# Parameterized Algorithms for the Subgraph Isomorphism Problem

Zachary Frenette
University of Waterloo

April 7, 2014

## 1 Introduction

Given two undirected graphs $F$ and $G$, the subgraph isomorphism problem asks whether there exists a subgraph of $G$ that is isomorphic to $F$. Although NP-complete [7], this problem has been very well studied in a variety of settings, particularly in the context of parameterized complexity. In 1978, Matula showed that the subgraph isomorphism problem is solvable in polynomial time when both $F$ and $G$ are trees [13]. Moreover, it was later shown by Matoušek and Thomas that the problem remains NP-complete, even when $F$ and $G$ have treewidth at most two [12]. Despite this result, many other papers have tackled the subgraph isomorphism problem when parameterized by treewidth. For example, by introducing the notion of color coding, Alon, Yuster, and Zwick gave an algorithm requiring exponential time and space to solve the subgraph isomorphism problem when $F$ has bounded treewidth [1]. Amini, Fomin, and Saurabh improved upon this result and presented an algorithm requiring a polynomial amount of space by combining the ideas of color coding and counting graph homomorphisms [2]. Furthermore, many of these techniques and ideas have since been adapted to create parameterized algorithms for the counting version of the subgraph isomorphism problem. In particular, by taking ideas from algebraic combinatorics and combining them with the notion of graph homomorphisms and dynamic programming, new algorithms have been developed for both the decision and counting version of the problem when parameterized by treewidth or pathwidth [2, 8, 9]. Although none of these results are fixed parameter tractable with respect to treewidth, there has been a substantial amount of work done in finding sets of parameters for which the problem does become fixed parameter tractable. In particular, Marx and Pilipczuk presented a series of results in which they provide reductions between different sets of parameters. Using these reductions, they then characterize under which parameterizations the subgraph isomorphism problem becomes fixed parameter tractable [11].

In this paper, we will focus on algorithms that are parameterized by treewidth. Section 2 of this paper will present some preliminary definitions and results regarding tree decompositions and treewidth. In sections 3 and 4, we will examine in more detail some of the algorithms mentioned above. In particular, section 3 will examine the color coding method for finding a subgraph in G that is isomorphic to F, while section 4 will examine the relationship between counting subgraphs and counting homomorphisms. In section 5, we will explore how the color coding technique and graph homomorphisms can be used to solve the annotated version of the subgraph isomorphism problem. In other words, given a set $\mathcal{R}$ of vertices from $G$, we will be interested in finding and counting the subgraphs of $G$ that are isomorphic to $F$, with the additional requirement that they

contain each of the vertices from $\mathcal{R}$. Lastly, we will finish with some concluding remarks and then briefly discuss some potential directions for future work.

## 2 Preliminary Definitions and Results

Consider an undirected graph $G$ without any loops or multiple edges. We will use the notation $V(G)$ and $E(G)$ to denote the set of vertices and the set of edges of $G$ respectively. Furthermore, for any subset $W \subseteq V(G)$ of vertices, we define $G[W]$ to be the subgraph of $G$ induced by $W$. In 1986, Robertson and Seymour introduced the important concepts of treewidth and tree decompositions [14], which have served as the framework for many algorithmic results on graphs.

**Definition 1.** *A* **tree decomposition** *of an undirected graph $G$ is a pair $(\mathcal{X}, T)$, where $T$ is a tree and $\mathcal{X} = \{X_i : i \in V(T)\}$ is a set of subsets of $V(G)$ satisfying three conditions:*

- *The union $\bigcup_{i \in V(T)} X_i = V(G)$.*
- *If $\{u, v\} \in E(G)$, then there is an $i \in V(T)$ such that both $u, v \in X_i$.*
- *For each vertex $v \in V(G)$ , the set of nodes $\{i : v \in X_i\}$ forms a connected subgraph of $T$.*

**Definition 2.** *Consider a tree decomposition $(\mathcal{X}, T)$ of $G$. Then we define the* **width** *of $(\mathcal{X}, T)$ to be $\max_{i \in V(T)}\{|X_i| - 1\}$. Furthermore, we define the* **treewidth** *of $G$ to be the minimum width over all possible tree decompositions of $G$, which we will denote by $\mathbf{tw}(G)$.*

Moreover, it was shown by Arnborg, Corneil, and Proskurowski that a tree decomposition of $G$ having width $t$ can be computed in $\mathcal{O}(|V(G)|^{t+2})$ time [3]. This result can be characterized by the following Lemma.

**Lemma 1** ([3])**.** *Consider an undirected graph $G$ with $n_G = |V(G)|$ vertices. Then a tree decomposition of width $t \geq \mathbf{tw}(G)$ and $\mathcal{O}(n_G)$ nodes can be computed in $\mathcal{O}(n_G^{t+2})$ time.*

In 1991, Bodlaender and Kloks extended the notion of tree decompositions [5]. In particular, they defined the notion of a nice tree decomposition as follows.

**Definition 3.** *A* **nice tree decomposition** *of an undirected graph $G$ is a triple $(\mathcal{X}, T, r)$ such that $T$ is rooted at $r$ and $(\mathcal{X}, T)$ is a tree decomposition with the following additional properties:*

- *Every node $i$ of $T$ has at most two children.*
- *If node $i \in V(T)$ has two children $j$ and $k$, then $X_i = X_j = X_k$.*
- *If node $i \in V(T)$ has one child $j$, then either $|X_i| = |X_j| + 1$ with $X_j \subset X_i$ or $|X_i| = |X_j| - 1$ with $X_i \subset X_j$.*

We can observe that the nodes $i \in V(T)$ from a nice tree decomposition have one of four possible types. In particular, $i$ is called a **start node** if it is a leaf of $T$, a **join node** if it has two children, an **introduce node** if it has one child $j$ such that $|X_i| = |X_j| + 1$, or a **forget node** if it has one child $j$ such that $|X_i| = |X_j| - 1$. Furthermore, Bodlaender and Kloks showed that a nice tree decomposition can be obtained from a regular tree decomposition of $G$ in $\mathcal{O}(|V(G)|)$ time [6]. In particular, they gave the following Lemma.

**Lemma 2** ([6]). *Consider an undirected graph $G$ with $n_G = |V(G)|$ vertices. Then for any tree decomposition of $G$ on $\mathcal{O}(n_G)$ nodes with width $t \geq \mathbf{tw}(G)$, there exists an $\mathcal{O}(n_G)$ algorithm that constructs a nice tree decomposition of $G$ with width bounded by $t$ and with at most $\mathcal{O}(n_G)$ nodes.*

This Lemma tells us that, from an ordinary tree decomposition, we can compute a nice tree decomposition without much additional work. Moreover, we will see that nice tree decompositions admit a structure that can be exploited nicely to create a variety of dynamic programming algorithms.

# 3 Finding Isomorphic Subgraphs Using Color Coding

One of the first non-trivial techniques used to solve the subgraph isomorphism problem in the context of parameterized complexity was the color coding technique, proposed by Alon, Yuster, and Zwick in 1995 [1]. The main idea behind the algorithm is to randomly color the vertices of $G$ in order to see if $G$ contains a colorful copy of $F$. That is, $G$ contains a colorful copy of $F$ if each of the vertices corresponding to the isomorphism is colored using a distinct color. This is accomplished by using dynamic programming and carefully processing a nice tree decomposition of $F$. We will formalize this idea by proving the following Theorem.

**Theorem 1** ([1]). *Consider an undirected graph $F$ having treewidth $t = \mathbf{tw}(F)$ and $n_F = |V(F)|$ vertices, and let $G$ be an undirected graph with $n_G = |V(G)|$ vertices. Then a subgraph of $G$ isomorphic to $F$ can be found in $2^{\mathcal{O}(n_F)} n_G^{t+1}$ expected time or $2^{\mathcal{O}(n_F)} n_G^{t+1} \log n_G$ worst case time.*

*Proof.* Using Lemma 1 and Lemma 2, we start by computing a nice tree decomposition $(\mathcal{X}, T, r)$ of $F$ having width $t$ in $\mathcal{O}(n_F^{t+2})$ time. Then we choose a random coloring $\Gamma : V(G) \to \{1, \ldots, n_F\}$ of $G$. With probability at least $e^{-n_F}$ (by Stirling's approximation), the subgraph of $G$ isomorphic to $F$, if it exists, will become colorful. For any node $i \in V(T)$, we define $F_i$ to be the graph induced by the set of vertices containing $X_i$ and $X_j$, for all descendants $j$ of $i$ in $T$. In particular, we note that $F_r = F$. We will now apply dynamic programming on our nice tree decomposition as follows. For any $i \in V(T)$, $\Delta : X_i \to V(G)$, and $C \subseteq \{1, \ldots, n_F\}$, we define $\Upsilon(i, \Delta, C) = 1$ if there exists an isomorphism $\psi$ from $F_i$ to a subgraph $G'$ of $G$ such that the following conditions hold:

- The subgraph $G'$ is a colorful copy of $F_i$ with color set $C$.
- For all $v \in X_i$, $\psi(v) = \Delta(v)$. That is, $\psi$ is an extension of $\Delta$.

In any other case, we set $\Upsilon(i, \Delta, C) = 0$. As there can be more than one vertex in any particular bag $X_i$, we can think of the mapping $\Delta$ as an assignment of the vertices in $X_i$ to a corresponding subset of vertices in $G$. Therefore, if we can find a colorful isomorphism $\psi$ that is consistent with $\Delta$, we have found a colorful copy of $F_i$ in $G$. We will now show how to compute, for each node $i \in V(T)$, the corresponding entries in $\Upsilon$. In particular, we have four cases to consider. If $i$ is a start node in $T$, then $|X_i| = 1$ and so $F_i$ contains a single vertex $x$. Hence, we can pick the isomorphism $\psi \cong \Delta$ which satisfies both required properties, and so we obtain $\Upsilon(i, \Delta, \{\Gamma(x)\}) = 1$.

If $i$ is a join node in $T$, then let $j$ and $k$ be both of its children. Since $X_i = X_j = X_k$, $G$ contains a colorful copy of $F_i$ on color set $C$ if we can find color sets $C_j$ and $C_k$ such that $G$ contains a colorful copy of $F_j$ on color set $C_j$ and a colorful copy of $F_k$ on color set $C_k$. In addition, we require that $C_j$ and $C_k$ contain all of the colors in $C$, while making sure that they both contain the colors required to color the vertices in $X_i$. Hence, $\Upsilon(i, \Delta, C) = 1$ if and only if there exists color sets

$C_j$ and $C_k$ such that $C_j \cap C_k = \{\Gamma(\Delta(x)) : x \in X_i\}$, $C_j \cup C_k = C$, and $\Upsilon(j, \Delta, C_j) = \Upsilon(k, \Delta, C_k) = 1$.

If $i$ is an introduce node in $T$, then let $j$ be its child and let $\{x\} = X_i - X_j$. Intuitively, $G$ contains a colorful copy of $F_i$ on color set $C$ if we can find a color set $C_j$, differing from $C$ only by the removal of the color assigned to $x$, such that $G$ contains a colorful copy of $F_j$ on color set $C_j$. More formally, $\Upsilon(i, \Delta, C) = 1$ if and only if there exists a mapping $\Delta' : X_j \to V(G)$ that is edge consistent with $\Delta$ and a color set $C_j = C - \{\Gamma(\Delta(x))\}$ such that $\Upsilon(j, \Delta', C_j) = 1$. Similarly, if $i$ is a forget node in $T$, then let $j$ be its child and let $\{x\} = X_j - X_i$. Since no new vertices have been added, we can reuse the same color set $C$. Hence, $G$ contains a colorful copy of $F_i$ on color set $C$ if $G$ contains a colorful copy of $F_j$ on color set $C$ as well. So $\Upsilon(i, \Delta, C) = 1$ if and only if there exists a mapping $\Delta' : X_j \to V(G)$ that is edge consistent with $\Delta$ such that $\Upsilon(j, \Delta', C) = 1$.

Now that we have presented the algorithm and argued its correctness, it suffices to prove its runtime. We have at most $2^{n_F}$ choices for $C$, at most $2^{n_F}$ choices for $C_j$ and $C_k$, and at most ${n_G}^{t+1}$ choices for $\Delta$ and $\Delta'$, and so repeating the algorithm at most $e^{n_F}$ times gives an expected runtime of $2^{\mathcal{O}(n_F)}{n_G}^{t+1}$ as required. To obtain the worst case bound, we can derandomize the algorithm by using an $n_F$-perfect family of hash functions. We refer the reader to section 4 of the color coding paper by Alon et al. for more information [1]. □

Although this algorithm has proven successful for finding isomorphic subgraphs, it requires an exponential amount of space. However, we will see in the next section that the color coding technique can be adapted to require a polynomial amount of space instead.

# 4 Counting Subgraphs Using Graph Homomorphisms

In this section, we will survey and discuss some of the recent techniques used to solve the counting version of the subgraph isomorphism problem. In this version of the problem, we are given two undirected graphs $F$ and $G$, and we would like to count the number of distinct subgraphs of $G$ that are isomorphic to $F$. One of the most recent techniques used to solve this problem is the idea of graph homomorphisms. In particular, the main idea is to relate counting isomorphic subgraphs to counting graph homomorphisms. Informally speaking, a graph homomorphism is a mapping from the vertices of $F$ to the vertices of $G$ such that it maps adjacent vertices in $F$ to adjacent vertices in $G$. More formally, we have the following definition.

**Definition 4.** *Given two undirected graphs $F$ and $G$, a* **graph homomorphism** *from $F$ to $G$ is a mapping $\psi : V(F) \to V(G)$ such that if $\{u, v\} \in E(F)$ then $\{\psi(u), \psi(v)\} \in E(G)$. In addition, when $\psi$ is an injective mapping, we say that $\psi$ is an* **injective homomorphism**.

We will also be interested in the case where $\psi$ is an injective homomorphism from $F$ to $F$. In such a scenario, we will call $\psi$ an **automorphism** of $F$.

The problem of counting subgraphs is directly related to the problem of counting injective homomorphisms. Let $\mathcal{S}(F, G)$ denote the number of subgraphs of $G$ that are isomorphic to $F$, let $\mathcal{H}(F, G)$ denote the number of homomorphisms from $F$ to $G$, let $\mathcal{I}(F, G)$ denote the number of injective homomorphisms from $F$ to $G$, and let $\mathcal{A}(F)$ denote the number of automorphisms of $F$. The following result presented by Amini, Fomin, and Saurabh can be used to characterize the relationship between both problems [2].

**Lemma 3.** *Consider two undirected graphs $F$ and $G$. Then $\mathcal{S}(F,G) = \mathcal{I}(F,G)/\mathcal{A}(F)$.*

The intuition behind this Lemma is that every injective homomorphism from $F$ to $G$ corresponds to some subgraph of $G$ that is isomorphic to $F$. However, multiple injective homomorphisms may correspond to the same subgraph of G (differing only by the labelling of the vertices). Hence, in order to avoid counting the same subgraph more than once, we must divide $\mathcal{I}(F,G)$ by the number of automorphisms of $F$.

It was shown by Babai, Kantor, and Luks that, given an undirected graph $F$ with $n_F = |V(F)|$ vertices, $\mathcal{A}(F)$ can be computed in time proportional to $2^{\mathcal{O}(\sqrt{n_F \log n_F})}$, which is subexponential in $n_F$ [4]. As a result, the crux of the work can be put into the computation of $\mathcal{I}(F,G)$. In particular, Amini et al. gave two results that are based on the inclusion-exclusion principle to compute $\mathcal{I}(F,G)$. More specifically, the inclusion-exclusion principle is used in order to create a relationship between $\mathcal{I}(F,G)$ and $\mathcal{H}(F,G)$. The first result can be seen as a special case of the second result, in which $|V(F)| = |V(G)|$.

**Theorem 2** ([2])**.** *Consider two undirected graphs $F$ and $G$ such that $n = |V(F)| = |V(G)|$. Then we obtain the following result:*

$$\begin{aligned}\mathcal{I}(F,G) &= \sum_{W \subseteq V(G)} (-1)^{|W|} \mathcal{H}(F, G[V(G) \backslash W]) \\ &= \sum_{W \subseteq V(G)} (-1)^{n-|W|} \mathcal{H}(F, G[W])\end{aligned}$$

*Proof.* The main idea behind this proof is to show that the contribution of a homomorphism $\psi$ is exactly one if and only if $\psi$ is injective. First, observe that $\psi$ can only be injective when $W$ is equal to the empty set, since we know that $|V(F)| = |V(G)|$. Therefore, since $\psi$ is only counted on the right hand side when $W = \emptyset$, we can conclude that $\psi$ is only counted once. In the case where $\psi$ is not injective, we will show that its total contribution to the right hand side is equal to zero. In particular, since $|V(F)| = |V(G)|$, there exists a non-empty set of vertices $V' \subseteq V(G)$ that is not part of the image of $\psi$. Observe that $\psi$ is only counted when we are counting homomorphisms from $F$ to $G[V(G)\backslash W]$ where $W \subseteq V'$. Hence the total contribution of $\psi$ on the right hand side is equal to $\sum_{i=0}^{|V'|} \binom{|V'|}{i} (-1)^i = (1-1)^{|V'|} = 0$, as required. Note that the second equality is simply obtained from the first by a change of variable from $W$ to $W' = V(G)\backslash W$. □

**Theorem 3** ([2])**.** *Consider two undirected graphs $F$ and $G$ such that $n_F = |V(F)| \leq n_G = |V(G)|$. Then we obtain the following result:*

$$\mathcal{I}(F,G) = \sum_{\substack{W \subseteq V(G) \\ |W| \leq n_F}} (-1)^{n_F - |W|} \binom{n_G - |W|}{n_F - |W|} \mathcal{H}(F, G[W])$$

The proof of this Theorem is omitted as it is similar to the proof of Theorem 2. We refer the reader to section 3 of the paper by Amini et al. for additional details [2]. From an algorithmic perspective, observe that if we can compute $\mathcal{H}(F, G[W])$ in $\mathcal{O}(f(n_G))$ time, then using Theorem 3, we can compute $\mathcal{I}(F,G)$ in $\mathcal{O}(f(n_G) \sum_{i=0}^{n_F} \binom{n_G}{i})$ time. In particular, we can use the following result for calculating the number of homomorphisms from one graph to another.

**Theorem 4** ([8]). *Consider two undirected graphs $F$ and $G$ on $n_F$ and $n_G$ vertices respectively. Given a tree decomposition of $F$ having width $t$, $\mathcal{H}(F, G)$ is computable in $\mathcal{O}(n_F \cdot n_G^{t+1} \cdot \min(t, n_G))$ time and in $\mathcal{O}(\log n_F \cdot n_G^{t+1})$ space.*

The techniques employed by this algorithm are quite similar to the ones behind the color coding algorithm presented earlier. In fact, this algorithm works by processing a nice tree decomposition of $F$ and applying dynamic programming on our table $\Upsilon$. However, instead of storing color sets in our table, we store partial homomorphisms whose domain is consistent with the vertices in the bags that have been processed so far. Moreover, since we are interested in the counting version of the problem, we will also store the number of such homomorphisms for every node in our nice tree decomposition. Combining this algorithm with the results of Theorem 3 gives us a procedure for computing the number of subgraphs in $G$ that are isomorphic to $F$ when $F$ has bounded treewidth. In particular, the procedure can be characterized by the following Theorem.

**Theorem 5** ([2]). *Given two undirected graphs $F$ and $G$ on $n_F$ and $n_G$ vertices respectively, together with a tree decomposition of $F$ having width $t$, we can compute $\mathcal{S}(F, G)$ in $\mathcal{O}(\sum_{i=0}^{n_F} \binom{n_G}{i} \cdot n_F^{t+2} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space.*

*Proof.* To prove this Theorem, we will need to show how to compute both $\mathcal{I}(F, G)$ and $\mathcal{A}(F)$ within the required amount of time and space. By Theorem 3, we can utilize the following expression to compute the number of injective homomorphisms from $F$ to $G$:

$$\mathcal{I}(F, G) = \sum_{\substack{W \subseteq V(G) \\ |W| \leq n_F}} (-1)^{n_F - |W|} \binom{n_G - |W|}{n_F - |W|} \mathcal{H}(F, G[W]).$$

Moreover, we can use Theorem 4 to compute the number of homomorphisms from $F$ to $G[W]$. Since $|W| \leq n_F$, computing $\mathcal{H}(F, G[W])$ can be done in $\mathcal{O}(n_F^{t+2} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space. Therefore, applying Theorem 4 to each subset $W$ gives us a final running time that is proportional to $\mathcal{O}(\sum_{i=0}^{n_F} \binom{n_G}{i} \cdot n_F^{t+2} \cdot t)$ and that uses $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space. To compute $\mathcal{A}(F)$, we first make the observation that $\mathcal{A}(F) = \mathcal{I}(F, F)$. As a result, we can apply Theorems 2 and 4 to compute $\mathcal{A}(F)$ in $\mathcal{O}(2^{n_F} \cdot n_F^{t+2} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space, which completes the proof. □

It is worth noting that the notion of color coding can be adapted to require only a polynomial amount of space by utilizing the inclusion-exclusion principle described above. In particular, the main idea will be to count colorful copies of $F$ in $G$ by making use of an analogous version of Theorem 3. However, we first need to define several new concepts.

**Definition 5.** *Given two undirected graphs $F$ and $G$ with a random coloring $\Gamma : V(G) \to \{1, \ldots, n_F\}$ of $G$, we say that an injective homomorphism $\psi$ is* **colorful** *if every vertex in the image of $\psi$ is colored with a distinct color. Moreover, we will use $\mathcal{I}_\Gamma(F, G)$ to denote the number of colorful injective homomorphisms from $F$ to $G$.*

**Definition 6.** *Given two undirected graphs $F$ and $G$ with a random coloring $\Gamma : V(G) \to \{1, \ldots, n_F\}$ of $G$, we define $\mathbf{G}^*$ to be the graph obtained from $G$ by deleting the monochromatic edges.*

Observe that the number of colorful copies of $F$ in $G$ is equal to $\mathcal{I}_\Gamma(F, G)/\mathcal{A}(F)$. Therefore, in order to solve the original problem, it suffices to adapt the inclusion-exclusion principle to count colorful injective homomorphisms. We can characterize this idea with the following Theorem.

**Theorem 6** ([2]). *Let $F$ and $G$ be two undirected graphs, let $\Gamma : V(G) \to \{1, \ldots, n_F\}$ be a random coloring of $G$, and let $\Gamma^{-1}(i)$ denote the set of vertices that have color $i$ in $G$. In addition, for any set $W \subseteq \{1, \ldots, n_F\}$ of colors, let $\Gamma_W^{-1} = \bigcup_{i \in W} \Gamma^{-1}(i)$. Then it can be shown that:*

$$\mathcal{I}_\Gamma(F, G) = \mathcal{I}_\Gamma(F, G^*) = \sum_{W \subseteq \{1, \ldots, n_F\}} (-1)^{|W|} \mathcal{H}(F, G^*[V(G^*) \backslash \Gamma_W^{-1}])$$

*Proof.* To prove the first equality, we will begin by aruging that every colorful injective homomorphism from $F$ to $G$ is also a colorful injective homomorphism from $F$ to $G^*$. If $\psi$ is a homomorphism from $F$ to $G$ that is both colorful and injective, then for every edge $\{u, v\} \in E(F)$, the corresponding edge $\{\psi(u), \psi(v)\} \in E(G)$ has its endpoints colored by two different colors. Hence $\psi$ will remain colorful and injective when we remove the monochromatic edges from $G$. A similar type of argument can be made to prove the other direction of the claim. In particular, if $\psi$ is a homomorphism from $F$ to $G^*$ that is both colorful and injective, then adding edges between pairs of vertices that have the same color will not be problematic since at most one of these vertices can be in the image of $\psi$. Therefore, it follows that $\mathcal{I}_\Gamma(F, G) = \mathcal{I}_\Gamma(F, G^*)$.

To prove the second equality, we will mimic the proof of Theorem 2. In particular, we will first show that a colorful injective homomorphism $\psi$ from $F$ to $G$ is counted exactly once on the right hand side. First, observe that $\psi$ can only be colorful and injective when $W$ is equal to the empty set. Since the number of vertices in $F$ is $n_F$, then removing the vertices in $\Gamma_W^{-1}$ for any non-empty set $W$ will leave $G^*$ with strictly less than $n_F$ different colors. Therefore, $\psi$ can only be counted when $W$ is the empty set and so its contribution to the right hand side is exactly 1. In the case where $\psi$ is not a colorful injective homomorphism, we will show that its total contribution to the right hand side is equal to zero. First, we define $\mathcal{C} = \{\Gamma(v) : v \text{ is in the image of } \psi\}$. Since $\psi$ is not a colorful injective homomorphism, it misses some of the color classes, and so $\mathcal{C}' = \{1, \ldots, n_F\} \backslash \mathcal{C}$ is non-empty. Observe that $\psi$ is only counted when we are counting homomorphisms from $F$ to $G^*[V(G^*) \backslash \Gamma_W^{-1}]$ such that $W \subseteq \mathcal{C}'$. Hence the total contribution of $\psi$ on the right hand side is equal to $\sum_{i=0}^{|\mathcal{C}'|} \binom{|\mathcal{C}'|}{i} (-1)^i = (1-1)^{|\mathcal{C}'|} = 0$, as required. □

Several important observations can be made from this Theorem. First, observe that we can view $G^*$ as a simple form of kernelization from $G$, as we are reducing the input size without effecting the number of solutions to our problem. Second, we can combine the results of Theorems 4 and 6 to efficiently count the number of colorful injective homomorphisms from $F$ to $G$. In particular, there are $2^{n_F}$ different subsets for $W$, and by Theorem 4, it takes $\mathcal{O}(n_F \cdot n_G^{t+1} \cdot t)$ time and $\mathcal{O}(\log n_F \cdot n_G^{t+1})$ space to compute $\mathcal{H}(F, G^*[V(G^*) \backslash \Gamma_W^{-1}]$. Hence we can compute $\mathcal{I}_\Gamma(F, G)$ in $\mathcal{O}(2^{n_F} \cdot n_F \cdot n_G^{t+1} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_G^{t+1})$ space. In addition, we can use this result to determine if $G$ has a subgraph that is isomorphic to $F$.

**Theorem 7** ([2]). *Consider two undirected graphs $F$ and $G$ on $n_F$ and $n_G$ vertices respectively such that $F$ has treewidth $t = \mathbf{tw}(F)$. Then a subgraph of $G$ that is isomorphic to $F$ can be found in $\mathcal{O}((2e)^{n_F} \cdot n_F \cdot n_G^{t+1} \cdot t)$ expected time or in $\mathcal{O}((2e)^{n_F + o(n_F)} \cdot n_F \cdot n_G^{t+1} \cdot t)$ worst case time. In either case, we will require $\mathcal{O}(\log n_F \cdot n_G^{t+1})$ space.*

*Proof.* We begin by choosing a random coloring $\Gamma : V(G) \to \{1, \ldots, n_F\}$ of $G$. With probability at least $e^{-n_F}$ (by Stirling's approximation), the subgraph of $G$ isomorphic to $F$, if it exists, will become colorful. Given this random coloring, the previous observation implies that $\mathcal{I}_\Gamma(F, G)$ can

be computed in $\mathcal{O}(2^{n_F} \cdot n_F \cdot n_G^{t+1} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_G^{t+1})$ space. If $\mathcal{I}_\Gamma(F, G) > 0$, then we know that there exists a subgraph of $G$ that is isomorphic to $F$. Hence repeating this procedure $e^{n_F}$ times gives the desired result. To obtain the worst case bound, we can derandomize the algorithm by using an $(n_G, n_F, n_F)$-perfect family of hash functions. We refer the reader to section 6 of the Amini et al. paper for additional information [2]. □

As a final comment, it is worth noting that these results can be slightly improved by coloring $G$ using additional colors. In particular, Hüffner, Wernicke, and Zichner showed that by using $1.3n_F$ different colors in the coloring of $G$, the copy of $F$ in $G$ becomes colorful with probability at least $\mathcal{O}(1.752^{-n_F})$. Additional information regarding this idea can be found in their paper [10].

# 5 Annotating The Subgraph Isomorphism Problem

In this section, we will explore how the color coding technique and graph homomorphisms can be used to solve the annotated version of the subgraph isomorphism problem. More specifically, we will be given two undirected graphs $F$ and $G$, together with a set $\mathcal{R} \subseteq V(G)$ of vertices, and we would like to determine if there exists a subgraph of $G$ containing $\mathcal{R}$ that is isomorphic to $F$. Similarly, for the counting version of the problem, we will be interested in counting the number of subgraphs containing $\mathcal{R}$ that are isomorphic to $F$. To begin, we define $\mathcal{S}(F, G, \mathcal{R})$ to be the number of subgraphs of $G$ containing $\mathcal{R}$ that are isomorphic to $F$. Furthermore, we define $\mathcal{I}(F, G, \mathcal{R})$ to be the number of injective homomorphisms from $F$ to $G$ that have each vertex in $\mathcal{R}$ contained in their image. Clearly, it follows that $\mathcal{S}(F, G, \mathcal{R}) = \mathcal{I}(F, G, \mathcal{R}) \backslash \mathcal{A}(F)$, and so it suffices to show how to compute $\mathcal{I}(F, G, \mathcal{R})$. We will begin by proving a Theorem that is analogous to Theorem 3.

**Theorem 8.** *Consider two undirected graphs $F$ and $G$ such that $n_F = |V(F)| \leq n_G = |V(G)|$, and let $\mathcal{R} \subseteq V(G)$ be a set of vertices. Then it can be shown that:*

$$\mathcal{I}(F, G, \mathcal{R}) = \sum_{\substack{W \subseteq V(G) \\ |W| \leq n_F}} (-1)^{n_F - |W|} \binom{n_G - |W \cup \mathcal{R}|}{n_F - |W \cup \mathcal{R}|} \mathcal{H}(F, G[W])$$

*Proof.* Observe that we can use Theorem 2 to obtain the following sum:

$$\begin{aligned} \mathcal{I}(F, G, \mathcal{R}) &= \sum_{\substack{\mathcal{R} \subseteq Y \subseteq V(G) \\ |Y| = n_F}} \mathcal{I}(F, G[Y]) \\ &= \sum_{\substack{\mathcal{R} \subseteq Y \subseteq V(G) \\ |Y| = n_F}} \left( \sum_{W \subseteq Y} (-1)^{n_F - |W|} \mathcal{H}(F, G[W]) \right) \qquad \text{(by Theorem 2)} \end{aligned}$$

To simplify things further, we need to determine, for each subset $W$, how many sets $\mathcal{R} \subseteq Y \subseteq V(G)$ of size $n_F$ contain $W$. In particular, this will depend on the size of $\mathcal{R} \cap W$. More specifically, we will need to add $|\mathcal{R} \backslash W|$ elements to $W$ in order for it to contain the elements of $\mathcal{R}$. Moreover, we can then add any other $n_F - |W| - |\mathcal{R} \backslash W|$ elements out of the remaining $n_G - |W| - |\mathcal{R} \backslash W|$ elements to get a set of size $n_F$. Since $|W| + |\mathcal{R} \backslash W| = |\mathcal{R} \cup W|$, this gives $\binom{n_G - |\mathcal{R} \cup W|}{n_F - |\mathcal{R} \cup W|}$ such sets $Y$.

Therefore, putting these observations together gives the desired result:

$$\mathcal{I}(F, G, \mathcal{R}) = \sum_{\substack{W \subseteq V(G) \\ |W| \leq n_F}} (-1)^{n_F - |W|} \binom{n_G - |W \cup \mathcal{R}|}{n_F - |W \cup \mathcal{R}|} \mathcal{H}(F, G[W])$$

□

Using this Theorem together with Theorem 4 gives an immediate algorithm for computing $\mathcal{S}(F, G, \mathcal{R})$. In particular, since $|W| \leq n_F$, we can use Theorem 4 to compute $\mathcal{H}(F, G[W])$ in $\mathcal{O}(n_F^{t+2} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space, where $t$ denotes the treewidth of $F$. Moreover, we are computing $\mathcal{H}(F, G[W])$ over $\sum_{i=0}^{n_F} \binom{n_G}{i}$ different sets $W$. Therefore, $\mathcal{I}(F, G, \mathcal{R})$ can be computed in $\mathcal{O}(\sum_{i=0}^{n_F} \binom{n_G}{i} \cdot n_F^{t+2} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space. In addition, since $\mathcal{A}(F) = \mathcal{I}(F, F)$, it follows that $\mathcal{A}(F)$ can be computed using the same amount of time and space.

**Theorem 9.** *Given two undirected graphs $F$ and $G$ on $n_F$ and $n_G$ vertices respectively, together with a set $\mathcal{R} \subseteq V(G)$ of vertices, we can compute $\mathcal{S}(F, G, \mathcal{R})$ in $\mathcal{O}(\sum_{i=0}^{n_F} \binom{n_G}{i} \cdot n_F^{t+2} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space, where $t$ denotes the treewidth of $F$.*

Now we would like to explore how the color coding technique can be adapted to solve the annotated version of the subgraph isomorphism problem. In particular, we would like to prove a Theorem that is analogous to Theorem 6. We define, for any coloring $\Gamma : V(G) \to \{1, \ldots, n_F\}$ of $G$, $\mathcal{I}_\Gamma(F, G, \mathcal{R})$ to denote the number of colorful injective homomorphisms from $F$ to $G$ that have each vertex in $\mathcal{R}$ contained in their image. From this, we obtain the following Lemma.

**Lemma 4.** *Consider two undirected graphs $F$ and $G$, and let $\Gamma : V(G) \to \{1, \ldots, n_F\}$ be a coloring of $G$. Then for any set $\mathcal{R} \subseteq V(G)$, it follows that $\mathcal{I}_\Gamma(F, G, \mathcal{R}) = \mathcal{I}_\Gamma(F, G^*, \mathcal{R})$.*

*Proof.* The proof of this Lemma is similar to the proof of the first part of Theorem 6. We will begin by showing that every colorful injective homomorphism from $F$ to $G$ containing $\mathcal{R}$ is also a colorful injective homomorphism from $F$ to $G^*$ containing $\mathcal{R}$. If $\psi$ is such a homomorphism, then for every edge $\{u, v\} \in E(F)$, the corresponding edge $\{\psi(u), \psi(v)\} \in E(G)$ has its endpoints colored by two different colors. Hence $\psi$ will remain colorful and injective when we remove the monochromatic edges from $G$. Furthermore, since we have only removed edges, it will still contain the vertices in $\mathcal{R}$. A similar type of argument can be made to prove the other direction of the claim. In particular, if $\psi$ is a colorful injective homomorphism from $F$ to $G^*$ containing $\mathcal{R}$, then adding edges between pairs of vertices that have the same color will not be problematic since at most one of these vertices can be in the image of $\psi$. Also, since we are only adding edges, $\psi$ will still contain the vertices in $\mathcal{R}$. Therefore, it follows that $\mathcal{I}_\Gamma(F, G, \mathcal{R}) = \mathcal{I}_\Gamma(F, G^*, \mathcal{R})$. □

Using this Lemma, the goal was to prove an analogous version of Theorem 6. In particular, the idea was to emulate the inclusion-exclusion principle in order to find a relationship between $\mathcal{I}_\Gamma(F, G^*, \mathcal{R})$ and $\mathcal{H}(F, G^*[X])$, for some appropriate set $X \subseteq V(G)$ of vertices. If one could find such a relationship, then it may be possible to adapt the color coding method to solve the annotated version of the subgraph isomorphism problem using only a polynomial amount of space. However, this proved to be quite difficult, with no clear way to incorporate the set $\mathcal{R}$ into the expression. In particular, since we are summing over sets of colors instead of sets of vertices, we could not use the same techniques that were used to prove Theorem 8, that is, making sure that each term of the summation contains $\mathcal{R}$ as a subset. As a result, we leave this as an avenue for future work.

## 6 Conclusions

In this paper, we have surveyed some of the techniques used to solve the subgraph isomorphism problem when parameterized by treewidth. More specifically, we have considered algorithms for both the decision and counting version of the problem. In regards to the counting version of the subgraph isomorphism problem, we discussed the relationship between counting isomorphic subgraphs and counting graph homomorphisms. In particular, we argued how the inclusion-exclusion principle could be adapted to compute $\mathcal{I}(F, G)$ and $\mathcal{A}(F)$ in an efficient manner. In regards to the decision version of the problem, we first investigated the details behind the original color coding algorithm, and then proceeded to see how it could be improved to use a polynomial amount of space. Lastly, we explored how these techniques can be used to solve the annotated subgraph isomorphism problem. We first gave an algorithm to compute $\mathcal{S}(F, G, \mathcal{R})$ in $\mathcal{O}(\sum_{i=0}^{n_F} \binom{n_G}{i} \cdot n_F^{t+2} \cdot t)$ time and in $\mathcal{O}(\log n_F \cdot n_F^{t+1})$ space, where $t$ denotes the treewidth of $F$. Then we discussed why, in the annotated case, the same technique used for counting subgraphs does not seem to work for counting colorful copies of $F$ in $G$.

In terms of future work, there are many avenues that one could explore. For example, it is still unclear how the color coding method can be adapted to solve the annotated version of the subgraph isomorphism problem, using only a polynomial amount of space. Given the difficulties discussed above, a natural approach might be to consider using dynamic programming directly. However, these dynamic programming algorithms will typically require an exponential amount of space. Hence perhaps a completely different technique will prove more successful than the original color coding algorithm. For example, it may be worth investigating if a technique like kernelization can be applied before color coding to simplify the amount of space needed. On a different note, it might be worth investigating algorithms over different parameterizations. Parameters like treewidth have been studied extensively, but perhaps there are other sets of parameters that will yield efficient algorithms, where this issue of exponential space does not arise.

## References


[1] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995.

[2] Omid Amini, Fedor V. Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. *SIAM J. Discrete Math.*, 26(2):695–717, 2012.

[3] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, April 1987.

[4] László Babai, William M. Kantor, and Eugene M. Luks. Computational complexity and the classification of finite simple groups. In *FOCS*, pages 162–171. IEEE Computer Society, 1983.

[5] Hans L. Bodlaender and Ton Kloks. Better algorithms for the pathwidth and treewidth of graphs. In *ICALP*, volume 510 of *Lecture Notes in Computer Science*, pages 544–555. Springer, 1991.

[6] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[7] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[8] Josep Díaz, Maria Serna, and Dimitrios M. Thilikos. Counting H-colorings of partial k-trees. *Theoretical Computer Science*, 281(12):291 – 309, 2002. Selected Papers in honour of Maurice Nivat.

[9] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B.V. Raghavendra Rao. Faster algorithms for finding and counting subgraphs. *J. Comput. Syst. Sci.*, 78(3):698–706, May 2012.

[10] Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008.

[11] Dániel Marx and Michal Pilipczuk. Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask). In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 542–553, Dagstuhl, Germany, 2014.

[12] Jirí Matoušek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.

[13] David W. Matula. Subtree Isomorphism in $O(n^{5/2})$. In *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 91 – 106. Elsevier, 1978.

[14] Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.