Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 31 (2002)

# Approximation Algorithms for some Parameterized Counting Problems

V. Arvind* Venkatesh Raman†

June 3, 2002

**Abstract**

As our main result, we give a randomized fixed parameter tractable algorithm to approximately count the number of copies of a $k$-vertex graph with bounded treewidth in an $n$ vertex graph. As a consequence, we get randomized algorithms with running time $k^{O(k)}n^{O(1)}$, approximation ratio $1/k^{O(k)}$, and error probability $2^{-n^{O(1)}}$ for the following problems:

- Approximately counting the number of matchings of size $k$ in an $n$ vertex graph.
- Approximately counting the number of paths of length $k$ in an $n$ vertex graph.

Our algorithm is based on the Karp-Luby approximate counting technique applied to fixed parameter tractable problems, and the color-coding technique (based on perfect hashing) of Alon, Yuster and Zwick.

It is interesting to note in contrast that the *exact* counting versions of the above two problems are recently shown by Flum and Grohe [6] to be hard for the parameterized complexity class W[1]. Hence no exact counting algorithm with running time $f(k)n^{b+O(1)}$ is likely to exist these two problems, for any computable function $f : \mathcal{N} \to \mathcal{N}$.

We also show some W-hardness results for parameterized exact counting problems, whose decision versions are known to be fixed parameter tractable. These problems include counting versions of weight $k$ satisfying assignment of a bounded DNF formula, and finding a $k$-clique or a $k$-independent set in a graph.

## 1 Introduction

We investigate some counting problems in the framework introduced by Downey and Fellows [3]. In this framework, efficient algorithms are sought for 'parameterized' problems - problems having an input whose size is $n$, and a fixed parameter $k$. Such a parameterized problem is said to be fixed parameter tractable (FPT) if there is an algorithm for the problem that takes $O(f(k)n^{O(1)})$ for some function $f$ of $k$. Examples of fixed parameter tractable problems include VERTEX COVER and UNDIRECTED FEEDBACK VERTEX SET. There is also a 'hardness' theory, and it is known, for example, that the DOMINATING SET and the CLIQUE problems are hard for the parameterized complexity classes $W[2]$ and $W[1]$ respectively. The best known algorithms for these two problems take $n^{O(k)}$ time. In the problems mentioned above, the parameter $k$ is the solution size.[1]

Recently there has been some attention in studying the parameterized complexity of counting versions of the parameterized problems [5, 6, 13]. In particular, Flum and Grohe [6] have shown

*The Institute of Mathematical Sciences, Chennai 600 113, arvind@imsc.ernet.in

†The Institute of Mathematical Sciences, Chennai 600 113, vraman@imsc.ernet.in

[1]The parameter need not always be the solution size; there are problems for which other parameterizations are more natural.



ISSN 1433-8092

that (i) exactly counting the number of paths of length $k$ and (ii) exactly counting the number of matchings of size $k$ in an undirected graph, are both hard problems for the class W[1]. More specifically, this means that a deterministic $f(k)n^{O(1)}$ for the exact counting versions of these problems are unlikely for any computable function $f : \mathcal{N} \to \mathcal{N}$. On the other hand, Alon, Yuster and Zwick [1] have shown that the *decision* problem of checking if a given graph has a path of length $k$ is fixed parameter tractable.

*Summary of new results*

In this paper we first consider the following counting problem:

**Problem 1.1**
**Input**: *An $n$ vertex graph $G$ and $k$-vertex graph $H$ such that $H$ has treewidth $b$.*
**Output**: *Find the number of subgraphs[2] of $G$ isomorphic to $H$.*
**Parameter**: *The parameter for the problem is $k$, which is the size of the graph $H$.*

*Although we treat $b$ as a* constant, *we will explicitly show the occurrence of $b$ in the running time bound for the main result: the approximation algorithm described in Theorem 2.2.*

This is a generic problem which includes various problems as special cases. E.g. the problems of (i) counting matchings of size $k$ in a graph, (ii) counting paths of length $k$, and (iii) counting the number of copies of any fixed tree/forest on $k$ nodes in an $n$-vertex graph.

As our main result, we give a randomized approximate counting algorithm with running time of the form $f(k)n^{b+O(1)})$ for the Problem 1.1 defined above. As a consequence, in one stroke we obtain randomized approximate counting algorithms with similar time bounds for the problems of counting paths of length $k$, matchings of size $k$ etc.

We now explain the main ingredients of our approximation algorithm. First, we reformulate Karp-Luby's Monte-Carlo sampling method for approximate counting [8] (see also [9]) for parameterized problems. We also suitably adapt the notion of FPRAS (fully polynomial randomized approximation schemes) and define fixed parameter tractable randomized approximation schemes (FPTRAS). We then observe, in Section 2, that under similar conditions as in the case of the approximate counting result of Karp and Luby, FPTRAS exist for parameterized counting problems.

Using this and the color coding technique of Alon, Yuster and Zwick [1], we give a randomized approximate counting algorithm taking $O(k^{O(k)}(c^{b^3}k + n^{b+O(1)}2^{b^2/2}))$ time for Problem 1.1, where the approximation ratio $\epsilon$ of the algorithm is $1/k^{O(k)}$ and the error probability $\delta$ is $1/2^{n^{O(1)}}$.

It is interesting to note that the factor $n^b$ appears to be a bottleneck in the above-mentioned time bound for the following reason: since a $k$-clique has treewidth $k$, it can be easily seen that an $n^{o(b)}f(k)$ algorithm for Problem 1.1 would imply, in particular, that there is an $f(k)n^{o(k)}$ algorithm for deciding whether a graph on $n$ vertices has a $k$-clique, which is a long-standing open problem (see e.g. [4] for a detailed discussion on the hardness of finding $k$-cliques in graphs and also see [3]).

In Section 3, we give some $W$-hard results on the *exact* counting versions of some fixed parameter tractable problems. Specifically we show that the problem of counting the number of weight $k$ satisfying assignments of a monotone 2-DNF formula is $W[1]$-hard. This problem has an FPTRAS algorithm as we show in that Section. We also show that the problems of

- counting the number of cliques *and* independent sets on $k$ vertices in a graph[3], and

[2]Notice that we are *not* asking for the number of induced subgraphs but for all subgraphs.

[3]Notice that this problem is different from counting just $k$-cliques or just $k$-independent sets separately. This problem is potentially easier because the decision version is easier [10]!

- counting the number of satisfying assignments of weight at most $k$ in a bounded CNF formula,

are $W[1]$-hard. The decision version of both these problems are fixed parameter tractable, and the approximate counting versions of these problems are open.

### 1.1 Definitions and Notation

For a positive integer $n$, by $[n]$, we mean the set $\{1, 2, \ldots, n\}$.

A tree decomposition (see, for example, [11]) of a graph $G = (V, E)$ is a pair $D = (S, T)$ with $S = \{X_i, i \in I\}$ a collection of subsets of vertices of $G$ and $T = (I, F)$ a tree, with one node for each subset in $S$, such that the following three conditions are satisfied:

1. $\bigcup_{i \in I} X_i = V$,
2. for all edges $(v, w) \in E$, there is a subset $X_i \in S$ such that both $v$ and $w$ are contained in $X_i$,
3. for each vertex $x \in V$, the set of nodes $\{i | x \in X_i\}$ forms a subtree of $T$.

The width of a tree-decomposition $(\{X_i | i \in I\}, T = (I, F))$ is $\max_{i \in I}(|X_i| - 1)$. The treewidth of a graph $G$ is the minimum width over all tree-decompositions of $G$.

For a subset $X$ of vertices of $G$, by $G[X]$, we mean the subgraph of $G$ induced by $X$. I.e $G[X]$ is the subgraph that contains all the vertices of $X$ and all the edges of $G$ incident on the vertices in $X$.

For a problem $\Pi$, let $\#(I)$ be the number of distinct solutions for an instance $I$ of $\Pi$. A fully polynomial randomized approximation scheme (FPRAS) (see, for example [14]) for a counting problem $\Pi$ is a randomized algorithm $A$ that takes an input instance $I$ with $|I| = n$, and real numbers $\epsilon > 0$ and $0 < \delta < 1$, and in time polynomial in $n$, $1/\epsilon$, and $\lg 1/\delta$. produces an output $A(I)$ such that

$$\text{Prob}[(1 - \epsilon)\#(\text{I}) \leq \text{A(I)} \leq (1 + \epsilon)\#(\text{I})] \geq 1 - \delta.$$

We define an FPTRAS for a parameterized counting problem $\Pi$ with the parameter $k$ as a randomized approximation scheme that takes an input instance $I$ with $|I| = n$, and real numbers $\epsilon > 0$ and $0 < \delta < 1$, and computes an $\epsilon$-approximation to $\#(I)$, with probability at least $1 - \delta$ in time $f(k)g(n, 1/\epsilon, \lg(1/\delta))$ where $f$ is any function of $k$ and $g$ is polynomially bounded in $n, 1/\epsilon$ and $\log 1/\delta$.

## 2 Randomized Approximate Counting Parameterized Solutions

We first give a parameterized version of the result of Karp-Luby [8] on approximate counting via random sampling.

**Theorem 2.1** *For every positive integer $n$ every integer $0 \leq k \leq n$, let $U_{n,k}$ be a finite universe, whose elements are strings of length $n^{O(1)}$ encoded in binary. Let $\mathcal{A}_{n,k} = \{A_1, A_2, \ldots A_m\} \subseteq U_{n,k}$ be a collection of $m$ given sets, and let $g : \mathcal{N} \to \mathcal{N}$ be a function and let $d > 0$ be a constant with the following conditions.*

1. *There is an algorithm that computes $|A_i|$ in time $g(k)n^d$, for each $i$, and every $\mathcal{A}_{n,k}$.*
2. *There is an algorithm that samples uniformly at random from $A_i$ in time $g(k)n^d$, for each $i$, and every $\mathcal{A}_{n,k}$.*

3. *There is an algorithm that takes $x \in U_{n,k}$ as input and determines whether $x \in A_i$ in time $g(k)n^d$, for each $i$, and every $\mathcal{A}_{n,k}$.*

*Then there is an FPTRAS for estimating the size of $A = A_1 \cup A_2 \cup \ldots A_m$ whenever $m$ is $l(k)n^{O(1)}$ for some function $l$. In particular, for $\epsilon = 1/g(k)$, and $\delta = 1/2^{n^{O(1)}}$, the running time of the FPTRAS algorithm is $g(k)^{O(1)}n^{O(1)}$.*

*Proof:* We omit the proof of this theorem as it can be proved exactly on the same lines as original Karp-Luby result [14]. □

Our goal in this section is to use Theorem 2.1 and design an FPTRAS algorithm for estimating the number of copies of a graph $H$ on $k$ vertices with treewidth bounded by $b$ in a given graph $G$ on $n$ vertices. We first give a high level description of the overall algorithm and explain how we use Theorem 2.1. Let

$$A = \{K \mid K \text{ is a } k\text{-vertex subgraph of } G \text{ such that } K \text{ is isomorphic to } H\}.$$

Our goal is to estimate $|A|$. We will express $A$ as a union $\bigcup_{i=1}^{m} A_i$ that fulfills the conditions of Theorem 2.1 and thereby we will get an FPTRAS algorithm for the problem.

*Defining the $A_i$'s*

A $k$-coloring of the graph $G$ is a mapping $f : V(G) \longrightarrow [k]$. Our interest is in $k$-colorings of $G$ that injectively map $k$ element subsets of $V(G)$. Towards this goal, we consider a family $\mathcal{F}$ of (perfect hash) functions from $[|V(G)|] \longrightarrow [k]$ that satisfies the following crucial property:

$$\forall S \subseteq V(G) \ : \ |S| = k \ \exists \ f \in \mathcal{F} \ : \ f \text{ is injective on } S.$$

Such a family $\mathcal{F}$ of hash functions with $|\mathcal{F}| = 2^{O(k)} \log^{O(1)} n$, where $n = |V(G)|$ exists [1].

Suppose $G$ is $k$-colored by some $f \in \mathcal{F}$. We say that a subset $S$ of vertices of $G$ is *colorful* under $f$ if $f$ restricted to $S$ is injective (i.e. $f(i) \neq f(j)$ whenever $i \neq j \in S$). A subgraph $H$ of $G$ is *colorful* under $f$ if $V(H)$ is colorful under $f$.

The index set $\mathcal{I}$ defining the collection $\{A_i\}_{i \in \mathcal{I}}$ is the following:

$$\mathcal{I} = \{\langle f, \pi \rangle \mid f \in \mathcal{F} \text{ and } \pi \text{ is a } k\text{-coloring of } H \text{ in which every vertex of } H \text{ is distinctly colored}\}.$$

Notice that if we identify $V(H)$ with $[k]$ then $\pi$ can be seen as a permutation on $[k]$.

We need one more definition: let $K_1$ and $K_2$ be two $k$-colored graphs (by colors from $[k]$). We say that $K_1$ and $K_2$ are *color-preserving isomorphic* if there is an isomorphism between them that also preserves the colors. Notice that if both $K_1$ and $K_2$ are $k$-vertex graphs with $k$-colorings and additionally $K_2$ is colorful, then there is an efficient algorithm to test if they are color-preserving isomorphic: we simply need to check that $K_1$ is colorful, and verify that the only color preserving mapping from $V(K_1)$ to $V(K_2)$ is an isomorphism. Now, for each $i = \langle f, \pi \rangle \in \mathcal{I}$ we define the set $A_i$ as follows:

$$A_i = \{K \mid K \text{ is a colorful } k\text{-vertex subgraph of } G \text{ under the coloring } f \text{ and } K \text{ is color-preserving isomorphic to } H \text{ colored by } \pi\}.$$

It is easy to verify that $A = \bigcup_{i \in \mathcal{I}} A_i$. Furthermore, $|\mathcal{I}| = 2^{O(k)} \cdot \log^{O(1)} n \cdot k!$. Thus it suffices to verify the three conditions necessary to apply Theorem 2.1 and we get the required FPTRAS algorithm.

Now, we begin with a lemma to prove the first condition.

**Lemma 2.1** *Let $G = (V, E)$ be a graph on $n$ vertices that is $k$-colored by some coloring $f : V(G) \longrightarrow [k]$, and let $H$ be a $k$-vertex graph of treewidth $b$ that is $k$-colored by some coloring $\pi$ such that $H$ is colorful. Then there is an algorithm taking time $O(c^{b^3}k + n^{b+2}2^{b^2/2})$ time to exactly compute the cardinality of the set $\{K \mid K$ is a $k$-vertex subgraph of $G$ and $K$ is color-preserving isomorphic to $H\}$, where $c > 0$ is some constant.*

*Proof:* Since $H$ has treewidth $b$, we can find a rooted tree-decomposition $D = (S, T)$ of $H$, with $S = \{X_i, i \in I\}$ and the rooted tree $T = (I, F)$ satisfying the following properties in time $c^{b^3}k$ (see [2] and [11, Lemma 13.1.3] for details).

1. $|I| \leq 4k$ and every node of $T$ has at most two children.
2. If a node $i$ has two children $j$ and $k$, then $X_i = X_j = X_k$.
3. If a node $i$ has one child $j$, then either $X_i = X_j \cup \{x\}$ or $X_j = X_i \cup \{x\}$ for some element $x \in V(H)$.

Furthermore, we know that each $|X_i| \leq b + 1$ for each $i \in I$. Such a tree decomposition is said to be a *nice rooted tree decomposition*, and we will call the set associated with a node $i$ of the tree as a *bag*.

Our goal is to count the number of colorful subgraphs $K$ of $G$ such that $K$ is color-preserving isomorphic to $H$. We will do this by counting subgraphs of $G$ that are color-preserving isomorphic to *subgraphs* of $H$ realized by subtrees of the tree decomposition $T$. As $T$ is a rooted tree, we can do this in a bottom-up fashion. Let the depth of a node in $T$ be the length (the number of edges) in the path from the root to that node (the depth of the root, therefore, is 0). We will process nodes of $T$ inductively in non-increasing order of their depths, i.e. in a bottom up fashion, starting from the leaves of $T$.

Let $y$ be a node of the tree $T$ and let $T_y$ be a subtree rooted at $y$. Let $H_y$ be the induced subgraph $H[V_y]$ where $V_y = \bigcup_{x \in T_y} X_x$. Then it is known [11, Lemma 13.1.1] that the nodes of $T_y$ along with their bags form a tree decomposition for $H_y$. For every node $y$ of $T$ and for each subgraph $K$ of $G$ with $|X_y|$ nodes (which is at most $b + 1$), we define the set

$S(H_y, K) = \{K' \mid K'$ contains $K$ as an induced subgraph, and $K'$ is color-preserving isomorphic to $H_y$ with the nodes of $H_y$ in the bag $X_y$ mapped to the subset $V(K)$ of $V(G)$ by the color-preserving isomorphism$\}$. Let $N(H_y, K)$ denote its cardinality $|S(H_y, K)|$.

At the time of processing a node $x$ of the tree $T$, we assume inductively that we know $N(H_y, K)$ for every subgraph $H_y$ corresponding to each child $y$ of $x$ in $T$, and for every subgraph $K$ of $G$ with $|X_y|$ vertices. This assumption is trivially true when processing leaf nodes of $T$. To prove the inductive step, we make use of the properties of the nice tree decomposition.

At the inductive step, let $x$ be the node of $T$ to be processed and let $K$ be any subgraph of $G$ with $|X_x|$ vertices. Clearly, $N(H_x, K) = 0$ if $K$ is not colorful. If $K$ is colorful then it is easy to check if there is a color-preserving isomorphism from $K$ to $H[X_x]$, the subgraph of $H$ induced by $X_x$. If there is no color-preserving isomorphism from $K$ to $H[X_x]$ then also $N(H_x, K) = 0$. Thus we need to only consider the case when $K$ is a subgraph of $G$ with $|X_x|$ nodes such that there is a color-preserving isomorphism from $K$ to $H[X_x]$. Then there are two cases depending on whether the node $x$ in $T$ has one or two children.

Case 1 *$x$ has two children $y$ and $z$*

In this case, by the property of nice tree decomposition, we have $X_x = X_y = X_z$. By induction hypothesis, we already know $N(H_y, K)$ and $N(H_z, K)$. We claim that $N(H_x, K) =$

$N(H_y, K) \times N(H_z, K)$. To see this it suffices to show that there is a bijective correspondence between $S(H_x, K)$ and the Cartesian product $S(H_y, K) \times S(H_z, K)$. We show this by first injectively mapping $S(H_x, K)$ into $S(H_y, K) \times S(H_z, K)$ and vice-versa.

Let $K'$ be in $S(H_x, K)$, where $h$ is the color-preserving isomorphism from $K'$ to $H_x$ such that $h$ maps $X_x$ to $V(K)$. By the properties of nice tree decompositions we have

$$V(H_x) \ = \ V(H_y) \cup V(H_z) \text{ and } X_x \ = \ V(H_y) \cap V(H_z)$$

Let $K'_1$ and $K'_2$ be the subgraphs of $K'$ induced by $V(H_y)$ and $V(H_z)$ respectively. Then it is clear that $h$ restricted to $K'_1$ gives a color-preserving isomorphism from $K'_1$ to $H_y$, and $h$ restricted to $K'_2$ gives a color-preserving isomorphism from $K'_2$ to $H_z$ (the additional properties that $X_y$ is mapped to $V(K)$ and $X_z$ is mapped to $V(K)$ holds obviously for the restrictions of $h$ as $X_x = X_y = X_z$). The mapping $K \mapsto (K'_1, K'_2)$ is clearly injective.

Conversely, suppose $(K'_1, K'_2) \in S(H_y, K) \times S(H_z, K)$. Let $h_y$ be the color-preserving isomorphism from $K'_1$ to $H_y$, and let $h_z$ be the color-preserving isomorphism from $K'_2$ to $H_z$ such that $h_y$ maps $V(K)$ to $X_y$ and $h_z$ maps $V(K)$ to $X_z$. Recall that $H$ is colorful under coloring $\pi$. Therefore, both $h_y$ and $h_z$ must coincide on $V(K)$. By the tree decomposition property, $V(H_x) \ = \ V(H_y) \cup V(H_z)$ and $X_x \ = \ V(H_y) \cap V(H_z)$. Thus, the mappings $h_y$ and $h_z$ in fact force $V(K'_1) \cap V(K'_2) = V(K)$. Now, let $K'$ be the subgraph of $G$ defined as follows: $V(K') = V(K'_1) \cup V(K'_2)$ and $(x, y) \in E(K')$ if and only if $(x, y) \in E(K'_1)$ or $(x, y) \in E(K'_2)$. It is easy to see that we can define a color-preserving isomorphism $h$ from $V(K')$ to $H_x$ by letting $h = h_y$ on $V(K'_1)$ and $h = h_z$ on $V(K'_2)$. Again, the mapping $(K'_1, K'_2) \mapsto K$ is easily seen to be injective. This completes Case 1.

Case 2 : *$x$ has one child $y$ and $X_y = X_x - \{j\}$ for some $j \in V(H)$.*

Firstly, by tree decomposition property, $j \notin V(H_y)$. Thus, $V(H_x) = V(H_y) \cup \{j\}$. Also, tree decomposition guarantees that in $H_x$, the node $j$ can be adjacent to only nodes in $X_x$ (because the edge $(j, k)$ must lie in some bag, and that bag must be $X_x$ for otherwise $j$ would lie in some other bag forcing it to belong to $X_y$ which is not possible).

Now, let $h$ be the unique color-preserving isomorphism mapping $K$ to the subgraph of $H$ induced by $X_x$. Let $K_1 = K - \{h^{-1}(j)\}$. Let $K' \in S(H_x, K)$. Clearly, it follows that $K' - \{h^{-1}(j)\}$ is in $S(H_y, K_1)$. Conversely, if $K' \in S(H_y, K_1)$ then there is a color-preserving isomorphism $g$ from $K'$ to $H_y$ that maps $V(K_1)$ to $X_y$. Since $j$ is adjacent to only nodes in $X_x$ and since only $h^{-1}(j)$ in $V(K)$ can be mapped to $j \in X_x$, it is clear that $g$ can be extended to a color-preserving isomorphism from $K' \cup \{h^{-1}(j)\}$ to $H_x$ by mapping $h^{-1}(j)$ to $j$.

Thus, $K' \mapsto K' \cup \{h^{-1}(j)\}$ is a bijection from $S(H_y, K_1)$ to $S(H_x, K)$. Therefore, $N(H_x, K)$ is the same as $N(H_y, K - \{h^{-1}(j)\})$ which is already computed by induction hypothesis.

Case 3 : *$x$ has one child $y$ and $X_y = X_x \cup \{j\}$ for some $j \in V(H)$.*

By induction hypothesis we already have computed $N(H_y, K')$ for all subgraphs $K'$ with $|X_y|$ vertices of $G$. Now, let $K$ be a subgraph of $G$ with $|X_x|$ vertices and let $h$ be the color preserving isomorphism from $K$ to the subgraph of $H$ induced by $X_x$. Let the color of $j$ in $H$ under the coloring $\pi$ be $c \in [k]$. Notice that we have

$$N(H_x, K) = \sum_{v \in V(G)} N(H_y, K \cup \{v\})$$

This is a direct consequence of $S(H_x, K) = \bigcup_{v \in V(G)} S(H_y, K \cup \{v\})$. In fact, in the above sum, only those $N(H_y, K \cup \{v\})$ are nonzero for which $v$ is colored $c$ in $G$.

As we have inductively computed $N(X_y, K \cup \{v\})$ for each $v$, we can add them up to get $N(H_x, K)$.

Finally, notice that we can compute the nice tree decomposition of $H$ in time $c^{b^3}k$ for some constant $c$ (see [2] and [11, Lemma 13.1.3]).

In cases 1 and 2, it takes $O(1)$ time (in the standard RAM model) to compute $N(H_x, K)$ given $N(H_y, K)$ and $N(H_z, K)$ for each subgraph $K$. So for all subgraphs $K$ on colorful sets of size $|X_x|$, it takes at most $O(n^{b+1}2^{\binom{b+1}{2}})$ time (as the number of such sets can be $O(n^{b+1})$ and the number of subgraphs on a $b+1$ element vertex set is at most $2^{\binom{b+1}{2}}$).

In case 3, it takes $O(n)$ time to compute $N(H_x, K)$ given $N(H_y, K')$. However, in case 3, $|X_x| \leq b$ as $|X_y| = |X_x| + 1 \leq b+1$. So to compute $N(H_x, K)$ for all subgraphs $K$ on $|X_x|$ vertices, it would take $O(n^{b+1}2^{\binom{b}{2}})$ time.

Since $T$ has at most $4k$ nodes, the claimed running time follows. □

**Lemma 2.2** *Let $G = (V, E)$ be a graph on $n$ vertices that is $k$-colored by some coloring $f : V(G) \longrightarrow [k]$, and let $H$ be a $k$-vertex graph of treewidth $b$ that is $k$-colored by some coloring $\pi$ such that $H$ is colorful. Then there is an algorithm taking time $O(c^{b^3}k + n^{b+O(1)}2^{b^2/2})$ time to sample uniformly at random from the set $\{K \mid K$ is a colorful $k$-vertex subgraph of $G$ under the coloring $f$ and $K$ is color-preserving isomorphic to $H$ colored by $\pi\}$.*

*Proof:* Using the algorithm in the proof of Lemma 2.1, we will first compute $N(H_x, K)$ for each node $x$ of the nice tree decomposition $T$ and for each subgraph $K$ on a colorful subset of $V(G)$ with $|X_x|$ vertices. Let $r$ be the root of $T$. The uniform random sampler will make use of the rooted tree structure of $T$ (like Lemma 2.1). For every vertex $x$ of $T$ let $k_x$ denote the number of nodes in $H_x$. Clearly, $k_r = k$. Now, define $S_x$ as $S_x = \{K \mid K$ is a colorful $k_x$-vertex subgraph of $G$ under the coloring $f$ and $K$ is color-preserving isomorphic to $H_x$ colored by $\pi\}$. In general, we will explain how to efficiently sample from each $S_x$ (and hence from the desired set $S_r$). The random sampling proceeds inductively using the tree structure: if $x$ is a leaf node in $T$, then random sampling from $S_x$ can be easily done by brute force in $n^{b+O(1)}$ time as $H_x$ has at most $b+1$ nodes. In order to sample uniformly from $S_x$, we pick a subgraph $K$ on a colorful subset of $V(G)$ on $|X_x|$ vertices with probability $\frac{N(H_x,K)}{\sum_{K:|V(K)|=|X_x|} N(H_x,K)}$. It is clear that if we can sample uniformly from $S(H_x, K)$ for each $K$ then we have uniform sampling from $S_x$. We will describe a bottom-up method for sampling uniformly from $S(H_x, K)$ for each $K$. More specifically, it suffices to show we can efficiently sample uniformly from $S(H_x, K)$, assuming that we can uniformly sample from $S(H_y, K')$ for each child $y$ of $x$ in $T$ and each subgraph $K'$ on $|X_y|$ vertices in $G$.

We need to consider the three cases for children of $x$:

If $x$ has two children $y$ and $z$, then the bijective mapping shown in Lemma 2.1 for Case 1, between $S(H_x, K)$ and $S(H_y, K) \times S(H_z, K)$, shows immediately that uniformly sampling from $S(H_y, K)$ and $S(H_z, K)$ gives uniform sampling from $S(H_x, K)$.

If $x$ has one child $y$, with $X_x = X_y \cup \{j\}$ then we argue using Case 2 of Lemma 2.1. Let $h$ be the unique color-preserving isomorphism mapping $K$ to the subgraph of $H$ induced by $X_x$. Let $K_1 = K - \{h^{-1}(j)\}$. Let $K' \in S(H_x, K)$. Then, $K' \mapsto K' \cup \{h^{-1}(j)\}$ is a bijection from $S(H_y, K_1)$ to $S(H_x, K)$. Thus, uniform sampling from $S(H_y, K_1)$ yields uniform sampling from $S(H_x, K)$.

Finally, if $x$ has one child $y$, with $X_y = X_x \cup \{j\}$ then we argue like in Case 3 of Lemma 2.1. We have that in that case $S(H_x, K) = \bigcup_{v \in V(G)} S(H_y, K \cup \{v\})$, where the union is a *disjoint* union.

Thus, if we can uniformly sample from each $S(H_y, K \cup \{v\})$ then we can design a uniform sampling procedure for $S(H_x, K)$ by first randomly picking $v \in V(G)$ with probability $\frac{N(H_y, K\cup\{v\})}{N(H_x,K)}$ and then sampling uniformly from $S(H_y, K \cup \{v\})$.

Thus, putting it together, we have an inductive procedure (following the inductive structure of Lemma 2.1) that samples uniformly from $S_r$ as desired. It is easy to see that the running time of the algorithm is $O(c^{b^3}k + n^{b+O(1)}2^{b^2/2})$. □

Now we are ready to prove our main result.

**Theorem 2.2** *Let $G$ be a graph on $n$ vertices $\{1, 2, \ldots n\}$ and let $H$ be a graph on $k$ vertices $\{1, 2, \ldots k\}$ having tree width $b$. Then there is an FPTRAS algorithm to approximate the number of copies of $H$ in $G$. Specifically, the algorithm has running time $k^{O(k)}(c^{b^3}k + n^{b+O(1)}2^{b^2/2})$ with $\epsilon = 1/k^{O(k)}$ and $\delta = 1/2^{n^{O(1)}}$.*

*Proof:*

Let $A = \{K \mid K$ is a $k$-vertex subgraph of $G$ such that $K$ is isomorphic to $H\}$. Our goal is to approximate $|A|$. We have written $A$ as $\bigcup_{i\in\mathcal{I}} A_i$, where $\mathcal{I} = \{\langle f, \pi\rangle \mid f \in \mathcal{F}$ and $\pi$ is a $k$-coloring of $H$ in which every vertex of $H$ is distinctly colored$\}$. For each $i = \langle f, \pi\rangle \in \mathcal{I}$:

$A_i = \{K \mid K$ is a colorful $k$-vertex subgraph of $G$ under the coloring $f$ and $K$ is color-preserving isomorphic to $H$ colored by $\pi\}$.

Notice that by Lemma 2.1 $|A_i|$ can be computed in time $c^{b^3}k + n^{b+2}2^{b^2/2}$ for each $i \in \mathcal{I}$, and by Lemma 2.2 we can uniformly sample from each $A_i$ in time $c^{b^3}k + n^{b+O(1)}2^{b^2/2}$. Lastly, given any subgraph $K$ it can be checked if $K$ is in $A_i$ in time $k^2$ as we just have to check if $K$ is colorful under coloring $f$ and that $K$ is color-preserving isomorphic to $H$.

Since $|\mathcal{I}| = 2^{O(k)} \cdot \log^{O(1)} n \cdot k!$ and the three conditions are satisfied by the collection $\{A_i\}_{i\in\mathcal{I}}$, we can apply Theorem 2.1 to get the desired randomized approximation algorithm for $|A|$. □

The following corollary is an immediate consequence of Theorem 2.2.

**Corollary 2.1** *Let $G$ be a graph on $n$ vertices and let $H$ be a forest on $k$ vertices. Then there is an FPTRAS that for $\epsilon = 1/k^{O(k)}$ and $\delta = 1/2^{n^{O(1)}}$ has running time $k^{O(k)}n^{O(1)}$ to approximate the number of copies of $H$ in $G$. As a consequence, for the same $\epsilon$ and $\delta$, and with running time $k^{O(k)}n^{O(1)}$, there is an FPTRAS for the following:*

1. *The number of matchings of size $k$ in $G$.*
2. *The number of paths of length $k$ in $G$.*

We can show the following theorem for directed graphs along the same lines as Theorem 2.2 (and an immediate corollary).

**Theorem 2.3** *Let $G$ be a directed graph on $n$ vertices $\{1, 2, \ldots n\}$ and let $H$ be a directed graph on $k$ vertices $\{1, 2, \ldots k\}$, such that the underlying undirected graph has treewidth $b$.[4] Then there is an FPTRAS algorithm to approximate the number of copies of $H$ in $G$.*

**Corollary 2.2** *Given a directed graph $G$ on $n$ vertices, there is an FPTRAS with running time $k^{O(k)}n^{O(1)}$ for $\epsilon = 1/k^{O(k)}$ and $\delta$, inverse exponential in $n$, for the following problems:*

1. *Given an arborescence $H$ on $k$ vertices, to count the number of copies of $H$ in $G$.*

[4]Note that this is *not* the notion of directed treewidth [7].

*2. The number of directed paths of length $k$ in $G$.*

**Remark on counting vertex covers:** Given an undirected graph $G$, the number of minimal vertex covers of size at most $k$ is at most $2^k$. This can be easily seen by a branching algorithm for finding minimal vertex covers, which branches on either end point of an edge (see [3], for example). Hence we can count the number of vertex covers of size at most $k$ by counting the number of subsets of $V(G)$ of size at most $k$ that contains one of the minimal vertex covers. Thus, this easily fits into the framework of approximate counting ($A_i$ of Theorem 2.1 would, for example, be the set of vertex covers containing the $i$-th minimal vertex cover in some ordering of minimal vertex covers.) and by applying Theorem 2.1, we can approximately count the number of vertex covers in $2^{O(k)}n^{O(1)}$ time. However, since $m$, the number of subsets, the size of whose union we are interested in counting, is a function of $k$, we can exactly count the number of vertex covers using the principle of inclusion-exclusion in deterministic $2^{O(k^2)}$ additional time after finding the minimal vertex covers [6]. Rossmanith [16] has recently obtained a $c^k n^{O(1)}$ deterministic algorithm for exactly counting vertex covers of size at most $k$, where $c < 2$ is some constant.

# 3 W-hard Exact Counting Problems

As we remarked in the introduction, the exact counting versions of paths of length $k$ or matchings of size $k$ have been proved to be $W[1]$-complete by Flum and Grohe [6]. In this section, we give more examples of problems whose decision versions are FPT, but whose counting versions are hard for some complexity class in the $W$-hierarchy.

## 3.1 Counting weight $k$ satisfying assignments of 2-DNF formulas

Consider checking for weight $k$ satisfying assignments of a monotone 2-DNF formula. Clearly, the decision problem is trivial (The answer is always YES). But counting the number of weight $k$ satisfying assignments is $W[1]$-hard by the following reduction from the $W[1]$-hard problem [3] which asks whether there is a weight $k$ satisfying assignment for a given antimonotone 2-CNF formula (In an antimonotone formula, all literals appear in negated form.). If $F$ is an antimonotone 2-CNF with $m$ clauses then its complement $\overline{F}$ is a monotone 2-DNF with $m$ terms. The $\binom{n}{k}$ assignments of weight $k$ are partitioned into those that satisfy $F$ and those that satisfy $\overline{F}$. If we can count the number that satisfies $\overline{F}$, then we can clearly decide if there is a weight $k$ assignment that satisfies $F$. Thus we have the following:

**Theorem 3.1** *It is $W[1]$-hard to find the number of weight $k$ satisfying assignments of a monotone 2-DNF formula.*

On the other hand, randomized approximate counting can be efficiently done by directly applying the Karp-Luby result [14].

**Corollary 3.1** *There is an FPRAS algorithm to approximate the number of weight $k$-satisfying assignments of DNF formulas.*

## 3.2 Counting weight at most $k$ satisfying assignments of 2-CNF formulas

Given a $c$-CNF formula $F$ (where each clause has at most $c$ literals), and an integer parameter $k$, it is FPT to decide whether or not the formula has a satisfying assignment of weight at most $k$

[12]. Simply find a clause in which all literals appear in positive form, and branch on its literals setting each one to true. If all clauses contain some literal in negative form at any stage, simply set all remaining variables to false. While branching, if some branch has more than $k$ variables set to true, simply abandon the branch. Answer YES if the branching algorithm leads to a satisfying assignment with weight at most $k$ and NO otherwise. It is clear that the algorithm takes $O(c^k|F|)$ time where $|F|$ is the size of the formula.

However counting the number of satisfying assignments of weight at most $k$ is $W[1]$-hard. This is because by finding the difference between the number of satisfying assignments of weight at most $k$ and weight at most $k-1$, we can determine whether or not the given formula has weight $k$ satisfying assignment. But finding this is known to be $W[1]$-hard even for antimonotone 2-CNF formulas [3]. Thus we have the following:

**Theorem 3.2** *It is $W[1]$-hard to find the number of satisfying assignments of weight at most $k$ in an antimonotone 2-CNF formula.*

### 3.3 Counting cliques and independent sets

While the CLIQUE and the INDEPENDENT SET problems are $W[1]$-complete, consider the question whether a given undirected graph $G$ has either a clique or an independent set of size $k$ for a given integer parameter $k$?

The decision version of this problem is known to be FPT [10] by the following simple argument.

- If the number of vertices in $G$ is at least $R(k,k)$, the Ramsey number, then answer YES.
- Otherwise, by brute force for each subset of size $k$ of the vertex set, check whether it forms a clique or an independent set. If any of them does, say YES and stop. Otherwise answer NO.

Since any graph on $R(k,k)$ vertices has a clique or an independent set of size $k$, the correctness of the algorithm is immediate. In the first case, we can try all $k$ elements subsets of a subset of size $R(k,k)$ (or a known upper bound of $R(k,k)$) to actually find a clique or an independent set of size $k$. It is also clear that the running time of the algorithm is $f(k)$ for some function of $k$ alone.

Now we will show that counting the number of solutions – i.e. of $k$ sized cliques and $k$ sized independent sets in a graph is $W[1]$- hard by a reduction from CLIQUE.

Let $G$ be an undirected graph in which we are interested in testing whether or not it has a clique of size $k$. Obtain a graph $G'$ by adding a new vertex and making it adjacent to all vertices of $G$. Let $CI_k(G)$ be the set of all cliques and independent sets of size $k$ in $G$. Now $G$ has a clique of size $k$ if and only if $|CI_{(k+1)}(G') - CI_{(k+1)}(G)| > 0$. Thus if finding $|CI_k(G)|$ is in FPT then so will testing whether a given graph has a $k$-clique. Thus we have the following:

**Theorem 3.3** *It is $W[1]$-hard to find the number of $k$ cliques and $k$ independent sets in an undirected graph $G$, where $k$ is an integer parameter.*

## 4 Conclusions and Open Problems

We have obtained a randomized approximate fixed parameter tractable algorithm for counting the number of copies of a bounded treewidth graph in a given graph. Our algorithm gives a nice combination and an interesting generalization of both the Karp-Luby approximate counting technique and the color-coding technique, based on perfect hashing, of Alon, Yuster and Zwick [1]. Two direct applications are fixed parameter tractable approximate counting algorithms for counting

paths of length $k$ in a graph and counting matchings of size $k$. Our results nicely complement the W[1]-hardness of exact counting for the above two problems shown by Flum and Grohe [6]. It would be interesting to explore further applications of the combination of Karp-Luby and color coding. Do we have *deterministic* approximate counting for Problem 1.1 with similar time bounds? The Karp-Luby approximate counting algorithm for #-DNF could be derandomized using the Nisan-Wigderson generator (see e.g. [15]). Unfortunately, it does not appear that the same technique will work for our algorithm in Theorem 2.2.

We also looked at exact counting versions of parameterized problems. We observed examples of fixed parameter tractable problems whose counting version are $W$-hard. It would be interesting to explore the approximate counting versions of the problems dealt with in Section 3.2 and 3.3. More specifically, is there a randomized fixed parameter tractable algorithm for approximately counting

- the number of weight at most $k$ satisfying assignments of a 2CNF formula, or
- the number of $k$ cliques and $k$ independent sets in a graph?

## References


[1] N. Alon, R. Yuster and U. Zwick, "Color-Coding", *Journal of the Association for Computing Machinery*, **42**(4) (1995) 844-856.

[2] H. Bodlaender, "A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth", *SIAM J. Computing* **25** (1996) 1305-1317.

[3] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, 1998.

[4] U. Feige and J. Kilian, "On Limited versus Polynomial Nondeterminism", *Chicago Journal of Theoretical Computer Science*, March (1997).

[5] H. Fernau, "Parameterized Enumeration", to appear *in the Proceedings of COCOON 2002.*

[6] J. Flum and M. Grohe, "The Parameterized Complexity of Counting Problems", manuscript.

[7] T. Johnson, N. Robertson, P. D. Seymour, R. Thomas, "Directed Tree-Width", preprint (1998) (available at http://www.math.gatech.edu/#thomas/).

[8] R. M. Karp and M. Luby, "Monte-Carlo Algorithms for Enumeration and Reliability Problems", *In Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science* (1983) 56-64.

[9] R. M. Karp, M. Luby and N. Madras, "Monte-Carlo Approximation Algorithms for Enumeration Problems", *Journal of Algorithms* **10** (1989) 429-448.

[10] S. Khot and V. Raman, "Parameterized Complexity of Finding Subgraphs with Hereditary Properties", *Proceedings of the Sixth Annual International Computing and Combinatorics Conference (COCOON*, July 2000, Sydney, Australia, Lecture Notes in Computer Science, Springer Verlag **1858** (2000) 137-147. Full version to appear in *Theoretical Computer Science.*

[11] T. Kloks, "Treewidth: Computations and Approximations", Lecture Notes in Computer Science, Springer-Verlag **842** 1994.

[12] M. Mahajan and V. Raman, "Parameterizing Above Guaranteed Values: MaxSat and MaxCut", *Journal of Algorithms* **31** (1999) 335-354.

[13] C. McCartin, "Parameterized Counting Problems", to appear in *the Proceedings of MFCS 2002 conference.*

[14] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[15] N. Nisan, *Using Hard Problems to Create Pseudorandom Generators,* MIT Press (1992).

[16] P. Rossmanith, private communication.




ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'