Nearly Optimal Time Bounds for \( k \)-Path in Hypergraphs

Lior Kamma \(^*\)  \hspace{1em}  Ohad Trabelsi \(^†\)

February 20, 2019

Abstract

We give almost tight conditional lower bounds on the running time of the \( k \)-HyperPath problem. Given an \( r \)-uniform hypergraph for some integer \( r \), \( k \)-HyperPath seeks a tight path of length \( k \). That is, a sequence of \( k \) nodes such that every consecutive \( r \) of them constitute a hyperedge in the graph. This problem is a natural generalization of the extensively-studied \( k \)-Path problem in graphs. We show that solving \( k \)-HyperPath in time \( O^*(2^{(1-\gamma)k}) \) where \( \gamma > 0 \) is independent of \( r \) is probably impossible. Specifically, it implies that Set Cover on \( n \) elements can be solved in time \( O^*(2^{(1-\delta)n}) \) for some \( \delta > 0 \). The only known lower bound for the \( k \)-Path problem is \( 2^{\Omega(k)} \text{poly}(n) \) where \( n \) is the number of nodes assuming the Exponential Time Hypothesis (ETH), and finding any conditional lower bound with an explicit constant in the exponent has been an important open problem.

We complement our lower bound with an almost tight upper bound. Formally, for every integer \( r \geq 3 \) we give algorithms that solve \( k \)-HyperPath and \( k \)HyperCycle on \( r \)-uniform hypergraphs with \( n \) nodes and \( m \) edges in time \( 2^k m \cdot \text{poly}(n) \) and \( 2^k m^2 \text{poly}(n) \) respectively, and that is even for the directed version of these problems. To the best of our knowledge, this is the first algorithm for \( k \)-HyperPath. The fastest algorithms known for \( k \)-Path run in time \( 2^k \text{poly}(n) \) for directed graphs (Williams, 2009), and in time \( 1.66^k \text{poly}(n) \) for undirected graphs (Björklund et al., 2014).

1 Introduction

In the \( k \)-Path problem, given a graph \( G \) and an integer \( k \), the goal is to decide whether \( G \) contains a simple path of length \( k \). This is a fundamental combinatorial optimization problem, and contains the Hamiltonian Path as a special case. The fastest algorithm known for directed graphs runs in time \( 2^k \text{poly}(n) \) [Wil09] and for the undirected version the fastest algorithm runs in time \( 1.66^k \text{poly}(n) \) [BHPK17]. The only known lower bound for these problems is \( 2^{\Omega(k)} \text{poly}(n) \) assuming the Exponential Time Hypothesis (by a simple reduction from Hamiltonian Path), and finding any conditional lower bound with explicit constant in the exponent is an important open problem.

Motivated by this challenge, we study the \( k \)-HyperPath problem, which is a natural generalization of \( k \)-Path. Given an \( r \)-uniform hypergraph \( G \), the goal is to find a tight path in \( G \) of length \( k \). That is, a sequence of \( k \) nodes such that every \( r \) consecutive nodes constitute an edge in the graph. The cycle variant of problem was previously studied by Lincoln et al. [LWW18] for large values of the uniformity parameter \( r = \Omega(k) \), where a conditional lower bound

\(^*\)Computer Science Department. Aarhus University. Supported by a Villum Young Investigator Grant. lior.kamma@cs.au.dk.

\(^†\)Weizmann Institute of Science, Rehovot, Israel ohad.trabelsi@weizmann.ac.il.
bound of $\Omega(n^k)$ was presented. Considering smaller values of $r$, we show that for every $\gamma > 0$ there is some integer $r \geq 3$ such that solving $k$-HyperPath in time $O^*(2^{(1-\gamma)k})$ on $r$-uniform hypergraphs is most likely impossible. Formally, we present a hierarchy of conditional lower bounds for $k$-HyperPath in undirected hypergraphs with explicit constants in the exponents, where these lower bounds approach $O^*(2^k)$ as $r$ increases.

We complement our conditional lower bounds with almost matching upper bounds. That is, for every integer $r \geq 3$ we show an algorithm with running time $2^k m \cdot \text{poly}(n)$ for $k$-HyperPath on $r$-uniform hypergraphs with $n$ nodes and $m$ edges even for the directed version of this problem, where every hyperedge is a sequence of $r$ nodes, and tight paths respect the ordering of the corresponding edges.

1.1 Our Results

Our main result shows that $k$-HyperPath cannot be solved faster than $O^*(2^k)$ by an exponential factor independent of the size $r$ of a hyperedge unless Set Cover on $n$ elements can be solved significantly faster than $O^*(2^n)$. The latter statement is known as the Set Cover conjecture, introduced by Cygan et al. [CDL+16]. Formally, we show the following.

**Theorem 1.** Let $\gamma > 0$ and let $r \geq 3$ be an integer. If $k$-HyperPath in undirected $r$-uniform hypergraphs can be solved in time $O^*(2^{(1-1/(r-1)-\gamma)k})$, then there exists $\delta = \delta(\gamma) > 0$, such that Set Cover on $n$ elements can be solved in time $O^*(2^{(1-\delta)n})$.

We prove Theorem 1 by showing a reduction from Set Cover to $k$-HyperPath. The reduction consists of two main parts. We first show that Set Cover can be reduced to Exact Cover, thus implying that Exact Cover is “at least as hard” as Set Cover.

**Lemma 2.** Let $\gamma > 0$. If Exact Cover on $n$ elements can be solved in time $O^*(2^{(1-\gamma)n})$, then there exists a constant $\delta > 0$ such that Set Cover on $n$ elements can be solved in time $O^*(2^{(1-\delta)n})$.

To the best of our knowledge, this “Set Cover hardness” of Exact Cover was not known before. As Exact Cover seems to be a more robust source of reductions than its optimization variant Set Partitioning, due to lack of budget constraints, this result may be of independent interest.

Secondly, we show that Exact Cover can be reduced to $k$-HyperPath. Combining these two reduction constitutes the proof of Theorem 1.

**Lemma 3.** Let $\gamma > 0$ and let $r \geq 3$ be an integer. If $k$-HyperPath in undirected $r$-uniform hypergraphs can be solved in time $O^*(2^{(1-1/(r-1)-\gamma)k})$, then there exists a constant $\delta > 0$, such that Exact Cover on $n$ can be solved in time $O^*(2^{(1-\delta)n})$.

Lemma 3 as well as its proof demonstrate that Exact Cover may be the right problem to associate with $k$-Path, given the connection it shows between its generalization $k$-HyperPath and Exact Cover. We note that by simple modifications to our proofs, all of our results can also be applied for the problem of finding cycles rather than paths with some small (polynomial in the input size) overhead. In addition, our conditional lower bounds for the undirected case imply the same bounds for the directed one by the following simple reduction. Given an $r$-uniform undirected hypergraph $H$, construct an $r$-uniform directed hypergraph $\tilde{H}$ by including for every edge $e \in H$ all possible orientations of $e$. Clearly there is a $k$-path in $\tilde{H}$ if and only if there is one in $H$. Moreover, the size of $\tilde{H}$ is at most $r!$ times the size of $H$. 


We accompany our lower bound by an almost tight upper bound showing that even directed $k$-HyperPath is not too difficult a generalization of $k$-Path. While directed $k$-Path can be solved in time $O^*(2^k)$, it seems that there is no trivial way to extend this algorithm to $k$-HyperPath. Our second result shows such an extension. That is, for every fixed integer $r$, directed $r$-uniform $k$-HyperPath admits an algorithm with running time $2^{kn}O^{*(1)}$.

**Theorem 4.** For every integer $r \geq 3$, directed $r$-uniform $k$-HyperPath (respectively $k$HyperCycle) can be solved in time $2^{kn}O^{*(1)}$ for some universal constant $c > 0$ (respectively $2^k m^2 n O^{*(1)}$), where $n$ and $m$ are the number of nodes and hyperedges respectively.

Our results show that while $k$-HyperPath can essentially be solved as fast as $k$-Path, we know that for the former it is nearly tight, assuming the hardness of Set Cover.

Note that once again by the previously described reduction our algorithm could be applied to the easier undirected case with the same running time (for $r = O(1)$). Our algorithm is achieved through a reduction to the $k$-Multilinear Monomial Detection problem ($k$-MLD), where the goal is to detect multilinear monomials of degree $k$ in a polynomial presented as a circuit. This problem has been utilized to solve $k$-Path [Kou08, Wil09], and our method can be seen as an extension to $k$-HyperPath. Our method uses as a black box an algorithm for $k$-MLD with running time $2^k s(n)n O^{*(1)}$ by Williams [Wil09], where $s(n)$ is the size of the circuit.

### 1.2 Previous Work

$k$-Path can be solved naively in time $\tilde{O}(n^k)$, however a long line of work devoted effort to find faster algorithms, starting with $O^*(f(k))$-time algorithms, specifically $O^*(k!)$ [Mon88] and $O^*(k!2^k)$ [Bod93], followed by a series of improvements [KMRR06, CLSZ07] with the color coding method [AYZ95] being a notable one. Finally, the fastest methods that were developed [Kou08, Wil09] utilize $k$-MLD to create $k$-Path algorithms with running times $O^*(2^{3k/2})$ and $O^*(2^k)$ respectively.

On the hardness front, it is well known that $k$-Path requires time $2^{\Omega(k)}\text{poly}(n)$ assuming ETH, as Hamiltonian path is a special case of this problem and there is a reduction from 3-SAT to Hamiltonian Path with the number of nodes in the produced instance linear in the formula size. If we care about the exact exponent in the running time, only a restricted lower bound is known. Koutis and Williams [KW16] used communication complexity to show that a faster algorithm for their intermediate problem $k$-MLD in some settings is not possible, and so among a specific class of algorithms, their $O^*(2^k)$ algorithms for $k$-Path and $k$-Tree, a generalization of $k$-Path whose goal is finding an isomorphic copy of a given tree of size $k$ in a given graph, is optimal. Krauthgamer and Trabelsi [KT17] show that $k$-Tree also requires $\Omega^*(2^k)$ time assuming Set Cover requires $\Omega^*(2^n)$ time. In the other direction, they show that if Set Cover on sets of size bounded by $O(\log n)$ can be solved in time significantly faster than $O^*(2^n)$, then also $n$-Tree can be solved in time significantly faster than $O^*(2^n)$, where $n$-Tree is $k$-Tree but with a pattern tree that has number of nodes equals $|V(G)|$ (thus $n$-Tree generalizes Hamiltonian Path).

**The Set Cover Conjecture.** The set-cover conjecture formally states that for every fixed $\varepsilon > 0$ there is an integer $\ell = \ell(\varepsilon) > 0$ such that Set Cover with sets of size at most $\ell$ cannot be solved in time $O^*(2^{(1-\varepsilon)n})$. The conjecture clearly implies that for every fixed $\varepsilon > 0$, Set Cover cannot be solved in time $O^*(2^{(1-\varepsilon)n})$. In spite of extensive effort, the
fastest algorithm for Set Cover is still essentially a dynamic programming algorithm that runs in time $O^*(2^n)$ [FKW04], with several improvements in special cases [Koi09, BHK09, Ned11, BHPK17]. Several conditional lower bounds were based on this conjecture in the recent decade, including for Set Partitioning, Connected Vertex Cover, Steiner Tree, Subset Sum [CDL+16] (though the last problem was later shown hard conditioned on the strong exponential time hypothesis [ABHS19]), Maximum Graph Motif [BKK16], parity of the number of solutions to Set Cover with at most $\ell$ sets [BHH15], Colorful Path and Colorful Cycle [KL16], the dynamic, general and connected versions of Dominating Set [KST17], and $k$-Tree [KT17].

2 Preliminaries

Paths in Hypergraphs. As there are several ways to generalize simple paths from ordinary graphs to hypergraphs we define it as follows, extending known definitions from the Hamiltonicity context. Let $H = (V, E)$ be an $r$-uniform hypergraph. For every $1 \leq \ell \leq r - 1$ and $k$ such that $(r - \ell)$ divides $(k - \ell)$, an $\ell$-overlapping $k$-path is a sequence of $k$ distinct nodes and $(k - \ell)/(r - \ell)$ hyperedges such that each hyperedge consists of $r$ consecutive nodes, and every pair of consecutive edges $e, e'$ satisfies $|e \cap e'| = \ell$. In the case where $\ell = r - 1$, such paths are called tight paths, and throughout we will focus on such paths. This natural generalization of paths to hypergraphs has been extensively studied in the area of Hamiltonian paths and cycles, either in the context of algorithms (e.g., [FKL11, GM16]) or combinatorics (e.g., [DFRS17, HZ15]). In particular, Lincoln et al. [LWW18] show that finding a tight cycle of size $k$ in $r$-uniform hypergraphs for $r = k - \lceil k/r' \rceil + 1$ cannot be solved significantly faster than $\tilde{O}(n^k)$ unless MAX-$r$-SAT can be solved significantly faster than $O^*(2^n)$, and also $r'$-uniform $k$-HyperClique can be solved significantly faster than $\tilde{O}(n^k)$.

Set Cover, Exact Cover and Set Partitioning. For sake of completeness of the text, and to avoid confusion, we give a formal definition for each of the three set covering problems discussed in this paper. The input for all three problems is the same. We are given a ground set $U$ and a family $S \subseteq 2^U$ of subsets of $U$. A sub-cover of $U$ from $S$ is a subfamily $S' \subseteq S$ whose union is $U$. In the Set Cover problem the goal is to find a sub-cover of minimal size. The Exact Cover problem is a decision problem that seeks to find whether there exists some sub-cover composed of pairwise disjoint subsets of $U$, also called a partition. Finally, the Set Partitioning problem is the optimization variant of Exact Cover. That is, the goal is to find a partition of minimal size.

3 A Conditional Lower Bound for $k$-HyperPath

This section is devoted to proving Theorem 1 by presenting a reduction from Set Cover to $k$-HyperPath. The reduction is presented in two steps. The first step, in which we prove Lemma 2 reduces Set Cover to Exact Cover. In the second step, which constitutes the technical crux of the proof, we present a reduction from Exact Cover to $k$-HyperPath, thus proving Lemma 3.

3.1 Reduction from Set Cover to Exact Cover

In this section we show that if Exact Cover can be solved in time significantly better than $O^*(2^n)$, then so does Set Cover, thus proving Lemma 2. More formally, we assume that
there exist $c, \gamma > 0$ such that \textbf{Exact Cover} on $n$ elements and $m$ sets can be solved in time $O(m^c2^{(1-\gamma)n})$ and construct an algorithm that solves \textbf{Set Cover} on $n$ elements and $m$ sets in time $O(poly(m)2^{(1-\delta)n})$ for some $\delta = \delta(c, \gamma) \in (0, 1/2)$ to be determined later.

By the self-reducibility property of \textbf{Set Cover}, it is enough to show that the decision version of \textbf{Set Cover} can be solved in time $O(poly(m)2^{(1-\delta)n})$. That is, given a \textbf{Set Cover} instance and a threshold $t \in \mathbb{N}$, the goal is to decide if there is a set cover of size at most $t$. We will show that for large values of $t$, this problem can be solved using an algorithm by Nederlof \cite{Nederlof16}. For small values of $t$, we use a reduction by Nederlof to the decision version of \textbf{Set Partitioning}, and then reduce this problem to solving many (but not too many) instances of \textbf{Exact Cover}.

To this end, and following the notation suggested by Nederlof, we refer to instances of \textbf{Set Cover} and \textbf{Set Partitioning} as $(n, m, t)$-instances to denote that the number of elements in the ground set is $n$, the number of subsets given is $m$ and the threshold given is $t$.

Assume therefore that we are given an $(n, m, t)$-instance of \textbf{Set Cover}. We first note that if $t \geq \sqrt[3]{n}n$, then the following result by Nederlof solves the problem in time $O(poly(m)2^{(1-\delta)n})$.

**Lemma 5 (\cite{Nederlof16}).** There is a Monte-Carlo algorithm that takes a \textbf{Set Cover} instance on $n$ elements and $m$ sets, as well as an integer $t$, and determines in time $O^*(2^{1-\frac{1}{2n}t})$ if there is a set cover of size $t$.

Assume therefore that $t \leq \sqrt[3]{n}n$. Applying the following reduction by Nederlof, we construct an $(n, m', t)$-instance of \textbf{Set Partitioning} with $m' \leq m2^{2n}$.

**Lemma 6 (\cite{Nederlof16}).** There is an algorithm that, given a real $0 < \delta < 1/2$, takes an $(n, m, t)$-instance of \textbf{Set Cover} as input and outputs an equivalent $(n, m', t)$-instance of \textbf{Set Partitioning} with $m' \leq m^{\delta n}$ sets in time $O(m2^{1-\delta n})$.

Given the $(n, m', t)$-instance of \textbf{Set Partitioning} constructed above we then construct $n2^{2t}$ instances of \textbf{Exact Cover} by repeating the following color-coding scheme independently at random. Sample a random coloring $f : m' \rightarrow [t]$ of the input sets to $t$ colors. Add $t$ new elements, each one associated with a different color, and add each such element to all subsets of that color. Note that this \textbf{Exact Cover} instance has $n' = n + t \leq (1 + \sqrt[3]{3})n$ elements, and $m' \leq m2^{2n}$ sets. The following lemma shows that this reduction reduces \textbf{Set Partitioning} to \textbf{Exact Cover}.

**Lemma 7.** If the \textbf{Set Partitioning} is a 'yes' instance, then with high probability at least one of the \textbf{Exact Cover} instances is a 'yes' instance. Conversely, if one of the \textbf{Exact Cover} instances is a 'yes' instance, then the \textbf{Set Partitioning} instance is a 'yes' instance with certainty.

**Proof.** Assume first that the $(n, m', t)$-instance of \textbf{Set Partitioning} is a 'yes' instance. Then there exists a partition of size $\ell \leq t$. With probability at least $\frac{\ell^t}{(t-1)!m'} \geq \frac{t^t}{t!} \geq e^{-t}$, a uniform coloring of the sets colors each set of the partition with a different color. Therefore for every new element added, there is at most one set in the partition with the same color, and therefore there is an exact cover in the corresponding instance. Since the reduction constructs $n : 2^{2t}$ independent instances, with high probability at least one will be a 'yes' instance. Conversely, assume that one of the \textbf{Exact Cover} instances is a 'yes' instance, and consider an exact cover for this instance. Since the sets in the cover are disjoint, for every new element added, there is at most one subset in the cover containing it. We can therefore conclude that no two subsets are colored with the same color. Removing the new elements yields a feasible solution to the \textbf{Set
Partitioning instance in which every subset is colored differently. Therefore the cover consists of at most $t$ subsets.

By our original assumption, each of the Exact Cover instances can be solved in time $m^c \cdot 2^{(1-\gamma)n'}$. Therefore the total time required to solve all instances is at most

$$n2^{2t}O\left(m^c \cdot 2^{(1-\gamma)n'}\right) \leq O\left(n2^{2\gamma\delta n} \cdot m^{\gamma+\delta n} \cdot 2^{(1-\gamma)(1+\gamma\delta)n}ight) = O^*\left(2^{(2\gamma\delta+\gamma\delta+2\gamma)(1+\gamma\delta)n}\right).$$

For small enough choice of $\delta$ (depending only on $\gamma$ and $c$), this is at most $O(nm^{c}2^{(1-\delta)n})$. This completes the proof of Lemma 3.

3.2 Reduction from Exact Cover to $k$-HyperPath

In this section we show that if $k$-HyperPath can be solved in time significantly better than $O^*(2^k)$, then so does Exact Cover, thus proving Lemma 3. More formally, we show that if there exists a constant $\gamma > 0$ and an integer $r \geq 3$ such that $k$-HyperPath on $n$ elements and $m$ sets can be solved in time $O^*(2^{(1-1/(r-2)-\gamma)k})$ then Exact Cover on $n$ elements and $m$ sets can be solved in time $O(\text{poly}(m)2^{(1-\delta)n})$ for some $\delta = \delta(\gamma) \in (0, 1/2)$ to be determined later.

To this end, let $X = \{x_1, \ldots, x_n\}$ and $S = \{S_1, \ldots, S_m\}$ be an Exact Cover instance, and let $r \geq 3$ be fixed. We will present a procedure that runs in time polynomial in $n, m$ and constructs a $k$-HyperPath instance on an undirected $r$-uniform hypergraph $H = (V_H, E_H)$ with $|V_H| = O(mn)$ nodes and $k = O(n)$. We first present the construction under the following assumption. For sake of fluency we will show how to discard it after presenting the construction.

Assumption 1. $n \geq 4r$, $r-2$ divides $n+2$, and for every $i \in [m]$, $|S_i| \geq 2r$.

We start by defining the vertex set $V_H$. For every $j \in [n]$, the hypergraph has a node labeled $x_j$ (the distinction between the node and the corresponding element will be clear from the context). In addition, for every $i \in [m]$ and $j \in [|S_i|]$ we define a node labeled $u^j_i$. The former set of nodes will be referred to as element-nodes and the latter one as set-nodes. Finally, we add two more element-nodes labeled $x_{\text{start}}, x_{\text{end}}$ and two set-nodes labeled $u_{\text{start}}, u_{\text{end}}$. Next we turn to define the set $E_H$ of hyperedges, also demonstrated in Figure 4. Loosely speaking $H$ has four types of edges. Internal hyperedges consist of nodes associated with a single set, transition edges consist of nodes associated with two disjoint sets, starting hyperedges (resp. ending hyperedges) consist of nodes associated with a single set together with a starting (resp. ending) node.

To define the edges formally, for every $i \in [m]$ and $j \in [|S_i|]$ we let $x^j_i$ denote the $j$th element in $S_i$. We emphasize that $x^j_i$ is simply a new notation for an element in $X$. Therefore while for every $i \neq i'$ and $j \in [|S_i|], j' \in [|S_{i'}|], u^j_i \neq u^{j'}_{i'}$, it might be the case that $x^j_i = x^{j'}_{i'}$. Loosely speaking, the element-nodes are a “common resource” shared by all sets, while each set has “its own” set-nodes. Finally, although the edges are undirected, intuitively it might prove useful for the reader to think of the edge as a sequence, rather than a set.

Every internal edge contains a sequence of either $r-2$ or $r-1$ elements-nodes, all belonging to the same set. In the former case, the edge begins and ends with a set-node. In the latter,
Let 1 ≤ i < i' ≤ m be such that $S_i, S_{i'}$ are disjoint. A transition edge contains a sequence of either $r - 2$ or $r - 1$ element-nodes, the first part of which consists of elements of $S_i$ and ends with $x_{i+1}^{|S_i|}$, and the second is either empty or consists of elements of $S_{i'}$ starting with $x_{i'}^{|S_i|}$. In the former case, the edge begins with a set-node associated with $S_i$ and ends with a set-node associated with $S_{i'}$. In the latter, the edge either contains a set-node associated with $S_i$ in the midst of the subsequence associated with $S_i$ elements or a set-node associated with $S_{i'}$ in the midst of the subsequence associated with $S_{i'}$ elements. Formally, for all $i < i'$ such
that $S_i \cap S_{i'} = \emptyset$ the transition hyperedges are defined as follows.

\[
\begin{align*}
\{u_i^{S_i \setminus j}\} & \cup \{x_{i}^{S_i \setminus j}-j\} & \forall j \in [0, r-3] \\
\{x_{i}^{S_i \setminus j}+h\} & \cup \{u_i^{S_i \setminus j}+h+1\} & \forall j \in [r-3], h \in [0, j-1] \\
\{x_{i}^{S_i \setminus j-1}\} & \cup \{x_{i}^{S_i \setminus j}+h+1\} & \forall j \in [r-3], h \in [r-j-2]
\end{align*}
\]

Finally, we define the starting and ending edges. Intuitively, these edges are destined to be the start and ending of the desired path. Formally, for every $i \in [m]$ we define the following.

\[
\begin{align*}
\{u_{\text{start}}, x_{\text{start}}\} & \cup \{x_{i}^{r-3}\} & \forall j \in [0, r-3] \\
\{x_{\text{start}}\} & \cup \{x_{i}^{r-2}\} \\
\{u_i^{S_i \setminus (r-3)}\} & \cup \{x_{i}^{S_i \setminus (r-4)}\} & \forall j \in [0, r-3] \\
\{u_i^{S_i \setminus (r-3)}\} & \cup \{x_{i}^{S_i \setminus (r-4)}\} & \forall j \in [0, r-3]
\end{align*}
\]

To complete the $k$-HyperPath instance we set $k = (n + 2)(1 + 1/(r - 2)) + 1$. Note that by Assumption $k \geq 3r$ is a positive integer. The following claim is straightforward.

**Claim 8.** Given the Exact Cover instance $X, S$, we can construct $H, k$ in polynomial time.

We will now show that $H$ contains a path of length $k$ if and only if the Exact Cover instance is a 'yes' instance. We first show that if $H$ contains a path of length $k$ then the Exact Cover instance is a 'yes' instance. To this end, assume there is a tight path

\[P = v_1 - v_2 - \ldots - v_k\]

of length $k$ in $H$. Let $S_P \subset S$ be the collection of all sets in $S$ that have a set-node in $P$. Formally, let $\mathcal{I}_P := \{i \in [m] : \exists j \in [S_i], u_i^j \in P\}$ then $S_P := \{S_i : i \in \mathcal{I}_P\}$. We will show the following.

**Lemma 9.** $S_P$ is an exact cover of $X$.

To prove the lemma we start with characterizing the structure of $P$. In what follows, we refer to hyperedges containing exactly two set-nodes as heavy and to hyperedges containing exactly one set-node as light. The following straightforward claim follows directly from the construction.

**Claim 10.** Let $e, e' \in E_H$ be two hyperedges, and let $i \in [m]$ and $j \in [S_i]$.

1. $e \cap e'$ contains at most one set-node.
2. If $e$ is light and $u_i^j \in e$ then $x_i^j \in e$. If $j \geq 2$ then $x_i^{j-1} \in e$.
3. If $e, e'$ are both light, and $u_i^j \in e \cap e'$, then $e \cap e'$ contains at least two element-nodes.
4. If $e, e'$ are both heavy, and $u_i^j \in e \cap e'$, then $e \cap e' = \{u_i^j\}$.

**Lemma 11.** Let $1 \leq \alpha < \beta \leq k$ be such that $v_\alpha, v_\beta$ are consecutive set nodes in $P$. Then

\[\beta - \alpha = r - 1.\]
Proof. We will start by proving $\beta - \alpha \leq r - 1$. Since $P$ is tight, $e = \{v_\ell \}^{\alpha+r}_{\ell=\alpha+1} \in E_H$, and since every edge contains at least one set-node, $\beta \leq \alpha + r$ thus $\beta - \alpha \leq r$. Assume towards contradiction that $\beta - \alpha = r$. Since $k \geq 3r$, either $k - \beta \geq r - 1$ or $\alpha \geq r$. Without loss of generality, assume $k - \beta \geq r - 1$ (the other case is analogous). Denote $e' = \{v_\ell \}^{\beta+r-2}_{\ell=\beta+1}$, $e'' = \{v_\ell \}^{\beta+r-1}_{\ell=\beta} \in E_H$. By Claim 10, since $e', e''$ share the set-node $v_\beta$, then $v_{\beta+1}, v_{\beta+2}, \ldots, v_{\beta+r-2}$ are all element-nodes. Therefore $e, e'$ both contain only one set-node, namely $v_\beta$, which they also share, however their intersection contains only one element-node, in contradiction to Claim 10. Therefore $\beta - \alpha \leq r - 1$. 

Next we prove that $\beta - \alpha \geq r - 1$. Assume first that $\alpha > 1$. Then $\{v_\ell \}^{\alpha+r-2}_{\ell=\alpha-1}$ and $\{v_\ell \}^{\alpha+r-1}_{\ell=\alpha}$ are both edges of $H$ that share $v_\alpha$, and by Claim 10 cannot share $v_\beta$. Therefore $\beta \geq \alpha + r - 1$ thus $\beta - \alpha \geq r - 1$. Otherwise, if $\alpha = 1$, then $k - \beta \geq 2r$. Since $\{v_\ell \}^{\beta+r}_{\ell=\beta+1} \in E_H$, there exists $\beta + 1 \leq \gamma \leq \beta + r$ such that $v_\gamma$ is a set-node. By the first part of the proof, $\gamma - \beta \leq r - 1$, and since $\beta > 1$, then by the proof for the case $\alpha > 1$ we get that $\gamma - \beta = r - 1$. Therefore $\{v_\ell \}^{\beta+r}_{\ell=\alpha}, \{v_\ell \}^{\beta+r-1}_{\ell=\beta}$ are both heavy, and they share $v_\beta$. By Claim 10, they do not have any element-node in common, and therefore $r \leq \beta$ thus $r - 1 \leq \beta - 1 = \beta - \alpha$. We conclude that $\beta - \alpha = r$. 

We can now give a complete characterization of $P$ in terms of set-nodes and element-nodes.

**Lemma 12.** For every $\alpha \in [k]$, $v_\alpha$ is a set node if and only if $r - 1$ divides $\alpha - 1$.

**Proof.** Let $\alpha_1, \alpha_2, \ldots, \alpha_t$ be such that $v_{\alpha_1}, v_{\alpha_2}, \ldots, v_{\alpha_t}$ are the only set-nodes in $P$ in consecutive order. From the previous lemma it follows that between every two set-nodes there are exactly $r - 2$ element-nodes. It follows in addition by induction that $\alpha_t - \alpha_1 = (t - 1)(r - 1)$. Therefore the number of element-nodes between $v_{\alpha_1}$ and $v_{\alpha_t}$ is $(t - 1)(r - 2)$. In addition, the first $\alpha_t - 1$ nodes, as well as the last $k - \alpha_t$ are element-nodes. Therefore

$$(t - 1)(r - 2) + (\alpha_1 - 1) + (k - \alpha_t) \leq n + 2$$

(1)

By the definition of $k$ we have that

$$(k - \alpha_t) + (\alpha_t - j_1) + (\alpha_1 - 1) = k - 1 = \frac{n + 2}{r - 2}(r - 1),$$

and therefore

$$n + 2 = (r - 2) \frac{k - 1}{r - 1} = (t - 1)(r - 2) + (r - 2) \frac{(k - \alpha_t) + (\alpha_1 - 1)}{r - 1}. \quad (2)$$

Combining (1) and (2) we get that $(\alpha_1 - 1) + (k - \alpha_t) \leq \frac{r-2}{r-1}[(k - \alpha_t) + (\alpha_1 - 1)]$. It follows that $(\alpha_1 - 1) + (k - \alpha_t) = 0$, and therefore $\alpha_1 = 1$ and $\alpha_t = k$. The lemma now follows by simple induction. 

**Corollary 13.** Every element-node appears exactly once in $P$. Moreover, $v_1 = u_{\text{start}}$, $v_k = u_{\text{end}}$, $v_2 = x_{\text{start}}$ and $v_{k-1} = x_{\text{end}}$.

**Proof.** From Lemma 12 the number of element-nodes in $P$ is $(r - 2) \cdot (k - 1)/(r - 1) = n + 2$. Since $P$ is simple, every element-node appears exactly once in $P$, and specifically $x_{\text{start}}$ and $x_{\text{end}}$. Therefore $P$ must contain at least one starting edge and one ending edge. For every $3 \leq \alpha \leq k - 2$, if $v_\alpha$ is an element-node, then there is a set node $u$ such that there are at least 2 edges on $P$ containing both $v_\alpha$ and $u$ and no other set-node. Since for every $v \in \{x_{\text{start}}, x_{\text{end}}\}$
and for every set node $u$, there is exactly one edge in $H$ that contains both $v$ and $u$ and no other set-node, it follows that $\{x_{start}, x_{end}\} = \{v_2, v_{k-1}\}$. Without loss of generality, $x_{start} = v_2$ and $x_{end} = v_{k-1}$. Since every base hyperedge containing $x_{start}$ (resp. $x_{end}$) must contain $u_{start}$ (resp. $u_{end}$), it follows that $v_1 = u_{start}$ and $v_k = u_{end}$.

In what follows, we show that all element nodes of a given set appear consecutively along the path, and moreover, every two sets whose nodes are visited by the path are, in fact, disjoint.

**Lemma 14.** Let $i \in I_P, j \in [m]$ and let $\alpha \in [k]$ be such that $v_\alpha = u^j_i$. Then $v_{\alpha + 1} = x^j_i$ and if $j \geq 2$ then $v_{\alpha - 1} = x^{j-1}_i$.

**Proof.** First note that since $v_\alpha = u^j_i$ then $r \leq \alpha \leq k - r + 1$. We prove the claim by induction on $\alpha$. If $\alpha = r$ then the edge $e = \{v_\ell\}_{\ell = 1}^r$ is a starting edge, and therefore $j = r - 2$. By Claim 10 every light edge that contains $u^j_i - 2$ must also contain $x^{r-2}_i$ and $x^{r-3}_i$. The only two nodes contained in all light edges in $P$ that contain $v_\alpha$ are $v_{\alpha - 1}$ and $v_{\alpha + 1}$, it follows that $\{x^j_i, x^{j-1}_i\} = \{v_{\alpha - 1}, v_{\alpha + 1}\}$. Since $e$ is a heavy starting edge, $x^{r-2}_i \notin e$. Therefore $v_{\alpha - 1} = x^{r-3}_i$ and $v_{\alpha + 1} = x^{r-2}_i$.

Assume therefore that the claim holds for all $r \leq \alpha' < \alpha$, and let $\beta = \alpha - (r - 1) \geq r$. Then $\beta$ is also a set node, and by the induction hypothesis, the claim holds for $v_\beta$. Let $i' \in [m]$ and $j' \in [|S_{i'}|]$ be such that $v_\beta = u^{j'}_{i'}$. If $i = i'$ then $e = \{v_\ell\}_{\ell = \beta}^r$ is a heavy internal edge containing $u^{j'}_{i'}, u^j_i$. By the induction hypothesis, $x^{j'+1}_{i'} = v_{\beta + 1} \in e$. Therefore by the construction of internal edges, $j = j' + r - 2 \geq 2$, and $x^{j-1}_i \notin e$ and $x^j_i \notin e$. By arguments similar to the first part of the proof we get that $\{x^j_i, x^{j-1}_i\} = \{v_{\alpha - 1}, v_{\alpha + 1}\}$, and therefore $v_{\alpha - 1} = x^{j-1}_i$ and $v_{\alpha + 1} = x^j_i$.

Otherwise, if $i \neq i'$, then $e = \{v_\ell\}_{\ell = \beta}^r$ is a heavy transition edge containing $u^{j'}_{i'}, u^j_i$. Again, by the induction hypothesis, $v_{\beta + 1} = x^{j'+1}_{i'}$. By the construction of transition hyperedges we get that $i < i'$ and $S_{i'} \cap S_i = \emptyset$. If $j \geq 2$ then in addition $x^{j-1}_i \in e$, and by arguments similar to the first part of the proof $v_{\alpha - 1} = x^{j-1}_i$ and $v_{\alpha + 1} = x^j_i$. If $j = 1$ then $e' = \{v_\ell\}_{\ell = \beta + 1}^r$ is a transition edge such that $x^1_i \in e'$. Since $x^1_i \notin e$ we get that $v_{\alpha + 1} = x^1_i$.

The following follows from the previous lemma by induction.

**Proposition 15.** Let $i \in I_P, j \in [|S_i|]$ and let $\alpha \in [k]$ be such that $u^j_i = v_\alpha$. Then

1. For every $1 \leq \ell \leq \min\{r - 2, j - 1\}$, $v_{\alpha - \ell} = x^{j-\ell}_i$;

2. For every $1 \leq \ell \leq \min\{r - 2, |S_i| - j + 1\}$, $v_{\alpha + \ell} = x^{j+\ell-1}_i$.

Moreover, if $j - 1 \geq r - 2$ then $v_{\alpha - r + 1} = u^{j+r-1}_i$ and if $|S_i| - j + 1 \geq r - 2$ then $v_{\alpha + r - 1} = u^{j-r-2}_i$.

**Corollary 16.** Let $i \in I_P$, then for every $x, x' \in S_i$ the only nodes between $x$ and $x'$ on $P$ are either element-nodes or set-nodes that are associated with $S_i$.

We are now ready to prove Lemma 3.

**Proof of Lemma 3.** We will first show that $\bigcup_{i \in I_P} S_i = X$. Let $x \in X$. Then there is some $\alpha \in [k]$ such that $x = v_\alpha$. Let $\beta < \alpha < \gamma$ be such that $v_\beta$ and $v_\gamma$ are the set-nodes preceding and following $v_\alpha$ in $P$ respectively. Then $x \in \{v_\ell\}_{\ell = \beta}^\gamma$. By the construction of $H$, there exist
\(i \in [m]\) and \(j, j' \in [|S_i|]\) such that either \(v_\beta\) or \(v_\gamma\) is \(u^i_j\) and \(x = x^{j'}_{i}\). Since \(u^i_j \in P\) then \(i \in \mathcal{I}_P\) and therefore \(x \in \bigcup_{i \in \mathcal{I}_P} S_i\).

Next, let \(i, i' \in \mathcal{I}_P\) be such that \(i \neq i'\). Then there exist \(j \in [|S_i|]\) and \(j' \in [|S_{i'}|]\) such that \(u^i_j, u'^{i'}_{j'} \in P\). That is, there exist \(\alpha, \alpha' \in [k]\) such that \(v_\alpha = u^i_j\) and \(v_{\alpha'} = u'^{i'}_{j'}\). Without loss of generality, we may assume \(\alpha < \alpha'\), and there are no set-nodes associated with \(S_i\) between \(v_\alpha\) and \(v_{\alpha'}\) (otherwise we take the last set-node associated with \(S_i\) before \(v_{\alpha'}\)). Then \(e = \{v_\ell\}_{\ell=\alpha}^{\alpha+r}\) is a heavy transition hyperedge. If \(\alpha' = \alpha + r\) then by definition of transition hyperedges, \(S_i \cap S_{i'} = \emptyset\). Otherwise, none of the element nodes in \(e\) are associated with \(S_{i'}\), and moreover, \(x^i_{|S|} \in e\). From Corollary \(\Box\) it follows that the elements of \(S_i\) are visited by \(P\) in the order \(x^i_1, x^i_2, \ldots, x^i_{|S_i|}\) and similarly the elements of \(S_{i'}\) are visited in the order \(x^{i'}_1, x^{i'}_2, \ldots, x^{i'}_{|S_{i'}|}\). Since \(x^i_1\) is visited by \(P\) after \(x^i_{|S_i|}\) then we get that \(S_i \cap S_{i'} = \emptyset\). \(\Box\)

We have therefore proved that if there is a tight path of length \(k\) in \(H\), then the Exact Cover is a ‘yes’ instance. Conversely, assume that the Exact Cover instance is a ‘yes’ instance and let \(S_C = \{S_1, \ldots, S_m\}\) be an exact cover. Since \(S_C\) is an exact cover, for every \(x \in X\) there exist a unique pair \(i \in [t], j \in [|S_i|]\) such that \(x = x^i_j\). We can therefore order the elements of \(X\) according to a lexicographic ordering. That is for all \(i, i' \in [s], j \in [|S_i|]\) and \(j' \in [|S_{i'}|]\), \(x^i_j\) comes before \(x^{i'}_{j'}\) if and only if \(i < i'\) or \(i = i'\) and \(j < j'\). Consider the following sequence of element-nodes.

\[
x_{\text{start}} - x^i_1 - \ldots - x^i_{|S_i|} - x^i_2 - \ldots - x^i_{|S_i|} - x^i_3 - \ldots - x^i_{|S_i|} - x_{\text{end}}.
\]

Between every \(r - 2\) element-nodes, insert the corresponding set-node. That is, if the element-node that follows the set-node is \(x^i_j\), then the set-node is \(u^i_j\). Add \(u_{\text{start}}, u_{\text{end}}\) in the beginning and end of the sequence respectively. The following claim is proved similarly to the claims used to prove the previous direction.

**Claim 17.** The sequence constructed above constitutes a tight path of length \(k\) in \(H\).

We have therefore proved that there exists an exact cover if and only if there is a tight path of length \(k\) in \(H\), thus completing the proof of Theorem \(\Box\).

**Discarding Assumption \(\Box\).** Given an Exact Cover instance \(X = \{x_1, \ldots, x_n\}, S = \{S_1, \ldots, S_m\}\), we can augment \(X\) by \(k\) new elements \(x_{n+1}, \ldots, x_{n+k}\) for some \(4r \leq k \leq 5r\) such that \(r - 2\) divides \(n + k + 2\). Call the new set \(X'\). We then create \(2r\) collections of subsets \(S_1, \ldots, S_{2r}\) by defining \(S_\ell := S \cup \{x_{n+1}, \ldots, x_{n+\ell}\}, \{x_{n+\ell+1}, \ldots, x_{n+k}\}\) for every \(\ell \in [2r]\). Note that \(m \leq |S_\ell| \leq m + 2r\). Next, for every \(\ell \in [2r]\) we define a collection \(S'_\ell\) as follows. For every \(2r\) pairwise-disjoint sets in \(S_\ell\), we add their union to \(S'_\ell\). The following is straightforward.

**Claim 18.** Given \(X, S\) and \(\ell \in [2r]\)

1. we can construct \(X', S'_\ell\) in time \(O(m^{2r+1})\);
2. for every \(\ell \in [2r]\) and every set \(S \in S_\ell\) we get that \(|S| \geq 2r\); and
3. \(X', S'_\ell\) is a ‘yes’ instance for Exact Cover if and only if \(X, S\) is a ‘yes’ instance for Exact Cover and has an exact cover of size \(2rt - \ell\) for some positive integer \(t\).
Therefore \( X, S \) is a 'yes' instance for Exact Cover if and only if there exists \( \ell \in [2r] \) such that \( X', S' \) is a 'yes' instance for Exact Cover. Performing the reduction to \( k\)-HyperPath on every instance separately we get a reduction of a general Exact Cover instance to \( 2r \) \( k\)-HyperPath instances.

4 Algorithm for \( k\)-HyperPath

In this section we show that \( k\)-HyperPath can be solved in time \( 2^{kn^{O(1)}} \), thus proving Theorem 4. We follow the ideas of Koutis [Kou08] and of Williams [Wil09], with additional ideas to take care of our generalization. A short introduction to arithmetic circuits follows.

Arithmetic Circuits An arithmetic circuit over a specified ring \( K \) is a directed acyclic graph with nodes labeled from \( \{+,\times\} \cup \{x_1,\ldots,x_n\} \cup K \), where \( \{x_1,\ldots,x_n\} \) are the input variables of the circuit. Nodes with zero out-degree are called output nodes, and nodes with labels from \( \{x_1,\ldots,x_n\} \cup K \) are called input nodes. The size of the circuit is the number of nodes in it. Clearly, every output node can be associated with a polynomial on \( K[x_1,\ldots,x_n] \). A polynomial \( p \in K[x_1,\ldots,x_n] \) is said to contain a multilinear term if it contains a term of the form \( c \prod_{i \in S} x_i \) for a non-empty \( S \subset [n] \) in the standard monomial expansion of \( p \). See [BCS07] for more details on arithmetic circuits.

The Algorithm Given a hypergraph \( H \), we define a polynomial \( P_k(X) \) on the set of variables \( X = \{ x_u : u \in V(H) \} \).

\[
P_k(x_1,\ldots,x_n) = \sum_{i_1,\ldots,i_k \text{ is a tight walk in } H} x_{i_1} \cdots x_{i_k},
\]

where a tight walk is a sequence of nodes (not necessarily distinct) such that every \( r \) consecutive nodes are contained in some edge in \( H \). Clearly, there is a \( k\)-hyperpath iff \( P_k(x_1,\ldots,x_n) \) contains a multilinear term. We first show that \( P_k \) can be succinctly implemented by an arithmetic circuit.

Lemma 19. Let \( H = (V_H, E_H) \) be a hypergraph on \( n \) nodes and \( m \) hyperedges, and let \( k \leq n \) be an integer. Then \( P_k(x_1,\ldots,x_n) \) is computable by an arithmetic circuit of size \( O(mnk) \).

For every hyperedge \( (u_{i_1}, u_{i_2},\ldots,u_{i_r}) \) and an integer \( t \in [n] \) we define a polynomial \( f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t) \). Intuitively, \( f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t) \) contains a sum of monomials where each one corresponds to all tight walks of length \( t \) that end with \( u_{i_1}, u_{i_2},\ldots,u_{i_r} \) and on the same set of nodes. Formally, \( f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t) \) can be defined inductively as follows. At the base level of the induction, for every edge \( (u_{i_1}, u_{i_2},\ldots,u_{i_r}) \in E_H \) we initialize \( f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t) = x_{i_1} \cdots x_{i_r} \) for \( t = r \), and \( f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t) = 0 \) for \( t < r \). Now, \( f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t+1) \) can be constructed by knowing \( f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t) \) for all edges \( (u_{i_1}, u_{i_2},\ldots,u_{i_r}) \), as follows.

\[
f_H(u_{i_1}, u_{i_2},\ldots,u_{i_r},t+1) = \sum_{u_{i_0} : (u_{i_0},u_{i_1},\ldots,u_{i_r-1}) \in E_H} f_H(u_{i_0}, u_{i_1},\ldots,u_{i_{r-1}},t) \cdot x_{i_r}
\]
Finally, $\tilde{P}_k$ which is the required polynomial, sums over all hyperedges $e$ the length-$k$ tight walks that end with $e$.

$$\tilde{P}_k(x_1, \ldots, x_n) = \sum_{(u_{i_1}, u_{i_2}, \ldots, u_{i_r}) \in E_H} f_H(u_{i_1}, u_{i_2}, \ldots, u_{i_r}, k)$$

The size of the $\tilde{P}_k$ is bounded by $1$ plus the number of possible labellings of the form $(u_{i_1}, u_{i_2}, \ldots, u_{i_r})$ as an input to $f_H$, which is bounded by $O(mk)$ since $f_H$ is only defined where the first $r$ entries are an edge in $E_H$. We use the following result by [Wil09] to achieve our bound.

**Theorem 20** (see Theorem 3.1 in [Wil09]). Let $P(x_1, \ldots, x_n)$ be a polynomial of degree at most $k$, represented by an arithmetic circuit of size $s(n)$ with $+$ gates (of unbounded fan-in), $\times$ gates (of fan-in two) and no scalar multiplications. There is a randomized algorithm that on every $P$ runs in $2^k s(n)n^{O(1)}$ time, outputs yes with high probability if there is a multilinear term in the sum-product expansion of $P$, and always outputs no if there there is no multilinear term.

Since the degree of $\tilde{P}_k$ is clearly $k$, the time to construct the circuit is $2^k mn^{O(1)}$. By a simple inductive argument it can be proved that $f_H(u_{i_1}, \ldots, u_{i_r}, t)$ indeed contains a multilinear monomial of degree $t$ iff there exists a length-$t$ tight path that ends with $(u_{i_1}, \ldots, u_{i_r})$.

If we care about cycles, the only difference is that we also need to remember from what edge we have started the path, thus adding to $f_H$ an additional amount of $r$ entries, and the circuit size would increase to $O(m^2k)$, and the running time to $2^k m^2 n^{O(1)}$, thus proving Theorem 4.

### 5 Conclusion and Open Problems

We have proved that $k$-HyperPath could be solved in time $2^k mn^{O(1)}$ and hard to solve significantly faster even for bounded $r$’s. It remains open to show for undirected $k$-HyperPath an algorithm with running time $O^*(2^{\gamma(r)k})$ (i.e. when $\gamma$ is a function of $r$). It is unclear how to extend the current fastest algorithms for undirected Hamiltonian path or $k$-Path to hypergraphs, and it may even be possible that a stronger $O^*(2^k)$ conditional lower bound holds for $k$-HyperPath even for $r = 3$. Also, an important open problem is to prove any conditional lower bound with explicit constant in the exponent to $k$-Path. This task has been hard to complete, maybe because a (graphic) path does not seem to have enough information to make manipulations needed to solve Set Cover. While $k$-HyperPath is a stronger variant, it still appears close to $k$-Path and shows a possible direction to prove conditional lower bounds to this problem.

### References

[ABHS19] A. Abboud, K. Bringmann, D. Hermelin, and D. Shabtay. Seth-based lower bounds for subset sum and bicriteria path. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 41–57, 2019.
[AYZ95] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995. doi:10.1145/210332.210337

[BCS97] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997.

[BHH15] A. Björklund, D. Holger, and T. Husfeldt. The parity of set systems under random restrictions with applications to exponential time problems. In *42nd International Colloquium on Automata, Languages and Programming (ICALP 2015)*, volume 9134, pages 231–242. Springer, 2015. doi:10.1007/978-3-662-47672-7_13

[BHK09] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, July 2009. doi:10.1137/070683933

[BHPK17] A. Björklund, T. Husfeldt, K. Pteri, and M. Koivisto. Narrow sieves for parameterized paths and packings. *Journal of Computer and System Sciences*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003

[BKK16] A. Björklund, P. Kaski, and Ł. Kowalik. Constrained multilinear detection and generalized graph motifs. *Algorithmica*, 74(2):947–967, February 2016. doi:10.1007/s00453-015-9981-1

[Bod93] H. L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, January 1993. doi:10.1006/jagm.1993.1001

[CDL+16] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):41:1–41:24, May 2016. doi:10.1145/2925413

[CLSZ07] J. Chen, S. Lu, S.-H. Sze, and F. Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’07, pages 298–307, 2007. Available from: http://dl.acm.org/citation.cfm?id=1283383.1283415

[DFRS17] A. Dudek, A. M. Frieze, A. Rucinski, and M. Sileikis. Embedding the erdős-rényi hypergraph into the random regular hypergraph and hamiltonicity. *J. Comb. Theory, Ser. B*, 122:719–740, 2017. Available from: https://doi.org/10.1016/j.jctb.2016.09.003

[FKL11] A. Frieze, M. Krivelevich, and P.-S. Loh. Packing tight hamilton cycles in 3-uniform hypergraphs. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’11, pages 913–932, 2011. Available from: http://dl.acm.org/citation.cfm?id=2133036.2133107

[FKW04] F. V. Fomin, D. Kratsch, and G. J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *30th International Conference on Graph-Theoretic Concepts in Computer Science*, WG’04, pages 245–256. Springer-Verlag, 2004. doi:10.1007/978-3-540-30559-0_23

[GM16] F. Garbe and R. Mycroft. The complexity of the hamilton cycle problem in hypergraphs of high minimum codegree. In *33rd Symposium on Theoretical Aspects
of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France, pages 38:1–38:13, 2016. Available from: https://doi.org/10.4230/LIPIcs.STACS.2016.38.

[HZ15] J. Han and Y. Zhao. Minimum codegree threshold for hamilton &amp;#8467;-cycles in k-uniform hypergraphs. J. Comb. Theory Ser. A, 132(C):194–223, May 2015. Available from: http://dx.doi.org/10.1016/j.jcta.2015.01.004.

[KL16] Ł. Kowalik and J. Lauri. On finding rainbow and colorful paths. Theoretical Computer Science, 628(C):110–114, May 2016. doi:10.1016/j.tcs.2016.03.017.

[KMRR06] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. Divide-and-color. In Proceedings of the 32Nd International Conference on Graph-Theoretic Concepts in Computer Science, WG’06, pages 58–67, 2006. doi:10.1007/11917496_6.

[Koi09] M. Koivisto. Parameterized and exact computation. chapter Partitioning into Sets of Bounded Cardinality, pages 258–263. Springer-Verlag, 2009. doi:10.1007/978-3-642-00136-9_21.

[Kou08] I. Koutis. Faster algebraic algorithms for path and packing problems. In 42nd International Colloquium on Automata, Languages and Programming (ICALP 2015), volume 5125, pages 575–586, 2008. doi:10.1007/978-3-540-70575-8_47.

[KST17] R. Krithika, A. Sahu, and P. Tale. Dynamic parameterized problems. In 11th International Symposium on Parameterized and Exact Computation (IPEC 2016), volume 63 of Leibniz International Proceedings in Informatics (LIPIcs), pages 19:1–19:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.IPEC.2016.19.

[KT17] R. Krauthgamer and O. Trabelsi. The set cover conjecture and subgraph isomorphism with a tree pattern. CoRR, abs/1711.08041, 2017. Available from: http://arxiv.org/abs/1711.08041 arXiv:1711.08041.

[KW16] I. Koutis and R. Williams. Limits and applications of group algebras for parameterized problems. ACM Transactions on Algorithms, 12(3):31:1–31:18, May 2016. doi:10.1145/2885499.

[LWW18] A. Lincoln, V. V. Williams, and R. Williams. Tight hardness for shortest cycles and paths in sparse graphs. In Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’18, pages 1236–1252, 2018.

[Mon85] B. Monien. How to find long paths efficiently. Annals of Discrete Mathematics, 25:239–254, 1985. Available from: https://ci.nii.ac.jp/naid/10013137275/ en/.

[Ned16] J. Nederlof. Finding large set covers faster via the representation method. In 24th Annual European Symposium on Algorithms (ESA 2016), volume 57 of Leibniz International Proceedings in Informatics (LIPIcs), pages 69:1–69:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.69.
[Wil09] R. Williams. Finding paths of length $k$ in $O^*(2^k)$ time. *Information Processing Letters*, 109(6):315–318, February 2009. doi:10.1016/j.ipl.2008.11.004.