Designing Deterministic Polynomial-Space Algorithms by Color-Coding Multivariate Polynomials*

Gregory Gutin[1], Felix Reidl[1], Magnus Wahlström[1], and Meirav Zehavi[2]

[1]Royal Holloway, University of London, TW20 0EX, UK
[2]University of Bergen, Norway

Abstract

In recent years, several powerful techniques have been developed to design *randomized* polynomial-space parameterized algorithms. In this paper, we introduce an enhancement of color coding to design deterministic polynomial-space parameterized algorithms. Our approach aims at reducing the number of random choices by exploiting the special structure of a solution. Using our approach, we derive polynomial-space $\mathcal{O}^*(3.86^k)$-time (exponential-space $\mathcal{O}^*(3.41^k)$-time) deterministic algorithm for k-Internal Out-Branching, improving upon the previously fastest *exponential-space* $\mathcal{O}^*(5.14^k)$-time algorithm for this problem. (The notation $\mathcal{O}^*$ hides factors polynomial in the input size.) We also design polynomial-space $\mathcal{O}^*((2e)^{k+o(k)})$-time (exponential-space $\mathcal{O}^*(4.32^k)$-time) deterministic algorithm for k-Colorful Out-Branching on arc-colored digraphs and k-Colorful Perfect Matching on planar edge-colored graphs. In k-Colorful Out-Branching, given an arc-colored digraph D, decide whether D has an out-branching with arcs of at least k colors. In k-Colorful Perfect Matching, given an undirected graph G, decide whether G has a perfect matching with edges of at least k colors. To obtain our polynomial space algorithms, we show that $(n, k, \alpha k)$-splitters ($\alpha \geqslant 1$) and in particular (n, k)-perfect hash families can be enumerated one by one with polynomial delay using polynomial space.

1 Introduction

In this paper, we modify color coding to treat multivariate polynomials, and thus design an improved deterministic polynomial-space algorithm for k-Internal

*Gutin was partially supported by Royal Society Wolfson Research Merit Award, Reidl and Wahlström by EPSRC grant EP/P007228/1, and Zehavi by ERC Grant Agreement no. 306992.

Out-Branching (k-IOB). Before we elaborate on this problem and our contribution, let us first review related previous works that motivate our study. In recent years, several powerful algebraic techniques have been developed to design *randomized* polynomial-space parameterized algorithms. The first approach was introduced by Koutis [1], strengthened by Williams [2], and is nowadays known as the multilinearity detection technique [3]. Roughly speaking, an application of this technique consists of reducing the problem at hand to one where the objective is to decide whether a given polynomial has a multilinear monomial, and then employing an algorithm for the latter problem as a black box.

One of the huge breakthroughs brought about by this line of research was Björklund's [4] proof that Hamiltonian Path is solvable in time $\mathcal{O}^*(1.66^n)$ by a randomized algorithm, improving upon the 50 year old $\mathcal{O}^*(2^n)$-time[1] algorithm [5]. The existence of a *deterministic* $\mathcal{O}^*((2-\varepsilon)^n)$-time algorithm for Hamiltonian Path, for a fixed $\varepsilon > 0$, is still a major open problem. Further, Björklund's result is on undirected graphs, and the existence of an $\mathcal{O}^*((2-\varepsilon)^n)$-time algorithm for Hamiltonian Path on digraphs, for a fixed $\varepsilon > 0$, is another interesting open problem.

Shortly afterwards, Björklund *et al.* [6] have transformed the ideas in [4] into a powerful technique to design randomized polynomial-space algorithms, referred to as *narrow sieves*. This technique is also based on the analysis of polynomials, but it is applied quite differently. Here one associates a monomial with each "potential solution" in such a way that actual solutions correspond to unique monomials while incorrect solutions appear in pairs. Thus, the polynomial summing these monomials, when evaluated over a field of characteristic 2, is not identically 0 if and only if the input instance of the problem at hand is a yes-instance. In this context, the relevance of the Matrix Tree Theorem was already noted by Gabizon *et al.* [7].

The narrow sieves technique, proven to be of wide applicability on its own, later branched into several new methods. The one most relevant to our study was developed by Björklund *et al.* [8] and was translated into the language of determinants by Wahlström [9]. Here, the studied problem was S-Cycle (or S-Path), where the goal is to determine whether an input graph contains a cycle that passes through all the vertices of an input set S of size k. Wahlström [9] considered a determinant-based polynomial (computed over a field of characteristic 2), and analyzed whether there exists a monomial where the variable-set representing S is present. Very recently, Björklund *et al.* [10] utilized the Matrix Tree Theorem to improve an FPT algorithm for k-IOB, where we are asked to decide whether a given digraph has a k-internal out-branching. Recall that an *out-tree* T is an orientation of a tree with only one vertex of in-degree zero (called the *root*). A vertex of T is a *leaf* if its out-degree in T is zero; non-leaves are called *internal vertices*. An *out-branching* of a digraph D is a spanning subgraph of D, which is an out-tree, and an out-branching is *k-internal* if it hasat least k internal vertices.

Björklund *et al.* [10] cleverly transformed k-IOB into a new problem, where

[1] We use the common notation $\mathcal{O}^*$ to hide factors polynomial in the input size.

the goal is to decide whether a given polynomial (computed over a field of characteristic 2 to avoid subtractions) has a monomial with at least k distinct variables.

In this paper, we present an easy-to-use[2] modification of color coding for designing deterministic polynomial-space parameterized algorithms, inspired by the principles underlying the above mentioned techniques. (A slight modification of our approach can be used to design faster, exponential-space algorithms, but we believe that the main value of the approach is for polynomial-space algorithms.) We will show that our approach brings significant speed-ups to algorithms for k-IOB. Roughly speaking, our approach can be applied as follows.

- Identify a polynomial such that it has a monomial with at least k distinct variables (called a *witnessing monomial*) if and only if the input instance of the problem at hand is a yes-instance. It should be possible to efficiently evaluate the polynomial (black box-access is sufficient here).
- Color the variables of the polynomial with k colors using a polynomial-delay perfect hash-family. To improve the running time of this step, we apply a problem-specific *coloring guide* to reduce the number of 'random' colors. Given a k-coloring, we obtain a smaller polynomial by identifying all variables of the same color.
- Use inclusion-exclusion to extract the coefficient of a colorful monomial from the reduced polynomial. By the usual color-coding arguments, if the coefficient is not equal to zero then the original polynomial contained a witnessing monomial.

While we were unable to obtain non-trivial coloring guides to the following problems, even limited application of our approach is useful for designing polynomial-space algorithms for these problems. It would be interesting to obtain non-trivial coloring guides for the problems.

Colorful Out-Branchings and Matchings Every subgraph-search problem can be extended quite naturally by imposing additional constraints on the solution, for example by letting the input graph have labels or weights. One class of such constraints states that a required subgraph of an edge-colored graph has to be *k-colorful*, i.e. to contain edges of at least k colors.

One prominent problem is Rainbow Matching[3](also known as Multiple Choice Matching), defined in the classical book by Garey and Johnson [11]. Itai *et al.* [12] showed, already in 1978, that Rainbow Matching is NP-complete on bipartite graphs. Three decades later, Le and Pfender [13] revisited this problem and showed that it is NP-hard on several restricted graph classes, which include (among others) paths, complete graph and P_4-free bipartite graphs in which every color is used at most twice. Further examples of

[2]In particular, no dynamic programming/recursive algorithms are required.

[3]In the problem, given an edge-colored graph G and an integer k, the aim is to decide whether G has a k-colorful matching of size k.

subgraph problems with color constraints can be found in a survey by Mikio and Xueliang [14]. In this paper, we focus on two color-constrained problems: given an edge-colored graph and an integer k, we ask for either a k-colorful spanning tree/outbranching or a k-colorful perfect matching.

We first rely on the Matrix Tree Theorem to present a deterministic polynomial-space $\mathcal{O}^*((2e)^{k+o(k)})$-time (exponential-space $\mathcal{O}^*(4.32^k)$-time) algorithm for k-Colorful Out-Branching, defined as follows:

Colorful Out-Branching parametrised by k

Input:	A arc-colored digraph D and an integer k
Problem:	Does D have a k-colorful out-branching?

We argue in Section 5 that k-Colorful Out-Branching is NP-hard on various restricted graph classes such as cubic graphs.

Next, we rely on a Pfaffian computation to present a deterministic polynomial-space $\mathcal{O}^*((2e)^{k+o(k)})$-time (exponential-space $\mathcal{O}^*(4.32^k)$-time) algorithm for k-Colorful Perfect Matching on planar graphs, defined as follows:

Planar Colorful Perfect Matching parametrised by k

Input:	A planar edge-colored graph G and an integer k
Problem:	Does G have a k-colorful perfect matching?

We will show in Section 6 by a simple reduction that Planar k-Colorful Perfect Matching is NP-hard even on planar graphs of pathwidth 2. It is worthwhile to note that while Rainbow Matching can be viewed as a special case of the well-known 3-Set k-Packing problem, and in particular, a solution for Rainbow Matching is small (containing only $2k$ vertices and k colors), the case of k-Colorful Perfect Matching is different in the sense that a solution is necessarily large since not every k-colorful matching can be extended to a perfect matching.

k-Internal Out-Branching By utilizing the method of bounded search trees on top of the above machinery, in Section 4, we present a deterministic polynomial-space $\mathcal{O}^*(3.86^k)$-time (exponential-space $\mathcal{O}^*(3.41^k)$-time) algorithm for the problem k-Internal Out-Branching (k-IOB), defined as follows:

Internal Out-Branching (IOB) parametrised by k

Input:	A digraph D and an integer k
Problem:	Does D have a k-internal out-branching?

The undirected version of k-IOB, called k-Internal Spanning Tree (k-IST), is defined similarly.

Internal Spanning Tree (IST) parametrised by k

Input: A graph G and an integer k
Problem: Does G have a k-internal spanning tree?

Note that k-IOB is a generalization of k-IST since the latter can easily be reduced to k-IOB on symmetric digraphs, i.e. digraphs in which every arc is on a directed cycle of length 2. Since k-IST is NP-hard (Hamiltonian Path appears as a special case for $k = n - 2$) it follows that so is k-IOB. The latter is, however, polynomial time solvable on acyclic digraphs [15]. While acyclic digraphs are precisely digraphs of directed treewidth 0, it turns out that k-IOB is NP-hard already for digraphs of directed treewidth 1 [16]. By constrast, the Directed Hamilton Path problems is polynomial-time solvable on digraphs of directed treewidth t [17] if t is a constant. The k-IOB problem a priori seems more difficult than the well-known k-Path problem (decide whether a given digraph has a path on k vertices) since a witness of a yes-instance of k-path is a subgraph of size k, that is, a path on k vertices. However, it is easy to see that a witness of a yes-instance of k-IOB (which has an out-branching) can be a subgraph of size $2k - 1$, that is, an out-tree with k internal vertices and $k - 1$ leaves. This simple but crucial observation lies at the heart of previous algorithms for k-IOB.

Reference	Det./Rand.	Space	Graph	Time $\mathcal{O}^*(\cdot)$
Prieto *et al.* [18]	*det*	*poly*	*undirected*	$2^{O(k \log k)}$
Gutin *et al.* [15]	*det*	*exp*	*directed*	$2^{O(k \log k)}$
Cohen *et al.* [19]	*det*	*exp*	*directed*	55.8^k
	rand	*poly*	*directed*	49.4^k
Fomin *et al.* [20]	*det*	*exp*	*directed*	$16^{k+o(k)}$
	rand	*poly*	*directed*	$16^{k+o(k)}$
Fomin *et al.* [21]	*det*	*poly*	*undirected*	8^k
Shachnai *et al.* [22]	*det*	*exp*	*directed*	6.855^k
Daligault [23]	*rand*	*poly*	*directed*	4^k
Li *et al.* [24]	*det*	*poly*	*undirected*	4^k
Zehavi [25]	*det*	*exp*	*directed*	5.139^k
	rand	*exp*	*directed*	3.617^k
Björklund *et al.* [26]	*rand*	*poly*	*undirected*	3.455^k
Björklund *et al.* [10]	*rand*	*poly*	*undirected*	2^k
Our work	*det*	*poly*	*directed*	3.86^k
	det	*exp*	*directed*	3.41^k

Table 1: Previously known FPT algorithms for k-IOB and k-IST.

Parameterized algorithms for k-IST and k-IOB were first studied by Prieto and Sloper [18] and Gutin *et al.* [15], who proved that both problems are *fixed-*

parameter tractable (FPT), i.e. admit deterministic algorithms of running time $\mathcal{O}^*(f(k))$, where $f(k)$ is an arbitrary recursive function depending on the *parameter* k only. Moreover, both papers showed that $f(k) = 2^{O(k \log k)}$. Since then several papers improved complexities of deterministic and randomized algorithms for both problems; we list these algorithms in Table 1. We also remark that approximation algorithms, exact exponential-time algorithms and kernelization algorithms for both k-IOB and k-IST were extensively studied, but the survey of such results is beyond the scope of this paper.

Our polynomial-space algorithm for k-IOB is faster, in terms of $f(k)$, than not only the previously fastest *exponential-space* $O^*(5.139^k)$-time deterministic algorithm for k-IOB by Zehavi [25], but also the previously fastest polynomial-space $\mathcal{O}^*(4^k)$-time deterministic algorithm for k-IST of Li *et al.* [24]. In contrast, it is not known how to design a deterministic polynomial-space $\mathcal{O}^*((4-\varepsilon)^k)$-time algorithm for the k-Path problem. Indeed, a deterministic polynomial-space $\mathcal{O}^*(4^{k+o(k)})$-time algorithm for k-Path has been known since 2006 [27],[4] yet so far no improvements without the help of exponential space [28, 25] have been made.

The rest of the paper is organized as follows. In the next section, we introduce necessary preliminary material. Section 3 describes our approach in detail. The main contents of the next three sections were discussed above. In Section 7, we prove a derandomization theorem forming part of our approach. In Section 8, we show a proposition, which assists us in upper-bounding running times of exponential-space algorithms. We conclude the paper in Section 9 discussing some open problems.

2 Preliminaries

We assume the reader is familiar with basic concepts and notations in Graph Theory and Linear Algebra, and refer readers to textbooks in Graph Theory [29, 30] and Linear Algebra [31] if additional details are required.

We will make use of the tighter version of Stirling's approximation due to Robbins [32], which states that

$$\sqrt{2\pi k}\big(\frac{k}{e}\big)^k e^{1/(12k+1)} \leqslant k! \leqslant \sqrt{2\pi k}\big(\frac{k}{e}\big)^k e^{1/12k}.$$

For $\alpha \geqslant 1$ we will frequently use the function

$$\tau(\alpha) = \Big(1 - \frac{1}{\alpha}\Big)^{\alpha-1} e,$$

with the convention that $\tau(1) = e$. We will sometimes use the symbol $\bullet$ in the following to denote a variable whose value is arbitrary.

[4] Although Chen et al. [27] do not explicitly prove that the space complexity of their deterministic algorithm can be made polynomial as well, this knowledge has become folklore.

Operations on polynomials For a polynomial P and a monomial M, we let $\text{coef}_P(M)$ denote the coefficient of M in P. For a polynomial $P(x_1, \ldots, x_n)$ and subset of the variables $C \subseteq \{x_1, \ldots, x_n\}$, let P/C denote the polynomial obtained from P by replacing all variables in C by a new variable y_C. We extend this notation to partitions $\mathcal{C} := C_1 \uplus \ldots \uplus C_p$ and let $P/\mathcal{C}$ denote the polynomial $(\ldots((P/C_1)/C_2)\ldots)/C_p$.

Structural Observation It is not hard to decide whether a digraph D has an out-branching in linear time: D contains an out-branching if and only if D has only one strongly connected component without incoming arcs (see, e.g., [29]). Thus, in what follows, whenever we discuss problems where a solution is in particular an out-branching, we will assume that the digraph D under consideration contains an out-branching.

Matrix Tree Theorem In 1948 Tutte [33] proved the (Directed) Matrix Tree Theorem, which shows that the number of out-branchings rooted at the same vertex r in a digraph D can be found efficiently by calculating the determinant of a certain matrix derived from D. Here, we require a generalization of this theorem, whose derivation from the original theorem is folklore (a proof can be found, e.g., in [10]).

The *(symbolic) Kirchoff matrix* $K = K(D)$ of a directed multigraph D on n vertices is defined as follows, where we assume that the vertices are numbered from 1 to n:

$$K_{ij} = \begin{cases} \sum_{\ell i \in A(D)} x_{\ell i} & \text{if } i = j, \\ -x_{ij} & \text{if } ij \in A(D), \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $A(D)$ is the arc set of D.

In what follows, $[n] := \{1, 2, \ldots, n\}$. For $i \in [n]$ we denote by $K_{\bar{i}}(D)$ the matrix obtained from $K(D)$ by deleting the ith row and the ith column. Moreover, let $\mathcal{B}_i$ denote the set of out-branchings rooted at i. The following version of the (Directed) Matrix Tree Theorem implies a natural one-to-one correspondence between the monomials of $\det(K_{\bar{i}}(D))$ and the out-branchings in $\mathcal{B}_i$.

Theorem 1. *For every directed multigraph D with symbolic Kirchoff matrix $K(D)$ and $i \in V(D)$,* $\det(K_{\bar{i}}(D)) = \sum_{B \in \mathcal{B}_i} \prod_{ij \in A(B)} x_{ij}$.

Planar Graphs, Perfect Matchings and Pfaffians The Pfaffian is an important tool for polynomial-time counting algorithms, closely related to perfect matchings of a graph. A square matrix $M \in \mathbb{R}^{n \times n}$ is *skew-symmetric* if for every $i, j \in [n]$ it holds that $M(i, j) = -M(j, i)$. Let M be skew-symmetric and of even dimension $2n$. For every partition of $[2n]$ into pairs $\{\{i_a, j_a\} \mid a \in [n]\}$, define a corresponding permutation $(i_1, j_1, \ldots, i_n, j_n)$ of $[2n]$ where $i_a < j_a$ for every $a \in [n]$ and $i_a < i_{a+1}$ for every $a \in [n-1]$. Note that this is unique for

every partition into pairs, and let Π_n denote the set of permutation of partitions of $[2n]$. Then the *Pfaffian* of M can be defined as

$$\mathrm{pf}(M) = \sum_{\pi \in \Pi_n} \sigma(\pi) \prod_{t=1}^{n} M(\pi(2t-1), \pi(2t)),$$

where $\sigma(\pi) = \pm 1$ is a sign term referred to as the *signature* of the permutation. The Pfaffian can be efficiently computed; in particular, $\mathrm{pf}(M) = (\det(M))^2$. Among other applications, the Pfaffian can be used to count the number of perfect matchings of a planar graph by computing an orientation of the graph where every signature in the above sum is $+1$; see Section 6.

Hash Families and Splitters The notion of perfect hash family was employed by Alon *et al.* [34] when they introduced the framework of color coding. Splitters are generalizations of perfect hash families; both are defined below.

Let $\bar{x} = (x_1, \ldots, x_n) \in [t]^n$ be a vector and $I = \{p_1, \ldots, p_{|I|}\} \subseteq [n]$, where $p_1 < \cdots < p_{|I|}$. Then $\bar{x}[I] := (x_{p_1} \ldots, x_{p_{|I|}})$. We say that $\bar{x}[I]$ *partitions* I *almost equally* if the cardinalities of the sets $\{i \in I : \ x_i = q\}$, $q \in [t]$ differ from each other by at most 1.

Definition 2 (Splitter). *An* (n, k, t)-splitter *is a family of vectors* $\mathcal{S} \subseteq [t]^n$ *such that for every index set* $I \in \binom{[n]}{k}$ *there exists at least one vector* $\bar{x} \in \mathcal{S}$ *such that* $\bar{x}[I]$ *partitions* I *almost equally. The* size *of the splitter is* $|\mathcal{S}|$.

Definition 3 (Perfect hash family). *An* (n, k, k)*-splitter* $\mathcal{S} \subseteq [k]^n$ *is called an* (n, k)-perfect hash family. *In other words, for every index set* $I \in \binom{[n]}{k}$ *there exists a vector* $\bar{x} \in \mathcal{S}$ *such that* $\bar{x}[I]$ *is a permutation of* $[k]$.

Note that the members of an (n, k, t)-splitter can equivalently be interpreted as vectors from $[t]^n$, as functions that map $[n]$ into $[t]$, or as partitions of $[n]$ into t blocks.

3 Our Approach

Let us now elaborate on the four main steps that constitute our approach.

① **Polynomial Identification**
Associate variables $X = \{x_1, x_2, \ldots, x_n\}$ with some elements (e.g., vertices or edges) of the input instance and identify a polynomial $P(x_1, \ldots, x_n)$ over the reals that satisfies the following properties.

- $P(x_1, \ldots, x_n)$ has a monomial with at least k distinct variables, called a *witnessing monomial* W, if and only if the input instance is a yes-instance;
- All witnessing monomials of $P(x_1, \ldots, x_n)$ (in standard form) have non-negative coefficients;

– For any partition $\mathcal{C}$ of X into k blocks and any assignment of the variables $y_1, \ldots, y_k$ (y_i corresponds to block i), the polynomial $(P/\mathcal{C})(y_1, \ldots, y_k)$ can be evaluated efficiently using polynomial space.[5]

② **Derivation of Coloring Guides**
Define a partition $\mathcal{S} = S_1 \uplus S_2 \ldots S_p \uplus S_\perp$ of the variables X with the following property: if there exists a witnessing monomial W, then in particular there exists such a monomial with $|W \cap S_i| = 1$ for $i \in [p]$ and $|W \cap S_\perp| = k - p$. We say that such a partition $\mathcal{S}$ is a *coloring guide*. Note that we might consider more than one guide, e.g. by a suitable branching procedure. In this case, we apply the following steps to all the generated guides and the above property needs to hold for at least one of the generated guides.

③ **Color Coding & Derandomization**
Color the variables $S_\perp$ with $k-p$ colors uniformly at random and let the resulting color partition be $\mathcal{C} := C_1 \uplus C_2 \uplus \ldots \uplus C_k$ (where $C_i = S_i$ for $i \leqslant p$ and $\bigcup_{i>p} C_i = S_\perp$). To derandomize this step, use an $(n, k-p)$-perfect hash family $\mathcal{F}$ that is enumerable with polynomial space (*cf.* Theorem 4 stated shortly) to color $S_\perp$. Proceed with the next step for every coloring $\mathcal{C}$.

④ **Coefficient Extraction**
Test whether $P/\mathcal{C}$ contains a monomial W in which all variables $y_1, \ldots, y_k$ appear with a standard inclusion-exclusion algorithm. For example, we can apply Lemma 6 stated shortly for $P/\mathcal{C}$ by evaluating the corresponding $Q(y_1, \ldots, y_k) \neq 0$ at $y_i = 1$ for all $i \in [k]$. Clearly, such a W exists if and only if $Q(1, 1, \ldots, 1) \neq 0$. If so, conclude that P contains a witnessing monomial and return that the instance is a yes-instance.

We remark that Naor *et al.* [35] proved that an (n, k)-perfect hash family of size $\mathcal{O}^*(e^{k+o(k)})$ can be computed in time $\mathcal{O}^*(e^{k+o(k)})$. They claimed that their construction can be modified so that it is not required to compute the family "at once", but the vectors in the family can be enumerated with polynomial delay. However, a proof of the latter claim has not been published (the proof was deferred to a full version of that paper which never appeared).

In Section 7 we prove a more general theorem from which this claim can be derived as a corollary. Importantly, our construction requires only polynomial space, a feature that the construction by Naor *et al.* does not achieve.

Theorem 4. *There exists an $(n, k, \alpha k)$-splitter of size $k^{O(1)}\tau(\alpha)^{k+o(k)} \log n$ for every $\alpha \geqslant 1$. Moreover, the members of the family can be enumerated deterministically with polynomial delay using polynomial space.*

In case we are interested in polynomial space, we compute $\mathcal{F}$ simply as an $(n, k-p)$-perfect hash family. Otherwise, we compute $\mathcal{F}$ as an $(n, k-p, \alpha^\star(k-p))$-splitter with a suitable constant $\alpha^\star \approx \frac{4}{3}$. In the latter scenario, we can run

[5] The evaluation may use operations such as subtraction and division.

Steps ③ and ④ of our approach in time $\mathcal{O}^*(\tau(\alpha^\star)^{k-p+o(k)}2^{\alpha^\star(k-p)+p})$. This requires exponential space, as there are $\binom{\alpha^\star(k-p)+p}{k}$ monomials for which we need to check divisibility (see Lemma 6), but for every $I \subseteq [\alpha^\star(k-p)+p]$, we would evaluate P_{-I} only once rather than once per such monomial. Then, we can rely on the following result proved in Section 8.

Proposition 5. *There exists a constant $\alpha^\star \approx \frac{4}{3}$ such that using an $(n, k-p, \alpha^\star(k-p))$-splitter and exponential space, we can run Steps ③ and ④ of our approach in time $\mathcal{O}^*(4.312^{k-p}2^p)$.*

In the context of randomized algorithms, a slightly weaker result ($\mathcal{O}^*(4.314^k)$) was obtained by Hüffner *et al.* [36].

The following Lemma 6 provides a way to extract from P only monomials divided by a certain term, the crucial operation in step ④. The lemma is a modification of a lemma by Wahlströhm [9] for polynomials over a field of characteristic two.

Lemma 6 (Monomial sieving). *Let $P(x_1, \ldots, x_n)$ be a polynomial over reals, and $J \subseteq [n]$ a set of indices. For a set $I \subseteq [n]$, define $P_{-I}(x_1, \ldots, x_n) = P(y_1, \ldots, y_n)$, where $y_i = 0$ for $i \in I$ and $y_i = x_i$ otherwise. Define*

$$Q(x_1, \ldots, x_n) = \sum_{I \subseteq J} (-1)^{|I|} P_{-I}(x_1, \ldots, x_n). \tag{2}$$

Then, for any monomial M divisible by $\Pi_{i \in J} x_i$, we have $\mathrm{coef}_Q(M) = \mathrm{coef}_P(M)$, and for every other monomial M we have $\mathrm{coef}_Q(M) = 0$.

Proof. Consider a monomial M with non-zero coefficient in P. Observe first that for every $I \subseteq [n]$, we have $\mathrm{coef}_{P_{-I}}(M) = \mathrm{coef}_P(M)$ if no variable x_i with $i \in I$ occurs in M, and $\mathrm{coef}_{P_{-I}}(M) = 0$, otherwise. Now, if $\Pi_{i \in J} x_i$ divides M, then out of the $2^{|J|}$ evaluations for $I \subseteq J$, the monomial M occurs in exactly one (namely, $I = \emptyset$). Thus, $\mathrm{coef}_Q(M) = \mathrm{coef}_P(M)$.

If $\Pi_{i \in J} x_i$ does not divide M, note that $J' = \{i \in J : x_i \text{ does not divide } M\}$, is nonempty and observe that $\mathrm{coef}_{P_{-I}}(M) = \mathrm{coef}_P(M)$ for every $I \subseteq J'$. Thus, sum (2) for M only is

$$\sum_{I \subseteq J} (-1)^{|I|} M = \sum_{I \subseteq J'} (-1)^{|I|} M = M \sum_{I \subseteq J'} (-1)^{|I|} = M(1-1)^{|J'|} = 0.$$

Applying the above results individually to every monomial in P accounts for all occurrences of monomials in the sum defining Q; the result follows. □

4 k-Internal Out-Branching

In a digraph D, a *matching* is a collection of arcs without common vertices. The following lemma establishes useful connections between matchings and out-trees/out-branchings, which is the key to our computation of a coloring guide.

Lemma 7. *The following statements hold.*

(i) Let T be an out-tree with $k \geqslant 0$ internal vertices. Then T has a matching of size at least $k/2$.

(ii) Let $D = (V, A)$ be a digraph containing an out-branching, and M a matching in D. Then, in polynomial time, we can find an out-branching of D for which no arc of M has both end-vertices as leaves.

Proof. (1) We prove it by induction on $k \geqslant 0$. The claim obviously holds for $k = 0$, so assume that $k \geqslant 1$. The *height* of a vertex v in T is the length of a longest path from v to a leaf of T reachable from v. Let k_i be the number of vertices of T of height i. Observe that $k_1 \geqslant k_2$ and that T has a matching M_1 with k_1 edges whose vertices are some leaves and all vertices of height 1. Let T' be an out-tree obtained from T by deleting all leaves and vertices of height 1. Observe that T' has $k - k_1 - k_2$ internal vertices and thus by induction hypothesis T' has a matching M_2 of size at least $(k - k_1 - k_2)/2 \geqslant k/2 - k_1$. Thus, the matching $M_1 \cup M_2$ of T is of size at least $k/2$.

(2) Let B be an out-branching of D and suppose that both end-vertices of some arc xy of M are leaves in B. Then add xy to B and delete zy from B, where z is the in-neighbor of y in B. In the resulting out-branching B', x is an internal vertex. Notice that zy does not belong to M. Hence, B' contains one more arc of M than B. Starting with an arbitrary out-branching and repeating the above exchange operation at most $|M|$ times, we will get an out-branching in which no arc of M has both end-vertices as leaves. This process can be completed in polynomial time. □

We now prove the main theorem of this section.

Theorem 8. *There exists a deterministic polynomial-space $\mathcal{O}^*(3.86^k)$-time algorithm for k-*Internal Out-Branching.

Proof. Let D be a digraph, $V(D) = [n]$, and M a maximum matching in D of size t. By Lemma 7, we may assume that $k/2 \leqslant t \leqslant k$ as otherwise we can solve k-IOB on D in polynomial time: For $t < k/2$, no out-tree with k internal vertices exists and for $t > k$, we can construct a solution in polynomial time. Now we will follow our approach.

(1): We associate one variable x_v with every vertex $v \in [n]$. Replace every variable x_{ij} in (1) by x_i and observe that now by Theorem 1 we have that if the polynomial $\det(K_{\bar{r}}(D))$ over variables $x_1, \ldots, x_n$ contains a monomial with at least k different variables, then there exists an out-branching rooted at r with at least k internal vertices. Note that for a k-coloring $\mathcal{C}$ of the variables we can evaluate the polynomial $\det(K_{\bar{r}}(D))/\mathcal{C}$ over variables $y_1, \ldots, y_k$ in polynomial time. We guess the root vertex r and fix it for the following steps.

(2): For every $c \in [k] \cup \{0\}$, we consider all sets M' of c edges of M in which both vertices are supposed to be internal vertices of some k-internal out-branching

(the edges of $M \setminus M'$ contain at least one internal vertex in some k-internal out-branchings by Lemma 7 (ii)).

For the current choice of M', our coloring guide $\mathcal{S}$ looks as follows.[6] For every arc $uv \in M'$, we add the sets $\{u\}$ and $\{v\}$ to $\mathcal{S}$. For every arc $xy \in M \setminus M'$, we add the set $\{x, y\}$. Finally, $S_\perp := V(D) \setminus V(M)$ contains all vertices outside of the matching.

(3): With $t+c$ internal vertices of the out-tree in $V(M)$ there are $k-t-c$ vertices left to be located in $V(D)\setminus V(M)$. Using an $(n, k-t-c)$-perfect hash family, we color the vertices of $S_\perp$ and obtain a k-coloring $\mathcal{C} := C_1 \uplus C_2 \uplus \ldots \uplus C_{t+c} \uplus C_\perp$ of the variables X by combining the coloring from the hash family with the coloring guide $\mathcal{S}$.

(4): We apply Lemma 6 to the polynomial $\det(K_{\bar{r}}(D))/\mathcal{C}$ over $y_1, \ldots, y_k$ to search for a monomial that contains all k variables. If such a monomial exists, then $\det(K_{\bar{r}}(D))$ contains a monomial with k distinct variables and we conclude that D contains an out-branching rooted at r with k or more internal vertices. Otherwise, if we have not yet exhausted all colorings in the hash family, we return to Step (3), and if we have not yet exhausted all guesses for M', we return to Step (2). If neither of these conditions is true, we conclude that there is no k-internal out-branching rooted at r.

Let us analyse the exponential part $f(k)$ of the running time for the above steps. Step (2) to (4) take time at most

$$\sum_{c=0}^{k-t} \binom{t}{c} \tau(1)^{(k-t-c)(1+o(1))} 2^k = \sum_{c=0}^{k-t} \binom{t}{c} e^{(k-t-c)(1+o(1))} 2^k,$$

using the $(n, k-t-c)$-perfect hash family from Theorem 4 (with $\alpha = 1$). Consider the term $\binom{t}{c} e^{-(t+c)}$ and set $c = \beta t$, where $\beta \in (0, 1)$. Using the well-known bound $\binom{t}{\beta t} \leqslant 2^{tH(\beta)}$, where $H(\beta) = -\log(\beta^\beta (1-\beta)^{1-\beta})$, we arrive at

$$\binom{t}{\beta t} e^{-t(1+\beta)} \leqslant (\beta^\beta (1-\beta)^{1-\beta} e^{1+\beta})^{-t} \leqslant \left(\frac{e+1}{e^2}\right)^t.$$

The second inequality above follows from the fact that $\min_{0<\beta<1} g(\beta) = \frac{e}{e+1}$, where $g(\beta) = \beta^\beta (1-\beta)^{1-\beta} e^\beta$, which can be verified by differentiating $g(\beta)$. It is easy to verify that $\binom{t}{\beta t} e^{-t(1+\beta)} \leqslant (\frac{e+1}{e^2})^t$ for $\beta = 0$ and $\beta = 1$ as well. Therefore,

$$\sum_{c=0}^{k-t} \binom{t}{c} e^{(k-t-c)(1+o(1))} 2^k \leqslant k \left(\frac{e+1}{e^2}\right)^t (2e)^{k(1+o(1))}.$$

Observe that $\frac{e+1}{e^2} < 1$ and thus $(\frac{e+1}{e^2})^t$ decreases monotonically as t grows. Therefore, $(\frac{e+1}{e^2})^t \leqslant (\frac{e+1}{e^2})^{k/2}$ as $t \geqslant k/2$. Thus, $f(k)$ is bounded from above by $k(4(e+1))^{k(1+o(1))/2} < k3.857^k$ for k large enough. Since all of the above steps require polynomial space, the algorithms requires polynomial space as well. □

[6] Formally, we somewhat abuse terminology and notation for coloring guide here, but since the notions are very close, we think it will not lead to a confusion.

If (D, k) is a positive instance of k-IOB, then we can find a k-internal out-branching of D also in polynomial space and time $O^*(3.857^k)$ by the usual self-reducibility argument: Consider every arc a of D at a time and remove it from D if $(D-a, k)$ is a positive instance of k-IOB until no arc can be removed. The non-empty remaining graph spans an out-branching with k internal leaves.

Let us now see how our approach yields a faster algorithm if we use exponential space.

Theorem 9. *There exists an exponential-space $\mathcal{O}^*(3.41^k)$-time algorithm for k-Internal Out-Branching.*

The exponential-space strategy will use the exponential-space version of the color-coding and sieving result, but we will in addition use *truncated* fast subset convolution to further speed up the algorithm, by rolling up the $\binom{t}{c}$ guesses in Step (2) and the applications of Lemma 6 into a single exponential-space computation. This was presented in Björklund *et al.* [37] (called trimmed Moebius inversion).

Proof. We will present the proof as close to the structure of the proof of Theorem 8 as possible. However, since the truncated fast subset convolution does not strictly speaking follow the coloring guide method, some discrepancies between the proofs are unavoidadble. Let D be a digraph with vertex set $V = [n]$.
(1): As before, let M a maximum matching in D of size t with $k/2 \leqslant t \leqslant k$. Also guess a root vertex r and keep it fixed through the following steps. Number the edges of M as $M = \{e_1, \ldots, e_t\}$. With every edge $e_i \in M$ we associate three variables x_i, x'_i, x''_i, and with every vertex $v \in V \setminus V(M)$ we associate a variable x_v. Further, for every edge $e_i = uv \in M$ define $x(u) = x_i x'_i$ and $x(v) = x_i x''_i$, and for every other vertex $v \in V \setminus V(M)$ define $x(v) = x_v$. Replace every variable x_{ij} in (1) by $x(i)$. We will sieve the polynomial $\det(K_{\bar{r}}(D))$ according to two modes for every edge $e_i \in M$: Either only one endpoint of e_i is internal in the out-branching, in which case we sieve for the variable x_i, or both are, in which case we sieve for both x'_i and x''_i. Using truncated fast subset convolution, we will sieve for these options in parallel.
(2)+(3): Guess the number $c \in [k-t] \cup \{0\}$ of edges of M for which both endpoints are internal in the solution, but do not guess an explicit subset M'. Let $k' := k - t - c$ be the number of internal vertices of the sought solution that lie outside $V(M)$, and recall the constant $\alpha^\star$ from Proposition 5. Further fix a coloring f of the vertices of $V \setminus V(M)$ into $\alpha^\star \cdot k'$ colors, and assume that the k' internal vertices not present in M all receive distinct colors by f. Repeat the following steps for every choice of f from an $(n, k', \alpha^\star k')$-splitter.
(4): We now describe the improved parallel sieving strategy. Define $X = \{x_i, x'_i, x''_i \mid i \in [t]\}$, and let $P(X, Y)$ be the result of replacing every variable x_j, $j \in V \setminus V(M)$, in $\det(K_{\bar{r}}(D))$ by $y_{f(j)}$. Consider one particular choice $M' \subseteq M$, $|M'| = c$, of edges where both endpoints are assumed to be internal vertices in the solution. We would then be seeking a monomial of $P(X, Y)$ which contains both variables x'_i, x''_i for every edge $e_i \in M'$, the variable x_i for

every edge $e_i \in (M \setminus M')$, and additionally variables of k' further colors $y_\ell \in Y$. Combining this across all choices of M' with $|M'| = c$, we will thus want to run the sieving algorithm for the family of sets

$$\mathcal{F}_c := \left\{ \{x'_i\}_{i \in I} \cup \{x''_i\}_{i \in I} \cup \{x_i\}_{i \in [t] \setminus I} \,\middle|\, I \in \binom{[t]}{c} \right\} \times \binom{[\alpha^\star k']}{k'}.$$

For each $F \in \mathcal{F}_c$, let $Q_F(X, Y)$ be polynomial defined in Lemma 6. Up to the choice of c and f, the instance is positive if and only if there is some F such that $Q_F(X, Y) \neq 0$, and we can use $X = \mathbf{1}$, $Y = \mathbf{1}$ for the evaluation, where $\mathbf{1}$ denotes a vector with all components equal 1.

Using the truncated fast subset convolution, we can compute all evaluations $Q_F(\mathbf{1}, \mathbf{1})$ as above in time proportional to the number of subsets of sets $F \in \mathcal{F}_c$, up to a polynomial factor [37]. Concretely, let $\mathcal{I}_c = \{I \subseteq F \mid F \in \mathcal{F}_c\}$. Then the truncated fast subset convolution runs in time $\mathcal{O}^*(|\mathcal{I}_c|)$, and we repeat this procedure for every choice c and for every member f of the $(n, p, \alpha^\star k')$-splitter,

Running time. To simplify the analysis of the running time, we write $\mathcal{J}_c = \{I \cap X \mid I \in \mathcal{I}_c\}$. Then $|\mathcal{I}_c| \leqslant |\mathcal{J}_c| \cdot 2^{\alpha^\star k'}$, and the product of the size of the splitter and $2^{\alpha^\star k'}$ can be bounded as $\mathcal{O}^*(4.312^{k'})$ as in Proposition 5. It remains to bound $|\mathcal{J}_c|$. We further split $\mathcal{J}_c$ into $c+1$ *levels*, where a set $I \in \mathcal{J}_c$ belongs to level i, $0 \leqslant i \leqslant c$, if there are exactly i edges $e_j \in M$ such that $I \cap \{x'_j, x''_j\} \neq \emptyset$. Thus, the number of sets at level i of $\mathcal{J}_c$ equals

$$\binom{t}{i} 3^i \sum_{j=0}^{t-c} \binom{t-i}{j} \leqslant \binom{t}{i} 3^i 2^{t-i},$$

where the factor of 3 comes from the three options $\{x'_j\}$, $\{x''_j\}$, $\{x'_j, x''_j\}$ and the expression in the summation corresponds to whether $x_j \in I$ for the remaining edges. Note that the upper bound is essentially tight assuming $t - c > (t-i)/2$. Therefore, if we split out the total work per level, we get

$$\max_{i \in [c] \cup \{0\}} \binom{t}{i} 3^i 2^{t-i} 4.312^{k-t-c} = 2^t 4.312^{k-t-c} \max_{i \in [c] \cup \{0\}} \binom{t}{i} 1.5^i. \tag{3}$$

To obtain an upper bound for (3), we will use the following observation.

Claim. *Let b be a real, t an integer, and x an integral variable. The function $g(x) = \binom{t}{x} b^x$ monotonically increases if and only if $x \leqslant \frac{b(t+1)}{b+1}$.*

Proof. It suffices to observe that $g(x)/g(x-1) \geqslant 1$ if and only if $x \leqslant \frac{b(t+1)}{b+1}$. □

Furthermore, since $g(x+1), g(x-1) \leqslant g(x) \cdot bt$ for the function $g(x)$ defined in the claim, up to lower-order terms we need not be concerned with the precise value of x. We now consider two cases. First, assume that $c \geqslant 0.6t$. Then by the claim above, we have

$$\binom{t}{i} 1.5^i \leqslant t^{O(1)} \binom{t}{\lceil 0.6t \rceil} 1.5^{0.6t} \leqslant t^{O(1)} 2^{H(0.6)t} 1.5^{0.6t},$$

where we used the well-known bound $\binom{t}{\alpha t} \leqslant 2^{H(\alpha)t}$ for the binary entropy function $H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$. Inserting the exponential part into (3), we get a bound of

$$2^t 4.312^{k-t-0.6t} 2^{H(0.6)t} 1.5^{0.6t} < 4.312^k 0.482484^t < 3^k$$

as $t \geqslant k/2$. Thus, we next consider $c < 0.6t$. Then by the claim, we have

$$\max_{i \in [c] \cup \{0\}} \binom{t}{i} 1.5^i = \binom{t}{c} 1.5^c.$$

Inserting it into (3) and rearranging, we get a bound of

$$2^t 4.312^{k-t} \binom{t}{c} (1.5/4.312)^c.$$

Let $a = 1.5/4.312$ and $\alpha = a/(a+1)$. By the claim, the latter half of the product above is maximized around $c = \alpha t$ giving the following upper bound to the product:

$$t^{O(1)} 2^t 4.312^{k-t} 2^{H(\alpha)t} a^{\alpha t} < t^{O(1)} 4.312^k 0.625172^t = \mathcal{O}^*(3.41^k),$$

as $t \geqslant k/2$. Finally, since $\alpha < 0.6$, an integer around αt is a valid value of c. □

We remark that while the running time bound is essentially tight for the algorithm we describe, we cannot exclude that there is a more efficient way of implementing the sieving, e.g., using less than three variables per edge in M.

5 k-Colorful Out-Branching

First, let us argue that k-Colorful Out-Branching is NP-hard. To this end, we have a simple reduction from Hamilton Path: in a digraph D define the color of every arc outgoing from $v \in V(D)$ to be c_v. Clearly, D has a Hamilton Path if and only if it has a k-colorful out-branching with $k = |V(D)| - 1$. Thus, on any graph class where Hamilton Path is NP-hard, k-Colorful Out-Branching is NP-hard as well. In fact, we can have the same simple reduction from k-IOB to k-Colorful Out-Branching, which shows that the latter generalizes the former.

We follow the structure of the proof of Theorem 8 to show the following result for the colorful variant.

Theorem 10. *There exists a deterministic polynomial-space $\mathcal{O}^*((2e)^{k+o(k)})$-time (and exponential-space $\mathcal{O}^*(4.312^k)$-time) algorithm for k-Colorful Out-Branching.*

Proof. Let $D = (V, A)$ be a directed multigraph, and let $c(a)$ be the color of a for every arc $a \in A$. We guess the root vertex r and fix it for the following steps. Let C be the full set of colors used, and without loss of generality let $C = [t]$, $t \geqslant k$,

and $A = \{a_1, \ldots, a_m\}$. Create a set $Z = \{z_1, \ldots, z_m\}$ of corresponding variables, and define the polynomial $P'(z_1, \ldots, z_m) = \det(K_{\bar{r}}(D))$. By Theorem 1, our problem has now been reduced to determining whether there is a monomial M in P' such that the corresponding out-branching contains arcs of at least k different colors. We solve this problem according to our approach as follows. Here, we first describe the polynomial-space algorithm, and later explain how it can be sped-up at the cost of exponential space.

(1): Create a set $X = \{x_1, \ldots, x_t\}$ of variables corresponding to colors. We define the polynomial P by $P(x_1, \ldots, x_t) = P'(x_{col(1)}, \ldots, x_{col(m)})$, where $col(i) = c(e_i)$ for all $i \in [m]$. Then, as we argued above, D has a k-colorful out-branching rooted at r if and only if P has a witnessing monomial. Moreover, note that for a k-coloring $\mathcal{C}$ of the variables, we can evaluate the polynomial $P/\mathcal{C}$ over variables $y_1, \ldots, y_k$ in polynomial time, as this can be done by evaluating P'.

(2): In this application, our coloring guide is empty ($S_\perp = V(D)$). Thus, it can be ignored in the following steps.

(3): Using a (t, k)-perfect hash family, we recolor the colors of C and obtain a k-coloring $\mathcal{C} := C_1 \uplus C_2 \uplus \ldots \uplus C_k$ of the variables X.

(4): We apply Lemma 6 to the polynomial $P/\mathcal{C}$ over $y_1, \ldots, y_k$ to search for a monomial that contains all k variables. If such a monomial exists, then P contains a monomial with k or more distinct variables and we conclude that D contains an out-branching rooted at r with k or more colors. Otherwise, if we have not yet exhausted all colorings in the hash family, we return to Step (3), and otherwise we conclude that the there is no k-colorful out-branching rooted at r.

Let us now analyse the exponential part $f(k)$ of the running time for the above steps. By Theorem 4 there are $\mathcal{O}^*(e^{k+o(k)})$ colorings in our (t, k)-perfect hash family, and by Lemma 6 the sieving can be performed in time $\mathcal{O}^*(2^k)$ for every individual coloring. Hence, the total time is $\mathcal{O}^*((2e)^{k+o(k)})$ as claimed.

Finally, we consider the case where we may use exponential space. Then, we use a $(t, k, \alpha^\star k)$-splitter rather than a (t, k)-perfect hash family. Thus, by Proposition 5, the total running time is $\mathcal{O}^*(4.312^k)$, as claimed. □

We can improve the above result if the input graph is colored with *exactly* k colors: In Step (2) of the above proof we do not use an empty coloring guide, but instead a partition of A such that two arcs are in the same block of the partition if and only if they have the same color. Notice that in this case, in Step (3) we construct a $(t, 0)$-perfect hash family, which means that every color simply retains its original color. Thus, we derive the following corollary.

Corollary 11. *There exists a deterministic polynomial-space $\mathcal{O}^*(2^k)$-time algorithm for k-*Colorful Out-Branching *on directed multigraph with exactly k colors.*

6 k-Colorful Perfect Matching on Planar Graphs

Let us first show the NP-hardness of k-Colorful Perfect Matching.

Lemma 12. *k-Colorful Perfect Matching is NP-hard on planar graphs of pathwidth 2.*

Proof. We present a reduction from Rainbow Matching on paths, which is known to be NP-hard [13]. To this end, let (P, k) be an instance of Rainbow Matching on paths. Denote $P = v_1 - v_2 - \cdots - v_n$. Now, we construct an instance $(G, k+1)$ of k-Colorful Perfect Matching as follows. Let $c^\star$ be a new color, and define $P' = v'_1 - v'_2 - \cdots - v'_n$ as a copy of P where the color of each edge is $c^\star$. Then, G is defined as the graph obtained from P and P' by adding the edge $v_i - v'_i$ of color $c^\star$ for all $i \in [n]$. Clearly, G is a planar graph of pathwidth 2.

On the one hand, suppose that (P, k) has a k-colorful matching M of size k. Denote $I = \{i \in [n-1] : v_i v_{i+1} \in M\}$ and $I^+ = \{i+1 : i \in I\}$. Then, $M' = M \cup \{v'_i v'_{i+1} : i \in I\} \cup \{v_i v'_i : i \notin I \cup I^+\}$ is a $(k+1)$-colorful perfect matching in G. On the other hand, suppose that $(G, k+1)$ has a $(k+1)$-colorful perfect matching M. Since the color of all edges outside P is $c^\star$, the set of edges of M that belong to P is k-colorful. In particular, this set contains a k-colorful matching of size k. □

For our algorithm, we will need the following famous result which implies that planar perfect matchings can be counted in polynomial time. We will need the following more detailed version of the original result:

Theorem 13 ([38]). *Let $G = (V, E)$ be a planar graph, and let $X = \{x_e \mid e \in E\}$ be a collection of real-valued variables. There is a polynomial-time algorithm that produces a matrix A over X such that*

$$\mathrm{pf}(A) = \sum_{M \in \mathcal{M}} \prod_{e \in M} x_e,$$

where $\mathcal{M}$ ranges over all perfect matchings of G.

Using our approach, we can now easily derive the following theorem in a manner very similar to the one used to prove Theorem 10. For the sake of completeness, we present the required details.

Theorem 14. *k-Colorful Perfect Matching on planar graphs can be solved by a deterministic algorithm in time $\mathcal{O}^*((2e)^{k+o(k)})$ with polynomial space, and in time $\mathcal{O}^*(4.32^k)$ with exponential space.*

Proof. Let $G = (V, E)$ be a planar graph, and let $c(e)$ be the color of e for every edge $e \in E$. Let C be the full set of colors used, and without loss of generality let $C = [t]$, $t \geqslant k$, and $E = \{e_1, \ldots, e_m\}$. Create a set $Z = \{z_1, \ldots, z_m\}$ of corresponding variables, and let A be the matrix computed by Theorem 13. Define the polynomial $P'(z_1, \ldots, z_m) = \mathrm{pf}(A)$. Recall that P can be evaluated in polynomial time. Our problem has now been reduced to determining whether there is a monomial M in P' such that the corresponding perfect matching contains edges of at least k different colors. We solve this problem according to

our approach as follows. Here, we first describe the polynomial-space algorithm, and later explain how it can be sped-up at the cost of exponential space.

(1): Create a set $X = \{x_1, \ldots, x_t\}$ of variables corresponding to colors. We define the polynomial P by $P(x_1, \ldots, x_t) = P'(x_{col(1)}, \ldots, x_{col(m)})$, where $col(i) = c(e_i)$ for all $i \in [m]$. Then, as we argued above, the input instance is a yes-instance if and only if P has a witnessing monomial. Moreover, note that for a k-coloring $\mathcal{C}$ of the variables, we can evaluate the polynomial $P/\mathcal{C}$ over variables $y_1, \ldots, y_k$ in polynomial time, as this can be done by evaluating P'.

(2): In this application, our coloring guide is empty ($S_\perp = V(D)$). Thus, it can be ignored in the following steps.

(3): Using a (t,k)-perfect hash family, we recolor the colors of C and obtain a k-coloring $\mathcal{C} := C_1 \uplus C_2 \uplus \ldots \uplus C_k$ of the variables X.

(4): We apply Lemma 6 to the polynomial $P/\mathcal{C}$ over $y_1, \ldots, y_k$ to search for a monomial that contains all k variables. If such a monomial exists, then P contains a monomial with k or more distinct variables and we conclude that G contains a perfect matching with k or more colors. Otherwise, if we have not yet exhausted all colorings in the hash family, we return to Step (3), and otherwise we conclude that the input instance is a no-instance.

Let us now analyse the exponential part $f(k)$ of the running time for the above steps. By Theorem 4 there are $\mathcal{O}^*(e^{k+o(k)})$ colorings in our (t,k)-perfect hash family, and by Lemma 6 the sieving can be performed in time $\mathcal{O}^*(2^k)$ for every individual coloring. Hence, the total time is $\mathcal{O}^*((2e)^{k+o(k)})$ as claimed.

Finally, we consider the case where we may use exponential space. Then, we use a $(t,k,\alpha^\star k)$-splitter rather than a (t,k)-perfect hash family. Thus, by Proposition 5, the total running time is $\mathcal{O}^*(4.312^k)$ as claimed. □

We conclude by observing that there is little hope to apply our approach to Rainbow Matching. In particular, counting not necessarily perfect matchings in a planar graph is #P-hard, so there is no plug-in replacement for Theorem 13 for general matchings.

7 Enumerating Splitters with Polynomial Delay

In this section, we prove Theorem 4. We essentially follow the construction by Naor *et al.* [35] while taking care to keep the space consumption polynomial. In particular, the idea by Fomin *et al.* [28] (used in the context of construction representative sets) to frame the construction in a way that makes a repeated application possible, turns out to be a crucial component. To this end, we will need the following definition of an *indexed splitter* which treats splitter families as data structures that enumerate vectors instead of fixed collections of vectors.

Definition 15 (Indexed splitter). *An* indexed (n,k,t)-splitter *of size m is a data structure $\mathcal{S}$ that for $i \in [m]$ returns a vector $\bar{x}_i \in [t]^n$ such that $\{\bar{x}_i\}_{i\in[m]}$ is an (n,k,t)-splitter. The* query time $t_{\mathsf{qr}}(n,k,t)$ *and* query space $s_{\mathsf{qr}}(n,k,t)$ *are the resources needed by $\mathcal{S}$ to compute any such $\bar{x}_i$, where we exclude the*

space needed by $\bar{x}_i$ from s_{qr}. The initialization time $t_{in}(n,k,t)$ *and* initialization space $s_{in}(n,k,t)$ *are the resources needed to compute $\mathcal{S}$ given n, k and t.*

We will call the tuple $(m, t_{\mathsf{in}}, s_{\mathsf{in}}, t_{\mathsf{qr}}, s_{\mathsf{qr}})$ the *profile* of an indexed splitter. Note that every splitter of size $f(n,k,t)$ is an indexed splitter with query-time proportional to $\log f(n,k,t)$: we simply store the whole splitter in memory according to some (arbitrary) order.

One of the main ingredients will be the following two splitters constructed by Naor, Schulman, and Srinivasan:

Proposition 16 (*cf.* [35]). *There exists an (n,k,k^2)-splitter $\mathcal{A}(n,k)$ of size $k^{O(1)} \log n$ that can be efficiently constructed using $k^{O(1)} n \log n$ space.*

Proposition 17 (*cf.* [35]). *For all $k, t \leqslant n$ there exists an indexed (n,k,t)-splitter $\mathcal{B}(n,k,t)$ of size $\binom{n}{t-1}$ with $t_{in}(n,k,t), s_{in}(n,k,t) = O(t \log n)$, $t_{qr}(n,k,t) = O(nt \log n)$, and $s_{qr}(n,k,t) = O(t \log n)$.*

Proof. The underlying splitter corresponds to the set of all ordered tuples $\bar{i} \in [n]^{t-1}$. Given such a tuple $\bar{i}$, we assign all elements in the range $\left[0, \bar{i}[0]-1\right]$ the value 0, all elements in the range $\left[\bar{i}[0], \bar{i}[1]-1\right]$ the values 1, and so on. Clearly, every index set of size k is partitioned almost equally by at least one of these vectors. This construction is easily transformed into an indexed splitter with the claimed profile by choosing an appropriate indexing of the ordered tuples from $[n]^{t-1}$. □

One further core component needed here is a k-wise independent *sampling space*, whose properties we can use to generate a small (n,k)-perfect hash family. Let us recall the very elegant construction by Joffe [39] for such a space:

Definition 18. *A probability space Ω with n random variables $\{X_i\}_{i \in [n]}$ is* k-wise independent *if for every index set $I \subseteq [n]$ of size k the random variables $\{X_i\}_{i \in I}$ are mutually independent.*

Theorem 19 (Joffe [39]). *Let p be prime and $k < p$. If $\{X_i\}_{i \in [k]}$ are random variables uniformly distributed on $\{0, \ldots, p-1\}$, then the variables $\{Y_i\}_{i \in [p]}$ defined via*

$$Y_i := (X_1 + iX_2 + \ldots + i^{k-1} X_k) \pmod{p}$$

are uniformly distributed over $\{0, \ldots, p-1\}$ and k-independent.

Note that the variables $\{Y_i\}_{i \in [p]}$ range over $\{0, \ldots, p-1\}$ and not over $[k]$. However, because they are uniformly distributed, we can easily 'downsample' them without much loss.

Lemma 20. *Let p be prime and $k \leqslant t < p$. There exists a k-wise independent sampling space $H^{\star}_{p,k,t}$ for random variables $\{\hat{Y}_i\}_{i \in [p]}$ over $[t]$ where the $\hat{Y}_i$ are identically distributed and almost uniform in the sense that $\left|\mathbb{P}[\hat{Y}_i = r] - \frac{1}{t}\right| \leqslant \frac{1}{p}$ for $r \in [t]$. Moreover, the members of $H^{\star}_{p,k,t}$ can be listed sequentially in time $O(p^t k^2 \log p)$ by an algorithm using $O(t \log p)$ bits.*

Proof. Let the variables $\{X_i\}_{i\in[k]}$ and $\{Y_j\}_{j\in[p]}$ be defined as above. We further define $\hat{Y}_j := Y_j \pmod t$. Since the variables Y_j are k-wise independent, so are the variables $\hat{Y}_j$. The distribution of each $\hat{Y}_j$ follows immediately from the fact that

$$\frac{\lfloor p/t \rfloor}{p} \leqslant \mathbb{P}\big[\hat{Y}_j = r\big] \leqslant \frac{\lceil p/t \rceil}{p}$$

for each $r \in [t]$. To list all members of $H^\star_{p,k,t}$, we enumerate all p^t possible assignments of $\{X_i\}_{i\in[k]}$ and compute the values for $\{\hat{Y}_i\}_{j\in[p]}$. We need $O(t \log p)$ bits to store a counter for the first enumeration as well as $O(k \log p)$ bits to enumerate the polynomials defining the $\hat{Y}_i$. Since each polynomial can be evaluated using $O(k^2)$ arithmetic operations, the claimed running time follows. □

The following construction of a basic splitter follows the one by Noar *et al.* closely, however, our rephrasing and analysis is more suitable for the final construction of the indexed splitter.

Lemma 21. *For every $k < \sqrt{n}$ and $\alpha \geqslant 1$ there is an $(n, k, \alpha k)$-splitter $\mathcal{C}_\alpha(n,k)$ of size $O\big(\tau(\alpha)^k\, k \log n\big)$ that can be constructed in $|\mathcal{C}_\alpha(n,k)|k^2\binom{n}{k}(2n)^{\alpha k}\log^{O(1)} n$ time using $O(|\mathcal{C}_\alpha(n,k)|\,\log n)$ space.*

Proof. Let $n \leqslant p \leqslant 2n$ be the smallest prime at least as large as n. We can identify p using the AKS test or any of its recent improvements in $n\log^{O(1)} p$ time. For ease of presentation, we will assume that αk is an integer. Let $H^\star = H^\star_{p,k,\alpha k}$ be the k-wise independent probability space defined in Lemma 20. For a vector $\hat{y} \in H^\star$, let $C(\hat{y})$ contain all index sets $I \in \binom{[n]}{k}$ for which $\hat{y}[I]$ contains k distinct values. Let us extend this notation to sets $H' \subseteq H^\star$ via $C(H') := \bigcup_{\hat{y}\in H'} C(\hat{y})$. Our goal is to find a set $H' \subseteq H^\star$ of small size such that $C(H') = \binom{[n]}{k}$.

Claim. *Let $\mathcal{F} \subseteq \binom{[n]}{k}$ and $n \geqslant 1.256\alpha k$. There exists a vector $\hat{y} \in H^\star$ such that $C(\hat{y})$ contains a fraction of at least $e^{-2\alpha k^2/n}\binom{\alpha k}{k}\frac{k!}{(\alpha k)^k}$ sets from $\mathcal{F}$.*

Proof. Fix any index set $I \in \mathcal{F}$. Since $|I| \leqslant k$ the random variables $\{\hat{Y}_i\}_{i\in I}$ are independent, the probability that a vector $\hat{y} \in H^\star$ chosen uniformly at random hits I is

$$\mathbb{P}[I \in C(\hat{y})] \geqslant \binom{\alpha k}{|I|} k! \Big(\frac{1}{\alpha k} - \frac{1}{p}\Big)^{|I|} \geqslant (1-\alpha k/p)^k \binom{\alpha k}{k}\frac{k!}{(\alpha k)^k}.$$

Accordingly, the expected number of sets in $\mathcal{F}$ hit by at least one member of $H^\star$ is at least $(1-\alpha k/p)^k\binom{\alpha k}{k}\frac{k!}{(\alpha k)^k}\cdot|\mathcal{F}|$ and we conclude that at least one vector of $H^\star$ must hit that many members of $\mathcal{F}$. Since $(1-c/x)^{x/c} \geqslant e^{-2}$ for $x \geqslant 1.256c$ we can bound the first term of this expression by

$$\big(1-\frac{\alpha k}{p}\big)^k \geqslant e^{-2\alpha k^2/p} \geqslant e^{-2\alpha k^2/n}$$

and arrive at the claimed bound. □

The above claim now gives us a method of constructing H' greedily: initialize $H' = \emptyset$ and list the members $\hat{y}_1, \hat{y}_2, \ldots$ of $H^\star$. For each vector $\hat{y}_i$, compute $|C(\hat{y}_i) \setminus C(H')|$, that is, the number of index sets hit by $\hat{y}_i$ that are not yet hit by any member of H'. If

$$|C(\hat{y}_i) \setminus C(H')| \geqslant e^{-2\alpha k^2/n} \binom{\alpha k}{k} \frac{k!}{(\alpha k)^k} \Big(\binom{n}{k} - |C(H')| \Big),$$

then add $\hat{y}_i$ to H', otherwise drop $\hat{y}_i$. In either case, proceed with $\hat{y}_{i+1}$ until $H^\star$ has been completely traversed. Note that in every step, we can compute the numbers $|C(\hat{y}_i) \setminus C(H')|$ and $|C(H')|$ in time $O\big(\binom{n}{k}|H'|k\big)$ by simply enumerating all index sets and testing the vectors $H' \cup \{\hat{y}_i\}$.

Finally, let us bound the number of steps the above algorithm takes and therefore the size of H'. Since every member added to H' covers at least a fraction of $e^{-2\alpha k^2/n}\binom{\alpha k}{k}k!/(\alpha k)^k$ uncovered index sets, the number of steps t taken by the algorithm can be bounded by solving

$$|\mathcal{C}|\Big(1 - e^{-2\alpha k^2/n} \binom{\alpha k}{k} \frac{k!}{(\alpha k))^k}\Big)^t = \binom{n}{k}\Big(1 - e^{-2\alpha k^2/n} \binom{\alpha k}{k} \frac{k!}{(\alpha k))^k}\Big)^t < 1$$

for the variable t.

Claim. *For $t \geqslant e^{2\alpha k^2/n} \frac{(\alpha k)^k}{\binom{\alpha k}{k} k!} k \ln 2n$ the above expression is smaller than one.*

Proof. Let us substitute $x = e^{2\alpha k^2/n} \frac{(\alpha k)^k}{\binom{\alpha k}{k} k!}$. In particular, we assume $t \geqslant xk \ln 2n$. Then the above expression becomes

$$\binom{n}{k}\Big(1 - \frac{1}{x}\Big)^t \leqslant \binom{n}{k}\Big(1 - \frac{1}{x}\Big)^{xk \ln 2n} \leqslant \binom{n}{k} e^{-k \ln 2n} \leqslant \Big(\frac{e}{2k}\Big)^k$$

where we used that $(1 - \frac{1}{x})^x \leqslant e^{-1}$. Clearly, the right hand side is smaller than one for $k \geqslant 2 > e/2$ and the claim follows. □

A more manageable expression for the bound on t (ignoring the small factor $e^{2\alpha k^2/n} k \ln 2n$) can be derived using Stirling's approximation:

$$\begin{aligned} \frac{(\alpha k)^k}{\binom{\alpha k}{k} k!} &= (\alpha k)^k \frac{\big((\alpha - 1)k\big)!}{(\alpha k)!} \sim (\alpha k)^k \sqrt{1 - 1/\alpha} \Big(\frac{(\alpha - 1)^{\alpha - 1}}{\alpha^\alpha} \frac{e}{k} \Big)^k \\ &= \sqrt{1 - 1/\alpha} \Big((1 - 1/\alpha)^{\alpha - 1} e \Big)^k = \Theta(\tau(\alpha)^k). \end{aligned}$$

We conclude that $|H'| = O(\tau(\alpha)^k k \log n)$. The bounds on time and space follow immediately from $\mathcal{C}(n, k) = H'$. □

Definition 22 (Rounded splitter). *Let $\mathcal{S}$ be an indexed splitter. For rational numbers $\tilde{n}, \tilde{k}, \tilde{t}$ we denote by $[\mathcal{S}(\tilde{n}, \tilde{k}, \tilde{t})]$ the collection of indexed splitters $\{\mathcal{S}(n, k, t) \mid n \in \{\lfloor \tilde{n} \rfloor, \lceil \tilde{n} \rceil\}, k \in \{\lfloor \tilde{k} \rfloor, \lceil \tilde{k} \rceil\}, t \in \{\lfloor \tilde{t} \rfloor, \lceil \tilde{t} \rceil\}, \}$.*

We will treat the collection $[\mathcal{S}(\tilde{n}, \tilde{k}, \tilde{t})]$ like an indexed splitter with the understanding that a query to $[\mathcal{S}]$ carries, besides the index i, also the appropriate values for n, k and t. Since the additional overhead of constructing and querying $\mathcal{S}$ differs only by a constant factor, we will not further analyse this overhead in the following.

The following lemma closely follows the construction presented in Theorem 3 in [35], but adapted to construct a hash family with a flexible amount of colors. We will in the following make use of the Iverson bracket notation $[\![\phi]\!]$ which evaluate to 1 if ϕ is true and 0 otherwise. Also recall that we use the symbol $\bullet$ for variables whose value is arbitrary or unimportant in the current context.

Lemma 23. *For $\alpha \geqslant 1$, let $\mathcal{S}(n,k,t)$ be an indexed (n,k,t)-splitter with profile $(f, t_{in}, s_{in}, t_{qr}, s_{qr})$. Then for every $\ell \leqslant k$ we can construct an indexed (n,k,t)-splitter $\hat{\mathcal{S}}(n,k,t)$ with profile $(\hat{f}, \hat{t}_{in}, \hat{s}_{in}, \hat{t}_{qr}, \hat{s}_{qr})$ where*

$$\begin{aligned}
\hat{f}(n,k,t) &= k^{O(1)} \binom{k^2}{\ell} f(k^2, \lceil k/\ell \rceil, \lceil t/\ell \rceil)^\ell \log n, \\
\hat{t}_{in}(n,k,t) &= k^{O(1)} n \log n + O(t_{in}(k^2, k/\ell, t/\ell)), \\
\hat{s}_{in}(n,k,t) &= k^{O(1)} n \log n + O(s_{in}(k^2, k/\ell, t/\ell)), \\
\hat{t}_{qr}(n,k,t) &= O(\ell\, t_{qr}(k^2, \lceil k/\ell \rceil, \lceil t/\ell \rceil) + n), \\
\text{and} \quad \hat{s}_{qr}(n,k,t) &= O(\ell\, s_{qr}(k^2, \lceil k/\ell \rceil, \lceil t/\ell \rceil) + \log n).
\end{aligned}$$

Proof. In order to initialize $\hat{\mathcal{S}}$, we construct the family $\mathcal{A} := \mathcal{A}(n,k)$ from Proposition 16, the family $\mathcal{B} := \mathcal{B}(k^2, k, \ell)$ as well as the indexed splitter $[\mathcal{S}] := [\mathcal{S}(k^2, k/\ell, t/\ell)]$ in time

$$\hat{t}_{\mathsf{in}}(n,k,t) = k^{O(1)} n \log n + O(t_{\mathsf{in}}(k^2, k/\ell, t/\ell))$$

using space

$$\hat{s}_{\mathsf{in}}(n,k,t) = k^{O(1)} n \log n + O(s_{\mathsf{in}}(k^2, k/\ell, t/\ell)),$$

as claimed.

In order to answer a query i for $\hat{S}$, we decompose the query into $i = (i_a, i_b, i_1, \ldots, i_\ell)$ according to some indexing scheme (e.g. choosing appropriate substrings of the bitwise representation of i). We then choose $\bar{a} \in \mathcal{A}$ according to i_a, a partition $B^1 \cup \ldots \cup B^\ell$ of $[k^2]$ into ℓ blocks according to i_b from $\mathcal{B}(k^2, k, \ell)$ and vectors $\bar{s}_{i_1}, \ldots, \bar{s}_{i_\ell} \in [\mathcal{S}]$. We assume that the indexing scheme is such that $\bar{s}_{i_1}, \ldots, \bar{s}_{i_j}$ are taken from $(k^2, \lfloor k/\ell \rfloor, \bullet)$-splitters and $\bar{s}_{i_{j+1}}, \ldots, s_{i_\ell}$ from from $(k^2, \lceil k/\ell \rceil, \bullet)$-splitters where j is chosen such that

$$\lfloor k/\ell \rfloor j + \lceil k/\ell \rceil (\ell - j) = k,$$

and further that $\bar{s}_{i_1}, \ldots, \bar{s}_{i_h}$ are taken from $(k^2, \bullet, \lfloor t/\ell \rfloor)$-splitters and $\bar{s}_{i_{h+1}}, \ldots, \bar{s}_{i_\ell}$ from $(k^2, \bullet, \lceil t/\ell \rceil)$-splitters where h is similarly chosen such that

$$\lfloor t/\ell \rfloor h + \lceil t/\ell \rceil (\ell - h) = t.$$

The result $\bar{y}$ of the query is now constructed as follows. For $c \in [n]$, let B^p be the block of the chosen partition $B_1 \cup \ldots \cup B^\ell$ where $\bar{a}[c] \in B^p$. Then $\bar{y}[c] = \bar{s}_{i_p}[\bar{a}[c]] + \mathsf{offset}(p)$, where

$$\mathsf{offset}(p) = \lfloor t/\ell \rfloor p + [\![h > p]\!](h - p)$$

shifts the colors taken from the vectors $\bar{s}_\bullet$ in order to combine them without collisions.

Let us at this point convince ourselves that the vectors $\bar{y}_i$ constructed this way indeed form a (n, k, t)-splitter. Clearly, $\bar{y}_i \in [t]^n$ so it is left to consider the splitting property. To that end, fix an arbitrary index set $I \in \binom{[n]}{k}$. First, there exists $a \in \mathcal{A}$ such that $\bar{a}[I]$ receives indeed k distinct values $C \in \binom{[k^2]}{k}$. For this subset C, there exists a vector from $\mathcal{B}(k^2, k, \ell)$ that splits C evenly into parts of size $\lfloor k/\ell \rfloor$ and $\lceil k/\ell \rceil$. For each such part C_i, there now exists a vector $\bar{s}_i \in [\mathcal{S}]$ such that $\bar{s}_i[C_i]$ contains $|C_i|$ distinct values. Since the values contained in the $\bar{s}_\bullet$ are offset to avoid collisions, we conclude that the specific choices of $\bar{a}$, $\mathcal{B}$, and $\bar{s}_1 \ldots \bar{s}_\ell$ result in a vector $\bar{y}$ such that $\bar{y}[I]$ indeed contains k distinct values.

In total, the time taken to answer a query is

$$\hat{t}_{\mathsf{qr}}(n, k, t) = O(\ell\, t_{\mathsf{qr}}(k^2, \lceil k/\ell \rceil, \lceil t/\ell \rceil)).$$

and it uses space

$$\hat{s}_{\mathsf{qr}}(n, k, t) = O(\ell\, s_{\mathsf{qr}}(k^2, \lceil k/\ell \rceil, \lceil t/\ell \rceil) + \log n).$$

The size of $\hat{S}$ is computed easily by considering the size of $\mathcal{A}$, $\mathcal{B}$ and the number of partitions of $[k^2]$ into ℓ parts:

$$\hat{f}(n, k, t) = k^{O(1)} \binom{k^2}{\ell} f(k^2, \lceil k/\ell \rceil, \lceil t/\ell \rceil)^\ell \log n.$$

□

We finally arrive at the main theorem of this section.

Theorem 24. *There exists a $(n, k, \alpha k)$-splitter $\hat{\hat{\mathcal{S}}}$ with*

$$\begin{aligned}
\hat{\hat{f}}(n, k, \alpha k) &= k^{O(1)} \tau(\alpha)^{k + o(k)} \log n, \\
\hat{\hat{t}}_{in}(n, k, \alpha k) &= k^{O(1)} \tau(\alpha)^{o(k)} n \log n, & \hat{\hat{s}}_{in}(n, k, \alpha k) &= k^{O(1)} n \log n, \\
\hat{\hat{t}}_{qr}(n, k, \alpha k) &= k^{O(1)} + O(n), & \hat{\hat{s}}_{qr}(n, k, \alpha k) &= k^{O(1)} + O(\log n).
\end{aligned}$$

Proof. We apply the construction of Lemma 23 *twice*, first with $\ell = k/\log^2 k$ and then with $\ell = \log k$, to the splitter $\mathcal{S}$ from Lemma 21. The resulting family has a size of

$$\begin{aligned}
\hat{\hat{f}}(n, k, \alpha k) &= k^{O(1)} \binom{k^2}{\log k} \hat{f}\big(k^2, k/\log k, \alpha k/\log k\big)^{\log k} \log n \\
&= k^{O(1)} e^{o(k)} \log n
\end{aligned}$$

$$\cdot \Big[\binom{(k/\log k)^2}{k/\log^2 k} f\big((k/\log k)^2, \log k, \alpha \log k\big)^{k/\log^2 k} 2\log k\Big]^{\log k}$$
$$= k^{O(1)} e^{o(k)} f\big((k/\log k)^2, \log k, \alpha \log k\big)^{k/\log k} \log n$$
$$= k^{O(1)} e^{o(k)} \Big(\tau(\alpha)^{\log k} 2\log \frac{k}{\log k}\Big)^{k/\log k} \log n$$
$$= k^{O(1)} \tau(\alpha)^{k+o(k)} \log n.$$

The remaining profile of $\hat{\hat{\mathcal{S}}}$ can be bounded as follows:

$$\begin{aligned}
\hat{\hat{t}}_{\mathsf{in}}(n,k,\alpha k) &= k^{O(1)} n\log n + O(\hat{t}_{\mathsf{in}}(k^2, k/\log k, \alpha k/\log k)) \\
&= k^{O(1)} n\log n + O(t_{\mathsf{in}}((k/\log k)^2, \log k, \alpha \log k)) \\
&= k^{O(1)} n\log n + \tau(\alpha)^{\log k} \binom{(k/\log k)^2}{\log k} \big(2(k\log k)^2\big)^{\alpha \log k} \log^{O(1)} k \\
&= k^{O(1)} \tau(\alpha)^{o(k)} n \log n. \\
\hat{\hat{s}}_{\mathsf{in}}(n,k,\alpha k) &= O(\hat{s}_{\mathsf{in}}(k^2, k/\log^2 k, \alpha k/\log^2 k)) + k^{O(1)} n\log n \\
&= O(s_{\mathsf{in}}((k/\log k)^2, \log k, \alpha \log k)) + k^{O(1)} n\log n \\
&= O(\tau(\alpha)^{\log k} \log^2(k/\log k)) + k^{O(1)} n\log n = k^{O(1)} n\log n. \\
\hat{\hat{t}}_{\mathsf{qr}}(n,k,\alpha k) &= O(k\, \hat{t}_{\mathsf{qr}}(k^2, k/\log k, \alpha/\log k) + n) \\
&= O\big(k^2\, t_{\mathsf{qr}}((k/\log k)^2, \log k, \alpha \log k) + n\big) \\
&= k^{O(1)} + O(n) \\
\hat{\hat{s}}_{\mathsf{qr}}(n,k,\alpha k) &= O(k\, \hat{s}_{\mathsf{qr}}(k^2, k/\log k, \alpha k/\log k) + \log n) \\
&= O\big(k^2\, s_{\mathsf{qr}}((k/\log k)^2, k/\log k, \alpha \log k) + k\log k + \log n\big) \\
&= k^{O(1)} + O(\log n).
\end{aligned}$$

□

Since the an indexed splitter with polynomial query time t_{qr} in particular means that we can enumerate its members with polynomial delay, Theorem 4 follows directly from Theorem 24.

8 Proof of Proposition 5

As Hüffner *et al.* noted [36], the running time of color coding algorithms can usually be improved by using more than k colors, balancing the success probability of the coloring with the running time of the algorithm that uses the coloring. In particular, they showed that a typical running time of $\frac{t^k}{\binom{t}{k}\cdot k!} \cdot 2^t$ (were the first term is the reciprocal of the success probability and the second term corresponds to an algorithm that needs time 2^t for an instance colored with t colors) can be bounded by

$$\frac{(\alpha k)^k}{\binom{\alpha k}{k} \cdot k!} \cdot 2^{\alpha k} = \mathcal{O}^*(4.314^k)$$

for $\alpha = 1.3$, a value that was derived numerically. Using the function $\tau(\alpha)$, we can shed some light on how this value comes about. Consider a running time

$$\frac{(\alpha k)^k}{\binom{\alpha k}{k} \cdot k!} \cdot 2^{\alpha k} \leqslant \tau(\alpha)^k 2^{\alpha k} = (h(\alpha) \cdot e)^k,$$

where $h(\alpha) = \big(1 - 1/\alpha\big)^{\alpha-1} 2^\alpha$. Let us minimize $h(\alpha)$ over $\alpha > 1$. Since

$$\frac{\mathrm{d}h(\alpha)}{\mathrm{d}\alpha} = (\alpha - 1)^{\alpha-1} \Big(\frac{2}{\alpha}\Big)^\alpha \Big(\alpha \ln\Big(\frac{2(\alpha-1)}{\alpha}\Big) + 1\Big)$$

and the first two terms of this expression have no roots for $\alpha > 1$, we are left with solving

$$\alpha \ln\Big(\frac{2(\alpha-1)}{\alpha}\Big) = -1 \iff \Big(1 - 1/\alpha\Big)^\alpha 2^\alpha = \frac{1}{e} \tag{4}$$

for α. The only solution is $\alpha^\star = 1.302017\ldots$ and note that $\frac{\mathrm{d}h(\alpha)}{\mathrm{d}\alpha}$ is negative for $\alpha = 1.2$ and positive for $\alpha = 1.4$. Thus, $h(\alpha)$ $(\alpha > 1)$ is minimized for $\alpha = \alpha^\star$. By (4),

$$h(\alpha^\star)\Big(1 - \frac{1}{\alpha^\star}\Big) = \frac{1}{e}$$

and hence the optimal running time is bounded from above by

$$(h(\alpha^\star)e)^k = \Big(\frac{\alpha^\star}{\alpha^\star - 1}\Big)^k \leqslant \Big(\frac{1.302018}{0.302017}\Big)^k = \mathcal{O}^*(4.312^k).$$

This proves Proposition 5 for $p = 0$ and thus for arbitrary p.

9 Discussion

In this paper, we presented an enhancement of color coding to design polynomial-space parameterized algorithms. We provided three applications centered around out-branchings, spanning trees, and perfect matchings. In particular, we obtained a deterministic polynomial-space algorithm for k-IOB that runs in time $\mathcal{O}^*(3.86^k)$. Along the way, we showed how to enumerate $(n, k, \alpha k)$-splitters one by one with polynomial delay. In addition, we demonstrated that our approach can be adapted to enable the development of faster parameterized algorithms in case the use of exponential space is permitted.

Let us conclude our paper with an interesting open question. Let $\alpha_1(D)$ be the maximum size of a matching in a digraph D. By Lemma 7 (ii), D has an $\alpha_1(D)$-internal out-branching. The following problem is natural: what is the parameterized complexity of deciding whether D has an $(\alpha_1(D) + k)$-internal out-branching, where k is the parameter?

Acknowledgment. We thank Saket Saurabh for helpful information concerning splitters.

References

[1] I. Koutis, Faster algebraic algorithms for path and packing problems, in: Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games, 2008, pp. 575–586.

[2] R. Williams, Finding paths of length k in $O^*(2^k)$ time, Inf. Process. Lett. 109 (2009) 315–318.

[3] I. Koutis, R. Williams, Limits and applications of group algebras for parameterized problems, ACM Trans. Algorithms 12 (2016) 31:1–31:18.

[4] A. Björklund, Determinant sums for undirected hamiltonicity, SIAM J. Comput. 43 (2014) 280–299.

[5] R. Bellman, Dynamic programming treatment of the travelling salesman problem, J. ACM 9 (1962) 61–63.

[6] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Narrow sieves for parameterized paths and packings, J. Comput. Syst. Sci. 87 (2017) 119–139.

[7] A. Gabizon, D. Lokshtanov, M. Pilipczuk, Fast algorithms for parameterized problems with relaxed disjointness constraints, in: Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings, 2015, pp. 545–556.

[8] A. Björklund, T. Husfeldt, N. Taslaman, Shortest cycle through specified elements, in: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012, 2012, pp. 1747–1753.

[9] M. Wahlström, Abusing the Tutte matrix: An algebraic instance compression for the k-set-cycle problem, in: N. Portier, T. Wilke (Eds.), 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, volume 20 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, pp. 341–352.

[10] A. Björklund, P. Kaski, I. Koutis, Directed hamiltonicity and out-branchings via generalized Laplacians (arxiv.org/abs/1607.04002), in: Automata, Languages and Programming, 44th International Colloquium, ICALP 2017, Warsaw, Poland, July 10-14, 2017, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games, 2017, p. TBA.

[11] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[12] A. Itai, M. Rodeh, S. L. Tanimoto, Some matching problems for bipartite graphs, J. ACM 25 (1978) 517–525.

[13] V. B. Le, F. Pfender, Complexity results for rainbow matchings, Theor. Comput. Sci. 524 (2014) 27–33.

[14] M. Kano, X. Li, Monochromatic and heterochromatic subgraphs in edge-colored graphs-a survey, Graphs and Combinatorics 24 (2008) 237–263.

[15] G. Gutin, I. Razgon, E. J. Kim, Minimum leaf out-branching and related problems, Theor. Comput. Sci. 410 (2009) 4571–4579.

[16] P. Dankelmann, G. Gutin, E. J. Kim, On complexity of minimum leaf out-branching problem, Discrete Applied Mathematics 157 (2009) 3000–3004.

[17] T. Johnson, N. Robertson, P. D. Seymour, R. Thomas, Directed tree-width, J. Comb. Theory, Ser. B 82 (2001) 138–154.

[18] E. Prieto, C. Sloper, Reducing to independent set structure – the case of k-internal spanning tree, Nord. J. Comput. 12 (2005) 308–318.

[19] N. Cohen, F. V. Fomin, G. Gutin, E. J. Kim, S. Saurabh, A. Yeo, Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem, J. Comput. Syst. Sci. 76 (2010) 650–662.

[20] F. V. Fomin, F. Grandoni, D. Lokshtanov, S. Saurabh, Sharp separation and applications to exact and parameterized algorithms, Algorithmica 63 (2012) 692–706.

[21] F. V. Fomin, S. Gaspers, S. Saurabh, S. Thomassé, A linear vertex kernel for maximum internal spanning tree, J. Comput. Syst. Sci. 79 (2013) 1–6.

[22] H. Shachnai, M. Zehavi, Representative families: A unified tradeoff-based approach, J. Comput. Syst. Sci. 82 (2016) 488–502.

[23] J. Daligault, Combinatorial techniques for parameterized algorithms and kernels, with applicationsto multicut, PhD thesis, Universite Montpellier II, Montpellier, Herault, France (2011).

[24] W. Li, Y. Cao, J. Chen, J. Wang, Deeper local search for parameterized and approximation algorithms for maximum internal spanning tree, Inf. Comput. 252 (2017) 187–200.

[25] M. Zehavi, Mixing color coding-related techniques, in: N. Bansal, I. Finocchi (Eds.), Algorithms - ESA 2015 - 23rd Annual European Symposium, volume 9294 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 1037–1049.

[26] A. Björklund, V. Kamat, L. Kowalik, M. Zehavi, Spotting trees with few leaves, in: Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I, 2015, pp. 243–255.

[27] J. Chen, J. Kneis, S. Lu, D. Molle, S. Richter, P. Rossmanith, S. H. Sze, F. Zhang, Randomized divide-and-conquer: Improved path, matching, and packing algorithms, SIAM J. on Computing 38 (2009) 2526–2547.

[28] F. V. Fomin, D. Lokshtanov, F. Panolan, S. Saurabh, Efficient computation of representative families with applications in parameterized and exact algorithms, J. ACM 63 (2016) 29:1–29:60.

[29] J. Bang-Jensen, G. Z. Gutin, Digraphs - theory, algorithms and applications, 2 ed., Springer, 2009.

[30] R. Diestel, Graph Theory, 4th Edition, volume 173 of *Graduate texts in mathematics*, Springer, 2012.

[31] D. C. Lay, Linear Algebra and Its Applications, Addison-Wesley, 2012.

[32] H. Robbins, A remark on stirling's formula, The American Mathematical Monthly 62 (1955) 26–29.

[33] W. Tutte, The dissection of equilateral triangles into equilateral triangles, Proc. Cambridge Philos. Soc. 44 (1948) 463–482.

[34] N. Alon, R. Yuster, U. Zwick, Color-coding, J. ACM 42 (1995) 844–856.

[35] M. Naor, L. J. Schulman, A. Srinivasan, Splitters and near-optimal derandomization, in: Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on, IEEE, 1995, pp. 182–191.

[36] F. Hüffner, S. Wernicke, T. Zichner, Algorithm engineering for color-coding with applications to signaling pathway detection, Algorithmica 52 (2008) 114–132.

[37] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Trimmed moebius inversion and graphs of bounded degree, Theory Comput. Syst. 47 (2010) 637–654.

[38] P. W. Kasteleyn, Graph theory and crystal physics, in: F. Harary (Ed.), Graph Theory and Theoretical Physics, Academic Press, 1967, pp. 43–110.

[39] A. Joffe, On a set of almost deterministic k-independent random variables, Annals of Probability 2 (1974) 161–162.