IOP Publishing J.Phys.Complex. 2 (2021) 035004 (14pp) https://doi.org/10.1088/2632-072X/abdb7d

Journal of Physics: Complexity

OPEN ACCESS

PAPER

Planted hitting set recovery in hypergraphs

Ilya Amburg*, Jon Kleinberg and Austin R Benson

Cornell University, United States of America

* Author to whom any correspondence should be addressed.

E-mail: ia244@cornell.edu, kleinber@cs.cornell.edu and arb@cs.cornell.edu

RECEIVED
12 August 2020

ACCEPTED FOR PUBLICATION
13 January 2021

PUBLISHED
5 May 2021

Keywords: network analysis, higher-order network analysis, hitting sets, combinatorial optimization

Abstract

In various application areas, networked data is collected by measuring interactions involving some specific set of *core* nodes. This results in a network dataset containing the core nodes along with a potentially much larger set of fringe nodes that all have at least one interaction with a core node. In many settings, this type of data arises for structures that are richer than graphs, because they involve the interactions of larger sets; for example, the core nodes might be a set of individuals under surveillance, where we observe the attendees of meetings involving at least one of the core individuals. We model such scenarios using *hypergraphs*, and we study the problem of core recovery: if we observe the hypergraph but not the labels of core and fringe nodes, can we recover the 'planted' set of core nodes in the hypergraph? We provide a theoretical framework for analyzing the recovery of such a set of core nodes and use our theory to develop a practical and scalable algorithm for core recovery. The crux of our analysis and algorithm is that the core nodes are a *hitting set* of the hypergraph, meaning that every hyperedge has at least one node in the set of core nodes. We demonstrate the efficacy of our algorithm on a number of real-world datasets, outperforming competitive baselines derived from network centrality and core-periphery measures.

1. Core and fringe nodes in networks

The data that we collect in practice is typically incomplete in several fundamental and recurring ways, and this is particularly true for graph and network data modeling complex systems throughout the social and biological sciences [29, 33, 40, 42, 43]. One common type of incompleteness arises from a particular data collection procedure where one wants to record all interactions involving a set of core nodes C. This results in what is sometimes called a 'core-fringe' structure [6], where the resulting network structure contains the core nodes along with a (potentially) much larger set of fringe nodes F that are observed in some interaction with a node in C. For example, in survey data, C might represent a set of respondents, and F a set of individuals with whom they interact; this is a common scenario in snowball and respondent-driven sampling [29, 31, 35]. Another situation arises in restrictions on surveillance. For example, the Enron [39] and Avocado[1] email datasets are common data sources for email and networks research. Both datasets come from collecting emails involving a core C representing the employees of a company. Through this process, the data includes emails involving a fringe F of people outside the company who receive email from or send email to members of C. However, we would not have access to emails sent between people outside the company. We will see in our data analysis later that the size of the fringe is often much larger than the size of the core in these datasets.

One can imagine similar scenarios in a variety of settings. In intelligence data, one might record all of the attendees at meetings that involve at least one of a set of individuals under surveillance. In a similar manner, telecommunications providers can observe group text messages where only some of the members of the group are subscribers. Furthermore, measurements of Internet structure from service providers at the IP-layer from individual service providers provide only a partial measurement [56], and large-scale Web crawls do not encapsulate the entire network structure of hyperlinks [7, 8].

[1] https://catalog.ldc.upenn.edu/LDC2015T03.

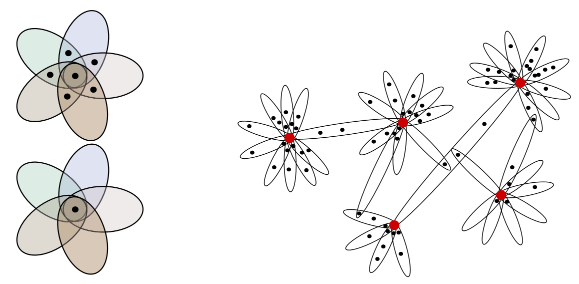

Figure 1. Hypergraphs and planted hitting sets. (Left) a three-uniform hypergraph on the top and a minimal hitting set on the bottom (with hyperedges superimposed to facilitate comparison to the hypergraph). (Right) a planted hitting set (red nodes) in a hypergraph, constituting the *core* nodes. Hyperedges are illustrated by ellipses encompassing nodes, which include both core and non-core (*fringe*) nodes. We study how well we can recover the core (red) nodes if they are not identified.

While a dataset may be equipped with labels of which nodes are members of the core and which are members of the fringe, there are several situations in which such labels are unavailable. For example, consider the email datasets described above. Such data could be released by a hacker from outside the organization, or a leaker from inside it, who has collected email from a set of core accounts but not released the identity of the accounts [36]. Similarly, in the case of intelligence data that records attendees of meetings, the data could be released with the identities of the individuals under surveillance redacted. In other cases, the metadata may be lost simply to issues of data maintenance—such concerns are central to the research areas of data provenance and preservation [13, 45, 53, 55].

The question of recovering the identity of the core nodes in a dataset with core-fringe structure is therefore a fundamental question arising in different applications, and for multiple different reasons.

Core-fringe recovery in hypergraphs. Here, we study the problem of identifying the set of core nodes, when the labels of which nodes are core and which are fringe are unavailable and the data is represented by a *hypergraph*, i.e., a set of nodes along with a set of *hyperedges* that each connect multiple nodes. Figure 1 illustrates this model. The hypergraph model is apt for many of the scenarios described above, including emails, meeting attendees, and group messages. While the hypergraph model is richer than a more standard graph model, the higher-order structure also presents challenges. If a hyperedge connects a (latent) core node and several fringe nodes, an algorithm trying to identify the core has to consider all nodes in the hyperedge as possible candidates.

We first provide theoretical results that motivate the development of an algorithm for recovery of the core. The key structural property of our analysis is that the core nodes constitute a *hitting set*, meaning every hyperedge in the hypergraph has at least one node in the core (figure 1). This property comes from how the data is measured—every included hyperedge comes from an interaction of a member of the core with some other set of nodes. We thus think of the core as a 'planted' hitting set that we must find in the data. Our first theoretical results rely on two assumptions (both relaxable, to some extent): the planted hitting set corresponding to the core is (i) minimal, that is, no node could be removed to make the hitting set smaller; and (ii) relatively small, i.e., bounded by a constant k. We analyze how large the *union of all minimal hitting sets*, denoted $U(k)$, can be. We build upon results in fixed-parameter tractability [18, 19] to show that $|U(k)|$ is $\Theta(k^r)$ in the worst case, where r is the size of the largest hyperedge. Importantly, this bound is independent of the hypergraph's size, and $U(k)$ is guaranteed to contain our planted hitting set if it meets the modeling assumptions.

Furthermore, we prove that a classical greedy approximation algorithm for set cover relates to partial recovery of the planted hitting set, even in cases where the above assumptions do not hold. Specifically, in a hypergraph where r is the size of the largest hyperedge, we show that the output of the algorithm must overlap with the planted hitting set by at least a $O(1/r)$ fraction of the nodes, provided that the hitting set size is within a constant factor of the minimum hitting set size.

Motivated by these two main results, we design a core recovery algorithm, which we call the union of minimal hitting sets (UMHS). This algorithm is practical and scalable, using only simple subroutines and having computational complexity roughly linear in the size of the data. The idea is to run the greedy algorithm multiple times with different random initializations to find several minimal hitting sets and then to simply take the union of these hitting sets. The first part of our theory says that the output should not grow too large, and the second part of our theory says the output should also overlap the planted hitting set. We show that our method consistently outperforms a number of baselines derived from network centrality and core-periphery structure for hypergraphs on several real-world datasets.

Our approach to the planted hitting set problem is purely combinatorial. This contrasts with typical planted problems such as the stochastic block model (SBM) [1, 21, 48] and planted clique recovery [2, 23], which are based on probabilistic structure. Thus, the methods we use are fundamentally different than what is common for these types of problems. Moreover, we show that if random block-model-type structure exists for our problem, then the problem becomes substantially easier.

2. Problem setup and theoretical results for core recovery

We start with some formal definitions to facilitate our presentation of the theoretical results. A hypergraph G consists of a set of nodes V and a set of hyperedges E, each a subset of V of size at least two. The *rank* r of a hypergraph is the maximum size of any hyperedge, i.e., $r = \max_{e\in E}|e|$. A hypergraph is called *r-uniform* if all hyperedges have cardinality equal to r.

In our setup, there is some unidentified subset $C \subseteq V$ that is designated as the set of 'core nodes.' Our goal is to find C, knowing that C is a hitting set of G (i.e., for every $e \in E$, there is some $v \in e \cap C$). We say that C is *planted* since we do not know its identity. Absent any additional information, we are helpless—it could be that $C = V$. However, we can do better by assuming two properties of C (both relaxable): (i) C is a minimal hitting set in G and (ii) C is bounded in size, i.e., $|C| \leqslant k$. Under these constraints, it is certainly true that C is contained in the union of *all* minimal hitting sets having at most k nodes, which is a principle object of our analysis:

Definition 1. Given a hypergraph G, $U(k)$ is the union of all minimal hitting sets of G of size at most k.

We next show bounds on $|U(k)|$ that are independent of the size of the graph. This says that C is contained in a relatively small set if it satisfies the conditions. Furthermore, we can find $U(k)$ in time exponential in k but polynomial in the size of the graph [18]; however, we develop more practical algorithms in section 3. After, we relax the assumptions on minimality and get results on *partial recovery*, i.e., algorithms that are guaranteed to find a part of C. Finally, we show how these results improve under a random hypergraph model.

2.1. Minimal hitting sets

First let us suppose that we are given G and asked to find C, where we know that C is minimal and $|C| \leqslant k$. We ask the following question: is it possible to find a small set $H \subset G$, whose size is independent of G, such that $C \subset H$? In this section, we answer the question in the affirmative.

We at least know that the union of *all* minimal hitting sets of size at most k, denoted $U(k)$ (definition 1), *must* contain C. But how large can this set be? The following result says that $U(k)$ actually cannot be too large.

Lemma 2. *In a hypergraph of rank r, $U(k)$ has size $\Theta(k^r)$ in the worst case.*

Damaschke and Molokov established most parts of this result using sophisticated techniques from parameterized complexity theory [18, 19]. Chitnis *et al* independently established similar (albeit looser) results for streaming data applications [14]. Here, we provide a concrete, complete characterization and proof that should be accessible to anyone familiar with basic graph theory by using the celebrated sunflower lemma of Erdös and Rado [25]. A consequence of lemma 2 is the following theorem about how small of a set we can find that is guaranteed to contain the core:

Theorem 3. *Let C be a planted minimal hitting set with $|C| \leqslant k$ in a hypergraph of rank r. Then we can find a set D of size $O(k^r)$ that is guaranteed to contain C.*

We begin by providing a self-contained proof of the upper bound in lemma 2 and theorem 3. Namely, let G be a hypergraph of rank r, and let $U(k)$ be the union of all minimal hitting sets in G of size at most k. We will find a function $g(r,k) = O(k^r)$ so that $|U(k)| \leqslant g(r,k)$. To find such a function, we use a combinatorial structure called a sunflower [10]. A collection of r sets $\{X_1, X_2, \dots, X_r\}$ is an r-sunflower if for all pairs of distinct sets X_i, X_j, the pairwise intersection $X_i \cap X_j$ is equal to the mutual intersection $\bigcap_{i=1}^{r} X_i$. The term comes from the fact that we can think of the X_i as the 'petals' of the sunflower: they share a 'center' $\bigcap_{i=1}^{r} X_i$ and are otherwise pairwise disjoint from each other. A famous theorem of Erdös and Rado asserts that every sufficiently large collection of sets must contain a large sunflower.

Lemma 4 (sunflower lemma [25]). *Define the function $\sigma(r,k) = r!(k-1)^r$. Any collection of more than $\sigma(r,k)$ sets, each of size at most r, must contain a k-sunflower.*

We use this result to find the desired function $g(r,k) = O(k^r)$. Later, we will establish an asymptotically matching lower bound. We now state and prove the upper bound.

Lemma 5. *Let G be a hypergraph of rank r. Then $|U(k)| = O(k^r)$.*

Proof. We assume that G has at least one hitting set of size at most k, as otherwise the statement of lemma 5 holds with $U(k)$ equal to the empty set. We modify G according to the following iterative procedure, which operates in discrete phases. We start by defining $H_0 = G$, and produce a succession of hypergraphs $H_1, H_2, \ldots$, with H_t the hypergraph at the end of phase t. For each hypergraph H_t, let $C(H_t)$ denote the set of all hitting sets of size at most k in H_t, and let $\mathcal{C}^*(H_t)$ denote the set of all minimal hitting sets of size at most k in H_t. We will establish by induction that each hypergraph H_t has at least one hitting set of size at most k.

At the start of phase t, we ask whether H_{t-1} has more than $\sigma(r, k+1)$ hyperedges, where σ is the function from lemma 4. If it does not, we declare the procedure to be over. If it does, then we find a $(k+1)$-sunflower in the set of hyperedges, consisting of hyperedges $X_{t,1}, X_{t,2}, \ldots, X_{t,k+1}$.

Now, the core of the sunflower found in phase t, which we will denote by $Y_t = \bigcap_{i=1}^{k+1} X_{t,i}$ must be non-empty, for if it were empty, then H_{t-1} would contain $k+1$ pairwise disjoint hyperedges, and hence could not have a hitting of size at most k. We define H_t as follows: starting from H_{t-1}, we remove the hyperedges $X_{t,1}, X_{t,2}, \ldots, X_{t,k+1}$ and add the hyperedge Y_t. We observe the following properties.

(a) Every hitting set in H_{t-1} of size at most k must intersect Y_t.
(b) $C(H_{t-1}) = C(H_t)$, and hence $\mathcal{C}^*(H_{t-1}) = \mathcal{C}^*(H_t)$.

To see property (a), by the definition of a sunflower, the sets $X_{t,i} \backslash Y_t$ are pairwise disjoint for $i = 1, 2, \ldots, k+1$. Thus, if C is a set of size at most k that does not contain any nodes of Y_t, then it must be disjoint from at least one of the sets $X_{t,i}$, and hence it cannot be a hitting set for H_{t-1}. The definition of H_t is as follows: starting from H_{t-1}, we remove the hyperedges $X_{t,1}, X_{t,2}, \ldots, X_{t,k+1}$ and add the hyperedge Y_t. Since H_{t-1} and H_t are hypergraphs on the same node sets, their respective collections of hitting sets $C(H_{t-1})$ and $C(H_t)$ are also over the same sets of elements.

Now, let $C \in \mathcal{C}(H_{t-1})$. Then by property (a), it must intersect Y_t. Since Y_t is the only hyperedge in H_t but not H_{t-1}, it follows that C is a hitting set for H_t, and hence $C \in \mathcal{C}(H_t)$. Conversely, let $C' \in \mathcal{C}(H_t)$. Since $Y_t \subset X_{t,i}$ for $i = 1, 2, \ldots, k+1$, it must be that C_0 intersects each $X_{t,i}$. Since $X_{t,1}, X_{t,2}, \ldots, X_{t,k+1}$ are the only hyperedges in H_{t-1} but not H_t, it follows that $C \in \mathcal{C}(H_{t-1})$. Thus, $\mathcal{C}(H_{t-1}) \subset \mathcal{C}(H_t) \subset \mathcal{C}(H_{t-1})$, and so they are equal as sets.

We apply the same process in each phase, producing H_1 from $G = H_0$, then producing H_2 from H_1 and so forth. By induction using property (b), since H_{t-1} contains at least one hitting set of size at most k, so does H_t, as required. Since the procedure reduces the number of hyperedges in each phase, it must eventually terminate, in some phase t^*. We write $H^* = H_{t^*}$ and by applying property (b) transitively,

$$\mathcal{C}^*(H) = \mathcal{C}^*(H^*). \tag{1}$$

We say that a node is isolated if it does not belong to any hyperedge. Let T^* be the set of non-isolated nodes of H^*. Since the procedure stopped when faced with H^*, H^* had at most $\sigma(r, k+1)$ hyperedges. Since each hyperedge has most r elements, we have

$$|T^*| \leqslant r\sigma(r, k+1) = r \cdot r!k^r, \tag{2}$$

and hence $|T^*|$ is bounded by $O(k^r)$. Finally, recall that $U(k)$ is the union of all minimal hitting sets in G of size at most k. We can write $U(k) = \bigcup_{C \in \mathcal{C}^*(H)} C$, and by equation (1), $U(k) = \bigcup_{C \in \mathcal{C}^*(H^*)} C$. No minimal hitting set C in H^* can contain an isolated node v, since then $C - \{v\}$ would also be a hitting set in H^*. From this, $U(k) \subset T^*$, and $|U(k)| \leqslant |T^*| = O(k^r)$ by equation (2). □

Next, we show asymptotic tightness for constant hypergraph rank r, by constructing a family of hypergraphs for which $|U(k)| = \Omega(k^r)$. The following lemma combined with lemma 5 proves theorem 3.

Lemma 6. *For each constant $r \geqslant 2$, there exists an infinite family of rank-r hypergraphs G and parameters k, with both the number of nodes in G and k going to infinity, for which $|U(k)| = \Omega(k^r)$.*

Proof. For a parameter b, consider a graph Γ_b that consists of b disjoint complete b-ary trees of depth r. We will refer to the roots of the b trees in Γ_b as the root nodes of Γ_b, and to the leaves of the trees in Γ_b as leaf nodes. Let T_b be the hypergraph on the node set of Γ_b whose hyperedges consist of all root-to-leaf paths in Γ_b, from the root of one of the b trees in Γ_b to a leaf of the corresponding tree (the example of T_2 for $r = 3$ is in figure 2). All edges in G have size r. Let λ be any function that maps nodes of Γ_b to numbers in $\{1, 2, \ldots, b\}$ with the following properties: (i) λ is a bijection from the roots of Γ_b to $\{1, 2, \ldots, b\}$; and (ii) for each non-leaf node v in Γ_b, λ is a bijection from the children of v to $\{1, 2, \ldots, b\}$. For each node v, we will call $\lambda(v)$ its label (with respect to λ), and we will say that λ is a consistent labeling if it satisfies (i) and (ii). Let v^*_λ be unique leaf node for which all the nodes on the root-to-leaf path have the label b.

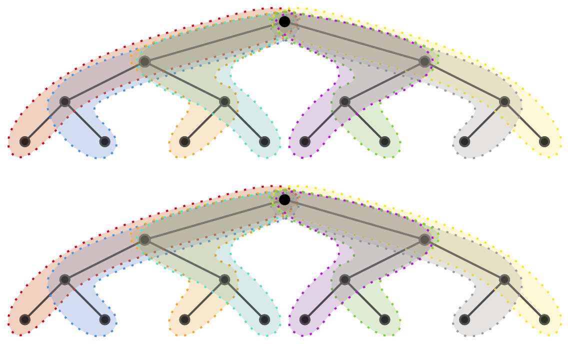

Figure 2. The hypergraph T_2 ($r = 3$) used in the proof of lemma 6 with hyperedges identified by dotted shapes.

For a consistent labeling λ, let C_λ be the set of all nodes v satisfying two properties: (i) $\lambda(u) = b$ for all nodes u on the unique path from a root of Γ_b to v (other than v itself), and (ii) $\lambda(v) \neq b$. Let $C^*_\lambda = C_\lambda \cap \{v^*_\lambda\}$. We observe the following two facts.

(a) For every consistent labeling λ, the set C^*_λ is a minimal hitting set for T_b.

(b) Every node v of T_b belongs to at least one set of the form C^*_λ for some consistent labeling λ.

To prove (a), we claim that C^*_λ is a hitting set. Consider any hyperedge X of T_b, consisting of a root-to-leaf path P in Γ_b. If all nodes on P have label b, then the leaf node of P is v^*_λ, and hence X intersects C^*_λ. Otherwise, consider the first node v on the path P that has a label unequal to b. Thus, $v \in C^*_\lambda$, and again X intersects C^*_λ and C^*_λ is a hitting set. We now argue that it is minimal. If we delete v^*_λ from C^*_λ, then $C^*_\lambda \backslash \{v^*_\lambda\}$ would be disjoint from the hyperedge X consisting of the root-to-leaf path to v^*_λ. If we delete some other v^*_λ, then a hyperedge X that passes through v does not pass through any other node of C^*_λ on its way from root to leaf, and so $C^*_\lambda \backslash \{v\}$ would be disjoint from X.

To prove (b), we simply choose any consistent labeling λ such that $\lambda(u) = b$ for all nodes u on the unique path from a root of Γ_b to v (other than v itself), and $\lambda(v) \neq b$.

Now, we consider the size of a set of the form C^*_λ. Since C^*_λ contains $b - 1$ nodes from each of the first $r - 1$ levels of the trees in Γ_b, and b from the lowest level, we have

$$|C^*_\lambda| = (r - 1)(b - 1) + b.$$

We define $k = (r - 1)(b - 1) + b$. Since each set C^*_λ is a minimal hitting set of size k, from (b) we see that the union $U(k)$ of all minimal hitting sets of size k is the entire node set of T_b.

The number of nodes in T_b is greater than b^r, and hence $|U(k)| > b^r$. Since $k \leqslant br$, we have that $|U(k)| > b^r \geqslant (k/r)^r = r^{-r}k^r$. □

These theoretical result establish justification for why a set of core nodes might be identifiable—if the core is a minimal hitting set, it is certainly contained in a set that has size independent of the graph. Next, we consider the case when the core may not be a minimal hitting set.

2.2. Non-minimal hitting sets

Here we present results on which types of nodes in a planted hitting set C must be contained in $U(|C|)$ when the hitting set is itself not minimal. This matters in practice, when C may not be minimal.

2.2.1. Greedy matchings intersect the planted hitting set

One first results are based on finding hitting sets with a greedy algorithm for the set cover problem. The classical algorithm is simple (algorithm 1): loop through each hyperedge and if no vertex in the current hyperedge is in the current cover S, add all of the nodes in the hyperedge to S. The greedy algorithm produces a set that is both a hitting set and a maximal matching. To see that it is a hitting set, suppose that all nodes in a hyperedge h were not added. Then h would have been added to S at the time the algorithm processed it. The output is a maximal matching because if we could append another hyperedge h to S, then h would have already been added to S when the algorithm processed it.

Let k^* be the *minimum* hitting set size (the smallest size of all minimal hitting sets). The output of algorithm 1 is an r-approximation if the hypergraph G is an r-uniform hypergraph: in the worst case, any hitting set contains at least one of r nodes from each hyperedge, and we therefore have $k^* \geqslant |S|/r$. It turns out that this

Algorithm 1. Maximal matching r-approximation to the minimum hitting set that intersects the core.

```
1: Input: hypergraph $G = (V, E)$ of rank $r$
2: Output: set cover $S$ with $|S| \leqslant rk^*$.
3: $S \leftarrow \emptyset$
4: for $(u_1, \dots, u_i) \in E$ do
5:     if $u_1, \dots, u_i \notin S$ then
6:         $S \leftarrow S \cup \{u_1, \dots, u_i\}$
7:     end if
8: end for
9: return $S$
```

greedy algorithm must also partially overlap with the planted hitting set. We formalize this in the following lemma.

Lemma 7. *Let $|B| \leqslant bk^*$ for some hitting set B. If the input to algorithm 1 is a rank-r hypergraph, then the output S satisfies*

$$|B \cap S|/|B| \geqslant \frac{1}{rb}.$$

Proof. Let r_1 be the number of vertices in the hyperedge containing the least vertices. The set S contains within itself all vertices from h hyperedges where h satisfies $h \leqslant |S|/r_1 \leqslant rk^*/r_1$. Since B is a hitting set, it must contain at least one vertex from each of the h hyperedges in S. From this, we obtain $|S \cap B| \geqslant h \geqslant k^*/r_1 \geqslant \frac{1}{br}|B|$. □

Lemma 7 guarantees that the greedy algorithm will output a result that overlaps any hitting set, including the planted one. Thus, we immediately get the following corollary.

Corollary 8. *If the planted hitting set C has size $|C| \leqslant ck^*$ then the output of algorithm 1 is a set S with $|S \cap C|/|C| \geqslant \frac{1}{cr}$, i.e., S intersects at least a fraction of $\frac{1}{cr}$ of C.*

We can process the hyperedges in any order within algorithm 1, and this will be important for our practical recovery algorithm that we develop later. Another immediate corollary of lemma 7 says that regardless of the processing, the outputs must be fairly similar.

Corollary 9. *Any two outputs S_1 and S_2 of algorithm 1 satisfy $|S_1 \cap S_2| \geqslant \frac{1}{r^2}\max(|S_1|, |S_2|)$.*

2.2.2. Classes of nodes that must be in $U(|C|)$

Our next results are on nodes that must appear in $U(|C|)$. First, a node in C must be in $U(|C|)$ if there is a hyperedge containing u where all of the other nodes in the hyperedge are not in C.

Lemma 10. *Let G be a rank-r hypergraph with planted hitting setting C. If $u \in C$ and there exists a hyperedge $(u, v_1, \dots, v_i)$ with $i \leqslant r - 1$ with $v_1, \dots, v_i \notin C$, then $u \in U(|C|)$.*

Proof. We perform a pruning of C as follows. Check whether there exists a node w with $C \backslash \{w\}$ still being a hitting set of smaller cardinality. If this is the case, update C to $C \backslash \{w\}$ and continue pruning. Upon termination, the remaining set is a minimal hitting set $C' \subseteq C$. Clearly, $|C'| \leqslant |C|$ so that $C' \subset U(|C|)$. However, $u \in U(|C|)$ since $(u, v_1, \dots, v_i)$ is a hyperedge where $v_1, \dots, v_i \notin C$. □

The above lemma provides a class of nodes in a planted hitting set which are guaranteed to be in the UMHS of size no more than that of the planted hitting set itself. Although this lemma is purely combinatorial, it will be useful for analyzing random hypergraphs in section 2.3.

Next, we show that nodes adjacent to other nodes deeply integrated into C must also be in $U(|C|)$. We do not use this result anywhere else in the text, but we include it for theoretical interest and intuition about unions of minimal hitting sets. Formally, define a node v to be in the *interior* of a hitting set C if all hyperedges containing v are comprised entirely of nodes in C. Our next result is that a node in C must appear in $U(|C|)$ if all the other nodes in a hyperedge containing u are in the interior of C.

Lemma 11. *Let G be a rank-r hypergraph with planted hitting set C. If $u \in C$ and there exists a hyperedge $(u, v_1, \dots, v_i)$ where $v_1, \dots, v_i$ are in the interior of C, then $u \in U(|C|)$.*

Proof. Since $v_1, \dots, v_i$ are in the interior of C, $C_0 = C \backslash \{v_1, \dots, v_i\}$ is a hitting set as well. Now perform pruning on C_0 as in the previous proof, the output of which is $C' \subset C_0$, and since clearly $|C'| \leqslant |C|$, we must have that $C' \subset U(|C|)$. Clearly u was not deleted during pruning since $(u, v_1, \dots, v_i)$ is a hyperedge with $v_1, \dots, v_i \notin C$, so that $u \in C' \subset U(|C|)$. □

Thus far, we have made no assumptions on the structure of the hypergraph. In the next section, we show that assuming a stochastic-block-model-type hypergraph structure gives substantial improvements in recovery results.

2.3. Recovery in a random hypergraph model

In this section we show that if we make additional assumptions on the structure of the problem, it becomes much easier. Most research that studies guarantees on planted structure recovery in graphs (e.g., community detection or planted clique detection) already makes these types of assumptions in order to make the problem feasible. The fact that we can do anything without such assumptions is quite remarkable.

More specifically, we show here that under certain stochastic-block-model-like assumptions for three-uniform hypergraphs laid out in this section—*which would already be standard in other graph recovery problems*—we can reduce the size of the UMHS (and hence of the upper bound of the size of the output of our algorithm) from $O(k^3)$ in the general case to $O(k^2 \log k)$ in this special structured case.

We emphasize that our main results and the algorithms we use for analyzing data are purely combinatorial, making no assumption on this type of probabilistic structure. And the algorithms we develop later perform extremely well even when no such structure is assumed. Instead, we simply wish to highlight that some of our bounds can be improved under random hypergraph models. We also have an additional goal of introducing our problem to the rich literature on finding planted structure in random graphs.

Let us assume that the hyperedges are generated according to a SBM for hypergraphs [28]. One block will be the core C and the other the fringe nodes F. We assume that there is zero probability of a hyperedge containing only nodes in F. Explicitly, we will state that the probability of a hyperedge between nodes in C is p while the probability of a hyperedge containing at least one node in C and at least one node in F is q.

We first provide a lemma on the independence number of hypergraphs drawn from the hypergraph SBM. This will help us control the size of $|U(|C|)|$.

Lemma 12. *Let $\alpha(G)$ be the independence number of an r-uniform hypergraph G drawn from the SBM on n vertices. Then*

$$\Pr(\alpha(G) < k) \geqslant 1 - n^{-\frac{1}{2}\left(\frac{3r!\ln\, n}{2p}+(r-1)\right)}, \quad k = \frac{3r!\ln\, n}{2p} + (r-1)$$

Proof. We follow a proof technique common in the combinatorics literature [41]. Let $z = \binom{|G|}{k}$, $S_1, \ldots, S_z$ the size-k subsets of vertices of G, and X_i an indicator random variable for S_i being an independent set of G, so $X_i = 1$ only happens when all $\binom{k}{r}$ possible hyperedges among the vertices of S_i do not appear in the hyperedge set of G. Note that this happens with probability $(1-p)^{\binom{k}{r}} = (1-p)^{\frac{k\cdots(k-r+1)}{r!}} \leqslant (1-p)^{\frac{k(k-r+1)}{r!}}$. Thus,

$$\begin{aligned}
\Pr(\alpha(G) \geqslant k) &= \Pr\left(\sum_i E(X_i) \geqslant 1\right) \leqslant \sum_i E(X_i) = \binom{n}{k}(1-p)^{\binom{k}{r}} \\
&\leqslant \binom{n}{k}(1-p)^{\frac{k(k-r+1)}{r!}} \leqslant n^k\left((1-p)^{(k-r+1)/r!}\right)^k \\
&\leqslant (n\,\mathrm{e}^{-p(k-r+1)/r!})^k \\
&= n^{-\frac{1}{2}\left(\frac{3r!\ln\, n}{2p}+(r-1)\right)}
\end{aligned}$$

so that $\Pr(\alpha(G) < k) \geqslant 1 - n^{-\frac{1}{2}\left(\frac{3r!\ln\, n}{2p}+(r-1)\right)}$, as desired. □

This lemma helps us prove a result for three-uniform hypergraphs.

Theorem 13. *For three-uniform hypergraphs,*

$$\Pr\left(|U(|C|)| \leqslant O(|C|^2\, \ln|C| + |C|^2 x) + o(|C|^3)\right) \geqslant 1 - n^{-\frac{1}{2}\left(\frac{9\,\ln|C|}{p}+2\right)},$$

with x as defined in [19].

Proof. First, $\alpha(G) = |G| - k^*$ [34, theorem 3.15], so $\alpha(C) = |C| - k^*$ and $k^* = |C| - \alpha(C) \geqslant |C| - 9\ln|C|/p - 2$, where the inequality follows from lemma 12 and the inequality holds with probability at least $1 - n^{-\frac{1}{2}\left(\frac{9\,\ln|C|}{p}+2\right)}$. For three-uniform hypergraphs, $|U(k)| \leqslant \frac{1}{4}k^*(k^2 - (k^*)^2 + 2k^* x) + o(k^3)$ [19, theorem 18]. □

Algorithm 2. UMHS algorithm.

```
1: Input: hypergraph $G = (V, E)$ and number of iterations $N$
2: Output: approximation $S'$ to the planted hitting set in $G$
3: $S' \leftarrow \emptyset$
4: for $n \in \{1, \ldots, N\}$ do
5:     $S \leftarrow$ algorithm1($G$)
6:     Prune $S$ to be minimal hitting set of $G$
7:     $S' \leftarrow S' \cup S$
8: end for
9: return $S'$
```

Theorem 13 enables us to find a small set J that contains the core C with high probability.

Theorem 14. *Let C with $|C| = k$ be a planted hitting set in a three-uniform hypergraph drawn from the SBM parameterized by p and q with ck nodes for some $c \geqslant 1$. Then with high probability in k, we can find a set J with $|J| \leqslant O(k^2 \ln k + k^2 x) + o(k^3)$ that is guaranteed to contain C.*

Proof. The number of ways to link a vertex in the core to two nodes outside the core is $\binom{k(c-1)}{2}$ so that the probability of a node having at least one hyperedge to two nodes outside the core is $w = 1 - (1-q)^{\binom{k(c-1)}{2}}$. Thus w^k is the probability of each of the nodes in the core having at least one hyperedge to two nodes outside the core. This probability tends to 1 as $k \to \infty$, so by using lemma 10, $C \subseteq U(|C|)$ with probability tending to 1. Setting $J = U(|C|)$ and using theorem 13 gives the result. □

Of course, the above theorem is only useful if we have a bound on the quantity x since only in that case are we able to improve our bound on the size of J from $O(k^3)$ to $O(k^2 \log k)$. Absent any additional information, $x \leqslant k^*$ [19].

In the following section, we build a practical algorithm for recovery of planted hitting sets based on the UMHS, the theory for which has been examined so far.

3. Union of minimal hitting sets (UMHS): a practical core recovery algorithm

Based on the theoretical results described above, we now develop a practical algorithm for recovering a planted hitting set in a hypergraph. To put our theory into practice, we place several of our theoretical results in context. First, lemma 7 says that algorithm 1 produces outputs that *must* overlap with the planted hitting set. Moreover, this was true *regardless of the order in which algorithm 1 processed the hyperedges*. Thus, the basic idea of our algorithm is straightforward: we find many hitting sets S from algorithm 1 and take their union.

By corollary 8, if the planted hitting set C is close to a minimum hitting set, then any output from algorithm 1 will recover a large fraction of it. Furthermore, by corollary 9, the outputs of algorithm 1 must overlap by a modest amount, so taking the unions of outputs cannot grow too fast. To limit growth further, we can prune the output of algorithm 1 to be a minimal hitting set (this also tends to give better results in practice). Now, theorem 3 says that the union of pruned outputs will be bounded, provided the graph is large enough and that the outputs are small enough. This turns out to be the case in our experimental results, as we will see in the next section. To summarize, our procedure is as follows:

(a) Find a hitting set S from algorithm 1, processing the hyperedges in a random order;
(b) Prune S to be a minimal hitting set S'; and
(c) Repeat steps (a) and (b) several times and output the union of the S' found in step (b).

This method is formalized in algorithm 2. Again, the key rationale is that lemma 7 holds regardless of the ordering in which the hyperedges are processed. A similar algorithm for the special case of two-uniform hypergraphs (i.e., graphs) was recently analyzed [6].

The output of algorithm 2 is by construction a UMHS of the input hypergraph G, and we will refer to the algorithm as UMHS. The theoretical guarantees from section 2.1 apply here. Specifically, under the assumption that the planted hitting set C is itself minimal, we are guaranteed full recovery with sufficient iterations. However, planted hitting sets are rarely minimal in practice (in fact, they are not for the datasets we consider). However, theorem 3 still guarantees that the output of the algorithm will not grow very quickly, and our experiments verify that this is the case in practice.

Importantly, algorithm 2 runs in time linear in the number of hyperedges. In particular, suppose the input is a hypergraph $G = (V, E)$ of rank r with a minimum hitting set of size k^* and that we invoke algorithm 1 N times. Then the algorithm produces an output in $O(Nr^2k^*|E|)$ time. This happens because every call to algorithm 1 takes $O(r|E||)$ time, while pruning H to be a minimal hitting set of G requires $O(r^2k^*|E|)$ time in the

Table 1. Summary statistics of core-fringe hypergraph datasets. We construct r-uniform hypergraphs from six corpora for rank $r = 3, 4, 5$ along with non-uniform hypergraphs where the rank is at most 25. The DBLP and Math tags datasets are collections of 50 hypergraphs, so we report the value range. The fringe tends to be much larger than the core in these datasets.

	Number of nodes				Size of core ($\|C\|$)				Number of hyperedges			
	$r = 3$	$r = 4$	$r = 5$	$r \leqslant 25$	$r = 3$	$r = 4$	$r = 5$	$r \leqslant 25$	$r = 3$	$r = 4$	$r = 5$	$r \leqslant 25$
Enron	1283	976	869	4423	84	73	77	132	2361	1048	614	15 653
Avocado	5521	3510	2965	36 244	227	218	211	243	21 690	12 455	6973	96 966
W3C	1778	749	308	14 317	353	237	125	1509	1882	389	88	19 821
DBLP	3–875	4–514	5–261	2–2277	3–114	4–90	5–55	2–142	1–753	1–265	1–78	1–1547
Math tags	93–1180	97–1239	121–1153	432–1370	3–15	4–15	5–15	2–15	112–13 771	128–17 424	98–13 410	2098–42 585

worst case (in practice, it is much faster). Thus, in the worst case, pruning to a minimal hitting set takes more time than one iteration of the greedy algorithm. However, this increase in processing time is not prohibitive since it only increases by the constant factor rk^*. Nonetheless, pruning is crucial as in practice we find that it drastically reduces the output size.

4. Experimental results

We now test algorithm 2 on a number of real-world datasets and compare to several baselines derived from network centrality and core-periphery structure. We find that it consistently outperforms these baselines. Lastly, we show that we do not need too many iterations within algorithm 2 (the parameter N) for our performance levels.

4.1. Data

The five datasets we use broadly fall into three broad classes based on the types of planted hitting problems we derive from them. The first group is three email datasets (Enron, Avocado, and W3C), where the planted hitting set is a group of people at an organization, and hyperedges come from emails involving multiple addresses. Next, in the Digital Bibliography & Library Project (DBLP) dataset, a planted hitting set is a set of authors who publish at the same conference in a given year, and we consider several such conferences. Finally, in the Math Stack Exchange tagging dataset, the core is a one-hop neighborhood of a node, which is a hitting set for the two-hop neighborhood of the node; again, we consider several hitting sets. In this sense, the DBLP and tagging datasets are collections of hypergraphs with core-fringe structure. We now provide additional details, and table 1 lists summary statistics of the datasets.

Enron [39]**, Avocado**[1]**, and W3C** [16]**.** The hyperedges in these datasets are emails, where the nodes are the sender and all receivers (more specifically, their email addresses). Repeated hyperedges are discarded. In the Enron data, the core is a set of 150 employees whose email was made public when the company was under United States federal investigation. The Avocado corpus comes from a now-defunct technology company, where the core are employee email accounts (we exclude email accounts associated with conference rooms or mailing lists from the core). Finally, the W3C dataset comes from emails on mailing lists related to W3C; the core is all email addresses with a `w3c.org` domain.

DBLP. DBLP is an online computer science bibliography. We construct a hyperedges from coauthorship data in conference proceedings. We randomly sampled 50 conferences and constructed a set of core nodes C from the authors of a given conference in a randomly selected year. We then took the core-fringe hypergraph to be all hyperedges involving at least one of these authors. In total, we have 50 core-fringe hypergraphs.

Math tags [5]**.** This dataset comes from tags applied to questions on the Math Stack Exchange web site[2]. Hyperedges are sets of tags that have been applied to the same question. We sampled 50 tags with relatively small cores and then formed a core set of nodes C from the one-hop neighborhood of that tag, i.e., from the tag and all tags that appear as co-tags with the tag. The set C is then a planted hitting set for the two-hop neighborhood of that tag (i.e., all hyperedges containing at least one tag from the planted hitting set C). In total, we again have 50 core-fringe hypergraphs.

We constructed two types of filtered hypergraph datasets. First, we constructed three-, four-, and five-uniform hypergraphs (derived as sub-hypergraphs). By imposing uniformity, we can facilitate comparison of UMHS performance to that of other centrality-based algorithms which work on only uniform hypergraphs (in particular, H- and Z-eigenvector centrality). This also lets us study the performance of our algorithm as

[2] https://math.stackexchange.com/.

Table 2. Planted hitting set recovery performance for our algorithm and competitive baseline algorithms. We compare our proposed UMHS method against five hypergraph centrality measures—degree [38], clique-motif graph eigenvector [4], a projected-graph version of PageRank centrality [47], Z-eigenvector [4], H-eigenvector [4]—as well as two core-periphery measurements—Borgatti–Everett [12] and k-core [52]. Each method produces an ordering of vertices, and we measure performance by precision at the core size (fraction of top-$|C|$ ranked nodes that are in C) and AUPRC for r-uniform hypergraphs ($r = 3, 4, 5$), as well as nonuniform hypergraphs with $r \leqslant 25$ nodes in each hyperedge. The DBLP and tags datasets are collections of 50 hypergraphs, and we report the mean and standard deviation for these. UMHS scores outperforming all baselines by at least an 8% relative improvement are bold. Any method's score outperforming UMHS by at least 8% is also bold. Our UMHS method performs the best on all the non-uniform hypergraphs and many of the uniform cases.

	Dataset	r	UMHS	Degree	Clique-eigen	PageRank	Z-eigen	H-eigen	Borgatti–Everett	k-core
P@$\|C\|$	Enron	3	**0.52**	0.33	0.14	0.36	0.11	0.13	0.14	0.33
		4	**0.53**	0.33	0.21	0.36	0.18	0.16	0.19	0.33
		5	**0.49**	0.25	0.16	0.35	0.14	0.17	0.14	0.17
		$\leqslant 25$	**0.64**	0.35	0.17	0.37	—	—	0.16	0.33
	Av.	3	**0.91**	0.60	0.56	0.62	0.54	0.58	0.59	0.58
		4	**0.82**	0.57	0.52	0.60	0.33	0.53	0.52	0.56
		5	**0.72**	0.56	0.49	0.62	0.34	0.50	0.49	0.33
		$\leqslant 25$	**0.98**	0.62	0.54	0.64	—	—	0.14	0.17
	W3C	3	**0.51**	0.38	0.20	0.46	0.11	0.21	0.19	0.35
		4	0.43	0.43	0.30	**0.54**	0.30	0.31	0.30	0.40
		5	0.30	**0.42**	0.31	**0.59**	**0.34**	**0.33**	**0.33**	**0.41**
		$\leqslant 25$	**0.60**	0.28	0.13	0.31	—	—	0.12	0.24
	DBLP	3	0.59 ± 0.15	**0.63**$\pm$ 0.15	0.44 ± 0.24	**0.66** ± 0.14	0.42 ± 0.28	0.41 ± 0.26	0.41 ± 0.26	0.44 ± 0.21
		4	0.48 ± 0.15	**0.58** ± 0.17	0.40 ± 0.27	**0.63** ± 0.18	0.36 ± 0.29	0.39 ± 0.28	0.38 ± 0.31	0.40 ± 0.22
		5	0.39 ± 0.15	**0.65** ± 0.22	**0.56** ± 0.31	**0.63** ± 0.20	**0.59** ± 0.35	**0.57** ± 0.31	**0.56** ± 0.31	**0.56** ± 0.25
		$\leqslant 25$	**0.72** ± 0.13	0.60 ± 0.13	0.24 ± 0.18	0.64 ± 0.13	—	—	0.24 ± 0.21	0.24 ± 0.16
	Tags	3	**0.80** ± 0.14	0.58 ± 0.13	0.43 ± 0.15	0.53 ± 0.14	0.41 ± 0.14	0.45 ± 0.13	0.50 ± 0.14	0.43 ± 0.12
		4	**0.71** ± 0.18	0.48 ± 0.12	0.38 ± 0.12	0.51 ± 0.12	0.31 ± 0.14	0.39 ± 0.11	0.31 ± 0.10	0.38 ± 0.11
		5	**0.61** ± 0.19	0.40 ± 0.10	0.33 ± 0.10	0.42 ± 0.13	0.24 ± 0.11	0.32 ± 0.10	0.31 ± 0.10	0.33 ± 0.10
		$\leqslant 25$	**0.83** ± 0.10	0.47 ± 0.13	0.33 ± 0.12	0.47 ± 0.13	—	—	0.30 ± 0.11	0.34 ± 0.12
AUPRC	Enron	3	0.18	0.15	0.08	0.17	0.07	0.07	0.08	0.15
		4	0.16	0.16	0.10	0.18	0.09	0.09	0.10	0.16
		5	0.17	0.13	0.10	0.18	0.10	0.10	0.10	0.10
		$\leqslant 25$	**0.33**	0.14	0.05	0.16	—	—	0.05	0.13
	Av.	3	**0.43**	0.38	0.33	0.40	0.31	0.36	0.37	0.36
		4	**0.38**	0.35	0.30	0.38	0.15	0.31	0.30	0.34
		5	0.34	0.34	0.28	**0.39**	0.16	0.28	0.27	0.33
		$\leqslant 25$	**0.69**	0.39	0.29	0.41	—	—	0.29	0.35
	W3C	3	0.36	0.27	0.20	0.32	0.19	0.20	0.20	0.25
		4	0.40	0.37	0.31	0.44	0.31	0.31	0.31	0.35
		5	0.41	0.41	0.38	**0.52**	0.38	0.38	0.38	0.41
		$\leqslant 25$	**0.40**	0.15	0.11	0.17	—	—	0.11	0.14
	DBLP	3	0.57 ± 0.17	0.52 ± 0.18	0.39 ± 0.24	0.54 ± 0.16	0.40 ± 0.28	0.38 ± 0.25	0.37 ± 0.25	0.47 ± 0.23
		4	0.47 ± 0.19	0.46 ± 0.17	0.36 ± 0.26	**0.53** ± 0.19	0.35 ± 0.27	0.36 ± 0.26	0.36 ± 0.29	0.39 ± 0.22
		5	0.54 ± 0.23	0.57 ± 0.24	0.53 ± 0.31	0.53 ± 0.22	**0.58** ± 0.34	0.54 ± 0.31	0.54 ± 0.31	**0.58** ± 0.26
		$\leqslant 25$	**0.57** ± 0.18	0.41 ± 0.17	0.15 ± 0.15	0.45 ± 0.18	—	—	0.16 ± 0.18	0.22 ± 0.16
	Tags	3	**0.68** ± 0.18	0.36 ± 0.14	0.21 ± 0.13	0.31 ± 0.14	0.19 ± 0.12	0.23 ± 0.12	0.27 ± 0.14	0.26 ± 0.12
		4	**0.56** ± 0.20	0.25 ± 0.11	0.17 ± 0.11	0.28 ± 0.12	0.12 ± 0.11	0.17 ± 0.09	0.11 ± 0.07	0.17 ± 0.10
		5	**0.48** ± 0.21	0.18 ± 0.08	0.13 ± 0.07	0.20 ± 0.11	0.08 ± 0.07	0.12 ± 0.06	0.11 ± 0.06	0.14 ± 0.07
		$\leqslant 25$	**0.70** ± 0.15	0.24 ± 0.12	0.13 ± 0.09	0.24 ± 0.13	—	—	0.11 ± 0.07	0.16 ± 0.10

we change the rank of the hypergraph. In addition, since our algorithm naturally handles *nonuniform* hypergraphs, we also constructed nonuniform sub-hypergraphs of rank 25 (in the Stack Exchange data, posts have at most five tags, so we include the entire hypergraph for this data). Limiting the rank to 25 cuts out outliers. In particular, it removes large mailing lists in email data and specialized papers with many authors. This subsampling omits a relatively small fraction of hyperedges—11% for the Enron dataset, 1% for the Avocado dataset, and 0.2% for the W3C dataset.

4.2. Recovery results

We tested the UMHS algorithm on several datasets, and it outperforms other algorithms consistently. For baselines, we use two techniques. First, we use notions of network centrality developed for hypergraphs with the idea that nodes in the core could be identified via large centrality scores. Specifically, we compare against

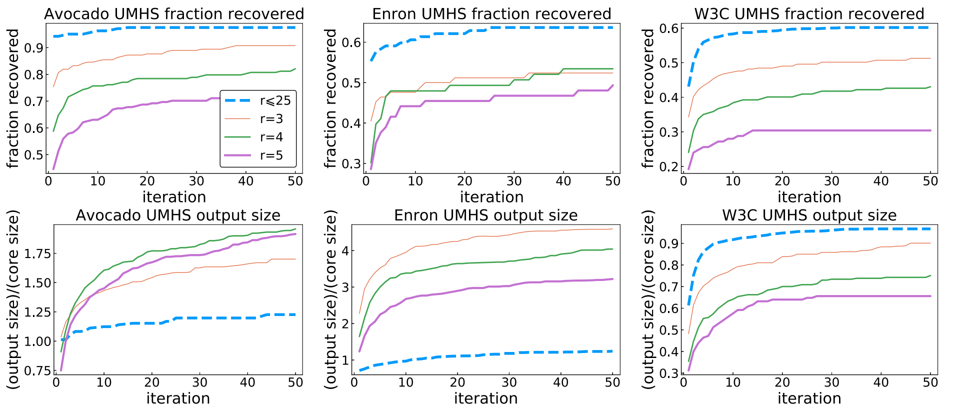

Figure 3. Planted hitting set recovery improves with more iterations before leveling (top row), and similarly, size increases with more iterations before leveling (bottom row). Fewer than 50 iterations typically reaches peak performance in our email datasets. The leveling of the output size is consistent with theorem 3 and corollary 8, which say that our algorithm's output cannot grow too large.

(a) hypergraph degree centrality (the number of hyperedges in which a node appears) [38];
(b) clique graph eigenvector (eigenvector centrality on the weighted clique graph, where w_{ij} is the number of hyperedges containing i and j) [4];
(c) PageRank on the weighted clique graph [47];
(d) Z-eigenvector hypergraph centrality [4]; and
(e) H-eigenvector hypergraph centrality [4].

Second, we use notions of network core-periphery decompositions, where we expect that nodes in C will be identified as 'core' in this sense. We use two algorithms:

(a) Borgatti–Everett network core-periphery scores [12] in the weighted clique graph; and
(b) the k-core decomposition [52] based on hypergraph degree.

All of these methods work equally well for uniform and non-uniform hypergraphs, except for Z-eigenvector and H-eigenvector eigenvector centrality, which are only designed for uniform hypergraphs. All of these methods also induce an ordering on the nodes. We induce an ordering on the output of algorithm 2 by degree and then sort the remaining nodes (those not in the output of algorithm 2) in order by degree. With an ordering on the nodes, we measure performance in terms of precision at core size, i.e., the fraction of the first $|C|$ in the ordering that are in C, as well as area under the precision-recall curve (AUPRC). We use AUPRC as opposed to area under the receiver operating characteristic (ROC) curve due to class imbalance [20]; namely, most nodes are not in the core.

Table 2 reports the results of all methods on all datasets. In the case of nonuniform hypergraphs, UMHS is superior to all other baselines in terms of both precision at core size and AUPRC by substantial margins. With regards to the uniform sub-hypergraphs, UMHS out-performs the baselines by wide margins on the Enron, Avocado, and Math tags datasets for all uniformities r in terms of precision at core size. UMHS also does well for the three-regular W3C hypergraphs. On DBLP and the other W3C hypergraphs, projected PageRank and the simple degree heuristic seem to perform well, although our algorithm still outperforms them on a large share of samples. There are similar trends in terms of AUPRC, with UMHS performing the best on the same set of datasets from above; and in this case, UMHS is more competitive even for the datasets where it is weakest—the four-uniform and five-uniform W3C hypergraphs, and even the DBLP hypergraphs.

Overall, the performance of many algorithms degrades as we increase the uniformity of the hypergraph. This makes sense in general, as the core nodes are increasingly hidden in larger hyperedges. In the case of UMHS, we have a more specific interpretation of what this means. As the hyperedges get larger, there may be cases of hyperedges that contain just one node in the core C, and the rest in the fringe. However, by the greedy structure of algorithm 1, we would put all nodes in the hitting set. Furthermore, the bounds in corollaries 8 and 9 also degrade as we increase the uniformity r. Nevertheless, our proposed UMHS algorithm still outperforms the baselines at an aggregate level.

4.3. Recovery as a function of output size

Our UMHS algorithm (algorithm 2) has a single tuning parameter, which is the number of iterations N, i.e., the number of calls to the sub-routine for greedy maximal matching (algorithm 1). Here we examine performance of UMHS as a function of N. Specifically, we analyze:

(a) the fraction of core nodes in the planted hitting set that are recovered and
(b) the output size of algorithm 2 as a function of the number of iterations

Figure 3 shows these statistics for the three email datasets.

We highlight a couple of important findings. First, we only need around 50 iterations to achieve high recovery rates. Each iteration is fast, and the entirety of the algorithm's running time takes at most a few minutes on the larger datasets. Second, the UMHS size tends to increase sharply with a few iterations and then levels off sharply. These results are consistent with our theory: theorem 3 and corollary 8 both provide theoretical justification for why the output should not grow too large.

5. Related work

On the theoretical side, our problem can be thought of as an instance of a 'planted' problem, where a certain type of graph structure is planted or hidden in a graph and one must recover the latent structure given the graph. Well-studied problems in this space include the planted clique, where one tries to find a clique placed in a sample from a $G_{n,1/2}$ graph [2, 22, 23, 26]; and the planted partition or SBM recovery, where a random graph is sampled with probabilities dependent on latent labels of nodes, and the goal is to recover these labels [1, 46, 48, 57]. These planted problems are based on some random way in which the graph was sampled. In our case of planted hitting sets, the graph was deterministic, although we could improve our results under a random hypergraph model. Most related to our results is recent work on planted vertex covers; this is a special case of hitting sets for the case of graphs (which mathematically are the same as two-uniform hypergraphs) [6]. As discussed above, the hypergraph model is more realistic for many datasets (especially email), given its ability to represent groups of more than two individuals at a time.

Within the field of network science, the idea of a small planted hitting set fits with two related ideas: node centrality and core-periphery structure. The former concept deals with finding important nodes (or ranking them) based on their connections, often modeled as a graph [9, 11, 30]. Nodes in hitting sets are central to hypergraphs almost by definition—every hyperedge must contain at least one of these nodes. Thus, we expect them to be 'central' in some sense. However, we found that existing measures of node centrality in hypergraphs did not recover planted hitting sets at the same levels as our UMHS algorithm.

Core-periphery structure is a mesoscale property of many networks, where there is a densely connected core set of nodes along with a loosely connected periphery [17]. Such a composition has been studied in sociology [24, 44] and international trade [54], where the core-periphery structure is due to differential status. Now, core-periphery identification is a broader tool for identifying structure in general networks [32, 37, 49, 51]. The planted hitting set that we aim to recover corresponds to an extreme type of core-periphery structure; due to the way in which we assume the hypergraph is measured, nodes on the periphery (the 'fringe nodes') cannot be connected without a core node as an intermediary in a hyperedge.

Finally, core-fringe structure itself has received some attention. For example, the behavior of a core group of employees at a hedge fund has been analyzed in the context of their relationships with contacts outside of the company [50], and the core-fringe structure has been shown to influence graph-based link prediction algorithms [3]. Our research highlights additional richness to the problem when the underlying data model is a hypergraph.

6. Discussion

Network data is a partial view of a larger system [43]. A common case is when the interactions of some specified set of actors or nodes are under surveillance. This provides a 'core-fringe' structure to the network—we can see all interactions involving the core but only the interactions of the fringe with the core. When data is leaked or metadata is lost over time due to data provenance issues, we would like to be able to recover these core and fringe labels for security or data maintenance purposes.

Here, we have studied this problem where the network data is a hypergraph, so the core is a hitting set. This setting is common in email data or situations in which groups of people are meeting. We used co-authorship as a proxy for the latter situation, but one can imagine situations in which one records the groups of attendees of meetings involving someone under surveillance. Theoretically, we showed that the UMHS cannot be too large and that the output of the well-known approximation algorithm for minimum hitting sets has to somehow overlap the core.

Using these results as motivation, we developed an extremely simple algorithm for recovering the core: take the UMHS that are output by randomly initialized instances of the approximation algorithm. This method out-performed several strong baselines, including methods based on centrality and core-periphery structure.

However, our simple algorithm opens several avenues for improvement in future work. For instance, our model assumed an undirected and unweighted hypergraph structure. There are models of directed and weighted hypergraphs [27], as well as other hypergraph generalizations [15], that could be used to improve the recovery algorithm in practice. In addition, theory on the number of calls to the approximation algorithm subroutine would be useful. In practice, only a few calls is sufficient and perhaps assuming particular structure on the hypergraph could yield additional theoretical insight.

Acknowledgments

This research was supported by NSF Award DMS-1830274, ARO Award W911NF19-1-0057, ARO MURI, JPMorgan Chase & Co., a Simons Investigator Award, a Vannevar Bush Faculty Fellowship, and a Grant from the AFOSR.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/ilyaamburg/Hypergraph-Planted-Hitting-Set-Recovery.

ORCID iDs

Ilya Amburg https://orcid.org/0000-0003-1632-5427

References

[1] Abbe E 2018 Community detection and stochastic block models: recent developments *J. Mach. Learn. Res.* **18** 6446–531
[2] Alon N, Krivelevich M and Sudakov B 1998 Finding a large hidden clique in a random graph *SODA*
[3] Benson A and Kleinberg J 2019 Link prediction in networks with core-fringe data *The World Wide Web Conf.* (ACM)
[4] Benson A R 2019 Three hypergraph eigenvector centralities *SIAM J. Math. Data Sci.* **1** 293
[5] Benson A R, Abebe R, Schaub M T, Ali J and Kleinberg J 2018 Simplicial closure and higher-order link prediction *Proc. Natl Acad. Sci.* **115** E11221
[6] Benson A R and Kleinberg J 2018 Found graph data and planted vertex covers *NeurIPS*
[7] Boldi P, Bruno C, Santini M and Vigna S 2004 UbiCrawler: a scalable fully distributed web crawler *Softw. - Pract. Exp.* **34** 711
[8] Boldi P, Marino A, Santini M and Vigna S 2014 BUbiNG: massive crawling for the masses *WWW*
[9] Boldi P and Vigna S 2014 Axioms for centrality *Internet Math.* **10** 222
[10] Bollobás B 1986 *Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability* (Cambridge: Cambridge University Press)
[11] Bonacich P 1987 Power and centrality: a family of measures *Am. J. Sociol.* **92** 1170
[12] Borgatti S P and Everett M G 2000 Models of core/periphery structures *Soc. Netw.* **21** 375
[13] Peter B, Khanna S and Wang-Chiew T 2001 Why and where: a characterization of data provenance *ICDT*
[14] Chitnis R, Cormode G, Esfandiari H, Hajiaghayi M T, McGregor A, Monemizadeh M and Vorotnikova S 2016 Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams *Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms* (SIAM) pp 1326–44
[15] Chodrow P and Mellor A 2020 Annotated hypergraphs: models and applications *Appl. Netw. Sci.* **5** 9
[16] Craswell N, de Vries A P and Soboroff I 2005 Overview of the trec 2005 enterprise track *TREC* vol 5 pp 199–205
[17] Csermely P, London A, Wu L-Y and Uzzi B 2013 Structure and dynamics of core/periphery networks *J. Complex Netw.* **1** 93
[18] Damaschke P 2006 Parameterized enumeration, transversals, and imperfect phylogeny reconstruction *Theor. Comput. Sci.* **351** 337
[19] Damaschke P and Molokov L 2009 The union of minimal hitting sets: parameterized combinatorial bounds and counting *J. Discrete Algorithms* **7** 391
[20] Davis J and Goadrich M 2006 The relationship between precision-recall and ROC curves *ICML*
[21] Decelle A, Krzakala F, Moore C and Zdeborová L 2011 Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications *Phys. Rev.* E **84** 066106
[22] Dekel Y, Gurel-Gurevich O and Peres Y 2011 Finding hidden cliques in linear time with high probability *ANALCO*
[23] Deshpande Y and Montanari A 2015 Finding hidden cliques of size N/e in nearly linear time *Found. Comput. Math.* **15** 1069
[24] Doreian P 1985 Structural equivalence in a psychology journal network *J. Am. Soc. Inf. Sci.* **36** 411
[25] Erdös P and Rado R 1960 Intersection theorems for systems of sets *J. Lond. Math. Soc.* **s1–35** 85
[26] Feige U and Krauthgamer R 2000 Finding and certifying a large hidden clique in a semirandom graph *Random Struct. Algorithms* **16** 195
[27] Gallo G, Longo G, Pallottino S and Nguyen S 1993 Directed hypergraphs and applications *Discrete Appl. Math.* **42** 177
[28] Ghoshdastidar D and Dukkipati A 2014 Consistency of spectral partitioning of uniform hypergraphs under planted partition model *NeurIPS*
[29] Gile K J and Handcock M S 2010 Respondent-driven sampling: an assessment of current methodology *Sociol. Method.* **40** 285

[30] Gleich D F 2015 PageRank beyond the web *SIAM Rev.* **57** 321
[31] Goel S and Salganik M J 2010 Assessing respondent-driven sampling *Proc. Natl Acad. Sci.* **107** 6743
[32] Govindan P, Wang C, Xu C, Duan H and Soundarajan S 2017 The *k*-peak decomposition *WWW*
[33] Guimerà R and Sales-Pardo M 2009 Missing and spurious interactions and the reconstruction of complex networks *Proc. Natl Acad. Sci.* **106** 22073
[34] Halldórsson M M and Losievskaja E 2009 Independent sets in bounded-degree hypergraphs *Discrete Appl. Math.* **157** 1773
[35] Heckathorn D D 2011 Comment: snowball versus respondent-driven sampling *Sociol. Methodol.* **41** 355
[36] Hier S P and Greenberg J 2009 *Surveillance: Power, Problems, and Politics* (Vancouver: UBC Press)
[37] Holme P 2005 Core-periphery organization of complex networks *Phys. Rev.* E **72** 046111
[38] Kapoor K, Sharma D and Srivastava J 2013 Weighted node degree centrality for hypergraphs *IEEE Network Science Workshop*
[39] Klimt B and Yang Y 2004 The Enron corpus: a new dataset for email classification research *Machine Learning: ECML 2004* (Berlin: Springer)
[40] Kossinets G 2006 Effects of missing data in social networks *Soc. Netw.* **28** 247
[41] Kostochka A, Mubayi D and Verstraëte J 2014 On independent sets in hypergraphs *Random Struct. Algorithms* **44** 224
[42] Kuny T 1997 A digital dark ages? challenges in the preservation of electronic information of electronic information *IFLA Council and General Conf.*
[43] Laumann E O, Marsden P V and Prensky D 1989 The boundary specification problem in network analysis *Research Methods in Social Network Analysis* (Lanham, MD)
[44] Lorrain F and White H C 1971 Structural equivalence of individuals in social networks *J. Math. Sociol.* **1** 49
[45] Lynch C 2008 How do your data grow? *Nature* **455** 28
[46] Mossel E, Neeman J and Sly A 2014 Belief propagation, robust reconstruction and optimal recovery of block models *COLT*
[47] Page L, Brin S, Motwani R and Winograd T 1999 *The Pagerank Citation Ranking: Bringing Order to the Web. Technical Report* Stanford InfoLab
[48] Peixoto T P 2017 Nonparametric Bayesian inference of the microcanonical stochastic block model *Phys. Rev.* E **95** 0123171
[49] Rombach M P, Porter M A, Fowler J H and Mucha P J 2017 Core-periphery structure in networks (revisited) *SIAM Rev.* **59** 619
[50] Romero D M, Uzzi B and Kleinberg J 2016 Social networks under stress *WWW*
[51] Sarkar S, Bhowmick S and Mukherjee A 2018 On rich clubs of path-based centralities in networks *CIKM*
[52] Seidman S B 1983 Network structure and minimum degree *Soc. Netw.* **5** 269
[53] Simmhan Y L, Plale B and Gannon D 2005 A survey of data provenance in e-science *ACM SIGMOD Rec.* **34** 31
[54] Smith D A and White D R 1992 Structure and dynamics of the global economy: network analysis of international trade 1965–1980 *Soc. Forces* **70** 857
[55] Tan W-C 2004 Research problems in data provenance *IEEE Data Eng. Bull.* **27** 45–52
[56] Alexander T, Saniee I, Narayan O and Andrews M 2013 Spectral analysis of communication networks using Dirichlet eigenvalues *WWW*
[57] Tsourakakis C 2015 Streaming graph partitioning in the planted partition model *COSN*