On the Query Complexity of Clique Size and Maximum Satisfiability

Richard Chang[†]
Department of Computer Science
and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21228, USA

November 6, 1995

Abstract

This paper explores the bounded query complexity of approximating the size of the maximum clique in a graph (Clique Size) and the number of simultaneously satisfiable clauses in a 3CNF formula (MaxSat). The results in the paper show that for certain approximation factors, approximating Clique Size and MaxSat are complete for corresponding bounded query classes under metric reductions. The completeness result is important because it shows that queries and approximation are interchangeable: NP queries can be used to solve NP-approximation problems and solutions to NP-approximation problems answer queries to NP oracles. Completeness also shows the existence of approximation preserving reductions from many NP-approximation problems to approximating Clique Size and MaxSat (e.g., from approximating Chromatic Number to approximating Clique Size). Since query complexity is a quantitative complexity measure, these results also provide a framework for comparing the complexities of approximating Clique Size and approximating MaxSat. In addition, this paper examines the query complexity of the minimization version of the satisfiability problem, MinUnsat, and shows that the complexity of approximating MinUnsat is very similar to the complexity of approximating Clique Size. Since MaxSat and MinUnsat share the same solution space, the "approximability" of MaxSat is not due to the intrinsic complexity of satisfiability, but is an artifact of viewing the approximation version of satisfiability as a maximization problem.

[†]Supported in part by NSF Research Grant CCR-9309137 and the University of Maryland Institute for Advanced Computer Studies. Email: chang@cs.umbc.edu.

1 Introduction

The study of NP-approximation problems is a well-established field of theoretical computer science. In the past, this area has been explored primarily from an algorithmic point of view. The main objective of this approach is to determine the extent to which an NP-optimization problem can be approximated by an efficient algorithm. For example, an NP-optimization problem is generally considered "approximable" if there exists a constant factor polynomial time approximation algorithm for the problem. The questions asked in this line of research tend to be qualitative in nature, hence the approximation classes such as MAXNP, MAXSNP and MAXΠ_1 were defined in an attempt to separate "approximable" problems from "non-approximable" ones [PY91, PR90]. Furthermore, these classes are syntax based rather than machine based. In contrast, the traditional complexity classes, such as time and space bounded classes, are resource-bounded complexity classes which allow quantitative comparisons between problems in those classes. In their seminal paper which defined MAXNP, Papadimitriou and Yannakakis [PY91] argue against using a machine model to classify approximation problems, claiming that "computation is an inherently unstable, non-robust mathematical object, in the sense that it can be turned from non-accepting to accepting by changes that would be insignificant in any metric."

It is difficult to argue against such conventional wisdom. However, we would like a model which allows us to address some basic questions about the complexity of approximation problems. For example:

- What computational resources are sufficient to solve the approximation problem?
- Could fewer resources be used?
- Is it easier to find the approximate solution than to find the exact solution?
- Are coarser approximations easier to find than finer approximations?
- How does the complexity of approximating different problems compare to each other?

In summary, we want a computational model which allows us to *quantitatively* measure the complexity of approximation problems.

In this paper, we argue that bounded query complexity is a natural way to measure the complexity of NP-approximation problems. The bounded query class $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ is the set of functions computed by polynomial time Turing machines which ask at most $q(n)$ queries to the SAT oracle. Bounded query classes have been well studied, beginning with Krentel's work which used bounded query classes to classify the difficulty of NP-optimization problems [Kre88]. Subsequent studies showed that for many query bounds $q(n)$ the class $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ is strictly larger than the class $\mathrm{PF}^{\mathrm{SAT}[q(n)-1]}$ [ABG90, AG88, HN93] unless some intractability assumptions are violated. Thus, bounded query complexity is a fine-grained complexity measure and, in our opinion, well suited for measuring the complexity of approximation problems.

In previous work, Chang, Gasarch and Lund [CG93, CGL94] gave upper bounds and relative lower bounds on the number of queries needed to compute an approximation of the maximum clique size of a graph. They showed that for a "nice" approximation factor

$k(n) \geq 2$, $\lceil \log \lceil \log_{k(n)} n \rceil \rceil$ queries to SAT can be used to find an approximation of the maximum clique size within a factor of $k(n)$. Moreover, no polynomial time computable function can find such an approximation with a constant fewer queries to any oracle X unless P = NP. This constant does not depend on the factor $k(n)$.[1] These results do depend upon breakthroughs on the non-approximability of clique size and probabilistically checkable proofs for NP languages [ALM+92]. Chang, Gasarch and Lund also obtained similar upper and lower bounds for approximating the chromatic number of a graph and explored the query complexity of finding the size of the minimum set cover.

In this paper, we show a tighter connection between bounded queries and the complexity of NP-approximation problems by showing that approximating Clique Size within a factor of $k(n)$ is *hard* for a corresponding bounded query class under metric reductions. To do this, we first show that the function $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ is complete for the bounded query classes. For several approximation factors, this hardness result can be restated as completeness results as well. In the following, let a and k be constants such that $a \leq 1$ and $k \geq 1$.

- $(1 + n^{-a})$-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[O(\log n)]}$.
- $(1 + 1/\log^k n)$-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[(k+1)\log\log n + O(1)]}$.
- k-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[\log\log n + O(1)]}$.
- $(\log^k n)$-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[\log\log n - \log\log\log n + O(1)]}$.

These completeness results show that functions which approximate Clique Size are natural representatives of the functions in the bounded query classes. Completeness is important because it shows that bounded queries and approximations are inseparable. Finding better approximations produces answers to more queries to SAT — and answering more queries to SAT generates better approximations. The completeness result also shows that many other NP-approximation problems can be reduced to approximating Clique Size. Moreover, these reductions are approximation preserving in the sense that a better approximation of Clique Size will result in a better approximation of the original problem.

In this paper, we also explore the query complexity of approximating the maximum number of simultaneously satisfiable formulas in a 3CNF formula (MAX3SAT). The MAX3SAT problem is different from the Clique Size problem because there exists a polynomial time algorithm which approximates MAX3SAT within a factor of 4/3. Due to results by Khanna, Motwani, Sudan and Vazirani [KMSV94], we can show that approximating MAX3SAT within a factor of $k(n)$ is hard for the class $\mathrm{PF}^{\mathrm{SAT}[\log\log_{k(n)}(1+\delta')]}$, where δ' is a constant that does not depend on $k(n)$. Again this lower bound differs from the upper bound of $\log\log_{k(n)}(4/3)$ by only a constant number of queries.

Finally, we address the following anomaly. Considering that the decision problems 3SAT and Clique are both NP-complete, why can MAX3SAT be approximated within a constant factor and not Clique Size? We could say that 3SAT is "easier" than Clique. Instead, we argue informally that MAX3SAT is approximable only because it contains padding. We show that by adding a linear amount of padding to Clique Size, the resulting approximation problem has a complexity that is very similar to MAX3SAT. Our intuitive interpretation of

[1] Under the assumption that RP $\neq$ NP, this constant is 6.

this observation is that MAX3SAT contains a lot of "fluff" that is approximable in polynomial time. We then argue that the "non-approximable" part of MAX3SAT requires just as many queries to approximate as Clique Size does. Hence, in its core, MAX3SAT is just as difficult as Clique Size. Note that such comparisons can only be made after we have established a uniform quantitative complexity measure to gauge the difficulty of approximating Clique Size and MAX3SAT. Furthermore, we show that by viewing the approximation version of 3SAT as a minimization problem, which we call MIN3UNSAT, the complexity of the resulting approximation problem is very similar to that of approximating Clique Size. Therefore, we conclude that the "approximability" of MAX3SAT is not due to 3SAT being easier than Clique, but is an artifact of treating the approximation version of 3SAT as a maximization problem.

The rest of this paper is organized as follows. Section 2 explains the terminology and notation used in the paper. Section 3 shows the hardness and completeness of approximating Clique Size. In Section 4, we construct an approximation preserving reduction from Chromatic Number to Clique Size. Section 5 establishes the query complexity of the MAX3SAT problem. Section 6 shows a connection between the structure of bounded query classes and the self-improvability of Clique Size. Section 7 compares the complexities of approximating Clique Size and MAX3SAT. Section 8 shows that by viewing the approximation version of 3SAT as a minimization problem instead of a maximization problem, the resulting NP-approximation problem has the same complexity as approximating Clique Size. Finally in Section 9, we provide some updated references to results from more recent works.

2 Preliminaries

We are interested in two NP-optimization problems: Clique and MAX3SAT. The Clique problem is identifying the vertices of a maximum clique in a graph and the MAX3SAT problem is finding an assignment to the variables in a 3CNF formula which satisfies the largest number of clauses. We will work with a variation of Clique and MAX3SAT, Clique Size and MaxSat, which merely asks for the *size* of the optimum solution. We now define precisely what it means for a function to approximate Clique Size and MaxSat.

Definition 1 Given a graph G with n vertices, we use $|G|$ to denote the number of vertices in G (rather than the length of the encoding of G) and we write $\omega(G)$ for the size of the maximum clique in the graph. We say that a number x is an approximation of $\omega(G)$ within a factor of $k(n)$ if $x \leq \omega(G) < k(n) \cdot x$. To shorten the terminology, we will also say that the number x $k(n)$-approximates $\omega(G)$. Let f be a function such that for all graphs G, $f(G)$ $k(|G|)$-approximates $\omega(G)$. Then we say that the function f $k(\cdot)$-approximates Clique Size.

Definition 2 Let F be a 3CNF formula with n clauses. We use $|F|$ to denote the number of clauses in the formula F (rather than the length of encoding of F), and we define MaxSat(F) to be the maximum number of simultaneously satisfiable clauses in F. Then, a number x is an approximation of MaxSat(F) within a factor of $k(n)$ if $x \leq \text{MaxSat}(F) < k(n) \cdot x$. In this case, we also say that the number x $k(n)$-approximates MaxSat(F). A function f $k(\cdot)$-approximates MaxSat if for all formulas F, $f(F)$ $k(|F|)$-approximates MaxSat.

Traditionally, approximation factors for maximum clause satisfiability are less than 1. E.g., Yannakakis's approximation algorithm for MAX3SAT [Yan92] is usually cited as a 3/4 approximation rather than a 4/3 approximation. In this paper, we use consistent terminology and always state approximation factors that are greater than 1.

Since approximation problems are solved by functions rather than languages, we have to use metric reductions between functions instead of many-one reductions. The following definition of metric reduction is due to Krentel [Kre88]:

Definition 3 Let f and g be two functions. A *metric* reduction from f to g, written $f \leq^{\mathrm{P}}_{\mathrm{mt}} g$, is a pair of polynomial time computable functions T_1 and T_2 such that for all x,

$$f(x) = T_2(x, g(T_1(x))).$$

Intuitively, T_1 transforms the input x into the domain of the function g, then T_2 takes the answer given by $g(T_1(x))$ and computes the value of $f(x)$. So, T_1 and T_2 allow you to solve $f(x)$ if you have a way of computing g.

The complexity measure we use in this paper is the number of oracle queries that a polynomial time Turing machine needs to compute a function.

Definition 4 Let $\mathrm{PF}^{X[q(n)]}$ 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.

Note that for constant k, $\mathrm{PF}^{X[k]}$ is closed under metric reductions, but for growing functions $q(n)$, $\mathrm{PF}^{X[q(n)]}$ might not be closed under metric reductions. For example, suppose that $f \leq^{\mathrm{P}}_{\mathrm{mt}} g$ and $g \in \mathrm{PF}^{\mathrm{SAT}[q(n)]}$. Then, it is possible that $f \notin \mathrm{PF}^{\mathrm{SAT}[q(n)]}$, because the reduction from f to g can stretch the input to g by a polynomial factor. Hence, computing $f(x)$ may require $q(n^{O(1)})$ queries to SAT. As a result, our completeness results are stated for classes like $\mathrm{PF}^{\mathrm{SAT}[\log\log n + O(1)]}$ which are closed under metric reductions. We will return to this topic in Section 6 where we observe a connection between closure under metric reductions and the self-improvability of NP-approximation problems.

In the next section, we will prove that some problems involving sequences of Boolean formulas are $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for corresponding bounded query classes.

Definition 5 For a sequence $F_1, \ldots, F_r$ of Boolean formulas, we define $\#^{\mathrm{SAT}}_r(F_1, \ldots, F_r)$ to be the number of formulas in $\{F_1, \ldots, F_r\}$ which are satisfiable and $\mathrm{RM}^{\mathrm{SAT}}_r(F_1, \ldots, F_r)$ to be the index of the rightmost satisfiable formula. That is,

$$\#^{\mathrm{SAT}}_r(F_1, \ldots, F_r) = |\{F_1, \ldots, F_r\} \cap \mathrm{SAT}|,$$

$$\mathrm{RM}^{\mathrm{SAT}}_r(F_1, \ldots, F_r) = \max_{1 \leq i \leq r} \{i \mid F_i \in \mathrm{SAT}\}.$$

If none of $F_1, \ldots, F_r$ are satisfiable, then we let $\mathrm{RM}^{\mathrm{SAT}}_r(F_1, \ldots, F_r) = 0$.

In this paper, we will also need to consider sets of sequences of Boolean formulas with growing "arity". Without loss of generality, we can assume that all of the formulas in a sequence have the same number of clauses n and we will let the number of formulas $r(n)$

vary as a function of n. This notational device makes it easier to discuss the complexity of functions like $\#^{\mathrm{SAT}}_{\log n}$. In this case, the function expects to see inputs of the form $F_1, \ldots, F_r$ where $r \leq \log |F_i|$. Strictly speaking, the size of $F_1, \ldots, F_{r(n)}$ is $n \cdot r(n)$. However, the query complexity of $\#^{\mathrm{SAT}}_{r(n)}$ and $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ depends on $r(n)$, the arity of the tuple, rather than the length of the tuple. Thus, we count the queries as a function of n or $r(n)$ rather than the length of the encoding of $F_1, \ldots, F_{r(n)}$. For running times, however, are still expressed in terms of the length of the input. We need this convention to state the upper and lower bounds on the number of queries needed to compute $\mathrm{RM}^{\mathrm{SAT}}_{r(t)}$.

For polynomial time computable $r(n)$, both $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ and $\#^{\mathrm{SAT}}_{r(n)}$ can be computed using $\lceil \log(r(n)+1) \rceil$ queries to SAT by binary search. In previous work, Chang, Gasarch and Lund [CGL94] showed a relative lower bound on the number of queries needed to solve a certain "promise problem" using standard tree pruning techniques [ABG90, HN93]. This result provides the following corollary on the number of queries needed to compute $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$.

Corollary 6 Let $r(n)$ be a logarithmically bounded polynomial time computable function. If there exists an oracle X such that some polynomial time function computes $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ using fewer than $\lceil \log(r(n)+1) \rceil$ queries to X, then P = NP.

In this paper, we need lower bounds on $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ for cases where $r(n) \in n^{O(1)}$. We could obtain some lower bounds by translating the terseness results by Amir, Beigel and Gasarch [ABG90] and Beigel [Bei88]. However, the lower bounds would not be optimal, because there would be some difficulty translating the terminology between the papers regarding the "size" of the input.[2] Thus, we prove the following lemma using the tree-pruning techniques from Amir, Beigel and Gasarch.

Theorem 7 Let $r(n) \in n^{O(1)}$ be a polynomial time computable function. If there exists an oracle X such that some polynomial time function computes $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ using fewer than $\lceil \log(r(n)+1) \rceil$ queries to X, then P = NP.

Proof: This is a straightforward tree-pruning argument. Let M be a polynomial time Turing machine that computes $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ using fewer than $\lceil \log(r(n)+1) \rceil$ queries to some oracle X. Given an input sequence $F_1, \ldots, F_{r(n)}$ to M, we can search the entire oracle query tree in polynomial time and enumerate a list of outputs from M for each sequence of oracle replies. This list will have fewer than $r(n)+1$ values, one of which is $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$. Since $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$ can range in value from 0 to $r(n)$, one of the possible values, call it z, is not in the list. We know that $z \neq \mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$ and we found z in deterministic polynomial time without using any oracle queries.

Now, given a Boolean formula F with n clauses, we will prune the disjunctive self-reduction tree of F to try to find a satisfying assignment for F. We expand the root of

[2] Previous versions of this paper [Cha94a, Cha95] stated that Corollary 6 can be used to obtain terseness results. While this is true, the statement of Corollary 9 in those versions is incorrect. In particular, the terseness results we could obtain would not supersede the theorems shown by Beigel [Bei88], Amir-Beigel-Gasarch[ABG90] or Krentel [Kre88]. The difficulty lies in the fact that in this paper we do not count queries based on the length of the input. Even though we can show that $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ can solve $\mathrm{RM}^{\mathrm{SAT}}_{2^{q(n)}-1}$ but $\mathrm{PF}^{\mathrm{SAT}[q(n)-1]}$ cannot (unless P = NP), the n here is not the length of the input.

the tree until we have at least $r(n)$ nodes in the deepest level. Let $F_1, \ldots, F_{r(n)}$ be the first $r(n)$ nodes in the deepest level. Use the algorithm outlined above to find an index z such that $z \neq \mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$. If $z = 0$, then we know that at least one of $F_1, \ldots, F_{r(n)}$ is satisfiable which implies that $F \in \mathrm{SAT}$. So, we can accept F outright. If $z \neq 0$, then we remove the node F_z from the tree. We can safely remove F_z because we know that F_z is *not* the rightmost satisfiable formula. So, if $F_1, \ldots, F_{r(n)}$ contains satisfiable formulas, then at least one remains after removing F_z. We repeat this process until there are only $r(n) - 1$ nodes in the deepest level. Then, we expand the tree and repeat the pruning process again. When the leaves of tree are reached, there are only polynomially many leaves left in the tree. We can check each one and accept if and only if a satisfying assignment is found. Thus, $\mathrm{SAT} \in \mathrm{P}$ and $\mathrm{P} = \mathrm{NP}$. □

In many parts of the paper, using *nested sequences* of formulas will greatly simplify our calculations. The corollary below adapts Theorem 7 for nested sequences.

Definition 8 We say that the sequence $F_1, \ldots, F_r$ of Boolean formulas is a *nested* sequence if for all i, $1 \leq i < r$, $F_{i+1} \in \mathrm{SAT} \Longrightarrow F_i \in \mathrm{SAT}$. For a nested sequence $F_1, \ldots, F_r$,

$$\#^{\mathrm{SAT}}_r(F_1, \ldots, F_r) = \mathrm{RM}^{\mathrm{SAT}}_r(F_1, \ldots, F_r).$$

Corollary 9 Let $r(n) \in n^{O(1)}$ be a polynomial time computable function and let X be any oracle. If there exists a polynomial time Turing machine which can determine the value of $\#^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$ using fewer than $\lceil \log(r(n) + 1) \rceil$ queries to X for every nested sequence $F_1, \ldots, F_{r(n)}$ then $\mathrm{P} = \mathrm{NP}$.

Proof: We will show that under the hypothesis we can also compute $\mathrm{RM}^{\mathrm{SAT}}_{r'(n)}$ using fewer than $\lceil \log(r'(n) + 1) \rceil$ queries to X for some $r'(n) \in n^{O(1)}$. Hence, $\mathrm{P} = \mathrm{NP}$ by Theorem 7. Now, let $F'_1, \ldots, F'_{r'(m)}$ be a sequence of Boolean formulas, not necessarily nested, such that $|F'_i| = m$. For each F'_i, we construct a formula F_i with n clauses such that

$$F_i \in \mathrm{SAT} \iff \exists j \leq i,\ F'_j \in \mathrm{SAT}.$$

We can simply let F_i be the disjunction of $F'_1, \ldots, F'_i$. (If it is necessary for F_i to be a formula in conjunctive normal form, we can resort to Karp's reduction.) By padding, we can assume without loss of generality that every F_i has exactly $n = p(m)$ clauses for some polynomially bounded $p()$. By defining $r'(m) = r(p(m)) = r(n)$, it is syntactically correct to say that

$$\mathrm{RM}^{\mathrm{SAT}}_{r'(m)}(F'_1, \ldots, F'_{r'(m)}) = \#^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)}).$$

(Recall that the domain of the function $\mathrm{RM}^{\mathrm{SAT}}_{r'(m)}$ is the set of sequences $F'_1, \ldots, F'_\ell$ such that $\ell \leq r'(|F'_i|)$.) Then, by hypothesis, $\#^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$ can be computed using fewer than $\lceil \log(r(n) + 1) \rceil$ queries to X. In terms of m, the same machine can be used to compute $\mathrm{RM}^{\mathrm{SAT}}_{r'(m)}(F'_1, \ldots, F'_{r'(m)})$ using fewer than $\lceil \log(r'(m) + 1) \rceil$ to X. By Theorem 7, this implies that $\mathrm{P} = \mathrm{NP}$. □

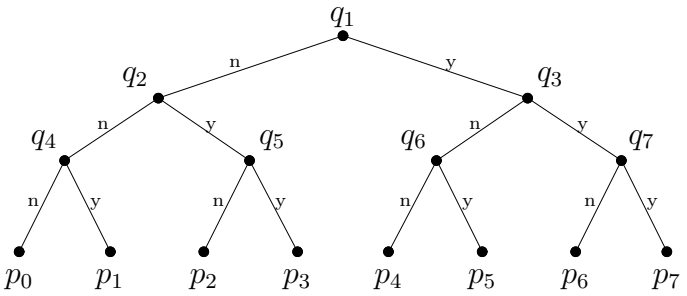

Figure 1: A query tree for 3 queries.

3 Hardness and Completeness

We start this section by showing that every function in $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ can be $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reduced to $\mathrm{RM}^{\mathrm{SAT}}_{2^{q(n)}-1}$, when $q(n)$ is bounded above by $O(\log n)$. Using this reduction, we show that any function which $k(\cdot)$-approximates Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-hard for a corresponding bounded query class. Then, combined with the upper bounds on the complexity of approximating Clique Size, we translate the hardness results into completeness results for certain approximation factors.

Theorem 10 Let $q(n) \in O(\log n)$ be a nondecreasing polynomial time computable function and let $r(n) = 2^{q(n)} - 1$. The function $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ is complete for $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ under metric reductions. In particular, there exist polynomial time computable functions T_1 and T_2 such that for all functions $f \in \mathrm{PF}^{\mathrm{SAT}[q(n)]}$ and for all x,

$$T_1(x) = \langle F_1, \ldots, F_{r(|x|)} \rangle$$
$$f(x) = T_2(x, \mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(|x|)})).$$

Proof: First, we need to show that $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ can be solved by a function in $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$. Without loss of generality, we can assume that $q(n)$ is a whole number since a $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ machine will make at most $\lfloor q(n) \rfloor$ queries. Using binary search and the SAT oracle, we can determine the rightmost formula in a sequence $F_1, \ldots, F_{r(n)}$ that is satisfiable. Note that the answer will range from 0 to $r(n)$, so there are $r(n)+1$ possible answers. Thus, binary search would use $\log(r(n)+1) = q(n)$ queries to SAT.

To show that every function in $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ can be reduced to $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$, look at the query tree with $q(n)$ serial queries. Each node of this tree represents a query to SAT. The left subtree of a node represents the computation of the $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ machine if the answer to the query is "no". The right subtree corresponds to a "yes" answer to the query. Each path in the query tree (from the root to a leaf) represents a full computation of the machine asking $q(n)$ queries to *some* oracle. The goal of the reduction is to determine which path is taken when the oracle is SAT. We call this path the *true path*.

Given any path in the query tree, we say that a query on the path is *positive* if the path goes through the right subtree of the query node. Otherwise, the query is *negative*. For

example in Figure 1, on path p_5, queries q_1 and q_6 are positive queries and q_3 is a negative query. Note that being positive or negative is relative to the path. Now, for each of the $r(n)$ paths $p_1, \ldots, p_r$, we construct a Boolean formula. For path p_i, the corresponding formula F_i is the conjunction of all the positive queries on the path. In our example, $F_5 = q_1 \wedge q_6$ and $F_3 = q_2 \wedge q_5$. We ignore the path p_0 because there are no positive queries on this path.

Let p_z be the true path in the query tree. We make two observations. First, $F_z \in \mathrm{SAT}$, since all the positive queries in path p_z are satisfiable formulas. Second, for all $i > z$, $F_i \notin \mathrm{SAT}$. To see this, note that $i > z$ implies that the path p_i is to the right of the path p_z. Hence some positive query q_t on path p_i must be a negative query on path p_z. Since p_z is the true path, q_t must be unsatisfiable. Hence, F_i is unsatisfiable. On the other hand, if $i < z$, then F_i may be satisfiable or unsatisfiable. For example, if p_5 is the true path in Figure 1, then F_2 is satisfiable iff q_2 is satisfiable, which is independent of the queries on the true path. The result is that we have produced a sequence $F_1, \ldots, F_{r(n)}$ such that $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$ is exactly the index of the true path.

We may assume that all of the F_i have the same number of clauses m. Since $m \geq n$ without loss of generality, using $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$ is an abuse of notation (it should be $\mathrm{RM}^{\mathrm{SAT}}_{r(m)}(F_1, \ldots, F_{r(m)})$. However, $r(\cdot)$ is a non-decreasing function, so $r(n) \leq r(m)$. Thus, $F_1, \ldots, F_{r(n)}$ is a properly formatted input to the function $\mathrm{RM}^{\mathrm{SAT}}_{r(m)}$.

Finally, let f be a function in $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$. We define a metric reduction to reduce the function f and an input string x to the function $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ as follows. We let T_1 be the function which prints out the sequence $F_1, \ldots, F_{r(n)}$ as described above. Now, if we are given $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$, then we have the index of the true path p_z. Thus, we can compute the value $f(x)$, by simulating the computation of $f(x)$ along the true path. This simulation does not require any oracle queries. □

The metric reduction in Theorem 10 is query preserving in the following sense. The transducer T_1 stretches the size of the input x; however, $\mathrm{RM}^{\mathrm{SAT}}_{2^{q(n)}-1}(F_1, \ldots, F_{2^{q(n)}-1})$ can always be solved with $q(n)$ serial queries to SAT. So, even when f is a function that *requires* $q(n)$ serial queries, the output of T_1 can be solved with $q(n)$ serial queries.

As in Corollary 9, since we can convert a sequence $F_1, \ldots, F_{r(n)}$ into a nested sequence $F'_1, \ldots, F'_{r(n)}$ where

$$\mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)}) = \#^{\mathrm{SAT}}_{r(n)}(F'_1, \ldots, F'_{r(n)}),$$

Theorem 10 also shows that $\#^{\mathrm{SAT}}_{r(n)}$ is complete.

Corollary 11 Let $q(n) \in O(\log n)$ be nondecreasing and polynomial time computable. Then, $\#^{\mathrm{SAT}}_{r(n)}$ is complete for $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ where $r(n) = 2^{q(n)} - 1$.

The following corollary is a special case of Corollary 11 when $r(n)$ is a constant k. This corollary also shows an interesting connection between bounded query function classes and bounded query languages classes, because the canonical $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete language for the language class $\mathrm{P}^{\mathrm{SAT}[k]}$ is the language $\mathrm{ODD}^{\mathrm{SAT}}_{2^k-1} \oplus \mathrm{EVEN}^{\mathrm{SAT}}_{2^k-1}$ [Bei91, CGH+88]. Here, $\mathrm{ODD}^{\mathrm{SAT}}_r$ (and respectively $\mathrm{EVEN}^{\mathrm{SAT}}_r$) is the set of nested sequences $F_1, \ldots, F_r$ such that $\#^{\mathrm{SAT}}_r(F_1, \ldots, F_r)$ is odd (even).

Corollary 12 For all constant k, the function $\#^{\mathrm{SAT}}_{2^k-1}$ is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[k]}$.

We are now ready to show the completeness of approximating Clique Size. The following upper bound is from Chang, Gasarch and Lund [CG93, CGL94]:

Lemma 13 Let Π be an NP-optimization problem and let $\mathrm{OPT}_\Pi(x)$ be the cost of the optimum solution for an instance x. Suppose that the best polynomial time approximation algorithm can approximate OPT_Π within a factor of $A(n)$. Then, for $k(n) \leq A(n)$, there exists a polynomial time algorithm which $k(\cdot)$-approximates OPT_Π using $\lceil \log \lceil \log_{k(n)} A(n) \rceil \rceil$ queries to SAT.

Proof: First, assume that Π is a maximization problem. Given an instance x of Π, we first use the polynomial time algorithm to approximate $\mathrm{OPT}_\Pi(x)$ within a factor of $A(n)$, where $n = |x|$. This algorithm will produce a number z and we know that $z \leq \mathrm{OPT}_\Pi(x) < A(n) \cdot z$. Now, we divide the number line between z and $A(n) \cdot z$ into intervals of the form $[zk(n)^i, zk(n)^{i+1})$. There are at most $\log_{k(n)} A(n)$ such intervals. Since Π is an NP-optimization problem, a query to SAT can determine whether $\mathrm{OPT}_\Pi(x) \geq y$ for some number y. Thus, using binary search $\lceil \log \lceil \log_{k(n)} A(n) \rceil \rceil$ queries is sufficient to find an interval that contains $\mathrm{OPT}_\Pi(x)$. We can then report the lower endpoint of the interval as a $k(n)$-approximation of $\mathrm{OPT}_\Pi(x)$. Minimization problems are handled analogously. □

For Clique Size, we know that the $\omega(G)$ ranges from 2 to n for a graph of size n (by first checking if the graph has any edges). Thus, we have the following corollary.

Corollary 14 Let $k(n)$ be a polynomial time computable function such that $1 < k(n) < n$. Then there exists a function in $\mathrm{PF}^{\mathrm{SAT}[\lceil \log \lceil \log_{k(n)} n \rceil \rceil]}$ which $k(\cdot)$-approximates Clique Size.

The upper bound on the complexity of approximating Clique Size is achieved by binary search. The difficult part is to show that every function computable using a constant fewer queries $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reduces to $k(\cdot)$-approximating Clique Size. To do so, we need to review some consequences of the existence of probabilistically checkable proofs for NP.

Lemma 15 [ALM+92] There exist constants s, b and d, $1 < s < b < d$, such that given a Boolean formula F with t clauses, we can construct in polynomial time a graph G with t^d vertices, where

$$F \in \mathrm{SAT} \Longrightarrow \omega(G) = t^b$$
$$F \notin \mathrm{SAT} \Longrightarrow \omega(G) = t^s.$$

The lemma as cited only shows that $\omega(G) < t^s$ in the case that $F \notin \mathrm{SAT}$. A simple padding argument will give equality in this case. We make this modification because it will greatly simplify our calculations.

Lemma 15 can be rephrased as a statement on the approximability of $\omega(G)$. Let $\epsilon = (b - s)/d$. Then the ratio of the size of the big clique, t^b, to the size of the small clique, t^s, is $(t^d)^\epsilon$. Since $|G| = t^d$, Lemma 15 shows that an n^ϵ-approximation of $\omega(G)$ will determine whether the original formula F is satisfiable. Throughout the rest of this paper, we will fix the values of ϵ, b, s and d to be these constants.

We would like to state our theorem about the hardness of approximating $\omega(G)$ in the most general terms possible. Since the number of queries required to solve the approximation

problem increases as the approximation factor becomes finer (smaller), the statement of our results depend on the approximation factor. In general, this factor does not have to be a constant or even an increasing function. Our theorem is restricted to the classes $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ where $q(n)$ is non-decreasing.

Theorem 16 (Main Theorem) Let $c = 1 + \log(1 + 1/\epsilon)$ and let $q(n) \in O(\log n)$ be a non-decreasing polynomial time computable function. Define the function $k(n)$ such that $q(n) = \log\log_{k(n)} n - c$ (i.e., $k(n) = n^{2^{-q(n)-c}}$). If h is a function which $k(\cdot)$-approximates Clique Size and $f \in \mathrm{PF}^{\mathrm{SAT}[q(n)]}$, then $f \leq^{\mathrm{P}}_{\mathrm{mt}} h$.

The proof of our main theorem modifies the construction of Chang, Gasarch and Lund [CGL94]. However, in this proof, we are able to remove several restrictions on the approximation factor $k(n)$. In particular, we can handle cases where $k(n)$ is a fractional value between 1 and 2. Also, the calculations are simplified in this proof. Our main construction is encapsulated in the following lemma which will also be used to prove Theorem 18. This lemma constructs a graph H with m vertices such that a $k(m)$-approximation of $\omega(G)$ will tell us the number of satisfiable formulas in a nested sequence $F_1, \ldots, F_{r(t)}$. In what follows, the constants b, s, d and $\epsilon = (b - s)/d$ are from Lemma 15.

Lemma 17 (Construction Lemma) Let $k(\cdot)$ be a polynomial time computable function such that $k(n) \geq 1 + \sqrt{2}/n^{\delta}$ for $\delta = \epsilon/(4 + 4\epsilon)$. Let $F_1, \ldots, F_{r(t)}$ be a nested sequence with $|F_i| = t$, $m = t^{b-s+d}$ and

$$r(t) \leq \left\lfloor \log_{k(m)} t^{(b-s)/2} \right\rfloor.$$

Then, in polynomial time we can construct a graph H with m vertices and whole numbers $y_0 < \cdots < y_{r(t)}$ such that $k(m)y_i < y_{i+1}$ for $0 \leq i \leq r(t) - 1$, and

$$\omega(H) = y_z \iff \#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)}) = z.$$

Proof: We will first construct a graph H' with fewer than m vertices, then the graph H is produced from H' by simply adding $m - |H'|$ unconnected vertices. To construct H', we take each F_i and produce a graph G_i with t^d vertices according to Lemma 15. We want to define two operations on graphs: addition $\oplus$ and scalar multiplication $\otimes$. These operations commute with $\omega()$:

$$\omega(G_1 \oplus G_2) = \omega(G_1) + \omega(G_2)$$
$$\omega(a \otimes G) = a \cdot \omega(G).$$

The vertices of the graph $G' = G_1 \oplus G_2$ is the disjoint union of the vertices of G_1 and G_2. The edges in G' are all the edges in G_1 and G_2 plus the edges (u, v) for each $u \in G_1$ and $v \in G_2$. Then, scalar multiplication for a whole number a is simply repeated addition.

To simplify our notation, let $g = k(m)$ and $r = r(t)$. Note that g does not have to be a whole number. We define a sequence of whole numbers $a_1, \ldots, a_r$ inductively:

$$a_i = \begin{cases} 1 & \text{if } i = 1 \\ \lceil g a_{i-1} \rceil & \text{for } 2 \leq i \leq r \end{cases}$$

We define the graph H' to be $(a_1 \otimes G_1) \oplus (a_2 \otimes G_2) \oplus \cdots \oplus (a_r \otimes G_r)$. Clearly, H' has $\sum_{i=1}^{r} a_i t^d$ vertices. To estimate the value of $\sum_{i=1}^{r} a_i$, note that for $i \geq 2$, we have $a_i < g \cdot a_{i-1} + 1$, so

$$a_i \leq g^{i-1} + g^{i-2} + \cdots + 1 = \frac{g^i - 1}{g - 1}. \tag{1}$$

Thus, we have the estimate

$$\sum_{i=1}^{r} a_i \;\leq\; \sum_{i=1}^{r} \frac{g^i - 1}{g-1} \;<\; \frac{g}{g-1} \sum_{i=0}^{r-1} g^i \;=\; \frac{g(g^r - 1)}{(g-1)^2}. \tag{2}$$

By the restriction on r, we know that

$$\frac{g(g^r - 1)}{(g-1)^2} < \frac{g}{(g-1)^2} \cdot t^{(b-s)/2}.$$

If $g \geq 2$, we know that $g/(g-1)^2 \leq 2$. If $g \leq 2$, then from the lower bound on $k(m)$ and the fact that $t^{b-s} = m^{\epsilon/(1+\epsilon)}$ we can deduce that

$$\frac{g}{(g-1)^2} \leq t^{(b-s)/2}.$$

In either case, H' has at most $t^d t^{b-s}$ vertices. Since $m = t^{b-s+d}$, we also have $|H'| < m$. Then, we can construct H by simply adding dummy vertices to H'. The result is that $|H| = m$ and $\omega(H) = \sum_{i=1}^{r} a_i \omega(G_i)$. Also, when $z = \#_r^{\mathrm{SAT}}(F_1, \ldots, F_r)$, by Lemma 15 we have

$$\omega(H) = \sum_{j=1}^{z} a_j t^b + \sum_{j=z+1}^{r} a_j t^s.$$

So, we can define the numbers $y_0, \ldots, y_r$ as:

$$y_i = \sum_{j=1}^{i} a_j t^b + \sum_{j=i+1}^{r} a_j t^s.$$

It is easy to see that $\omega(H) \in \{y_0, \ldots, y_r\}$ and that $z = \#_r^{\mathrm{SAT}}(F_1, \ldots, F_r) \Longrightarrow \omega(H) = y_z$. We still need to show that $y_{i+1} > g y_i$. From the definition of y_i and the fact that $g a_i \leq a_{i+1}$, it follows that

$$y_{i+1} - g y_i \geq t^b - g a_r t^s.$$

By Equation 1, it suffices to show that

$$\frac{g(g^r - 1)}{g - 1} < t^{b-s}. \tag{3}$$

From the restriction on r, we know that

$$\frac{g(g^r - 1)}{g - 1} < \frac{g}{g-1} t^{(b-s)/2}.$$

As before, when $g \geq 2$, we have $g/(g-1) \leq 2$. When $g \leq 2$, the lower bound on $k(m)$ provides us with the bound:

$$\frac{g}{g-1} \leq \sqrt{2} \cdot t^{(b-s)/4}.$$

In either case, we have satisfied the inequality in Equation 3 for large enough t. Since, $g = k(m)$, we can finally conclude that $k(m) \cdot y_i < y_{i+1}$. Thus, $y_0 < y_1 < \cdots < y_r$ and

$$\omega(H) = y_z \iff \#_r^{\mathrm{SAT}}(F_1, \ldots, F_r) = z.$$

□

Proof of Main Theorem: Let $k(n)$ be as defined in the hypothesis. First, we will assume that $k(n) \geq 1 + \sqrt{2}/n^{\delta}$ for some $\delta < \epsilon/(4+4\epsilon)$. Let f be any function in $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$. Since $q(n) \in O(\log n)$, we can use Theorem 10 to reduce f to $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}$ via T_1 and T_2 where $r(n) = 2^{\lfloor q(n) \rfloor} - 1$. Thus, given an input x to the function f, with $|x| = n$, we can produce $F'_1, \ldots, F'_{r(n)}$ such that

$$f(x) = T_2(x, \mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F'_1, \ldots, F'_{r(n)})).$$

In addition, we transform $F'_1, \ldots, F'_{r(n)}$ into a nested sequence $F_1, \ldots, F_{r(n)}$ as described in Corollary 9. Without loss of generality, assume that every F_i has the same length t which is greater than n. The function $k(n)$ was defined so that $\log \log_{k(n)} n - c = q(n)$. Using the definition of $r(n)$, we can conclude that

$$\log(r(n)+1) = \lfloor q(n) \rfloor \leq q(n) = \log \log_{k(n)} n - c.$$

Thus,

$$r(n) + 1 < 2^{-c} \cdot \log_{k(n)} n.$$

Now, $q(n)$ is nondecreasing and c is a constant, so the function $\log_{k(n)} n$ must also be nondecreasing. Let $m = t^{b-s+d}$. Then, $n < t < m$, so

$$r(n) + 1 < 2^{-c} \cdot \log_{k(n)} n \leq 2^{-c} \cdot \log_{k(m)} m.$$

From the definition of c, we obtain:

$$2^{-c} = \frac{\epsilon}{2(1+\epsilon)} = \frac{b-s}{2(b-s+d)}.$$

Thus, we can rewrite $2^{-c} \log_{k(m)} m$ as

$$2^{-c} \log_{k(m)} m = \frac{b-s}{2(b-s+d)} \log_{k(m)} t^{b-s+d} = \log_{k(m)} t^{(b-s)/2}.$$

This provides the desired restriction on r, since

$$r(n) < \log_{k(m)} t^{(b-s)/2} - 1 < \left\lfloor \log_{k(m)} t^{(b-s)/2} \right\rfloor$$

which satisfies the hypothesis of the Construction Lemma. Thus, we obtain a graph H and numbers $y_1, \ldots, y_{r(n)}$ such that

$$\omega(H) = y_z \iff \#^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)}) = z.$$

Since $F_1, \ldots, F_{r(n)}$ is nested, we also know that $\mathrm{RM}^{\mathrm{SAT}}_{r(n)}(F'_1, \ldots, F'_{r(n)}) = \#^{\mathrm{SAT}}_{r(n)}(F_1, \ldots, F_{r(n)})$. The Construction Lemma also guarantees that the y_i's are separated by a factor of $k(m)$, where $m = |H|$. So, if a function h $k(\cdot)$-approximates $\omega(\cdot)$, then given $y = h(H)$, we can find the unique z such that $y_z \leq y < k(m) \cdot y_z$. Then, $f(x) = T_2(x, z)$ and we have reduced f to h by a metric reduction.

Now, if $k(n) < 1 + \sqrt{2}/n^\delta$, then we cannot use the Construction Lemma directly. Instead, let $k'(n) = 1 + \sqrt{2}/n^\delta$ and $q'(n) = \log\log_{k(n)} n - c$. Then, we can reduce any function f' in $\mathrm{PF}^{\mathrm{SAT}[q'(n)]}$ to $k'(\cdot)$-approximating $\omega(\cdot)$. Since $k(n) < k'(n)$, f' also reduces to $k(\cdot)$-approximating $\omega(\cdot)$. Now, $q'(n)$ is $\Theta(\log n)$, so every function in $\mathrm{PF}^{\mathrm{SAT}[O(\log n)]}$ reduces to some $f' \in \mathrm{PF}^{\mathrm{SAT}[q'(n)]}$ by a simple padding argument. Thus, every function in $\mathrm{PF}^{\mathrm{SAT}[q(n)]}$ reduces to $k(\cdot)$-approximating $\omega(\cdot)$.

To see that $q'(n)$ is indeed $\Theta(\log n)$, note that the Taylor series for $\ln(1 + 1/N)$ allows us to approximate $\ln(1 + 1/N)$ as $1/N$ for large values of N. Thus,

$$q'(n) = \log\ln n - \log\ln(1 + 1/n^\delta) - c \approx \log\ln n + \delta \log n - c.$$

For large n, the lower order terms of the Taylor series would not contribute significantly, so the $\delta \log n$ term dominates the value of $q'(n)$. So, $q'(n)$ is indeed $\Theta(\log n)$. □

Remark: If we are willing to settle for a randomized reduction with exponentially small error, then combining the techniques of Bellare, Goldwasser, Lund and Russell [BGLR93] with those of Zuckerman [Zuc93] as described by Chang, Gasarch and Lund [CGL94, Section 6], we have $\epsilon = 1/31$ and $c = 1 + \log(1 + 31) = 6$.

Using the same Construction Lemma, we can obtain a relative lower bound on the number of queries needed to compute a $k(n)$-approximation of $\omega(G)$. The improvement over the results in Chang, Gasarch and Lund [CGL94] is twofold. First, we no longer need to make any "niceness" assumptions on $k(n)$. Also, the proof works for $k(n) < 2$.

Theorem 18 Let $k(n)$ be polynomial time computable such that $1 + \sqrt{2}/n^\delta \leq k(n) \leq n$, for some $\delta < \epsilon/(4 + 4\epsilon)$. If there exists a polynomial time computable function which $k(\cdot)$-approximates $\omega(G)$ using $\log\log_{k(n)} n - c$ queries to any oracle X for all graphs G with n vertices and $c = 1 + \log(1 + 1/\epsilon)$, then P = NP.

Proof: Suppose there exists a function h which $k(n)$-approximates $\omega(G)$ using no more than $\log\log_{k(n)} n - c$ queries to X. We will show that for some $r(t) \in n^{O(1)}$, $\#^{\mathrm{SAT}}_{r(t)}$ for nested sequences can be computed using fewer than $\lceil \log(r(t) + 1) \rceil$ queries to X. This in turn implies that P = NP by Corollary 9.

As required by the Construction Lemma, let $m = t^{b-s+d}$ and

$$r(t) = \left\lfloor \log_{k(m)} t^{(b-s)/2} \right\rfloor.$$

Let $F_1, \ldots, F_{r(t)}$ be a nested sequence, where $|F_i| = t$. We build the graph H with $|H| = m$ according to the Construction Lemma. We then compute a number y which is a $k(m)$-approximation of $\omega(H)$ using the function h and $\log\log_{k(m)} m - c$ queries to X. As before,

the unique z such that $y_z \leq y < k(m) \cdot y_z$ allows us to compute $\#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$. As in the proof of Theorem 16,

$$2^{-c} \log_{k(m)} m = \log_{k(m)} t^{(b-s)/2}.$$

So, $\log \log_{k(m)} m - c < \log(r(t)+1) \leq \lceil \log(r(t)+1) \rceil$. Since $k(m) \geq 1 + \sqrt{2}/n^{\delta}$, the function $r(t)$ is polynomially bounded. Thus, the number of queries we used to compute $\#^{\mathrm{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)})$ violates the relative lower bound of Corollary 9 and P = NP. □

Remark: Slightly better bounds can be achieved for the Main Theorem and Theorem 18 in certain cases. For example, when $k(n) \geq 1 + 1/n^{o(1)}$ (e.g., $k(n) = 1 + 1/\log n$ and $k(n) = 2$), these theorems hold for all $c > \log(1 + 1/\epsilon)$. This is accomplished by restricting $r(t)$ in the Construction Lemma by

$$r(t) \leq \left\lfloor \log_{k(m)} \frac{t^{b-s}(k(m)-1)^2}{k(m)^2} \right\rfloor$$

and showing that for large enough t,

$$2^{-c} \cdot \log_{k(m)} m < \log_{k(m)} \frac{t^{b-s}(k(m)-1)^2}{k(m)^2}.$$

The calculations are somewhat involved and omitted in this paper.

For several approximation factors, we can restate the hardness results as completeness results. Note that the bounded query classes in the following corollary are closed under metric reductions and are different unless P = NP.

Corollary 19 For all constants $k \geq 1$ and all constants a with $0 < a \leq 1$:

- $(1 + n^{-a})$-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[O(\log n)]}$.[3]
- $(1 + 1/\log^k n)$-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[(k+1)\log\log n + O(1)]}$.
- k-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[\log\log n + O(1)]}$.
- $(\log^k n)$-approximating Clique Size is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[\log\log n - \log\log\log n + O(1)]}$.

Proof: Consider the case of a constant approximation factor k. Corollary 14 shows that we can k-approximate Clique Size using $\log\log_k n + O(1)$ queries to SAT.

Let h be a function which k-approximates Clique Size and take any f in $\mathrm{PF}^{\mathrm{SAT}[\log\log n + q]}$ for some constant q. We want to show that $f \leq^{\mathrm{P}}_{\mathrm{mt}} h$, but Theorem 16 only shows that every function in $\mathrm{PF}^{\mathrm{SAT}[\log\log_k n - c]}$ $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reduces to h. To overcome the deficit of $c + q$ queries, we use a simple padding trick and define a new function f' to be:

$$f'(x \# 1^{|x|^{c+q}}) = f(x).$$

Now, f' is a function in $\mathrm{PF}^{\mathrm{SAT}[\log\log n - c]}$, so $f' \leq^{\mathrm{P}}_{\mathrm{mt}} h$. Since $f \leq^{\mathrm{P}}_{\mathrm{mt}} f'$, it follows that $f \leq^{\mathrm{P}}_{\mathrm{mt}} h$. Thus, approximating Clique Size within a constant factor is complete under metric reductions for $\mathrm{PF}^{\mathrm{SAT}[\log\log n + O(1)]}$.

[3] This statement can also be derived directly from the self-improvability of the Clique problem.

The other cases are proven using a similar padding argument and the Taylor series approximation $\ln(1 + 1/N) \approx 1/N$ for large N. □

The results in this section show that bounded queries and approximating Clique Size are inseparable. That is, you can obtain good approximations of Clique Size if and only if you can answer oracle queries to SAT. Moreover, if you want a closer approximation of Clique Size, then you need more queries to the oracle. For example, we have shown that finding $\log n$-approximations of Clique Size is equivalent to answering $\log\log n - \log\log\log n + O(1)$ queries to SAT and that finding constant factor approximations of Clique Size is equivalent to answering $\log\log n + O(1)$ queries to SAT.

Completeness, as opposed to lower bounds, is important here because completeness implies that any complexity class that is closed under metric reductions and contains these two problems must also contain the two bounded query classes. Thus, any measure of complexity which can distinguish between constant factor approximations and $\log n$ factor approximations must be fine enough to distinguish the between the number of oracle queries. We view this as evidence that using bounded queries is a *natural* complexity measure for approximation problems.

These results can also be extended to some other NP-approximation problems, most notably to Chromatic Number. The hardness results follow from the work of Lund and Yannakakis [LY94] which provides a reduction from Clique to Coloring. Thus, for example, we can claim that 2-approximating Chromatic Number is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[\log\log n + O(1)]}$ under metric reductions. In the next section, we show that the hardness of these approximation problems provides another benefit. We can now reduce many NP-approximation problems to approximating Clique Size in an approximation preserving manner.

4 Reductions to Clique Size

The results of the preceding section showed that every function $f \in \mathrm{PF}^{\mathrm{SAT}[O(\log n)]}$ $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reduces to any function h that $k(n)$-approximates Clique Size, for some approximation factor $k(n)$. The more queries it takes to compute f, the smaller the approximation factor $k(n)$. However, when f itself is also solving an NP-approximation problem, this reduction takes on some approximation preserving characteristics. We demonstrate these characteristics by way of an example.

Suppose that we are given a graph G with n vertices and we want to 2-approximate $\chi(G)$, the chromatic number of G. This approximation can be found with $\lceil \log \lceil \log n \rceil \rceil$ queries to SAT using binary search. By Corollary 19, 2-approximating Chromatic Number $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reduces to 2-approximating Clique Size, but this reduction has some additional properties.

Let us examine this reduction more closely. First, the binary search routine which approximates $\chi(G)$ does so by asking questions of the form:

Is $\chi(G) \leq n/2^i$?

There are $\lceil \log n) \rceil$ questions of this form. Turn each question into a Boolean formula, F_i, so that $F_i \in \mathrm{SAT}$ if and only if $\chi(G) \leq n/2^i$. Since this results in a nested sequence, we can reduce it directly to 2-approximating Clique Size. To do this, we build a graph H as

described in Theorem 16 and compute $y_0, \ldots, y_r$ such that $2y_i < y_{i+1}$ and $\omega(H) = y_i \iff \#_r^{\mathrm{SAT}}(F_1, \ldots, F_r) = i$. Now, suppose we are given a number x and are told that it is a 2-approximation of $\omega(H)$. Then, let y_z be the unique y_i such that $x \leq y_z < 2x$ and we have determined that $\omega(H) = y_z$. So, $n/2^z$ is a 2-approximation of $\chi(G)$. On the other hand, if we are told that x is a 4-approximation of $\omega(H)$, then we cannot determine $\omega(H)$ exactly. Instead, we could have the situation that $x \leq y_z < y_{z+1} < 4x$. Again, we use $n/2^z$ as an approximation of $\chi(G)$, but this time it is a 4-approximation. In general we do not have to be given a guarantee that x is within a certain factor of $\omega(H)$. We just find the unique y_z such that $x \leq y_z < 2x$ and use $n/2^z$ as an approximation of $\chi(G)$. If x is a k-approximation of $\omega(H)$, then $n/2^z$ is an approximation of $\chi(G)$ within $2^{\lceil \log k \rceil} < 2k$. Thus, we have the following theorem:

Theorem 20 There exist polynomial time computable functions T_1 and T_2 such that for all graphs G and all numbers x, if x is a k-approximation of $\omega(T_1(G))$, then $T_2(G, x)$ is a $2k$-approximation of $\chi(G)$.

Of course, we can replace Chromatic Number with any NP-approximation problem where the solution is between 1 and n^a. Thus, approximating Clique Size is complete in the sense that all these NP-approximation problems reduce to approximating Clique Size in an approximation preserving manner.

5 The Complexity of MAX3SAT

In this section, we examine the complexity of the NP-approximation problem MAX3SAT. As we have mentioned before, the MAX3SAT problem does have polynomial time algorithms which achieve a constant factor approximation. For example, Yannakakis [Yan92] presents a 4/3 factor approximation. A consequence of this approximation algorithm is that the number of queries needed to $k(n)$-approximate MaxSat is lower than the number of queries needed to $k(n)$-approximate Clique Size. Using Lemma 13, we have:

Corollary 21 Let $k(n)$ be a polynomial time computable function such that $1 < k(n) < n$. Then there exists a function in $\mathrm{PF}^{\mathrm{SAT}[\lceil \log \log_{k(n)}(4/3) \rceil]}$ which $k(n)$-approximates MaxSat.

To prove the lower bounds, we need a result from probabilistically checkable proofs [ALM+92] which states that there exists a constant k such that no polynomial time algorithm can k-approximate MaxSat unless P = NP. As pointed out by Khanna *et al.*, [KMSV94] this result can be interpreted as follows:

Lemma 22 [KMSV94] There exist constants δ, c_1, c_2 and a polynomial time computable function f such that $0 < \delta < 1$, $1 \leq c_1$, $1 \leq c_2$ and given a 3CNF formula F with t clauses, f produces a 3CNF formula F' with $\ell(t) = c_1 t^{c_2}$ clauses, where

$$F \in \mathrm{SAT} \Longrightarrow F' \in \mathrm{SAT} \Longrightarrow \mathrm{MaxSat}(F') = \ell(t)$$
$$F \notin \mathrm{SAT} \Longrightarrow \mathrm{MaxSat}(F') = \ell(t)/(1+\delta).$$

Proof Sketch: This proof was originally reported by Khanna, *et al.* [KMSV94], but we include a sketch of this proof for the sake of completeness. We start with the verifier for a probabilistically checkable proof for SAT. The verifier V uses $c \log n$ random bits and looks at $O(1)$ bits of the proof. If the given formula F is satisfiable then there exists a proof that causes V to accept with probability 1. If the formula is not satisfiable, then we modify the standard verifier to accept some proof with probability exactly 1/2. Note that in this case, no proof can cause the verifier to accept with probability greater than 1/2.

Now, the computation of the verifier after each random string z has been chosen depends only on the $O(1)$ bits of the proof. This computation can be described by a formula F_z using Karp's reduction. Since the size of the input to this part of the verifier's computation is constant, $|F_z|$ is also bounded by a constant — even as the length of the original formula F grows.

Let c_1 be the number of clauses in F_z. A closer examination of Karp's reduction will show that if the verifier V using the random string z rejected, then $\text{MaxSat}(F_z) = c_1 - 1$. (Essentially, the only unsatisfiable clause is the one that forces the final state to be an accepting state. This condition holds even when Karp's reduction is modified to give 3CNF formulas.) Now, let F' be the conjunction of every F_z for each random string z. Since there are t^{c_2} random strings, F' has $\ell(t) = c_1 t^{c_2}$ clauses. If F is satisfiable, then F' is satisfiable and $\text{MaxSat}(F') = \ell(t)$. On the other hand, if F is not satisfiable, then we can satisfy exactly half of the F_z's. Thus,

$$\text{MaxSat}(F') = \frac{c_1 t^{c_2}}{2} + \frac{(c_1 - 1)t^{c_2}}{2} = \ell(t) - \frac{\ell(t)}{2c_1} = \frac{\ell(t)}{1+\delta}$$

by choosing $\delta = 1/(2c_1 - 1)$. □

Using this lemma, Khanna *et al.* [KMSV94] showed that every NP-optimization problem which has a constant factor polynomial time approximation algorithm can be reduced to MAX3SAT by what they call an L-reduction with scaling. Our results show a quantitative relationship between the complexity of approximating MaxSat and bounded queries. Our results also allow a direct comparison between the complexity of approximating MaxSat and approximating Clique Size.

Theorem 23 Let $q(n) \in O(\log n)$ be a non-decreasing polynomial time computable function. Define the function $k(n)$ such that $q(n) = \log\log_{k(n)}(1 + \delta')$ for some constant δ', $0 < \delta' < \delta$. If h is a function which $k(\cdot)$-approximates MaxSat and $f \in \text{PF}^{\text{SAT}[q(n)]}$, then $f \leq^{\text{P}}_{\text{mt}} h$.

As in Theorem 16, the proof of Theorem 23 is straightforward after we prove the following Construction Lemma.

Lemma 24 (Construction Lemma for MAX3SAT)
Let $k(n)$ be a polynomial time computable function such that $1 + n^{-1/3} \leq k(n) \leq 1 + \delta'$, for some constant δ' with $0 < \delta' < \delta$. Let $F_1, \ldots, F_{r(t)}$ be a nested sequence with $|F_i| = t$, $m = \ell(t)^2$ and

$$r(t) \leq \left\lfloor \log_{k(m)}(1 + \delta') \right\rfloor .$$

Then, in polynomial time we can construct a 3CNF formula H with m clauses and whole numbers $y_0 < \cdots < y_{r(t)}$ such that $k(m)y_i < y_{i+1}$ for $0 \leq i \leq r(t) - 1$, and

$$\text{MaxSat}(H) = y_z \iff \#^{\text{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)}) = z.$$

Proof: Let F_0 be a known unsatisfiable formula with t clauses. Let $F'_0, F'_1, \ldots, F'_{r(t)}$ be formulas with $\ell(t)$ clauses constructed by applying Lemma 22 to $F_0, F_1, \ldots, F_{r(t)}$. We will use F'_0 for padding. We construct a formula H which is the conjunction of $\ell(t)$ formulas taken from $\{F'_0, \ldots, F'_{r(t)}\}$. So, we know that $|H| = \ell(t)^2 = m$.

To simplify our notation, let $\ell = \ell(t)$, $r = r(t)$ and $g = k(m)$. We now define a sequence of whole numbers $a_0, \ldots, a_r$ as follows:

$$a_i = \begin{cases} 0 & \text{if } i = 0 \\ \lceil g a_{i-1} + \ell(g-1)/\delta \rceil + 1 & \text{for } 1 \leq i \leq r \end{cases}$$

The formula H is the conjunction of $a_i - a_{i-1}$ copies of F_i, for $1 \leq i \leq r$, and $\ell - a_r$ copies of F'_0. We will show below that $\ell \geq a_r$. For now, let $\alpha = \delta/(1+\delta)$ and $\beta = 1/(1+\delta)$. That is, $\delta = \alpha/\beta$ and for each F_i,

$$F_i \in \text{SAT} \Longrightarrow \text{MaxSat}(F) = \alpha\ell + \beta\ell$$

$$F_i \notin \text{SAT} \Longrightarrow \text{MaxSat}(F) = \beta\ell.$$

When $\#^{\text{SAT}}_r(F_1, \ldots, F_r) = 0$, $\text{MaxSat}(H) = \beta\ell^2$ since H is composed of ℓ formulas from $\{F'_0, \ldots, F'_r\}$ each with ℓ clauses. In general, if $\#^{\text{SAT}}_r(F_1, \ldots, F_r) = z$, then

$$\text{MaxSat}(H) = \alpha a_z \ell + \beta\ell^2.$$

Thus, we define $y_i = \alpha a_z \ell + \beta\ell^2$. The fact that $k(m)y_i \leq y_{i+1}$ follows easily from the definition of a_i because $y_{i+1} - g y_i \geq 1$.

Proving that $a_r \leq \ell$ requires a bit more work. We use the relationship

$$a_i \leq g a_{i-1} + \frac{\ell(g-1)}{\delta} + 2$$

to show that

$$a_i \leq \left(\frac{\ell(g-1)}{\delta} + 2\right) \cdot \sum_{j=1}^{i-1} g^j = \left(\frac{\ell(g-1)}{\delta} + 2\right) \cdot \left(\frac{g^i - 1}{g-1}\right).$$

Thus, $a_r \leq \ell$ when

$$g^r \leq 1 + \delta - \frac{2\delta^2}{\ell(g-1) + 2\delta}. \tag{4}$$

Since $k(m) \geq 1 + n^{-1/3}$ and $\ell = \sqrt{m}$, it follows that $\ell(g-1) \geq m^{1/6}$. Therefore, the last term in the right hand side of Equation 4 approaches zero as t gets larger. Thus, Equation 4 is satisfied by the requirement that for some $\delta' < \delta$,

$$r(t) \leq \left\lfloor \log_{k(m)}(1 + \delta') \right\rfloor.$$

□

Corollary 25 For all constants $k \geq 1$ and all constants a with $0 < a \leq 1$:

- $(1 + n^{-a})$-approximating MaxSat is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[O(\log n)]}$.
- $(1 + 1/\log^k n)$-approximating MaxSat is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-complete for $\mathrm{PF}^{\mathrm{SAT}[k \log\log n + O(1)]}$.

As in Theorem 18, we can use the Construction Lemma for MAX3SAT to derive a lower bound on the number of queries needed to approximate MaxSat. Note that the upper bound of $\log\log_{k(n)}(4/3)$ queries and the lower bound of $\log\log_{k(n)}(1+\delta')$ queries given below differ only by a constant: $\log\log_{1+\delta'}(4/3)$. This difference represents the fact that for k between $(1+\delta)$ and $4/3$, we do not know if there exist polynomial time algorithms that k-approximate MaxSat. If such algorithms exist then we can reduce the upper bound.

Theorem 26 Let $k(n)$ be polynomial time computable such that $1 + n^{-1/3} \leq k(n) \leq 1 + \delta'$ for some constant δ' with $0 < \delta' < \delta$. If there exists a polynomial time computable function which $k(\cdot)$-approximates MaxSat using $\log\log_{k(n)}(1 + \delta')$ or fewer queries to any oracle X, then P = NP.

Proof: Analogous to the proof of Theorem 18. □

The results in this section can be extended to cover any MAXNP-complete and MAXSNP-complete problem, since every MAXNP problem has a constant factor polynomial time approximation algorithm and MAX3SAT reduces to these problems by L-reductions. In the next section, we discuss the relationship between bounded queries and self-improvability.

6 Queries and Self-improvement

In this section, we explore some connections between the structure of bounded query classes and the self-improvability of NP-approximation problems. An NP-approximation problem is self-improvable if for all constants k_1 and k_2, with $k_1 < k_2$, k_1-approximating the problem can be reduced to any function that k_2-approximates the problem. Intuitively, when a function is self-improvable, we can reduce a finer approximation problem to a coarser approximation problem for constant factor approximations.

The Clique Size problem is known to be self-improvable [GJ79]. For example, we can reduce 2-approximating Clique Size to 4-approximating Clique Size as follows. Given a graph G with n vertices, construct a graph G' with n^2 vertices such that $\omega(G') = \omega(G)^2$. Now if a number x 4-approximates $\omega(G')$, $\sqrt{x}$ would 2-approximate $\omega(G)$.

Our first observation is that the self-improving reduction above does not really reduce the number of queries needed to 2-approximate $w(G)$. This is because it takes roughly $\log\log_4(n^2)$ queries to SAT to 4-approximate $\omega(G')$. However, $\log\log_4(n^2) = \log\log n$ and using $\log\log n$ queries we can 2-approximate $\omega(G)$ directly. So, self-improving reductions are not query saving reductions.

Secondly, we point out that the self-improvement property of Clique Size can be deduced from the hardness of Clique Size shown in Theorem 16. Suppose that k_1 and k_2 are two constant approximation factors where $k_1 < k_2$. Let $\log\log n + q_1$ be the number of queries

needed to k_1-approximate Clique Size using binary search. Also, let q_2 be the constant from Theorem 16 such that every function in $\mathrm{PF}^{\mathrm{SAT}[\log\log n - q_2]}$ $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reduces to any function that k_2-approximates Clique Size. By a simple padding argument, every function in $\mathrm{PF}^{\mathrm{SAT}[\log\log n + q_1]}$ $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reduces to a function in $\mathrm{PF}^{\mathrm{SAT}[\log\log n - q_2]}$. Thus, we can reduce k_1-approximating Clique Size to any function which k_2-approximates Clique Size.

In contrast, if MaxSat were self-improvable, then we can use Yannakakis's algorithm to k-approximate MaxSat for every constant k. By the lower bounds in Section 5, this would imply that P = NP. Again, the fact that MaxSat is not self-improvable can be deduced from its query complexity. The padding argument for the self-improvability of Clique Size works because stretching the input size increases the number of queries available to a $\mathrm{PF}^{\mathrm{SAT}[\log\log n]}$ machine. For example, squaring the input size makes 1 more query available. On the other hand, we only use a constant number of queries to k-approximate MaxSat. Since the number of queries does not depend on the input size, padding the input will not increase the number of queries available. This is also the reason why we only have a hardness result, and not a completeness result, on the query complexity of $n^{1/a}$-approximating Clique Size.

Thus, we can often deduce the self-improvability of an NP-approximation problem after we have discovered its query complexity. For example, we can now show that Chromatic Number and similar problems have the self-improvement property.[4]

Lemma 27 The NP-approximation problems Chromatic Number, Clique Partition, Clique Cover, and Biclique Cover have the self-improvement property. That is, given constants k_1 and k_2, with $k_1 < k_2$, and a function h that k_2-approximates the given problem, there exists a function f which k_1-approximates the problem such that $f \leq^{\mathrm{P}}_{\mathrm{mt}} h$.

Proof: To prove this lemma, we use the results of Lund and Yannakakis [LY94] which showed an approximation preserving reduction from Clique Size to each of the given problems. Thus, for constant k, k-approximating each of these problems is complete for $\mathrm{PF}^{\mathrm{SAT}[\log\log n + O(1)]}$. Hence, by the preceding discussion, these problems are self-improvable. □

7 Clique Size versus MaxSat

In this section, we compare the complexity of approximating Clique Size and approximating MaxSat using bounded queries as a quantitative complexity measure. We use these comparisons to argue that the approximability of MaxSat is not an indication that 3SAT is "easier" than Clique. Instead, it is an indication that MaxSat has a lot of "fluff" that is approximable. We argue this case by showing that if we add a linear amount of padding to Clique Size, then the resulting problem has a similar complexity to MaxSat. We also show that finding good approximations to the "non-approximable" part of MaxSat requires just as many queries as approximating Clique Size.

In our first comparison, we compare the query complexities of approximating Clique Size and MaxSat directly. By Corollaries 19 and 25, both 2-approximating Clique Size and

[4]These self-improvability results can be derived from previous work [PR90, LY94, KMSV94]. Our main point here is the relationship between bounded query classes and structural properties of NP-approximation problems.

$(1 + 1/\log n)$-approximating MaxSat are complete for $\mathrm{PF}^{\mathrm{SAT}[\log\log n + O(1)]}$. Thus, these two problems are inter-reducible. For finer approximations, we note that for both problems, obtaining a $(1+n^{-a})$-approximation for constant a, $0 < a < 1$ is complete for $\mathrm{PF}^{\mathrm{SAT}[O(\log n)]}$. Thus, in either case, approximating within a factor of $1 + n^{-a}$ is $\leq^{\mathrm{P}}_{\mathrm{mt}}$-equivalent to finding the optimal solution. For coarser approximations, note that approximating MaxSat within a constant factor requires only a constant number of queries. On the other hand, k-approximating Clique Size is complete for $\log\log n + O(1)$ queries. Thus, for constant factor approximations the complexity of approximating Clique Size and approximating MaxSat diverges. This can be explained by examining the upper bound given in Lemma 13. The number of queries used to find an $k(n)$-approximation of OPT_Π is the following (assuming there is a polynomial time algorithm which $A(n)$-approximates OPT_Π):

$$\log\log A(n) - \log\log k(n).$$

When $k(n) \geq 2$, the $\log\log A(n)$ term is significant. However, when $k(n)$ diminishes to 1 (e.g., $k(n) = 1 + 1/\log n$), the second term tends to dominate. For example, if we write $k(n)$ as $1 + 1/f(n)$ for some unbounded $f(n)$, we can use the Taylor series approximation $\ln(1 + 1/N) \approx 1/N$ for large N and rewrite the number of queries to be approximately:

$$\log\ln A(n) + \log f(n).$$

For large $f(n)$, the $\log f(n)$ term dominates the sum. This relationship explains why computing the $\omega(G)$ and $\mathrm{MaxSat}(F)$ exactly have the same complexity, but finding constant factor approximations of these two problems have different complexities.

Another method of comparing the complexities of MaxSat and Clique Size is to alter the Clique Size problem so we can match the upper and lower bounds of MaxSat. To do this, we define an admittedly contrived problem: Padded Clique Size. An instance of this problem is a padded graph — one that has a large obvious clique. To produce a padded graph from a general graph G with n vertices we can attach a $3n$-clique to G. The resulting graph G' will have $4n$ vertices and $\omega(G') = \omega(G) + 3n$. So, the number $3n$ is always a $4/3$-approximation of $\omega(G')$. Then, by altering Lemma 15, we can find constants ϵ_1, ϵ_2, with $3/4 < \epsilon_1 < \epsilon_2 < 1$, and a polynomial time computable function f such that given any 3CNF formula F, f produces a padded graph G' with n vertices with the property that:[5]

$$\begin{aligned} F \in \mathrm{SAT} &\implies \omega(G') = \epsilon_2 n \quad \text{and} \\ F \notin \mathrm{SAT} &\implies \omega(G') = \epsilon_1 n. \end{aligned}$$

Thus, no polynomial time algorithm can (ϵ_2/ϵ_1)-approximate Padded Clique Size, unless $\mathrm{P} = \mathrm{NP}$. In fact, every function in $\mathrm{PF}^{\mathrm{SAT}[\log a]} \leq^{\mathrm{P}}_{\mathrm{mt}}$-reduces to a function h that $(\epsilon_2/\epsilon_1)^{1/a}$-approximates Padded Clique Size. Therefore, the complexities of approximating MaxSat and Padded Clique Size are very similar.

[5]Here we do not want to use the graphs generated by the function in Lemma 15. These graphs have t^d vertices and cliques of size t^b or t^s, so the ratio t^b/t^d diminishes to zero. Instead, we will take the graph constructed from the probabilistically checkable proof for SAT which uses $O(\log n)$ random bits and a constant number of query bits. When $F \in \mathrm{SAT}$, the graph constructed from these probabilistically checkable proofs have cliques with a constant fraction of the vertices.

The preceding discussion shows that the traditional approach of using approximation factors to measure the quality of an approximation is highly sensitive to padding. We wonder if at its core, the complexity of approximating MaxSat is equivalent to the complexity of approximating Clique Size. For example, let F' be a formula produced by the reduction in Lemma 22. This formula has m clauses, but we know *a priori* that at least $m/(1+\delta)$ clauses are simultaneously satisfiable. Thus, if we want a function to guarantee good approximations, it would make sense to ask for the approximation x to guarantee that

$$\text{MaxSat}(F') - \frac{m}{1+\delta} < k(m) \cdot \left(x - \frac{m}{1+\delta} \right),$$

instead of simply asking for $\text{MaxSat}(F') < k(m)x$. That is, we would consider only the number of clauses in excess of the number we already know to be satisfiable. The complexity of finding an approximation x with such a guarantee would require roughly $\log \log_{k(m)} m$ queries — essentially the same as approximating Clique Size. Note that the value of δ in Lemma 22 is constructible and can be deduced from the size of the verifier in the probabilistically checkable proofs for SAT. The difficulty of generalizing this approach is that it would be difficult, if not impossible, to find the value of "δ" for general 3CNF formulas.[6] So, this measure of the quality of an approximation of $\text{MaxSat}(F)$ would only make sense for the formulas produced by the reduction in Lemma 22. Nevertheless, this example suggests that approximating the "hard part" of $\text{MaxSat}(F)$ is just as difficult as approximating Clique Size. In the next section, we show that if we treat the approximation version of 3SAT as a *minimization* problem instead of a maximization problem, then approximating 3SAT has the same complexity as approximating Clique Size.

8 Minimizing Unsatisfiability

The optimization version of the satisfiability problem is usually defined as a maximization problem: if we cannot find an assignment that satisfies all of the clauses, then we should at least try to find one that satisfies the largest number of clauses. This optimization problem has an equivalent minimization analog: finding an assignment that minimizes the number of unsatisfied clauses. We call this problem MIN3UNSAT. Note that the solution space of MAX3SAT and MIN3UNSAT are identical—they are truth assignments to the variables of the given Boolean formula. In fact, a better solution to the MIN3UNSAT problem *is* a better solution to the MAX3SAT problem. The only difference between the two problems is the cost which we associate with each feasible solution. For example, take a formula F with 120 clauses, let solution A be an assignment which satisfies 100 clauses and solution B be an assignment which satisfies 110 clauses. It would be typical to say that solution B is a 10 percent improvement over solution A. However, from the point of view of minimizing the number of unsatisfied clauses, solution B leaves only 10 clauses unsatisfied while solution A leaves 20 clauses unsatisfied. In this measure, solution B is twice as good as solution A. However, both measures regard solution B as the better solution. Only the quantitative ratio between the solutions changes from one measure to another.

[6] In general, if F is a 3CNF formula with n clauses, we know that at least $n/2$ of the clauses are simultaneously satisfiable.

In this section we show that the complexity of approximating MIN3UNSAT is equivalent to the complexity of approximating Clique Size. As a result, we can conclude that the approximability of MAX3SAT is not due to a lower intrinsic complexity of 3SAT, but is an artifact of our choice of viewing satisfiability as a maximization problem.

Definition 28 Let F be a 3CNF formula with n clauses. We define MinUnsat(F) to be the minimum number of unsatisfied clauses resulting from any assignment of truth values to the variables of F. A number x $k(n)$-approximates MinUnsat(F) if $x/k(n) < \text{MinUnsat}(F) \leq x$. In the special case where MinUnsat(F) $= 0$, we will allow 0 to be a $k(n)$-approximation of MinUnsat(F). A function f $k(\cdot)$-approximates MIN3UNSAT if $f(F)$ $k(|F|)$-approximates MinUnsat(F) for all formulas F.

For a 3CNF formula F with n clauses, we know that MinUnsat(F) $\leq n/2$, because given any assignment of truth values to the variables in F, either the assignment or its complement will satisfy half of the clauses in F. So, we know that MinUnsat(F) ranges from 0 to $n/2$. We can handle the special case where MinUnsat(F) $= 0$ by making 0 a singleton interval. Thus, using Lemma 13 we can produce an upper bound on the number of queries needed to approximate MinUnsat.

Corollary 29 Let $k(n)$ be polynomial time computable such that $1 < k(n) < n/2$. Then there exists a function in $\text{PF}^{\text{SAT}[q(n)]}$ which $k(\cdot)$-approximates MinUnsat, where $q(n) = \lceil \log \lceil \log_{k(n)}(n/2+1) + 1 \rceil \rceil$.

Theorem 30 Let $q(n) \in O(\log n)$ be a non-decreasing polynomial time computable function. Define $k(n)$ such that $q(n) = \log \log_{k(n)} n - 2$. If h is a function which $k(\cdot)$-approximates MinUnsat and $f \in \text{PF}^{\text{SAT}[q(n)]}$, then $f \leq^{\text{P}}_{\text{mt}} h$.

As before, the key to the theorem is a construction lemma. For this lemma, we need to translate Lemma 22 to the context of minimizing unsatisfied clauses.

Corollary 31 There exist constants α, c_1, c_2 and a polynomial time computable function f such that $0 < \alpha < 1$, $1 \leq c_1$, $1 \leq c_2$ and given a 3CNF formula F with t clauses, f produces a 3CNF formula F' with $\ell(t) = c_1 t^{c_2}$ clauses, where

$$F \in \text{SAT} \Longrightarrow F' \in \text{SAT} \Longrightarrow \text{MinUnsat}(F') = 0$$
$$F \notin \text{SAT} \Longrightarrow \text{MinUnsat}(F') = \alpha \ell(t).$$

Remark: The proof of Theorem 30 exploits the fact that MinUnsat(F') may be 0 to simplify the combinatorial analysis. However, this convenient feature is not an essential element of the proof. Similar theorems can be proven even if we restrict F' to be unsatisfiable or modified the objective function so that MinUnsat(F') ≥ 1.

Lemma 32 (Construction Lemma for MIN3UNSAT)
Let $k(n)$ be polynomial time computable such that $1 + 2n^{-1/4} \leq k(n) \leq n/2$ and let $F_1, \ldots, F_{r(t)}$ be a nested sequence where $|F_i| = t$, $m = \ell(t)^2$ and

$$r(t) \leq \left\lfloor (\log_{k(m)} m)/4 \right\rfloor .$$

Then, in polynomial time we can construct a 3CNF formula H with m clauses and whole numbers $y_0 > y_1 > \ldots > y_{r(t)}$ such that $y_i > k(m)y_{i+1}$ for $0 \leq i \leq r(t) - 1$, and

$$\text{MinUnsat}(H) = y_z \iff \#^{\text{SAT}}_{r(t)}(F_1, \ldots, F_{r(t)}) = z.$$

Proof: This proof is analogous to the construction in Lemma 24. One main difference is that MIN3UNSAT is a minimization problem. To construct H, let F_0 be an obviously satisfiable formula with t clauses. We take each F_i and use Lemma 31 to construct a formula F'_i with $\ell(t)$ clauses. As usual we let $\ell = \ell(t)$, $r = r(t)$ and $g = k(m)$. We define

$$a_i = \begin{cases} 0 & \text{if } i = 0 \\ \lceil g a_{i-1} \rceil + 1 & \text{for } 1 \leq i \leq r \end{cases}$$

The combined formula H is the conjunction of $\ell - a_r$ copies of F'_0, $a_r - a_{r-1}$ copies of F'_1, $a_{r-1} - a_{r-2}$ copies of F'_2, ...and one copy of F'_r. Now, if all of the formulas in $F_1, \ldots, F_r$ are satisfiable, then $\text{MinUnsat}(H) = 0$. If F_r is the only unsatisfiable formula, then we know that $\text{MinUnsat}(H) = \alpha\ell$. In general, if $\#^{\text{SAT}}_r(F_1, \ldots, F_r) = r - i$, then $\text{MinUnsat}(H) = \alpha a_i \ell$. So, we can define y_{r-i} to be $\alpha a_i \ell$, for $0 \leq i \leq r$. The requirement that $y_i > g y_{i+1}$ follows directly from the definition of a_{i+1}. Furthermore, the restriction that $k(m) \geq 1 + 2n^{-1/4}$ and $r \leq \lfloor (\log_{k(m)} m)/4 \rfloor$ guarantees that $a_r \leq \ell$. We omit the calculations here. □

Corollary 33
Let $k(n)$ be polynomial time computable such that $1 + 2n^{-1/4} \leq k(n) \leq n$. If there exists a polynomial time function which $k(\cdot)$-approximates $\text{MinUnsat}(F)$ using $\log \log_{k(n)} n - 2$ queries to an oracle X for all 3CNF formulas F with n clauses, then P = NP.

The results in this section show that the complexity of approximating MinUnsat is very close to the complexity of approximating Clique Size. The difference between the number of queries only differ by a constant. For classes closed under $\leq^{\text{P}}_{\text{mt}}$-reductions, approximating MinUnsat has the same complexity as approximating Clique Size.

Corollary 34 For all constants $k \geq 1$ and all constants a with $0 < a \leq 1$:

- $(1 + n^{-a})$-approximating MinUnsat is $\leq^{\text{P}}_{\text{mt}}$-complete for $\text{PF}^{\text{SAT}[O(\log n)]}$.
- $(1 + 1/\log^k n)$-approximating MinUnsat is $\leq^{\text{P}}_{\text{mt}}$-complete for $\text{PF}^{\text{SAT}[(k+1)\log\log n + O(1)]}$.
- k-approximating MinUnsat is $\leq^{\text{P}}_{\text{mt}}$-complete for $\text{PF}^{\text{SAT}[\log\log n + O(1)]}$.
- $(\log^k n)$-approximating MinUnsat is $\leq^{\text{P}}_{\text{mt}}$-complete for $\text{PF}^{\text{SAT}[\log\log n - \log\log\log n + O(1)]}$.

Thus, approximating Clique Size within a constant factor reduces to approximating MinUnsat within a constant factor, and vice versa. Furthermore, by the discussion in Section 4, these reductions are "approximation preserving." Hence, we can argue that approximating Clique Size and approximating MinUnsat really have the same complexity. Thus, the "approximability" MAX3SAT is not an indication that 3SAT has an intrinsically lower complexity than Clique. It is merely an artifact of viewing MAX3SAT as a maximization problem instead of a minimization problem. We prefer to think of MAX3SAT and MIN3UNSAT as complementary views of the same approximation problem.

There are other examples of such complementary NP-approximation problems. For a graph $G = (V, E)$, V' is a vertex cover if and only if $V - V'$ is an independent set [GJ79]. An approximate solution to the vertex cover problem leads immediately to an approximate solution to independent set (which is equivalent to Clique) and vice versa. Thus, approximating the size of the smallest vertex cover has the same complexity as approximating MaxSat and approximating the size of the largest independent set has the same complexity as approximating Clique Size and MinUnsat. Finally, it is interesting to point out that finding the *optimal* solutions to these problems are all complete for $\mathrm{PF}^{\mathrm{SAT}[O(\log n)]}$ and are thus equivalent under $\leq^{\mathrm{P}}_{\mathrm{mt}}$-reductions [Kre88].

9 Discussion

The results in this paper along with the results in Chang, Gasarch and Lund [CGL94] show that counting queries is a good measure of the complexity of NP-approximation problems. We have derived close upper bounds and relative lower bounds on the number of queries needed to solve various NP-approximation problems. Furthermore, the completeness results show that queries to SAT and finding approximate solutions to these NP-approximation problems are equivalent. That is, with more queries we can find better solutions, and if we can find better solutions then we can answer more queries. In addition, we can quantify the relationship between the approximation factor and the number of queries. This quantitative relationship allows us to deduce some structural results about the NP-approximation problems — namely, self-improvability and inter-reducibility between NP-approximation problems. Finally, using a uniform and quantitative complexity measure has allowed us to compare the difficulty of approximating clique size and satisfiability. We have shown that whether one prefers to view these two problems as having equivalent complexity (or whether one prefers the conventional wisdom that maximum clique is more difficult than satisfiability) does not depend upon the intrinsic complexity of these problems. Instead, it depends on whether one prefers to treat satisfiability as a maximization problem or a minimization problem.

Further studies in this area have expanded the applicability of using bounded queries as a complexity measure for NP-approximation problems. Crescenzi, Kann, Silvestri and Trevisan [CKST95] have considered whether there exists a reduction from finding the vertices of the largest clique to a function that merely finds the vertices of a 2-approximate clique. They have shown that if such a reduction exists, then the Polynomial Hierarchy must collapse. These results must deal with not just finding an approximation of the size of the largest clique, but also producing the vertices in the approximate clique. It turns out that the quantitative results in this paper can also be extended to the domain of finding approximate

solutions to an NP-optimization problem Π instead of simply approximating the cost of the optimal solution OPT_Π. For example, an NP machine with $\lceil \log \lceil \log n \rceil \rceil$ queries to SAT in its entire computation tree can find the vertices of a 2-approximate clique. Moreover, every multi-valued function computed by such NP machines can be reduced to finding the vertices of a 2-approximate clique. Furthermore, one can show that an NP machine with additional queries to SAT can compute more functions unless PH collapses. These theorems depend upon results about the Boolean Hierarchy and is beyond the scope of this paper. We encourage the reader to consult the literature on expositions of these recent developments [Cha94b].

Acknowledgments

The author would like to thank Bill Gasarch, Carsten Lund, Madhu Sudan, Suresh Chari, Christine Piatko and Ming Li for several fruitful discussions. In addition many thanks go to Richard Beigel, Jacobo Torán and Luca Trevisan for suggestions in revising this paper. Finally, the author would like to acknowledge the journal referee for several simplifications in the proof of the main theorem.

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.

[Bei88] R. Beigel. NP-hard sets are p-superterse unless R = NP. Technical Report 4, Department of Computer Science, The Johns Hopkins University, 1988.

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

[CG93] R. Chang and W. I. Gasarch. On bounded queries and approximation. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 547–556, November 1993.

[CGH+88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The Boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, December 1988.

[CGL94] R. Chang, W. I. Gasarch, and C. Lund. On bounded queries and approximation. Technical Report TR CS-94-05, Department of Computer Science, University of Maryland Baltimore County, April 1994. To appear in *SIAM Journal on Computing*.

[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.

[Cha95] R. Chang. On the query complexity of clique size and maximum satisfiability. Technical Report TR CS-95-01, Department of Computer Science, University of Maryland Baltimore County, May 1995.

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

[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.

[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*, volume 665 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[KMSV94] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 819–830, November 1994.

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

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

[PR90] A. Panconesi and D. Ranjan. Quantifiers and approximation. In *ACM Symposium on Theory of Computing*, pages 446–456, 1990.

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[Yan92] M. Yannakakis. On the approximation of maximum satisfiability. In *Proceedings of the 3rd Symposium on Discrete Algorithms*, pages 1–9, 1992.

[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.