I started thinking about this problem not realizing that much work has been done about it already. In particular, the sequence of works [4, 2, 5, 3] contains all the ideas that I came-up with. Nevertheless, given that I drafted this memo while thinking about the problem and that my perspectives are somewhat different, I feel that it makes sense to post it as is (modulo some postscripts that are clearly marked as such).

The starting point of my thoughts was the reduction of approximate-counting solutions for NP-search problems to NP and the relationship between the approximate-counting and uniform generation of solutions. These well-known results of [8, 9] and [7], respectively, are among my favorites (see [6, Sec. 6.2]).

What is described in this memo is a crude approximation for the number of $t$-cliques in an $n$-vertex graph (which is very similar to the proof of [3, Thm. 1.3], although the perspective offered here is different). This (almost linear-time randomized) approximation algorithm uses $O(\log n)^t$ calls to a decision procedure (for the existence of $t$-cliques in graphs), and yields an approximation up to a factor of $O(\log n)^t$. In addition, the memo contains a sketch of the fine-grained equivalence between approximate counting and uniform generation of $t$-cliques in graphs (which is very similar to the proof of [5, Thm. 2]).

Let me stress that the contents of [4, 2, 5, 3] goes beyond $t$-cliques, which is (or may be) viewed as a special case. Furthermore, the oracles used in these works are easily implemented in the context of $t$-cliques. Specifically, these works are within a framework that refers to $t$-uniform hypergraphs and to oracle calls that answer queries regarding induced substructures of the input graphs (cf. [1, 4, 5]), where in case of $t$-clique these queries concern the existence of $t$-clique in some induced subgraphs (of the input graph).

## Contents



# 1 Crude approximation for $t$-cliques

Our main and first aim is to reduce approximately counting of the number of $t$-cliques in a graph to deciding whether a graph has $t$-cliques. In the context of NP, such reductions are randomized, and we aim for the same here.

## 1.1 The starting point

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, selecting each vertex with probability $\sqrt{1/m}$ is 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.

First, 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 there are at least $n' = 10 \cdot 2^i$ vertices of degree at least $m'/n'\ell = \omega(m/2^i)$. 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 crude, but this can be improved by a more sophisticated scheme (*postscript*: which I was hoping to develop). A minor issue is that we were handling bipartite graphs rather than general graphs, but this is easy to fix (see below). More importantly, the foregoing idea generalize to any $t \geq 2$, and can also be applied for reducing approximate counting of hyper-edges in $t$-uniform hypergraphs to detecting hyper-edges in such hypergraphs.

**Reducing the general case 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. Then, for each $i \neq j$ and $u \neq v$, if $\{u, v\}$ is an edge in the original graph, then we connect the $i^{\text{th}}$ copy of $u$ with the $j^{\text{th}}$ copy of $v$. In other words, for each $i \neq j$, we place a double-cover of the original graph between the $i^{\text{th}}$ part and the $j^{\text{th}}$ part. In the context of counting $t$-cliques, it is important to note that the number of $t$-cliques in the resulting $t$-partite graph is $t!$ times the number of $t$-cliques in the original graph.

## 1.2 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 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 1.1** (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 $i_1, ..., t_t \in \{0, 1, ..., \ell - 1\}$, 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 $\cup_{j \in [t]} S_{i_j}$ 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.*

Algorithm 1.1 makes $O(\log n)^t$ calls to a decision procedure (for deciding the existence of a $t$-clique in various induced subgraphs). It is labeled "take 1" because it is quite wasteful, and can be easily improved. But let us analyze it first.

**Claim 1.2** (upper-bounding the output of Algorithm 1.1): *Suppose that the number of $t$-cliques in $G = ([n], E)$ is $m$. Then, with probability at least 5/6, Algorithm 1.1 outputs a non-negative integer that does not exceed $(6 \cdot \log_2^t n) \cdot m$.*

**Proof:** Algorithm 1.1 outputs the value $m'$ 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 $\ell^t$, the claim follows. ■

**Claim 1.3** (lower-bounding the output of Algorithm 1.1): *Suppose that the number of $t$-cliques in $G = ([n], E)$ is $m \geq 1$. Then, with probability at least 5/6, Algorithm 1.1 outputs an integer that is at least $\lceil m/O(\log n)^{t-1} \rceil$.*

**Proof:** 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 $\sum_{i=0}^{\ell-1} c \cdot 2^i \cdot \frac{m}{2^i \ell}$).[1] 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 iteration that corresponds to $(i_1, i_2)$. We observe that, with probability at least $1 - \exp(-c)$, some vertex of degree at least $\frac{m}{2c2^{i_1}\ell} = c2^{i_2}$ was selected, and with probability at least $1 - \exp(-c)$, one of its neighbors was selected. Hence, with probability at least $(1 - \exp(-c))^2 > 5/6$, the value $2^{i_1} \cdot 2^{i_2} \geq m/4c^2\ell$ was 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, for each such vertex $v$, we may consider the subgraph that is induced by its neighbors, and apply the same reasoning to this induced subgraph, which is $(t-1)$-partite.

[1] Specifically, we partition the vertices (of the first part) into buckets such that the $j^{\text{th}}$ bucket, denoted $B_j$, contains all vertices of degree in $[2^j\ell, 2^{j+1})$. 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 $i = \log_2(m/2c\ell) - j$).

We view the $\ell^t$ trials of Algorithm 1.1 as being arranged in a $\ell$-ary tree of depth $t$ such that each internal node of level $j \in \{0, 1, ..., t-1\}$ corresponds to choice of $i_1, ..., i_j$, and in such a node (if $j < t$) we branch to all possible values of $i_{j+1}$. Given this perspective, we consider the branch of the root that corresponds to the foregoing value of $i_1$. Then, with probability at least $1 - \exp(-c)$, some vertex of clique-degree at least $m_1$ was selected. Fixing this vertex $v$, we consider the $(t-1)$-partite subgraph induced by $v$'s neighbors, and approximate the number of $(t-1)$-cliques in this subgraph. Observing that Algorithm 1.1 is monotone (i.e., in each execution (i.e., per each choice of randomness), the output can only decrease when edges are omitted), and assuming that (an adequate version of) the claim holds for $t-1$, the claim would hold for $t$. Specifically, we refer to the following induction claim.

> Induction claim: Suppose that the number of $t$-cliques in a subgraph of a $t$-partite $n$-vertex graph $G$ is $m \geq 1$. Then, for every $c \geq 3$, with probability at least $1 - t \cdot \exp(-c)$, on input $G$, Algorithm 1.1 outputs a value that exceeds $m/((2c)^t \cdot \log_2^{t-1} n)$.
>
> Induction step: Let $i_1$ be as above and suppose that $v$ is a vertex of clique-degree at least $m_1 = \frac{m}{2c2^{i_1}\ell}$. Then, we consider an execution of Algorithm 1.1 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 $t-1$. (Actually, we consider an execution on an $n$-vertex $(t-1)$-partite graph that contains this induced subgraph.) By the induction hypothesis, with probability at least $1-(t-1)\cdot\exp(-c)$, this execution outputs 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 output by the execution on $G$ itself exceeds
>
> $$\begin{aligned} 2^{i_1} \cdot \frac{m_1}{(2c)^{t-1} \cdot \log_2^{t-2} n} &= 2^{i_1} \cdot \frac{m/2c2^{i_1}\ell}{(2c)^{t-1} \cdot \log_2^{t-2} n} \\ &= \frac{m}{(2c)^t \cdot \log_2^{t-1} n} \end{aligned}$$

Having established the base case (i.e., $t = 2$), 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 $t \geq 3$)). The main source of waste (w.r.t the approximation factor) is the fact that 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 "crude bucketing" (i.e., using 2 rather than $1+\epsilon$ as a base). Lastly, Claim 1.2 is based on Markov Inequality, whereas a better estimate can be provided by repeating each trial $O(\log(\ell^t)) = O(t \log\log n)$ times and using a Chernoff bound.

**Postscript.** I failed in my attempt to design a scheme that will use all buckets rather than only the heaviest one. The other two issues are easy to handle, but I see no point in doing this now. Note that even if my ideas would have worked out, the resulting approximation algorithm would have made $O(\epsilon^{-1} \log n)^{t-1}$ calls to a decision oracle, whereas [3, Thm 1.4] make $O(\log n)^{3k+5}/\epsilon^2$ such calls. By the way, the scheme described above is very similar to the one that underlies the proof of [3, Thm. 1.3]

## 2 Approximate counting versus uniform generation

Following [7], we related approximate counting of $t$-cliques in $t$-partite $n$-vertex graphs to uniform generation of $t$-cliques in such graphs. Both reductions proceed by considering the (depth $t \log_2 n$) binary tree of all possible $t$-cliques in which the various 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 their next move along a path that leads from the root to some vertex.

On the one hand, approximate counting of the number of $t$-cliques that fit a one-bit extensions of a prefix allows for a proportional random selection of the next bit (in the description of a $t$-clique). On the other hand, uniform generation of a $t$-clique that fits a given prefix allows for approximating the number of $t$-cliques that agree with the one-bit extended prefix. 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 is the very issue addressed next.

Let us look at a generic prefix of a potential $t$-clique in an $n$-vertex graph $G$, where $\ell = \lceil \log_2 n \rceil$. Such a prefix has the form $(v_1, ..., v_{t-t'}, \alpha)$, where $v_i$ denotes a vertex in the $i^{\text{th}}$ part of $G$ and $\alpha \in \bigcup_{k \in \{0,1,...,\ell-1\}} \{0,1\}^k$ is a prefix of the description of a vertex (in the $(t-t'+1)^{\text{th}}$ part). Now, the set of $t$-cliques in $G$ that fit the foregoing prefix corresponds to the set of $t'$-cliques in the $t'$-partite graph $G'$ defined as the subgraph induced by the set of vertices that neighbor all $v_i$'s (for every $i \in [t-t']$) and contains in its first part only vertices in the $(t-t'+1)^{\text{st}}$ part of $G$ that start with the prefix $\alpha$. That is, letting $(V_1, ..., V_t)$ be the $t$-partition of $G$, we consider the subgraph of $G$ induced by $V'_{t-t'+1} \cup \bigcup_{i \in [t-t'+2,t]} V_i$ such that $v \in V'_{t-t'+1}$ if and only if $\alpha$ is a prefix of $v$.

**Conclusion.** In direct analogy to [6, Thm. 6.31], we get

**Theorem 2.1** (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 at least negligible, is $O(t \log n)$ larger than the deviation of the uniform generation.*

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

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

## 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] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[7] M. Jerrum, L. Valiant, and V.V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *TCS*, Vol. 43, pages 169–188, 1986.

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

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