Hamiltonian Cycle Parameterized by Treedepth in Single Exponential Time and Polynomial Space

Jesper Nederlof[1], Michał Pilipczuk[2], Céline M. F. Swennenhuis[3], and Karol Węgrzycki[2(✉)]

[1] Utrecht University, Utrecht, The Netherlands
j.nederlof@uu.nl
[2] Institute of Informatics, University of Warsaw, Warsaw, Poland
{michal.pilipczuk,k.wegrzycki}@mimuw.edu.pl
[3] Eindhoven University of Technology, Eindhoven, The Netherlands
c.m.f.swennenhuis@tue.nl

Abstract. For many algorithmic problems on graphs of treewidth t, a standard dynamic programming approach gives an algorithm with time and space complexity $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$. It turns out that when one considers the more restrictive parameter treedepth, it is often the case that a variation of this technique can be used to reduce the space complexity to polynomial, while retaining time complexity of the form $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$, where d is the treedepth. This transfer of methodology is, however, far from automatic. For instance, for problems with connectivity constraints, standard dynamic programming techniques give algorithms with time and space complexity $2^{\mathcal{O}(t \log t)} \cdot n^{\mathcal{O}(1)}$ on graphs of treewidth t, but it is not clear how to convert them into time-efficient polynomial space algorithms for graphs of low treedepth.

Cygan et al. (FOCS'11) introduced the Cut&Count technique and showed that a certain class of problems with connectivity constraints can be solved in time and space complexity $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$. Recently, Hegerfeld and Kratsch (STACS'20) showed that, for some of those problems, the Cut&Count technique can be also applied in the setting of treedepth, and it gives algorithms with running time $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ and polynomial space usage. However, a number of important problems eluded such a treatment, with the most prominent examples being Hamiltonian Cycle and Longest Path.

In this paper we clarify the situation by showing that Hamiltonian Cycle, Hamiltonian Path, Long Cycle, Long Path, and Min Cycle Cover all admit $5^d \cdot n^{\mathcal{O}(1)}$-time and polynomial space algorithms on graphs of treedepth d. The algorithms are randomized Monte Carlo with only false negatives.

Keywords: Hamiltonian cycle · Connectivity · Polynomial space · Treedepth

I. Adler and H. Müller (Eds.): WG 2020, LNCS 12301, pp. 27–39, 2020.
https://doi.org/10.1007/978-3-030-60440-0_3

1 Introduction

It is widely believed that no NP-hard problem admits a polynomial time algorithm. However, actual instances of problems that we are interested in solving often admit much more structure than a general instance. This observation gave rise to the field of *parameterized complexity*, where the hardness of an instance does not depend exclusively on the input size. In the parameterized regime, we assume that each instance is equipped with an additional parameter k and the goal is to give a *fixed-parameter algorithm*: an algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a function independent of n. After settling that a problem admits such an algorithm, it is natural to lookfor one with function f as low as possible. We refer to [5,10,12] for an introduction to parameterized complexity.

One of the most widely used parameters is the *treewidth* t of the input graph. Usually, problems that involve only constraints of local nature admit an algorithm with running time of the form $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ [5]. For a long time, such algorithms remained out of reach for problems involving connectivity constraints, and for those only $2^{\mathcal{O}(t \log t)} \cdot n^{\mathcal{O}(1)}$-time algorithms were known. The breakthrough came with the Cut&Count technique, introduced by Cygan et al. in [7], that allows one to design randomized Monte-Carlo algorithms with running times of the form $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ for a wide range of connectivity problems, e.g., Hamiltonian Path, Connected Vertex Cover, Connected Dominating Set, etc. The technique was subsequently derandomized [4,13].

One of the main issues with standard dynamic programming algorithms is that they tend to have prohibitively large space usage. The natural goal is therefore to reduce the space complexity while not sacrificing much on the time complexity. Unfortunately, Drucker et al. [11] and Pilipczuk and Wrochna [23] gave some complexity-theoretical evidence that for dynamic programming on graphs of bounded treewidth, such a reduction is probably impossible. For example, they showed that under plausible assumptions, there is no algorithm that works in time $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ and uses $2^{o(t)} \cdot n^{\mathcal{O}(1)}$ space for the 3-Coloring or Independent Set problem.

Treedepth. The aforementioned issues motivate the research on a different, more restrictive parameterization, for which the reduction of space complexity would be possible. In this paper we will consider the parameterization by *treedepth*, defined as follows.

Definition 1. An *elimination forest* of a graph G is a rooted forest F on the same vertex set as G such that for every edge uv of G, either u is an ancestor of v in F or v is an ancestor of u in F. The *treedepth* of G is the minimum possible depth of an elimination forest of G.

The treedepth of a graph is never smaller than its treewidth, but it is also never larger than the treewidth times $\log n$. In many concrete cases, the two parameters have the same advantages. For example, planar graphs have treewidth $\mathcal{O}(\sqrt{n})$, but also *treedepth* $\mathcal{O}(\sqrt{n})$.

It has been recently realized that on graphs of treedepth d, many algorithmic problems indeed can be solved in time $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ and using only polynomial space.[1] For the most basic problems, such as 3-Coloring and Independent Set, a simple branching algorithms achieves such complexity. However, in contrast to the treewidth parameterization, for many more complex problems it is highly non-trivial, yet possible to establish similar bounds. One technique that turns out to be useful here is the framework of algebraic transforms introduced by Loksthanov and Nederlof [18], who demonstrated how to reduce the space requirements of many dynamic programming algorithms to polynomial in the input size by reorganizing the computation using a suitable transform. Fürer and Yu [14] applied this framework to give $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$-time and polynomial space algorithms on graphs of treedepth d for the Dominating Set problem and for the problem of counting the number of perfect matchings. Pilipczuk and Wrochna [23] considered algorithms with even more restricted space requirements: they showed that 3-Coloring, Dominating Set, and Vertex Cover admit algorithms that work in $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ time and use $\mathcal{O}(d + \log n)$ space. For Dominating Set they avoided the explicit use of algebraization and instead provided a more combinatorial interpretation based on what one could call *inclusion-exclusion branching*. Later, Pilipczuk and Siebertz [22] used color-coding to give an $2^{\mathcal{O}(d \log d)} \cdot n^{\mathcal{O}(1)}$-time and polynomial space algorithm for the Subgraph Isomorphism problem. Recently, Belbasi and Fürer [1] presented an algorithm for counting Hamiltonian cycles in time $(4t)^d \cdot n^{\mathcal{O}(1)}$ and using polynomial space, where t is the width of a given tree decomposition and d is its (suitably defined) depth.

Treedepth and Cut&Count. Very recently, Hegerfeld and Kratsch [16] demonstrated that the Cut&Count technique can be also applied in the setting of the treedepth parameterization. Consequently, they gave randomized algorithms with running times $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ and polynomial space usage for a number of problems with connectivity constraints such as Connected Vertex Cover, Connected Dominating Set, Feedback Vertex Set, or Steiner Tree. However, Hegerfeld and Kratsch found it problematic to apply the methodology to several important problems originally considered by Cygan et al. [7] in the context of Cut&Count. Specifically, these are problems based on selection of edges rather than vertices, such as Hamiltonian Cycle or Long Cycle. For this reason, Hegerfeld and Kratsch explicitly asked in [16] whether Hamiltonian Cycle, Hamiltonian Path, Long Cycle, and Min Cycle Cover

[1] Throughout the introduction, when we speak about a graph of treedepth d, we mean a graph supplied with an elimination forest of depth d. While in the case of treewidth, a tree decomposition of approximately (up to a constant factor) optimum width can be computed in time $8^t \cdot n^{\mathcal{O}(1)}$ [5,26], the existence of such an approximation algorithm for treedepth is a notorious open problem.

also admit $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$-time and polynomial space algorithms[2] on graphs of treedepth d (see the full version of this paper [21] for problem definitions[3]).

Our Contribution. In this paper we introduce additional techniques that allow us to extend the results of [16] and to answer the abovementioned open problem of Hegerfeld and Kratsch in the affirmative. More precisely, we prove the following theorem.

Theorem 1. *There is a randomized algorithm that given a graph G together with its elimination forest of depth d, and number $k \in \mathbb{N}$, solves* Hamiltonian Cycle, Hamiltonian Path, k-Cycle, k-Path *and* Min Cycle Cover *in time $5^d \cdot n^{\mathcal{O}(1)}$ and using polynomial space. The algorithm has a one-sided error: it may give false negatives with probability at most $\frac{1}{2}$.*

In fact, Theorem 1 is an easy corollary of the following result for a generalization of the considered problems. In the Partial Cycle Cover problem we are given an undirected graph G and integers k and ℓ, and we ask whether in G there is a family of at most k vertex-disjoint cycles that jointly visit exactly ℓ vertices. We will prove the following theorem.

Theorem 2. *There is a randomized algorithm that given a graph G together with its elimination forest of depth d, and numbers $k, \ell \in \mathbb{N}$, solves the* Partial Cycle Cover *problem for G, k, ℓ in time $5^d \cdot n^{\mathcal{O}(1)}$ and using polynomial space. The algorithm has a one-sided error: it may give false negatives with probability at most $\frac{1}{2}$.*

To see that Theorem 2 implies Theorem 1, note that Hamiltonian Cycle, Min Cycle Cover and Long Cycle are special cases of the Partial Cycle Cover (for fixed parameters k and ℓ).

To solve Long Path, we can simply iterate through all pairs of non-adjacent vertices s, t and apply the Long Cycle algorithm to the graph G with edge st added; this increases the treedepth by at most 1 and the provided elimination forest can be easily adjusted. It is easy to see that then the original graph G contains a simple path on ℓ vertices if and only if for some choice of s and t, we find a cycle of length ℓ in G augmented with the edge st. Finally, Hamiltonian Path is just Long Path applied for $\ell = |V(G)|$.

We remark that our algorithmic findings have concrete applications outside of the realm of structural parameterizations. For instance, Lokshtanov et al. [17] gave a $2^{\mathcal{O}(\sqrt{\ell}\log^2 \ell)} \cdot n^{\mathcal{O}(1)}$-time polynomial space algorithm for the Long Path problem on H-minor-free graphs, for every fixed H. In the full version of this paper [21] we present how using our results one can improve the running time to $2^{\mathcal{O}(\sqrt{\ell}\log \ell)} \cdot n^{\mathcal{O}(1)}$ while keeping the polynomial space complexity.

[2] Note, that graphs of treedepth at most k cannot contain a path of length 2^k. This leads to trivial FPT algorithms for these problems, however with doubly-exponential running time dependency on k.

[3] Note that when discussing the Long Path and the Long Cycle problems, we use the letter ℓ to denote the required length of a path, respectively of a cycle, instead of the letter k that is perhaps more traditionally used in this context.

Our Techniques. Similarly to Hegerfeld and Kratsch [16] we use the Cut&Count framework, but we apply a different view on the Count part, suited for problems based on edge selection. The main idea is that instead of counting cycle covers, as a standard application of Cut&Count would do, we count perfect matchings in an auxiliary graph, constructed by replacing every vertex with two adjacent copies. The number of such perfect matchings can be related to the number of cycle covers of the original graph. However, the considered perfect matchings can be counted within the claimed complexity by either employing the previous "algebraized" dynamic programming algorithm, or the algorithm based on inclusion-exclusion branching (our presentation chooses the latter).

Applying this approach naïvely would give us a polynomial space algorithm with running time $8^d \cdot n^{\mathcal{O}(1)}$. We improve the running time to $5^d \cdot n^{\mathcal{O}(1)}$ by employing several observations about the symmetries of recursive calls of our algorithms, in a similar way as in the algorithm for #k-Multi-Set-Cover of Nederlof [20].

Organization of the Paper. The remainder of the paper is devoted to the proof of Theorem 2. In Sect. 2 we introduce the notation and present basic definitions. In Sect. 3 we discuss the Cut&Count technique in a self-contained manner and explain the Cut part. In Sect. 4 we reduce the Count part to counting perfect matchings in an auxiliary graph. In Sect. 5 we give an intuition behind counting such matchings. We conclude with several open questions in Sect. 6. Due to space restrictions, proofs of statements marked with ♢ are deferred to the full version of this paper [21]. The full version [21] also contains applications of our results to the Long Path problem in H-minor free graphs.

2 Preliminaries

Notation. For a graph G, by $\mathtt{cc}(G)$ we denote number of connected components of G. Let F be a subset of edges of G. By $\mathtt{cc}(F)$ we denote the number of connected components of the graph consisting of all the edges of F and vertices incident to them. For a vertex u, by $\deg_F(u)$ we mean the number of edges of F incident to u. Then F is a *matching* if $\deg_F(u) \in \{0, 1\}$ for every vertex u, is a *perfect matching* if $\deg_F(u) = 1$ for every vertex u, and is a *partial cycle cover* if $\deg_F(u) \in \{0, 2\}$ for every vertex u. Note that thus we treat partial cycle covers as sets of edges.

A *cut* of a set U is just an ordered partition of U into two sets, that is, a pair (L, R) such that $L \cap R = \emptyset$ and $L \cup R = U$. A cut (L, R) of the vertex set of a graph is *consistent* with a subset of edges F if there is no edge in F with one endpoint in L and second in R.

For a function f and elements x, y, where x is not in the domain of f, by $f[x \mapsto y]$ we denote the function obtained from f by extending its domain by x and setting $f(x) = y$.

We use the $\mathcal{O}^\star(\cdot)$ notation to hide factors polynomial in the input size. For convenience, throughout the paper we assume the RAM model: every integer

takes a unit of space and arithmetic operations on integers have unit cost. However, it can be easily seen that all the numbers appearing during the computation have bit length bounded polynomially in the input size. Since we never specify the polynomial factors in the time or space complexity of our algorithms, without any influence on the claimed asymptotic bounds we may assume that the representation of any number takes polynomial space and arithmetic operations on the numbers take polynomial time.

Treedepth. A *rooted forest* is a directed acyclic graph T where every vertex has outdegree at most 1. The vertices of outdegree 0 in T are called the *roots*. Whenever a vertex u is reachable from a vertex v by a directed path in T, we say that u is an *ancestor* of v, and v is a *descendant* of u. Note that every vertex is its own ancestor as well as descendant. The *depth* of a rooted forest is the maximum number of vertices that can appear on a directed path in it.

We use the following notation from previous works [16,23]. For a vertex u of a rooted forest T, we denote:

$$\begin{aligned} \mathsf{subtree}[u] &\coloneqq \{v\colon u \text{ is ancestor of } v\}, & \mathsf{subtree}(u) &\coloneqq \mathsf{subtree}[u] \setminus \{u\},\\ \mathsf{tail}[u] &\coloneqq \{v\colon v \text{ is ancestor of } u\}, & \mathsf{tail}(u) &\coloneqq \mathsf{tail}[u] \setminus \{u\},\\ \mathsf{broom}[u] &\coloneqq \mathsf{tail}[u] \cup \mathsf{subtree}[u]. \end{aligned}$$

Additionally, $\mathsf{children}(u)$ denotes the set of children of u, whereas $\mathsf{parent}(u)$ is the *parent* of u, that is, the only outneighbor of u. If u is a root, we set $\mathsf{parent}(u) = \bot$.

For a graph G, an *elimination forest* of G is a rooted forest T on the same vertex set as G that satisfies the following property: whenever uv is an edge in G, then in T either u is an ancestor of v, or v is an ancestor of u. The *treedepth* of a graph is the minimum possible depth of an elimination forest of G.

Isolation Lemma. The only source of randomness in our algorithm is the Isolation Lemma of Mulmuley et al. [19]. Suppose U is a finite set and $\omega\colon U \to \mathbb{Z}$ is a weight function on U. We say that ω *isolates* a non-empty family of subsets $\mathcal{F} \subseteq 2^U$ if there is a unique $S \in \mathcal{F}$ such that

$$\omega(S) = \min_{X \in \mathcal{F}} \omega(X),$$

where $\omega(X) \coloneqq \sum_{x \in X} \omega(x)$. Then the Isolation Lemma can be stated as follows.

Lemma 1 (Isolation Lemma [19]). *Let U be a finite set and $\mathcal{F} \subseteq 2^U$ be a non-empty family of subsets of U. Suppose for every $u \in U$ we choose its weight $\omega(u)$ uniformly and independently at random from the set $\{1, \ldots, N\}$, where $N \in \mathbb{N}$. Then ω isolates $\mathcal{F}$ with probability at least $1 - \frac{|U|}{N}$.*

3 The Cut Part

We now proceed to the proof of Theorem 2. Throughout the proof we fix the input graph $G = (V, E)$, its elimination forest T of depth d, and numbers $k, \ell \in \mathbb{N}$. We

may assume that G is connected, as otherwise we may apply the algorithm to each connected component separately. Thus T has to be a tree, so we will call it an *elimination tree* to avoid confusion. Also, we denote $n \coloneqq |V|$.

As mentioned before, we shall apply the Cut&Count technique of Cygan et al. [7]. This technique consists of two parts: the Cut part and the Count part. The idea is that in the first part, we relax the connectivity requirements and show that it is enough to count the number of relaxed solutions together with cuts consistent with them, as this number is congruent to the number of non-relaxed solutions modulo a power of 2. The Isolation Lemma is used here to ensure that with high probability, the number of solutions does not accidentally cancel out modulo this power of 2. More precisely, having drawn a weight function at random, for each possible total weight w we count the number of solutions of total weight w. Then the Isolation Lemma asserts that, with high probability, for some w there will be a unique solution of total weight w. Then comes the Count part, where the goal is to efficiently count the number of relaxed solutions together with cuts consistent with them.

We refer the reader to [7] for a more elaborate discussion of the Cut&Count technique, while now we apply it to the particular case of Partial Cycle Cover. A relaxed solution is just a partial cycle cover consisting of ℓ edges. Then a solution is a relaxed solution that spans at most k cycles. Formally, the sets of *solutions* ($\mathcal{S}$) and *relaxed solutions* ($\mathcal{R}$) are defined as follows:

$$\mathcal{R} \coloneqq \{\, F \subseteq E \,:\, |F| = \ell \text{ and } \deg_F(u) \in \{0,2\} \text{ for every } u \in V \,\};$$
$$\mathcal{S} \coloneqq \{\, F \in \mathcal{R} \,:\, \mathtt{cc}(F) \leqslant k \,\}.$$

Suppose now that the input graph G is supplied with a weight function on edges $\omega\colon E \to \mathbb{Z}$. Then we can stratify the families above using the total weight. That is, for every $w \in \mathbb{Z}$ we define:

$$\mathcal{R}_w \coloneqq \{\, F \in \mathcal{R} \,:\, \omega(F) = w \} \qquad \text{and} \qquad \mathcal{S}_w \coloneqq \{\, F \in \mathcal{S} \,:\, \omega(F) = w \,\}.$$

Now, let

$$\mathcal{C}_w \coloneqq \{\, (F, (L,R)) \,:\, F \in \mathcal{R}_w \text{ and } (L,R) \text{ is a cut of } V \text{ consistent with } F \,\}.$$

The following observation is the key idea in the Cut&Count technique.

Lemma 2 ($\diamondsuit$). *For every $w \in \mathbb{Z}$, we have*

$$|\mathcal{C}_w| \equiv \sum_{F \in \mathcal{S}_w} 2^{n-\ell+\mathtt{cc}(F)} \mod 2^{n-\ell+k+1}.$$

In the next sections we will present the Count part of the technique, which boils down to proving the following lemma.

Lemma 3. *Given $w \in \mathbb{Z}$ and a weight function $\omega\colon E \to \{1,\ldots,N\}$, where $N = \mathcal{O}^\star(1)$, the number $|\mathcal{C}_w|$ can be computed in time $\mathcal{O}^\star(5^d)$ and space $\mathcal{O}^\star(1)$.*

In the full version of this paper [21] we show how to combine Lemma 2 with Lemma 3 to prove Theorem 2. Therefore, it remains to prove Lemma 3.

4 From Cycle Covers to Matchings

For the proof of Lemma 3, instead of counting the number of suitable partial cycle covers, we find it more convenient to count the number of perfect matchings in an auxiliary graph. Note, that this concept is natural when using *inclusion-exclusion branching* technique. A similar auxiliary graph arises in the algorithm for #k-Multi-Set-Cover [20].

We define a graph G' as follows. The vertex set V' of G' is $V' \coloneqq \{ u^0, u^1 : u \in V \}$. That is, we put two copies of each vertex of G into the vertex set of G'. The edge set E' of G' is the union of the following two sets:

$$E'_0 \coloneqq \{ u^0u^1 : u \in V \}, \qquad E'_1 \coloneqq \{ u^0v^0, u^0v^1, u^1v^0, u^1v^1 : uv \in E \}.$$

In other words, for every vertex $u \in V$ we put an edge in E'_0 connecting the two copies of u in V', while for every edge $uv \in E$ we put four different edges in E'_1, each connecting a copy of u with a copy of v in V'.

Let $\pi \colon E'_1 \to E$ be the natural projection from E'_1 to E: for each $uv \in E$ and $s, t \in \{0, 1\}$, we set $\pi(u^s v^t) = uv$. We extend the mapping π to all subsets $F \subseteq E'$ by setting $\pi(F) \coloneqq \pi(F \cap E'_1)$. We also extend the weight function ω to the edges of E' by putting $\omega(e) = 0$ for each $e \in E'_0$ and $\omega(e) = \omega(\pi(e))$ for each $e \in E'_1$.

A set of edges F in G' shall be called *simple* if for every $e \in E$, we have

$$|F \cap \pi^{-1}(e)| \leqslant 1.$$

For now, we mainly focus on simple perfect matchings in G'. We observe that they are in correspondence with partial cycle covers in G, as explained next.

Lemma 4 (♢). *For every simple perfect matching M in G', the set $\pi(M)$ is a partial cycle cover in G of size $|M \cap E'_1|$. Moreover, for every partial cycle cover F in G, there are exactly $2^{|F|}$ simple perfect matchings M in G' for which $F = \pi(M)$.*

Lemma 4 motivates introducing the following analogues of the sets $\mathcal{C}_w$. For $w \in \mathbb{Z}$, we define

$$\begin{aligned} \mathcal{M}_w \coloneqq \{ (M, (L, R)) \colon\ & M \text{ is a simple perfect matching in } G', \\ & (L, R) \text{ is a cut of } V \text{ consistent with } \pi(M), \\ & |M \cap E'_1| = \ell \text{ and } \omega(M) = w \}. \end{aligned}$$

Since for every simple perfect matching M in G' we have $\omega(M) = \omega(\pi(M))$, from Lemma 4 we immediately obtain the following.

Corollary 1. *For every $w \in \mathbb{Z}$, we have $|\mathcal{M}_w| = 2^\ell \cdot |\mathcal{C}_w|$.*

Therefore, to prove Lemma 3 it suffices to apply the algorithm provided by the following lemma and divide the outcome by 2^ℓ.

Lemma 5 (♢). *Given $w \in \mathbb{Z}$ and a weight function $\omega \colon E \to \{1, \ldots, N\}$, where $N = \mathcal{O}^\star(1)$, the number $|\mathcal{M}_w|$ can be computed in time $\mathcal{O}^\star(5^d)$ and space $\mathcal{O}^\star(1)$.*

5 The Count Part

Due to space restrictions we defer the formal proof of Lemma 5 to the full version of this paper [21]. In this section we discuss only the intuition behind the approach.

The basic idea is that we will compute the number $|\mathcal{M}_w|$ using bottom-up dynamic programming over the given elimination tree T. In order to achieve polynomial space complexity, this dynamic programming will be cast as a standard recursion, but for this to work, we need that the recurrence equations governing the dynamic programming have a specific form. In essence, whenever we compute an entry of the dynamic programming table at some vertex u, the value should be obtained as a simple aggregation of single entries from the tables of the children of u. The most straightforward approach to computing $|\mathcal{M}_w|$ would be to count partial perfect matchings and to remember, in the states corresponding to u, subsets of $\mathsf{tail}[u]$ consisting of vertices matched to $\mathsf{subtree}(u)$. This would yield a dynamic programming algorithm that is *not* of the form required for the space complexity reduction. However, we show that by counting different objects than partial perfect matchings, and using the inclusion-exclusion principle at every computation step, we can reorganize the computation so that the space reduction is possible.

We remark that even though at the end of the day our algorithm relies only on basic ideas such as branching and inclusion-exclusion, there is a deeper intuition behind the definitions of the computed values. In fact, from the right angle our algorithm can be seen as an application of the technique of *saving space by algebraization*, introduced by Lokshtanov and Nederlof [18], which boils down to applying the Fourier transform on the lattice of subsets in order to turn subset convolutions into pointwise products. We refer the reader to [1,14,16,23] for other applications of this technique in the context of treedepth-based algorithms.

6 Conclusion and Further Research

In this paper we answered the open question of Hegerfeld and Kratsch [16] by presenting an $\mathcal{O}^\star(5^d)$-time and polynomial space algorithm for Hamiltonian Path, Hamiltonian Cycle, Longest Path, Longest Cycle Min Cycle Cover, where d is the depth of a provided elimination forest of the input graph. However, there are still multiple open problems around time- and space-efficient algorithms on graphs of bounded treedepth. We list here a selection.

Approximation of Treedepth. Recall that the treewidth of a graph can be approximated up to a constant factor in fixed-parameter time. For instance, the classic algorithm of Robertson and Seymour [26] (see also [5]) takes on input a graph G and integer t, works in time $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ and in polynomial space, and either concludes that the treewidth of G is larger than t, or finds a tree decomposition of G of width at most $4t + 4$. This means that for the purpose of designing $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$-time algorithms on graphs of treewidth t, we may assume that

a tree decomposition of approximately optimum width is given, as it can be always computed from the input graph within the required complexity bounds. Unfortunately, no such approximation algorithm is known for the treedepth. Namely, it is known that the treedepth can be computed exactly in time and space $2^{\mathcal{O}(d^2)} \cdot n$ [25] and approximated up to factor $\mathcal{O}(t \log^{3/2} t)$ in polynomial time [8], where d and t are the values of the treedepth and the treewidth of the input graph, respectively. A piece of the theory that seems particularly missing is a constant-factor approximation algorithm for treedepth running in time $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$; polynomial space usage would be also desired.

Faster Algorithms. The bases of the exponent of the running times of the algorithms given by Hegerfeld and Kratsch [16] for the treedepth parameterization match the ones obtained by Cygan et al. [7] for the treewidth parameterization. In the case of our results, the situation is different: while Hamiltonian Cycle can be solved in time $4^t \cdot n^{\mathcal{O}(1)}$ in graphs of treewidth t [7] and in time $(2+\sqrt{2})^p \cdot n^{\mathcal{O}(1)}$ in graphs of pathwidth p [6], we needed to increase the base of the exponent to 5 in order to achieve polynomial space complexity for the treedepth parameterization. As the treedepth of a graph is never smaller than its pathwidth, it is natural to ask whether there is an $(2+\sqrt{2})^d \cdot n^{\mathcal{O}(1)}$-time polynomial-space algorithm for Hamiltonian Cycle on graphs of treedepth d. In fact, reducing the base 5 to any $c < 5$ would be interesting.

Derandomization. Shortly after its introduction, the Cut&Count technique for the treewidth parameterization has been derandomized. Bodlaender et al. [4] presented two approaches for doing so. The first one, called the *rank-based approach*, boils down to maintaining a small set of representative partial solutions along the dynamic programming computation, and pruning irrelevant partial solutions on the fly using Gaussian elimination. Fomin et al. [13] later reinterpreted this technique in the language of matroids and extended it. The second approach, called *determinant-based*, uses the ideas behind Kirchoff's matrix-tree theorem to deliver a formula for counting suitable spanning trees of a graph, which can be efficiently evaluated by a dynamic programming over a tree decomposition.

It seems to us that none of these approaches applies in the context of the treedepth parameterization, where we additionally require polynomial space complexity. For the rank-based and matroid-based approaches, they are based on keeping track of a set of representative solutions, which in the worst case may have exponential size. In the determinant-based approach, when computing the formula for the number of spanning trees over a tree decomposition, the aggregation of dynamic programming tables is done using operations that are algebraically more involved, and which in particular are non-commutative. See the work of Włodarczyk [27] for a discussion. It is unclear whether this computation can be reorganized so that in the aggregation we use only pointwise product—which, in essence, is our current methodology from the algebraic perspective.

Hence, it is highly interesting whether our algorithm, or the algorithms of Hegerfeld and Kratsch [16], can be derandomized while keeping running time $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ and polynomial space usage.

Other Graph Parameters. Actually, Hegerfeld and Kratsch [16] were not the first to employ Cut&Count on structural graph parameters beyond treewidth. Pino et al. [24] used Cut&Count and rank-based approach to get single-exponential time algorithms for connectivity problems parametrized by *branch-width*. Recently, Cut&Count was also applied in the context of *cliquewidth* [2], and of $\mathbb{Q}$-*rankwidth*, *rankwidth*, and *MIM-width* [3]. All these algorithms have exponential space complexity, as they follow the standard dynamic programming approach. One may expect that maybe for the depth-bounded counterparts of cliquewidth and rankwidth—*shrubdepth* [15] and *rankdepth* [9]—time-efficient polynomial-space algorithms can be designed, similarly as for treedepth.

Acknowledgements. We would like to thank anonymous reviewers and Petr Hliněný for their suggestions and comments. The second author would like to thank Marcin Wrochna for some early discussions on combining treedepth and Cut&Count. The work leading to the results presented in this paper was initiated during the Parameterized Retreat of the algorithms group of the University of Warsaw (PARUW), held in Karpacz in February 2019. This Retreat was financed by the project CUTACOMBS, which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 714704).

Jesper Nederlof is supported by the project CRACKNP that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 853234). Michał Pilipczuk is supported by the project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651). Céline M. F. Swennenhuis is supported by the Netherlands Organization for Scientific Research under project no. 613.009.031b. Karol Węgrzycki is supported by the grants 2016/21/N/ST6/01468 and 2018/28/T/ST6/00084 of the Polish National Science Center and project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 677651).

References

1. Belbasi, M., Fürer, M.: A space-efficient parameterized algorithm for the hamiltonian cycle problem by dynamic algebraization. In: van Bevern, R., Kucherov, G. (eds.) CSR 2019. LNCS, vol. 11532, pp. 38–49. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19955-5_4
2. Bergougnoux, B., Kanté, M.M.: Fast exact algorithms for some connectivity problems parameterized by clique-width. Theor. Comput. Sci. **782**, 30–53 (2019)

3. Bergougnoux, B., Kanté, M.M.: More applications of the d-neighbor equivalence: connectivity and acyclicity constraints. In: 27th Annual European Symposium on Algorithms, ESA 2019. LIPIcs, vol. 144 , pp. 17:1–17:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2019)
4. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015)
5. Cygan, M.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
6. Cygan, M., Kratsch, S., Nederlof, J.: Fast Hamiltonicity checking via bases of perfect matchings. J. ACM **65**(3), 12:1–12:46 (2018)
7. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, pp. 150–159. IEEE (2011)
8. Czerwiński, W., Nadara, W., Pilipczuk, M.: Improved bounds for the excluded-minor approximation of treedepth. In: 27th Annual European Symposium on Algorithms, ESA 2019. LIPIcs, vol. 144, pp. 34:1–34:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2019)
9. DeVos, M., Kwon, O., Oum, S.: Branch-depth: Generalizing tree-depth of graphs. arXiv preprint, abs/1903.11988 (2019)
10. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. TCS. Springer, London (2013). https://doi.org/10.1007/978-1-4471-5559-1
11. Drucker, A., Nederlof, J., Santhanam, R.: Exponential time paradigms through the polynomial time lens. In: 24th Annual European Symposium on Algorithms, ESA 2016. LIPIcs, vol. 57, pp. 36:1–36:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2016)
12. Flum, J., Grohe, M.: Parameterized Complexity Theory. TTCSAES. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29953-X
13. Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4), 29:1–29:60 (2016)
14. Fürer, M., Yu, H.: Space saving by dynamic algebraization based on tree-depth. Theor. Comput. Syst. **61**(2), 283–304 (2017)
15. Ganian, R., Hliněný, P., Nešetřil, J., Obdržálek, J., de Mendez, P.O.: Shrub-depth: capturing height of dense graphs. Log. Meth. Comput. Sci. **15**(1), 71–725 (2019)
16. Hegerfeld, F., Kratsch, S.: Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. arXiv preprint, abs/2001.05364 (2020). Accepted to STACS 2020
17. Lokshtanov, D., Mnich, M., Saurabh, S.: Planar k-path in subexponential time and polynomial space. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 262–270. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25870-1_24
18. Lokshtanov, D., Nederlof, J.: Saving space by algebraization. In: 42nd ACM Symposium on Theory of Computing, STOC 2010, pp. 321–330. ACM (2010)
19. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. Combinatorica **7**(1), 105–113 (1987)
20. Nederlof, J.: Inclusion exclusion for hard problems. Master's thesis, Utrecht University (2008)

21. Nederlof, J., Pilipczuk, M., Swennenhuis, C.M.F., Wegrzycki, K.: Hamiltonian cycle parameterized by treedepth in single exponential time and polynomial space. CoRR, abs/2002.04368 (2020)
22. Pilipczuk, M., Siebertz, S.: Polynomial bounds for centered colorings on proper minor-closed graph classes. In: 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, pp. 1501–1520. SIAM (2019)
23. Pilipczuk, M., Wrochna, M.: On space efficiency of algorithms working on structural decompositions of graphs. ACM Trans. Comp. Theor. **9**(4), 181–1836 (2018)
24. Pino, W.J.A., Bodlaender, H.L., van Rooij, J.M.M.: Cut and Count and representative sets on branch decompositions. In: 11th International Symposium on Parameterized and Exact Computation, IPEC 2016. LIPIcs, vol. 63, pp. 27:1–27:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2016)
25. Reidl, F., Rossmanith, P., Villaamil, F.S., Sikdar, S.: A faster parameterized algorithm for treedepth. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8572, pp. 931–942. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_77
26. Robertson, N., Seymour, P.D.: Graph Minors. XIII. The disjoint paths problem. J. Comb. Theor. Ser. B **63**(1), 65–110 (1995)
27. Włodarczyk, M.: Clifford algebras meet tree decompositions. Algorithmica **81**(2), 497–518 (2019)