# On the approximability of Clique and related maximization problems ⋆

Aravind Srinivasan [1]

**Proposed running head:** Approximability of maximization problems

**Abstract**

We consider approximations of the form $n^{1-o(1)}$ for the Maximum Clique problem, where $n$ is the number of vertices in the input graph and where the "$o(1)$" term goes to zero as $n$ increases. We show that sufficiently strong negative results for such problems, which we call *strong* inapproximability results, have interesting consequences for exact computation. A simple sampling method underlies most of our results.

*Key words:* Inapproximability, approximation algorithms, clique, independent set, packing integer programs, random sampling

⋆ A preliminary version of this work appears as "The value of strong inapproximability results for clique" in the *Proc. ACM Symposium on Theory of Computing*, pages 144–152, 2000.

[1] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA. Most of this work was done while at Bell Laboratories, Lucent Technologies; part of this work was supported by NSF Award CCR-0208005. E-mail: `srin@cs.umd.edu`. URL: `http://www.cs.umd.edu/~srin`

## 1 Introduction

By presenting a connection between interactive proof systems and approximation algorithms, the seminal work of [8] has led to several recent results on the hardness of *approximating* various optimization problems. These involve ingenious reductions from decision problems to suitable approximation problems; see [3,2] and their several interesting followup results. Here, we take the approach of reducing approximation to optimization/approximation: crucially, these reductions lead to instances that are *much smaller* than the original instance. In particular, this lets us show some interesting implications of sufficiently good hardness-of-approximation results for the Maximum Clique problem.

The Maximum Clique problem is a classical problem in combinatorial optimization. Given an undirected graph $G$, a clique in $G$ is a subset of the vertices of $G$ in which every vertex is adjacent to every other vertex. The Maximum Clique problem (henceforth referred to just as "clique") is that of finding a clique of maximum cardinality in a given graph, and was one of the early ones shown to be $NP$-hard. Given a parameter $\rho \geq 1$, recall that a $\rho$-approximation algorithm for a maximization problem is one that always returns a feasible solution that is at least $1/\rho$ times optimal. Such an algorithm is also said to approximate the problem to within $\rho$, and $\rho$ is called the *approximation guarantee* or *approximation ratio* of the algorithm. The current-best approximation ratio for clique that is achievable in polynomial time is $O(n/\log^2 n)$ [5], where $n$ denotes the number of vertices in the input graph. Recent years have seen much progress in understanding the inapproximability of the problem, culminating in the celebrated result of [12] that it cannot be approximated in $ZPP$ to within $n^{1-\epsilon}$ for any fixed $\epsilon > 0$, unless $NP \subseteq ZPP$; see [16] for a simplified proof. Arguably, the Lovász $\vartheta$-function may yield the "best" approximation guarantee for clique. It has been shown that the worst-case approximation guarantee of this function is at least as high as $n/2^{\Omega(\sqrt{\log n})}$ [7]. The reader is referred to [10] for a survey of the (in)approximability of clique.

It would be of much interest to understand the worst-case polynomial-time approximability of clique. In particular, it is mentioned in [10] that "It is not as presumptuous now to conjecture that the ultimate ratio is $n/polylog(n)$ as it was in 1991 ...". It would be very interesting to resolve this conjecture.

Before proceeding to discuss our results, we now introduce some notation.

**Notation.** $Z_+$ stands for the set of non-negative integers; $\log x$ denotes $\log_2 x$. Throughout, we let $N$ denote the input size of an instance of an $NP$ problem; for all of §1, $n$ will denote the number of vertices in an input graph, and "$o(1)$" will denote some function of $n$ that goes to zero as $n$ increases.

$DTIME$, $BPTIME$ and $ZPTIME$ respectively denote deterministic time, bounded-error probabilistic time, and zero-error probabilistic time as usual. [Though these classes are defined for decision problems, we will also use them to describe (optimization) algorithms in the obvious way.] For randomized algorithms, we will consider Las Vegas algorithms in some contexts, and Monte Carlo algorithms in others. However, "randomized complexity of $NP$" will throughout mean the worst-case expected running time of randomized algorithms that always return the correct answer, for $NP$ languages: i.e., Las Vegas algorithms. More specifically, in every usage of the phrase "randomized complexity of $NP$", we will either say that the randomized complexity of $NP$ is $O(T_1(N))$, or that it is $\Omega(T_2(N))$. The former would mean that every $NP$ language has a Las Vegas algorithm with worst-case expected running time being $O(T_1(N))$; the latter means that there exists an $NP$ language for which the worst-case expected running time of every Las Vegas algorithm is $\Omega(T_2(N))$. Similar remarks hold when we speak of the deterministic complexity of $NP$.

As seen in the above few paragraphs, the "best-possible" polynomial-time approximation ratio for clique seems to be in the $n^{1-o(1)}$ range. One of the main themes of our work is showing why the (in-)approximability of clique in the $n^{1-o(1)}$ range would be very useful to understand. Let us call a result such as "Clique is hard to approximate to within $n/f(n)$", where $f(n)$ is a function that grows as $n^{o(1)}$, a *strong* inapproximability result for clique. Strong inapproximability results for clique are already on their way; using different notation than in [6], one key result of [6] is

**Theorem 1.1 ([6])** *Let $\mathcal{EHA}$ be the complexity-theoretic assumption "$NP \not\subseteq ZPTIME[2^{O(\log N (\log\log N)^{3/2})}]$"; define $EH(a,x) = x^{a/\sqrt{\log\log x}}$. Then, assuming $\mathcal{EHA}$, there is a constant $\gamma > 0$ such that clique cannot be approximated to within $n/EH(\gamma, n)$ by any $ZPP$ algorithm.*

Following this work, an even stronger inapproximability result for clique has been obtained [13]. The consequences of this result (and of any future strong inapproximability result) that follow from our work can be derived in a manner identical to how we derive some consequences of Theorem 1.1; this issue is sketched in §6.

Our approach reduces certain families of approximation problems to optimization or approximation problems of much smaller size, using elementary (random) sampling. As a simple starting point, suppose we want to approximate clique to within some factor $\rho$ in an $n$-vertex graph $G$. We can assume that the maximum clique size, say *opt*, in $G$ is at least $\rho$ (since otherwise any single vertex will be a $\rho$-approximation). Fix a maximum clique $C$. Suppose, for some parameter $1 \leq t \leq \rho$, we choose a set $S$ of $nt/\rho$ vertices from $G$ at random. Note that the expected number of vertices of $C$ in $S$ is $opt \cdot t/\rho$; if indeed $S$ has this many vertices of $C$, then a $t$-approximation for the clique problem on the subgraph induced by $S$, will be a $\rho$-approximation of clique in $G$, as desired. In other words, we have reduced $\rho$-approximation for a problem of size $n$ to $t$-approximation for a problem of size $nt/\rho$. Various ways of viewing and building on this idea lead to the following applications.

**(a) A sufficient condition for a gap in the location of $NP$.** A step toward further understanding the complexity of $NP$ would be to show the following, *under no assumptions*: for some explicit functions $f_1$ and $f_2$ such that $\lim_{N\to\infty} f_1(N)/f_2(N) = 0$, $NP$'s (deterministic or randomized) complexity is either $O(f_1(N))$ or $\Omega(f_2(N))$. Our approach helps show that sufficiently good strong inapproximability results for clique lead to such results. Suppose a strong inapproximability result is shown for clique, under some hardness assumption $\mathcal{A}$. We show that any such result yields explicit super-polynomial lower bounds on $NP$, under the assumption $\mathcal{A}$. For instance, let $f$ and $F$ be any functions satisfying

$$\lim_{n\to\infty} f(n) = \infty \text{ and } f(n) = n^{o(1)}; \tag{1}$$

$$F(z) = \min\{y \in Z_+ :\ 3(f(y))^2 \geq z\}. \tag{2}$$

An example result of ours is as follows. *Suppose clique cannot be approximated to within $n/f(n)$ in $ZPP$ assuming $\mathcal{A}$ holds; then, assuming $\mathcal{A}$, the randomized complexity of $NP$ is $\Omega((F(N))^{\omega(1)})$*; "$\omega(1)$" refers to some function of $N$ that goes to infinity as $N$ increases. Similar analogs hold for the deterministic complexity of $NP$. Note that $F$ is **superpolynomial** since $f(n) = n^{o(1)}$. For instance, if $f(n) = 2^{\Theta((\log n)^a)}$ for some constant $0 < a < 1$, then $F(N) = 2^{\Theta((\log N)^{1/a})}$; if $f(n) = (\log n)^c$, then $F(N) = 2^{\Theta(N^{1/c})}$. An interesting case here is when $\mathcal{A} \equiv (NP \not\subseteq ZPP)$: a proof (which is likely to be very challenging to obtain) that clique cannot be approximated to within $n/f(n)$ in $ZPP$ assuming $NP \not\subseteq ZPP$, will provably show that $NP$'s randomized complexity is either polynomially bounded or is at least the explicit super-polynomial function $(F(N))^{\omega(1)}$. Our italicized result above shows that *any* strong inapproximability result for clique has a partial converse. For instance, our partial converse to Theorem 1.1 is: *if clique cannot be approximated to within $n/EH(\gamma, n)$ by any $ZPP$ algorithm, then the randomized complexity of $NP$ is at least $N^{\omega(\sqrt{\log\log N})}$.*

**(b) An approach to $P$ vs. $NP$.** Our results also show the following. Let the randomized unit-cost RAM (random-access machine) be the standard unit-cost RAM with the additional power to draw any number of unbiased, independent random bits, each bit in unit time. We start with a definition:

**Definition 1.1** *Let $f$ be any function satisfying (1); let $\epsilon$ be any positive constant. Consider the problem of approximating clique to within $n/f(n)$; using* *the notation* $[X, Y, Z]$ *for "lower bound of* $X$ *for input graph represented in format* $Y$*, in computational model* $Z$*" for this problem, we define four types of lower-bound assertions:*

**(LB1)** *denotes* $[n^{2+\epsilon},$ *adjacency list, deterministic unit-cost RAM*$]$*;*

**(LB2)** *denotes* $[n^{1+\epsilon},$ *adjacency matrix, deterministic unit-cost RAM*$]$*;*

**(LB3)** *denotes* $[n^{1+\epsilon},$ *adjacency list, randomized unit-cost RAM*$]$*; and*

**(LB4)** *denotes* $[n^{\epsilon},$ *adjacency matrix, randomized unit-cost RAM*$]$*.*

*The lower bounds* (LB3) *and* (LB4) *should be on the running time of any Monte Carlo algorithm that produces an* $n/f(n)$*-approximation with probability at least* $1/2$*.*

Fix any $f$ and $\epsilon$ that satisfy the hypotheses of Definition 1.1. We show that (LB1) implies $NP \neq P$, (LB2) implies $NP \neq P$, (LB3) implies $NP \not\subseteq BPP$, and (LB4) implies $NP \not\subseteq BPP$. In fact, given $f$, let $F$ be any function satisfying (2); we get super-polynomial lower bounds of $(F(N))^{\Omega(1)}$ on $NP$ in all these cases. (Very small lower bounds such as in (LB4) may seem obvious: e.g., one may think that it could take $n^{\Omega(1)}$ time to even write out an $n/f(n)$-approximate solution. This is not the case. For instance, if the graph has a clique number of at least $f(n)$, an $n/f(n)$-approximation algorithm just needs to output some clique of size $f(n) = n^{o(1)}$; if the clique number is less than $f(n)$, the algorithm can just output a one-vertex "clique".)

These seem potentially fruitful for two reasons. First, another result of [6] is that for a certain constant $\lambda > 0$, clique cannot be approximated to within $n/EH(\lambda, n)$ by any Monte-Carlo polynomial-time algorithm with probability at least $1/2$, unless $NP \subseteq BPTIME[2^{O(\log N (\log\log N)^{3/2})}]$. Thus, lower bounds such as $n^{\epsilon}$ and $n^{2+\epsilon}$ seem likely to hold for this approximation problem. To us, the second interesting point is that these required lower bounds are quite small; e.g., $n^{\epsilon}$ is sub-linear. A simple argument in §4.2 shows that the $n/EH(\lambda, n)$–inapproximability result of [6] holds even when restricted to graph families with $\Theta(n^2)$ edges; $n^{1+\epsilon}$ is also sub-linear in these cases. These reasons suggest that this may be a worthwhile path to explore for goals such as "$NP \not\subseteq BPP$" and "$NP \neq P$" in some restricted models of computation.

**(c) Bootstrapping hardness results.** Our approach leads to a bootstrapping of hardness-of-approximation results. This yields higher lower bounds on the running time of clique approximation algorithms, by appropriately lessening the approximation ratio. Applying this to Theorem 1.1, we get, under the assumption $\mathcal{EHA}$, that: (i) for any constants $\nu > \gamma$ and $0 < b < 1/2$, clique cannot be approximated to within $n^{1-\nu/(\log\log n)^b}$ in $ZPTIME[n^{O((\log\log n)^{1/2-b})}]$;

(ii) for any fixed $\epsilon > 0$, clique cannot be approximated to within $n^{1-\epsilon}$ in $ZPTIME[n^{O(\sqrt{\log\log n})}]$. Thus, $\mathcal{EHA}$ yields explicit superpolynomial lower bounds on the running time of even approximating clique (say, to within $n^{1-\epsilon}$). We remark that our bootstrapping only requires hardness results such as Theorem 1.1 as a black-box (i.e., we do not require the internal details of these proofs).

**(d) From hardness to almost-everywhere hardness.** Our method (or, more accurately, reduction) also leads to a way of strengthening hardness-of-approximation assumptions/results to certain "almost-everywhere hardness" versions. §4.4 demonstrates an instance of this on the $n/EH(\lambda, n)$–hardness result of [6] defined in (b) above.

**(e) Time-approximation tradeoffs.** For an abstract family of maximization problems that includes clique, hypergraph matching etc., earlier work of [11] presents approximation algorithms that achieve a certain tradeoff between approximation ratio and running time. Our approach yields similar results. This tradeoff is also shown to be essentially optimal for general problems belonging to this family.

After setting up some preliminary notions in §2, we present the basic approach in §3. Various ways of using and viewing this approach lead to the hardness results shown in §4. Tradeoffs between approximability and a measure of running time (the "query complexity") are studied in §5. Finally, concluding remarks are made in §6.

## 2 Preliminaries

**Large-deviation bounds.** We recall a large-deviation bound for a relative of the hypergeometric distribution. Let $[s]$ denote the set $\{1, 2, \ldots, s\}$. Suppose that we are given positive integers $r, s$ with $r \leq s$, and reals $\psi_1, \psi_2, \ldots, \psi_s \in [0, 1]$. Choose a random $r$-element subset $T$ of $[s]$, each $r$-element subset being equally likely. Consider any $A \subseteq [s]$; if $X$ denotes $\sum_{i \in (A \cap T)} \psi_i$, then $\mathbf{E}[X] = (r/s) \cdot \sum_{i \in A} \psi_i$. Tail bounds from, e.g., [17] show for any $\delta \in [0, 1]$ that

$$\mathbf{Pr}[X \leq (1-\delta)\mathbf{E}[X]] \leq e^{-\mathbf{E}[X]\delta^2/2}; \tag{3}$$

here and from now on, $e$ denotes the base of the natural logarithm.

**A class of optimization problems.** In presenting our approximation algorithms, it will help to consider the following generalization of the clique and maximum independent set (MIS) problem on graphs. This class also generalizes the *weighted* versions of these problems (e.g., given a non-negative weight for each vertex, weighted maximum clique asks for a clique of maximum total weight). Given $\vec{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ and $\vec{y} = (y_1, y_2, \ldots, y_n) \in \{0,1\}^n$, we will say that $\vec{y} \preceq \vec{x}$ iff $y_i \leq x_i$ for each $i$. Call a Boolean function $g : \{0,1\}^n \rightarrow \{0,1\}$ *monotone decreasing* iff: for every $\vec{x} \in \{0,1\}^n$ such that $g(\vec{x}) = 1$, and for all $\vec{y} \preceq \vec{x}$, $g(\vec{y}) = 1$. (Such functions are sometimes also called *hereditary properties.*) Let $\Re_+$ denote the set of non-negative reals. The abstract class of maximization problems we consider consists of maximization problems called $(n, g, \vec{w})$-OPT, where $g$ is a *monotone decreasing* Boolean function mapping $\{0,1\}^n$ to $\{0,1\}$, and $\vec{w} \in \Re_+^n$. The $(n, g, \vec{w})$-OPT problem is to find some $\vec{x} \in \{0,1\}^n$ that maximizes $\vec{w} \cdot \vec{x}$, subject to $g(\vec{x}) = 1$. We are also assured that $g(0, 0, \ldots, 0) = 1$, which is equivalent to saying that there exists some $\vec{x}$ for which $g(\vec{x}) = 1$. We assume oracle access to $g$: given any $\vec{x} \in \{0,1\}^n$, an oracle for $g$ gives us the value of $g(\vec{x})$.

In the case of weighted clique, given a vector $\vec{v}$ that is the incidence vector of a subset $S$ of the vertices, we define $g(\vec{v}) = 1$ iff $S$ is a clique. It is immediate that this $g$ is polynomial-time computable, and that $(n, g, \vec{w})$-OPT here is the weighted maximum clique problem. Similarly, it is easy to see how this framework captures the weighted MIS problem. §5 will also discuss applications to certain types of *packing integer programs.*

**A family of "$n^{o(1)}$" functions.** Henceforth, a "$o(1)$" term will always be associated either with $N$ (typically, the size of an input instance for an $NP$ language) or $n$ (typically, associated with an $(n, g, \vec{w})$-OPT problem). The $o(1)$ term will be some value that goes to 0 as either $N$ or $n$ tends to infinity; there will be no ambiguity here, as there will be no usage of "$o(1)$" in terms where both $N$ and $n$ occur. We will define a family of functions, all of which are of the form "$n^{o(1)}$"; we call these *desirable* functions. The two required properties of a desirable function $f(n)$ are that $\lim_{n\to\infty} f(n) = \infty$ and that $f(n) = n^{o(1)}$; i.e., $\lim_{n\to\infty}(\log f(n))/(\log n) = 0$. (In particular, we require that these two limits exist; however, these requirements can be weakened. Since this technicality is not important in our applications, we do not discuss how to weaken these requirements.) Functions such as $(\log n)^c$, $2^{\Theta((\log n)^a)}$ or $n^{\Theta((\log\log n)^{-c})}$, where $c > 0$ and $a \in (0,1)$ are constants, are desirable in this sense.

## 3 A simple sampling-based reduction

We now present an elementary sampling approach that motivates most of our results. The discussion in this section is defined by four parameters: a positive integer $n$, an arbitrary constant $\kappa > 0$, and reals $1 + \kappa \leq \rho \leq n$ and $1 \leq t \leq \rho/(1 + \kappa)$. It may be helpful to keep the following interpretations of these parameters in mind. The parameter $n$ will in general be a measure of the input size of a given problem: e.g., we could be considering the maximum clique problem on $n$-vertex graphs. The parameter $\rho$ will typically be our targeted approximation guarantee. Note that $t \leq \rho$; we will often be interested in the case where $t \ll \rho$. The goal of the discussion below is to reduce the given $\rho$-approximation problem on instances of size $n$ to a $t$-approximation problem on instances of size $\Theta(nt/\rho)$. In particular, note that if $t \ll \rho$, then we may have achieved a significant size-reduction. Finally, $\kappa$ will typically be a constant in the range $(0, 1)$.

Suppose we wish to approximate a given instance of $(n, g, \vec{w})$-OPT to within $\rho$. We show a simple way of reducing this to optimization/approximation on certain "much smaller" instances of the problem. We first define what "much smaller" means here. Given $S \subseteq [n]$ and $\vec{v} = (v_1, v_2, \ldots, v_n)$, let $\vec{v}_S$ be the vector $(v_1', v_2', \ldots, v_n')$, where $v_i' = v_i$ if $i \in S$ and $v_i = 0$ if $i \notin S$. Now, if $|S|$ is "small" compared to $n$, we informally consider $(n, g, \vec{w}_S)$-OPT to be a "much smaller" problem than the original $(n, g, \vec{w})$-OPT. This is because the indices $i \notin S$ have effectively been eliminated from consideration: for any $\vec{x}$ with $g(\vec{x}) = 1$, we have $g(\vec{x}_S) = 1$ and $\vec{w}_S \cdot \vec{x} = \vec{w}_S \cdot \vec{x}_S$. So, we can restrict attention to those $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ for which $x_i = 0$ for all $i \notin S$. When specialized to concrete problems such as maximum clique, we will get interesting consequences.

Suppose we are given an instance of $(n, g, \vec{w})$-OPT which we wish to approximate within $\rho$. Letting $\vec{w} = (w_1, w_2, \ldots, w_n)$, we will assume throughout by scaling that $\max_i w_i = 1$. Given a bit vector $\vec{v}$, its *Hamming weight*, denoted *Hweight* here, is the number of "1" bits in it. (Our usage of the term "Hweight" is to distinguish this usage of "weight" from usage such as "weighted clique".) Let $\mathbf{e}_i$ be the $n$-bit vector of Hweight 1 with a "1" in precisely the $i$th coordinate. If $g(\mathbf{e}_i) = 0$, the $i$th coordinate is essentially irrelevant for the problem, since any $\vec{x}$ for which $g(\vec{x}) = 1$, will have a "0" in the $i$th coordinate (as $g$ is monotone decreasing). So, letting $I$ denote the set of indices $i$ for which $g(\mathbf{e}_i) = 1$, our problem reduces to $(n, g, \vec{w}_I)$-OPT. To avoid extra notation, we will assume without loss of generality that $I = [n]$.

Our reduction is given by

**Theorem 3.1** *Let $\kappa$ be some positive constant. Suppose we are given an instance of $(n, g, \vec{w})$-OPT, as well as two reals $\rho$ and $t$ such that $1 + \kappa \leq \rho \leq n$ and $1 \leq t \leq \rho/(1 + \kappa)$. We assume that $g(\mathbf{e}_i) = 1$ for all $i$, and that $\vec{w} = (w_1, w_2, \ldots, w_n)$, where $\max_i w_i = 1$; let $i_0$ be an index such that $w_{i_0} = 1$. Let $\vec{u}$ be an optimal solution for the given $(n, g, \vec{w})$-OPT instance. Then:*

***(i)*** $\max\{\lceil nt(1 + \kappa)/\rho\rceil, \lceil n/\lfloor \rho/t\rfloor\rceil\} \leq \min\{2nt(1 + \kappa)/\rho, n\}$.
***(ii)*** *Define $C = \lceil 2(1 + \kappa)\ln(10)/(t\kappa^2)\rceil$. Independently choose $C$ subsets*

*$S_1, S_2, \ldots, S_C$ of $[n]$, each from the uniform distribution over subsets of size $\lceil nt(1+\kappa)/\rho \rceil$. Then:*

- *If the optimal solution value for the given problem is at least $\rho$, then for each $i$,*

$$\mathbf{Pr}[\vec{w}_{S_i} \cdot \vec{u} < opt \cdot t/\rho] \leq e^{-t\kappa^2/(2(1+\kappa))}. \tag{4}$$

- *Let $x_i$ be a $t$-approximation for $(n, g, \vec{w}_{S_i})$-OPT for $i \geq 1$. Then, any vector among $\{\mathbf{e}_{i_0}, x_1, x_2, \ldots, x_C\}$ that maximizes $\vec{w} \cdot x_i$, is a $\rho$-approximation for $(n, g, \vec{w})$-OPT, with probability at least $0.9$; this probability is only over the random choice of the $S_i$.*

***(iii)*** *Define $\ell = \lfloor \rho/t \rfloor$, and partition $[n]$ arbitrarily into $\ell$ subsets $A_1, A_2, \ldots, A_\ell$, each of cardinality at most $\lceil n/\ell \rceil$. Let $y_i$ be a $t$-approximation for $(n, g, \vec{w}_{A_i})$-OPT for $i \geq 1$. Then, any vector among $\{\mathbf{e}_{i_0}, y_1, y_2, \ldots, y_\ell\}$ that maximizes $\vec{w} \cdot y_i$, is a $\rho$-approximation for $(n, g, \vec{w})$-OPT.*

**Proof:** Part (i) is a simple consequence of the facts that $1 + \kappa \leq \rho \leq n$ and $1 \leq t \leq \rho/(1+\kappa)$.

Let *opt* denote the optimal objective function value for the given $(n, g, \vec{w})$-OPT instance. If $opt \leq \rho$, then it is easy to see that $\mathbf{e}_{i_0}$ would provide the required $\rho$-approximation; this will then complete the proofs of parts (ii) and (iii). Thus, we are left with the (interesting) case where $opt > \rho$. Recall that $\vec{u}$ is an optimal solution for the given $(n, g, \vec{w})$-OPT instance: so, $g(\vec{u}) = 1$ and $\vec{w} \cdot \vec{u} = opt$.

(ii) For each $i$, we have $g(\vec{u}_{S_i}) = 1$ with probability one, since $g$ is monotone decreasing. Simple use of (3) along with the fact that $opt > \rho$, shows (4). Thus, the probability that "$\vec{w}_{S_i} \cdot \vec{u} < opt \cdot t/\rho$" held for all $i$ is at most

$$e^{-tC\kappa^2/(2(1+\kappa))} \leq e^{-\ln(10)} = 0.1.$$

Now, if $\vec{w}_{S_i} \cdot \vec{u} \geq opt \cdot t/\rho$, then a $t$-approximation for $(n, g, \vec{w}_{S_i})$-OPT is also a $\rho$-approximation for the given $(n, g, \vec{w})$-OPT problem, completing the proof of part (ii).

(iii) It is immediate that there is some $i$ for which $\vec{w}_{A_i} \cdot \vec{u} \geq opt/\ell \geq opt \cdot t/\rho$. Hence, if we can $t$-approximate all of the $(n, g, \vec{w}_{A_i})$-OPT, the best among the $\ell$ solutions produced will be a $\rho$-approximation for the given $(n, g, \vec{w})$-OPT instance. ■

Thus, part (ii) is a simple randomized reduction which, with high probability, reduces the original $\rho$-approximation problem to $t$-approximation problems on a constant number of instances $(n, g, \vec{w}_{S_1})$-OPT, $(n, g, \vec{w}_{S_2})$-OPT, …,

$(n, g, \vec{w}_{S_C})$-OPT, where $|S_1| = |S_2| = \cdots = |S_C| = \lceil nt(1+\kappa)/\rho \rceil$. (Note that $C$ is bounded by the constant $\lceil 2(1+\kappa)\ln(10)/\kappa^2 \rceil$.) In other words, if we can correctly $t$-approximate all of the constant number of randomly generated $(n, g, \vec{w}_{S_i})$-OPT, we get a $\rho$-approximation for the original $(n, g, \vec{w})$-OPT instance with high probability. Part (iii) is a slightly slower deterministic version of this reduction. Also note from part (i) that the "sizes" of the size-reduced instances in our randomized as well as deterministic reductions, are at most $2nt(1+\kappa)/\rho$. Finally, if we plug $t = 1$ into Theorem 3.1, we get a reduction from approximation to exact optimization.

Specialized to the (unweighted) maximum clique problem, where we desire a $\rho$-approximation for $n$-vertex graphs, we get the following. Part (ii) of Theorem 3.1 basically shows a randomized reduction to a constant number of instances of $t$-approximation for graphs with at most $2nt(1+\kappa)/\rho$ vertices; part (iii) presents a deterministic reduction to $\lfloor \rho/t \rfloor$ instances of $t$-approximation for graphs with at most $2nt(1+\kappa)/\rho$ vertices. (Furthermore, the new graphs produced are induced subgraphs of the given graph.)

## 4 Hardness results

We now move on to various negative results that follow from Theorem 3.1. Suppose $f(n)$ is an arbitrary *desirable* function; let $\mathcal{A}$ be some complexity-theoretic assumption (such as "$NP \not\subseteq ZPP$"). We will present consequences of results of the following type:

**(R1)** "If $\mathcal{A}$ holds, then clique cannot be approximated to within $n/f(n)$, in $DTIME[poly(n)]$."

**(R2)** "If $\mathcal{A}$ holds, then clique cannot be approximated to within $n/f(n)$, in $ZPTIME[poly(n)]$."

In both (R1) and (R2), $n$ denotes the number of vertices in the input graph.

Note that an unweighted clique problem on a graph with $v$ vertices can be expressed with "input size" at most $v^2/2$: e.g., by writing down $v$ and, for each of the $\binom{v}{2}$ possible edges, a bit denoting whether the edge is in the graph or not. For all of §4, any use of Theorem 3.1 will set $\kappa = \sqrt{1.5} - 1$. Thus, whenever we specialize this reduction to the unweighted clique problem on an original graph that has $n$ vertices, the size-reduced instances (which have at most $2nt(1+\kappa)/\rho$ vertices) can be expressed with input size at most $(2nt(1+\kappa)/\rho)^2/2 = 3(nt/\rho)^2$.

The results of this section will hold for any $(n, g, \vec{w})$-OPT problem, but we use the clique problem for concreteness.

### *4.1 A sufficient condition for a gap in the location of NP*

We now consider application (a) presented in §1.

**Theorem 4.1** *Let $f$ be an arbitrary desirable function; define $h(z) \doteq \min\{y \in Z_+ : 3(f(y))^2 \geq z\}$. Then:*

*(i) $h$ is superpolynomial; i.e.,* $\lim_{z\to\infty}(\log h(z))/\log z = \infty$.

*(ii) Suppose (R1) is true for $f$ and for some assumption $\mathcal{A}$. Then under assumption $\mathcal{A}$, the deterministic complexity of $NP$ is $\Omega((h(N))^{s(N)})$, for some function $s(N)$ such that* $\limsup_{N\to\infty} s(N) = \infty$.

*(iii) Suppose (R2) is true for $f$ and for some assumption $\mathcal{A}$. Then, under assumption $\mathcal{A}$, the randomized complexity of $NP$ is $\Omega((h(N))^{s(N)})$, for some function $s(N)$ such that* $\limsup_{N\to\infty} s(N) = \infty$.

**Proof:** (i) This easily follows from the fact that $f(n) = n^{o(1)}$.

(ii) Suppose for a contradiction that $\mathcal{A}$ and (R1) are true, but that the deterministic complexity of $NP$ is at most $(h(N))^{O(1)}$. Set $\rho = n/f(n)$, $t = 1$, and $\kappa = \sqrt{1.5} - 1$. Then, Theorem 3.1(iii) shows that approximating clique (on an $n$-vertex graph) to within $n/f(n)$ reduces to $O(n/f(n))$ many instances of clique, each instance having at most $2\sqrt{1.5}f(n)$ vertices; thus, each of these instances can be expressed in $poly(n)$ time with input size at most $I = 3(f(n))^2$. So, if the deterministic complexity of $NP$ is $(h(N))^{O(1)}$, each of these "small" maximum clique instances can be solved in time $(h(I))^{O(1)} = n^{O(1)}$; this would show that we can approximate clique to within $n/f(n)$ in $DTIME[poly(n)]$, contradicting (R1). This proves (ii); the proof of (iii) is essentially identical. ■

Some of the most interesting applications of Theorem 4.1 are when $\mathcal{A}$ is "$NP \not\subseteq ZPP$" or "$NP \neq P$". For instance, suppose $\mathcal{A}$ is "$NP \not\subseteq ZPP$", and that result (R2) is shown. Then, as seen above, we will *provably* get a gap in the location of the randomized complexity of $NP$: either bounded by $poly(N)$, or being at least the explicit super-polynomial function $(h(N))^{\omega(1)}$.

In addition to Theorem 1.1, there are known partial results toward goals such as (R1) and (R2). For instance, Theorem 9.6 of [4] is as follows: "Suppose there is a (randomized) quasipolynomial-time algorithm $\mathcal{A}$ for Independent Set with performance guarantee $N/2^{\sqrt{c \log N}}$ on $N$-vertex graphs, then there is a randomized quasipolynomial-time algorithm $\mathcal{B}$ to color any $n$-vertex $k$-colorable graph with $O(n^\epsilon)$ colors, where $\epsilon = (20 \log k)/c$." Thus, good lower bounds for approximate graph coloring will lead to progress on showing results like (R1) and (R2).

### *4.2 An approach to NP vs. BPP and to NP vs. P*

We now consider application (b) of §1.

**Theorem 4.2** *Let $f$ be an arbitrary desirable function, and $\epsilon$ be any positive constant. Consider the lower-bound assertions* (LB1)*,* (LB2)*,* (LB3) *and* (LB4) *of Definition 1.1 for this pair $(f, \epsilon)$. We have the following implications:* (LB1) *implies $NP \neq P$,* (LB2) *implies $NP \neq P$,* (LB3) *implies $NP \not\subseteq BPP$, and* (LB4) *implies $NP \not\subseteq BPP$.*

**Proof:** We start by showing that (LB3) implies $NP \not\subseteq BPP$. In fact, letting $h$ be as in Theorem 4.1 and letting $\delta$ be any positive constant smaller than $\epsilon$, we show that (LB3) implies

$$NP \not\subseteq BPTIME[(h(N))^{1+\delta}];$$

recall that $h$ is a superpolynomial function. The main observation is that the reduction of Theorem 3.1(ii) runs *very fast.* Suppose we specialize the optimization problem of §3 to the problem of approximating clique in an $n$-vertex graph $G$ to within $n/f(n)$. In the notation of Theorem 3.1, let $\rho = n/f(n)$, $t = 1$, and $\kappa = \sqrt{1.5} - 1$. We randomly choose a set $S$ of $\lceil nt(1+\kappa)/\rho \rceil$ vertices of $G$ without replacement, and construct the subgraph $G[S]$ of $G$ that is induced by $S$. This subgraph construction takes at most

$$T_1(n) = O(n^{1+o(1)} f(n)) = n^{1+o(1)}$$

time, for the following reason. Note that $|S| = \Theta(f(n)) = n^{o(1)}$. Generating the set $S$ takes $n^{o(1)}$ time. Next, for each of the $\Theta(f(n))$ vertices in $S$, we need at most $n^{1+o(1)}$ time to examine its adjacency list, and to extract out its neighbors which lie in $S$.

Now suppose $NP \subseteq BPTIME[(h(N))^{1+\delta}]$, for some positive constant $\delta < \epsilon$; define $\epsilon' = (\delta + \epsilon)/2$. Then, by the standard reduction from the optimization version of clique to the decision version, there is a Monte Carlo algorithm for the (optimization version of) clique, which runs within time, say, $(h(N))^{1+\epsilon'}$ and succeeds with probability at least $1/2$. Thus, since $G[S]$ has input size at most $I = 3(f(n))^2$, there is a Monte Carlo algorithm to solve the maximum clique problem on $G[S]$, with running time at most $T_2(n) = (h(I))^{1+\epsilon'} = O(n^{1+\epsilon'})$ and success probability at least $1/2$. So, (4) shows that there is an algorithm with running time at most $T_1(n) + T_2(n) = O(T_2(n))$, which can approximate clique on $G$ to within $n/f(n)$ with probability at least $(1 -$

$e^{-\kappa^2/(2(1+\kappa))}) \times (1/2)$. This constant success probability can be boosted to 1/2 by repeating the algorithm a constant number of times. Thus, we will have a Monte Carlo $n/f(n)$-approximation algorithm for clique with running time $O(n^{1+\epsilon'})$, which would contradict (LB3).

Next, here is a sketch of how negative result "(LB4)" of §1 will imply $NP \not\subseteq BPP$. We will largely follow the proof that (LB3) implies $NP \not\subseteq BPP$: if the input graph is expressed in adjacency matrix form, we can implement the algorithm in that proof even faster, as follows. Instead of defining $S$ to be a random set of $\lceil nt(1+\kappa)/\rho \rceil = \lceil \sqrt{1.5} f(n) \rceil$ vertices of $G$ chosen without replacement, we choose these vertices independently and uniformly at random (i.e., *with* replacement). Since $f(n) = o(\sqrt{n})$, the probability that all these vertices are distinct is

$$\prod_{i=1}^{\lceil \sqrt{1.5} f(n) \rceil - 1} (1 - i/n) = \prod_{i=1}^{\lceil \sqrt{1.5} f(n) \rceil - 1} e^{-\Theta(i/n)} = e^{-\Theta((f(n))^2/n)} = 1 - o(1);$$

so, the analysis behind our above proof that (LB3) implies $NP \not\subseteq BPP$, is essentially unchanged. Moreover, since the elements of $S$ are chosen with replacement, this random choice of $S$ can be done in $n^{o(1)}$ time since $|S| = n^{o(1)}$. Also, since $G$ is expressed in adjacency matrix format, we can construct $G[S]$ in $O((f(n))^2) = n^{o(1)}$ time. We now use the above proof that (LB3) implies $NP \not\subseteq BPP$, to conclude that (LB4) implies $NP \not\subseteq BPP$. In contrapositive form, we get that $NP \subseteq BPP$ implies very fast clique approximations: e.g., for all $\delta > 0$ there exist $\epsilon > 0$ and $n_0$ such that for all graphs with $n \geq n_0$ vertices input in adjacency matrix form, clique can be approximated to within $n^{1-\epsilon}$ in $n^\delta$ time, by a Monte Carlo algorithm with probability at least 1/2 on the randomized unit-cost RAM.

Proving the claimed consequences of (LB1) and (LB2) is very similar, and we merely point out how the above proofs need to be modified. We use part (iii) of Theorem 3.1, instead of part (ii). Also, in the notation of Theorem 3.1(iii), we define each subset $A_i$ to have consecutive-numbered elements of $[n]$. If $G$ is input in adjacency list format, all the $G[A_i]$ can be constructed in $n^{2+o(1)}$ time; if $G$ is expressed in adjacency matrix format, all the $G[A_i]$ can be constructed in $n^{1+o(1)}$ time. ■

Thus, we see some complexity-theoretic consequences of certain (quite low) polynomial lower bounds for clique approximation. As mentioned in §1, work of [6] shows that for a certain constant $\lambda > 0$, clique cannot be approximated to within $n/EH(\lambda, n) = n^{1-\lambda/\sqrt{\log\log n}}$ by any Monte-Carlo polynomial-time algorithm with probability at least 1/2, unless $NP \subseteq BPTIME[2^{O(\log N (\log\log N)^{3/2})}]$. Thus, a lower bound as small as $n^{1+\epsilon}$ seems likely to hold for this problem.

Also, this result of [6] holds even when restricted to graph families with, e.g., $\Theta(n^2)$ edges. [This can be seen as follows. We may assume that the input $n$-vertex graph $G$ is nonempty; so its clique number is at least 2. Construct a new graph $G'$ by adding an $n/2 + n/2$ complete bipartite graph as a new connected component to $G$. $G'$ has $2n$ vertices and $\Theta(n^2)$ edges; the clique numbers of $G$ and $G'$ are the same. Thus, any polynomial-time $n/EH(\lambda, n)$–approximation of clique for $G'$, is also such an approximation for $G$.] This may be an approach worth exploring for the $P$ vs. $NP$ and $NP$ vs. $BPP$ questions, in some restricted models of computation.

Also, recall that the clique problem on a graph $G$ is identical to the MIS problem on $G$'s complement. We observe that Theorem 4.2 holds if we replace clique by MIS in the assertions (LB1)–(LB4). This observation is not interesting for the case where we seek an $n^{2+\epsilon}$ lower bound for an $n/f(n)$-approximation, since we can always construct a graph's complement in $O(n^2)$ time. However, this observation may be useful for the issue of proving an $n^{1+\epsilon}$ or $n^{\epsilon}$ lower bound.

### *4.3 Bootstrapping hardness results*

Recall the sample hardness results (R1) and (R2) defined in the beginning of §4. We now show how results such as (R1) and (R2) can be "bootstrapped" to yield new hardness results that have a smaller approximation guarantee and stronger lower bounds on their running times.

Theorem 4.1 basically followed by using the reduction of §3 with $\kappa = \sqrt{1.5} - 1$ and $t = 1$. The proof approach of Theorem 4.1, using Theorem 3.1 with $\kappa = \sqrt{1.5} - 1$, $\rho = n/f(n)$, and an arbitrary $t = t(n)$, leads to

**Theorem 4.3** *Let $f(\cdot)$ be an arbitrary desirable function. Suppose $n_0$, $t(\cdot)$ and $\Phi(\cdot)$ are such that for all $n \geq n_0$: (a) $1 \leq t(n) \leq n/(\sqrt{6}f(n))$, and (b) $\Phi(\sqrt{6}t(n)f(n)) \leq t(n)$. Define $D(z) \doteq \min\{y \in Z_+ : \ \sqrt{6}t(y)f(y) \geq z\}$. Then:*

*(i) Suppose (R1) is true for $f$ and for some assumption $\mathcal{A}$. Then, under assumption $\mathcal{A}$, clique cannot be approximated in $v$-vertex graphs to within $\Phi(v)$, in $DTIME[(D(v))^{O(1)}]$.*

*(ii) Suppose (R2) is true for $f$ and for some assumption $\mathcal{A}$. Then, under assumption $\mathcal{A}$, clique cannot be approximated in $v$-vertex graphs to within $\Phi(v)$, in $ZPTIME[(D(v))^{O(1)}]$.*

**Proof:** (i) Suppose (R1) and $\mathcal{A}$ hold, and that clique can be approximated in $v$-vertex graphs to within $\Phi(v)$, in $DTIME[(D(v))^{O(1)}]$. Now suppose we are given an $n$-vertex graph $G$ for which we want to approximate clique to within $n/f(n)$. We use Theorem 3.1(iii) to construct $O(n/(f(n)t(n)))$ graphs, each with at most $\sqrt{6}t(n)f(n)$ vertices; a $t(n)$-approximation – in particular, a $\Phi(\sqrt{6}t(n)f(n))$-approximation – of clique applied to each of these graphs, yields an $(n/f(n))$–approximation for clique in $G$. Let $v = \sqrt{6}t(n)f(n)$, and note that $D(v) \leq n$. Thus, if clique can be approximated in $v$-vertex graphs to within $\Phi(v)$ in $DTIME[(D(v))^{O(1)}]$ (i.e., in time $poly(n)$), then we can get an $(n/f(n))$–approximation for clique in $G$ in $poly(n)$ time, violating (R1).

The proof of (ii) is identical, except that we now need to use Theorem 3.1(ii). ∎

**Corollary 1** *In the notation of Theorem 1.1, the assumption $\mathcal{EHA}$ implies that: (a) Clique cannot be approximated to within $n^{1-\epsilon}$ in $ZPTIME[n^{O(\sqrt{\log\log n})}]$, for any fixed $\epsilon > 0$; and (b) Let $\nu > \gamma$ and $0 < b < 1/2$ be any constants. Then, clique cannot be approximated to within $n^{1-\nu/(\log\log n)^b}$ in $ZPTIME[n^{O((\log\log n)^{1/2-b})}]$.*

**Proof:** Recall Theorem 1.1; we just need to combine Theorem 4.3 (setting $f(n) = EH(\gamma, n)$) and Theorem 1.1 appropriately. Part (a) follows by using $t(n) = (\sqrt{6} \cdot EH(\gamma, n))^{(1-\epsilon)/\epsilon}$, $\Phi(x) = x^{1-\epsilon}$, and $D(z)$ of the form $z^{\Theta(\sqrt{\log\log z})}$. For part (b), we choose $t(n) = n^{c/(\log\log n)^{1/2-b}}$, $\Phi(x) = x^{1-\nu/(\log\log x)^b}$, and $D(z)$ of the form $z^{\Theta((\log\log z)^{1/2-b})}$, where $c > \gamma/\nu$ is an arbitrary constant. Let $u(n) = \sqrt{6} \cdot EH(\gamma, n)t(n)$; we need to show that $\Phi(u(n)) \leq t(n)$, i.e., that $\log(\Phi(u(n))) \leq \log(t(n))$. Since $\log\log(u(n)) \leq \log\log n$ and since

$$\log(u(n)) = \left(1 + \frac{\gamma + o(1)}{c(\log\log n)^b}\right) \cdot \log(t(n)),$$

it suffices to show that

$$\left(1 - \frac{\nu}{(\log\log n)^b}\right) \cdot \left(1 + \frac{\gamma + o(1)}{c(\log\log n)^b}\right) \leq 1,$$

which holds for all large enough $n$ since $c > \gamma/\nu$. ∎

### *4.4 From hardness to almost-everywhere hardness*

Our approach can also be used to show certain "hardness on average"-type results. We remark that "hardness on average" here does not necessarily mean that most instances are hard, but that there exist instances on which most "sub-instances" are hard. Suppose $f(n)$ is an arbitrary *desirable* function, and that $\mathcal{A}$ is some complexity-theoretic assumption. We will construct "almost-everywhere hardness" results from results of the form

**(R3)** "If $\mathcal{A}$ holds, then clique cannot be approximated to within $n/f(n)$ by any Monte Carlo polynomial-time algorithm with success probability at least 1/2."

We prove the following theorem:

**Theorem 4.4** *Suppose that (R3) holds for some desirable function $f$ and some complexity-theoretic assumption $\mathcal{A}$. Let $W(n)$ be any desirable integer-valued function such that $W(n) \geq af(n) \ln n$ for some fixed $a > 0$. Define $Y(z) \doteq \min\{y \in Z_+ : W(y) \geq z\}$; $Y$ is superpolynomial since $W$ is desirable. Then, there is* ***no*** *Monte Carlo Algorithm $\mathcal{M}$ for the clique problem such that:*

*(i) given any $v$-vertex graph $H$ for arbitrary $v$, $\mathcal{M}$ runs in time $(Y(v))^{O(1)}$, and*

*(ii) for each $n$-vertex graph $G$ for arbitrary $n$, and for at least an $n^{-a/4}$ fraction of the induced subgraphs $H$ of $G$ that have $W(n)$ vertices, $\mathcal{M}$, when given $H$ as input, finds a maximum clique in $H$ with probability at least* 1/2.

**Proof:** We specialize Theorem 3.1(ii) to the unweighted maximum clique problem. Suppose (R3) holds, but that there is indeed an algorithm $\mathcal{M}$ satisfying items (i) and (ii) in the statement of the theorem. Then, given any $n$-vertex graph $G$, we now present a Monte Carlo $poly(n)$-time that computes an $n/f(n)$-approximation to the maximum clique of $G$ with probability at least 1/2, contradicting (R3). This will prove the theorem.

Let $\rho = n/f(n)$; let *opt* be the clique number of $G$, and let $T$ denote an arbitrary maximum clique of $G$. As in our discussion in the proof of Theorem 3.1, we can assume that $opt > \rho$ without loss of generality. Choose a set $Z$ of $W(n)$ vertices of $G$ at random. Let $X$ be the number of chosen vertices of $T$; since $opt \geq n/f(n)$ and $W(n) \geq af(n) \ln n$, (3) shows for all large enough $n$ that

$$\mathbf{Pr}[X < opt/\rho] \leq n^{-a/3}. \tag{5}$$

Consider the subgraph $G[Z]$ of $G$ induced by $Z$. By item (ii) in the statement of Theorem 4.4, there is at least an $n^{-a/4}$ chance that $\mathcal{M}$ works correctly on $G[Z]$ with probability at least 1/2. Thus, by (5), if we run $\mathcal{M}$ on $G[Z]$, we will compute a clique in $G$ of size at least $opt/\rho$ with probability at least $(1/2) \cdot (n^{-a/4} - n^{-a/3}) \sim n^{-a/4}/2$. Further, by item (i) in the statement of Theorem 4.4, this will only take $(Y(W(n)))^{O(1)} = n^{O(1)}$ time. So, by repeating this basic algorithm a polynomial number of times, we can find a clique in $G$ of size $opt/\rho$ in $G$ with probability at least 1/2. ■

Thus, a hypothesis such as (R3) shows the following. Suppose we desire an algorithm that correctly finds the maximum clique on even a certain negligible fraction of some induced subgraphs, for every input graph. Then, any such algorithm needs time at least an explicit super-polynomial function of its input size. As mentioned in §1, (R3) is shown in [6] with $f(n) = EH(\lambda, n)$ for some positive constant $\lambda$, with $\mathcal{A}$ being the assumption "$NP \not\subseteq BPTIME[2^{O(\log N (\log\log N)^{3/2})}]$". We can plug these choices into Theorem 4.4 to see such hardness-almost-everywhere results (e.g., if $W(n) = poly(f(n))$, then the superpolynomial lower bound on the running time here is $(Y(n))^{\omega(1)} = n^{\omega(\sqrt{\log\log n})}$).

Furthermore, in the notation of Theorem 3.1, we have implicitly taken the approximation factor $t$ to be 1 above. (That is, item (ii) in the statement of Theorem 4.4 asks for an exact computation–not an approximation–of a maximum clique in $H$.) By choosing suitable other values for $t$, we can see that hypotheses such as (R3) imply that even *approximating* clique appropriately is "hard almost everywhere". We get the following result:

**Theorem 4.5** *Suppose that (R3) holds for some desirable function $f$ and some complexity-theoretic assumption $\mathcal{A}$. Suppose $a$ and $n_0$ are positive constants, and $t(\cdot)$ and $W(\cdot)$ are functions such that for all $n \geq n_0$:*

- $1 \leq t(n) \leq n/(af(n)\ln n)$*; and*
- $W(n)$ *is integer-valued and lies in the range* $[af(n)t(n)\ln n, n]$.

*Define $Y(z) \doteq \min\{y \in Z_+ : W(y) \geq z\}$. Then, there is **no** Monte Carlo Algorithm $\mathcal{M}$ for the clique problem such that:*

*(i) given any $v$-vertex graph $H$ for arbitrary $v$, $\mathcal{M}$ runs in time $(Y(v))^{O(1)}$, and*

*(ii) for each $n$-vertex graph $G$ for arbitrary $n$, and for at least an $n^{-a/4}$ fraction of the induced subgraphs $H$ of $G$ that have $W(n)$ vertices, $\mathcal{M}$, when given $H$ as input, approximates clique in $H$ to within $t(n)$ with probability at least $1/2$.*

**Proof Sketch:** The proof is very close to the proofs of Theorems 4.4 and 4.3; we merely give a sketch here. Suppose we desire to approximate clique in a given $n$-vertex graph $G$, to within $n/f(n)$. We can assume that *opt*, the maximum clique size in $G$, is at least $n/f(n)$. Choose a set $Z$ of $W(n)$ vertices of $G$ at random; as in the proof of Theorem 4.4, we can show that the maximum clique size in $G[Z]$ is at least $opt \cdot t(n)f(n)/n$, with probability at least $1 - n^{-a/3}$. If the clique size in $G[Z]$ is indeed this large, then we will be done if we can approximate clique in $G[Z]$ to within $t(n)$. The rest of the proof is as for Theorem 4.4. ∎

## 5 Time-approximation tradeoffs

A concrete measure of running time when working with a general $(n, g, \vec{w})$-OPT problem is the query complexity: the number of queries to the oracle for $g$ that we need to make, for a given problem. Earlier work of [11] shows an approach that yields a certain query complexity for approximating $(n, g, \vec{w})$-OPT problems to within a given bound; Theorem 5.1 yields the same result (and uses the same type of algorithm). For completeness, we give the short proof of Theorem 5.1. This query complexity is then shown to be essentially best-possible, by Theorem 5.2.

Let us call an $(n, g, \vec{w})$-OPT problem *unweighted*, if $\vec{w}$ is the $n$-bit vector $\vec{1}$ of all ones. (To avoid extra notation, we will always assume $\vec{1}$ to be an $n$-bit vector.)

We start by recalling a known fact:

**Fact 2** *Suppose* $0 \leq a \leq b \leq c$ *are integers. Then: (a)* $\sum_{i=0}^{a} \binom{b}{i} \leq (be/a)^a$. *(b)* $\binom{c-a}{b-a} / \binom{c}{b} \leq (b/c)^a$.

**Proof:** (a) $\sum_{i=0}^{a} \binom{b}{i} \leq \sum_{i=0}^{a} (b/a)^a (a/b)^i \binom{b}{i} \leq (b/a)^a \cdot \sum_{i=0}^{b} \binom{b}{i} (a/b)^i = (b/a)^a \cdot (1 + a/b)^b \leq (be/a)^a$.

(b) $\binom{c-a}{b-a} / \binom{c}{b} = (\prod_{i=0}^{a-1} (b-i)/(c-i)) \leq (b/c)^a$. ■

**Theorem 5.1** *(i) General* $(n, g, \vec{w})$*-OPT problems can be* $\rho$*-approximated deterministically using* $2^{O(\log n + n/\rho)}$ *queries to the oracle for* $g$*, for any* $\rho \in [1, n]$*, in* $2^{O(\log n + n/\rho)}$ *time.*

*(ii) Suppose we are given, for an arbitrary* $(n, g, \vec{1})$*-OPT problem, that the optimal solution value is at most* $\ell$*. Then for any* $\rho \in [1, n]$*, we can deterministically find a* $\rho$*-approximation making* $2^{O(\log n + \lceil \ell/\rho \rceil \cdot \log(2n/\ell))}$ *queries to the oracle for* $g$*, and in* $2^{O(\log n + \lceil \ell/\rho \rceil \cdot \log(2n/\ell))}$ *time.*

**Proof:** We start with the reduction of Theorem 3.1(iii), for both parts (i) and (ii). For both parts, we do this reduction with $\kappa = \sqrt{1.5} - 1$, say, and $t = 1$; so, we basically reduce the problem to $\lfloor \rho \rfloor$ smaller instances, each of size at most $M \doteq \lceil n / \lfloor \rho \rfloor \rceil$. We now describe how to proceed further for parts (i) and (ii).

An optimal solution to an $(n, g, \vec{w}_{A_i})$-OPT instance can trivially be found using $2^{|A_i|}$ queries to the oracle for $g$, in $2^{O(\log n + |A_i|)}$ time. The claim of part (i) follows from this. For part (ii), having done the reduction to problems of size $M$, we only need to find a solution of value at most $\lceil \ell/\rho \rceil$; simple enumeration thus yields a total query complexity of at most

$$O\left(\rho \sum_{i=0}^{\lceil \ell/\rho \rceil} \binom{M}{i}\right) = 2^{O(\log n + \lceil \ell/\rho \rceil \cdot \log(2n/\ell))}.$$

The time complexity follows similarly. ■

**Remark.** We use the notation "$\log(2n/\ell)$" instead of "$\log(n/\ell)$" in Theorem 5.1(ii), to handle the case where $\ell = n(1 - o(1))$. In this case, $\log(n/\ell) = o(1)$, while we really want a term that is $\Omega(1)$. If $n/\ell = 1+\Omega(1)$, then $\log(2n/\ell)$ and $\log(n/\ell)$ are within a multiplicative constant of each other. Also, for Theorem 5.1(ii), an algorithm using $\sum_{i=0}^{\lceil \ell/\rho \rceil} \binom{n}{i} = 2^{O(\log(2n/\lceil \ell/\rho \rceil) \cdot \lceil \ell/\rho \rceil)}$ queries is obvious: just enumerate all $n$-bit vectors of Hweight at most $\lceil \ell/\rho \rceil$. Theorem 5.1(ii) shows that we can develop algorithms that are faster than this obvious one.

Also, given the recent interest in massive data sets and, e.g., clique problems in very large graphs [1], we remark that our observations of §4.2 imply certain types of very fast clique-approximation results. For instance, those observations show that if a graph is expressed in adjacency matrix format, then we can probabilistically approximate clique to within $n/(\epsilon \log n)$ in $n^{c\epsilon}$ time, where $c$ is an absolute constant and $\epsilon$ is any given positive constant.

Another interesting family of problems captured by the $(n, g, \vec{w})$-OPT model are the packing integer programs with all variables constrained to lie in $\{0, 1\}$: given a system of linear inequalities $A\vec{x} \leq \vec{b}$ where all entries of $A$ and $\vec{b}$ are non-negative, we wish to find some $\vec{x} \in \{0, 1\}^n$ which maximizes $\vec{w} \cdot \vec{x}$ subject to $A\vec{x} \leq \vec{b}$. This generalizes, e.g., the weighted MIS, (multiple) knapsack, and hypergraph matching problems. Letting $m$ denote the number of constraints in the system "$A\vec{x} \leq \vec{b}$", the worst-case approximation guarantee for arbitrary packing integer programs is about $O(\sqrt{m})$ [15,18]. Note that this is weak if $m$ is large: even if, say, $m \geq \Omega(n^2)$. Theorem 5.1(i) shows that we start getting benefits if $m$ is significantly larger than $n$. Consider families of packing integer programs where, e.g., $m = 2^{\Omega(n^{3/4})}$. For such families, we get $\rho = O(n^{1/4})$-approximation algorithms that run in time polynomial in the length of the input (which is $\Omega(m + n)$).

Next, Theorem 5.2 shows that the query complexities of both parts of Theorem 5.1 are optimal up to the constant factor in the exponent, if $g$ is an arbitrary monotone decreasing function available through its oracle. (To see why Theorem 5.2 implies that part (i) of Theorem 5.1 is essentially optimal, we can substitute, e.g., $\ell = n/2$ in Theorem 5.2.)

**Theorem 5.2** *Let $n, \ell, \rho \geq 1$ be arbitrary; define $\ell' = \lfloor \min\{\ell, n/2\} \rfloor$. Consider the class of $(n, g, \vec{1})$-OPT problems where we are also given that the optimal solution value is at most $\ell$. Any randomized algorithm that produces a $\rho$-approximation with at least a positive constant probability for this class of problems, should make $\Omega(n + (n/\ell')^{\lceil \ell'/\rho \rceil})$ queries to $g$'s oracle.*

**Proof:** Suppose the desired success probability of the randomized algorithm is some constant $\alpha > 0$. By Yao's theorem [19], it will suffice to present a randomized construction of an $(n, g, \vec{1})$-OPT problem, for which any *deterministic* algorithm needs to make $\Omega(n + (n/\ell')^{\lceil \ell'/\rho \rceil})$ queries to have a success probability of at least $\alpha$. (This probability is only w.r.t. the random choice of the problem.)

Suppose $n \geq (n/\ell')^{\lceil \ell'/\rho \rceil}$. Choose a random vector $\vec{v}$ of Hweight 1, and define $g(\vec{v}) = g(0, 0, \ldots, 0) = 1$; $g(\vec{w}) = 0$ for all other $\vec{w}$. In this case, it is immediate that for any $\rho \geq 1$, any deterministic algorithm needs at least $\Omega(n)$ oracle queries to achieve a $\rho$-approximation with probability $\alpha$.

We move on to showing an $\Omega((n/\ell')^{\lceil \ell'/\rho \rceil})$ lower bound on the query complexity (in the case where $n < (n/\ell')^{\lceil \ell'/\rho \rceil}$): the random construction of a Boolean function $g$ that we now use is as follows. Let $a = \lceil \ell'/\rho \rceil$. We choose an $n$-bit vector $\vec{r}$ of Hweight $\ell'$ at random. All $n$-bit vectors $\vec{s}$ such that: (i) $\vec{s} \preceq \vec{r}$, or (ii) the Hweight of $\vec{s}$ is at most $a - 1$, have $g(\vec{s}) = 1$; all other $\vec{s}$ have $g(\vec{s}) = 0$. It is easy to check that this random $g$ is monotone decreasing. Recalling that we have an $(n, g, \vec{1})$-OPT problem, we see that the optimal solution value is $\ell' \leq \ell$, with $\vec{r}$ being the unique optimal solution; a $\rho$-approximation must thus produce some $\vec{s} \preceq \vec{r}$ whose Hweight is at least $a$.

Fix any deterministic $\rho$-approximation algorithm $\mathcal{DA}$ in this setting; we start with some observations about the queries made by $\mathcal{DA}$. Clearly, $\mathcal{DA}$ need not query the value of $g(\vec{s})$ for any $\vec{s}$ of Hweight at most $a-1$, since $g(\vec{s})$ is known to be 1 in this case. We also argue that if $\mathcal{DA}$ queries the value of $g(\vec{s})$ for some $\vec{s}$ of Hweight more than $a$ at some point in its execution, it can do no worse by choosing an arbitrary $\vec{t} \preceq \vec{s}$ of Hweight exactly $a$, and querying the value of $g(\vec{t})$ in place of $g(\vec{s})$. To see this, note that if $g(\vec{s}) = 1$, then querying $g(\vec{s})$ or $g(\vec{t})$ will make $\mathcal{DA}$ successfully terminate after this query. On the other hand, if $g(\vec{s}) = 0$, then by querying $g(\vec{t})$, $\mathcal{DA}$ can only get more information. (If $g(\vec{t}) = 1$, $\mathcal{DA}$ will halt successfully; if $g(\vec{t}) = 0$, $\mathcal{DA}$ can infer that $g(\vec{s}) = 0$.) So we may assume without loss of generality that $\mathcal{DA}$ only queries the value of $g(\vec{s})$ for vectors $\vec{s}$ of Hweight precisely $a$.

For a vector $\vec{s}$ of Hweight $a$, define $S(\vec{s})$ to be the set of vectors $\vec{t}$ of Hweight $\ell'$ such that $\vec{s} \preceq \vec{t}$. Also define $U$ to be the set of all $n$-bit vectors of Hweight exactly $\ell'$. Let $Q_i$ denote the $i$th $n$-bit vector whose $g(\cdot)$ value is queried by $\mathcal{DA}$, and let $\mathcal{E}_i$ be the event that $g(Q_i) = 0$. $Q_i$ is a random variable, since the choice of $g$ is random. Fix any $i \geq 0$, and any sequence of vectors $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{i+1}$, each of Hweight $a$. We argue that

$$\mathbf{Pr}[\vec{r} \notin S(\vec{q}_{i+1}) | \bigwedge_{j=1}^{i} ((Q_j = \vec{q}_j) \wedge \mathcal{E}_j)] = \frac{|U - \bigcup_{j=1}^{i+1} S(\vec{q}_j)|}{|U - \bigcup_{j=1}^{i} S(\vec{q}_j)|}.$$

This is because, conditional on "$\bigwedge_{j=1}^{i}((Q_j = \vec{q}_j) \wedge \mathcal{E}_j)$", $\vec{r}$ is uniformly distributed in the set $U - \bigcup_{j=1}^{i} S(\vec{q}_j)$. We have $|S(\vec{t})| = z_1 \doteq \binom{n-a}{\ell'-a}$ for each vector $\vec{t}$ of Hweight $a$; thus, defining $z_2 \doteq \binom{n}{\ell'}$, we get

$$\begin{aligned}
\mathbf{Pr}[\vec{r} \notin S(\vec{q}_{i+1}) | \bigwedge_{j=1}^{i} ((Q_j = \vec{q}_j) \wedge \mathcal{E}_j)] &\geq 1 - \frac{z_1}{|U - \bigcup_{j=1}^{i} S(\vec{q}_j)|} \\
&\geq 1 - \frac{z_1}{z_2 - iz_1} \\
&= \frac{z_2 - (i+1)z_1}{z_2 - iz_1}.
\end{aligned}$$

Since this is true for all choices of $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{i+1}$, we get

$$\mathbf{Pr}[\mathcal{E}_{i+1} | \bigwedge_{j=1}^{i} \mathcal{E}_j] \geq \frac{z_2 - (i+1)z_1}{z_2 - iz_1};$$

Bayes' theorem and a telescoping product yield

$$\mathbf{Pr}[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \cdots \wedge \mathcal{E}_i] \geq 1 - iz_1/z_2.$$

Now, $z_1/z_2 \leq (\ell'/n)^{\lceil \ell'/\rho \rceil})$ by Fact 2(b). Hence, for $\mathcal{DA}$ to have at least a positive constant success probability, the number of queries should be at least $\Omega(z_2/z_1) = \Omega((n/\ell')^{\lceil \ell'/\rho \rceil})$. ∎

## 6 Conclusions

We have shown new benefits of mapping out the terrain of "$n^{1-o(1)}$ approximability" for the Maximum Clique or related maximization problems. Our approach makes elementary use of (random) sampling, and basically exploits the property that suitably chosen samples will have approximately the expected number of elements from some fixed optimal solution. However, especially for instances that have a "large" number of (near-)optimal solutions, we may be able to do better: with some reasonable probability, our sample may get *many more elements than expected* from some optimal solution. Could this idea be useful in some settings? For instance, consider the random graph model $G(n, 1/2)$, where we take $n$ labeled vertices and put an edge between every pair of distinct vertices, independently with probability 1/2. It is well-known that with high probability, the maximum clique size here is $(2 - o(1)) \log n$. Thus, a Monte Carlo $2 \log n$–approximation for clique is trivial for this model: just output a one-vertex clique. However, what if we want a *Las Vegas* $\rho$–approximation, i.e., an algorithm that always produces a $\rho$-approximation, and has polynomial expected running time? (This expectation is taken over the random choice of the graph, as well as the internal coin flips of the algorithm if it is randomized.) In this case, the current-best value of $\rho$ is $n^{1/2-o(1)}$ [14]. Note that in $G(n, 1/2)$, we expect a large number of near-optimal solutions: for instance, the expected number of $(\log n)$-sized cliques is $n^{\Theta(\log n)}$. It would be interesting to see if the above remark on the sample holding many more elements than expected, would be applicable in such settings.

Our results show that: (i) if clique is $NP$-hard (under polynomial-time reductions) to approximate within $n/polylog(n)$, then either $NP \subseteq BPP$ or $NP$ requires almost exponential time; and (ii) if clique is unconditionally shown to be inapproximable to within $n/polylog(n)$, then $NP$ requires almost-exponential time. On the contrapositive, evidence of the hardness of proving such results on $NP$, will show the hardness of proving sufficiently good strong inapproximability results for clique. We have also shown an approach to issues such as $NP$ vs. $P$: that certain very low-degree *polynomial* lower bounds for some clique/MIS approximation problems (which seem likely to need super-polynomial time) suffice to show results such as $NP \not\subseteq BPP$ and $NP \neq P$. Can the recent improvements/proof methods in time-space tradeoffs (see, e.g., [9]) be put to use in this context for some restricted models of computation?

An interesting direction would be to strengthen Theorem 1.1. Recall that our results show the following: "If, for some constant $\psi > 0$, clique cannot be approximated to within $n/EH(\psi, n)$ by any $ZPP$ algorithm, then the randomized complexity of $NP$ is $N^{\omega(\sqrt{\log \log N})}$". Thus, if the hypothesis $\mathcal{EHA}$ of Theorem 1.1 can be weakened to

$$\mathcal{EHA}' \equiv (NP \not\subseteq ZPTIME[N^{O(\sqrt{\log \log N})}]),$$

then we will have that $\mathcal{EHA}'$ is *equivalent* to the existence of a constant $\psi > 0$ such that clique cannot be approximated to within $n/EH(\psi, n)$ by any $ZPP$ algorithm. Such equivalences, if true and provable, would be of much interest. A similar point can be made for any strong inapproximability result for clique: it is easy to observe that we can treat any strong inapproximability result for clique in a manner similar to the way we treat/utilize Theorem 1.1

in this work. For instance, it has recently been shown in [13] that for a certain constant $\gamma > 0$, clique cannot be approximated to within $n/2^{(\log n)^{1-\gamma}}$ unless $NP \subseteq ZPTIME[2^{(\log N)^{O(1)}}]$. The partial converse for this result that follows from our work is: *if clique cannot be approximated to within $n/2^{(\log n)^{1-\gamma}}$, then the randomized complexity of $NP$ is at least $N^{(\log N)^{\Omega(1)}}$.*

Finally, although we have presented a near-optimal tradeoff between approximation ratio and number of queries for $(n, g, \vec{w})$-OPT problems in general, it will be interesting to develop better tradeoffs for specific problems such as Maximum Clique. One possible approach in this context is to do a "size reduction" via sampling, and to then apply some problem-specific method on the size-reduced instance. Another direction in this context is to study interesting sub-families of the monotone decreasing functions, and to prove tradeoffs between approximation ratio and query complexity for such families.

**Acknowledgements.** I thank Lars Engebretsen and Jonas Holmerin for clarifying issues about their results. My thanks also to Avrim Blum, Johan Håstad, Seffi Naor, D. Sivakumar, Francis Zane and David Zuckerman for helpful discussions. Finally, I thank the three referees for their detailed and helpful suggestions.

## References

[1] J. Abello, P. Pardalos, and M. Resende. On maximum clique problems in very large graphs. In *External Memory Algorithms* (J. Abello and J. Vitter, Editors), American Mathematical Society, Series in Discrete Mathematics and Theoretical Computer Science, Volume 50, pages 119–130, 1999.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof Verification and the Hardness of Approximation Problems. *Journal of the ACM*, 45:501–555, 1998.

[3] S. Arora and S. Safra. Probabilistic Checking of Proofs: A New Characterization of $NP$. *Journal of the ACM*, 45:70–122, 1998.

[4] A. Blum. Algorithms for approximate graph coloring. Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, MIT, 1991.

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

[6] L. Engebretsen and J. Holmerin. Clique is hard to approximate within $n^{1-o(1)}$. In *Proc. International Colloquium on Automata, Languages and Programming*, pages 2–12, 2000.

[7] U. Feige. Randomized graph products, chromatic numbers, and the Lovász $\vartheta$-function. *Combinatorica*, 17:79–90, 1997.

[8] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43:268–292, 1996.

[9] L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60:337–353, 2000.

[10] M. M. Halldórsson. Approximations of independent sets in graphs. In *Proc. APPROX '98 Conference*, Springer-Verlag Lecture Notes in Computer Science, Vol. 1444, pages 1–13, 1998.

[11] M. M. Halldórsson. Approximations of weighted independent set and hereditary subset problems. *Journal of Graph Algorithms and Applications*, 4:1–16, 2000.

[12] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[13] S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 600–609, 2001.

[14] M. Krivelevich and V. Vu. Approximating the independence number and the chromatic number in expected polynomial time. In *Proc. International Colloquium on Automata, Languages and Programming*, pages 13–24, 2000.

[15] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[16] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proc. ACM Symposium on Theory of Computing*, pages 191–199, 2000.

[17] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8:223–250, 1995.

[18] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM J. Comput.*, 29:648–670, 1999.

[19] A. C.-C. Yao. Probabilistic computations: towards a unified measure of complexity. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.