Paths, Cycles and Sprinkling in Random Hypergraphs

Oliver Cooley

Submitted: Apr 8, 2021; Accepted: Jan 30, 2025; Published: Aug 22, 2025

Abstract

We prove a lower bound on the length of the longest j-tight cycle in a k-uniform binomial random hypergraph for any $2 \leqslant j \leqslant k-1$. We first prove the existence of a j-tight path of the required length. The standard "sprinkling" argument is not enough to show that this path can be closed to a j-tight cycle – we therefore show that the path has many extensions, which is sufficient to allow the sprinkling to close the cycle.

Mathematics Subject Classifications: 05C80, 05C65, 05C38

1 Introduction

1.1 Paths and cycles in random graphs

Over the years there has been a considerable amount of research into the length of the longest paths and cycles in random graphs. This goes back to the work of Ajtai, Komlós and Szemerédi [1], who showed that in the Erdős-Rényi binomial random graph $G(n,p)$, the threshold $p = 1/n$ for the existence of a giant component is also the threshold for a path of linear length. In the supercritical regime, a simple first moment argument shows that whp[1] there are no pairs of sets of linear size with no edges between them, which implies that the lengths of the longest path and the longest cycle are asymptotically the same, and therefore whp $G(n,p)$ also contains a cycle of linear length. This has been strengthened by various researchers, including Łuczak [14], and Kemkes and Wormald [13], in particular to the *weakly supercritical regime* when $p = (1+\varepsilon)/n$ for some $0 \leqslant \varepsilon = \varepsilon(n) \xrightarrow{n\to\infty} 0$ which does not tend to zero too quickly. In this regime the crude first moment argument mentioned above is no longer sufficient, but a still standard *sprinkling argument* nevertheless shows that the longest path and longest cycle have asymptotically the same length.

We note, however, that in this regime the asymptotic length of the longest cycle is still not known precisely: The best known lower and upper bounds are approximately $4\varepsilon^2 n/3$

OJNCooley@gmail.com

[1] *with high probability*, meaning with probability tending to 1 as n tends to infinity.

(see [14]) and $1.7395\varepsilon^2 n$ (see [13]) respectively. Very recently, Anastos [2] announced an improvement on the lower bound to $1.581\varepsilon^2 n$, but this still falls short of the upper bound. On the other hand, Anastos and Frieze [3] determined the asymptotic length of the longest cycle precisely when $p = d/n$ for any sufficiently large constant d.

A similar problem, although one requiring very different techniques, is to determine the length of the longest *induced* cycle, which was achieved recently by Draganić, Glock and Krivelevich [9] in the regime when $p \geqslant d/n$ for some sufficiently large constant d.

1.2 Paths and cycles in random hypergraphs

Given an integer $k \geqslant 2$, a k-uniform hypergraph consists of a set V of vertices and a set $E \subset \binom{V}{k}$ of edges. (A 2-uniform hypergraph is simply a graph.) Among the many possible definitions of paths and cycles in hypergraphs, perhaps the most natural and well-studied is that of *j-tight paths and cycles*, which is in fact a family of definitions for $1 \leqslant j \leqslant k-1$.

Definition 1. Given integers $1 \leqslant j \leqslant k-1$ and a natural number ℓ, a *j-tight path of length ℓ* in a k-uniform hypergraph consists of a sequence of distinct vertices $x_1, \ldots, x_{j+(k-j)\ell}$ and a sequence of edges $e_1, \ldots, e_\ell$ such that $e_i = \{x_{(k-j)(i-1)+1}, \ldots, x_{(k-j)(i-1)+k}\}$ for $1 \leqslant i \leqslant \ell$.

A *j-tight cycle of length ℓ* is defined identically except that $x_i = x_{(k-j)\ell+i}$ for $1 \leqslant i \leqslant j$ (and otherwise all vertices are distinct), see Figure 1.

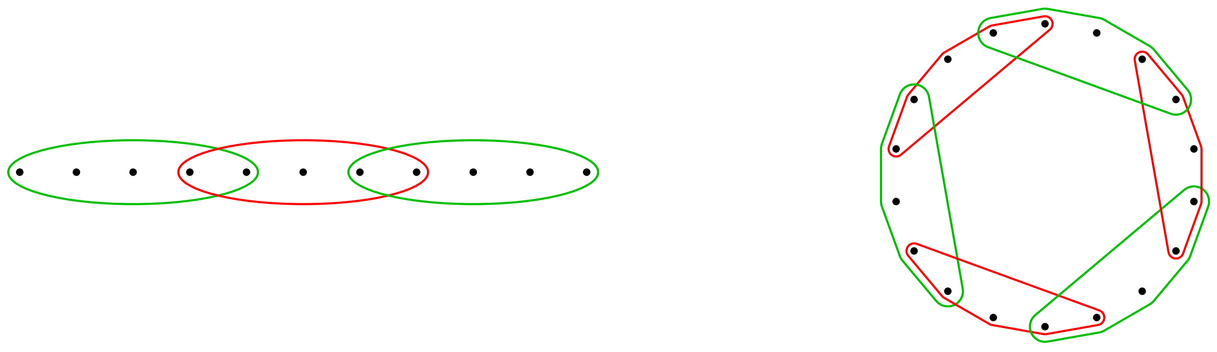

Figure 1: A 2-tight path of length 3 and a 2-tight cycle of length 6 in a 5-uniform hypergraph.

In the literature, 1-tight paths/cycles are often called *loose* paths/cycles, while $(k-1)$-tight is often abbreviated simply to tight.

Let $\mathcal{H}^k(n,p)$ denote the k-uniform binomial random hypergraph, in which each k-set of vertices forms an edge with probability p independently. The analogue of the result of Ajtai, Komlós and Szemerédi showing a threshold for the existence of a j-tight path of linear length in $\mathcal{H}^k(n,p)$ was proved by the author together with Garbe, Hng, Kang, Sanhueza-Matamala and Zalla [4] for all k and j. Interestingly, in contrast to the graph case, in general the threshold is *not* the same as the threshold for a giant j-tuple component (which was determined in [7]).

At the other end of the scale, one can ask when a j-tight cycle of the longest possible length appears, namely a j-tight Hamilton cycle. Dudek and Frieze determined an asymptotic threshold for loose [10] and the sharp threshold (when $k \geqslant 4$) for tight [11] Hamilton cycles (i.e. the cases $j = 1$ and $j = k-1$), along with some further related results. Recently, Narayanan and Schacht [15] extended this by proving the sharp threshold for all $2 \leqslant j \leqslant k-1$ (i.e. all non-loose Hamilton cycles). We are not aware of any previous work looking at lengths of j-tight paths or cycles in random hypergraphs between very small linear length and Hamiltonian.

Let a be the unique integer satisfying $1 \leqslant a \leqslant k-j$ and $a \equiv k \mod k-j$, and let $p_0 = p_0(n,k,j) := \frac{1}{\binom{k-j}{a}\binom{n}{k-j}}$. The results of [4] show that p_0 is a threshold for the existence of a j-tight path of linear length in $\mathcal{H}^k(n,p)$. This can be heuristically justified by using a branching process to approximate a natural search process to discover j-tight paths and observing that p_0 is exactly the edge probability at which this branching process is critical. While conceptually similar to the analogous heuristic for graphs, the argument for general j and k is slightly more involved – see Section 2.3 for details. Furthermore, in the case when $p = (1+\varepsilon)p_0$ for some constant $\varepsilon > 0$, upper and lower bounds on the length of the longest j-tight path were proved in [4]. In the case when $j \geqslant 2$, these bounds are $\Theta(\varepsilon n)$ and differ by a factor of 8. In the case when $j = 1$, the lower bound was $\Theta(\varepsilon^2 n)$ while the upper bound was $\Theta(\varepsilon n)$.

This upper bound in the case when $j = 1$ was subsequently improved by the author, Kang and Zalla [8] and shown to be $\Theta(\varepsilon^2 n)$ in the range when $p = (1+\varepsilon)p_0$ (although the results of that paper also cover the range $p = d/n$ for any constant $d > 1$). The strategy used was to prove an upper bound on the length of the longest loose *cycle* which transfers to loose paths using a standard sprinkling argument, just as has often been observed for graphs.

The standard argument for graphs which shows that the longest path and longest cycle have approximately the same length proceeds roughly as follows. We reveal the edges of the random graph in two independent rounds, first with edge probability just a little smaller than p, after which we can find a path of asymptotically the desired length. Subsequently, we expose the remaining edges, now with much smaller edge probability (informally, we *sprinkle* a few more random edges onto the graph), and seek an edge between vertices near the start and near the end of the path obtained in the first round, which will close the path into a cycle of almost the same length. If we select the parameters carefully, such an edge is very likely to exist. (A cruder argument, but which is still sufficient in many cases, is simply to reveal all of the random edges in one round and to observe, via a union bound, that with high probability there are no pairs of large sets with no edges between them. In particular, if there is a long path then whp there is also an edge from near the start to near the end of the path.)

The natural hypergraph analogue of this argument for loose paths and cycles was used in [8], and can also be used to extend the lower bound on loose paths from [4] to an asymptotically identical lower bound for loose cycles.

1.3 Sprinkling in hypergraphs

This raises an obvious question: can we also use the sprinkling technique for $j \geqslant 2$, and obtain a j-tight cycle from a j-tight path without significantly decreasing the length? Unfortunately, the naive approach does not work.

To see why, let us outline the obvious generalisation of the standard argument, first in the case $j \leqslant k/2$ when we have $p = \Theta\left(n^{-(k-j)}\right)$ and a path of length $\Theta(n)$. Now for some $\omega \to \infty$ arbitrarily slowly, sprinkle additional edges with probability of p/ω. We can identify n/ω many j-sets of vertices from the start and from the end of the path with which we attempt to close to a cycle of almost the same length as the path[2], and we need a further $k - 2j$ vertices from outside the cycle to complete an edge. Thus the number of potential edges which would close the cycle is $\Theta\left((n/\omega)^2 n^{k-2j}\right)$, and the expected number of suitable edges we find is

$$\frac{p}{\omega} \cdot \Theta\left(\frac{n^{k-2j+2}}{\omega^2}\right) = \Theta\left(\frac{n^{2-j}}{\omega^3}\right).$$

This will clearly be enough if $j = 1$ and if ω tends to infinity sufficiently slowly, but for $j \geqslant 2$ the argument fails. Indeed, for $j > k/2$, the situation becomes even worse: Here we even need more than one edge in order to be able to close the path to a cycle.

The essential reason why this standard version of the argument no longer works stems from the interplay between the j-sets and the vertices: A j-tight path "lives" on vertices, but is extended (or closed to a cycle) via j-sets. The number of j-sets within the path is naturally bounded by $\Theta(n)$, but this is tiny compared to the number of j-sets in the world (namely $\binom{n}{j}$).

1.4 Main result

The main contribution of this paper is to provide a variant of the sprinkling argument which does work for $j \geqslant 2$ by strengthening the conclusions drawn in the first round of exposure. In particular, we provide a search algorithm which whp will construct a long j-tight cycle in $\mathcal{H}^k(n,p)$. We thus provide a lower bound on the length of the longest j-tight cycle.

Let $L_C = L_C(n,p,k,j)$ be the random variable denoting the length of the longest j-tight cycle in $\mathcal{H}^k(n,p)$.

Theorem 2. *Let $k, j \in \mathbb{N}$ satisfy $2 \leqslant j \leqslant k-1$ and let a be the unique integer satisfying $1 \leqslant a \leqslant k-j$ and $a \equiv k \mod k-j$. Let $p_0 = p_0(n,k,j) := \frac{1}{\binom{k-j}{a}\binom{n}{k-j}}$.*

For any $\delta > 0$, for any constant $d > 1$ and for any sequence $(d_n)_{n\in\mathbb{N}}$ satisfying $d_n \to d$ the following is true. Suppose that $p = d_n p_0$. Then whp

$$L_C \geqslant (1-\delta) \cdot \frac{1 - d^{-1/(k-j)}}{k-j} \cdot n.$$

[2] These j-sets are chosen from among those j-sets at which a subpath can end. From the structure of paths, it is not hard to verify that a path of length ℓ admits $\Theta(\ell)$ such potential ends – the factor of ω ensures that we only consider those subpaths which contain almost all of the edges of the original path.

Note that it is trivially true that $L_P \geqslant L_C - O(1)$, where L_P denotes the length of the longest j-tight path in $\mathcal{H}^k(n,p)$. Therefore an immediate corollary of this result is a lower bound on L_P which generalises the one in [4] so that it is applicable for a larger range of p.[3] Having said that, the *proof* of Theorem 2 first requires us to prove such a lower bound on L_P (see Lemma 6 later).

We note further that while for d close to 1 the bound matches the lower bound for paths provided by [4], for larger d this is far weaker than already known results. Specifically, the aforementioned result of Narayanan and Schacht [15] states that for $d > e^{k-j}$, whp the hypergraph contains a Hamilton cycle. We discuss this result, and other open problems, in more detail in Section 7.

2 Preliminaries

2.1 Notation and terminology

In this section we introduce some notation and terminology, and fix various parameters for the rest of the paper.

Throughout the paper, let k, j be fixed natural numbers satisfying $2 \leqslant j \leqslant k-1$. In particular, for the rest of the paper we will usually simply refer to paths and cycles rather than j-tight paths and j-tight cycles, since j is understood.

All asymptotics in the paper are as $n \to \infty$, and in particular we will use the standard Landau notation $o(\cdot), O(\cdot), \Theta(\cdot)$ with respect to these asymptotics. We consider k, j to be constants, so for example a bound of $O(n)$ may have a constant that is implicitly dependent on k and j. Since our main result is asymptotic in n, we will always assume that n is sufficiently large. In particular we frequently make approximations which are only valid for large n.

Let us further define the following parameters. As in Theorem 2, let $a = a(k,j)$ be the unique integer satisfying $1 \leqslant a \leqslant k-j$ and

$$a \equiv k \mod k-j.$$

The motivation for this parameter will become clear in Section 2.3. Given a natural number ℓ, let $v_\ell = v_\ell(k,j) := j + \ell(k-j)$ denote the number of vertices in a path of length ℓ. When $\ell = \Theta(n)$, observe that $v_\ell = (1+O(1/n))\ell(k-j)$, and therefore we will often use the approximation $\ell(k-j)$ in place of v_ℓ.

Let

$$p_0 = p_0(n,k,j) := \frac{1}{\binom{k-j}{a}\binom{n}{k-j}}$$

denote the threshold for a long tight path. Given $p = p(n) = d_n p_0$ for some sequence d_n of

[3]Note, however, that in [4] the range considered is $p = (1+\varepsilon)p_0$, where ε may be constant but is also allowed to tend to zero sufficiently slowly, which is *not* allowed in Theorem 2 since we assume $d > 1$. Therefore Theorem 2 is not strictly stronger.

positive real numbers which tend to $d > 1$, let

$$L_1 = L_1(p) := \frac{1 - d^{-1/(k-j)}}{k-j} \cdot n. \tag{1}$$

To interpret this notation, observe that the parameters n, d, k, j are implicit in p and will be clear from the context. Further, let $L_C = L_C(n, p, k, j)$ denote the length of the longest j-tight cycle in $\mathcal{H}^k(n, p)$.

Throughout the paper we use the notation $a \ll b$ to mean that there exists some large constant C such that $a \leqslant b/C$, and $b \gg a$ is defined identically. The constant C must be chosen sufficiently large for subsequent approximations to be valid, but in general we do not calculate the required dependencies explicitly.

Remark 3. Let us observe that Theorem 2 becomes stronger for smaller δ, and therefore we may (and for technical convenience frequently do) assume that $\delta \ll 1$. Indeed, we may even assume that δ is sufficiently small as a function of d – more precisely, given any function $f : (1, \infty) \to (0, \infty)$, we may assume that $\delta \leqslant f(d)$.

For an integer m, we denote $[m] := \{1, \dots, m\}$ and $[m]_0 := [m] \cup \{0\}$. We omit floors and ceilings when this does not significantly affect calculations.

2.2 Chernoff bound

We will frequently use the following version of the Chernoff bound (see e.g. [12, Corollary 2.3]).

Lemma 4. *Let $N \in \mathbb{N}$, let $q \in [0, 1]$ and let $\alpha > 0$ be a real number. Suppose that $X \sim \mathrm{Bin}(N, q)$. Then*

$$\mathbb{P}\left(|X - Nq| \geqslant \alpha Nq\right) \leqslant 2 \exp\left(\frac{-\alpha^2 Nq}{3}\right).$$

In particular, if $Nq \to \infty$ and $\alpha = \Theta(1)$, then $\mathbb{P}\left(|X - Nq| \geqslant \alpha Nq\right) = o(1)$.

2.3 The structure of paths

In graphs, there are only two paths with the same edge set (the second is obtained by reversing orientation), but depending on the values of k and j, there may be many ways of reordering the vertices of a j-tight path which give a different path with precisely the same edges. For example, in Figure 2, we may re-order x_1, x_2, x_3 arbitrarily. Even in the middle of the path, we may switch the order of x_8 and x_9 to give a new path. Nevertheless, we will identify paths which have the same set of edges, and indeed often identify a path with its edge set. We similarly identify cycles with their edge sets. Note that this is equivalent to identifying paths obtained by permuting sets of vertices which lie within precisely the same set of edges (such as $\{x_8, x_9\}$ in Figure 2), as well as potentially reversing the order of the edges, while for the identification of cycles we additionally allow an arbitrary choice of starting edge, which then determines the starting vertex within that edge. In both cases, since the edge set and the order of edges remains the same (up to the choice of starting point and direction), the overlap structure is not affected.

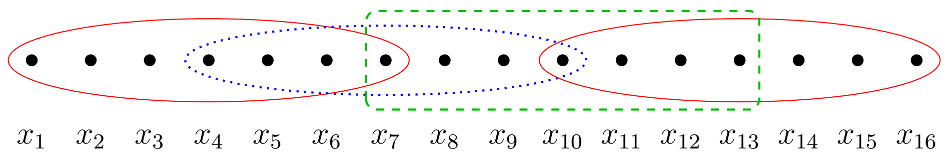
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}

Figure 2: A 4-tight path of length 4 in a 7-uniform hypergraph.

A further important point to note when constructing a path is which j-sets we can continue from: For example, in Figure 2, it seems natural to continue from the 4-set $\{x_{13}, \ldots, x_{16}\}$, but since the vertices x_{11}, x_{12}, x_{13} may be rearranged arbitrarily, we could just as well replace x_{13} by either of x_{11}, x_{12} in this 4-set.

To account for this, we will borrow the following terminology from [4]. Recall that a is defined to be the unique integer satisfying $1 \leqslant a \leqslant k-j$ and $a \equiv k \mod k-j$.

Definition 5. An *extendable partition* of a j-set J is an ordered partition $(C_0, C_1, \ldots, C_r)$ of J, where $r = \lceil \frac{j}{k-j} \rceil$-1, with $|C_0| = a$ and $|C_i| = k-j$ for all $i \in [r]$.

In the example in Figure 2, the 4-set $\{x_{10}, \ldots, x_{13}\}$ would have extendable partition (C_0, C_1), where $C_0 = \{x_{10}\}$ and $C_1 = \{x_{11}, x_{12}, x_{13}\}$. In a search process, the final edge $\{x_{10}, \ldots, x_{16}\}$ added to this path would give rise to three new 4-sets from which we can continue, namely $J_i := \{x_i, x_{14}, x_{15}, x_{16}\}$, where $i = 11, 12, 13$. The extendable partition of J_i would be $(C_0^{(i)}, C_1^{(i)})$, where $C_0^{(i)} = \{x_i\}$ and $C_1^{(i)} = \{x_{14}, x_{15}, x_{16}\}$.

For general k and j, when we discover an edge K from a j-set J with extendable partition $(C_0, \ldots, C_r)$, the new j-sets from which we can continue will — in the case when $j > k-j$, so we have $r \geqslant 1$ — be those consisting of a vertices from C_1, all vertices of $C_2, \ldots, C_r$ and all vertices of $K \setminus J$. In the case when $j \leqslant k-j$, the new j-sets from which we can continue are those consisting of a vertices of $K \setminus J$ (which would be equivalent to the previous case if, with slight abuse of notation, we were to interpret $K \setminus J$ as the otherwise non-existent C_1). Furthermore, the intersections of the new j-set with $C_1, C_2, \ldots, C_r, K \setminus J$ (in this order) will naturally form an extendable partition $\tilde{C}_0, \tilde{C}_1, \ldots, \tilde{C}_r$ of the new j-set, so there is a shift of indices. Note in particular that the number of new j-sets we can continue from is $\binom{|C_1|}{a} = \binom{k-j}{a}$, which is the reason this term appears in the threshold p_0 for a long path.

We will refer to the j-sets $\{x_1, \ldots, x_j\}$ and $\{x_{(k-j)\ell+1}, \ldots, x_{(k-j)\ell+j}\}$ as the *ends* of the path. With this definition, a path has precisely two ends, but recall that we often identify a path with its edge set, in which case there may be many potential ends (depending on the chosen order of vertices).

We refer the reader to [4] for a more detailed discussion of the structure of paths.

3 Proof outline

The initial, naive proof idea for Theorem 2 is to construct a long path using a search process, and then apply a sprinkling argument to close this path into a cycle. However, in its most basic form this argument fails for the reasons outlined in the introduction: We have too few potential attachment j-sets and too many required edges for the sprinkling to work.

Nevertheless, this will still be our overarching strategy; it just needs to be modified slightly. More precisely, we will aim to construct a *family* of long paths, all of which are identical along most of their length, but which diverge towards the two ends. This will give us many more potential attachment j-sets, and allow us to push the sprinkling argument through.

As such, we have two main lemmas in the proof. Let L_P denote the length of the longest path in $\mathcal{H}^k(n,p)$. Recall the definition of $L_1(p)$ from (1).

Lemma 6. *Under the conditions of Theorem 2, whp* $L_P \geqslant \left(1-\frac{\delta}{2}\right)L_1(p)$.

The proof of this lemma is essentially the same as the proof for the special case of $p=(1+\varepsilon)p_0$ in [4]. We first define an appropriate depth-first search process for constructing j-tight paths. Heuristically, this DFS is supercritical for as long as the path constructed has length significantly smaller than L_1. However, the algorithm will avoid re-using j-sets that have already been tried, if they led to dead-ends; to justify the heuristic, we need to know that this will not slow down the growth too much. For this, we need a *bounded degree lemma* which shows that, in an appropriate sense, these j-sets are evenly distributed in the hypergraph, rather than clustered together. The proof of Lemma 6 is given in Section 4.

The main original contribution of this work is the second lemma, which guarantees the existence of a family of j-tight paths with many different endpoints. The DFS algorithm is well-suited to creating long paths quickly, but in order to fan out towards the ends, we will switch to a *breadth*-first search algorithm. The result of this algorithm will be the structure described in Lemma 8, for which we first need the following definition.

Definition 7. Given an integer ℓ, two j-sets J, J' and a path P with end J, we say that a path P' *path-(J',ℓ)-augments* the pair (P,J) if P' has length at most ℓ and has ends J, J', and furthermore $P\cup P'$ is again a path with end J'. When J' is clear from the context, we will simply say that P' *path-ℓ-augments* the pair (P,J). We call P' the *augmenting path.*

In other words, P' extends the path P by length at most (but *not* necessarily exactly) ℓ to end at J' instead of J. Theorem 2 can be proved, with a bit of work, from the following lemma.

Lemma 8. *Under the conditions of Theorem 2, and in particular given $\delta>0$, there exists some constant $\varepsilon\in(0,1)$ such that the following holds. Whp $\mathcal{H}^k(n,p)$ contains a j-tight path P_0 of length $(1-\delta)L_1(p)$ with ends J_s, J_e, collections $\mathcal{A},\mathcal{B}$ of j-sets and collections of paths $\mathscr{P}_{\mathcal{A}}=\{P_{A,J_s}: A\in\mathcal{A}\}$ and $\mathscr{P}_{\mathcal{B}}=\{P_{B,J_e}: B\in\mathcal{B}\}$ such that:*

(i) $|\mathcal{A}|,|\mathcal{B}|=\varepsilon^2 n^j$;

(ii) Every path $P_{A,J_s} \in \mathscr{P}_{\mathcal{A}}$ path-$(A, \delta n/3)$-augments (P_0, J_s);

(iii) Every path $P_{B,J_e} \in \mathscr{P}_{\mathcal{B}}$ path-$(B, \delta n/3)$-augments (P_0, J_e);

(iv) For at least $(1-\varepsilon)\varepsilon^4 n^{2j}$ pairs $(A, B) \in \mathcal{A} \times \mathcal{B}$, the augmenting paths P_{A,J_s}, P_{B,J_e} are vertex-disjoint.

We will show how to prove Lemma 8 from Lemma 6 in Sections 5 and 6. Before continuing with the proofs of these two lemmas, we first show how Lemma 8 implies our main theorem.

Proof of Theorem 2. Let ω be some function of n tending to infinity arbitrarily slowly, and let $p' := (1 - 1/\omega)p$. We apply Lemma 8 with p' in place of p. Let us observe that $p' = d'_n p_0$, where $d'_n := (1 - 1/\omega)d_n \to d$, and therefore we have $L_1(p') = L_1(p)$. It follows that the path P_0 provided by Lemma 8 has length at least $(1 - \delta)L_1(p)$.

Now for each pair $(A, B) \in \mathcal{A} \times \mathcal{B}$ satisfying condition (iv) of Lemma 8, concatenating the paths $P_{A,J_s}, P_0, P_{B,J_e}$ gives a path $P_{A,B}$ with ends A and B and containing P_0. Thus $P_{A,B}$ certainly has length at least $(1 - \delta)L_1(P)$ (the length of P_0), but by (ii) and (iii) also has length at most $(1 - \delta)L_1(p) + 2\delta n/3 \leqslant (1 - \delta/3)n$. In other words, $P_{A,B}$ leaves a set $U_{A,B}$ of at least $\delta n/3 = \Theta(n)$ vertices uncovered.

Let us now sprinkle an additional probability of $p'' := p - p' = p/\omega$ onto the hypergraph. Formally, if our first random hypergraph was $\mathcal{H}' \sim \mathcal{H}^k(n, p')$, we construct a second, independent random hypergraph $\mathcal{H}'' \sim \mathcal{H}^k(n, p'')$ and let $\mathcal{H} := \mathcal{H}' \cup \mathcal{H}''$. Note that $\mathcal{H} \sim \mathcal{H}^k(n, p''')$, where $p''' := p' + p'' - p'p'' \leqslant p$. Since the existence of a cycle of a certain length is a monotonically increasing property, it suffices to prove this existence whp within $\mathcal{H}$.

We will assume for simplicity that $P_{A,B}$ induces total orders of the vertices within A and within B (in fact, it only induces partial orders related to the extendable partitions, but total orders are clearly more restrictive). In order to close $P_{A,B}$ to a cycle, we need to choose a set R of $b = k - j - a$ vertices from $U_{A,B}$ and the $s = \lceil \frac{j}{k-j} \rceil$ appropriate edges within $A \cup B \cup R$ must be present, see Figure 3.

Note that since each extending edge from B contains $k - j$ vertices not in the previous edge, s is the minimum number of edges required for the final j-set to be disjoint from B, which must be the case if this j-set is to be A; furthermore b intermediate vertices are required to hit A precisely modulo $k - j$. Note also that, due to our total order assumption on the vertices in A and in B, the s edges required to close the cycle are determined by the choice of the triple (A, B, R), where $(A, B) \in \mathcal{A} \times \mathcal{B}$ and $R \subset U_{A,B}$ has size b.

Given a choice of (A, B, R), define $E_{A,B,R}$ to be the set of s many k-sets which are required to be edges; the probability that these edges all exist is simply $(p'')^s$. Further, for fixed (A, B) but for different choices of R, the sets $E_{A,B,R}$ are all disjoint (since all edges of $E_{A,B,R}$ certainly contain R, but no other vertices of $U_{A,B}$). However, given two choices (A_1, B_1, R_1) and (A_2, B_2, R_2) for (A, B, R), it is possible that E_{A_1,B_2,R_1} and E_{A_2,B_2,R_2} contain a common k-set, and thus we no longer have independence. We therefore claim that there are sufficiently many choices for which the sets *are* disjoint.

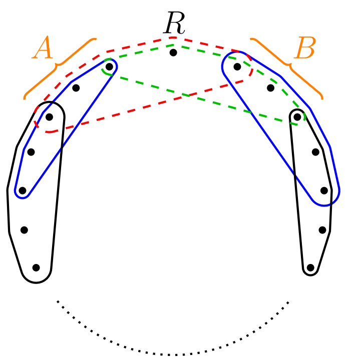

Figure 3: Closing a 3-tight path in a 5-uniform hypergraph to a cycle. The two dashed 5-sets would close the cycle if they both form edges.

To see this, observe that by conditions (i) and (iv) of Lemma 8 there are $\Theta\left(\varepsilon^4 n^{2j+b}\right)$ choices for the triple (A, B, R), and thus also for the sets $E_{A,B,R}$. Furthermore, any particular k-set lies in at most $O\left(n^{2j+b-k}\right)$ of the $E_{A,B,R}$, because a triple spans $2j+b$ vertices, of which k have already been fixed by the k-set. Since for any triple we have $|E_{A,B,R}| = s = O(1)$, any one of these sets $E_{A,B,R}$ shares a k-set with at most $O\left(n^{2j+b-k}\right)$ other sets, and we may greedily choose $\Theta\left(\varepsilon^4 n^{2j+b-(2j+b-k)}\right) = \Theta\left(\varepsilon^4 n^k\right)$ triples (A, B, R) such that the corresponding sets $E_{A,B,R}$ are disjoint.

By choosing this many triples, we observe that the probability that none of the corresponding $E_{A,B,R}$ do indeed form edges (and therefore close a cycle) is

$$(1-(p'')^s)^{\Theta\left(\varepsilon^4 n^k\right)} \leqslant \exp\left(-\Theta\left(\frac{\varepsilon^4 n^k}{\omega^s n^{s(k-j)}}\right)\right).$$

Now recall that $s = \lceil \frac{j}{k-j} \rceil \leqslant \frac{k-1}{k-j}$. Thus the probability that none of the choices of A, B, R admits the edges necessary to close a cycle is at most

$$\exp\left(-\Theta\left(\frac{\varepsilon^4 n}{\omega^{(k-1)/(k-j)}}\right)\right) = o(1),$$

where the last estimate follows since ω tends to infinity arbitrarily slowly, so in particular we have $\omega^{(k-1)/(k-j)} = o(n)$. □

4 Depth-first search: Proof of Lemma 6

The proof of Lemma 6 is essentially the same as that of the special case when $p = (1+\varepsilon)p_0$ from [4]. Nevertheless we provide the proof here partly to make this paper self-contained and partly because some of the ideas will reappear in the more complicated proof of Lemma 8 in Section 6.

In order to prove the existence of a long path, we borrow the `Pathfinder` algorithm from [4]. This is in essence a depth-first search algorithm; however, there are a few complications in comparison to the graph case. We note that while [4] introduced the `Pathfinder` algorithm for general k and j, it includes a detailed description of the special case when $k = 3$ and $j = 2$ (see Section 4.1 of that paper) which the interested reader may find helpful to gain an intuition for how the algorithm proceeds. We will describe only the general case here.

Recall from Section 2.3 that, depending on the values of k and j, when we add an edge to the current path, we may have multiple (in fact precisely $\binom{k-j}{a}$) new j-sets from which we could extend the path. For this reason, each time we increase the length of the path, we produce a *batch* of j-sets with which the path could potentially end. In the example in Figure 2, the batch would consist of the three 4-sets containing the three new vertices and one of the three preceding vertices; more generally, a batch will contain any j-set from which the path can be extended if we discover a further edge containing that j-set (and no other vertices from the current path).

During the algorithm, at each time step t we will *query* a k-set to determine whether it forms an edge or not. This may be thought of as revealing the outcome of a Bernoulli(p) random variable corresponding to this k-set (with these variables being mutually independent).

We will describe j-sets as being *neutral*, *active* or *explored*; initially all j-sets are neutral; a j-set J becomes active if we have discovered a path which can end in J (in which case a whole batch becomes active at the same time as J); J becomes explored once we have queried all possible k-sets from J.

Of course, in order to produce a path we will not query any k-sets from J that contain any further vertices (apart from J) of the current path. But more than this, in order to allow *analysis* of the algorithm, we place an additional restriction: We do not query any k-set that contains any other active or explored j-set. This ensures that we never query the same k-set twice from different j-sets, and therefore the outcome of each query is independent of all other queries.

We store the active j-sets in an ordered stack. Whenever a new j-set becomes active, it is added to the top of the current stack, and it is removed when it becomes explored, corresponding to backtracking along the path. Since we are considering a depth-first search, we will always query k-sets from the *top* active j-set in the stack. Whenever the stack of active j-sets is empty (so also the current path is empty), we choose a new neutral j-set uniformly at random from which to continue, and this j-set becomes active. In particular, this happens immediately at the very start of the algorithm. We will choose an extendable partition of this j-set arbitrarily, and subsequently each j-set which becomes active will inherit an extendable partition induced by the extendable partition of its parent (i.e. the j-set it was discovered from).

Given an edge K with which a path P can be extended to a longer path, we denote this new path $P + K$. Given a set X and an element x, we denote $X + x := X \cup \{x\}$ and $X - x := X \setminus \{x\}$. If X is ordered, then we interpret $X + x$ as meaning that x is added to the end of X. A formal description of the `Pathfinder` algorithm can be found below. We emphasise that the algorithm is essentially identical to the version in [4], with only minor simplifications and some slight adaptations in the notation to be consistent with this paper.

Recall that during the algorithm, whenever we find an edge, $\binom{k-j}{a}$ new j-sets become active. Heuristically, towards the start of the process we will query approximately $\binom{n-v_\ell}{k-j}$ many k-sets from a j-set, where ℓ is the current length of the path (and recall that $v_\ell = j + \ell(k-j)$ denotes the number of vertices in a path of length ℓ). This gives a clear intuition for why we should find a path of length approximately $L_1(p)$: The expected number of j-sets that become active from any current j-set is approximately

$$\binom{k-j}{a}\binom{n-v_\ell}{k-j}p = (1+o(1))\left(1-\frac{\ell(k-j)}{n}\right)^{k-j} d.$$

When $\ell = L_1 = \frac{1-d^{-1/(k-j)}}{k-j}\cdot n$, up to the $1+o(1)$ error term this gives precisely 1; in other words, L_1 is asymptotically the length at which this process changes from being supercritical to subcritical.

The main difficulty in the proof comes in the approximation of the number of k-sets that we query from each j-set, which above we estimated by $\binom{n-v_\ell}{k-j}$. In fact, this is an obvious *upper* bound, whereas we need a *lower* bound. The upper bound takes account of k-sets which may not be queried because they contain a vertex from the current path, but k-sets may also be forbidden because they contain another active or explored j-set (apart from the one we are currently querying from).

We call a j-set *discovered* if it is either active or explored. The sets of j-sets which are active and explored at time t will be denoted by $\mathcal{Z}_t$ and $\mathcal{E}_t$ respectively. The set $\mathcal{G}_{\text{disc}} = \mathcal{G}_{\text{disc}}(t) := \mathcal{Z}_t \cup \mathcal{E}_t$ of discovered j-sets at time t may be thought of as the edge set of a j-uniform hypergraph. It is intuitive that at the start of the search process (i.e. for small t), this hypergraph is sparse, but we need to quantify this more precisely. Given $0 \leqslant i \leqslant j-1$, let $\Delta_i(t) = \Delta_i(\mathcal{G}_{\text{disc}}(t))$ denote the maximum i-degree of $\mathcal{G}_{\text{disc}}(t)$, i.e. the maximum over all i-sets I of the number of j-sets of $\mathcal{G}_{\text{disc}}(t)$ that contain I. Note in particular that, since a 0-set is simply $\emptyset$, we have $\Delta_0(t) = |\mathcal{G}_{\text{disc}}(t)|$. The purpose of this parameter is highlighted in the following proposition.

Proposition 9. *Suppose that a j-set J becomes active when the length of the path is ℓ_J and that $n - v_{\ell_J} = \Theta(n)$. Then the number of k-sets that are eligible to be queried from J at time t is at least*

$$\left(1 - \sum_{i=0}^{j-1} O\left(\frac{\Delta_i(t)}{n^{j-i}}\right)\right)\binom{n-v_{\ell_J}}{k-j}.$$

Proof. Let us consider how many k-sets may not be queried from a j-set J because they contain a second, already discovered j-set J'. We will make a case distinction based on the possible intersection size $i = |J \cap J'| \in [j-1]_0$, and note that for each $i \in [j-1]_0$, the number of discovered j-sets J' which intersect J in i vertices is at most $\binom{j}{i}\Delta_i(t)$, and the number of k-sets that are forbidden because they contain both J and J' is (crudely) at most n^{k-2j+i}. Therefore the number of forbidden k-sets in total is certainly at most

$$\sum_{i=0}^{j-1}\binom{j}{i}\Delta_i(t)n^{k-2j+i} = \sum_{i=0}^{j-1} O\left(\frac{\Delta_i(t)}{n^{j-i}}\right)\binom{n-v_{\ell_J}}{k-j},$$

Algorithm: `Pathfinder`

```
   Input: Integers $k, j$ such that $2 \leqslant j \leqslant k-1$
   Input: $\mathcal{H}$, a $k$-uniform hypergraph on vertex set $[n]$
 1 Let $a \in [k-j]$ be such that $a \equiv k \bmod (k-j)$
 2 Let $r = \lceil \frac{j}{k-j} \rceil - 1$
 3 For $i \in \{j, k\}$, let $\sigma_i$ be a permutation of the $i$-sets of $[n]$, chosen uniformly at random
 4 $\mathcal{N} \leftarrow \binom{[n]}{j}$                                  // neutral $j$-sets
 5 $\mathcal{Z}, \mathcal{E} \leftarrow \emptyset$                          // active, explored $j$-sets
 6 $P \leftarrow \emptyset$                                                  // current $j$-tight path
 7 $\ell \leftarrow 0$                                                       // index tracking the current length of $P$
 8 $t \leftarrow 0$                                                          // "time", number of queries made so far
 9 while $\mathcal{N} \neq \emptyset$ do
10     Let $J$ be the smallest $j$-set in $\mathcal{N}$, according to $\sigma_j$    // "new start"
11     Choose an arbitrary extendable partition $\mathcal{P}_J$ of $J$
12     $\mathcal{B}_0 = \{J\}$                                              // Starting batch of $j$-sets
13     $\mathcal{Z} \leftarrow \{J\}$
14     while $\mathcal{Z} \neq \emptyset$ do
15         Let $J$ be the last $j$-set in $\mathcal{Z}$
16         Let $\mathcal{K}$ be the set of $k$-sets $K \subset V(H)$ such that $K \supset J$, $K$ was not queried from $J$ before,
             $K \setminus J$ is vertex-disjoint from $P$, and $K$ does not contain any $J' \in \mathcal{E}$
17         if $\mathcal{K} \neq \emptyset$ then
18             Let $K$ be the first $k$-set in $\mathcal{K}$ according to $\sigma_k$
19             $t \leftarrow t+1$                                          // a new query is made
20             if $K \in \mathcal{H}$ then                                  // "query $K$"
21                 $e_\ell \leftarrow K$
22                 $P \leftarrow P + e_\ell$                                // $P$ is extended by adding $K = e_\ell$
23                 $\ell \leftarrow \ell + 1$                                // length of $P$ increases by one
24                 Let $(C_0, C_1, \dots, C_r) = \mathcal{P}_J$ be the extendable partition of $J$
25                 for each $Z \in \binom{C_1}{a}$ do
26                     $J_Z \leftarrow Z \cup C_2 \cup \dots \cup C_r \cup (K \setminus J)$     // $j$-set to be added
27                     $\mathcal{P}_{J_Z} \leftarrow (Z, C_2, \dots, C_r, K \setminus J)$      // extendable partition
28                     $\mathcal{Z} \leftarrow \mathcal{Z} + J_Z$           // $j$-set becomes active
29                 $\mathcal{B}_\ell \leftarrow \{J_Z : Z \in \binom{C_1}{a}\}$   // $j$-sets added to new batch
30             $(\mathcal{Z}_t, \mathcal{E}_t, P_t) \leftarrow (\mathcal{Z}, \mathcal{E}, P)$   // update "snapshot" at time $t$
31         else if $\mathcal{K} = \emptyset$ then                          // all extensions from $J$ were queried
32             $\mathcal{Z} \leftarrow \mathcal{Z} - J$                       // $J$ becomes explored
33             $\mathcal{E} \leftarrow \mathcal{E} + J$
34             if $\mathcal{B}_\ell \subset \mathcal{E}$ then               // the current batch is fully explored
35                 $\mathcal{B}_\ell \leftarrow \emptyset$                    // empty this batch
36                 $P \leftarrow P - e_\ell$                                // last edge of $P$ is removed
37                 $\ell \leftarrow \ell - 1$                                // length of $P$ decreases by one
   Output: $(\mathcal{Z}_t, \mathcal{E}_t, P_t)_{t \in \mathbb{N}_0}$
```

where the approximation follows because $\binom{n-v_{\ell_J}}{k-j} = \Theta\left(n^{k-j}\right)$. □

It follows from this proposition that if $\Delta_i(t) \ll n^{j-i}$ for each i, the number of forbidden k-sets is insignificant compared to the number of k-sets that may be queried, and the heuristic calculation above will go through with the addition of some smaller order error terms.

We will therefore run the `Pathfinder` algorithm until one of the following three stopping conditions is satisfied. Let us fix a constant $0 < \varepsilon \ll \delta^2$ and further constants $c_0, c_1, \ldots, c_{j-1}$ satisfying $1 \ll \frac{c_0}{d^{k+2}} \ll \frac{c_1}{d^{2(k+2)}} \ll \ldots \ll \frac{c_{j-1}}{d^{j(k+2)}} \ll 1/\sqrt{\varepsilon}$. [4]

Stopping conditions:

(DFS1) $\ell = (1-\delta/2)L_1$;

(DFS2) $t = t_0 := \varepsilon^2 n^k$;

(DFS3) $\Delta_i(t) \geqslant \varepsilon c_i n^{j-i}$ for some $0 \leqslant i \leqslant j-1$.

Now our goal is simply to show that whp the algorithm terminates when (DFS1) is invoked. As such, we have two main auxiliary results.

Proposition 10. *Whp (DFS2) is not invoked.*

Lemma 11. *Whp (DFS3) is not invoked.*

We note that Lemma 11 is a form of *bounded degree lemma* similar to the one first proved in [7] and subsequently used in one form or another in [4, 6], among others. A far stronger form also appeared in [5]. In its original form, the bounded degree lemma roughly states that no i-degree is larger than the average i-degree by more than a bounded factor. The stronger form in [5] even provides a lower bound, showing that whp all i-degrees are approximately equal, a phenomenon we call *smoothness*.

For our purposes we need only the upper bound, and allow a multiplicative deviation of $\Theta(c_i/\varepsilon)$ from the average. This could certainly be improved, and it seems likely that even smoothness is satisfied, but since we do not require an especially strong statement for the arguments in this paper, for simplicity we make no effort to optimise the parameters.

We first prove Proposition 10, which states that whp the algorithm does not run for too long before terminating.

Proof of Proposition 10. Let us suppose (for a contradiction) that at time $t_0 = \varepsilon^2 n^k$, neither (DFS1) nor (DFS3) has been invoked. These conditions imply that for all $i \in [j-1]_0$ and for all $t \leqslant t_0$ we have $\ell = \ell(t) \leqslant (1-\delta/2)L_1$ and $\Delta_i = \Delta_i(t) \leqslant \Delta_i(t_0) \leqslant \varepsilon c_i n^{j-i} \leqslant \sqrt{\varepsilon} n^{j-i}$. Therefore by Proposition 9, from each explored j-set we certainly made at least

$$\left(1 - O\left(\sqrt{\varepsilon}\right)\right)\binom{n - v_{(1-\delta/2)L_1}}{k-j} \geqslant \left(1+\delta^2\right)\binom{n - v_{L_1}}{k-j} \tag{2}$$

[4]Note that this is possible since by Remark 3 we may assume that $\delta \leqslant d^{-j(k+2)}$, and therefore if $\varepsilon \ll \delta^2$ we also have $d^{j(k+2)} \leqslant 1/\delta \ll 1/\sqrt{\varepsilon}$.

queries. We also observe that at time t_0 the number of edges we have discovered is distributed as $\mathrm{Bin}(t_0,p)$, which has expectation $t_0p=\Theta(\varepsilon^2 n^j)\to\infty$. By the Chernoff bound (Lemma 4), whp we have discovered at least $(1-\delta^3)t_0p$ edges, and therefore at least $(1-\delta^3)t_0p\binom{k-j}{a}$ many j-sets have become active. At any time, the number of currently active j-sets, which all lie within edges of the current path, is $O(L_1)=O(n)$, and therefore the number of fully explored j-sets is at least

$$\left(1-\delta^3\right)t_0p\binom{k-j}{a}-O(n)\geqslant\left(1-\delta^2/2\right)t_0p\binom{k-j}{a},\tag{3}$$

since $t_0p=\Theta\left(\varepsilon^2 n^j\right)\gg n$ (recall that $j\geqslant 2$).

Combining (2) and (3), the total number of queries made by time t_0 is at least

$$\left(1-\delta^2/2\right)t_0p\binom{k-j}{a}\left(1+\delta^2\right)\binom{n-v_{L_1}}{k-j}\geqslant\left(1+\delta^2/3\right)t_0,$$

which is a contradiction since by definition we have made precisely t_0 queries. □

We now go on to prove Lemma 11; we note that an essentially identical argument was used to prove [4, Lemma 34], but we repeat the argument here to make this paper self-contained.

Proof of Lemma 11. First consider the case $i=0$, when the desired bound follows from the fact that, by a Chernoff bound (Lemma 4) whp we have found at most $2pt_0=O(\varepsilon^2 n^j)$ edges, each of which leads to $O(1)$ many j-sets becoming active. Some further j-sets may also become active without finding an edge each time the stack of active j-sets is empty and we pick a new j-set from which to start. It is easy to bound the number of times this happens by $O(t_0 n^{j-k})=\Theta(\varepsilon^2 n^j)$ (see the argument for "new starts" below).

Now given $i\in[j-1]$ and an i-set I, there are three ways in which the degree of i in $\mathcal{G}_{\mathrm{disc}}$ could increase.

- A *new start* at I occurs when the current path is fully explored and we pick a new (ordered) j-set from which to continue the exploration process. If this j-set contains I, then the degree of I increases by 1.
- A *jump* to I occurs when a k-set containing I is queried from a j-set not containing I and this k-set is indeed an edge. Then the degree of I increases by at most $\binom{k-j}{a}$.
- A *pivot* at I occurs when an edge is discovered from a j-set already containing I. Then the degree of I increases by at most $\binom{k-j}{a}$.

We bound the contributions to the degree of I made by these three possibilities separately.

New starts

We can crudely bound the number of new starts by observing that for each *starting* j-set, by condition (DFS3) and Proposition 9, we must certainly have made at least

$$\left(1-\sum_{i=1}^{j-1} O\left(\frac{\varepsilon c_i n^{j-i}}{n^{j-i}}\right)\right)\binom{n-j}{k-j} \geqslant \frac{1}{2}\binom{n-j}{k-j}$$

queries to fully explore it, where we used the fact that $c_0,\dots,c_{j-1} \ll 1/\sqrt{\varepsilon}$. Therefore at time $t \leqslant t_0 = \varepsilon^2 n^k$ we can have made at most $N := 2t_0\binom{n-j}{k-j}^{-1}$ many new starts in total. We assume for an upper bound that this many new starts are indeed made. Since we chose the j-set for a new start uniformly at random, the probability that such a new start contains I is at most

$$\frac{\binom{n-i}{j-i}}{\binom{n}{j} - |\mathcal{G}_{\mathrm{disc}}(t)|} \leqslant \frac{2\cdot j!}{n^i} =: q,$$

where we used the fact that $|\mathcal{G}_{\mathrm{disc}}(t)| = \Delta_0(t) = O(\varepsilon c_0 n^j) \ll \binom{n}{j}$ by conditions (DFS2) and (DFS3) and the fact that $c_0 \ll 1/\sqrt{\varepsilon}$. Therefore the number of new starts containing I can be bounded above (formally, upper-coupled) by a random variable $X \sim \mathrm{Bin}(N,q)$. Note that X has expectation $Nq = \Theta(t_0 n^{j-k-i}) = \Theta(\varepsilon^2 n^{j-i}) \gg \sqrt{n}$. The Chernoff bound (Lemma 4) tells us that the probability that the number of new starts at I by time t_0 is larger than $\frac{11}{10}Nq$ is at most

$$\mathbb{P}\left(|X-Nq| \geqslant \frac{Nq}{10}\right) \leqslant 2\exp\left(\frac{-Nq}{300}\right) \leqslant \exp\left(-\sqrt{n}\right).$$

Assuming this low probability event does not occur, the total contribution to the degree of I made by new starts is at most

$$\frac{11}{10}Nq = \frac{11}{10}\cdot\frac{2t_0}{\binom{n-j}{k-j}}\cdot\frac{2j!}{n^i} \leqslant 5t_0\frac{j!(k-j)!}{n^{k-j+i}} = 5j!(k-j)!\varepsilon^2 n^{j-i} \leqslant \frac{c_i\varepsilon}{6(4d)^{k+1}}n^{j-i},$$

where we used the fact that $\varepsilon \ll 1 \ll \frac{c_i}{d^{k+1}}$, so in particular $5j!(k-j)!\varepsilon \leqslant \frac{c_i}{6(4d)^{k+1}}$.

We note that the rather artificial-looking denominator has been chosen with foresight of the calculations for the last contribution, made by pivots, which will depend on this term.

Jumps

We further subclassify jumps according to the size z of the intersection $I \cap J$ between I and the j-set J from which the jump to I occurs. Observe that since for a jump we cannot have $I \subset J$, we must have $0 \leqslant z \leqslant i-1$. The number of j-sets of $\mathcal{G}_{\mathrm{disc}}(t)$ with intersection of size z with I is at most $\binom{i}{z}\Delta_z(t) \leqslant \binom{i}{z}\varepsilon c_z n^{j-z}$, where we used condition (DFS3) with z

in place of i. For each such j-set J, the number of k-sets which contain both J and I, and which could therefore result in a jump to I, is at most $\binom{n}{k-j-i+z} \leqslant n^{k-j-i+z}$.

Thus the total number of queries made which could result in jumps to I is at most

$$\sum_{z=0}^{i-1} \binom{i}{z} \varepsilon c_z n^{j-z} \cdot n^{k-j-i+z} \leqslant 2^i \varepsilon c_{i-1} n^{k-i} =: N,$$

where we used the fact that $c_{i-1} = \max_{z=0}^{i-1} c_z$. Each such query gives a jump with probability $p = \Theta(n^{k-j})$, and so we can bound from above the number of jumps to I by a random variable $X \sim \mathrm{Bin}(N,p)$. The Chernoff bound (Lemma 4) implies that the probability that the number of jumps is larger than $2Np = \Theta(\varepsilon c_{i-1} n^{j-i}) \gg \sqrt{n}$ is at most

$$\mathbb{P}\left(|X - Np| \geqslant Np\right) \leqslant 2\exp\left(-\frac{Np}{3}\right) \leqslant \exp\left(-\sqrt{n}\right).$$

Assuming this low probability event does not occur, since each jump contributes at most $\binom{k-j}{a}$ to the degree of I, the total contribution made by jumps is at most

$$\binom{k-j}{a} 2Np \leqslant \binom{k-j}{a} 2^{i+1} \varepsilon c_{i-1} n^{k-i} \cdot \frac{2d}{\binom{k-j}{a}\binom{n}{k-j}} \leqslant \frac{\varepsilon c_i}{6(4d)^{k+1}} n^{j-i},$$

where we used the fact that $\frac{c_i}{d^{k+2}} \gg c_{i-1}$, so in particular $2^{i+2} d \cdot 2(k-j)! \cdot c_{i-1} \leqslant \frac{c_i}{6(4d)^{k+1}}$.

Pivots

We observe that from any j-set containing I, the expected number of pivots at I is at most $\binom{n}{k-j} p \leqslant \frac{2d}{\binom{k-j}{a}}$, and each pivot gives rise to at most $\binom{k-j}{a}$ new j-sets. Furthermore since we are studying a DFS process creating a path, the number of consecutive pivots at I can be at most $\frac{k-i}{k-j} \leqslant k$ before the path has left I. Since the number of new starts and jumps to I is at most $\frac{\varepsilon c_i}{3(4d)^{k+1}} n^{j-i}$, it follows that the expected number of pivots at I is at most

$$\left(\sum_{z=1}^{k} (2d)^z\right) \frac{\varepsilon c_i}{3(4d)^{k+1}} n^{j-i} \leqslant \frac{\varepsilon c_i}{3 \cdot 2^{k+1}} n^{j-i},$$

and by Lemma 4 with $\alpha = 2$, with probability at least $1 - \exp\left(-\Theta\left(\varepsilon c_i n^{j-i}\right)\right) \geqslant 1 - \exp\left(-\sqrt{n}\right)$, the number of pivots at I is at most $\frac{\varepsilon c_i}{3 \cdot 2^k} n^{j-i}$. Since each pivot contributes at most $\binom{k-j}{a} \leqslant 2^k$ to the degree of I, if this high probability event holds, the contribution to the degree of I made by pivots is at most $\frac{\varepsilon c_i}{3} n^{j-i}$.

Now we have bounded the contribution to the degree of I made by each of the three possibilities as certainly at most $\frac{\varepsilon c_i}{3} n^{j-i}$, and summing these three terms gives the desired result for I. It remains to observe that the failure probabilities are small enough that, applying a union bound over all choices of I, we still have only $o(1)$ failure probability: This follows because the failure probabilities are all at most $\exp(-\sqrt{n})$, while there are at most n^i choices for I. $\square$

5 The buffer zone

Lemma 6 guarantees the existence of a long path whp. We would like to go on to extend this long path using a BFS process to find many potential ends of long paths as required in Lemma 8. However, we first need to handle the fact that, by running the `Pathfinder` algorithm, we have already revealed information about the random hypergraph which could influence its distribution, and therefore the behaviour of the BFS process. To ensure that this does not happen, we need to ensure that every k-set queried in the BFS process has not been queried before.

To achieve this, in this section we will show how we can make a few extra queries to create a "buffer zone" between the DFS and BFS processes. We will first set aside a relatively small number of vertices as a "reservoir", and run the DFS algorithm on the remaining vertices. Since almost all vertices are still available, whp we discover a long path. We then extend this long path into the reservoir. We then have the property that from any of the new ends, no queries have been made which do not contain vertices of the long path or of the reservoir, and it is from these new ends that we will begin the BFS process.

Ultimately, these considerations allow us to prove the following lemma. In the statement of the lemma, the sets U and $\mathcal{D}$ are used to restrict which queries have already been made, which will later allow us to make further queries while maintaining independence.

Lemma 12. *We can run a search algorithm in $\mathcal{H} = \mathcal{H}^k(n,p)$ that outputs*

- *a vertex set U,*
- *a path P_0 with ends $\hat{J}_1, \hat{J}_2$,*
- *collections $\mathcal{J}_1, \mathcal{J}_2, \mathcal{D}$ of j-sets and*
- *paths P_J for each $J \in \mathcal{J}_1 \cup \mathcal{J}_2$*

which have the following properties whp:

(A1) $|U| = (1-\delta/2)(k-j)L_1$;

(A2) $\ell(P_0) = (1-\delta)L_1$;

(A3) $|\mathcal{J}_1|, |\mathcal{J}_2| = (\log n)^2$;

(A4) for each $s \in \{1,2\}$ and for each $J \in \mathcal{J}_s$ the path P_J lies in $\mathcal{H}[U]$ and path-$(J,(2\log n)^4)$-augments $(P_0, \hat{J}_s)$;

(A5) For each $i \in [j-1]_0$ we have $\Delta_i(\mathcal{D}) \leqslant \varepsilon c_i n^{j-i}$;

and furthermore

(B1) For any pair $(J_1, J_2) \in \mathcal{J}_1 \times \mathcal{J}_2$, we have that $P_{J_1} \cup P_{J_2}$ is a path with ends J_1, J_2;

(B2) For any $J \in \mathcal{J}_1 \cup \mathcal{J}_2$, all k-sets containing J which have been queried contain at least one further vertex of U;

(B3) Only k-sets containing a j-set of $\mathcal{D}$ have been queried.

Proof. Let $n_2 := \frac{n}{\log\log n}$ and $n_1 := n - n_2$. We begin by setting $R := [n] \setminus [n_1]$, which is a reservoir of n_2 vertices which we set aside for later. We then run the `Pathfinder` algorithm in $\mathcal{H}^k(n_1, p)$, which has vertex set $[n_1] = [n] \setminus R$. It is easy to check that, even with the change from n to n_1, Lemma 6 still implies that whp the algorithm will discover a path P_0^* of length $L^* := (1-\delta)L_1 + 2(\log n)^4 \leqslant (1-\delta/2)L_1$. We set $U^* := V(P_0^*) \cup R$, and observe that

$$|U^*| = v_{L^*} + n_2 = (1-\delta)\,(k-j)L_1 + 2(\log n)^2(k-j) + j + \frac{n}{\log\log n}$$
$$\leqslant \left(1 - \frac{\delta}{2}\right)(k-j)L_1.$$

To obtain U of the precise size required by (A1), we artificially add the appropriate number of further vertices to U^*.

Now let P_0 be the path obtained from P_0^* by removing $(\log n)^4$ edges from each end, so $\ell(P_0) = (1-\delta)L_1$, as required by (A2). Furthermore let $\mathcal{J}_1^*, \mathcal{J}_2^*$ be the collections of j-sets contained in the intersection of two such removed edges, split into two classes in the natural way. Clearly for any $J^* \in \mathcal{J}_1^* \cup \mathcal{J}_2^*$ we have a path P_{J^*} ending in J^* which contains P_0 and is contained in P_0^*. We choose P_{J^*} to be the minimal such path, so in particular if $J \in \mathcal{J}_1^*$ the corresponding path contains no j-sets of $\mathcal{J}_2^*$ and vice versa.

We next split R into two disjoint sets R_1, R_2 each of size $n_2/2$, and query any k-set consisting of a j-set of $\mathcal{J}_1^*$ and $k-j$ vertices of R_1. For each edge K we discover from $J_1^* \in \mathcal{J}_1^*$ in this way, we can extend $P_{J_1^*}$ by adding K and we find at least one j-set with which the path $P_{J_1^*} + K$ could end. The number of queries we make in this step is

$$|\mathcal{J}_1^*|\binom{|R_1|}{k-j} = (\log n)^4\binom{n_2/2}{k-j} = \Theta\left((\log n)^4\left(\frac{n}{2\log\log n}\right)^{k-j}\right)$$
$$\geqslant (\log n)^3 n^{k-j}.$$

It follows by the Chernoff bound (Lemma 4) that whp the number of edges we discover from $\mathcal{J}_1^*$ is at least $\frac{1}{2}p(\log n)^3 n^{k-j} \geqslant (\log n)^2$.

We would like to deduce that whp we find a large set of potential ends in this way. However, we need to be careful to ensure that we do not discover the same end from many different j-sets of $\mathcal{J}_1^*$. To exclude this possibility, we distinguish two cases.

Case 1: $j \leqslant k/2$

In this case, given a j-set $J \subset R_1$ and a j-set $J_1^* \in \mathcal{J}_1^*$, there are at most n^{k-2j} many k-sets $K \supset J \cup J_1^*$ (crudely ignoring the restriction that the remaining vertices must come from R_1), and so the probability that a particular j-set $J \subset R_1$ is discovered from two distinct j-sets of $\mathcal{J}_1^*$ is at most

$$\left(|\mathcal{J}_1^*|n^{k-2j}p\right)^2 = O\left((\log n)^8 n^{2k-4j} n^{-2(k-j)}\right) = O\left(\frac{(\log n)^8}{n^{2j}}\right).$$

Thus the expected number of j-sets discovered twice in this way is at most

$$\binom{n_2/2}{j} \cdot O\left(\frac{(\log n)^8}{n^{2j}}\right) = O\left(\frac{(\log n)^8}{n^j(\log\log n)^j}\right) = o(1).$$

Case 2: $j > k/2$

We first observe that we have at most $|\mathcal{J}_1^*|n_2^{k-j} \cdot O(1)$ ways of choosing the newly discovered j-set (at most $O(1)$ for each k-set queried). Second, observe that since such a j-set can only contain $k-j$ vertices of R_1, it must still contain vertices of P_0. In particular, there are certainly only $O(1)$ many j-sets of $\mathcal{J}_1^*$ from which a fixed j-set J can be discovered. We can therefore estimate the expected number of j-sets discovered twice in this way by

$$|\mathcal{J}_1^*|n_2^{k-j}O(1)p^2 = O\left(\frac{(\log n)^4}{(n\log\log n)^{k-j}}\right) = o(1).$$

In both cases, the expected number of j-sets discovered twice is $o(1)$, so by Markov's inequality, whp we do not discover any j-set twice. Therefore the arguments above show that whp we discover a set of at least $(\log n)^2$ new ends, each of which contains $k-j$ vertices in R_1. We set $\mathcal{J}_1$ to be an arbitrary subset of precisely $(\log n)^2$ of these ends, and for each $J_1 \in \mathcal{J}_1$ which is discovered from some $J_1^* \in \mathcal{J}_1^*$ via an edge K, we set $P_{J_1} := P_{J_1^*} + K$. By construction, $\ell(P_{J_1} \setminus P_0) = \ell(P_{J_1^*} \setminus P_0) + 1 \leqslant (\log n)^4 + 1 \leqslant 2(\log n)^4$.

Analogously, we also query any k-set consisting of a j-set of $\mathcal{J}_2^*$ and $k-j$ vertices of R_2; an identical argument shows that whp we discover a set of $(\log n)^2$ ends, which we call $\mathcal{J}_2$, and we define P_{J_2} for each $J_2 \in \mathcal{J}_2$ analogously. We have just proved that $\mathcal{J}_1, \mathcal{J}_2$ satisfy property (A3) whp, and the paths P_J for $J \in \mathcal{J}_1 \cup \mathcal{J}_2$ satisfy (A4) by construction.

We further observe that for any $J_1 \in \mathcal{J}_1$ and $J_2 \in \mathcal{J}_2$, since we discovered these sets by extending a path with ends $J_1^* \in \mathcal{J}_1^*$ and $J_2^* \in \mathcal{J}_2^*$, and since the sets R_1, R_2 used for these extensions are disjoint, we have a path between J_1 and J_2, as required for (B1).

We also observe that the only queries we made containing a j-set $J \in \mathcal{J}_1 \cup \mathcal{J}_2$ were those seeking extensions, i.e. those made from a j-set in $\mathcal{J}_1^* \cup \mathcal{J}_2^*$, which therefore contain at least one vertex of $V(\mathcal{J}_1^* \cup \mathcal{J}_2^*) \setminus J \subset U \setminus J$, as required in (B2).

It remains to prove that, for the appropriate choice of $\mathcal{D}$, we also have (A5) and (B3). Recall that P_0^* was constructed using a depth-first search, which we may terminate as soon as it has found a path of the appropriate size. Let t_0^* be the time at which this process is terminated, and recall that $\mathcal{G}_{\text{disc}}(t_0^*)$ denotes the set of j-sets which are discovered up to time t_0^*. We set $\mathcal{D} := \mathcal{G}_{\text{disc}}(t_0^*)$ and claim that it has the required properties whp.

First observe that (B3) certainly holds by construction: during the DFS process we only queried k-sets from active j-sets, which must be in $\mathcal{G}_{\text{disc}}(t) \subset \mathcal{G}_{\text{disc}}(t_0^*)$ if the query is made at time t; on the other hand, the only k-sets queried outside of the DFS process contain j-sets of $\mathcal{J}_1^* \cup \mathcal{J}_2^* \subset \mathcal{G}_{\text{disc}}(t_0^*)$.

Finally, note that Lemma 11 precisely guarantees that (A5) holds whp. $\square$

6 Breadth-first search: Proof of Lemma 8

In this section we show how we can use the configuration guaranteed whp by Lemma 12 and extend this to obtain a family of paths with many ends (far more than the $2(\log n)^2$ we have so far), as required by Lemma 8.

6.1 The BFS algorithm: `Pathbranch`

6.1.1 Motivation and setup

We will use Lemma 12 as a black box. We therefore assume the high probability event and fix $U, P_0, \mathcal{J}_1, \mathcal{J}_2, \mathcal{D}, \{P_J : J \in \mathcal{J}_1, \mathcal{J}_2\}$ as defined in Lemma 12. In a slight abuse of notation, we will sometimes talk of "the path P_0" when we mean one of the paths running between some $J_1 \in \mathcal{J}_1$ and $J_2 \in \mathcal{J}_2$, which contain P_0 but are longer and not all identical (although they are all very similar).

Our aim is to run two breadth-first processes starting at $\mathcal{J}_1, \mathcal{J}_2$ to extend P_0, and to show that these processes quickly grow large. Figure 4 shows a simplified visualisation of the inputs and outputs of the `Pathbranch` algorithm.

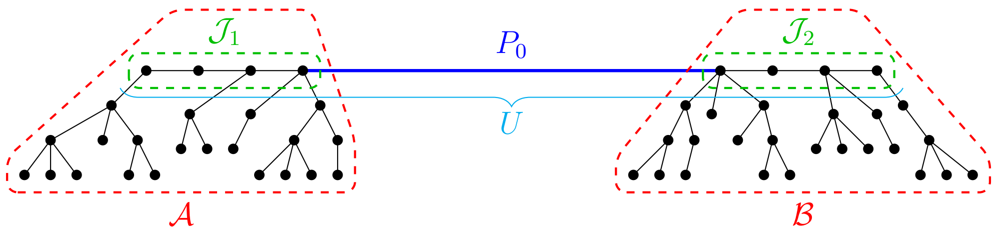

Figure 4: The configuration constructed by our two applications of the `Pathbranch` algorithm. Note that the vertices in this figure represent j-sets, and the j-sets belonging to $\mathcal{A}$ and $\mathcal{B}$ need not be disjoint. Note also that in general $\mathcal{J}_1, \mathcal{J}_2$ do not lie in a single common path, but already branch out.

Simply proving that the processes grow large would be relatively easy by adapting the proof strategy from Section 4, since the size of U is such that the processes are (just) supercritical, and intuitively we only need a logarithmic number of steps to grow to polynomial size. More delicate, however, is to show that the search process produces path ends that are compatible with each other, in the sense that there are many choices of pairs of ends between which we have a path (as required for Property (iv) of Lemma 8). In order to construct compatible sets of ends, having run the algorithm once to find augmenting paths at one end, we will have a set $\mathcal{F}_1$ of *forbidden vertices*; roughly speaking, these are vertices which lie in too many of the augmenting paths from the first application of the algorithm, and therefore we would like to avoid them when constructing augmenting paths at the other end.

6.1.2 Informal description

Let us first describe the algorithm informally. We will start with a set of vertices U, a path P_0 with $V(P_0) \subset U$ and a set of ends $\mathcal{J}$ (which will be either $\mathcal{J}_1$ or $\mathcal{J}_2$). These ends come with the natural extendable partition induced by P_0. We will also have sets $\mathcal{F}_1$ of forbidden vertices and $\mathcal{F}_j$ of forbidden j-sets. As in the DFS algorithm, we will label j-sets as *neutral*, *active* or *explored.* Initially the j-sets of $\mathcal{J}$ are active, the j-sets of $\mathcal{F}_j$ are explored, and all others are neutral.

At each time t we will *query* a k-set K from an active j-set $J \subset K$ to determine whether K is an edge. If it is, then $\binom{k-j}{a}$ new j-sets are potential ends with which we can extend the path from J, and these become active, also inheriting an appropriate extendable partition. In order to ensure that we are always creating a path, we will forbid queries of k-sets which contain vertices of the path ending in J (except the vertices of J itself). We will also forbid k-sets with vertices from U or from the forbidden set $\mathcal{F}_1$. Finally, to ensure independence of the queries we will forbid k-sets which contain some explored j-set – since we initially set $\mathcal{F}_j$ to be explored, our choice of $\mathcal{F}_j$ will also guarantee independence from previous exploration processes. (Note that since we are using a breadth-first search, we do not need to forbid k-sets which contain another active j-set J', because J' will be dealt with later once J is explored, and such k-sets will be forbidden from J' because they contain J.)

We will proceed in a standard BFS manner, i.e. from the first active j-set in the queue we will query all permissible k-sets, and any new j-sets we discover are added to the end of the queue.

We note that during the BFS process, the j-sets which become explored, including those in $\mathcal{J}_1 \cup \mathcal{J}_2$, are certainly ends of a path containing P_0, and therefore candidates in our later sprinkling step.

6.1.3 Formal description

Given a path P_J ending in a j-set J, we denote by $\mathcal{P}[P_J]$ the extendable partition of J which is naturally induced by P_J (which end of P_J we mean will be clear from both the notation and the context).

The formal description of the `Pathbranch` algorithm appears below.

We will run this algorithm twice, once with $\mathcal{J} = \mathcal{J}_1$ and once with $\mathcal{J} = \mathcal{J}_2$. To guarantee independence of the two processes, we will make use of the ability to forbid a set $\mathcal{F}_j$ of j-sets from being queried – in particular, during the second application of the algorithm, $\mathcal{F}_j$ will include the discovered j-sets from the first application. More generally, $\mathcal{F}_j$ will also include discovered j-sets from the previous DFS process. Forbidding this set ensures that no k-set will ever be queried twice.

Algorithm: `Pathbranch`

```
Input: Integers $k, j$ such that $2 \leqslant j \leqslant k-1$
Input: $\mathcal{H}$, a $k$-uniform hypergraph on vertex set $[n]$
Input: Vertex set $U \subset V(\mathcal{H})$
Input: Path $P_0 \subset \mathcal{H}[U]$
Input: Collection of $j$-sets $\mathcal{J} \in \binom{U}{j}$
Input: Collection of paths $\{P_J : J \in \mathcal{J}\}$ such that $P_0 \subset P_J \subset \mathcal{H}[U]$ and $P_J$ ends in $J$
Input: $\mathcal{F}_1$, a set of forbidden vertices
Input: $\mathcal{F}_j$, a set of forbidden $j$-sets
1  Let $a \in [k-j]$ be such that $a \equiv k \bmod (k-j)$
2  Let $r = \lceil \frac{j}{k-j} \rceil - 1$
3  $\mathcal{Z} \leftarrow \mathcal{J}$ ordered lexicographically                 // active $j$-sets
4  $\mathcal{E} \leftarrow \mathcal{F}_j$                                          // explored $j$-sets
5  forall $J \in \mathcal{J}$ do
6      $\ell_J \leftarrow |P_J|$                                                 // length of $P_J$
7      $\mathcal{P}_J \leftarrow \mathcal{P}_J[P_J]$                             // extendable partition of $J$
8  $t \leftarrow 0$                              // "time", number of queries made so far
9  while $\mathcal{Z} \neq \emptyset$ do
10     Let $J$ be the first $j$-set in $\mathcal{Z}$
11     Let $\mathcal{K}$ be the set of $k$-sets $K \supset J$, such that $K \setminus J \in [n] \setminus (U \cup \mathcal{F}_1)$, and such that $K$ does not
         contain any $J' \in \mathcal{E}$                                      // Available queries from $J$
12     while $\mathcal{K} \neq \emptyset$ do
13         Let $K$ be the first $k$-set in $\mathcal{K}$ according to the lexicographic order
14         $t \leftarrow t + 1$                                                  // a new query is made
15         if $K \in \mathcal{H}$ then                                           // "query $K$"
16             Let $(C_0, C_1, \ldots, C_r) = \mathcal{P}_J$ be the extendable partition of $J$
17             for each $Z \in \binom{C_1}{a}$ do
18                 $J_Z \leftarrow Z \cup C_2 \cup \cdots \cup C_r \cup (K \setminus J)$       // $j$-set to be added
19                 $P_{J_Z} \leftarrow P_J + K$                                  // Path ending at $J_Z$
20                 $\mathcal{P}_{J_Z} \leftarrow (Z, C_2, \ldots, C_r, K \setminus J)$   // extendable partition of $J_Z$
21                 $\ell_{J_Z} \leftarrow \ell_J + 1$
22                 $\mathcal{Z} \leftarrow \mathcal{Z} + J_Z$                    // $j$-set becomes active
23         $(\mathcal{Z}_t, \mathcal{E}_t,) \leftarrow (\mathcal{Z}, \mathcal{E})$        // update "snapshot" at time $t$
24         $\mathcal{K} \leftarrow \mathcal{K} - K$                              // update $\mathcal{K}$
25     $\mathcal{E} \leftarrow \mathcal{E} + J$                                  // $J$ becomes explored
26     $\mathcal{Z} \leftarrow \mathcal{Z} - J$                                  // $J$ is no longer active
Output: $(\mathcal{Z}_t, \mathcal{E}_t)_{t \in \mathbb{N}_0}$
```

6.2 Analysing the algorithm

When we apply the `Pathbranch` algorithm, the inputs will satisfy various helpful properties. For convenience, we collect all of these properties together.

BFS Setup:

(C1) $\mathcal{H} = \mathcal{H}^k(n,p)$;

(C2) $|U| = (1-\delta/2)(k-j)L_1$;

(C3) $\ell(P_0) = (1-\delta)L_1$;

(C4) $|\mathcal{J}| = (\log n)^2$;

(C5) $\ell(P_J \setminus P_0) \leqslant 2(\log n)^4$ for all $J \in \mathcal{J}$;

(C6) $|\mathcal{F}_1| \leqslant \delta^2 n$;

(C7) $\Delta_i(\mathcal{F}_j) \leqslant 2\varepsilon c_i n^{j-i}$ for each $i \in [j-1]_0$;

(C8) Any k-set of $\mathcal{H}$ previously queried contains a j-set of $\mathcal{F}_j$.

For the remainder of this section, in which we analyse the likely behaviour of this algorithm, we will assume that all of these properties are indeed satisfied. In particular, (C8) ensures that we have no prior information about any queries that we make during the `Pathbranch` algorithm, which we will make use of extensively without explicitly mentioning (C8) again. Not all of these properties will be used for each result we prove, but for simplicity we will always assume them all. When applying the results of this section, we will need to check these assumptions.

Note that in the algorithm, $\mathcal{Z}_t$ and $\mathcal{E}_t$ are the sets of active and explored j-sets respectively at time t. In particular, if $J \in \mathcal{Z}_t$, then J is the end of a path P_J of length ℓ_J. Let us define $\ell_t := \max_{J\in\mathcal{Z}_t} \ell_J$. Since each path P_J contains the path P_0 of length $(1-\delta)L_1$, let us also define $\ell_t^* := \ell_t - (1-\delta)L_1$, which is an upper bound on $\ell(P_J \setminus P_0)$ for all j-sets J which are active at time t, so using (C2) and (C6) we have

$$|U \cup \mathcal{F}_1 \cup V(P_J)| \leqslant \left(1 - \frac{\delta}{2}\right)(k-j)L_1 + \delta^2 n + (k-j)\ell_t^*. \tag{4}$$

Recall from Section 4 that we have constants satisfying $0 < \varepsilon \ll \delta^2 \ll 1$ and $1 \ll c_0 \ll c_1 \ll \ldots \ll c_{j-1} \ll 1/\sqrt{\varepsilon}$. As in the DFS algorithm, let $\mathcal{G}_{\text{disc}}(t) := \mathcal{Z}_t \cup \mathcal{E}_t$, and we denote $\Delta_i(t) := \Delta_i(\mathcal{G}_{\text{disc}}(t))$ for any $0 \leqslant i \leqslant j-1$ and $t \in \mathbb{N}$. We will run the `Pathbranch` algorithm until time T_{stop}, the first time at which one of the following stopping conditions is satisfied.

Stopping conditions:

(S1) The algorithm has terminated;

(S2) $\ell_t = (1-\delta)\,L_1 + 3(\log n)^4 =: L_0$;

(S3) $\Delta_i(t) \geqslant \varepsilon c_i n^{j-i}$ for some i;

(S4) $|\mathcal{G}_{\mathrm{disc}}(t)| = \varepsilon^2 n^j$.

Let us note that (S2) is equivalent to $\ell_t^* = 3(\log n)^4$. Our principal aim is to show that whp it is (S4) that is invoked. The following proposition will be critical.

Proposition 13. *Assuming that the properties of BFS Setup hold, for any $t \leqslant T_{\mathrm{stop}}$, the number of k-sets which are eligible to be queried from an active j-set is at least $\binom{n-v_{(1-\delta/3)L_1}}{k-j} = (1+\Theta(\delta))\binom{n-v_{L_1}}{k-j}$.*

Proof. We first observe that an essentially identical proof to that of Proposition 9 shows the analogous result in this case: Here we have that $\ell_J \leqslant L_0$ because the stopping condition (S2) has not been invoked. As a consequence, we also have

$$\begin{aligned}|U \cup \mathcal{F}_1 \cup V(P_J)| &\overset{(4)}{\leqslant} \left(1-\frac{\delta}{2}\right)(k-j)L_1 + \delta^2 n + (k-j)3(\log n)^4 \\ &= \left(1-\frac{\delta}{2}+O(\delta^2)+o(1)\right)(k-j)L_1 \\ &\leqslant \left(1-\frac{2\delta}{5}\right)(k-j)L_1,\end{aligned}$$

where the second estimate holds since $L_1 = \Theta(n)$, and the third holds since $\delta \ll 1$ is constant. Combining (C7) and (S3), we also have

$$\Delta_i(\mathcal{F}_j \cup \mathcal{E}_t) \leqslant \Delta_i(\mathcal{F}_j) + \Delta_i(\mathcal{E}_t) \leqslant 3\varepsilon c_i n^{j-i} \leqslant \sqrt{\varepsilon}\, n^{j-i}.$$

Therefore the number eligible k-sets is at least

$$\left(1-O\left(\sqrt{\varepsilon}\right)\right)\binom{n-|U\cup\mathcal{F}_1\cup V(P_J)|}{k-j} \geqslant \binom{n-\left(1-\frac{\delta}{3}\right)(k-j)L_1}{k-j},$$

where the last estimate holds since $L_1 = \Theta(n)$ and since $\varepsilon \ll \delta \ll 1$. Recalling that by definition $v_\ell = (k-j)\ell + j \geqslant (k-j)\ell$, the claim follows. □

Claim 14. *Assuming that the properties of BFS Setup hold, whp (S1) is not invoked.*

Proof. In order for (S1) to be invoked, all j-sets which became active would need to be fully explored at time T_{stop}. Let $m := |\mathcal{G}_{\mathrm{disc}}(T_{\mathrm{stop}})| = |\mathcal{E}_{T_{\mathrm{stop}}}|$ denote the number of j-sets which became active (or were active initially).

By Proposition 13, the algorithm has made at least $M := m \cdot (1+c\delta)\binom{n-v_{L_1}}{k-j}$ queries for some constant c, from which we certainly discovered at most $m' := m\binom{k-j}{a}^{-1}$ edges (since each edge gives rise to $\binom{k-j}{a}$ new active j-sets). Thus we have

$$M/m' = (1+c\delta)\binom{n-v_{L_1}}{k-j}\binom{k-j}{a} \geqslant \frac{1}{(1-\delta^2)p},$$

or in other words $m' \leqslant (1-\delta^2)pM$. Thus we have made at least M queries during which we discovered at most $m' \leqslant (1-\delta^2)pM$ edges. The Chernoff bound will show that this is very unlikely provided pM is large enough.

More precisely, note that since the j-sets of $\mathcal{J}$ were initially active, we certainly have $m \geqslant |\mathcal{J}| \geqslant (\log n)^2$ by (C4), and therefore $M = \Omega\left((\log n)^2 n^{k-j}\right)$. For any $t = \Omega\left((\log n)^2 n^{k-j}\right)$, by Lemma 4 the probability that in the first t queries we find at most $(1-\delta^2)pt$ edges is at most $\exp(-\Theta(pt)) \leqslant \exp(-\Theta((\log n)^2)) = o(n^{-k})$. Therefore we may take a union bound over all times t between 0 and n^k (which is a trivial upper bound on the total number of queries that can be made) and deduce that whp (S1) was not invoked during this time. $\square$

Proposition 15. *Assuming that the properties of BFS Setup hold, whp (S2) is not invoked.*

Proof. We consider the *generations* of the BFS process, where the j-sets of $\mathcal{J}$ form generation 0 and a j-set lies in generation i if it was discovered from a j-set in generation $i-1$. Observe that since for each $J \in \mathcal{J}$ by (C3) and (C5) we have

$$\ell_J \leqslant (1-\delta)\,L_1 + 2(\log n)^4,$$

(S2) can only be invoked if we have reached generation at least $(\log n)^4$.

Let us define X_i to be the number of j-sets in generation i, so in particular $X_0 = |\mathcal{J}| = (\log n)^2$ deterministically. By Proposition 13, we make at least $X_i(1+\Theta(\delta))(\binom{k-j}{a}p)^{-1}$ queries to obtain generation $i+1$ from generation i, and therefore $\mathbb{E}(X_{i+1}|X_i = x_i) \geqslant (1+\Theta(\delta))x_i$. Therefore by Lemma 4,

$$\mathbb{P}\left(X_{i+1} \leqslant (1+\delta^2)x_i | X_i = x_i\right) \leqslant 2\exp\left(-\Theta(\delta^2 x_i)\right).$$

Inductively applying a union bound, we deduce that with probability at least $1 - 2i\exp\left(-\Theta(\delta^2 X_0)\right)$ we have $X_{i'} \geqslant (1+\delta^2)^{i'} X_0 \geqslant X_0$ for all $0 \leqslant i' \leqslant i$ until time T_{stop} (after which Proposition 13 no longer applies).

In order to reach generation $(\log n)^4$ (and therefore potentially invoke (S2)), this would involve discovering a generation of size at least $(1+\delta^2)^{(\log n)^4} X_0 > n^j$, which is clearly impossible since this is larger than the total number of j-sets available (and indeed (S4) would already have been applied long before this point). Meanwhile, the error probability is at most $2(\log n)^4 \exp\left(-\Theta\left(\delta^2 (\log n)^2\right)\right) = o(1)$. $\square$

Lemma 16. *Assuming that the properties of BFS Setup hold, whp (S3) is not invoked.*

Proof. The proof is broadly similar to the proof of Lemma 11. The case when $i = 0$ is trivial, since $\Delta_0(t) = |\mathcal{G}_{\rm disc}(t)| \leqslant \varepsilon^2 n^j$ because of (S4). Let us therefore suppose that $1 \leqslant i \leqslant j-1$ and I is an i-set. We observe that there are three ways in which the degree of I can increase in $\mathcal{G}_{\rm disc}$.

- A *new start* at I consists of a j-set of $\mathcal{J}$ which contains I. This contributes one to the degree of I.
- A *jump* to I occurs when an edge containing I is discovered from a j-set not containing I. Then the degree of I increases by at most $\binom{k-j}{a}$.
- A *pivot* at I occurs when an edge is discovered from a j-set already containing I. Then the degree of I increases by at most $\binom{k-j}{a}$.

We bound the contributions made by new starts, jumps and pivots separately.

New starts

Note that in contrast to the DFS algorithm, new starts are already determined by the input, since we start only once with active j-sets $\mathcal{J}$ and terminate the algorithm if we have no more active j-sets. It is also clear that the degree of I in $\mathcal{J}$ is trivially at most $|\mathcal{J}| = (\log n)^2$ by (C4).

Jumps

The number of j-sets of $\mathcal{G}_{\rm disc}$ which intersect I in $z \leqslant i-1$ vertices is at most $\binom{i}{z}\Delta_z(\mathcal{G}_{\rm disc}) \leqslant \binom{i}{z}\varepsilon c_z n^{j-z}$, where we have used (S3) with z in place of i.

Furthermore, each such j-set gives rise to at most $\binom{n}{k-j-i+z}$ queries which would result in a jump to I if the corresponding k-set is an edge, and thus the total number of queries which would result in a jump to I is at most

$$\sum_{z=0}^{i-1} \binom{i}{z} \varepsilon c_z n^{j-z} \binom{n}{k-j-i+z} \leqslant 2^i \varepsilon c_{i-1} n^{k-i},$$

where we used the fact that $c_{i-1} = \max_{z=0}^{i-1} c_z$. Each such query gives a jump with probability $p \leqslant \frac{2d(k-j)!}{\binom{k-j}{a} n^{k-j}}$, and so the expected number of jumps is at most

$$\frac{2^{i+1} d(k-j)!}{\binom{k-j}{a}} \varepsilon c_{i-1} n^{j-i} \gg \sqrt{n}.$$

The Chernoff bound (Lemma 4) implies that with probability at least $1 - \exp(-\sqrt{n})$ the number of jumps is at most twice its expectation, in which case, since each jump contributes at most $\binom{k-j}{a}$ to the degree of I in $\mathcal{G}_{\rm disc}$, the total contribution is at most $2^{i+2} d(k-j)! \varepsilon c_{i-1} n^{j-i}$.

Pivots

For each j-set J arising from either a new start at I or a jump to I, we start a new *pivot process* consisting of J and all those descendants of J which contain I. It is important to note that, while we now have a BFS rather than a DFS process, we are still constructing paths (albeit many simultaneously) and therefore the number of consecutive pivots at I is at most $\frac{k-i}{k-j} \leqslant k$. In other words, each pivot process runs for at most k generations.

Furthermore, the number of queries made from each j set in the pivot process is at most $\binom{n}{k-j}$, and therefore the expected number of pivots discovered from each j-set is at most $p\binom{n}{k-j} \leqslant \frac{2d}{\binom{k-j}{a}}$. Since each pivot gives rise to at most $\binom{k-j}{a}$ many j-sets, the expected growth factor from one generation to the next is at most $2d$, and so the expected total size of each pivot process is at most $\sum_{i=0}^{k}(2d)^i \leqslant (2d)^{k+1}$.

Now since we start at most $(\log n)^2 + 2^{i+2}d(k-j)!\varepsilon c_{i-1}n^{j-i} \leqslant \frac{\varepsilon c_i}{(2d)^{k+2}}n^{j-i}$ pivot processes (the sum of the contributions from new starts and jumps), the expected total size of all these pivot processes is at most $\frac{\varepsilon c_i}{2d}n^{j-i} \gg \sqrt{n}$. Once again, the Chernoff bound (Lemma 4) shows that with probability at least $1-\exp(-\sqrt{n})$, the total size is at most twice this (upper bound on the) expectation, or $\frac{\varepsilon c_i}{d}n^{j-i}$.

Now the total degree is in fact precisely the total size of all the pivot processes (the new starts and jumps simply tell us how many of these pivot processes there are), so assuming all the high probability events above hold, the degree of I is at most $\frac{\varepsilon c_i}{d}n^{j-i} \leqslant \varepsilon c_i n^{j-i}$, as required.

It remains to observe that the error probability was always at most $\exp(-\sqrt{n})$, and taking a union bound over all $\binom{n}{i} = o(\exp(\sqrt{n}))$ choices for I completes the argument. □

Now combining Claim 14, Proposition 15 and Lemma 16 immediately gives the following corollary.

Corollary 17. *Assuming that the properties of BFS Setup hold, whp stopping condition (S4) is invoked first.* □

6.3 Proof of Lemma 8

We can now use this corollary to prove Lemma 8.

As described above, let $U, P_0, \mathcal{J}_1, \mathcal{J}_2, \mathcal{D}, \{P_J : J \in \mathcal{J}_1, \mathcal{J}_2\}$ be the outputs of Lemma 12. We will assume that the high probability event of that lemma is satisfied, so we have properties (A1)-(A5) and (B1)-(B3).

We first run the `Pathbranch` algorithm with input $k, j, \mathcal{H} = \mathcal{H}^k(n,p), P_0, \mathcal{J} = \mathcal{J}_1, \{P_J : J \in \mathcal{J}_1\}$ and with forbidden vertex set $\mathcal{F}_1 = \emptyset$ and forbidden j-sets $\mathcal{F}_j = \mathcal{D}$. Then (C1) and (C6) are trivially satisfied, while Property (B3) and our choice of $\mathcal{F}_j = \mathcal{D}$ ensures that (C8) is satisfied. The remaining conditions of BFS Setup follow from our assumption that the high probability event of Lemma 12 is satisfied.

Let $\mathcal{A}$ be the resulting outcome of $\mathcal{G}_{\text{disc}}(t) = \mathcal{Z}_t \cup \mathcal{E}_t$ at the stopping time $t = T_{\text{stop}}$.

We now aim to run the algorithm again, with $\mathcal{J}_2$ in place of $\mathcal{J}_1$. However, it is in theory possible that all of the augmenting paths ending in a j-set of $\mathcal{A}$ share some common vertex x, and that the same happens when we construct augmenting paths at the other end, meaning that all pairs of paths will be incompatible. Of course, intuitively this is very unlikely. To formalise this intuition, we will make use of our ability to forbid a set $\mathcal{F}_1$ of vertices, which we did not need to do in the first iteration.

Let us define a *heavy* vertex to be a vertex which does not lie in P_0, but which lies in at least $\varepsilon^2(\log n)^5 n^{j-1}$ many augmenting paths P_J, where $J \in \mathcal{A}$ (i.e. which lies in at least a $(\log n)^5/n$ proportion of the augmenting paths).

Claim 18. *Whp there are at most $\delta^2 n$ heavy vertices.*

Proof. Let q be the number of pairs (v, P) consisting of a heavy vertex v and an augmenting path P containing v. We will estimate q in two different ways.

By (S4), there are at most $\varepsilon^2 n^j$ choices for P, each of which contains at most $j + 3(\log n)^4(k-j) \leqslant 3k(\log n)^4$ vertices due to (S2), and therefore certainly at most $3k(\log n)^4$ heavy vertices v, which implies that $q \leqslant \varepsilon^2 n^j \cdot 3k(\log n)^4$.

On the other hand, letting h denote the number of heavy vertices, we have $q \geqslant h \cdot \varepsilon^2(\log n)^5 n^{j-1}$ by the definition of a heavy vertex.

Combining these two estimates, we deduce that $h \leqslant 3kn/(\log n) \leqslant \delta^2 n$, as required. $\square$

We now run the algorithm again, this time with input $\mathcal{J} = \mathcal{J}_2$ and with $\mathcal{F}_1$ being precisely the set of heavy vertices, with $\mathcal{F}_j = \mathcal{D} \cup \mathcal{A}$ and with all other inputs as before. Note that our choice of $\mathcal{F}_j$ ensures that we never query a k-set which was already queried in a previous search process (either DFS or BFS).

Most of the properties (C1)-(C7) are true for the same reason as before – we now also need to observe that (C6) holds whp because of Claim 18, while (C7) follows from (B3) and (S3) because $\Delta_i(\mathcal{F}_j) \leqslant \Delta_i(\mathcal{D}) + \Delta_i(\mathcal{A})$.

Let $\mathcal{B}$ be the resulting outcome of $\mathcal{G}_{\text{disc}}(t) = \mathcal{Z}_t \cup \mathcal{E}_t$ at the stopping time. We claim that whp P_0, $\mathcal{A}$ and $\mathcal{B}$ satisfy the conditions of Lemma 8.

First, recall that P_0 has length $(1-\delta)L_1$ since it was provided by Lemma 12. Next, observe that by Corollary 17 we have $|\mathcal{A}|, |\mathcal{B}| = \varepsilon^2 n^j$ whp.

Recall that $\hat{J}_1, \hat{J}_2$ are the two end j-sets of P_0, and we set $J_s := \hat{J}_1$ and $J_e := \hat{J}_2$. Recall also that for each $A \in \mathcal{A}$ we have a path P_A ending in A which was constructed during the first application of the `Pathbranch` algorithm. Let us define P_{A,J_s} to be the path $P_A \setminus P_0$ (where for the purposes of this notation, the paths are equated with their edge sets). We define P_{B,J_e} similarly for each $B \in \mathcal{B}$.

Note that because of the stopping condition (S2), P_{A,J_s} has length at most $3(\log n)^4 \leqslant \delta n/3$, and therefore it is also clearly true by construction that P_{A,J_s} path-$(A, \delta n/3)$-augments (P_0, J_s). Similarly, for every j-set $B \in \mathcal{B}$ the path P_{B,J_e} path-$(B, \delta n/3)$-augments (P_0, J_e).

Finally, we show the trickiest of the properties, that for most pairs $(A, B) \in \mathcal{A} \times \mathcal{B}$ the augmenting paths are disjoint. Recall that $\mathscr{P}_\mathcal{A} = \{P_{A,J_s} : A \in \mathcal{A}\}$ and $\mathscr{P}_\mathcal{B} = \{P_{B,J_e} : B \in \mathcal{B}\}$ denote the sets of these augmenting paths. Recall that for each $B \in \mathcal{B}$ the augmenting path P_{B,J_e} has length at most $3(\log n)^4$, and therefore contains $O\left((\log n)^4\right)$ vertices. Since we excluded heavy vertices when constructing $\mathcal{B}$, any vertex in P_{B,J_e} lies in at most $\varepsilon^2(\log n)^5 n^{j-1}$ of the paths $\mathscr{P}_{\mathcal{A}}$, and therefore each path of $\mathscr{P}_{\mathcal{B}}$ intersects with at most $O\left(\varepsilon^2(\log n)^9 n^{j-1}\right) = O\left(\frac{(\log n)^9}{n}|\mathcal{A}|\right)$ of the paths in $\mathscr{P}_{\mathcal{A}}$. Therefore the number of pairs $(A, B) \in \mathcal{A} \times \mathcal{B}$ such that the paths P_{A,J_s}, P_{B,J_e} are vertex-disjoint is at least

$$|\mathcal{B}|\left(1 - O\left(\frac{(\log n)^9}{n}\right)\right)|\mathcal{A}| \geqslant (1-\varepsilon)\varepsilon^4 n^{2j},$$

as required.

7 Concluding remarks

7.1 Upper bounds

In this paper we proved a lower bound (whp) on the length L_C of the longest cycle in $\mathcal{H}^k(n,p)$. One can obtain an upper bound (whp) on L_C by applying a first moment method. When $p = dp_0$ for some $1 < d < e^{k-j}$, some careful but elementary calculation gives that $L_C \leqslant (1+o(1))\frac{xn}{k-j}$, where x is the unique positive solution to the equation $d^{\frac{x}{k-j}} e^{-x}(1-x)^{x-1} = 1$. By contrast, our lower bound is approximately $\frac{1-d^{-1/(k-j)}}{k-j} \cdot n$, and the constant factor can be seen to be strictly smaller than $\frac{x}{k-j}$ for any constant $d > 1$. Indeed for d very close to 1, the upper bound is approximately twice the lower bound, and while this multiplicative gap decreases to $1 - 1/e$ as d approaches e^{k-j}, it does not disappear entirely. It would be interesting to close the gap and thus determine L_C precisely.

We note that when $d = e^{k-j}$ the unique positive solution is $x = 1$, corresponding to a Hamilton cycle, and indeed this was proved to be the threshold for a Hamilton cycle by Narayanan and Schacht [15], strengthening earlier results of Dudek and Frieze [11]. This provides some evidence that perhaps the upper bound is the correct one. However, the length L_1, where the search process becomes subcritical, represents a natural barrier to the constructive approach presented here, and it seems that new ideas would be required to go beyond it.

7.2 Critical window

As observed in the introduction, the lower bound for paths proved in [4] allowed for $p = (1+\varepsilon)p_0$, where $\varepsilon > 0$ may tend to zero sufficiently slowly. Our sprinkling argument to close the path to a cycle would also allow this. However certain aspects of the proof were non-optimal, meaning we are unlikely to get the best possible conditions on ε, therefore for simplicity we did not attempt this. Nevertheless, a more careful version of the argument might indeed provide the best possible conditions on ε, which would describe the width of the critical window.

7.3 Further structures

It is natural to ask whether the approach used in this paper could also be applied to find further large structures in random hypergraphs. The search process for j-tight paths is relatively simple because of their regular periodic nature, meaning that the algorithm behaves identically at every step, so what other structures might also be tractable?

One structure with a similar nature is the natural analogue of the $(m-k+1)$-*th power* of a tight cycle of length ℓ in a k-uniform hypergraph, in which for some integer $m \geqslant k$, every set of k vertices within an interval of length m around a cyclic ordering of ℓ vertices forms an edge. (The case $m = k$ is simply a tight cycle.) Let $L_C^* = L_C^*(n, m, k, p)$ denote the length of the longest $(m-k+1)$-th power of a cycle in $\mathcal{H}^k(n, p)$. It seems likely that the techniques of this paper could be adapted to prove the following analogue of Theorem 2.

Conjecture 19. *Let $k, m \in \mathbb{N}$ satisfy $3 \leqslant k \leqslant m-1$ and let $p_0 = p_0(n, k, m) := n^{-\frac{1}{\binom{m-1}{k-1}}}$.*

For any $\delta > 0$, for any constant $d > 1$ and for any sequence $(d_n)_{n\in\mathbb{N}}$ satisfying $d_n \to d$ the following is true. Suppose that $p = d_n p_0$. Then whp

$$L_C^* \geqslant (1-\delta) \cdot \left(1 - d^{-\binom{m-1}{k-1}}\right) \cdot n.$$

One might also consider, for example, cycles in which the size of the intersection of two consecutive edges alternates between two different values, or any obvious generalisation of this idea. Both for this structure and for Conjecture 19 we suspect that the main difficulties in adapting the proof of Theorem 2 would be purely technical, but we have not attempted this.

More generally, the idea of branching out to search many possible ways of closing a cycle-like structure would likely be helpful whenever we are looking for a subgraph which is both *non-tree-like* in the sense that some long cyclic structure exists, and *non-linear* in the sense that edges intersect in more than one vertex.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments and suggestions, which have greatly improved this paper.

Supported by Austrian Science Fund (FWF): I3747.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi, *The longest path in a random graph*, Combinatorica **1** (1981), 1–12.

[2] M. Anastos, *An improved lower bound on the length of the longest cycle in random graphs*, 2022, arXiv: 2208.06851.

[3] M. Anastos and A. Frieze, *A scaling limit for the length of the longest cycle in a sparse random graph*, J. Combin. Theory Ser. B **148** (2021), 184–208. MR 4200751

[4] O. Cooley, F. Garbe, E. K. Hng, M. Kang, N. Sanhueza-Matamala, and J. Zalla, *Longest paths in random hypergraphs*, SIAM J. Discrete Math. **35** (2021), no. 4, 2430–2458. MR 4325741

[5] O. Cooley, M. Kang, and C. Koch, *The size of the giant high-order component in random hypergraphs*, Random. Struct. Algor. **53** (2018), no. 2, 238–288.

[6] ———, *The size of the giant component in random hypergraphs: a short proof*, Electron. J. Combin. **26** (2019), no. 3, #P3.6. MR 3982315

[7] O. Cooley, M. Kang, and Y. Person, *Largest components in random hypergraphs*, Combin. Probab. Comput. **27** (2018), no. 5, 741–762.

[8] O. Cooley, M. Kang, and J. Zalla, *Loose Cores and Cycles in Random Hypergraphs*, Electron. J. Combin. **29** (2022), no. 4, #P4.13. MR 4498685

[9] N. Draganić, S. Glock, and M. Krivelevich, *The largest hole in sparse random graphs*, Random Structures & Algorithms **61** (2022), no. 4, 666–677.

[10] A. Dudek and A. Frieze, *Loose Hamilton cycles in random uniform hypergraphs*, Electron. J. Combin. **18** (2011), no. 1, #P48. MR 2776824

[11] ———, *Tight Hamilton cycles in random uniform hypergraphs*, Random Structures Algorithms **42** (2013), no. 3, 374–385. MR 3039684

[12] S. Janson, T. Łuczak, and A. Ruciński, *Random graphs*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, New York, 2000. MR 1782847 (2001k:05180)

[13] G. Kemkes and N. Wormald, *An improved upper bound on the length of the longest cycle of a supercritical random graph*, SIAM J. Discrete Math. **27** (2013), no. 1, 342–362. MR 3032923

[14] T. Łuczak, *Cycles in a random graph near the critical point*, Random. Struct. Algor. **2** (1991), no. 4, 421–439. MR 1125957

[15] B. Narayanan and M. Schacht, *Sharp thresholds for nonlinear Hamiltonian cycles in hypergraphs*, Random Structures Algorithms **57** (2020), no. 1, 244–255. MR 4120599