $n''/2k - t$ vertices. When $t < n''/6k$, then this number is at least $n''/3k$. Hence the number of iterations needed to reduce $|V''|$ from $|V'|$ to $6kt$ is at least $\log_{3k}(|V'|/6kt)$. This gives the desired lower bound on the number of vertices in C.

The fact that the vertices of C form a clique in G' (and hence also in G) follows from the fact that each subset of vertices that is added into C is a clique and makes a complete bipartite graph with all vertices added after it. ∎

3. An $O(n(\log\log n)^2/(\log n)^3)$ approximation ratio. Without loss of generality we assume that the approximation algorithm for max clique knows the size of the maximum clique in the input graph. (There are only n possible values to try out, or even only $\log n$, as it suffices for our purpose to know the size within a factor of 2.) We divide possible maximum clique sizes into three ranges, applying a different algorithm in each case.

If the maximum clique size is below $n/(\log n)^3$, simply output a single vertex, achieving an $O(n/(\log n)^3)$ approximation ratio. If the maximum clique size is above $n/(\log n)^3$, the algorithm presented in section 2 finds in polynomial time a clique of size $\Omega((\log n/\log\log n)^2)$. This gives an $O(n(\log\log n)^2/(\log n)^3)$ approximation

ratio for max clique whenever the size of the maximum clique is $O(n/\log n)$. If the maximum clique size is above $n/\log n$, we use a modified version of our algorithm, as described below, so as to find cliques of size larger than $(\log n/\log\log n)^2$.

The key to the improvement is the use of a specialized algorithm for finding large cliques in graphs that have cliques of size larger than $2n\log\log n/\log n$. For this purpose we shall use the algorithm of Boppana and Halldorsson [4]. (Potentially, the more complicated algorithm of Alon and Kahale [1] can be used here instead of [4].)

The algorithm of [4] is based on the known fact from Ramsey theory that any graph on $n = \binom{s+r-2}{s-1}$ vertices contains either an independent set of size r or a clique of size s. Moreover, there is an efficient algorithm for finding one of the two. In the context of approximating clique, finding a clique of size s may be the desirable event of the algorithm, whereas finding an independent set of size r can serve as the event of discovering a poor subgraph (in the terminology of our paper, provided that the input graph has a clique of size greater than n/r), and this subgraph can be removed. We shall use the following proposition regarding the performance guarantee of the algorithm of [4]. (For a proof, see [6], for example.)

PROPOSITION 3.1. *In a graph that has a clique larger than* $2n\log\log n/\log n$*, the algorithm of* [4] *produces a clique of size at least* $(\log n)^3/6\log\log n$.

The above immediately implies an $O(n(\log\log n/\log n)^3)$ approximation algorithm for max clique. If the input graph has a clique larger than $2n\log\log n/\log n$, use the algorithm of [4]. Otherwise, use our algorithm from section 2.

We can save an $\Omega(\log\log n)$ factor in the approximation ratio by adapting the approach of Halldorsson [6] (which he used to save an $\Omega(\log\log n)$ factor in the approximation ratio for the chromatic number) to our context.

Recall the notion of a good subgraph S_{ij} from section 2. It required in particular that $|N(S_{ij})| \geq n''/2k - t$. Modify the definition of good to require that $|N(S_{ij})| > n_{\text{test}} - t$, where n_{test} is the largest value still satisfying

$$n_{\text{test}} \leq \left(\frac{\log n_{\text{test}}}{2\log\log n_{\text{test}}}\right)\cdot\left(\frac{n''}{2k}\right).$$

Include also the following test which is done in the case that $\frac{n''}{2k} - t \leq |N(S_{ij})| \leq n_{\text{test}} - t$. Run the algorithm of [4] on the subgraph induced on $S_{ij} \cup N(S_{ij})$. If it finds a clique of size at least $(\log n_{\text{test}})^3/6\log\log n_{\text{test}}$, join this clique to C and end the algorithm. Otherwise, do not consider S_{ij} to be good (and if no subset of size t is found to be good in the new sense, declare V'' poor).

The analysis of the modified algorithm is similar in many respects to that of the algorithm of section 2. We present here the changes to the proofs of Lemmas 2.3 and 2.4.

LEMMA 3.2. *If a phase of the modified algorithm declares a set* V'' *poor, then indeed the subgraph of* G *induced on* V'' *does not contain a clique of size* $|V''|/2k$.

Proof. Assume that the subgraph induced on V'' contains a clique of size $|V''|/2k$. Then by the pigeon-hole principle, at least one of the parts P_i will contain at least t vertices from this clique. Let S_{ij} be such a subset. Then $|N(S_{ij})| \geq n''/2k - t$. If $|N(S_{ij})| > n_{\text{test}} - t$, then S_{ij} is good, and V'' will not be declared poor. If $|N(S_{ij})| \leq n_{\text{test}} - t$, then the subgraph induced on $S_{ij} \cup N(S_{ij})$ contains n_{test} vertices (if it contains fewer vertices, add to it vertices arbitrarily) and a clique of size $n''/2k = n_{\text{test}}2\log\log n_{\text{test}}/\log n_{\text{test}}$. Then by Proposition 3.1, the algorithm of [4] finds a clique of size $(\log n_{\text{test}})^3/6\log\log n_{\text{test}}$, and the phase ends without declaring V'' poor. ∎

LEMMA 3.3. *Let k and t be such that $\log n/2\log\log n < k < \log n$ and $t = \log n/\log\log n$. If a phase ends by outputting the set C, then this set is a clique on at least $\Omega(t\log_b n')$ vertices, where $b = \Theta(k\log\log n'/\log n')$. In particular, if a graph has a clique of size $\Theta(n\log\log n/\log n)$, the algorithm finds a clique of size $\Omega((\log n)^2/\log\log n)$.*

Proof. The proof is a modification of the proof of Lemma 2.4. We present the differences. The reader is advised to recall the new definition of a good subgraph (that appears prior to Lemma 3.2).

Consider iterations only up to the point where $n'' < \sqrt{n'}$ (ensuring that $\log n'' = \Theta(\log n')$, a fact that simplifies our computations). If before that point the new test finds a clique of size $(\log n_{\text{test}})^3/6\log\log n_{\text{test}}$, then we are done, because n_{test} is large enough to make this clique size $\Omega((\log n')^3/\log\log n')$. If the new test does not find such a clique, then in every iteration the good set S_{ij} that was found had $|N(S_{ij})| > n_{\text{test}}$ (where the value of n_{test} depends on the particular iteration). This means that n'' decreases by a factor of $O(k\log\log n'/\log n')$ between iterations, rather than $O(k)$. The number of iterations becomes at least $\log_b\sqrt{n'}$, where $b = \Theta(k\log\log n'/\log n')$. □

Summing up, for every value of k we can approximate a clique within a ratio of $O(n(\log\log n)^2/(\log n)^3)$, in graphs with cliques of size n/k. For $k \le \log n/2\log\log n$, use the algorithm of [4]; for $\log n/2\log\log n < k < \log n$, use the algorithm of this section; for $\log n \le k \le (\log n)^3$, use either the algorithm of this section or that of section 2; and for $k > (\log n)^3$, just output a single vertex.

4. Discussion. Extending ideas from [4, 2, 6], an $O(n(\log\log n)^2/(\log n)^3)$ approximation ratio is obtained for max clique. This matches the best approximation ratio for the chromatic number. The fact that the two approximation ratios are essentially the same is a consequence of a general framework that we explain below.

Some algorithms for approximating the chromatic number (including [2, 6]) are based on repeatedly finding large independent sets (which serve as color classes). To find a large independent set, they use the fact that every subgraph of a k-colorable graph is itself k-colorable. Hence every subgraph S has an independent set of size at least $|S|/k$.

This principle cannot be applied directly when approximating maximum independent set or max clique. It is not true that in a graph with a clique of size n/k every subgraph S has a clique of size $|S|/k$. The new idea of our paper is to ignore this fact. We run our approximation algorithms for max clique under the assumption that every subgraph does have a clique of size $|S|/k$ or, in fact, slightly smaller. (We chose $|S|/2k$, but the constant 2 is arbitrary and can be replaced by any other constant greater than 1.) For some subgraphs encountered by the algorithm, this assumption is incorrect. However, then one of two things happens: either the algorithm works anyway, or it gets stuck. The point is that, in any case, we make progress. If the algorithm works, we do not care that the assumption was incorrect. If the algorithm gets stuck, then we deduce that the subgraph on which the algorithm got stuck is poor and remove it from the input graph. In the graph that remains the relative size of the maximum clique increases, making the task of finding a large clique easier.

Some other principles that are used in algorithms for approximate coloring also have an analogue in the context of max clique (or max independent set). An instructive example is the algorithm of Alon and Kahale [1] for finding independent sets of size roughly $n^{3/4}$ in graphs that have independent sets of size somewhat larger than

$n/3$. This algorithm is based on the approach of Karger, Motwani, and Sudan [9] for coloring 3-colorable graphs with roughly $n^{1/4}$ colors. The approach of [9] uses semidefinite programming to obtain a so-called vector 3-coloring of the graph and also uses the idea of Wigderson [12] that the neighbors of a vertex in a 3-colorable graph make a bipartite subgraph. Interestingly, both these principles have their analogues in the algorithm of [1]. On the other hand, it is not clear to what extent the principles used in the algorithms of [3, 7] can be used in the context of finding large independent sets in graphs that are not k-colorable but do have an independent set of size roughly n/k.

Let us note that approximate coloring can be performed by repeatedly approximating maximum independent set. This combined with the known hardness of approximation results for maximum independent set implies that the approximation ratio for max clique can be at most a constant factor better than that of min coloring. (See [6] for more details.) This leads to the following interesting question.

- Are the best possible approximation ratios for max clique and the min chromatic number the same (up to multiplicative constant factors)?

Boppana and Halldorsson [4] pointed out connections between approximating max clique and Ramsey theory. In an approach similar to our algorithm (in fact, their algorithm inspired ours), they remove "poor" subgraphs from the input graph. In their case, the poor subgraphs are large enough independent sets, whose existence (if the graph has no large clique) is guaranteed by Ramsey theory. Moreover, their nonexistence suggests an efficient algorithm for finding relatively large cliques (because the relevant arguments in Ramsey theory are constructive). In our case, we define a poor subgraph in such a liberal way that Ramsey theory becomes unnecessary in order to argue about its existence. Specifically, a poor subgraph S is one that does not contain a clique of size $|S|/2k$, whereas our approximation algorithm is satisfied by finding a clique which is very much smaller than $n/2k$. Clearly, either such a clique exists, or the whole graph is poor. Hence unlike Ramsey theory, existence is not an issue here. The only issue is to have an *efficient* algorithm that finds either a clique or a poor subgraph.

Nevertheless, there are connections between our algorithm and Ramsey theory, and we point them out as they may prove fruitful in the future. There is a more general version of the classical Ramsey numbers. Given parameters r and s, let $f(r, s, n)$ denote the minimum over all n vertex graphs that have no s-clique of the maximum cardinality of a subgraph that has no r-clique. In our context of approximating clique, we could use lower bounds on $f(r, s, n)$, provided that certain conditions hold:

1. $f(r, s, n) > kr$.
2. The lower bound is constructive: there is an efficient algorithm for finding either an s-clique or a subgraph on $f(r, s, n)$ vertices without an r-clique.

Using an algorithm similar to that of section 2 we could then find cliques of size roughly s in graphs that have cliques of size n/k. The current bounds known for the function $f(r, s, n)$ [11] are too weak to offer improved approximation ratios for max clique. Let us remark that previously published work on $f(r, s, n)$ dealt only with the case that $r < s$, which is the only case that makes sense in the context of Ramsey theory. However, in our context, where we seek a constructive version, the case $r \geq s$ also makes sense.

Acknowledgments. The author thanks Shimon Kogan for helpful discussions and thanks Magnus Halldorsson, Robert Krauthgamer, and Michael Langberg for useful comments on earlier versions of this manuscript.

REFERENCES

[1] N. Alon and N. Kahale, *Approximating the independence number via the θ-function*, Math. Programming, 80 (1998), pp. 253–264.

[2] B. Berger and J. Rompel, *A better performance guarantee for approximate graph coloring*, Algorithmica, 5 (1990), pp. 459–466.

[3] A. Blum and D. Karger, *An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs*, Inform. Process. Lett., 61 (1997), pp. 49–53.

[4] R. Boppana and M. Halldorsson, *Approximating maximum independent sets by excluding subgraphs*, BIT, 32 (1992), pp. 180–196.

[5] U. Feige and J. Kilian, *Zero knowledge and the chromatic number*, J. Comput. System Sci., 57 (1998), pp. 187–199.

[6] M. Halldorsson, *A still better performance guarantee for approximate graph coloring*, Inform. Process. Lett., 45 (1993), pp. 19–23.

[7] E. Halperin, R. Nathaniel, and U. Zwick, *Coloring k-colorable graphs using relatively small palettes*, J. Algorithms, 45 (2002), pp. 72–90.

[8] J. Håstad, *Clique is hard to approximate within $n^{1-\epsilon}$*, Acta Math., 182 (1999), pp. 105–142.

[9] D. Karger, R. Motwani, and M. Sudan, *Approximate graph coloring by semidefinite programming*, J. ACM, 45 (1998), pp. 246–265.

[10] S. Khot, *Improved inapproximability results for maxclique, chromatic number and approximate graph coloring*, in Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, Las Vegas, NV, 2001, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp. 600–609.

[11] B. Sudakov, *A new lower bound for a Ramsey-type problem*, Combinatorica, to appear.

[12] A. Wigderson, *Improving the performance guarantee of approximate graph coloring*, J. ACM, 30 (1983), pp. 729–735.