On Bounded Queries and Approximation

Richard Chang[*]
Department of Computer Science
University of Maryland Baltimore County
Baltimore, MD 21228
chang@cs.umbc.edu

William I. Gasarch[†]
Department of Computer Science
University of Maryland College Park
College Park, MD 20742
gasarch@cs.umd.edu

Carsten Lund
AT&T Bell Laboratories
Murray Hill, NJ 07974
lund@research.att.com

May 29, 2007

Abstract

This paper investigates the computational complexity of approximating several NP-optimization problems using the number of queries to an NP oracle as a complexity measure. The results show a trade-off between the closeness of the approximation and the number of queries required. For an approximation factor $k(n)$, $\log \log_{k(n)} n$ queries to an NP oracle can be used to approximate the maximum clique size of a graph within a factor of $k(n)$. However, this approximation cannot be achieved using fewer than $\log \log_{k(n)} n - c$ queries to *any oracle* unless P = NP, where c is a constant that does not depend on k. These results hold for approximation factors $k(n) \geq 2$ that belong to a class of functions which includes any integer constant function, $\log n$, $\log^a n$ and $n^{1/a}$. Similar results are obtained for graph coloring, set cover and other NP-optimization problems.

[*]Supported in part by NSF Research Grant CCR-9309137.
[†]Supported in part by NSF Research Grant CCR-9020079.

1 Introduction

The approximability of NP-optimization problems is a central theme both in the study of algorithms and in computational complexity theory. Most NP-optimization problems have decision versions that are NP-complete and are hence equivalent to each other as decision problems. However, the approximability of the optimization problems may vary greatly. For some NP-optimization problems, there are efficient algorithms that find good approximate solutions. For others, no such algorithm can exist unless some standard intractability assumption is violated (e.g., P = NP or the Polynomial Hierarchy collapses). Recently, Arora, Lund, Motwani, Sudan and Szegedy [ALM+92] showed that the problem of finding the largest clique in a graph is in the latter category. Following a series of breakthrough results [AS92, BFL91, FGL+91, LFKN90, Sha90], they showed that there exists a constant ϵ such that no deterministic polynomial time algorithm can approximate the maximum clique size $\omega(G)$ of a graph G with n vertices within a factor of n^ϵ, unless P = NP. While this result strongly suggests that no efficient algorithm can find good approximations to the maximum clique problem, it does not resolve all of the questions about the computational complexity of approximating the maximum clique size of a graph. In particular, it is not clear what computational resources are sufficient and/or necessary to compute an approximation of the maximum clique size using any of the traditional resource bounded measures (e.g., time, space, random bits and alternation).

In this paper we use the number of queries to an NP-complete oracle as a complexity measure. Krentel [Kre88] used this measure to show that the maximum clique size is complete for polynomial time functions which use only $O(\log n)$ queries, denoted $\mathrm{PF}^{\mathrm{NP}[O(\log n)]}$. Since Krentel's original work, many connections between bounded query classes and standard complexity classes have been discovered [ABG90, AG88, Bei87, Bei91, CK90, CKR95, GKR, Kad88, Wag86, WW85]. In many circumstances, these results show that one cannot decrease the number of queries needed to solve a problem by even a single query, unless P = NP or PH collapses. For example, Hoene and Nickelsen [HN93] showed that to determine how many of the formulas in $F_1, \ldots, F_r$ are satisfiable, $\lceil \log(r+1) \rceil$ queries are both sufficient and necessary (unless P = NP). A naive binary search can determine the number of satisfiable formulas using $\lceil \log(r+1) \rceil$ queries to the NP oracle. The number of queries needed is $\lceil \log(r+1) \rceil$ rather than $\lceil \log r \rceil$ because there are $r+1$ possible answers ranging from 0 to r. A tree pruning technique shows that no polynomial time machine can determine the number of satisfiable formulas using one fewer query to any oracle unless P = NP. Thus, the algorithm which uses the fewest queries is simply the naive binary search algorithm. In this paper, we show that approximating several NP-optimization problems also have this property. In the first parts of the paper, we will focus on the complexity of approximating the size of the maximum clique in a graph.

In order to state the results in this paper correctly we need to be more precise with the term "approximation". Let $\omega(G)$ denote the size of the maximum clique in the graph G. We say that a number x is an approximation of $\omega(G)$ within a factor of $k(n)$ if $\omega(G)/k(n) \leq x \leq \omega(G)$. Our results show a tradeoff between the closeness of the approximation and the number of queries needed to solve the approximation problem — finding closer approximations require more queries.

For example we can approximate $\omega(G)$ within a factor of 2 using only $\log \log n$ queries to

Factor	Upper bound	Lower bound (unless P = NP)
2	$\log\log n$	$\log\log n - \log 1/\epsilon$
k	$\log\log n - \log\log k$	$\log\log n - \log\log k - \log 1/\epsilon$
$\log^a n$	$\log\log n - \log\log\log^a n$	$\log\log n - \log\log\log^a n - \log 1/\epsilon$
$n^{1/a}$	$\log a$	$\log a - \log 1/\epsilon$

Table 1: Upper and lower bounds for approximating the maximum clique size.

NP, where n is the number of vertices in the graph G. In contrast, computing $\omega(G)$ exactly can be done with $\log n$ queries and requires $\Omega(\log n)$ queries (unless P = NP) [Kre88]. Moreover, we show that no function using fewer than $(\log\log n) - \log 1/\epsilon$ queries to *any oracle* can approximate $\omega(G)$ within a factor 2 unless P = NP. (Here ϵ is the constant given in Corollary 3 of [ALM+92].) In general our results show that for any "nice" approximation factor $k(n) \geq 2$, $\omega(G)$ can be approximated within a factor of $k(n)$ using $\log\log_{k(n)} n$ queries but not with fewer than $\log\log_{k(n)} n - \log 1/\epsilon$ queries to any oracle unless P = NP. In Corollary 6, we show that the difference, $\log 1/\epsilon$, between the upper and lower bounds has a natural interpretation. Table 1 summarizes our results for some common approximation factors.

We make a few observations about these results. First, since ϵ is a constant, for a large enough constant k, $\log\log k$ would exceed $\log 1/\epsilon$. Hence, for this k, the upper bound on approximating $\omega(G)$ within a factor of k will be strictly less than the lower bound for approximating within a factor of 2. Hence, for this large k the problem of approximating $\omega(G)$ within a factor k has strictly lower complexity in terms of the number of queries than approximating within a factor of 2 unless P = NP. Similarly, approximating within a factor of $\log n$ has a lower complexity than approximating within any constant; and approximating within a factor of $n^{1/k}$ has an even lower complexity. We believe that these are the first results which show a tradeoff between complexity and closeness of approximation. In contrast, Garey and Johnson [GJ79] showed that if $\omega(G)$ can be approximated within a constant factor in P then it can be approximated within *any* constant factor in P. While these are not contradictory theorems, they certainly have very different flavors.

In the next section, we state the lemmas, definitions and notations needed to prove the results. In Section 3, we show that a uniform binary search routine provides the upper bounds we have mentioned. To prove the lower bound results, we start in Section 4.2 with a simple proof which shows that no function in $\mathrm{PF}^{X[1]}$, for any oracle X, can approximate $\omega(G)$ within a factor of 2, unless P = NP. We go on to the proof of the general lower bound results. In Section 5 we discuss how these results extend to other NP optimization problems (e.g., Chromatic Number). In Section 6, we ponder upon the value of the constant ϵ. Finally, in Section 7 we combine our techniques with the results of Lund and Yannakakis [LY94] on the hardness of approximating the Set Cover problem to derive upper and lower bounds on the query complexity of approximating the minimum set cover size.

2 Preliminaries

Definition 1 For every graph G, let $|G|$ denote the number of vertices in the graph and let $\omega(G)$ denote the size of the largest clique in G. We say that a function $A(G)$ approximates the maximum clique size within a factor of $k(n)$ if for all graphs G with n vertices

$$\omega(G)/k(n) \leq A(G) \leq \omega(G).$$

Some papers use the alternative condition that $\omega(G)/k(n) \leq A(G) \leq k(n) \cdot \omega(G)$, but we find it unintuitive to consider $A(G)$ an approximation of $\omega(G)$ if there does not exist a clique of size $A(G)$ in G. However, the results in this paper still hold under the alternative definition.

To prove the lower bounds, we need the result of Arora *et al* [ALM+92] which showed that there exists an $\epsilon > 0$ such that the maximum clique size cannot be approximated within a factor of n^ϵ in deterministic polynomial time unless P = NP, where n is the number of vertices in the graph. Their construction yields the next lemma.

Lemma 2 [ALM+92] There exist constants s, b and d, $0 < s < b < d$, such that given a Boolean formula F with t variables, we can construct in polynomial time a graph G with $m = t^d$ vertices, where

$$F \in \text{SAT} \Longrightarrow \omega(G) = t^b \quad \text{and}$$
$$F \notin \text{SAT} \Longrightarrow \omega(G) < t^s.$$

The constants b, s and d will be fixed for the remainder of the paper. Of particular interest is the ratio $(b-s)/d$, because it is equal to the ϵ mentioned before Lemma 2. To prove the intractability of n^ϵ approximation, first assume that we have a polynomial time algorithm to approximate $\omega(G)$ within n^ϵ. Take any Boolean formula F, and let t be the number of variables in F. Construct G as described in Lemma 2. Use the polynomial time algorithm to obtain an approximation x of $\omega(G)$. Since $|G| = t^d$, an n^ϵ approximation is within a factor of $(t^d)^\epsilon = t^{b-s}$. So, if $F \in \text{SAT}$, the algorithm must guarantee that $x \geq t^b/t^{b-s} = t^s$. On the other hand, if $F \notin \text{SAT}$ then $x < t^s$. Thus, $F \in \text{SAT} \iff x \geq t^s$.

Definition 3 Let $\text{PF}^{X[q(n)]}$ be the class of functions computed by polynomial time oracle Turing machines which ask at most $q(n)$ queries to the oracle X. Since the queries are adaptive, the query strings may depend on answers to previous queries.

3 Upper bounds

We first examine the upper bounds on the complexity of approximating NP-optimization problems in terms of bounded query classes. The general idea is to use each query to SAT to narrow down the range where the optimal solution exists. For example, for a graph G with n nodes, $1 \leq \omega(G) \leq n$. We can compute $\omega(G)$ exactly using binary search and $\log n$ queries of the form: "Is $\omega(G)$ greater than x?"

On the other hand, to approximate $\omega(G)$ within a factor of 2, we only need to find a number x such that $x \leq \omega(G) \leq 2x$. Thus, we first partition the numbers from 1 to n into the intervals

$$[1,2],\ (2,4],\ (4,8],\ \ldots,\ \left(2^{\lceil \log n \rceil - 1}, 2^{\lceil \log n \rceil}\right].$$

If $\omega(G)$ is in the interval $(2^i, 2^{i+1}]$, then we can simply use 2^i as a factor 2 approximation of $\omega(G)$. Thus, our strategy is to use binary search on the left endpoints of the intervals to determine which interval contains $\omega(G)$. Since there are only $\lceil \log n \rceil$ intervals, we only need $\lceil \log \lceil \log n \rceil \rceil$ queries to SAT to perform this binary search. In the general case, if we want to find an approximation of $\omega(G)$ that is within a factor of $k(n)$, there will be $\left\lceil \log_{k(n)} n \right\rceil$ intervals of the form $(k(n)^i, k(n)^{i+1}]$ and binary search would use $\left\lceil \log \left\lceil \log_{k(n)} n \right\rceil \right\rceil$ queries to SAT. Thus, we have the following lemma.

Lemma 4 Let $k(n)$ be a polynomial time computable function such that $1 < k(n) < n$. Then there exists a function in $\mathrm{PF}^{\mathrm{SAT}\left[\left\lceil \log \left\lceil \log_{k(n)} n \right\rceil \right\rceil\right]}$ which approximates $\omega(G)$ within a factor of $k(n)$ for all graphs G with n vertices.

The lemma is stated for the maximum clique size problem, but it obviously holds for any NP-optimization problem where the solution is an integer ranging from 1 to n. Note that it does not matter if $k(n)$ is not an integer, because checking whether $\omega(G)$ is greater than x is still an NP question when x is a fractional number. Once we have determined that $\omega(G)$ is contained in the interval $(k(n)^i, k(n)^{i+1}]$, we can output $\lceil k(n)^i \rceil$ as an approximation to $\omega(G)$. Also, if we drop the ceilings from our notation, then we can derive more readable upper bounds on the complexity of approximating $\omega(G)$ for some common approximation factors (see Table 1).

The binary search strategy used to find the approximation to $\omega(G)$ may seem naive. However, we shall see later that the upper bounds differ from the lower bounds by at most an additive constant. In any case, we can improve the upper bounds if there exists a polynomial time algorithm which gives an approximate solution within a factor of $f(n)$. Then, our strategy is to first use the polynomial time algorithm to obtain an approximation x. We know that the solution is between x and $x \cdot f(n)$. Now, to find an approximation that is within a factor of $k(n)$, we divide the numbers from x to $x \cdot f(n)$ into intervals:

$$[x, xk(n)],\ \left(xk(n), xk(n)^2\right], \ldots, \left(x\frac{f(n)}{k(n)}, xf(n)\right].$$

In this case, the number of intervals is $\left\lceil \log_{k(n)} f(n) \right\rceil$, and we have the lemma:

Lemma 5 Let $k(n)$ be a polynomial time computable function such that $1 < k(n) < n$. Suppose that there exists a polynomial time algorithm which approximates $\omega(G)$ within a factor of $f(n)$. Then there exists a function in $\mathrm{PF}^{\mathrm{SAT}\left[\left\lceil \log \left\lceil \log_{k(n)} f(n) \right\rceil \right\rceil\right]}$ which approximates $\omega(G)$ within a factor of $k(n)$ for all graphs G with n vertices.

Again, we note that this lemma applies to any NP-optimization problem whose solutions range from 1 to n. This lemma may seem somewhat useless since the best known polynomial time algorithm can only approximate the size of the maximum clique within a factor of

$O(n/(\log n)^2)$ [BH92]. If we were to use the new strategy outlined above, we would reduce the number of queries needed to find a factor 2 approximation of $\omega(G)$ to $\log(\log n - 2\log\log n)$, which would save us at most one query for various values of n. However, the following corollary of the lemma does allow us to gauge the quality of our lower bound results.

Corollary 6 If no function in $\mathrm{PF}^{\mathrm{SAT}[\log\log_{k(n)} n - \log 1/\delta]}$ approximates $\omega(G)$ within a factor of $k(n)$, then no polynomial time algorithm can approximate $\omega(G)$ within a factor of n^δ.

Corollary 6 gives us a natural interpretation of the difference between the upper bound of $\log\log_{k(n)} n$ in Lemma 4 and the lower bound of $\log\log_{k(n)} - \log 1/\epsilon$ (Theorem 15). This difference of $\log 1/\epsilon$ reflects the fact that we do not know if there exists a polynomial time algorithm which approximates $\omega(G)$ within a factor of n^δ for $\epsilon < \delta < 1$. Thus, an improvement of either the upper or the lower bound is possible.

Moreover, the observations about Lemma 5 are most useful when they are applied to NP-optimization problems such as Set Cover. In the Set Cover problem we are given a finite collection C of subsets of $\{1, \ldots, n\}$ and asked to find the size of the smallest subcollection $C' \subseteq C$ that covers $\{1, \ldots, n\}$. Since the size of the smallest set cover *can* be approximated within a factor $\ln(n) + 1$ [Joh74, Lov75], we have the following lemma:

Lemma 7 Let $k(n)$ be a polynomial time computable function such that $1 < k(n) < n$. Then there exists a function in $\mathrm{PF}^{\mathrm{SAT}[\lceil\log\lceil\log_{k(n)}(\ln n+1)\rceil\rceil]}$ which approximates the size of the minimum set cover within a factor of $k(n)$.

Now, by comparing Lemma 4 against Lemma 7, we can obtain a quantitative difference between the complexity of approximating $\omega(G)$ and the complexity of approximating the size of the minimum set cover — not just a qualitative difference. For example, if we are allowed to make $\log\log(\ln n + 1)$ queries to SAT, then we can approximate the minimum set cover within a factor of 2. However, using the same number of queries, we can only approximate $\omega(G)$ within a factor of $n^{1/\log(\ln n+1)}$. Thus, we can conclude that approximating set cover within a factor of 2 has about the same complexity as approximating $\omega(G)$ within a factor of $n^{1/\log(\ln n+1)}$. Such a comparison is only possible by taking a quantitative view of the complexity of approximations in terms of the number of queries.

Note that the existence of a "good" polynomial time approximation algorithm for a problem has a greater effect on the complexity of approximating the problem than on the complexity of finding the exact solution. For example, suppose that we have some NP-optimization problem where the solution ranges from 1 to n. Without the help of an approximation algorithm, we would need $\log n$ queries to find the exact solution and $\log\log n$ queries to find a factor 2 approximation. Now, suppose that we are given a polynomial time algorithm that guarantees a factor 4 approximation. To find the exact solution we would still need $\log(n - n/4) = O(\log n)$ queries. However, we only need one query ($\log\log 4 = 1$) to approximate within a factor of 2.

The upper bounds proven in this section uses a naive binary search strategy to determine an interval that contains the optimal solution. One might suspect that a more clever algorithm could use substantially fewer queries to the oracle. In the rest of the paper, we show that unless some intractability assumption is violated (e.g., P = NP and RP = NP),

no polynomial time algorithm can reduce the number of queries by more than an additive constant. These results give us relative lower bounds on the number of queries needed to approximate the maximum clique size of a graph. We are also able to extend these techniques to determine the query complexity of approximating the chromatic number of a graph and the minimum set cover.

4 Lower bounds

4.1 Promise Problems and clique arithmetic

To prove our lower bound results, we need to introduce some more definitions and notations.

Definition 8 Let $\#^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$ be the number of formulas in $\{F_1, \ldots, F_{r(n)}\}$ which are satisfiable. I.e., $\#^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)}) = |\{F_1, \ldots, F_{r(n)}\} \cap \mathrm{SAT}|$.

Definition 9 Let $r(t)$ be a polynomially bounded polynomial time function. We define $\mathcal{P}_{r(t)}$ to be the following promise problem. Given a sequence of Boolean formulas $F_1, \ldots, F_{r(t)}$ where each F_i has t variables and the promise that for all i, $2 \leq i \leq r(t)$, $F_i \in \mathrm{SAT}$ implies that $F_{i-1} \in \mathrm{SAT}$, output $\#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$.

Technically the size of the input to the promise problem is $|F_1, \ldots, F_{r(t)}|$. This size is $O(tr(t))$ when the Boolean formulas are restricted to ones where each variable occurs only a constant number of times. To simplify our notation, in the following lemma we count the queries as a function of t rather than $|F_1, \ldots, F_{r(t)}|$. The following lemma provides a tight lower bound on the complexity of $\mathcal{P}_r$ and can be proven using the self-reducibility of SAT and a tree pruning technique [HN93]. We include the proof for the sake of completeness.

Lemma 10 Let $r(t)$ be a logarithmically bounded polynomial time computable function. If there exists an oracle X such that some polynomial time function solves $\mathcal{P}_{r(t)}$ using fewer than $\lceil \log(r(t)+1) \rceil$ queries to X, then P = NP.

Proof: Let $q(t) = \lceil \log(r(t)+1) \rceil - 1$ and let M be a polynomial time oracle Turing machine which solves $\mathcal{P}_r$ using $q(t)$ queries to X. We know that $q(t) = O(\log \log n)$, so the entire oracle computation tree of M on input $F_1, \ldots, F_{r(t)}$ is polynomially bounded and can be searched deterministically. In fact, the oracle computation tree has at most $r(t)$ leaves, since for all x, $2^{\lceil \log x \rceil} < 2x$. One of these leaves represents the correct computation of $M^X(F_1, \ldots, F_{r(t)})$ and contains the value of $\#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$. However, there are $r(t)+1$ possible answers for $\#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$ ranging from 0 to $r(t)$. So, one possible answer, call it z, does not appear in any leaf. Moreover, one of the leaves contains the correct answer, so $z \neq \#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$. Thus, we can construct a polynomial time Turing machine M' which on input $F_1, \ldots, F_{r(t)}$ prints out a number $z \leq r(t)$ such that $z \neq \#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$.

Now we show that P = NP under this condition. Given a Boolean formula F, consider its disjunctive self-reduction tree. Each node in the tree is a formula; the children of a node are the two formulas obtained by instantiating one of the variables in the formula by 0 and by 1. With F at the root of the tree, the tree has height t and exponential size. However,

we will only examine $r(t)$ levels of the tree at a time. Now, let $F = F_1, \ldots, F_{r(t)}$ be a path in this tree. Suppose that $F_{i+1} \in \text{SAT}$. Since F_{i+1} is a child of F_i, we can assume that $F_i \in \text{SAT}$. Thus, the promise condition of $\mathcal{P}_{r(t)}$ holds and we can use M' to find a number z, such that $z \neq \#^{\text{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$. Now, replace the subtree rooted at F_z with the subtree rooted at F_{z+1}. If $F_z \notin \text{SAT}$, then $F_{z+1} \notin \text{SAT}$ by the promise condition. If $F_z \in \text{SAT}$ and $F_{z+1} \notin \text{SAT}$, then z would equal $\#^{\text{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$ which contradicts our construction of M'. Thus, $F_{z+1} \in \text{SAT}$ iff $F_z \in \text{SAT}$ and we have shortened the path by 1. Repeat this process for all paths in any order until all the paths in the tree have length less than $r(t)$. Now, the original formula F is satisfiable iff one of the leaves (in which all the variables have been instantiated) evaluates to true. Since the final tree is polynomially bounded, we can check every leaf exhaustively. □

In the proofs that follow we need to construct graphs in which the maximum clique size can occur only at restricted intervals. To assist in this construction, we define two operators on graphs: $\oplus$ and $\otimes$. We also use K_i to denote the complete graph with i vertices.

Definition 11 Given two graphs G_1 and G_2, the graph $H = G_1 \oplus G_2$ is constructed by taking the disjoint union of the vertices of G_1 and G_2. The edges of H are all the edges of G_1 and G_2, plus the edges (u, v) for each vertex u in G_1 and v in G_2. (So, every vertex in G_1 is connected to every vertex in G_2).

Definition 12 [1] Given two graphs G_1 and G_2, the graph $H = G_1 \otimes G_2$ is constructed by replacing each vertex of G_1 with a copy of G_2. Furthermore, for each edge (u, v) in G_1, each vertex in the copy of G_2 replacing u is connected to every vertex in the copy of G_2 replacing v. Note that $\otimes$ is not commutative and that we give $\otimes$ higher precedence than $\oplus$.

Lemma 13 Let G_1 and G_2 be any two graphs, then

$$\omega(G_1 \oplus G_2) = \omega(G_1) + \omega(G_2) \quad \text{and}$$
$$\omega(G_1 \otimes G_2) = \omega(G_1) \cdot \omega(G_2).$$

4.2 A simple lower bound

As a first lower bound result, we show that for all oracles X, no function in $\text{PF}^{X[1]}$ can approximate $\omega(G)$ within a factor of 2 unless P = NP. To do this, we start with the assumption that some function $f \in \text{PF}^{X[1]}$ does approximate $\omega(G)$ within a factor of 2 and show that using this function we can solve the promise problem $\mathcal{P}_2$ using only one query to X. Then, P = NP by Lemma 10.

In our construction, we start with the input to the promise problem $\mathcal{P}_2$, which is a pair of Boolean formulas F_1 and F_2 each with t variables. Now using the reduction from Lemma 2, we construct two graphs G_1 and G_2 with t^d vertices such that

$$F_i \in \text{SAT} \Longrightarrow \omega(G_i) = t^b \quad \text{and}$$
$$F_i \notin \text{SAT} \Longrightarrow \omega(G_i) < t^s.$$

[1] This is identical to graph composition in Garey and Johnson [GJ79]. The cover of Garey and Johnson is the picture of a 3-clique composed with a graph that has three points connected by 2 edges.

Then, let $H = G_1 \oplus (K_2 \otimes G_2)$. By Lemma 13, $\omega(H) = \omega(G_1) + 2 \cdot \omega(G_2)$. Also, assume that t is sufficiently large so that $t^b > 6 \cdot t^s$.

Consider the effect of the satisfiability of F_1 and F_2 on $\omega(H)$. If $F_1 \notin \text{SAT}$ and $F_2 \notin \text{SAT}$, then

$$\omega(H) = \omega(G_1) + 2 \cdot \omega(G_2) < 3 \cdot t^s.$$

On the other hand, if $F_1 \in \text{SAT}$ and $F_2 \notin \text{SAT}$ then $t^b \leq \omega(H) < t^b + 2 \cdot t^s$. Finally, if $F_1 \in \text{SAT}$ and $F_2 \in \text{SAT}$ then $\omega(H) = 3 \cdot t^b$.

Because of the promise condition of the promise problem, we do not have the case where $F_1 \notin \text{SAT}$ and $F_2 \in \text{SAT}$. This restricts the value of $\omega(H)$ to 3 non-overlapping intervals. In fact, these intervals are separated by a factor of 2. Thus, a factor of 2 approximation of $\omega(H)$ will tell us which formulas are satisfiable. For example, if we are given an approximation x guaranteed to be within a factor of 2, and $x \geq 1.5 \cdot t^b$, then we know that both F_1 and F_2 are satisfiable. If this approximation can be achieved using only one query to X, then we can solve the promise problem $\mathcal{P}_2$ in $\text{PF}^{X[1]}$ and P = NP.

4.3 The general construction

In this section we prove the general theorem for the lower bound on approximating the size of the maximum clique in a graph. First, we define a class of approximation factors.

Definition 14 We call a function $k : \mathbb{N} \to \mathbb{R}$ a *nice approximation factor* if it is computable in polynomial time and all of the following hold:

1. $\exists n_0 \ \forall n > n_0, \ 2 \leq k(n) < n.$
2. $\exists n_0 \ \forall m > n > n_0, \ k(m) \geq k(n).$
3. $\forall \delta > 0, \ \exists n_0, \ \forall m > n > n_0,$

$$(1 + \delta)\frac{\log m}{\log k(m)} > \frac{\log n}{\log k(n)}.$$

Please note that $k(n)$ is nice if $k(n)$ equals $n^{1/a}$, $\log n$, $(\log n)^a$, or a constant ≥ 2. The natural interpretation of the third condition is the following. Consider the function $f(n) = (\log n)/(\log k(n))$. The function $f(n)$ is also related to $k(n)$ by: $k(n) = n^{1/f(n)}$. The third condition is satisfied if $f(n)$ is increasing, if $f(n)$ is constant or if $f(n)$ is decreasing but converges to a limit (e.g., when $k(n) = \sqrt{n}/2$). Since $k(n) < n$, $f(n)$ is bounded below by 1. Thus, if $f(n)$ is decreasing almost everywhere, it must converge to a limit. Hence, the third condition is not satisfied only when $f(n)$ alternates between increasing and decreasing infinitely often. We rule out these functions as approximation factors.

Theorem 15 Let $k(n)$ be a nice approximation factor which is (1) unbounded, (2) converges to an integral constant or (3) converges to a sufficiently large constant. Then, for all oracles X, no polynomial time function can approximate $\omega(G)$ within a factor of $k(n)$ using $\lfloor \log \log_{k(n)} n - c \rfloor$ or fewer queries to X unless P = NP, where $c = 1 + \log(1/\epsilon)$.

Proof: The general strategy of this proof is to reduce the promise problem $\mathcal{P}_r$ to the problem of approximating the maximum clique size of a graph H within a factor of $k(n)$. Since Lemma 10 provides us with a lower bound on the complexity of $\mathcal{P}_r$ (assuming that P $\neq$ NP), we obtain a lower bound on the complexity of approximating $\omega(H)$.

So, we begin with the input to the promise problem $\mathcal{P}_r$, the Boolean formulas, $F_1, \ldots, F_r$, each with t variables. (The actual value of r will be chosen later.) We convert each formula F_i into a graph G_i with $m = t^d$ vertices according to the construction described in Lemma 2. The values of $\omega(G_i)$ are restricted by:

$$\begin{aligned} F_i \in \text{SAT} &\Longrightarrow \omega(G_i) = t^b \quad \text{and} \\ F_i \notin \text{SAT} &\Longrightarrow \omega(G_i) < t^s. \end{aligned}$$

Then we choose a gap size g. In the simple example above, g is 2. In this proof, the value of g and r will depend on t. However, for notational convenience, we do not make this dependency explicit. Moreover, we can only choose g to be a whole number. Now, given the choices of r and g, we construct the graph H as follows:

$$\begin{aligned} H &= G_1 \oplus (K_g \otimes G_2) \oplus (K_{g^2} \otimes G_3) \oplus \cdots \\ &\quad \oplus (K_{g^{i-1}} \otimes G_i) \oplus \cdots \oplus (K_{g^{r-1}} \otimes G_r). \end{aligned}$$

At first glance, this does not appear to be a polynomial time construction because we could double the size of each succeeding graph. However, r will turn out to be logarithmically bounded, so $|H|$ will be polynomially bounded. Finally, let

$$\nu_j = \sum_{i=0}^{j-1} g^i = \frac{g^j - 1}{g - 1},$$

then $n = |H| = \nu_r \cdot m = \nu_r \cdot t^d$.

Now suppose there exists a polynomial time function which approximates $\omega(H)$ within a factor of $k(n)$ using $\log \log_{k(n)} n - c$ queries to X. We want to show that the factor $k(n)$ approximation of $\omega(H)$ also tells us the value of $\#_r^{\text{SAT}}(F_1, \ldots, F_r)$. Also, we want to constrain the choice of g and r so that $\log \log_{k(n)} n - c < \lceil \log(r+1) \rceil$. Then, by Lemma 10, being able to approximate $\omega(H)$ within a $k(n)$ factor using only $\log \log_{k(n)} n - c$ queries to X would imply that P = NP. To make this claim our choices of g and r are critical. We have already encountered one constraint on g and r. The other constraints arise when we analyze the possible values of $\omega(H)$.

For the moment, assume that there are exactly z satisfiable formulas in $F_1, \ldots, F_r$. Then, $F_1, \ldots, F_z$ are in SAT and $F_{z+1}, \ldots, F_r$ are in $\overline{\text{SAT}}$ by the promise condition of $\mathcal{P}_r$. We can calculate $\omega(H)$ as follows:

$$\begin{aligned} \omega(H) &= \sum_{i=1}^{r} g^{i-1} \omega(G_i) \\ &= \left(\sum_{i=1}^{z} g^{i-1} \omega(G_i) \right) + \left(\sum_{i=z+1}^{r} g^{i-1} \omega(G_i) \right) \\ &= \nu_z t^b + \left(\sum_{i=z+1}^{r} g^{i-1} \omega(G_i) \right). \end{aligned}$$

Since F_i is unsatisfiable, for $z+1 \leq i \leq r$, we can estimate the size of the second term by $\nu_r \cdot t^s$. Then, we can bound the size of the maximum clique of H:

$$\nu_z t^b \leq \omega(H) \leq \nu_z t^b + \nu_r t^s.$$

We want to show that an approximation of $\omega(H)$ within a factor of g will also allow us to calculate $\#_r^{\mathrm{SAT}}(F_1, \ldots, F_r)$. So, assume that we are provided with a function which approximates $\omega(H)$ within a factor g. To distinguish the case where z formulas are satisfiable from the case where $z+1$ formulas are satisfiable, we must have: $g \cdot (\nu_z t^b + \nu_r t^s) < \nu_{z+1} t^b$. That is, the upper bound on $\omega(H)$ when z formulas are satisfiable must be smaller, by a factor of g, than the lower bound on $\omega(H)$ when $z+1$ formulas are satisfiable. Since $\nu_{z+1} = g\nu_z + 1$, this condition is satisfied if we have the constraint that $g\nu_r t^s < t^b$. Similarly, under this constraint we would also have: $g(\nu_{z-1} t^b + \nu_r t^s) < \nu_z t^b$. Hence, we have restricted the possible values of $\omega(H)$ to $r+1$ disjoint intervals which are separated by a factor of g. Thus, given a number x which is guaranteed to be an approximation of $\omega(H)$ within a factor of $k(n)$, $k(n) \leq g$, we can find the largest z such that $gx \geq \nu_z t^b$. This largest z is equal to $\#_r^{\mathrm{SAT}}(F_1, \ldots, F_r)$. It is important to note here that the approximation factor $k(n)$ depends on n which is the size of H and not the size of the original input. Furthermore, the value of n depends on the value of g. Thus, it is not a simple task to choose g and have $k(n) \leq g$. (Recall that $n = \nu_r t^d$.)

In summary, we must choose g and r such that the following constraints hold. (Recall that we use g and $g(t)$ interchangeably.)

Constraint 1: $g(t)\nu_r t^s \stackrel{\text{def}}{=} g \cdot \dfrac{g^r - 1}{g - 1} \cdot t^s < t^b$.

Constraint 2: $k(n) \leq g(t)$.

Constraint 3: $\left\lfloor (\log \log_{k(n)} n) - c \right\rfloor < \lceil \log(r(t) + 1) \rceil$.

The main difficulty of this proof is to choose the correct values for the parameters. For example, we can satisfy Constraint 3 by picking a large value for r. However, if r is large, then Constraint 1 is harder to satisfy. We also satisfy Constraint 2 easily by choosing a large g, but a large g would force us to pick a small r which then violates Constraint 3. Let $n' = t^{b-s+d}$. We will show that the following choices satisfy the constraints.

- $g = \left\lceil k(t^{b-s+d}) \right\rceil \stackrel{\text{def}}{=} \lceil k(n') \rceil$.
- $r = \left\lfloor \log_g \dfrac{t^{b-s}(g-1)}{g} \right\rfloor$.

Note that g and r are chosen to be integers. This is important for our construction, but it is also the source of some difficulties in our calculations. The reader may find the proof easier to follow by substituting 2 or $\sqrt{n}$ for g. These are the two extreme possibilities.

First we show that Constraint 1 holds. By our choice of r, $r \leq \log_g(t^{b-s}(g-1)/g)$. By removing the $\log_g$, isolating t^b and using the fact that $g^r - 1 < g^r$, we obtain the following and hence Constraint 1 holds.

$$g\nu_r t^s = g \cdot \frac{g^r - 1}{g - 1} \cdot t^s < g \cdot \frac{g^r}{g - 1} \cdot t^s \leq t^b. \tag{1}$$

We can also use Equation 1 to estimate n in terms of t. From our construction, we know that $n = \nu_r t^d$. From Equation 1, we know that $\nu_r < t^{b-s}$. Hence, $n < t^{b-s+d} = n'$. Since $k(n)$ is a nice approximation factor, it is monotonic after some point. Hence, for large enough t, $k(n) \leq k(n') \leq \lceil k(n') \rceil = g(t)$. Thus, Constraint 2 holds.

Finally, we have to show that Constraint 3 holds. Since $c = 1 + \log(1/\epsilon)$, $2^{-c} = \epsilon/2$. It suffices to show that $(\log \log_{k(n)} n) - c < \log(r+1)$, or that $(\epsilon/2) \log_{k(n)} n < r + 1$. By substituting the definition of r, and being careful with the floor notation, we can satisfy Constraint 3 by showing

$$\frac{2}{\epsilon} \log_g \frac{g}{g-1} + \log_{k(n)} n < \frac{2}{\epsilon} \log_g t^{b-s}. \tag{2}$$

In the next step, we rewrite $(2/\epsilon)$ as follows. Recall that $\epsilon = (b-s)/d$.

$$\frac{2}{\epsilon} = \frac{2}{(1+\epsilon)} \frac{(1+\epsilon)}{\epsilon} = \left(\frac{1-\epsilon}{2+2\epsilon} + \frac{3+\epsilon}{2+2\epsilon} \right) \frac{b-s+d}{b-s}.$$

Thus,

$$\frac{2}{\epsilon} \log_g t^{b-s} = \frac{1-\epsilon}{2+2\epsilon} \log_g n' + \frac{3+\epsilon}{2+2\epsilon} \log_g n'.$$

Since ϵ must be less than 1, $(3+\epsilon)/(2+2\epsilon) > 1$ and $(1-\epsilon)/(2+2\epsilon) > 0$. We can satisfy Constraint 3 by showing the following two inequalities:

$$\frac{2}{\epsilon} \log_g \frac{g}{g-1} < \frac{1-\epsilon}{2+2\epsilon} \log_g n', \tag{3}$$

$$\log_{k(n)} n < \frac{3+\epsilon}{2+2\epsilon} \log_g n'. \tag{4}$$

Equation 3 holds since $g/(g-1)$ is bounded by 2 and $n' = t^{b-s+d}$ is unbounded.

Now, we have to show that Equation 4 holds. First, pick $\delta > 0$ small enough so that

$$(1+\delta)^2 < \frac{3+\epsilon}{2+2\epsilon}.$$

Recall that $g = \lceil k(n') \rceil$. Thus,

$$\frac{3+\epsilon}{2+2\epsilon} \log_g n' > (1+\delta)^2 \cdot \frac{\log n'}{\log k(n')} \cdot \frac{\log k(n')}{\log \lceil k(n') \rceil}.$$

Consider the ratio: $\log k(n') / \log \lceil k(n') \rceil$. If $k(x)$ is always an integer, then the ratio is just 1. If $k(x)$ is growing monotonically, then the ratio converges to 1 from below and will eventually rise above the constant $1/(1+\delta)$. In fact, it is sufficient to assume that $k(x)$ is an unbounded function. The proof also works when $k(x)$ converges to an integral constant or a sufficiently large constant. The proof *does not* work when $k(x)$ is a small fractional constant (e.g., 2.5). Hence, we exclude this case in the hypothesis of the theorem. (In that case, however, we can prove the same theorem with $c = 2 + \log(1/\epsilon)$, which results in a worse lower bound.) Thus, we may assume that for sufficiently large t

$$\frac{3+\epsilon}{2+2\epsilon} \log_g n' > (1+\delta) \frac{\log n'}{\log k(n')}.$$

Using the third niceness property of $k(x)$, for sufficiently large t

$$(1+\delta)\frac{\log n'}{\log k(n')} > \frac{\log n}{\log k(n)} = \log_{k(n)} n.$$

Thus, Equations 3 and 4 are true, and all of the constraints are satisfied for large enough t. □

In the preceding theorem, we make some additional assumptions about $k(n)$ beyond niceness to show that Equation 4 holds. If we are willing to settle for a slightly worse lower bound, we can prove a similar result for all nice approximation functions $k(n)$. Alternatively, we can make the assumption that $\epsilon \leq 1/4$. The proofs for each case is nearly identical to the proof of Theorem 15 except we use the fact that the ratio $\log k(n')/\log\lceil k(n')\rceil$ is bounded below by $\log 2/\log 3 \approx 0.6309$ since $k(n) \geq 2$ for all n.

Corollary 16 Let $k(n)$ be a nice approximation factor. Then, for all oracles X, no polynomial time function can approximate $\omega(G)$ within a factor of $k(n)$ using $\lfloor \log\log_{k(n)} n - c \rfloor$ or fewer queries to X unless P = NP, where $c = 2 + \log(1/\epsilon)$.

Corollary 17 Let $k(n)$ be a nice approximation factor. If $\epsilon \leq 1/4$, then for all oracles X, no polynomial time function can approximate $\omega(G)$ within a factor of $k(n)$ using $\lfloor \log\log_{k(n)} n - c \rfloor$ or fewer queries to X unless P = NP, where $c = 1 + \log(1/\epsilon)$.

5 Approximating the chromatic number

The results that we have stated so far also hold for many other NP-optimization problems. For any NP-optimization problem where the solutions range from 1 to n (or even n^a), the upper bound on the number of queries needed to approximate the problem can be derived easily from the techniques in Section 3. To show that the lower bounds hold we need a reduction from SAT to the new problem similar to the one for Clique in Lemma 2. Recently, Lund and Yannakakis have discovered such reductions for Graph Coloring and some related problems [LY94]. To repeat the proof, special attention must be given to any differences that may arise between minimization and maximization (see Section 7). Thus, we could obtain results analogous to the ones in Theorem 15 for the problem of approximating the chromatic number of a graph.

In this section we take an alternative approach and prove the lower bounds using an approximation preserving reduction from the maximum clique size problem to the chromatic number of a graph [LY94]. However, this reduction increases the size of the graphs, so the proof does not produce the best lower bounds. This approach can be extended to most of the problems which Lund and Yannakakis show to have approximability properties similar to that of Graph Coloring. We can show that Clique Partition, Clique Cover and Biclique Cover have similar lower bounds. This follows from approximation preserving reductions due to Simon [Sim90] for Clique Cover and Biclique Cover. These reductions preserve the approximation ratio within $1+\epsilon$ for any $\epsilon > 0$ where the new problems have size $n^{O(1/\epsilon)}$. On the other hand, we have not been able to obtain a similar result for Fractional Chromatic

Number. The reason is that the reduction there only preserves the approximation ratio with a $\log n$ multiplicative factor.

In the following, let $\alpha(G)$ and $\chi(G)$ denote, respectively, the size of the largest independent set and the chromatic number of a graph G. The next lemma is an application of the results of Lund and Yannakakis.

Lemma 18 There exists a polynomial time transformation $T(G)$ such that for all graphs G with n vertices and for all primes p, with $n \leq p \leq 2n$, $H = T(G)$ has the property that

$$\frac{p^3 n^2}{\alpha(G)} \leq \chi(H) \leq \frac{p^3 n^2 (1 + 1/n)}{\alpha(G)}.$$

Furthermore $|H| = n^2 \cdot p^5 \leq 32 \cdot n^7$.

Proof: By Proposition 4.1[2] in the appendix of Lund-Yannakakis [LY94], there exists a polynomial time transformation $T'(G, p, r)$ such that if G is a graph, p is a prime, r is a number, $p^2 \geq r \geq \alpha(G)$ and $p \geq n$, and $H = T'(G, p, r)$ then

$$p^3 \cdot \frac{r}{\alpha(G)} \leq \chi(H) \leq p^3 \cdot \left\lceil \frac{r}{\alpha(G)} \right\rceil.$$

Given a graph G on n vertices we define T as follows. Let $r = n^2$ and find a prime p such that $n \leq p \leq 2n$ (such primes exist by Bertrand's Theorem and can be found easily since the length of the input is n not $\log n$). Let $T(G) = T'(G, p, r)$. If $H = T(G)$ then

$$p^3 \cdot \frac{n^2}{\alpha(G)} \leq \chi(H) \leq p^3 \cdot \left\lceil \frac{n^2}{\alpha(G)} \right\rceil < p^3 \cdot \frac{n^2(1 + 1/n)}{\alpha(G)}.$$

□

The following theorem shows that we can derive a lower bound on the complexity of approximating the chromatic number of a graph using the lemma above.

Theorem 19 Let $k(n)$ be a nice approximation factor such that for large n, $k(n^8) < n/2$. Then for all oracles X no polynomial time function can approximate $\chi(G)$ within a factor of $k(n)$ using $\lfloor \log \log_{k(n)} n - c \rfloor$ or fewer queries to X unless P = NP, where $c = 6 + \log(1/\epsilon)$.

Proof: We reduce approximating the clique size of a graph Q to approximating the chromatic number of a graph H. If $\chi(H)$ can be approximated using too few queries, then the lower bound on approximating $\omega(Q)$ from Corollary 16 would be violated and we can conclude that P = NP.

We are given a nice approximation factor $k(n)$. Suppose that some polynomial time algorithm $A(G)$ approximates $\chi(G)$ within a factor of $k(n)$ using $\lfloor \log \log_{k(n)} n - c \rfloor$ queries to X for all graphs G with n vertices. Let $k'(n) = k(n^8)$. It is simple to check that $2k'(n)$ is also a nice approximation factor.

[2]Proposition 4.1 as stated by Lund and Yannakakis requires that $p > r$ but the proofs show in fact that $p^2 > r$ and $p \geq n$ are sufficient. We use this proposition instead of their main theorem because it deals with general graphs instead of the special graphs produced by the reduction in Lemma 2.

Now, given any graph Q with n vertices, we approximate $\omega(Q)$ within a factor of $2k'(n)$ as follows. Construct $H' = T(Q')$ using Lemma 18, where Q' is the complement of Q. So, we know that $\alpha(Q') = \omega(Q)$ and $|H'| \leq 32 \cdot n^7$. Now, we construct the graph H by adding dummy vertices to H' so that $|H| = n^8 = N$ and $\chi(H) = \chi(H')$. Finally, we use the algorithm A to compute an approximation of $\chi(H)$ within a factor of $k(N)$. This uses no more than

$$\left\lfloor \log \log_{k(N)} N - 6 - \log(1/\epsilon) \right\rfloor = \left\lfloor \log \log n - \log \log k(n^8) - 3 - \log(1/\epsilon) \right\rfloor$$

queries to X. Since $A(H)$ is a factor $k(N)$ approximation, we know

$$\chi(H) \leq A(H) \leq k(N)\chi(H).$$

From Lemma 18, we also know that

$$\frac{p^3 n^2}{\alpha(Q')} \leq \chi(H') < \frac{p^3 n^2 (1 + 1/n)}{\alpha(Q')}.$$

Since $\alpha(Q') = \omega(Q)$ and $\chi(H) = \chi(H')$, we obtain

$$\frac{\omega(Q)}{k(N)(1 + 1/n)} \leq \frac{p^3 n^2}{A(H)} \leq \omega(Q).$$

Thus, the value $p^3 n^2 / A(H)$ approximates $\omega(Q)$ within a factor of

$$k(N)(1 + 1/n) \leq 2k(n^8) = 2k'(n).$$

Since $2k'(n)$ is a nice approximation factor, by Corollary 16, approximating $\omega(Q)$ within factor $2k'(n)$ has a lower bound of

$$\left\lfloor \log \log_{2k'(n)} n - 2 - \log(1/\epsilon) \right\rfloor = \left\lfloor \log \log n - \log \log 2k(n^8) - 2 - \log(1/\epsilon) \right\rfloor.$$

Finally, since $(\log \log k(n^8)) + 1 > \log((\log k(n^8)) + 1)$, computing $A(H)$ used no more than $\left\lfloor \log \log_{2k'(n)} n - 2 - \log(1/\epsilon) \right\rfloor$ queries to X. Thus, if such an algorithm exists, P = NP. □

Theorem 19 decreases the lower bound of Corollary 16 by 4 queries. This decrease is due in part to the fact that $|H| \approx |Q|^7$. Thus, a more efficient reduction from clique size to chromatic number would yield a tighter lower bound. Also, for specific approximation factors, especially where the relationship between $k(n)$ and $k(n^7)$ is explicit, we can obtain slightly better lower bounds by reproducing the proof of Theorem 15.

6 The value of ϵ

The lower bound results in the preceding sections depend on the value of the constant ϵ, where $0 < \epsilon \leq 1$. Recall that this is the same ϵ used by Arora *et al.* to show no polynomial time algorithm can approximate $\omega(G)$ within a factor of n^ϵ unless P = NP. Note that a

larger value of ϵ indicates a better non-approximability result which in turn provides tighter upper and lower bounds in the results of previous sections. Also, recall that for bounded query classes even an improvement of one query is significant.

Currently, the exact value of ϵ is not known. However, by weakening the assumption that P $\neq$ NP to BPP $\neq$ NP, Bellare, Goldwasser, Lund and Russell [BGLR93] have shown that no polynomial time algorithm can approximate $\omega(G)$ within a factor of $n^{1/30-o(1)}$. A further improvement was made by Bellare and Sudan [BS94] who showed that no polynomial time algorithm can approximate $\omega(G)$ within a factor of $n^{1/5-o(1)}$ unless NP = ZPP. In this section, we use these results to obtain explicit lower bounds on the number of queries needed to approximate $\omega(G)$ under the stronger assumption that RP $\neq$ NP.

To prove these lower bound results, we need to adapt the proof techniques of the previous section to work with randomized functions instead of deterministic functions. A naive approach would use a straightforward modification of Lemma 2 to produce a randomized reduction f from SAT to Clique Size such that:

$$F \in \text{SAT} \Longrightarrow \text{Prob}_z[\ \omega(f(F,z)) = t^b\ ] = 1$$
$$F \notin \text{SAT} \Longrightarrow \text{Prob}_z[\ \omega(f(F,z)) < t^s\ ] \geq 1 - \delta.$$

The difficulty with this approach is that the randomized version of Lemma 10 will use this randomized reduction polynomially many times. Thus, when we show that RP = NP if the lower bounds are violated, we have to be very careful with the value of δ to make certain that the success probability of the overall procedure remains high enough. Such detailed analysis is possible using the techniques developed by Rohatgi [Roh95]. However, the analysis can be made much simpler by observing that the randomized reduction from SAT to Clique Size can be achieved with "uniform" probability — that is, the same random string z can be used to correctly reduce any instance of SAT of a certain length. In the following, let $|F|$ denote the length of the encoding of the Boolean formula F.

Lemma 20 There exist constants s, b and d, with $0 < s < b < d$, polynomials $p(\cdot)$ and $q(\cdot)$ and a deterministic polynomial time function f such that

$$(\forall z \leq p(t))\ (\forall F,\ |F| = t)[\ F \in \text{SAT} \Longrightarrow \omega(f(F,z)) = t^b\ ], \quad \text{and}$$
$$\text{Prob}_{z \leq p(t)}[\ (\forall F,\ |F| = t)[F \notin \text{SAT} \Longrightarrow \omega(f(F,z)) < t^s]\ ] \geq 1 - 2^{-q(t)},$$

where for each Boolean formula F, $|F| = t$, and each random string $z \leq p(t)$, $f(F,z)$ produces a graph with t^d vertices.

Proof Sketch: This reduction is implicit in the work of Zuckerman [Zuc93], we include a proof sketch for completeness. In this proof, the reduction f constructs a graph G from the formula F and a random string z. The random string z is used to choose a disperser graph H which allows f to amplify probability bounds. Choosing this disperser is the only random step used by f. If the randomly chosen H is indeed a disperser, then the same H can be used to reduce any formula F with t variables to a graph G. Hence, the success probability of the reduction is independent of the particular formula F.

The starting point of the proof is not the deterministic reduction in Lemma 2, but the probabilistically checkable proof for SAT. As reported by Arora *et al.*, there is a probabilistically checkable proof for SAT where the verifier V uses $c_1 \log n$ random bits and looks at c_2 bits of the proof such that

$$F \in \text{SAT} \Longrightarrow \exists \pi,\ \forall z \in \{0,1\}^{c_1 \log n},\ V^{\pi}(F,z) \text{ accepts}$$

$$F \notin \text{SAT} \Longrightarrow \forall \pi,\ \text{Prob}_{z \in \{0,1\}^{c_1 \log n}}[V^{\pi}(F,z) \text{ accepts}] < 1/2.$$

From the verifier V and the input F with t variables, we can use the construction of Feige, Goldwasser *et al.* [FGL+91] to construct a graph G such that

$$F \in \text{SAT} \Longrightarrow \omega(G) = t^{c_1}$$
$$F \notin \text{SAT} \Longrightarrow \omega(G) < \tfrac{1}{2} \cdot t^{c_1}.$$

In this construction, each vertex of G represents one computation path of the verifier V (for every possible random string and every sequence of answer bits from the proof). An edge is added between two vertices, if some proof π contains the answer bits which agree with both computation paths.

The construction above gives an approximation "gap" of only 1/2. To obtain better results, we have to decrease the probability that V accepts an incorrect proof when $F \notin \text{SAT}$ *without using too many additional random bits.* This decrease can be achieved deterministically [CW89], which leads to Lemma 2, but then we lose track of the values of b, s and d. Alternatively, as Zuckerman [Zuc93] pointed out, we can decrease the verifier's error randomly.

The key idea to Zuckerman's construction is to use special graphs called dispersers which were first introduced by Sipser [Sip86]. For our purposes, a disperser may be defined as a bipartite graph with t^{c_3} vertices on the left and t^{c_1} vertices on the right such that each left vertex has degree $D = c_3 \log t + 2$ and every set of t^{c_1} left vertices is connected to at least $t^{c_1}/2$ right vertices. The value of the constant c_3 will be chosen later. By a theorem which Zuckerman attributes to Sipser [Zuc93, Theorem 3], such a disperser can be randomly generated with probability $1 - 2^{-t^{c_1}}$ by choosing the D neighbors of each left vertex randomly.

Suppose that we have a disperser H. We use H to construct a new verifier V'. We interpret each left vertex of H as a random string for V' and each right vertex of H as a random string for V. V' simulates V by randomly choosing a left vertex of H. This uses $c_3 \log t$ random bits. Let $z_1, \ldots, z_D$ be the right vertices connected to the chosen left vertex. V' simulates V D times using each of $z_1, \ldots, z_D$ as the random string for V. If every simulation of V accepts the proof π, then V' accepts the proof π. The complete simulation uses $c_3 \log t$ random bits and looks at Dc_2 bits of the proof.

Clearly, if $F \in \text{SAT}$, then V' will always accept. In fact, V' will accept even when H does not have the desired properties. Conversely, consider the case when $F \notin \text{SAT}$. We want to show that V' accepts with probability less than t^{c_1}/t^{c_3}. So, suppose that V' accepted with probability $\geq t^{c_1}/t^{c_3}$. Then, t^{c_1} of the left vertices cause V' to accept. Thus, all the right vertices connected to these left vertices must cause V to accept. Since H is a disperser with the properties mentioned above, at least $t^{c_1}/2$ right vertices must cause V to accept. This contradicts our assumption about the error probability of V. Thus, when $F \notin \text{SAT}$, V' accepts a proof π with probability less than t^{c_1}/t^{c_3}.

Now we construct a graph G from V' as described above. Since V' uses $c_3 \log t$ random bits and looks at Dc_2 bits of the proof, the graph G will have $n = 2^{c_3 \log t + Dc_2}$ vertices. When $F \in$ SAT, G has a "big" clique of size t^{c_3}. When $F \notin$ SAT, G has a "small" clique of size no more than t^{c_1}. The ratio of the size of the "big" clique to the size of the "small" clique expressed in terms of n is:

$$t^{c_3 - c_1} = n^{(c_3 - c_1)\log t/(c_3 \log t + Dc_2)}.$$

Substituting $c_3 \log t + 2$ for D, we derive:

$$\epsilon = \frac{b-s}{d} = \frac{1 - c_1/c_3}{1 + c_2 + 2c_2/(c_3 \log t)}.$$

Thus, for all $\delta > 0$, we can have $\epsilon > (1+c_2)^{-1} - \delta$ by choosing c_3 to be large enough. Recall that c_2 is the number of bits in the proof that the verifier reads. This calculation shows the effect of c_2 on the value of ϵ. Finally, observe that the only random step used in the reduction is choosing the disperser H. □

The results of Bellare, Goldwasser, Lund and Russell [BGLR93] can be used to show that in Lemma 20 the ratio $(b-s)/d = 1/30 - o(1)$, because they produce a verifier for SAT that uses only 29 bits of the proof and follow Zuckerman's construction as described above. However, we are unable to exploit the results of Bellare and Sudan [BS94], because they use randomness for sampling — not just to generate pseudo-random bits. Nevertheless, we can give explicit lower bounds on the query complexity of approximating $\omega(G)$.

Theorem 21 Let $k(n)$ be a nice approximation factor. Then, for all oracles X, no polynomial time function can approximate $\omega(G)$ within a factor of $k(n)$ using $\lfloor \log \log_{k(n)} n - 6 \rfloor$ or fewer queries to X unless RP = NP.

Proof Sketch: To prove this theorem, we simply follow the proof of Theorem 15 except that we use $\epsilon = 1/31$ and Lemma 20 instead of Lemma 2 to reduce SAT to Clique Size. Since the success probability of the reduction is independent of the formula F, repeated use of the reduction does not decrease the success probability of the overall procedure. Again, the only random step in the entire procedure is randomly choosing a disperser graph H. This is also the case when we use the tree pruning procedure in Lemma 10 to look for a satisfying assignment for the given Boolean formula. Since our procedure accepts a formula only when a satisfying assignment is found, it will never accept an unsatisfiable formula. The procedure may reject a satisfiable formula if the graph H turns out not to be a disperser. However, the probability of this happening is small. Thus, the overall procedure is an RP algorithm for SAT.

Finally, note that in Equation 2 of Theorem 15, we need to show that

$$\frac{2}{\epsilon} \log_g \frac{g}{g-1} + \log_{k(n)} n < \frac{2}{\epsilon} \log_g t^{b-s}.$$

In this proof, the value of ϵ is known, so we can rewrite the $2/\epsilon$ as

$$\frac{2}{\epsilon} = \frac{2}{(1+\epsilon)} \frac{(1+\epsilon)}{\epsilon} = 1.9375 \cdot \frac{1+\epsilon}{\epsilon} = (0.1375 + 1.8) \cdot \frac{b-s+d}{b-s}.$$

Then, as before,

$$\frac{2}{\epsilon}\log_g \frac{g}{g-1} < 0.1375 \cdot \log_g n'$$

because $g/(g-1) \leq 2$. Also, we do not need any additional assumptions on $k(n)$ to show

$$1.8 \cdot \log_g n' = 1.8 \cdot \frac{\log n'}{\log k(n')} \cdot \frac{\log k(n')}{\log \lceil k(n') \rceil} > (1+\delta)\frac{\log n'}{\log k(n')}$$

because $\log k(n')/\log \lceil k(n') \rceil \geq \log 2/\log 3 \approx 0.6309$. Hence, $1.8 \cdot \log k(n')/\log \lceil k(n') \rceil > 1.1$ and we can let $\delta = 0.1$. Thus, the assumption that $k(n)$ is nice and $c = 1 + \log 31 < 6$ suffices. □

Using the proof techniques described above we can also extend our results to lower bounds for approximating the chromatic number.

Corollary 22 Let $k(n)$ be a nice approximation factor such that for large n, $k(n) < n/2$. Then for all oracles X no polynomial time function can approximate $\chi(G)$ within a factor of $k(n)$ using $\lfloor \log \log_{k(n)} n - 10 \rfloor$ or fewer queries to X unless RP = NP.

7 Lower bounds for set cover

An instance of the Set Cover problem is a set system $\mathcal{S} = (n; S_1, \ldots, S_m)$ such that for all i, $S_i \subseteq \{1, \ldots, n\}$. We are asked to find the size of the smallest collection of S_i which covers $\{1, \ldots, n\}$. We denote the size of the minimum cover of $\mathcal{S}$ as $\texttt{SETCOVER}(\mathcal{S})$.

As we have discussed in the section on upper bounds, the Set Cover problem has a different complexity compared to Clique or Chromatic Number because there exist polynomial time algorithms that approximate the size of the minimum set cover within a factor of $\ln n + 1$. In this section we derive a lower bound on the complexity of approximating the size of the minimum set covering in terms of the number of queries to SAT.

One difficulty arises when we apply our techniques to minimization problems. To illustrate this, consider the construction of the graph H in Section 4.2. We use the reduction from SAT to Clique to construct a graph G_1 from a formula F_1. This reduction has the special property that if $F_1 \in$ SAT, then $\omega(G_1) = t^b$. This *equality* is very important, because it allows us to put only two copies of G_2 in H. If we only knew that $\omega(G_1) \geq t^b$, then $\omega(G_1)$ could be as large as m. Thus, to ensure a large enough gap between the case where $F_1 \in$ SAT and $F_2 \notin$ SAT from the case where $F_1, F_2 \in$ SAT, we would have to use $2m/t^b$ copies of G_2. This would make the graph H too big and produce very bad lower bounds in the general theorem.

If equality is not possible, then we can settle for a good upper bound. For example, if the reduction from SAT to Clique guaranteed that $F_1 \in \text{SAT} \Longrightarrow t^b \leq \omega(G_1) \leq 3t^b$, then we only need to put 6 copies of G_2 in H. In the general case, we would obtain a lower bound of $\log \log_{3k(n)} n - c$.

The reduction from SAT to Set Cover in [LY94] does not give an upper bound on the size of the minimum set cover when the original formula is unsatisfiable. Thus, we must use the following lemma and theorem.

Lemma 23 There exists a constant CONST such that for all $l, m \geq \texttt{CONST}$ where $l \leq m$ and $\log \ln m < 0.09l$, there exist a set B and subsets $C_1, C_2, \ldots, C_m \subseteq B$ such that:

1. For any sequence of indices $1 \leq i_1 < i_2 < \cdots < i_l \leq m$, no collection $D_{i_1}, D_{i_2}, \ldots, D_{i_l}$ covers B where D_{i_j} is either C_{i_j} or the complement of C_{i_j}.
2. $C_1, \ldots, C_{1.1l}$ do cover B.
3. $|B| = (l + l\ln(m) + 2)2^l$.

Furthermore, there exists a probabilistic Turing Machine which, on input l, m (in binary) produces such a set system with probability 2/3.

Proof: Let $B = \{1, \ldots, (l + l\ln(m) + 2)2^l\}$. Let the subsets $C_1, \ldots, C_m$ be a collection of subsets of B chosen randomly and independently — i.e., for each $x \in B$ and each C_i, $x \in C_i$ with probability one half. We show that with probability over 2/3, this collection suffices. Fix $D_{i_1}, D_{i_2}, \ldots, D_{i_l}$ as in the statement of the lemma. The probability that $B = D_{i_1} \cup \cdots \cup D_{i_l}$ is $(1 - (1/2)^l)^{|B|}$. The number of different D_{i_j}'s is at most $2^l\binom{m}{l}$. Thus the probability that some collection of $D_{i_1}, \ldots, D_{i_l}$ covers B is bounded by

$$2^l \binom{m}{l} (1 - (1/2)^l)^{|B|} \leq e^{l + l\ln(m) - |B|/2^l} < 1/6.$$

Hence the probability that item 1 occurs is at least 5/6.

Secondly, note that the probability that the first $1.1l$ sets cover B is

$$(1 - (1/2)^{1.1l})^{|B|} \geq e^{-2(l + l\ln(m) + 2)2^l / 2^{1.1l}} = e^{-2(l + l\ln(m) + 2)2^{-0.1l}},$$

using the fact that $(1 - x) \geq e^{-2x}$ for $x \in [0, 1/2]$. We need this quantity to be greater than $5/6 = e^{-c}$ where $c = \ln(6/5) > 0$. Hence we need

$$\begin{aligned} 2(l + l\ln(m) + 2)2^{-0.1l} &< 5/6 \\ 12/5(l + l\ln(m) + 2) &< 2^{0.1l} \\ \log 12/5 + \log l + \log(\ln(m) + 2) &< 0.1l \end{aligned}$$

Since $\log \ln m < 0.09l$, this holds for large enough l and m. Therefore, the desired constant CONST can be found. Hence the probability of satisfying item 2 is at least 5/6. Since the probability of satisfying item 1 is $\geq 5/6$ and the probability of satisfying item 2 is $\geq 5/6$, the probability of satisfying both is at least 2/3. □

Theorem 24 Given a formula φ, let $\mathcal{S}_\varphi$ be the instance of Set Cover described below. Let N be the size of $\mathcal{S}_\varphi$. Then, there exists an integer K (depending only on the size of φ) such that,

$$\varphi \in \text{SAT} \Longrightarrow \texttt{SETCOVER}(\mathcal{S}_\varphi) = K$$

$$\varphi \notin \text{SAT} \Longrightarrow \frac{0.99K}{2} \log N \leq \texttt{SETCOVER}(\mathcal{S}_\varphi) \leq \frac{1.1K}{2} \log N,$$

where the last property holds with probability at least 2/3. Furthermore, the reduction can be applied to any number of formulas φ of the same size, and with probability 2/3, all the instances of Set Cover obtained will have the property above.

Proof: The proof of this theorem is a modification of the construction by Lund and Yannakakis [LY94]. In the rest of this proof we assume that the reader is familiar with the notation and the proof in [LY94]. Given a formula φ, we carry out the construction in Section 3.1 of [LY94] except that we use the sets $B, C_1, \ldots, C_m$ as specified in Lemma 23 as our building blocks. Using the ideas in [LY94] we obtain:

1. $\varphi \in$ SAT implies $\texttt{SETCOVER}(\mathcal{S}_\varphi) = K$, where $K = |Q_1| + |Q_2|$.
2. $\varphi \notin$ SAT implies $\texttt{SETCOVER}(\mathcal{S}_\varphi) \geq \frac{l}{2}(1-o(1))K$. Moreover, $\texttt{SETCOVER}(\mathcal{S}_\varphi) \leq 1.1l(|Q_2|)$, since the first $1.1l$ answers for every query in Q_2 cover all the points, by Lemma 23. Then, since $|Q_1| = |Q_2|$,

$$\frac{l}{2}(1 - o(1))K \leq \texttt{SETCOVER}(\mathcal{S}_\varphi) \leq \frac{1.1l}{2}K.$$

Now the theorem follows since l can be chosen such that $l = (1 - \epsilon)\log N$ for any $\epsilon > 0$. Also, note that $m = 2^{O(l)}$ and thus we can apply Lemma 23.

Furthermore, note that once we have chosen one set system from Lemma 23, we can use it in any number of reductions involving instances of the same size. Thus if the set system has the required property then all the reduced instances will have the required properties. □

The consequence of this theorem is that there exists a *randomized* reduction from SAT to Set Cover which runs in time $O(n^{\text{polylog}\, n})$. This reduction allows us to duplicate the construction of Theorem 15 for Set Cover and obtain the following lower bound.

Theorem 25 Let $k : \mathbb{N} \to \mathbb{R}$ be a function such that for all n large enough, $2 \leq k(n) < n$; $m > n$ implies $k(m) \geq k(n)$; and $\forall \delta > 0,\ m > n$ implies that

$$(1 + \delta)\frac{\log\log m}{\log k(m)} > \frac{\log\log n}{\log k(n)}.$$

Let $\mathcal{S}$ be an instance of Set Cover. Then, for all oracles X, no polynomial time function can approximate the size of the minimum set cover within a factor of $k(n)$ using $\log\log_{k(n)} \ln n - 1$ or fewer queries to X unless NP $\subseteq$ RTIME$[n^{\text{polylog}\, n}]$.

Proof: This proof is analogous to the proof of Theorem 21. We start with the promise problem $\mathcal{P}_r$, with r to be chosen later. Since our construction is in time $O(n^{\text{polylog}\, n})$, we need a different lower bound on the complexity of $\mathcal{P}_r$. Using the proof technique of Lemma 10, one can show that for $r = O(\text{polylog}\, n)$, if some DTIME$[n^{\text{polylog}\, n}]$ machine solves $\mathcal{P}_r$ using fewer than $\lceil \log(r + 1) \rceil$ queries to any oracle X, then NP $\subseteq$ DTIME$[n^{\text{polylog}\, n}]$. As in Theorem 21, it is important that in Theorem 24 the randomized reduction from SAT to Set Cover can be used repeatedly without decreasing the success probability of the overall procedure.

Now, let $F_1, \ldots, F_r$ be a sequence of Boolean formulas with t variables each which satisfies the promise condition of the promise problem $\mathcal{P}_r$. We use Theorem 24 to construct r instances

of Set Cover $\mathcal{S}_1, \ldots, \mathcal{S}_r$. Let m be the size of the underlying set of $\mathcal{S}_i$. Since the construction takes time $O(n^{\text{polylog}\, n})$, m is $O(t^{\text{polylog}\, t})$. We know with probability 2/3 that for each i:

$$F_i \in \text{SAT} \implies \texttt{SETCOVER}(\mathcal{S}_i) = K$$

$$F_i \notin \text{SAT} \implies \frac{0.99K}{2} \log m \leq \texttt{SETCOVER}(\mathcal{S}_i) \leq \frac{1.1K}{2} \log m.$$

We will now restrict our attention to this case. We combine the r instances of Set Cover into a single instance of Set Cover $\mathcal{T}$:

$$\mathcal{T} = g^{r-1} \otimes \mathcal{S}_1 \oplus g^{r-2} \otimes \mathcal{S}_2 \oplus \cdots \oplus g \otimes \mathcal{S}_{r-1} \oplus \mathcal{S}_r.$$

Let n be the size of the underlying set for $\mathcal{T}$. We define $\oplus$ and $\otimes$ so that

$$\texttt{SETCOVER}(\mathcal{S}_1 \oplus \mathcal{S}_2) = \texttt{SETCOVER}(\mathcal{S}_1) + \texttt{SETCOVER}(\mathcal{S}_2),$$

and for any positive integer a

$$\texttt{SETCOVER}(a \otimes \mathcal{S}) = a \cdot \texttt{SETCOVER}(\mathcal{S}).$$

This is accomplished as follows. Let $\mathcal{S}_1 = (n_1; S_{1,1}, \ldots, S_{1,p})$ and $\mathcal{S}_2 = (n_2; S_{2,1}, \ldots, S_{2,q})$ be two instances of Set Cover. We define $\mathcal{S}_1 \oplus \mathcal{S}_2$ to be $(n_1 + n_2; S_{1,1}, \ldots, S_{1,p}, S'_{2,1}, \ldots, S'_{2,q})$, where $S'_{2,i} = \{x + n_1 \mid x \in S_{2,i}\}$. Then we can define $a \otimes \mathcal{S}$ simply as $\mathcal{S} \oplus \cdots \oplus \mathcal{S}$ repeated a times.

As in the proof of Theorem 15, the value of g in the construction of $\mathcal{T}$ will be chosen later. Note that we construct $\mathcal{T}$ using g^{r-1} copies of $\mathcal{S}_1$ instead of g^{r-1} copies of $\mathcal{S}_r$. This is "backwards" compared to the construction H in Theorem 15. We need to make this change because $\texttt{SETCOVER}(\mathcal{S}_i)$ is small when $F_i \in \text{SAT}$ and large when $F_i \notin \text{SAT}$. (Again, backwards compared to Clique.) Then, a good approximation of $\texttt{SETCOVER}(\mathcal{T})$ will solve the promise problem $\mathcal{P}_r$. However, here the approximation must be within a factor of $0.9g$ instead of g because we only know that $\texttt{SETCOVER}(\mathcal{S}_i)$ is in the interval between $\frac{0.99}{2} K \log m$ and $\frac{1.1}{2} K \log m$ when $F_i \notin \text{SAT}$.

In this construction, when there are exactly z satisfiable formulas in $F_1, \ldots, F_r$, we have the following bounds:

$$\frac{0.99}{2} v_{r-z} K \log m \leq \texttt{SETCOVER}(\mathcal{T}) \leq v_r K + \frac{1.1}{2} v_{r-z} K \log m.$$

Thus, we can obtain a lower bound for approximating the size of the minimum set cover (assuming NP $\not\subseteq$ RTIME$[n^{\text{polylog}\, n}]$) if we show that there exist r and g which satisfy the following constraints. Recall that m and n are the sizes of the underlying sets for $\mathcal{S}$ and $\mathcal{T}$ respectively. The value of m is expressible in terms of t. To make our notation simpler, we express g and r in terms of m instead of t or drop the argument altogether.

Constraint 1: $g(m)\nu_r < \frac{1.1}{2} \log m$.

Constraint 2: $k(n) \leq 0.9g(m)$.

Constraint 3: $\left\lfloor \log \log_{k(n)} \ln n - 1 \right\rfloor < \lceil \log(r(m) + 1) \rceil$.

For this proof, we let $n' = \frac{1.1}{2} m \log m$ and choose g and r as follows:

- $g(m) = \lceil k(n')/0.9 \rceil$.
- $r = \left\lfloor \log_g \frac{(\frac{1.1}{2} \log m) \cdot (g-1)}{g} \right\rfloor$.

As in the proof of Theorem 15, our choice of r implies that Constraint 1 holds and that $n' > n$. Since $k(n)$ is non-decreasing almost everywhere, it follows that $k(n) \leq 0.9g(m)$ and that Constraint 2 also holds.

Finally, we show that Constraint 3 holds. It suffices to show that the following equations hold. (They are analogous to Equations 3 and 4 in Theorem 15.)

$$2 \cdot \log_g \frac{g}{g-1} < 0.25 \cdot \log_g \left(\frac{1.1}{2} \log m \right) \tag{5}$$

$$\log_{k(n)} \ln n < 1.75 \cdot \log_g \left(\frac{1.1}{2} \log m \right) \tag{6}$$

Equation 5 is satisfied for m large since $g \geq 2$, $g/(g-1) \leq 2$. Equation 6 is proved as follows. We start with the "niceness" assumptions on $k(n)$ and obtain

$$\log_{k(n)} \ln n \leq \frac{\log \log n}{\log k(n)} < 1.05 \cdot \frac{\log \log n'}{\log k(n')}.$$

Recall that $n' = \frac{1.1}{2} m \log m$. Then for large m, $(\log n')^{1.05} < (\log m)^{1.1}$. Thus, we have:

$$1.05 \cdot \frac{\log \log n'}{\log k(n')} < 1.1 \cdot \frac{\log \log m}{\log k(n')}.$$

Then, using the fact that for $x \geq 2$, $\log 3 \cdot \log x / \log \lceil x/0.9 \rceil > 1$, we have:

$$1.1 \cdot \frac{\log \log m}{\log k(n')} < 1.1 \cdot \log 3 \cdot \frac{\log \log m}{\log k(n')} \cdot \frac{\log k(n')}{\log \lceil k(n')/0.9 \rceil} = 1.1 \cdot \log 3 \cdot \frac{\log \log m}{\log g}.$$

Now, $\log 3 \approx 1.584963\ldots$, so $1.1 \cdot \log 3 < 1.75 - 0.005$. Also, we know that for m large enough, $0.005 \log \log m > 1.75 \cdot \log(2/1.1)$. Therefore,

$$1.1 \cdot \log 3 \cdot \frac{\log \log m}{\log g} < \frac{1.75 \cdot \log \log m - 0.005 \cdot \log \log m}{\log g} < 1.75 \cdot \log_g \left(\frac{1.1}{2} \log m \right).$$

Therefore, Equation 6 holds and we have completed the proof. □

Updates

Since the original submission of this paper, additional connections between bounded query classes and NP-approximation problems have been discovered. Chang [Cha94a] showed

that approximating clique size is actually complete for certain bounded query classes. This completeness produces reductions between NP-approximation problems (e.g., from approximating the chromatic number to approximating clique size). In addition, bounded query classes can also be used to measure the difficulty of finding the vertices of an approximate clique, not just its size [Cha94b]. Also, Crescenzi, Kann, Silvestri and Trevisan [CKST95] have shown that finding the vertices of the largest clique cannot be reduced to finding the vertices of an approximate clique unless the Polynomial Hierarchy collapses. Bounded query classes have also been used to measure the hardness of optimization problems [CT94] and to compare various kinds of approximation preserving reductions [CKST95].

Acknowledgements

The authors would like to thank Richard Beigel and Suresh Chari for many helpful discussions. Thanks also go to Samir Khuller, Martin Kummer and Frank Stephan for proofreading drafts of this paper.

References

[ABG90] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 232–243, 1990.

[AG88] A. Amir and W. I. Gasarch. Polynomial terse sets. *Information and Computation*, 77:37–56, April 1988.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[Bei87] R. Beigel. A structural theorem that depends quantitatively on the complexity of SAT. In *Proceedings of the 2nd Structure in Complexity Theory Conference*, pages 28–32, June 1987.

[Bei91] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84(2):199–223, July 1991.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *ACM Symposium on Theory of Computing*, pages 294–304, 1993.

[BH92] R. B. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, 1992.

[BS94] M. Bellare and M. Sudan. Improved non-approximability results. To appear in the *Proceedings of the 26th Symposium on Theory of Computing*, 1994.

[Cha94a] R. Chang. On the query complexity of clique size and maximum satisfiability. In *Proceedings of the 9th Structure in Complexity Theory Conference*, pages 31–42, June 1994.

[Cha94b] R. Chang. Structural complexity column: A machine model for NP-approximation problems and the revenge of the Boolean hierarchy. *Bulletin of the European Association for Theoretical Computer Science*, 54:166–182, October 1994.

[CK90] R. Chang and J. Kadin. The Boolean hierarchy and the polynomial hierarchy: a closer connection. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 169–178, July 1990. To appear in *SIAM Journal on Computing*.

[CKR95] R. Chang, J. Kadin, and P. Rohatgi. On unique satisfiability and the threshold behavior of randomized reductions. *Journal of Computer and System Sciences*, 50(3):359–373, 1995.

[CKST95] P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in approximation classes. To appear in *Proceedings of the 1st Computing and Combinatorics Conference*, August 1995.

[CT94] P. Crescenzi and L. Trevisan. On approximation scheme preserving reducibility and its applications. In *Proceedings of the 14th Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 880 of *Lecture Notes in Computer Science*, pages 330–341. Springer-Verlag, December 1994.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 14–19, 1989.

[FGL+91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 2–12, 1991.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[GKR] W. Gasarch, M. W. Krentel, and K. Rappoport. OptP-completeness as the normal behavior of NP-complete problems. To appear in *Mathematical Systems Theory*.

[HN93] A. Hoene and A. Nickelsen. Counting, selecting, sorting by query-bounded machines. In *Proceedings of the 10th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science. Springer-Verlag, 1993.

[Joh74] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, December 1988.

[Kre88] M. W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990.

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, September 1994.

[Roh95] P. Rohatgi. Saving queries with randomness. *Journal of Computer and System Sciences*, 50(3):476–492, 1995.

[Sha90] A. Shamir. IP = PSPACE. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 11–15, 1990.

[Sim90] H. U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 3:294–310, 1990.

[Sip86] M. Sipser. Expanders, randomness, or time versus space. In *Structure in Complexity Theory*, volume 223 of *Lecture Notes in Computer Science*, pages 325–329. Springer-Verlag, 1986.

[Wag86] K. Wagner. More complicated questions about maxima and minima and some closures of NP. In *Proceedings of the 13th International Colloquium on Automata, Languages, and Programming*, volume 226 of *Lecture Notes in Computer Science*, pages 434–443. Springer-Verlag, 1986.

[WW85] K. Wagner and G. Wechsung. On the Boolean closure of NP. In *Proceedings of the 1985 International Conference on Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 485–493. Springer-Verlag, 1985.

[Zuc93] D. Zuckerman. NP-complete problems have a version that's hard to approximate. In *Proceedings of the 8th Structure in Complexity Theory Conference*, pages 305–312, 1993.