On coarse and fine approximate counting of t-cliques

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science, Rehovot, Israel.
oded.goldreich@weizmann.ac.il

September 21, 2023

Abstract

For any fixed t, we present two fine-grained reductions of the problem of approximately counting the number of t-cliques in a graph to the problem of detecting a t-clique in a graph. One of our reductions is slightly better than the prior reduction of Dell, Lapinskas, and Meeks (*SODA20*) and its improvement by Bhattacharya, Bishnu, Ghosh, and Mishra (*STACS22*). More importantly, we provide alternative presentations of their reductions, which we believe to be conceptually simpler.

The pivot of the foregoing works is the notion of *coarse* approximate counting; for example, think of approximating the number of t-cliques in n-vertex graphs up-to a $O(\log n)^{O(t)}$ factor. While it is easy to reduce *fine* (i.e., $1+\epsilon$ factor) approximate counting of solutions to NP-complete search problems to their *coarse versions* (ditto for natural problems in P such as perfect matching), these simple reductions fail in the context of fine grained complexity. One of the contributions of Dell *et al.* is providing a fine-grained reduction of standard (i.e., fine) approximate counting of t-cliques to coarsely counting them. We survey this reduction, and also provide an alternative one. The alternative (alas inferior) reduction composes a reduction of uniform generation of t-cliques to *coarsely*-approximate counting them with the standard reduction of (fine) approximate counting to uniform generation.

In addition, we survey the coarse approximate counter of t-cliques of Bhattacharya *et al.*, and improve its performance by a small twist.

Given that I am not an expert on fine-grained complexity and that this memo is mostly of pedagogical value, I allowed myself a non-conventional structure and contents. In particular, I do not start with an introduction to the wider context and do not provide a scholarly review of prior works. My impression is that the most relevant prior works are [4, 2, 5, 3], and that the interested reader may find an adequate scholarly account there.

The nature of this memo. As hinted in the abstract, this memo is mainly a survey of some technical ideas. Specifically, it provide an alternative presentation of results that appeared in [5, 3]. In particular:

- Algorithm 3.1, which reduces coarse approximate counting of t-cliques to detecting the existence of t-cliques, is due to [3].
- Theorem 5.2, which directly reduces fine approximate counting of t-cliques to coarse approximate counting of t-cliques, is essentially due to [5].
- The reduction (presented in Section 2) of *auxiliary problems regarding t-cliques that fit a given prefix* to problems regarding t-cliques (proper) is due to [5].

Our own technical contributions are confined to improving the performance of Algorithm 3.1 (see Theorem 3.5) and to directly reducing uniform generation of t-cliques to coarse approximate counting of t-cliques (see Theorem 4.3). The latter reduction is inferior to the indirect (and more complex) reduction that is given in [5], which builds on Theorem 5.2.

Applicability to hyper-edges in t-uniform hyper-graphs. Our presentation refers to t-cliques in graphs, but parts of it are quite generic and apply to any search problem (see, e.g., Theorems 4.3 and 5.2). We stress that the input graph is given explicitly to all algorithms and reductions that we discuss. In contrast, the aforementioned results of [5, 3] refer to a model of t-uniform hyper-graphs in which the input is represented by an "independence" (resp., "colored independence") oracle that answer queries of the type "does set S contain a hyper-edge?" (resp., "does the t-partition $(S_1, ..., S_t)$ of S contain a hyper-edges with a single vertex in reach part?").[1] With the exception of the standard reduction from the general case to the t-partite case, our entire presentation applies to the context of t-uniform hyper-graphs. Hence, Theorems 4.3 and 5.2 are applicable to general t-uniform hyper-graphs, but Theorem 3.5 is applicable to t-uniform *t-partite* hyper-graphs only.

[1] The foregoing model, which was introduced and generalized in [1, 4, 5], is aimed to address the fact that an explicit representation of the hyper-graph trivializes the problem of counting hyper-edges, whereas representing the n-vertex hyper-graph by a "multi-edge oracle" (i.e., $e : \binom{[n]}{t} \to \{0,1\}$ such that $e(v_1, ..., v_t) = 1$ if and only if $\{v_1, ..., v_t\}$ is a hyper-edge) eliminates all non-trivial algorithms. Furthermore, the foregoing model is tailored for the presentation of reductions from counting problems to decision problems, alas this model restricts these reductions to querying sub-instances of the given instance (i.e., induced subgraphs of the input graph). We stress that, with the exception of the standard reduction from the general case to the t-partite case, the reductions presented in this memo can be stated within this (restricted) framework.

Contents

1 Overview

This memo is pivoted at the notion of a coarse approximate counting of solutions to NP-search problems with a focus on fine-grained complexity and problems such as t-Clique. By coarse approximation we means approximation up-to large, but non-trivial factors. Specifically, we shall consider the problem of approximating the number of t-cliques in n-vertex graphs up-to a $O(\log n)^{O(t)}$ factor. This stands in contrast to the standard notion of approximation, where on input a proximity parameter $\epsilon > 0$, one seeks approximation up-to a factor of $1 + \epsilon$. We call the latter approximation fine.

In the standard context of complexity theory it is easy to reduce fine approximate counting to coarse approximate counting. This is done by taking Cartesian products, while noting that the number of solutions to a instance that consists of k copies of the original instance equals the k^{th} power of the number of solutions to the original instance. Hence, an F-factor approximation of the number of solutions to the "duplicated" instance yields a $F^{1/k}$-factor approximation of the number of solutions to the original instance. Needless to say, this reduction cannot be applied in the context of t-Clique and similar fine-grained complexity problems.

In light of the above, it may be that it is easier to obtain coarse approximations for the number of t-cliques than to obtain fine approximations to that number. This seems to be the case, at least slightly and empirically, since Dell *et al.* [5] first designed a coarse approximation algorithm for counting the number of t-cliques, and then used it to get a fine approximation (via a reduction) that is somewhat slower. The issues at hand are thus the following:

1. Obtaining a *coarse* approximate counter for t-cliques.
2. Reducing *fine* approximation of counting t-cliques to a corresponding *coarse* approximation.

We address these two issues in the following two subsections.

Applicability to hyper-edges in t-uniform hyper-graphs. We stress that although our presentation proceeds in terms of t-cliques, it can be adapted to the framework of [1, 4, 5] that refers to t-uniform hyper-graphs and to oracle calls that answer queries regarding induced substructures of the input hyper-graph. Note, however, that the adaptation of one of our reductions (i.e., reducing coarse approximate counting to detection) works for t-partite hyper-graphs only. The source of the problem is that the standard reduction from the general case to the t-partite case cannot be implemented in the foregoing framework.

1.1 Obtaining a coarse approximate counter for t-cliques

Coarse approximate counters of t-cliques in n-vertex graphs were presented in [5] and in [3]. We provide a more conceptual presentation of the procedure that underlies the proof of [3, Thm. 1.3], and improve upon it by introducing a twist. The main improvement is in the approximation factor; specifically, reducing it from $O(\log^{2t-2} n)$ to $O(\log^{t-1} n)$. Using the reduction of [5], we obtain an improvement on the complexity of approximately counting t-cliques; specifically,

Theorem 1.1 (reducing approximate counting t-cliques to detecting them): *There exists an almost linear-time procedure that, on input an n-vertex graph and slackness parameter $\epsilon > 0$, approximates the number of t-cliques in the graph up-to a factor of $1+\epsilon$, by making $O(1/\epsilon^2) \cdot \widetilde{O}(\log n)^{2t+3}$ queries to an oracle for deciding the existence of t-cliques in n-vertex graphs.*

This improves over the query complexity of [3, Thm. 1.4], which is $O(\epsilon^{-2} \cdot (\log n)^{3t+5})$.

Our perspective on the problem of (coarsely) approximating the number of t-cliques is rooted in the abstract idea that underlies the reduction of approximate counting NP-witnesses to deciding their existence. Specifically, in order to verify that the number of t-cliques in a graph is at least m, we apply a "random sieve" of density $1/m$ to the set of t-cliques and check whether the resulting graph had a t-clique. The question at hand is how to implement a random sieve, given that hashing (which is the method of choice in the context of NP) does not seem adequate in the current setting. Nevertheless, an appealing and straightforward way of implementing a random sieve in the current context consists of selecting each vertex with probability $p = (1/m)^{1/t}$, and considering the induced subgraph.

Unfortunately, this straightforward implantation does not work. While each specific t-clique passes this random sieve with probability $1/m$, these choices are not independent enough, and the dependency leads to our failure. Focusing (w.l.o.g) on the case of t-partite graphs, we note that it not necessarily the case that there are $\Omega(m^{1/t})$ vertices of the first part such that each of them appears in $\Omega(m^{(t-1)/t})$ cliques. However, for some $i \in [\log_2 n]$, there exists $\Omega(2^i)$ vertices in the first part such that each of them appears in $\Omega(2^{-i} \cdot m/\log n)$ cliques. This suggests trying all possible choices of $i_1, ..., i_t \in [\log_2 n]$, and randomly sieving the vertices of part j, with probability 2^{-i_j}.

The foregoing suggestion is implemented and analyzed in one way in [3], and we implement and analyze it a bit differently. In any case, it is evident that this approach, which tries all sequences $(i_1, ..., i_t) \in [log_n]^t$ while benefiting only from one of them, comes with an overhead of $O(\log n)^t$ in its query complexity and $O(\log n)^{t-1}$ its approximation factor. We meet these intuitive bounds. Specifically, we get

Theorem 1.2 (reducing coarse approximate counting t-cliques to detection): *There exists an almost linear-time procedure that, on input an n-vertex graph, approximates the number of t-cliques in the graph up-to a factor of $O(\log n)^{t-1}$ by making $O(1/\epsilon^2) \cdot \widetilde{O}(\log n)^t$ queries to an oracle for deciding the existence of t-cliques in n-vertex graphs.*

This improves over [3, Thm. 1.3], which achieves an approximation factor of $O(\log^{2t-2} n)$ and makes $O(\epsilon^{-2} \cdot (\log n)^{t+2})$ queries.

1.2 Reducing fine approximation to a coarse approximation

We present two different reductions of fine approximate counting of t-cliques to coarse approximate counting of t-cliques. The first reduction goes through the problem of uniform generation of t-cliques; it is simpler but yields a weaker time bound. The second reduction is a direct one and is due to [5].

Both reductions refer to the prefixes of possible t-cliques (in an input n-vertex graph). Associating the vertex set of the input graph with $[n] \equiv \{0,1\}^{\log_2 n}$, a prefix of $(v_1, ..., v_t) \in [n]^t$ has the form $(v_1, ..., v_{t-t'}, \alpha)$, where α is a string of length smaller than $\log_2 n$ and $t' \in \{0, 1, ..., t\}$. As observed in [5], the set of t-cliques (in the original t-partite graph) that fit the latter prefix corresponds to the set of t'-cliques in the t'-partite subgraph induced by the vertices that neighbor all v_i's (for $i \in [t - t']$) such that the first part of the t'-partite subgraph contains only vertices that have prefix α. Hence, problems regarding this auxiliary problems (i.e., problems regarding t-cliques that fit a given prefix) are reduced to corresponding problems regarding t'-cliques, which in turn are easily reducible to corresponding problems regarding t-cliques.

1.2.1 The first reduction: going through uniform generation

This reduction is the main conceptual contribution of the current memo. Here we take to the extreme the known fact that the *standard reduction of uniform generation of solutions* (for NP-search problems) *to approximately counting solutions* is insensitive to the preciseness of the approximation. This fact is typically observed with respect to fine approximate counters, and it holds provided that the slackness parameter decreases at least linearly with the length of solutions; that is, when using a $(1+(1/5\ell))$-factor approximation, where ℓ denotes the length of solutions (see, e.g., [6, Sec. 6.2.4.1] or the overview in Section 4.1).

We observe that using a coarse approximate counter still yields a meaningful (alas weak) notion of uniform generation. Specifically, the uniform generation procedure will output a solution with probability that is inversely related to the coarseness of the approximation. Even more specifically, using an F-factor approximation of the number of ℓ-bit long solutions, for every $m \in [\ell]$, we can obtain a procedure that makes $m \cdot \exp(\ell/m)$ oracle calls (to the coarse approximator) and produces output with probability $F^{-(m+1)}$. This procedure, generalizes the standard one, which uses $m = \ell$, by using m iterations and extending the current prefix by ℓ/m bits in each iteration. In the special case of t-cliques in n-vertex graphs, it holds that $\ell = t \cdot \log_2 n$ and $F = O(\log n)^{O(t)}$, and using $m = \sqrt{\ell \log F}$ we get a *uniform generator that makes* $M \stackrel{\text{def}}{=} \exp(\widetilde{O}(\sqrt{\log n})) = n^{o(1)}$ *calls to the coarse approximator and produces output with probability* $1/M$.

The next observation is that it is easy to amplify the success probability of uniform generation (i.e., the probability that it produced output) by repetitions. Hence, we get a standard uniform generator by repeating the foregoing weak one for M times. Using the known reduction of (fine) approximate counting to uniform generation, we obtained a fine approximate counter that makes $O(M^2/\epsilon^2) = \exp(\widetilde{O}(\sqrt{\log n}))/\epsilon^2 = n^{o(1)}/\epsilon^2$ calls to the coarse approximator.

1.2.2 The second reduction: the direct route

This reduction is due to [5], although our presentation of it is somewhat different. The reduction proceeds in iterations such that, in each iteration, we hold a (partial) list L of prefixes (of potential t-cliques) of corresponding length. For each prefix in the list (i.e., each $y' \in L$), we have a multiplier (i.e., $m_{y'}$) such that the linear combination determined by these multipliers of the number of t-cliques that correspond to the prefixes in L yields a fine approximation of the number of t-cliques in the original graph. That is, if $c_{y'}$ denotes the number of t-cliques that fit the prefix y' and $m_{y'}$ is the corresponding multiplier, then $\sum_{y' \in L} m_{y'} \cdot c_{y'}/|L|$ is a fine approximation of the number of t-cliques in the graph.

In the current iteration, for each one-bit extension $y'\sigma$ of a prefix y' in L, we obtain coarse approximations for $c_{y'\sigma}$. Denoting these approximations by $\widetilde{c}_{y'\sigma}$, we let $a_{y'\sigma} = m_{y'} \cdot \widetilde{c}_{y'\sigma}$ and $p_{y'\sigma} = a_{y'\sigma}/a$, where $a = \sum_{(y',\sigma) \in L \times \{0,1\}} a_{y'\sigma}$. We then generate a random sample of $|L|$ strings by picking $y'\sigma$ with probability $p_{y'\sigma}$ and let $m_{y'\sigma} \stackrel{\text{def}}{=} m_{y'}/p_{y'\sigma}$ be the corresponding multiplier. (The first iteration starts with $|L|$ copies of the empty prefix, which are all associated with the multiplier 1.)

The foregoing sample, which is generated in accordance with the "importance sampling" paradigm, is analyzed by considering a random variable X that is assigned the value $m_{y'\sigma} \cdot c_{y'\sigma} = (m_{y'}/p_{y'\sigma}) \cdot$

$c_{y'\sigma}$ with probability $p_{y'\sigma}$. Then,

$$\mathsf{E}[X] \;=\; \sum_{(y',\sigma)\in L\times\{0,1\}} p_{y'\sigma}\cdot(m_{y'\sigma}\cdot c_{y'\sigma}) \;=\; \sum_{(y',\sigma)\in L\times\{0,1\}} m_{y'}\cdot c_{y'\sigma}$$

regardless of the quality of the approximation. However, the variance of X does depend on this quality; specifically, as shown in the proof of Theorem 5.2, *if we use an F-factor approximation, then* $\mathsf{V}[X] \leq (F-1)\cdot\mathsf{E}[X]^2$. It follows that keeping a list of $\widetilde{O}(\ell^3\cdot F/\epsilon^2)$ prefixes, which are not necessarily distinct, suffices to guaranteed that, in each of the $\ell = t\log_2 n$ iterations, with probability at least $1-(0.1/\ell)$, the resulting corresponding sum (i.e., $\sum_{y'\sigma\in L'} m_{y'\sigma}\cdot c_{y'\sigma}$) is within a $(1+(\epsilon/\ell))$-factor of the initial one (i.e., $\sum_{y'\in L} m_{y'}\cdot c_{y'}$). Hence, we reduced fine approximation of the number of t-cliques to an F-factor approximation of that number by making $\widetilde{O}(F\cdot\ell^4/\epsilon^2)$ oracle calls. Theorem 1.1 follows by combining this reduction with the improved coarse approximator of Theorem 1.2.

1.3 Organization

We start with a brief preliminary section (i.e., Section 2), which justifies the focus on t-partite graphs and formally discusses the auxiliary search problem that refers to prefixes of solutions to an original search problem.

In Section 3 we present a coarse approximation procedure for the number t-cliques. This procedure follows the strategy of the proof of [3, Thm. 1.3], but our presentation is quite different (and, in our opinion, more intuitive). Furthermore, a twist on our initial presentation allows to prove the stronger Theorem 1.2 (see Theorem 3.5).

In Section 4 we present a weak uniform generation procedure of t-cliques that uses a coarse approximation for the number of t-cliques. This weak uniform generation procedure is then amplified and used to obtain a fine approximation for the number of t-cliques (see Corollary 4.5 (1)), which is inferior to [5, Thm. 1]. In Section 5, we follow the ideas of [5] and directly reduces fine approximate counting of t-cliques to coarse approximate counting of t-cliques. This establishes Theorem 1.1, which improves over [5, Thm. 1] and [3, Thm. 1.4].

2 Preliminaries

Throughout this memo, we neglect integrally issues, which can be easily resolved by padding.

All algorithms we present are actually reductions: In Section 3 the reductions are from coarse approximate counting of t-cliques (in graphs) to deciding the existence of t-cliques (in graphs), whereas in Sections 4 and 5 we present reductions among various notions of approximate counting and uniform generation of ℓ-bit long solutions. Hence, while our focus is on problems concerning t-cliques in n-vertex graphs, when the actual reductions are oblivious to these specifics (i.e., in Sections 4 and 5), we we use a general formulation that only refers to the length of the solutions.

Reducing problems regarding t-cliques in general graphs to the t-partite case. The following reduction is well known. Given a general n-vertex graph, we make t copies of each vertex $v\in[n]$, placing one copy in each of the t parts, and connecting vertices in the natural manner: Specifically, for each $i\neq j$ and $u\neq v$, if $\{u,v\}$ is an edge in the original graph, then we connect the i^{th} copy of u with the j^{th} copy of v; that is, given $G=([n],E)$, we let $V_i=\{\langle i,v\rangle : v\in[n]\}$

and $E_{i,j} = \{\{\langle i,v\rangle, \xi j, w\} : \{v,w\} \in E\}$, and produce the graph with vertex-set $\cup_{i\in[t]} V_i$ and edge-set $\cup_{i\neq j\in[t]} E_{i,j}$. In other words, for each $i \neq j$, we place a double-cover of the original graph between the i^{th} part and the j^{th} part. The key observation is that the number of t-cliques in the resulting t-partite graph is $t!$ times the number of t-cliques in the original graph.

The auxiliary problems regarding solutions that fit a prefix. We shall focus on problems that are associated with binary relations of the type

$$R \subseteq \bigcup_{n\in\mathbb{N}} \left(\{0,1\}^n \times \{0,1\}^{\ell(n)} \right) \tag{1}$$

for some (time-constructible) function $\ell : \mathbb{N} \to \mathbb{N}$. Letting $R(x) \stackrel{\text{def}}{=} \{y : (x,y) \in R\}$ denote the set of solutions to the instance x (w.r.t R) and $S_R \stackrel{\text{def}}{=} \{x : R(x) \neq \emptyset\}$, the (candid) search problem associated with R is to find $y \in R(x)$ when given $x \in S_R$. The corresponding counting problem is to compute $\#R : \{0,1\}^* \to (\mathbb{N} \cup \{0\})$ defined as $\#R(x) \stackrel{\text{def}}{=} |R(x)|$, whereas uniform generation essentially requires outputting a uniformly distributed solution (i.e., on input $x \in S_R$, output a uniformly distributed element of $R(x)$). The auxiliary problems regarding solutions that fit a prefix are the corresponding problems that refer to the relation R' defined as follows

$$R' \stackrel{\text{def}}{=} \{(\langle x, y'\rangle, y'') : (x, y'y'') \in R\}. \tag{2}$$

For example, the corresponding search problem is to find $y'' \in R'(x,y')$ when given $\langle x, y'\rangle \in S_{R'}$; that is, given an instance x and a prefix y' of a solution to x (w.r.t R), find an extension of this prefix to a solution (to x w.r.t R).

In the standard complexity setting (of polynomial-time solvability), in natural cases as well as when R is an NP-complete search problem, it is easy to reduce the search problem of R' to the search problem of R', whereas parsimonious reductions are used for reduction among the corresponding counting (resp., uniform generation) problems. As observed in [5], such a reduction also exists for the problems regarding t-cliques. Specifically, consider the auxiliary search problem in which prefixes of t-cliques in an n-vertex graph have the form $(v_1, ..., v_{t-t'}, \alpha)$, where $v_1, ..., v_{t-t'} \in [n]$ and $\alpha \in \bigcup_{i=0}^{(\log_2 n)-1} \{0,1\}^i$. Then, the set of t-cliques in the t-partite graph $G = ([n], E)$ that fit the latter prefix corresponds to the set of t'-cliques in the t'-partite subgraph of G that is induced by the vertices that neighbor all v_i's (for $i \in [t-t']$) such that the first part contains only vertices with prefix α. That is, if $(V_1, ..., V_t)$ is the t-partition of $G = ([n], E)$ and S denote the set of vertices that negighbor $v_1, ..., v_{t-t'}$ (i.e., $u \in S$ iff $\{u, v_i\} \in E$ for every $i \in [t-t']$), then we consider the subgraph induced by $\left(\{v \in V_{t-t'+1} : \exists \beta \text{ s.t. } v = \alpha\beta\} \cup \bigcup_{i\in[t-t'+2,t]} V_i \right) \cap S$.

3 Coarse approximation for t-cliques

In this section we present a reduction of approximate counting the number of t-cliques in a graph to deciding whether a graph has t-cliques. In the context of NP-search problems such reductions are randomized, and we aim for the same here.

3.1 The starting point

As stated in Section 1.1, our starting point is the abstract idea that underlies the reduction of approximate counting NP-witnesses to deciding their existence. Specifically, in order to verify that the number of t-cliques in a graph is at least m, we apply a "random sieve" of density $1/m$ to the set of t-cliques and check whether the resulting graph had a t-clique. The question at hand is how to implement a random sieve, given that hashing (which is the method of choice in the context of NP) does not seem adequate in the current setting. Nevertheless, an appealing and straightforward way of implementing a random sieve in the current context consists of selecting each vertex with probability $p = (1/m)^{1/t}$, and considering the induced subgraph.

Unfortunately, this does not work. While each specific t-clique passes this random sieve with probability $1/m$, these choices are not independent enough, and the dependency leads to our failure. To see this fact, consider the case of $t = 2$ and an n-vertex bipartite graph with $m = o(n^2)$ edges such that $2m/n$ vertices (on one side) are each connected to $n/2$ vertices (on the other side). Then, when selecting each vertex with probability $\sqrt{1/m}$, we are likely to end-up selecting no vertex of degree $n/2$, because $(2m/n) \cdot (1/m)^{1/2} = 2m^{1/2}/n = o(1)$.

Nevertheless, a small twist on the foregoing suggestion does work. Consider, for simplicity, the case of $t = 2$ and bipartite graphs. Then, for every $i \in [\log_2 m]$, we select at random each vertex on one side of the graph with probability 2^{-i}, while selecting each vertex on the other side with probability $2^i/m$, where all these choices are independent of one another. As before, we consider the induced subgraph, and the question is whether it contains any edge.

On the one hand, observe that if the number of edges in the bipartite graph is $o(m/\log n)$, then, for each value of i the expected number of edges in the induced subgraph is $o(1/\log n)$. Hence, with probability $1 - o(1)$, in all $\ell \stackrel{\text{def}}{=} \log_2 m$ attempts (i.e., all $i \in [\ell]$) the resulting subgraph contains no edges, which means that m is not accepted as a valid approximation (to the number of edges in the graph).

On the other hand, if the n-vertex bipartite graph contains $m' = \omega(m \log n)$ edges, then there exists an $i \in [\ell]$ such that the first side of the bipartite graph contains at least $n' = 10 \cdot 2^i$ vertices of degree at least $\frac{m'}{n'\ell} = \omega(m/2^i)$. Hence, when using this i, we are likely to select a vertex of degree $\omega(m/2^i)$ along with at least one of its neighbors.

Indeed, the foregoing approximation (i.e., a factor of $O(\log^2 n)$) is very coarse, but this is all that we promised in the overview. A minor issue is that we were handling bipartite graphs rather than general graphs, but this is easy to fix (see Section 2). More importantly, the foregoing idea generalize to any $t \geq 2$.

3.2 The actual procedure (for t-cliques)

The generalization from $t = 2$ to any $t \geq 2$ is straightforward. After guessing the number of t-cliques up to a factor of 2, we guess densities (up to a factor of 2) in each of the t parts. Actually, we don't guess these parameters, but rather try all the possibilities. Furthermore, the number of t-cliques is not guessed but rather set to equal the reciprocal of the product of the relevant densities.

Algorithm 3.1 (reducing approximate counting to decision, take 1): *On input a t-partite graph $G = ([n], E)$, letting $\ell \stackrel{\text{def}}{=} \lfloor \log_2 n \rfloor$, for every $(i_1, ..., i_t) \in \{0, 1, ..., \ell - 1\}^t$, we perform the following trial.*

1. *For each $j \in [t]$, each vertex in part j is placed in the set S_j with probability 2^{-i_j}.*

2. *If the subgraph of G induced by $\cup_{j\in[t]}S_j$ contains a t-clique, then we declare $\prod_{j\in[t]}2^{i_j}$ as a candidate.*

We output the largest declared candidate, and if no candidate has been declared then we output 0.

Indeed, we view Algorithm 3.1 as making oracle calls to a decision procedure for deciding the existence of a t-clique in n-vertex graphs. Specifically, Algorithm 3.1 makes $O(\log n)^t$ such calls , and all call refer to induced subgraphs of the input graph. It is labeled "take 1" because it is quite wasteful, and can be easily improved. But let us analyze it first.

Claim 3.2 (upper-bounding the output of Algorithm 3.1): *Suppose that the number of t-cliques in $G=([n],E)$ is m. Then, with probability at least 5/6, Algorithm 3.1 outputs a non-negative integer that does not exceed $(6\cdot\log_2^t n)\cdot m$.*

Proof: Algorithm 3.1 outputs the value $m'>0$ only if for some integers $i_1,....,i_t\in\{0,1,...,\ell-1\}$ such that $\prod_{j\in[t]}2^{i_j}=m'$ the corresponding trial (in which $(i_1,....,i_t)$ is used) declared m' as a candidate. Observing that (when using $(i_1,....,i_t)$) the expected number of t-cliques in the induced subgraph equals $m\cdot\prod_{j\in[t]}2^{-i_j}=m/m'$, it follows that this event (i.e., this trial declaring a candidate) occurs with probability at most m/m'. Noting that the total number of trials is ℓ^t and applying the union bound, the claim follows. ■

Claim 3.3 (lower-bounding the output of Algorithm 3.1): *Suppose that the number of t-cliques in $G=([n],E)$ is m. Then, with probability at least 5/6, Algorithm 3.1 outputs an integer that is at least $\lceil m/O(\log n)^{t-1}\rceil$.*

Proof: Assuming that $m\geq 1$ and using a constant $c\geq 3$, we start by proving the claim for $t=2$. In this case, there exists $i_1\in\{0,1...,\ell-1\}$ such that the first part of the graph G contains at least $n_1\stackrel{\text{def}}{=}c\cdot 2^{i_1}$ vertices that are each of degree at least $\frac{m}{2\ell\cdot n_1}=\frac{m}{2c\cdot 2^{i_1}\cdot\ell}$ (because otherwise the total number of edges is smaller than $\sum_{i=0}^{\ell-1}c\cdot 2^i\cdot\frac{m}{c\cdot 2^i\cdot\ell}$).[2] Letting $i_2\stackrel{\text{def}}{=}\lfloor\log_2(m/2c^2\ell)\rfloor-i_1$ (equiv., $2^{i_1+i_2}\approx m/2c^2\ell$), we consider the trial that corresponds to (i_1,i_2). We observe that, with probability at least $1-\exp(-c)$, some vertex of degree at least $\frac{m}{2c\cdot 2^{i_1}\cdot\ell}=c\cdot 2^{i_2}$ is selected, and with probability at least $1-\exp(-c)$, one of its neighbors is selected. Hence, with probability at least $(1-\exp(-c))^2>5/6$, the value $2^{i_1}\cdot 2^{i_2}\geq m/4c^2\ell$ is declared a candidate.

Turning to the case of $t>2$, we define the clique-degree of a vertex (in the first part) as the number of t-cliques in which this vertex participates, and observe that there exists $i_1\in\{0,1...,\ell-1\}$ such that the first part of G contains at least $n_1\stackrel{\text{def}}{=}c\cdot 2^{i_1}$ vertices that are each of clique-degree at least $m_1\stackrel{\text{def}}{=}\frac{m}{2\ell\cdot n_1}=\frac{m}{2c\cdot 2^{i_1}\cdot\ell}$. Intuitively, with high probability, a vertex of clique-degree at least m_1 is selected (i.e., placed in S_1), and we consider the subgraph that is induced by its neighbors, and apply the same reasoning to this induced subgraph, which is $(t-1)$-partite.

Seeking a rigorous argument, we view the ℓ^t trials of Algorithm 3.1 as being arranged in a ℓ-ary tree of depth t such that each internal node of level $j\in\{0,1,...,t-1\}$ corresponds to a choice of $(i_1,...,i_j)\in\{0,1,...,\ell-1\}^j$. At such a node, we consider a recursive procedure that branches over all possible values of $i_{j+1}\in\{0,1,...,\ell-1\}$, and, for each such value, proceeds as follows (when given a t-partite graph and a parameter $j\in\{0,1,...,t-1\}$):

[2] Specifically, we partition the vertices (of the first part) into buckets such that the j^{th} bucket, denoted B_j, contains all vertices of degree in $[2^j,2^{j+1})$. Then, the number of edges is smaller than $\sum_{j=0}^{\ell-1}|B_j|\cdot 2^{j+1}$, which implies that $|B_j|>\frac{m}{\ell\cdot 2^{j+1}}$ for some j (equiv., $|B_i|>c2^i$ for some $i=\log_2(m/2c\ell)-j$).

- it selects each vertex in the $j+1^{\text{st}}$ part with probability $2^{-i_{j+1}}$;
- it constructs a subgraph of its own input graph by including in the $j+1^{\text{st}}$ part only the selected vertices (and including all vertices of the other parts);
- it invokes the procedure (recursively) on the resulting the subgraph;
- it returns with the value $2^{i_{j+1}} \cdot \nu$ if the recursive call returned the value ν (see Footnote 3).

The procedure itself returns the highest value among the values returned by its ℓ branches. At the leaves (i.e., $j = t$), the procedure returns the value 1 if its own input graph contains a t-clique, and returns 0 otherwise.[3] Note that invoking the foregoing recurvise procedure, on input G (with $j = 0$), is equivalent to invoking Algorithm 3.1 on input G.

Given this perspective, we consider the branch of the root that corresponds to the foregoing value of i_1 (i.e., i_1 such that there exists $n_1 = c \cdot 2^{i_1}$ vertices of clique-degree $m_1 = \frac{m}{2\ell \cdot n_1}$). Then, with probability at least $1 - \exp(-c)$, some vertex of clique-degree at least m_1 is selected. Fixing this vertex and denoting it by v, we consider the $(t-1)$-partite subgraph induced by v's neighbors, and consider the answer of the recursive procedure when applied to this subgraph (which has at least m_1 cliques of size $t-1$). Noting that procedure's answer is monotone with respect to adding edges (i.e., in each execution (i.e., per each choice of randomness), its answer can only increase when edges are added), we observe that the validity of (an adequate version of) the claim for $t-1$ implies its validity for t. Specifically, we refer to the following induction claim.

> Induction claim: *Suppose that the number of t-cliques in the t-partite n-vertex graph G is $m \geq 1$. Then, for every $c \geq 3$, on input G* (and $j = t$)*, with probability at least $1 - t \cdot \exp(-c)$, the recursive procedure answers with a value that exceeds $m/((2c)^t \cdot \log_2^{t-1} n)$.*
>
> Induction step: Let i_1 be as above (i.e., at least $c \cdot 2^{i_1}$ vertices in the first part of G have clique-degree at least $\frac{m}{2c \cdot 2^{i_1} \cdot \ell}$). Recall that, with probability at least $1 - \exp(-c)$, when extending the branch labeled i_1, a vertex of clique-degree at least $m_1 = m/(2c \cdot 2^{i_1} \cdot \ell)$, denoted v, is selected. In this case, we consider an execution of recursive procedure on the $(t-1)$-partite subgraph of G that is induced by the neighbors of v, and note that this subgraph has at least m_1 cliques of size $t-1$. (As stated above, the recursive procedure is actually invoked on a t-partite graph that includes all the vertices that were selected in the first part, but the value of its answer may only decrease when considering the t-partite graph that includes only v in the first part.)[4] By the induction hypothesis (i.e., for $t-1$), with probability at least $1-(t-1)\cdot\exp(-c)$, this execution returns a value that exceeds $m_1/((2c)^{t-1} \cdot \log_2^{t-2} n)$; hence, with probability least $1-(t-1)\cdot\exp(-c)-\exp(-c)$, the value returned by the current execution on G exceeds
>
> $$\begin{aligned} 2^{i_1} \cdot \frac{m_1}{(2c)^{t-1} \cdot \log_2^{t-2} n} &= 2^{i_1} \cdot \frac{m/(2c \cdot 2^{i_1} \cdot \ell)}{(2c)^{t-1} \cdot \log_2^{t-2} n} \\ &= \frac{m}{(2c)^t \cdot \log_2^{t-1} n} \end{aligned}$$
>
> which establishes the induction claim for t.

[3] Indeed, to streamline the analysis, we replaced the case of no declaration by the answer 0.

[4] Indeed, considering t-cliques in this t-partite graph is equivalent to considering $(t-1)$-cliques in the $(t-1)$-partite subgraph induced by the neighbors of v.

Having established already the base case of the induction (i.e., $t = 2$), and using $c = \ln(6t)$ (in order to guarantee that $t \cdot \exp(-c) \leq 1/6$), our original claim follows. ■

Digest and beyond. Our algorithm is based on bucketing the vertices according to their "degrees" (be it the actual degrees (in case of $t = 2$) or the clique-degrees (for $t \geq 3$)). This bucketing is performed for each part of the t-partite graph, when the buckets in step j refers to a fixing of $j-1$ vertices (one per each of the previous parts), and applies only to vertices in the j^{th} part that neighbor all fixed vertices (see the proof of Claim 3.3). A corresponding random sieve is shown to work for the heaviest sequence of t buckets (one per each part); that is, with probability at least $5/6$, at least one of the t-cliques that have vertices in this sequence of buckets passes the sieve.

The main source of waste (w.r.t the approximation factor) is the fact that (in each part) we only use the "heaviest" bucket rather than using all buckets. This translates to losing a logarithmic factor in each of the $t-1$ iterations of the analysis. In addition, we lose a factor of 2 by using "coarse bucketing" (i.e., using 2 rather than $1+\epsilon$ as a base). Lastly, Claim 3.2 is based on Markov Inequality, whereas a better estimate can be provided by repeating each trial several times and using a Chernoff bound. Applying the last idea, we get the following algorithm.

Algorithm 3.4 (reducing approximate counting to decision, take 2): *On input a t-partite graph $G = ([n], E)$, letting $\ell \stackrel{\text{def}}{=} \lfloor \log_2 n \rfloor$ and $r \stackrel{\text{def}}{=} O(t \log\log n)$, for every $(i_1, ..., i_t) \in \{0, 1, ..., \ell - 1\}^t$, we repeat the trial made in Algorithm 3.1 for r^t times, and combine the results using a slightly complicated scheme. Specifically, for each $\bar{i} = (i_1, ..., i_t) \in \{0, 1, ..., \ell-1\}^t$ and $\bar{k} = (k_1, ..., k_t) \in [r]^t$, the $(\bar{i}, \bar{k})$-*trial *proceeds as follows:*

1. *For each $j \in [t]$, each vertex in the j^{th} part of G is placed in the set $S_j^{(i_j,k_j)}$ with probability 2^{-i_j}.*

   Hence, all the $(\bar{i}, \bar{k})$-trials that agree on (i_j, k_j) use the same sample of the vertices of part j, whereas trials that differ on (i_j, k_j) use independent samples of the vertices of part j.

2. *If the subgraph of G induced by $\cup_{j \in [t]} S_j^{(i_j,k_j)}$ contains a t-clique, then we say that $(\bar{i}, \bar{k})$* supports *the value 1. Otherwise, we say that it* supports *the value 0.*

Using a backward recursion, we define the values supported by various pairs of sequences as follows:

> *For $j = t-1, ..., 1, 0$, we say that the pair $((i_1, ..., i_j), (k_1, ..., k_j))$* supports *the value ν if there exists $i_{j+1} \in \{0, 1, ..., \ell - 1\}$ such that for at least $r/2$ of the choices of $k_{j+1} \in [r]$ it holds that $((i_1, ..., i_j, i_{j+1}), (k_1, ..., k_j, k_{j+1}))$ supports the value $\nu/2^{i_{j+1}}$.*

We output the largest value that is supported by the empty pair (i.e., the pair of empty seqeuences corresponding to $j = 0$).

Algorithm 3.4 makes slightly more calls to the decision procedure (for existence of t-cliques) than Algorithm 3.1; that is, Algorithm 3.4 makes $\widetilde{O}(\log n)^t$ (rather than $O(\log n)^t$) such calls. In light of the fact that the main source of waste (i.e., the use of heavy buckets only) dominates the effect of the "coarse bucketing" (i.e., using 2 as a base), we left the latter aspect intact. Multiplying the output of Algorithm 3.4 by $O(\log n)^{t-1}$, we obtain a "normalized version" that satisfies the following statement.

Theorem 3.5 (the coarse approximation, a normalized form): *Suppose that the number of t-cliques in $G = ([n], E)$ is m. Then, with probability $1 - o(1)$, the normalized Algorithm 3.4 outputs an integer in the interval $[m, O(\log n)^{t-1} \cdot m]$.*

Proof: We focus analyzing the non-normalized version of Algorithm 3.4, proving that, with probability $1 - o(1)$, it outputs an integer in the interval $[O(\log n)^{-(t-1)} \cdot m, 3^t \cdot m]$. Our analysis of Algorithm 3.4 follows the basic strategy of the analysis of Algorithm 3.1, while focusing on the actual adaptations. We start with an *overview of the analysis*, which relies heavily on the definition of *supporting a value* that determines the output of Algorithm 3.4.

When upper-bounding the output of Algorithm 3.4 (akin Claim 3.2), we note that, for each $i_1 \in \{0, 1, ..., \ell - 1\}$, *the number of t-cliques in G equals 2^{i_1} times the expected number of t-cliques in the subgraph of G induced by the set of vertices that passed the random sieve of density 2^{-i_1} that is applied to the first part of G* (while including in the subgraph all vertices of the other parts). Thus, when applying r random sieves of density 2^{-i_1}, with probability at least $1 - \exp(-\Omega(r)) = o(1/\ell^t)$, the average number of t-cliques in the r corresponding induced subgraphs is within a 1 ± 0.1 factor of the expected number. Hence, in the typical case, in at most $\frac{1.1}{3} \cdot r < r/2$ of these r subgraphs, the number of t-cliques exceeds the expectation by a factor of 3. Letting m denoting the number of t-cliques in G, it follows that more than $r/2$ of these induced subgraphs have at most $3 \cdot m/2^{i_1}$ cliques (of size t). On the other hand, letting m' denote the output of the algorithm, it follows that at least $r/2$ subgraphs were supported by a claimed value of $m'/2^{i_1}$ (equiv., invoking Algorithm 3.4 on each of these subgraphs yields an output of $m'/2^{i_1}$). Hence, at least one of the subgraphs satisfies both conditions, and the argument proceeds by considering this subgraph and the choice of i_2, and then moves to i_3 and so on up to i_t. It follows that $m' = \prod_{j \in [t]} 2^{i_j}$ may hold only if $3^t \cdot m / \prod_{j \in [t]} 2^{i_j} \geq 1$, which implies $m' \leq 3^t \cdot m$. Applying a union bound on all possible $(i_1, ..., i_t) \in \{0, 1, ..., \ell - 1\}^t$, the desired upper bound follows. For details, see Claim 3.5.1.

When lower-bounding the output of Algorithm 3.4 (akin Claim 3.3), we consider the index $i_1 \in \{0, 1, ..., \ell - 1\}$ that corresponds to the heaviest bucket (w.r.t clique-degrees); that is, a bucket that is "responsible" for at least a $1/2\ell$ fraction of the number of t-cliques in G (i.e., at least 2^{i_1} of the vertices have clique-degree at least $2^{-i_1} \cdot n/2\ell$). In this case, with probability at least $1 - (1 - 2^{-i_1})^{2^{i_i}} > 0.6$, *the number of t-cliques in the subgraph of G induced by the set of vertices that passed the random sieve of density 2^{-i_1} that is applied to the first part of G is at least $2^{-i_1} \cdot m/2\ell$*, where m denotes the number of t-cliques in G. Thus, when applying r random sieves of density 2^{-i_1} (to the first part of G, while including all vertices of the other parts), with probability at least $1 - \exp(-\Omega(r)) = o(1/\ell^t)$, at least $r/2$ of these subgraphs contain at least $\frac{m}{2\ell \cdot 2^{i_1}}$ cliques (of size t). If each of the $r/2$ corresponding pairs supports a value of $m'/2^{i_1}$, then the algorithm outputs the value m', which means that we lose a factor of at most 2ℓ in this reduction (which is effected by the choice of i_1). We then consider the $r/2$ corresponding subgraphs and proceed as above (i.e., first to corresponding choices of i_2, and then to $i_3, ..., i_t$). Intuitively, at the end of each such iterative process it holds that $\frac{m}{\prod_{j \in [t]} (2\ell \cdot 2^{i_j})} \geq 1$, and this contributes to an eventual support for the value $\prod_{j \in [t]} 2^{i_j} = m'$, which implies that $m' \geq m/(2\ell)^t$. Actually, the last iteration (i.e., $j = t$) is handled differently (i.e., trivially), and so we get $m' \geq m/(2\ell)^{t-1}$. For details, see Claim 3.5.2.

Claim 3.5.1 (upper-bounding the output of Algorithm 3.4): *Suppose that the number of t-cliques in $G = ([n], E)$ is m. Then, with probability $1 - o(1)$, Algorithm 3.4 outputs a non-negative integer that does not exceed $3^t \cdot m$.*

Proof: The claim is proved by backward induction on the index $j \in \{0,1,...,t\}$. Letting $\overline{S}_h = (S_h^{(i,k)})_{i\in\{0,1,...,\ell-1\},k\in[r]}$, the induction claim is that, for fixed $\overline{S}_1,....,\overline{S}_j$, in an execution of Algorithm 3.4, *with probability* $1-o(1)$ *over the choice of* $\overline{S}_{j+1},...,\overline{S}_t$, *if some pair of* j*-long sequences, denoted* $((i_1,...,i_j),(k_1,...,k_j))$, *supports the value* m'_j, *then the corresponding induced subgraph* (i.e., the subgraph induced by $S_1^{(i_1,k_1)},...,S_j^{(i_j,k_j)}$ and all vertices of the other parts) *contains at least* $3^{-(t-j)}\cdot m'_j$ *cliques* (of size t). Equivalently, m'_j is upper-bounded by 3^{t-j} times the number of t-cliques in this subgraph.

The base case (i.e., $j=t$) is trivial, since pairs of ℓ-long sequences support only binary values (which are always correct), whereas the case of $j=0$ implies the main claim. In the induction step, we assume that the claim holds for $j+1\in[t]$, and prove that it holds for j. The induction step mimics the proof of Claim 3.2: Fixing any $((i_1,...,i_j),(k_1,...,k_j)) \in \{0,1,...,\ell-1\}^j\times[r]^j$, we observe that this pair supports the value $m'_j>0$ only if, for some $i_{j+1}\in\{0,1,...,\ell-1\}$, the value $m'_j/2^{i_{j+1}}$ is supported by at least $r/2$ of the r pairs of the form $((i_1,...,i_j,i_{j+1}),(k_1,...,k_j,.))$. Letting $(V_1,...,V_t)$ denote the t-partition of the vertices of G, the key observation is that, for every $k\in[r]$, *the* **expected** *nunber of* t*-cliques in the subgraph of* G *induced by*

$$R_k \stackrel{\rm def}{=} \bigcup_{h\in[j]} S_h^{(i_h,k_h)} \cup S_{j+1}^{(i_{j+1},k)} \cup \bigcup_{h\in\{j+2,...,t\}} V_h$$

equals $m_j/2^{i_{j+1}}$, where m_j denotes the nunber of t-cliques in the subgraph of G induced by $\bigcup_{h\in[j]} S_h^{(i_h,k_h)} \cup \bigcup_{h\in\{j+1,...,t\}} V_h$ and *the expectation is over the choice of* $S_{j+1}^{(i_{j+1},k)}$.

While the proof of Claim 3.2 applies the Markov Inequality at this point, here we use the fact that the foregoing experiment (i.e., the random choice of $S_{j+1}^{(i_{j+1},k)}$) is repeated r times (i.e., for each $k\in[r]$), and the fact that we only need $r/2$ of these experiments to yield values that do not exceed the expectation by much. Specifically, we first observe that, with probability at least $1-\exp(-\Omega(r)) = o(1/\ell^{2t})$, the average (over $k\in[r]$) number of t-cliques in the subgraphs of G induced by R_k's is smaller than $1.1\cdot m_j/2^{i_{j+1}}$. It follows that, in this (highly typical) case, there are less than $r/2$ indices $k\in[r]$ such that the number of t-cliques in the subgraph of G induced by R_k exceeds $\frac{1.1}{1/2}\cdot\frac{m_j}{2^{i_{j+1}}}$, which means that for more that $r/2$ indices $k\in[r]$ the number of t-cliques is at most $\frac{2.2\cdot m_j}{2^{i_{j+1}}} < \frac{3\cdot m_j}{2^{i_{j+1}}}$. Hence, there exists a $k_{j+1}\in[r]$ such that the value $\frac{m'_j}{2^{i_{j+1}}}$ is supported by $((i_1,...,i_j,i_{j+1}),(k_1,...,k_j,k_{j+1}))$ and the number of t-cliques in the subgraph of G induced by $R_{k_{j+1}}$ is smaller than $\frac{3\cdot m_j}{2^{i_{j+1}}}$. Using the induction hypothesis (for $j+1$), it follows that

$$\frac{m'_j}{2^{i_{j+1}}} \leq 3^{t-(j+1)}\cdot\frac{3\cdot m_j}{2^{i_{j+1}}}$$

which in turn implies that $m'_j \leq 3^{t-j}\cdot m_j$.

Recalling that the foregoing holds, for each $((i_1,...,i_j,i_{j+1}),(k_1,...,k_j))$, with probability $o(1/\ell^t)$, and using a union bound on all possible pairs in $\{0,1,...,\ell+1\}^{j+1}\times[r]^j$, this establishes the induction step, and the entire claim follows. □

Claim 3.5.2 (lower-bounding the output of Algorithm 3.4): *Suppose that the number of* t*-cliques in* $G=([n],E)$ *is* m. *Then, with probability* $1-o(1)$, *Algorithm 3.4 outputs an integer that is at least* $\lceil m/(2\ell)^{t-1}\rceil$.

Proof: When lower-bounding the output of Algorithm 3.4 (akin Claim 3.3), we analyze the general case of $t \geq 2$ directly. We shall prove that, with probability $1 - o(1)$, a value of at least $m/(2\ell)^{t-1}$ is supported by the empty pair (i.e, the pair corresponding to $j = 0$), where m is the number of t-cliques in G.

As in the case of Claim 3.3, the proof is by induction on t, while noting that the claim is trivial for $t = 1$. Recall that, since G has $m \geq 1$ cliques (of size t), there exists $i_1 \in \{0, 1, ..., \ell - 1\}$ such that the first part of G has at least $n_1 \stackrel{\text{def}}{=} 3 \cdot 2^{i_1}$ vertices of clique-degree at least $m_1 \stackrel{\text{def}}{=} \frac{m}{2\ell \cdot n_1} = \frac{m}{2 \cdot 2^{i_1} \cdot \ell}$. Hence, with probability at least $1 - (1 - 2^{-i_1})^{2^{i_i}} > 0.6$, for each $k_1 \in [r]$, the set $S_1^{(i_1,k_1)}$ contains some vertex of clique-degree at least m_1, denoted $v^{(k_1)}$. In this case we say that k_1 is i_1-good. With probability at least $1 - \exp(-\Omega(r))$, at least $r/2$ of the indices $k_1 \in [r]$ are i_1-good. For each of these good $k_1 \in [r]$, we consider the subgraph of G induced by the neighbors of $v^{(k_1)}$, and note that this $(t-1)$-partite subgraph contains at least m_1 cliques (of size $t-1$). When properly defining the induction claim (see below), it follows that, with high probability, each of the residual $r/2$ executions supports a value that is at least $m_1/(2 \log_2 n)^{t-2}$, which establishes the induction claim. Details follow.

Our induction claim is that *if the number of t-cliques in the t-partite n-vertex graph G is $m \geq 1$, then, on input G, with probability at least $1 - r^t \cdot \exp(-\Omega(r))$, Algorithm 3.4 outputs a value that is at least $m/(2\ell)^{t-1}$*. The induction step is proved by viewing Algorithm 3.4 as branching on all $(i_1, k_1) \in \{0, 1, ..., \ell - 1\} \times [r]$ and invoking itself recursively with a corresponding induced subgraph (i.e., the $(t-1)$-partite subgraph of G that is induced by the neighbors of $v^{(k_1)}$). We actually care only about the branches associated with (i_1, k_1) such that the first part of G has at least $3 \cdot 2^{i_1}$ vertices of clique-degree at least $m_1 = \frac{m}{2\ell \cdot 2^{i_1}}$ and k_1 is i_1-good.

Note that if the number of i_1-good k's is at least $r/2$ and each of the corresponding recursive invocation (of the algorithm on the corresponding $(t-1)$-partite induced subgraphs) outputs a value of at least $m_1' \stackrel{\text{def}}{=} m_1/(2\ell)^{t-2}$, then the algorithm itself outputs a value that is at least $2^{i_1} \cdot m_1' = m/(2\ell)^{t-1}$. By the induction hypothesis (for $t-1$), with probability at least $1 - r^{t-1} \cdot \exp(-\Omega(r))$, each of these invovations return a value of at least m_1'. Recalling that, with probability at least $1 - \exp(-\Omega(r))$, the first event holds (i.e., the number of i_1-good k's is at least $r/2$), this establishes the induction claim (for t), since failure occurs with probability at most $(1 + (r/2)) \cdot (r^{t-1} \cdot \exp(-\Omega(r)))$. The main claim follows by observing that $r^t = o(\exp(\Omega(r)))$. □

Combining Claims 3.5.1 and 3.5.2 (and recalling the normalization), and the theorem follows. ■

Digest. The fact that Algorithm 3.4 uses r^t trials per each sequence $(i_1, ..., i_t) \in \{0, 1, ..., \ell - 1\}^t$ allows to improve the upper bound on its output (see Claim 3.5.1 vs Claim 3.2), while having little effect on the lower bound (see Claim 3.5.2 vs Claim 3.3).[5] The proof of Claim 3.5.2 proceeds from shorter sequences to longer ones (i.e., from $(i_1, ..., i_j)$ to $(i_1, ..., i_j, i_{j+1})$). This is to be expected because here each subgraph, which is defined based on prior choices that include $i_1, ..., i_j$, has its own favorable choice of the next i_{j+1}. In contrast, in the proof of Claim 3.5.1, we have to analyze all possible choices of $(i_1, ..., i_t) \in \{0, 1, ..., \ell - 1\}^t$, and it feels more natural to proceed by a backward induction (i.e., from $(i_1, ..., i_j, i_{j+1})$ to $(i_1, ..., i_j)$).

[5] Actually, we get a minor improvement also on the lower bound: The hidden constant in Claim 3.3 is $O(\log t)$, whereas the explicit constant in Claim 3.5.2 is 2.

4 Approximate counting versus uniform generation

In Section 4.1 we briefly review the standard reductions between approximate counting solutions (to NP-search problems) and uniform generation of such solutions, while adapting both reductions to the current context of problems concerning t-cliques in (t-partite) graphs.[6] In Section 4.2 we show that the known reduction of uniform generation to approximate counting is meaningful also when the approximation is coarse. Actually, we generalize the known reduction in order to obtain a meaningful result for the context of t-cliques.

4.1 The standard presentation adapted to t-cliques

Following [8], we show that approximate counting of t-cliques in t-partite n-vertex graphs and uniform generation of t-cliques in such graphs are reducible to one another (in a fine-grained complexity sense). Both reductions proceed by considering the (depth $t \cdot \log_2 n$) binary tree of all possible t-cliques such that the internal nodes corresponds to prefixes of possible t-cliques (see [6, Sec. 6.2.4.1]). In particular, the root corresponds to an empty prefix, whereas each leaf corresponds to the full description of a possible t-clique (i.e., a sequence of t vertices). Each reduction uses its oracle in order to determine its next move along a path that leads from the root to some vertex. The reduction proceeds in iterations such that in the i^{th} iteration it extends the current $(i-1)$-bit long prefix by one bit.

In the case of uniform generation, the current prefix is extended at random with probability that is proportional to the approximate number of t-cliques that fit each of the possible one-bit extensions. That is, the current prefix γ is extended by the bit σ with probability that is proportional to the approximate number of t-cliques that are described by strings with prefix $\gamma\sigma$. (See more details in Section 4.2.) In the case of approximate, we use a sample of uniformly generated t-cliques that fit the current prefix in order to approximate the fraction of t-cliques that agree with each possible one-bit extension, and proceed with the seemingly more frequent extension. (At the end, the approximate count will be set to equal the reciprocal of the the product of these frequencies.)

The issue at hand is *reducing the tasks that refer to fixed prefixes of potential t-cliques to tasks regarding t'-cliques for some $t' \in [t]$*. This issue was addressed by [5], see Sections 1.2 and 2. Thus, in direct analogy to [6, Thm. 6.31], we get the following

Theorem 4.1 (approximate counting t-cliques versus uniform generation of t-cliques):

1. From approximate counting to uniform generation: *Almost uniform generation of t-cliques in a t-partite n-vertex graph is reducible in almost linear-time to approximating the number of t-cliques in such graphs up to a factor of $1 \pm (1/5t\log_2 n)$, where almost uniform generation allows for a negligible deviation* (i.e., the deviation is smaller than $1/\text{poly}(n)$).

2. From uniform generation to approximate counting: *Approximate counting of t-cliques in a t-partite n-vertex graph is reducible in almost linear-time to uniformly generating t-cliques in such graphs. The deviation of the approximation, which is negligible, is $O(t\log n)$ times larger than the deviation of the uniform generation.*

In both cases, the reduction makes $\text{poly}(t\log n)$ *oracle calls.*

[6] Recall that the case of general graphs can be easily reduced to the case of t-partite graphs.

Needless to say, the result also holds for general graphs (i.e., graphs that are not necessarily t-partite).

Digest. Using the terminology of Section 2, we mention that, in both cases, a problem regarding a relation R (such that $|y| = \ell(|x|)$ for every $(x,y) \in R$) is reduced to a problem regarding the corresponding auxiliary relation R', and in this case $t \log_2 n$ is replaced by $\ell(|x|)$. In the case of t-cliques, a reduction of R' to R is used in order to remove R' from the theorem's statement.

4.2 Using coarse approximate counters

In this section we show that the reduction of uniform generation to approximate counting is meaningful also when the approximation is coarse. The following text reproduces part of [6, Sec. 6.2.4.1], while making the relevant adaptions. We first generalize the definition of uniform generation (as in [6, Def. 6.30]), allowing for an arbitrary lower bound on the success probability (rather that setting it to 1/2).

Definition 4.2 (uniform generation, quantified success probability): *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a search problem, $R(x) \stackrel{\text{def}}{=} \{y : (x,y) \in R\}$ and $S_R = \{x : |R(x)| \geq 1\}$. For $\eta, \epsilon : \mathbb{N} \to [0,1]$, we say that a randomized process $(1-\eta)$-solves the $(1-\epsilon)$-approximate uniform generation problem of R if, on input $x \in S_R$, the process, denoted Π, outputs either an element of $R(x)$ or a special symbol, denoted $\perp$, such that $\mathsf{Pr}[\Pi(x) \in R(x)] \geq 1 - \eta(|x|)$ and*

$$\sum_{y \in R(x)} \left| \mathsf{Pr}\left[\Pi(x) = y \,\middle|\, \Pi(x) \in R(x)\right] - \frac{1}{|R(x)|} \right| \leq \epsilon(|x|).$$

That is, η upper-bounds the failure probability of Π (i.e., the probability that it outputs $\perp$)*, whereas ϵ upper-bounds the deviation of $\Pi(x)$ from uniform distribution on $R(x)$, when conditioning on $\Pi(x) \in R(x)$* (equiv., on $\Pi(x) \neq \perp$)*.*

- *If ϵ is negligible* (i.e., vanishes faster than the reciprocal of any positive polynomial)*, then we say that the process $(1-\eta)$-solves the uniform generation problem of R.*
- *In addition, if η is also negligible, then we say that the process* solves the uniform generation problem of R.

The following result generalizes the essence of the first direction of [6, Thm. 6.31]. This generalization, which explicitly supports coarse approximations, introduces an additional parameter (i.e., m) that governs the trade-off between the query complexity of the procedure (i.e., q) and its success probability (i.e., $F^{-(m+1)}$). We shall capitalize on this trade-off in Corollary 4.4.

Theorem 4.3 (from coarse approximate counting to weak uniform generation): *Let R be as in Definition 4.2, $R' \stackrel{\text{def}}{=} \{(\langle x, y' \rangle, y'') : (x, y'y'') \in R\}$ and $R'(x,y') \stackrel{\text{def}}{=} \{y'' : (x, y'y'') \in R\}$. Suppose that for a time-constructible and at least logarithmic[7] function $\ell : \mathbb{N} \to \mathbb{N}$ it holds that $|y| \leq \ell(|x|)$ for every $(x,y) \in R$. For $F : \mathbb{N} \to \mathsf{R}$ such that $F(n) \geq 1$ for every n, suppose that A approximates*

[7] This condition is made for sake of simplicity. If $\ell(n) = o(\log n)$, then we need to add a $\log n$ factor to q.

the number of solutions wrt R' in the sense that for every x and y', with probability at least $2/3$, it holds that

$$|R'(x,y')| \leq A(x,y') \leq F(|x|) \cdot |R'(x,y')|. \tag{3}$$

Then, for any time-constructible $m : \mathbb{N} \to \mathbb{N}$, given oracle access to A, one can $F^{-(m+1)}$-solve the uniform generation problem of R by making $q = \widetilde{O}(m \cdot 2^{\ell/m})$ oracle calls to A and running in time $q(|x|) \cdot (|x| + \ell(|x|))$ on input x.

(The statement of the first direction of [6, Thm. 6.31] essentially postulated that $F = \frac{5\ell+1}{5\ell-1} \leq 1 + (1/2\ell)$ and $m = \ell$; indeed, in that case $F^{-(m+1)} > 1/2$.)[8]

Proof: Throughout the proof, we assume for simplicity (and in fact without loss of generality) that $R(x) \neq \emptyset$ and $R(x) \subseteq \{0,1\}^{\ell(|x|)}$ for every x. Generalizing the overview provided in Section 4.1, we view a generic $y \in R(x)$ as partitioned to $\ell'(|x|)$ blocks, each of length $\ell' = \ell(|x|)/m(|x|)$. On input x, we shall generate a uniformly distributed $y \in R(x)$ by iteratively generating at random its ℓ'-bit blocks, one after the other.

Let us first describe the reduction assuming that we have oracle access to $\#R'$ (rather than to A that only approximates $\#R'$). We proceed in iterations, entering the i^{th} iteration with an $(i-1) \cdot \ell'$-bit long string y' such that $R'(x,y')$ is not empty. For each $z \in \{0,1\}^{\ell'}$, with probability $|R'(x,y'z)|/|R'(x,y')|$ we set the i^{th} block to equal z. Hence, after $m(|x|)$ iterations, we obtain a uniformly distributed element of $R(x)$. Recalling that we only have oracle access to a (coarse) approximation of $\#R'$, a careful implementation of the foregoing strategy is in place.

Firstly, by adequate error reduction, we may assume that Eq. (3) fails with probability $o(\mu/q)$, where μ is a negligible function. In the rest of the analysis we ignore the probability that the estimate of $\#R'(x,y')$ provided by the randomized oracle A (on query (x,y')) violates Eq. (3). (We note that these rare events are the only source of the possible deviation of the output distribution from the uniform distribution on $R(x)$.)[9] Next, let us assume for a moment that A is *deterministic* and that for every x and $y' \in (\{0,1\}^{\ell'})^*$ it holds that

$$\sum_{z \in \{0,1\}^{\ell'}} A(x,y'z) \leq A(x,y'). \tag{4}$$

We also assume that the approximation is actually perfect at the "trivial level" (where it corresponds to whether or not (x,y) is in R); that is, for every $y \in \{0,1\}^{\ell(|x|)}$, it holds that

$$A(x,y) = 1 \text{ if } (x,y) \in R \text{ and } A(x,y) = 0 \text{ otherwise.} \tag{5}$$

Relying on these assumptions, we modify the i^{th} iteration of the foregoing procedure such that, when entering with the $(i-1) \cdot \ell'$-bit long prefix y', we set the i^{th} block to $z \in \{0,1\}^{\ell'}$ with probability $A(x,y'z)/A(x,y')$ and halt (with output $\perp$) with the residual probability (i.e., $1 - \sum_z A(x,y'z)/A(x,y')$). Indeed, Eq. (4) guarantees that the latter instruction is sound, since the $2^{\ell'}$

[8] Furthermore, Theorem 4.3 provides more explicit complexity bounds. On the other hand, in some technical aspects, the first direction of [6, Thm. 6.31] is more general; however, the actual proof starts by reducing the more general formulation to the foregoing formulation.

[9] Note that the (negligible) effect of these rare events may not be easy to correct. For starters, we do not necessarily get an indication when these rare events occur. Furthermore, these rare events may occur with different probability in the different invocations of algorithm A (i.e., on different queries).

main probabilities sum-up to at most 1. Hence, in each iteration we make $2^{\ell'}$ oracle calls, where these calls are required in order to compute $\sum_z A(x,y'z)$.

If we completed the last (i.e., $m(|x|)^{\text{th}}$) iteration, then we output the $\ell(|x|)$-bit long string that was generated. Thus, as long as Eq. (4) holds (but regardless of other aspects of the quality of the approximation), every $y = z_1 \cdots z_{m(|x|)} \in R(x)$, where the z_i's are in $\{0,1\}^{\ell'}$, is output with probability

$$\frac{A(x,z_1)}{A(x,\lambda)} \cdot \frac{A(x,z_1z_2)}{A(x,z_1)} \cdot \cdots \cdot \frac{A(x,z_1z_2\cdots z_{m(|x|)-1}z_{m(|x|)})}{A(x,z_1z_2\cdots z_{m(|x|)-1}))} \tag{6}$$

which equals $1/A(x,\lambda)$, where the equality relies on Eq. (5). Thus, the procedure outputs each element of $R(x)$ with equal probability, and never outputs a non-$\perp$ value that is outside $R(x)$. It follows that the quality of approximation only effects the probability that the procedure outputs a non-$\perp$ value, which in turn equals $|R(x)|/A(x,\lambda)$. The key point is that, as long as Eq. (4) and Eq. (5) hold, the specific approximate values obtained by the procedure are immaterial – with the exception of $A(x,\lambda)$, all these values "cancel out".

We now turn to enforcing Eq. (4) and Eq. (5). In most settings, one can enforce Eq. (5) by performing the straightforward check (of whether or not $(x,y) \in R$) rather than invoking $A(x,y)$. However, since we made no hypothesis regarding the complexity of recognizing R, we take the alternative of modifying A such that $A(x,y) \in \{0,1\}$ whenever $|y| = \ell(|x|)$; specifically, we reset $A(x,y) = 1$ if $A(x,y) > 1$ (and observe that $A(x,y) \in (0,1)$ is impossible).[10] As for Eq. (4), we enforce it artificially by using $A'(x,y') \stackrel{\text{def}}{=} F(|x|)^{m(|x|)-i} \cdot A(x,y')$, for every $y' \in (\{0,1\}^{\ell'})^i$, instead of $A(x,y')$. Recalling Eq. (3), we have for every $y' \in (\{0,1\}^{\ell'})^i$ and $z \in (\{0,1\}^{\ell'}$,

$$\begin{aligned} A'(x,y') &\geq F(|x|)^{m(|x|)-i} \cdot |R'(x,y')| \\ A'(x,y'z) &\leq F(|x|)^{m(|x|)-(i+1)} \cdot F(|x|) \cdot |R'(x,y'z)| \end{aligned}$$

and the claim (that Eq. (4) holds) follows (because $R'(x,y') = \bigcup_{z\in\{0,1\}^{\ell'}} R'(x,y'z)$). Note that the foregoing modification only effects the probability of outputting a non-$\perp$ value; this good event now occurs with probability $|R'(x,\lambda)|/A'(x,\lambda)$, which is lower-bounded by $F(|x|)^{-(m(|x|)+1)}$.

Finally, we refer to our assumption that A is deterministic. This assumption was only used in order to identify the value of $A(x,y')$ obtained and used in the $((|y'|/\ell') - 1)^{\text{st}}$ iteration with the value of $A(x,y')$ obtained and used in the $(|y'|/m)^{\text{th}}$ iteration. The same effect can be achieved by just re-using the former value (in the $(|y'|/m)^{\text{th}}$ iteration) rather than re-invoking A in order to obtain it. The theorem follows. ■

Combining Theorem 4.3 with a starightforward amplification of the success probability of uniform generation, while using a suitable setting of m, we get the following

Corollary 4.4 (from coarse approximation to uniform generation): *Let R, R', F and A be as in Theorem 4.3. Then, given oracle access to A, one can solve the uniform generation problem of R by making $q = \exp(O(\sqrt{\ell \cdot \log F}))$ oracle calls to A and running in time $q(|x|) \cdot (|x| + \ell(|x|))$ on input x.*

[10] Note that Eq. (3) implies that $A(x,y') = 0$ whenever $R'(x,y') = \emptyset$, whereas $A(x,y') \geq 1$ whenever $R'(x,y') \neq \emptyset$. Hence, for $|y| = \ell(|x|)$, we may reset $A(x,y) = 1$ whenever $A(x,y) > 1$, while noting that this modification cannot cause violation of Eq. (3).

Note that for $F = \text{poly}(\ell)$, we get $q = \exp(\widetilde{O}(\sqrt{\ell}))$. Specifically, in case of t-cliques in n-vertex graphs, we have $\ell(n) = t \cdot \log_2 n$ and $F(n) = O(\log n)^{O(t)}$, which yields $q(n) = \exp(\widetilde{O}(\sqrt{\log n})) = n^{o(1)}$. This is inferior to the result of [5, Thm. 2], but the proof is conceptually simpler.

Proof: Setting $m = \sqrt{\ell/\log F}$, we merely invoke the weak uniform generator provided by Theorem 4.3 for a sufficient number of times and output the first non-$\perp$ value provided. Specifically, on input x, we invoke this $F^{-(m+1)}$-solver for $\Theta(\log |x|)^2 \cdot F(|x|)^{m(|x|)+1}$ times. Hence, we fail to produce output with probability at most $\exp(-\Omega(\log |x|)^2) \ll 1/\text{poly}(|x|)$. The number of queries made by the resulting reduction, on input x, is

$$\begin{aligned}
&\left(O(\ell(|x|)^2) \cdot F(|x|)^{m(|x|)+1}\right) \cdot \widetilde{O}\left(m(|x|) \cdot 2^{\ell(|x|)/m(|x|)}\right) \\
&= \text{poly}(\ell(|x|)) \cdot F(|x|)^{\sqrt{\ell(|x|)/\log F(|x|)}} \cdot 2^{\sqrt{\ell(|x|)\cdot \log F(|x|)}} \\
&= \exp(O(\sqrt{\ell(|x|) \cdot \log F(|x|)})).
\end{aligned}$$

The claim follows. ■

Combining Corollary 4.4 with Theorem 3.5 (and Theorem 4.1), we get

Corollary 4.5 (approximate counting and uniform generation of t-cliques):

1. *Given oracle access to a decision procedure for t-cliques in n-vertex graphs, one can solve the corresponding uniform generation problem in almost linear-time by making $q(n) = \exp(\widetilde{O}(\sqrt{\log n}))$ queries.*
2. *Given oracle access to a decision procedure for t-cliques, one can approximate the number of t-cliques in an n-vertex graph up-to a factor of $1 \pm \epsilon$ within complexities that are only $O((\log q(n))/\epsilon^2)$ times larger than those in Item 1.*

Again, this is inferior to the results of [5], but the proof is conceptually simpler.

5 Directly reducing fine to coarse approximation of t-cliques

In this section we follow [5], who showed how to use coarse approximate counters towards obtaining fine approximate counters. The key observation is that we can obtain a fine approximation of $N = \sum_{j \in J} N_j$ by using *coarse approximations of each N_j* along with *fine approximations of few individual N_j's*. Specifically, the number of fine approximations that we use is proportional to the approximation factor of the coarse approximator. We detail this idea next.

Suppose that, for some F and every $j \in J$, we get an approximation $\widetilde{N}_j$ such that $\widetilde{N}_j \in [N_j, F \cdot N_j]$. Then, letting $\widetilde{N} = \sum_{j \in J} \widetilde{N}_j$, we set $p_j \stackrel{\text{def}}{=} \widetilde{N}_j/\widetilde{N}$ and select i with probability p_j. The key observation is that the value N_j/p_j (which corresponds to a random selection of j) is an unbiased estimator of N and its variance is bounded in terms of F. This follows from the next generic ("importance sampling") claim, which is stated next. Hence, if we obtain a fine approximation of few random N_j's, then we get a fine approximation of N.

Claim 5.1 (importance sampling, a generic claim): *For positive v_j's and $v = \sum_{j \in J} v_j$, let X be a random variable that equals v_j/p_j with probability $p_j > 0$ (such that $\sum_{j \in J} p_j = 1$), and suppose that $p_j \geq \frac{v_j}{Fv}$. Then, $\mathsf{E}[X] = v$ and $\mathsf{V}[X] \leq (F-1) \cdot v^2$.*

For the foregoing application, we shall indeed use $p_j = \widetilde{N}_j/\widetilde{N}$ and rely on the guarantee $\widetilde{N}_j \geq N_j$ and $\widetilde{N} \leq F \cdot N$. In this case $p_j \geq \frac{N_j}{F \cdot N}$, and using a $1 \pm 0.5\epsilon$ factor approximation of an individual N_j yields a estimator of N that has expectaion $(1 \pm 0.5\epsilon) \cdot N$ and variance approximately $F \cdot N^2$. Hence, taking the average of $O(F/\epsilon^2)$ samples of X yields a value that is within $1 \pm \epsilon$ factor of N.

Proof: Note that $\mathsf{E}[X] = \sum_{j \in J} p_j \cdot (v_j/p_j) = v$, whereas

$$\begin{aligned}
\mathsf{E}[X^2] &= \sum_{j \in J} p_j \cdot (v_j/p_j)^2 \\
&= \sum_{j \in J} \frac{v_j^2}{p_j} \\
&\leq \sum_{j \in J} \frac{v_j^2}{v_j/Fv} \\
&= \sum_{j \in J} Fv \cdot v_j
\end{aligned}$$

which equals $F \cdot v^2$. Hence, $\mathsf{V}[X] = \mathsf{E}[X^2] - \mathsf{E}[X]^2 \leq F \cdot v^2 - v^2$. ■

The following result will be proved by using an iterative process as outlined in Section 1.2.2. Specifically, each of the samples (and corresponding values) generated at the current iteration depends on all samples and values generated in the previous iteration. We shall use Claim 5.1 to relate the average value associated with the samples produced in current iteration to the average value associated with the samples produced in previous iteration.

Theorem 5.2 (from coarse approximate counting to fine approximate counting): *Let R, ℓ, R', F and A be as in Theorem 4.3; loosely speaking, ℓ denotes the length of solutions w.r.t R, and A approximates the number of solutions w.r.t R' up to a factor of T* (see Eq. (3)). *Then, given oracle access to A, one can approximate the number of solutions wrt R up-to a factor of $1 \pm \epsilon$ by making $q = \widetilde{O}(\epsilon^{-2} \cdot F \cdot \ell^4)$ oracle calls to A and running in time $q(|x|) \cdot (|x| + \ell(|x|))$ on input x.*

Proof: Following the outline provided in Section 1.2.2, we proceed in ℓ iterations. For each $i \in [\ell]$, we enter the i^{th} iteration with a list of $k = O(F\ell^3/\epsilon^2)$ (not necessarily distinct) $(i-1)$-bit long strings coupled with corresponding "multipliers" (used in the final result), and leave it with random samples of one-bit extensions of these strings (coupled with corresponding multiplier).

The foregoing strings are prefixes of possible solutions for the given input x, and the multipliers are supposed to represent the ratio between the total number of solutions to x and the number of solutions that fit the various prefixes. Specifically, the sequence of pairs given to the i^{th} iteration is denoted $((y^{(i-1,1)}, m^{(i-1,1)}), ..., (y^{(i-1,k)}, m^{(i-1,k)}))$ and we shall show that for every $i \in [\ell + 1]$, it holds that

$$\frac{1}{k} \cdot \sum_{j \in [k]} m^{(i-1,j)} \cdot N_{y^{(i-1,j)}} \approx N_\lambda \tag{7}$$

where λ denotes the empty string and $N_{y'}$ denotes the number of solutions that fit the prefix y'. In particular, $y^{(0,j)} = \lambda$ and $m^{(0,j)} = 1$ for every $j \in [k]$; hence, Eq. (7) holds trivially for $i = 1$. Each iteration generates a new list of pairs, which is given to the next iteration, whereas the last (i.e.,

ℓ^{th}) iteration produces a sequence that is used to produce the output. Specifically, since all $y^{(\ell,j)}$'s are valid solutions, the final output is $\sum_{j\in[k]} m^{(\ell,j)}/k$.

Needless to say, the crucial issue is the way in which the next list of pairs is generated. In each iteration, we generate each of the pairs in the same manner, independently of the generation of the other pairs. Specifically, on input x, for each $i \in \ell]$, in the i^{th} iteration we are given the sequence pf pairs $((y^{(i-1,1)}, m^{(i-1,1)}), ..., (y^{(i-1,k)}, m^{(i-1,k)}))$, and generate each of the new pairs as follows:

1. For each $j \in [k]$ and each $\sigma \in \{0,1\}$, we use the approximate counter A to obtain an estimate, denoted $\widetilde{N}_{y^{(i-1,j)}\sigma}$, of the number of solutions that fit the prefix $y^{(i-1,j)}\sigma$; that is, $\widetilde{N}_{y^{(i-1,j)}\sigma} \leftarrow A(x, y^{(i-1,j)}\sigma)$.

   The same values $\widetilde{N}_{y^{(i-1,j)}\sigma}$ may be used to generate all k pairs of the current iteration. In fact, it is simpler to analyze the process this way.

   Actually, since we need all $\ell \cdot 2k$ approximations to be correct, we reduce the error probability of A by invoking it $O(\log(\ell \cdot k))$ times and take the median value.

2. For each $j \in [k]$ and each $\sigma \in \{0,1\}$, we let $a_{j,\sigma} \leftarrow m^{(i-1,j)} \cdot \widetilde{N}_{y^{(i-1,j)}\sigma}$ and $p_{j,\sigma} \leftarrow a_{j,\sigma}/a$, where $a = \sum_{j',\sigma'} a_{j',\sigma'}$.

3. We select $(j,\sigma) \in [k]\times\{0,1\}$ with probability $p_{j,\sigma}$, and generate the pair $(y^{(i-1,j)}\sigma, m^{(i-1,j)}/p_{j,\sigma})$.

To evaluate the features of this generation process, we let $X^{(i)}$ denote the corresponding value of the generated pair (i.e., its contribution to Eq. (7)); that is, $X^{(i)} = \frac{m^{(i-1,j)}}{p_{j,\sigma}} \cdot N_{y^{(i-1,j)}\sigma}$ with probability $p_{j,\sigma}$. We first observe that

$$\begin{aligned} p_{j,\sigma} &= \frac{a_{j,\sigma}}{\sum_{j',\sigma'} a_{j',\sigma'}} \\ &= \frac{m^{(i-1,j)} \cdot \widetilde{N}_{y^{(i-1,j)}\sigma}}{\sum_{j,\sigma'} m^{(i-1,j')} \cdot \widetilde{N}_{y^{(i-1,j')}\sigma'}} \\ &\geq \frac{m^{(i-1,j)} \cdot N_{y^{(i-1,j)}\sigma}}{\sum_{j,\sigma'} m^{(i-1,j')} \cdot F \cdot N_{y^{(i-1,j')}\sigma'}} \end{aligned}$$

where the inequality uses $\widetilde{N}_{y^{(i-1,j)}\sigma} \geq N_{y^{(i-1,j)}\sigma}$ and $\widetilde{N}_{y^{(i-1,j')}\sigma'} \leq F \cdot N_{y^{(i-1,j')}\sigma'}$. Using $v_{j',\sigma'} = m^{(i-1,j')} \cdot N_{y^{(i-1,j')}\sigma'}$ and $v = \sum_{j',\sigma'} v_{j',\sigma'}$, we have $p_{j,\sigma} \geq \frac{v_{j,\sigma}}{F \cdot v}$. Applying Claim 5.1, it follows that $\mathsf{E}[X^{(i)}] = v$ and $\mathsf{V}[X^{(i)}] \leq (F-1) \cdot \mathsf{E}[X^{(i)}]^2$. Letting $X^{(i)}_j$ denote the contribution of the j^{th} pair (generated in i^{th} iteration) to Eq. (7), using $N_{y^{(i-1,j)}} = N_{y^{(i-1,j)}0} + N_{y^{(i-1,j)}1}$ (which implies $v_{j',0} + v_{j',1} = m^{(i-1,j')} \cdot N_{y^{(i-1,j')}}$), and applying Chebyshev's Inequality, we get

$$\begin{aligned} &\mathsf{Pr}\left[\frac{1}{k} \cdot \sum_{j\in[k]} X^{(i)}_j \notin \left[(1 \pm (\epsilon/\ell)) \cdot \frac{1}{k} \cdot \sum_{j\in[k]} m^{(i-1,j)} \cdot N_{y^{(i-1,j)}}\right]\right] \\ &= \mathsf{Pr}\left[\left|\frac{1}{k} \cdot \sum_{j\in[k]} X^{(i)}_j - \mathsf{E}[X^{(i)}]\right| > (\epsilon/\ell) \cdot \mathsf{E}[X^{(i)}]\right] \end{aligned}$$

$$\begin{aligned} &\leq \frac{\mathsf{V}[X^{(i)}]}{(\epsilon \cdot \mathsf{E}[X^{(i)}]/\ell)^2 \cdot k} \\ &\leq \frac{F-1}{(\epsilon/\ell)^2 \cdot k} \end{aligned}$$

which is upper-bounded by $\frac{0.1}{\ell}$ by our choice of k. Hence, for each $i \in [\ell+1]$, with probability $1 - \frac{(i-1)\cdot 0.1}{\ell}$, it holds that

$$\frac{1}{k} \cdot \sum_{j \in [k]} m^{(i-1,j)} \cdot N_{y^{(i-1,j)}} = \left(1 \pm \frac{e)}{\ell}\right)^{i-1} \cdot N_\lambda \tag{8}$$

which establishes Eq. (7) in concrete terms. Hence, with probability at least 0.9, the output (i.e., $\sum_{j \in [k]} m^{(\ell,j)}/k$) is an $1 \pm \epsilon$ factor approximation of the number of t-cliques in the input graph. Observing that, in each of the ℓ iterations, we make $2k$ oracle calls (to the error-reduced version of A), the theorem follows. ■

Digest. In a sense, the proof of Theorem 5.2 applies the reductions of *fine approximate counting to uniform generation* and of *uniform generation to coarse approximate counting* at the level of each iteration. In contrast, in Section 4, these two reductions were applied at the level of the problem itself (see Theorem 4.3 as well as Section 4.1). Note the analogy between Claim 5.1 and the argument used within the proof of Theorem 4.3.

6 The story behind this memo

As noted in [7], I started thinking about these computational problems while being unaware of the prior work. The starting point of my thoughts was the *reduction of approximate-counting solutions for NP-search problems to decision problems in NP* and the *tight relationship between approximate-counting and uniform generation of solutions for NP-search problems*. These well-known results of [9, 10] and [8], respectively, are among my favorites (see [6, Sec. 6.2]). My goal was to obtain *fine-grained complexity* analogues of these results for the corresponding problems regarding t-cliques. Below I reproduce my original train of thoughts.

As mentioned in the main text, in the context of NP, the standard reduction of approximate counting to decision is based on hashing and on the fact that fitting a specific image under a hash function can be encoded into the NP-complete problem. The first observation is that replacing the hashing by a (careful) random selection of vertices yields a coarse approximation of the number of t-cliques (when using an oracle for deciding the existence of t-cliques).

Recalling that the standard reduction of uniform generation to approximate counting is insensitive to the quality of the approximation (provided it is fine and above some level), I observed that any *coarse* approximate counter of t-cliques yields a *weak* procedure for uniform generation of t-cliques, which can then be *amplified*, and in turn yield a *fine* approximate counter of t-cliques. Here, as well as in the opposite direction, problems regarding t-cliques that fit a given prefix are reduced to problems regarding t-cliques (proper). Fortunately, the latter auxiliary problems can be reduced to the orignal ones.

Originally, I was hoping to improve the coarse approximate counter of t-cliques into a fine one. As noted in the main text, the acute source of coarseness is that the analysis of Algorithm 3.1 only

capitalizes on the best choice of a sequence of indices $(i_1, ..., i_t)$ (which corresponds to the "heaviest bucket"). I failed in my attempts to benefit from all the sequences of indices.

At this point, I became aware of previous works on the subject (specifically, the sequence [1, 4, 2, 5, 3]); in particular, I found out that Dell *et al.* [5] discovered a direct and superior method of reducing fine approximate counting of t-cliques to coarse approximate counting of t-cliques, and that the coarse approximate counter that I "discovered" was discovered before by Bhattacharya *et al.* [3]. So I decided to turn my working draft into a survey.

I believe that the fact that my starting points and conceptual frameworks are somewhat different than those of the authors of [5, 3] is responsible for the differences in the presentation style as well as for some quantitative improvements.

Acknowledgments

I am grateful to Amir Abboud for helpful discussions and comments.

References

[1] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. In *9th ITCS*, pages 38:1–38:21, 2018.

[2] Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge Estimation using Polylogarithmic Subset Queries. `CoRR abs/1908.04196`, 2019.

[3] Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Faster Counting and Sampling Algorithms Using Colorful Decision Oracle. In *39th STACS*, pages 10:1–10:16, 2022.

[4] Holger Dell and John Lapinskas. Fine-Grained Reductions from Approximate Counting to Decision. In *50th STOC*, pages 281–288, 2018. *ACM Trans. Comput. Theory*, Vol. 13 (2), pages 8:1–8:24, 2021.

[5] Holger Dell, John Lapinskas, and Kitty Meeks. Approximately Counting and Sampling Small Witnesses Using a Colourful Decision Oracle. In *31st SODA*, pages 2201–2211, 2020. *SIAM J. Comput.*, Vol. 51 (4), pages 849–899, 2022.

[6] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[7] Oded Goldreich. On approximately counting t-cliques. Unpublished manuscript, 2023. Available from `https://www.wisdom.weizmann.ac.il/~oded/MC/cnt-cliq.pdf`

[8] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *TCS*, Vol. 43, pages 169–188, 1986.

[9] Michael Sipser. A Complexity Theoretic Approach to Randomness. In *15th STOC*, pages 330–335, 1983.

[10] Larry J. Stockmeyer. The Complexity of Approximate Counting. In *15th STOC*, pages 118–126, 1983.