On Computing Maximal Independent Sets of Hypergraphs in Parallel

Ioana O. Bercea
Department of Computer
Science
University of Maryland
College Park, MD, USA
ioana@cs.umd.edu

Navin Goyal
Microsoft Research India
Bangalore, India
navingo@microsoft.com

David G. Harris
Department of Applied
Mathematics
University of Maryland
College Park, MD, USA
davidgharris29@hotmail.com

Aravind Srinivasan
Department of Computer
Science and UMIACS
University of Maryland
College Park, MD, USA
srin@cs.umd.edu

ABSTRACT

Whether or not the problem of finding maximal independent sets (MIS) in hypergraphs is in (R)NC is one of the fundamental problems in the theory of parallel computing. Unlike the well-understood case of MIS in graphs, for the hypergraph problem, our knowledge is quite limited despite considerable work. It is known that the problem is in RNC when the edges of the hypergraph have constant size. For general hypergraphs with n vertices and m edges, the fastest previously known algorithm works in time $O(\sqrt{n})$ with $\text{poly}(m, n)$ processors. In this paper we give an EREW PRAM algorithm that works in time $n^{o(1)}$ with $\text{poly}(m, n)$ processors on general hypergraphs satisfying $m \leq n^{\frac{\log^{(2)} n}{8(\log^{(3)} n)^2}}$, where $\log^{(2)} n = \log \log n$ and $\log^{(3)} n = \log \log \log n$. Our algorithm is based on a sampling idea that reduces the dimension of the hypergraph and employs the algorithm for constant dimension hypergraphs as a subroutine.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Hypergraphs—*Independent Sets*; F.1.2 [**Computation by Abstract Devices**]: Parallelism and concurrency; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Computations on discrete structures; 6.1.0 [**Numerical Analysis**]: Parallel algorithms

Keywords

hypergraphs; independent sets;randomized algorithms; parallel algorithms

1. INTRODUCTION

Fast parallel algorithms for constructing maximal independent sets (MIS) in graphs are well studied and very efficient algorithms are now known (see, e.g., [3] for a brief survey). These algorithms serve as a primitive in numerous applications. The more general problem of fast parallel MIS in hypergraphs is also well studied but is not so well understood. Let us first formally state the problem before describing what is known about it.

A *hypergraph* $H = (V, E)$ is a set of vertices V and a collection of edges $e \in E$ such that $e \subseteq V$. The *dimension* of a hypergraph is the maximum edge size. Let n be the number of vertices, m the number of edges and d the dimension of the hypergraph. A subset of vertices of H is called *independent* in H if it contains no edge. We call an independent set *maximal* if it is not contained in a larger independent set. Karp and Ramachandran [3] asked whether the problem of finding an MIS in hypergraphs is in NC. While the general problem remains open, progress has been made on some special classes of hypergraphs. We now briefly survey some of the previous work; further references can be found in the papers mentioned below.

In a seminal paper, Beame and Luby [2] gave an algorithm (called the BL **algorithm** henceforth) and showed that the problem is in RNC for hypergraphs with edges of size at most 3 ([2] claimed that their algorithm was in RNC for all constant dimension hypergraphs; this however turned out to be erroneous). This algorithm is similar to some of the MIS algorithms for graphs and is based on independently marking vertices and unmarking if all vertices in an edge get marked. Kelsen [5] extended the analysis of the BL algorithm to hypergraphs with constant dimension (the dimension can actually be super-constant; we state the precise bound later in the paper where we use this fact). Luczak and Szymanska [7] showed that the problem is in RNC for linear hypergraphs (linear hypergraphs satisfy $|e \cap e'| \leq 1$ for all distinct edges e, e'). Beame and Luby [2] also gave another appealing algorithm based on random permutations which they conjectured to work in RNC for the general problem. Shachnai and Srinivasan [9] made progress towards the

analysis of this algorithm. For general hypergraphs, Karp, Upfal and Wigderson [4] gave an algorithm with running time $O(\sqrt{n})$ and $\text{poly}(m, n)$ processors (their algorithm actually works in a harder model of computation where the hypergraph is accessible only via an oracle, but it can be adapted to run in time $O(\sqrt{n}) \cdot (\log n + \log m)$ with high probability on mn processors).

Our contribution.

We give a parallel algorithm that we call the SBL (*sampling* BL) algorithm. The algorithm works on hypergraphs that do not have too many edges but no other restrictions and works in time $O(n^{o(1)})$. This is the first parallel algorithm that works on general hypergraphs with a relatively weak restriction on the cardinality of the edge set and a running time of $o(\sqrt{n})$.

More precisely,

THEOREM 1. *The* SBL *algorithm finds a maximal independent set in hypergraphs with n vertices and m edges and* $m \le n^{\frac{\log^{(2)} n}{8(\log^{(3)} n)^2}}$. *It runs in time $O(n^{2/\log^{(3)} n})$ on EREW PRAM with $\text{poly}(m, n)$ processors.*

The parameters above have been chosen to keep the computation in the analysis simple and there is some flexibility in their choice.

Our algorithm crucially uses BL as a subroutine. However, we need to use it on hypergraphs with slightly superconstant dimensions. Kelsen's original analysis [5] of BL is formulated for constant dimension hypergraphs. A slight modification of this analysis, specifically in the potential function used to describe progress being made in each round, allows it to be applicable without the assumption that the dimension is $O(1)$. We present this modification. We also discuss two additional improvements that can be made to Kelsen's analysis of the BL algorithm, which could be of independent interest: (1) Kelsen developed concentration inequalities for polynomials in independent random variables. Since then much stronger versions of such inequalities have become available [6, 8]. We employ one such inequality and obtain an improved upper bound which we later use in the analysis. (2) We also show that a significantly larger marking probability can be employed, without affecting the runtime negatively. Unfortunately, both of the above modifications do not lead to a significant improvement in the final running time of the algorithm. Nevertheless, we hope that this identifies the main bottlenecks in Kelsen's approach and will be useful for the future work.

Organization.

The next section is devoted to the SBL algorithm. Section 3 delves into Kelsen's analysis of the BL algorithm. We note that there is a large overlap with Kelsen's paper in Sec. 3 owing to the fact that we are mainly talking about modifications to his analysis and this requires us to restate many of his results and proofs to make the paper somewhat self-contained.

2. SBL ALGORITHM

We now explain the SBL algorithm which mainly uses the BL algorithm as a subroutine. Denote the input hypergraph by $H = (V, E)$. Intuitively, we can think of the BL algorithm as iteratively coloring the vertices in V red or blue; at the end of the run of the algorithm the blue vertices will form the final MIS. The idea of our algorithm is to randomly sample a subset V' of vertices by independently marking each vertex in V with probability p (to be carefully chosen). With high probability, the hypergraph $H' = (V', E')$, where $E' = \{e \in E : e \subseteq V'\}$ is the set of edges with all vertices marked, has dimension at most d, where d is suitably small (if H' has an edge with size more than d then we declare failure and start over). We then apply the BL algorithm to H' to get a red-blue coloring of its vertices, where blue vertices form an MIS in H'. This coloring will be the permanent coloring of the vertices of H'. Going back to H, we remove the edges of H that have a red vertex as these edges cannot be all blue in any completion of the coloring of H'. For the remaining edges, we remove their blue vertices and thus get a hypergraph on $V \setminus V'$. We repeat the above process on this updated hypergraph until the number of edges becomes at most $1/p^2$. At this point we can just use the algorithm that takes time linear in the number of vertices or alternatively the Karp–Upfal–Wigderson algorithm [4] which we shall call KUW. The vertices in the MIS returned by this last call will again be colored blue, while the rest will be colored red.

Algorithm 1 SBL

Input: A hypergraph $H = (V, E)$
Output: A maximal independent set $I \subseteq V$

1: Let $p = 1/n^\alpha$ and $d = \frac{\log^{(2)} n}{4 \log^{(3)} n}$ where $n = |V|$ and $\alpha = 1/\log^{(3)} n$.
2: $I \leftarrow \emptyset$
3: **if** $\max_{e \in E} |e| > d$ **then**
4:     **while** $|V| \ge 1/p^2$ **do**
5:       *Invariant:* If I' is an IS in H', then $I \cup I'$ will be an IS in H.
6:       Select vertices independently at random with probability p
7:       Let V' be the collection of such selected vertices, $E' = \{e \in E : e \subseteq V'\}$ and $H' = (V', E')$.
8:       **if** $\max_{e' \in E'} |e'| > d$ **then**
9:         FAIL
10:       **else**
11:         Run $I' =$BL(H')
12:         Update $I = I \cup I'$, $V = V \setminus V'$
13:         **for all** $e \in E$ **do**
14:           **if** $e \cap (V' \setminus I') \ne \emptyset$ **then**
15:             $E \leftarrow E \setminus e$.
16:         **for all** $e \in E$ **do**
17:           $e \leftarrow e \setminus I'$.
18:     Run $I' =$ KUW(H).
19:     Update $I = I \cup I'$.
20: **else**
21:     Run $I =$BL(H).
22: Return I.

2.1 Correctness of SBL

We claim that in the final coloring produced by the SBL algorithm, the set of blue vertices forms an MIS in the original hypergraph H. Let H_i be the hypergraph being colored in round i, where by round we mean one iteration of the **while** loop or the last call we make either to BL or KUW. We use the fact that the set I' returned either by BL or KUW is

indeed an MIS in H_i, so every violation of independence or maximality in H_i leads to a contradiction.

If the final set of blue vertices is not independent then there is some round i of SBL in which some edge e became fully blue. This means that a nonempty subset e' of e must be an edge in the hypergraph H_i, since $e \setminus e'$ is fully blue and e' is not yet colored. But now round i cannot color e' fully blue because it finds an MIS in H_i—a contradiction.

If the final set of blue vertices is not maximal, then it means that some red vertex can be recolored blue without violating independence. Let v be such a vertex and suppose that it was colored red in round i. Then in H_i, there exists a hyperedge e' such that recoloring v blue will make it fully blue which in turn would lead to some edge e in H being fully blue—a contradiction.

2.2 Analysis of SBL

We use the BL algorithm as a subroutine and use the following theorem about its performance:

THEOREM 2. *On hypergraph with n vertices, m edges, and dimension $d \leq \frac{\log^{(2)} n}{4 \log^{(3)} n}$, BL algorithm terminates after $O(\log n)^{(d+4)!}$ time with probability at least $1 - 1/n^{\Theta(\log n \log^{(2)} n)}$. It uses $\mathrm{poly}(m, n)$ processors and can be implemented on EREW PRAM.*

The above result is essentially the same as the corresponding statement in [5] when $d = O(1)$. As mentioned before, the proof follows from a slight modification of the potential function of [5]; it appears in Section 3.1.

In the analysis of the running time below we will focus on the number of rounds of SBL algorithm. The time for each round of SBL is dominated by the time for running BL in that round. Specifically, notice that the only other call we make is to KUW(H) when the number of vertices in H is less than $1/p^2 = n^{2/\log^{(3)} n}$. In the worst case, the runtime of the algorithm is linear in the number of vertices, so we get an additional factor of $O(n^{2/\log^{(3)} n})$ in our overall runtime.

We begin by setting values of the parameters used in the algorithm:

- $p := 1/n^\alpha$,
- $m := n^\beta$,
- $\alpha := 1/\log^{(3)} n$,
- $\beta := \frac{\log^{(2)} n}{8(\log^{(3)} n)^2}$.

We will use the following form of the Chernoff bound (see, e.g., [1]).

LEMMA 1. *Let X be a random variable taking value 1 with probability $p \in [0, 1]$ and value 0 with probability $1 - p$. Then the sum $X_1 + \ldots + X_n$ of i.i.d. copies of X for $a > 0$ satisfies*

$$\Pr[(X_1 + \ldots + X_n) \leq pn - a] \leq e^{-a^2/2pn}.$$

There are three kinds of events (A, B, and C) that can happen during the execution of the SBL algorithm resulting in failure or high running time. We will show that the union of these events has small probability by upper bounding each event separately and then applying the union bound. We use $\Pr[B] \leq \Pr[A] + \Pr[B|\neg A]$ and similarly for $\Pr[C]$, resulting in the bound

$$\Pr[A \vee B \vee C] \leq 3 \Pr[A] + \Pr[B|\neg A] + \Pr[C|\neg A]. \quad (1)$$

1. With high probability in each round, the fraction of vertices colored is substantial and thus the number of rounds is small.

2. The probability that a large hyperedge is ever fully marked in a round is small and thus all our applications of BL algorithm are valid.

3. The probability that a run of BL algorithm fails in some round is small.

We now prove these three claims.

(1) Denote the number of marked vertices in round i of SBL by n_i; thus $n_1 = n$, and in round i, the BL algorithm is invoked on a hypergraph with $n_i - n_{i+1}$ vertices (the set of marked vertices). Then, for each i, by Lemma 1 we have:

$$\Pr[(n_i - n_{i+1}) \leq pn_i/2] \leq e^{-pn_i/8} \leq e^{-1/(8p)}.$$

The inequality above holds because SBL algorithm maintains that $n_i \geq 1/p^2$ and so $pn_i \geq 1/p$ for all i. If the above event holds in each round, then the smallest r satisfying $(1 - p/2)^r \leq \frac{1}{p^2 n}$ is an upper bound on the number of rounds. Setting $r := \frac{2 \log n}{p}$ gives an upper bound on the number of rounds. Then the probability of the event not holding in some round becomes $\frac{2 \log n}{p} \cdot e^{-1/8p} \leq \frac{1}{n^{\log n}}$, for sufficiently large n.

(2) Conditioning on the number of rounds being upper bounded by r, the probability that an edge of size at least $d + 1$ is fully marked in some round is at most $rmp^{(d+1)}$. If we want this probability to be upper bounded by $1/n$ then we can take

$$d := \frac{\log(rmn)}{\log(1/p)} - 1.$$

Substituting the values of r, p, m as chosen above we get:

$$d = \frac{\log 2 + \log^{(2)} n}{\log 1/p} + \frac{\log m}{\log 1/p} + \frac{\log n}{\log 1/p}$$

$$= \frac{(\log 2 + \log^{(2)} n) \log^{(3)} n}{\log n} + \frac{\beta (\log^{(3)} n)(\log n)}{\log n} + \frac{\log^{(3)} n \log n}{\log n}$$

$$\leq 2\beta \cdot \log^{(3)} n$$

$$= \frac{\log^{(2)} n}{4 \log^{(3)} n},$$

where the inequality holds for all sufficiently large n.

(3) Theorem 2 gives that the probability of failure of BL algorithm in round i is at most $\frac{1}{n_i^{\Theta(\log n_i \log^{(2)} n_i)}}$. Using $n_i \geq 1/p^2$, this probability is at most $\frac{1}{n^{\log n}}$ for sufficiently large n. Hence, conditioning on the number of rounds being upper bounded by r, the probability of any one round failing is upper bounded by $r \cdot \frac{1}{n^{\log n}} \leq \frac{1}{n^{(\log n)/2}}$.

Thus the total probability of failure using (1) is at most $\frac{3}{n^{\log n}} + \frac{1}{n} + \frac{1}{n^{(\log n)/2}} \leq 2/n$, for sufficiently large n.

Now we account for the time taken by the algorithm. The first round takes time at most $(\log pn)^{d^d}$ (with high prob.), and the subsequent rounds have the same upper bound. Thus the total time is bounded by $r(\log pn)^{d^d}$ (here we are upper bounding $(d+4)!$ somewhat crudely by d^d which holds for all sufficiently large n). For our choice of d above we can upper bound this by

$$r(\log n)^{d^d} \le r(\log n)^{(\log n)^{1/4}} = \frac{2\log n}{p}(\log n)^{(\log n)^{1/4}} =$$

$$2n^{1/\log^{(3)} n}(\log n)^{(\log n)^{1/4}+1} \le n^{2/\log^{(3)} n},$$

for sufficiently large n.

This completes the analysis.

3. ANALYSIS OF BL

In this section, we present a streamlined analysis of BL and show that it can accommodate for a larger d while maintaining the running time of $O(\log n)^{(d+4)!}$.

Before we describe the improvements in the analysis, we give a brief overview of the algorithm. In the first step, each vertex is marked independently at random with some probability p. After the marking step, for any edge that is fully marked, we unmark all its vertices. We add the remaining marked vertices to the independent set and perform the a cleanup operation in which we update the vertex and edge set (by trimming them), remove singleton edges and discard all edges that now contain smaller edges as subsets. We then recurse on this new hypergraph. For a pseudocode of the algorithm we refer the reader to Appendix A.

Like usual, the general strategy in upper bounding the number of rounds necessary for the algorithm to finish, is to define an appropriate quantity and show that progress is being made in each round. Intuitively, we can pick one of several such quantities (the number of vertices, the maximum degree of a vertex, the number of edges etc.) and show that it is reduced by a constant fraction every couple of rounds. The trouble comes from the fact that, in the case of hypergraphs and of the BL algorithm in particular, none of these quantities are easy to track. For example, the probability that a vertex gets discarded in one round depends on whether it was marked and never participated in a fully marked edge. When it comes to the degree of a vertex, more evolved measures are needed than in the classical graph case, since now, several vertices can participate together in multiple edges. In this context, we define some essential notation. Let $H = (V, E)$ be a hypergraph with dimension d. For $\emptyset \ne x \subseteq V$ and an integer j with $1 \le j \le d - |x|$, we define the number of edges of size $|x| + j$ that include x as a subset:

$$N_j(x, H) = \{y \subseteq V : x \cup y \in E \wedge x \cap y = \emptyset \wedge |y| = j\}.$$

We also define the normalized degree of x with respect to dimension $|x| + j$ edges

$$d_j(x, H) = (|N_j(x, H)|)^{1/j}.$$

The maximum normalized degree with respect to dimension i edges then becomes

$$\Delta_i(H) = \max\{d_{i-|x|}(x, H) : x \subseteq V \wedge 0 < |x| < i\}.$$

Finally, the maximum normalized degree is defined as

$$\Delta(H) = \max\{\Delta_i(H) : 2 \le i \le d\}.$$

At this point, notice that, as noted in [5], the main bottleneck in the analysis is the migration of higher dimensional edges to lower dimensional ones. Specifically, in each round, we need to account for the decrease in $N_j(x, H)$ due to edges of size $|x| + j$ decreasing in size, but also for the potential increase due to edges of size $|x| + k$, $k > j$ becoming edges of size $|x| + j$. In order to upper bound such an increase, [5] develops a bound on the upper tail of sums of dependent random variables defined on the edges of a hypergraph. We mention the general bound here and defer the description of its application to later in the paper.

In order to state the result, we need to describe the probabilistic setting: we consider a hypergraph $H = (V(H), E(H))$ with $n(H)$ vertices, $m(H)$ edges and dimension $dim(H)$. We also consider a weight function w on its edges $w(e) > 0$ for any edge e. The random variables C_v will correspond to each vertex being colored independently at random with probability p for the color blue and $1 - p$ for the color red. Alternatively, the random variable will take the value 1 with probability p and value 0 with probability $1 - p$. The random variable whose upper tail we will bound will be expressed as the polynomial $S(H, w, p)$. The terms of this polynomial will correspond to an edge e being fully colored blue $C_e = \prod_{v \in e} C_v$. The weights $w(e)$ will become the corresponding coefficients. The polynomial $S(H, w, p)$ then represents the sum of all the weighted edges being colored blue:

$$S(H, w, p) = \sum_{e \in E(H)} w(e) \cdot C_e.$$

Unlike general concentration bounds, we will not compare $S(H, w, p)$ just against its expectation. We will, instead, consider the expected values of all partial derivatives of the polynomial $S(H, w, p)$ with respect to a subset of vertices x. Specifically, for a given $x \subseteq V(H)$, we will consider quantities of the form

$$P(H, w, p, x) = \sum_{e \in E(H), x \subseteq e} w(e) \cdot p^{|e|-|x|}.$$

Essentially, this term represents the expected sum of the weighted edges around x that are colored all blue, given that x is already colored blue. Notice that this is the same setting used by more recent and considerably better concentration inequalities (e.g. [6],[8]) to describe their results and in that sense, Kelsen's bound is surprisingly advanced. We then define:

$$D(H, w, p) = \max\{P(H, w, p, x) : x \subseteq V(H)\}.$$

Notice that $D(H, w, p)$ is greater than the expectation of $S(H, w, p)$. The final result follows:

THEOREM 3. *(Theorem 1 in [5]) Let (H, w) be a weighted hypergraph with $dim(H) = d > 0$ and $n(H) = n \ge 3$. For $0 < p \le 1$ and $\delta > 1$, we have*

$$\Pr[S(H, w, p) > k(H) \cdot D(H, w, p)] < p(H)$$

where

$$k(H) = (\log n + 2)^{2^d - 1} \cdot \delta^{2^{d-1}} \text{ and}$$
$$p(H) = (2^d \cdot \lceil \log n \rceil \cdot m(H))^{d-1} \cdot \log n \cdot (\tfrac{4e}{\delta-1})^{(\delta-1)/4}.$$

We are now ready to describe the complete analysis. In order to prove Theorem 2, we present a succinct version of the analysis that emphasizes the main ingredients of the proof and our contribution. For full details on the original analysis, we refer the reader to the papers of Beame and Luby [2] and Kelsen [5]. In the following subsection, we will revisit some of the tools used and show that the analysis goes through even when we consider a higher sampling probability.

3.1 Theorem 2

The main purpose of Theorem 2 is to show that the analysis follows even when we allow $d \leq \frac{\log^{(2)} n}{4 \log^{(3)} n}$. We start by setting the initial sampling probability to $p = 1/(a\Delta)$ where $a = 2^{d+1}$. The first crucial step is lower bounding the probability that a particular set of vertices X is added to the independent set. We begin by defining random variables C_v for when a vertex v is initially marked (i.e. $C_v = 1$ when the v is marked and 0 otherwise) and E_v for when a vertex is unmarked later due its participation in fully marked edges (i.e. $E_v = 1$ when v is unmarked, 0 otherwise). We also define the random variable $A_v = C_v \wedge \neg E_v$ to stand for when the vertex v gets added to the independent set. This notion can be extended to subsets of vertices, by defining $C_X = \bigwedge_{v \in X} C_v$ and $E_X = \bigvee_{v \in X} E_v$, and $A_X = C_X \wedge \neg E_X$. Notice that

$$\Pr[A_X] = \Pr[C_X] \cdot (1 - \Pr[E_X | C_X]).$$

Lemma 1 from [2] shows that $\Pr[A_X] < 1/2$ by proving that $\Pr[E_X | C_X] < 1/2$.

LEMMA 2. *(Lemma 1 in [2]) Given a hypergraph $H = (V, E)$ of dimension d, and a set of vertices $X \subseteq V$ with $|X| < d$ such that no $e \subset X$ is an edge, we have $\Pr[E_X | C_X] < 1/2$. (I.e. given that X is marked, it will be added to the IS with probability $> 1/2$.)*

We will use the preceding lemma to ensure that progress is being made at each stage of the algorithm. Specifically, we will focus our attention on those sets X that have a large degree with respect to edges of size $|X| + j$. To this extent, **Lemma** 2 in [2]) shows that if such a large degree set exists, then one of the edges that contains it is likely to decrease and turn X into an edge by itself. Once that event occurs, the degree of X becomes 0.

LEMMA 3. *(Lemma 2 in [2]) For any set of vertices X and j such that $|X| + j \leq d$, if $d_j(X, H) \geq \epsilon\Delta$, then*

$$Pr[\exists Y \in N_j(X, H) \ s.t. \ A_Y] \geq \frac{1}{4}(\epsilon/a)^j,$$

where $a = 2^{d+1}$.

We now discuss the last ingredient of the proof: the upper bound on the migration of edges from higher dimensions to lower dimensions. Notice that the previous lemma is not enough to show that the degree of X will become 0 in a polylogarithmic number of stages. This is because over each stage, $d_j(X, H)$ can actually increase through the migration of edges from $N_k(X, H)$ where $k > j$. In this context, we employ Theorem 3. The hypergraph H' we construct consists of all the vertices in H and has as edges all subsets of size $k - j$ of the elements in $N_k(X, H)$, i.e all the potential ways in which an edge of size $|X| + k$ can lose $k - j$ vertices and become an edge of size $|X| + j$ around X. Formally, let $X_{j,k}$ be the edge set:

$$X_{j,k} = \{Y : Y \subseteq V(H') \wedge |Y| = k - j \wedge \exists Z \in N_k(X, H'), Y \subseteq Z\}.$$

The random variables C_v correspond to the situation in which a vertex v gets marked, with probability p. The weight w' of each edge $Y \in X_{j,k}$ represents the number of edges of size $|X| + j$ around X that would be formed if Y were to be fully added to the MIS. Formally:

$$w'(Y) = |N_j(X \cup Y, H')|.$$

The polynomial $S(H', w', p)$ then becomes an upper bound on the potential increase in $N_j(X, H)$ due to edges in $N_k(X, H)$. Notice that, in our case, H' has dimension at most $d - 1 < \frac{\log^{(2)} n}{4(\log^{(3)} n)}$, but by choosing $\delta = \log^2 n$, we can arrive at a cleaner formulation of **Theorem 3** from [5]:

COROLLARY 1. *(Corollary 1 in [5]) Fix a $d > 0$ and a real number p, $0 < p \leq 1$. For any weighted hypergraph (H', w) of dimension at most d with at most n vertices,*

$$\Pr[S(H', w', p) > (\log n)^{2^{d+1}} \cdot D(H', w', p)] < \frac{1}{n^{\Theta(\log n \cdot \log \log n)}}.$$

When it comes to $D(H', w', p)$, we can bound it by something more meaningful in our context:

LEMMA 4. *(Lemma 3 in [5]) Let H', w' and p be defined as above. Then:*

$$D(H', w', p) \leq (\Delta_{|X|+k}(H))^j.$$

Notice that the same bound applies when we consider the increase in the normalized degree $d_j(X, H')$ and since $\Delta_{|X|+k}(H) \leq \Delta$, we obtain the following **Corollary** 3 from [5]:

COROLLARY 2. *(Corollary 3 in [5]) With high probability, for $2 \leq j \leq d$, the maximum increase in $d_{j-|X|}(X, H)$ for any non-empty $X \subseteq V$ during a single stage of the algorithm is less than*

$$\sum_{k>j} (\log n)^{2^{k-j+1}} \cdot \Delta_k(H).$$

Notice that this bound is meaningful in comparison with the trivial bound we would obtain by considering the worst case scenario of *all* higher dimensional edges migrating down:

$$(\textstyle\sum_{k>j} \Delta_k(H)^{k-|X|})^{1/(j-|X|)} \geq \sum_{k>j} \Delta_k(H),$$

since $\Delta_k(H)$ could be as high as n.

At this point in the analysis, we can describe the behaviour of each individual $d_j(X, H)$ by a lower bound on the probability that it diminishes when it is too large (Lemma 3) and an upper bound on how much it can increase in each stage (Corollary 2). We would like to be able to somehow compare these quantities with a universal threshold that we can show will eventually decrease. The trouble comes from expressing the latter of the quantities in terms of this universal threshold: if we compare each $\Delta_k(H)$ to the threshold in the same way (suppose by saying that it is smaller than $1/2$ of the threshold value), we obtain a trivial upper bound on the increase in $\Delta_j(H)$. A solution to this problem would be to define an individual threshold for each $\Delta_k(H)$ separately and relate all of these back to a universal threshold. In this context, [5] defines the values $v_i(H)$ inductively by $v_d(H) = \Delta_d(H)$ and:

$$v_i(H) = \max\{\Delta_i(H), (\log n)^{f(i)} \cdot v_{i+1}(H)\},$$

for $2 \leq i < d$, where f is a carefully chosen function (to be defined later) that accommodates for the increase in $\Delta_j(H)$ due to migration from higher edges. Essentially, $v_i(H)$ tries to take into account the most significant term in the increased $\Delta_i(H)$: it is either the $\Delta_i(H)$ from the previous

round or the most significant term from larger edges offset by a scaling factor $(\log n)^{f(i)} \cdot v_{i+1}(H)$. These individual thresholds relate to the universal threshold by considering the quantities $T_j = v_2(H)/(\log n)^{F(j-1)}$, where $F(i) = \sum_{j=2}^{i} f(j)$ for $2 \leq i \leq d$. Notice that for any hypergraph H', $v_i(H') \leq v_2(H')/(\log n)^{F(j-1)}$. The rest of the analysis focuses on showing that the universal threshold $v_2(H)$ is reduced by a constant fraction every several rounds.

Let $H_s = (V(H_s), E(H_s))$ be the hypergraph used in stage s of the algorithm and let $v_i = v_i(H_{s_0})$ be the values of these potential functions at the start of a fixed stage of the algorithm. Similarly, let T_j be defined with respect to v_j. The main technical lemma is the following:

LEMMA 5. *(Lemma 4 in [5]) Let r be an arbitrary positive constant. Then, with high probability, at any stage s with $s_0 \leq s \leq s_0 + (\log n)^r$, we have*

$$v_2(H_s) \leq v_2 \cdot (1 + o(1)).$$

In fact, [5] proves that something stronger holds with high probability:

$$v_j(H_s) \leq T_j \cdot (1 + \lambda(n)),$$

where $\lambda(n) = 2 \cdot \log^{(2)} n / \log n$.

The main argument is by induction on $d - j$ and we will not reproduce it entirely. We will, instead, give the general intuition and focus on the parts of the argument that could change if we allow d to be non-constant. Notice that $v_j(H_s) = \max\{\Delta_j(H_s), (\log n)^{f(j)} \cdot v_{j+1}(H_s)\}$. By induction,

$$(\log n)^{f(j)} \cdot v_{j+1}(H_s) \leq (\log n)^{f(j)} \cdot T_{j+1} \cdot (1 + \lambda(n))$$
$$\leq T_j \cdot (1 + \lambda(n)).$$

So we only need to focus on showing that

$$\Delta_j(H_s) \leq T_j \cdot (1 + \lambda(n)),$$

with high probability. The tactic is to show that, if $\Delta_j(H_s)$ ever becomes greater than $\frac{1}{2} \cdot T_j \cdot (1 + \lambda(n))$, then in q_j consecutive stages it will decrease with high probability, taking into account the potential migration of edges during those stages. Specifically, suppose there exists an $x \subseteq V(H_s)$ such that

$$d_{j-|x|}(x, H_s) \geq \frac{1}{2} \cdot T_j \cdot (1 + \lambda(n)).$$

One can show that this implies that

$$d_{j-|x|}(x, H_s) \geq \frac{\Delta(H_s)}{2(\log n)^{F(j-1)}}.$$

At this point, we can apply Lemma 3 with $\epsilon = \frac{1}{2(\log n)^{F(j-1)}}$ and get that

$$\Pr[d_{j-|x|}(x, H_{s+1}) > 0] \leq$$
$$1 - \frac{1}{2^{d(d+1)} \cdot (\log n)^{F(j-1)(j-1)}}.$$

In other words, the probability that in the next round we still have a high normalized degree is small. Notice that if we repeat the argument for

$$q_j = 2^{d(d+1)} \cdot (\log \log n) \cdot (\log n)^{F(j-1)(j-1)+2}$$

stages, we have that this remains true with probability at most $1/n^{\Theta(\log n \log \log n)}$. This is the first place in which we differ from the conventional analysis in [5] since we cannot ignore the $2^{d(d+1)}$ factor because it is not constant any more.

The only step left missing is to guarantee that the increase in $d_{j-|x|}(x, H_s)$ during those q_j stages is not large. We apply Corollary 2 and get that the total increase is $q_j \cdot \sum_{k>j} (\log n)^{2^{k-j+1}} \cdot \Delta_k(H_s)$. We want to show that such an increase is smaller than $\lambda(n) \cdot T_j$ and since by the inductive assumption we have that $\Delta_k(H_s) \leq T_k \cdot (1 + \lambda(n))$, we are left to show that

$$q_j \cdot \sum_{k>j} (\log n)^{2^{k-j+1}} \cdot T_k \cdot (1 + \lambda(n)) \leq \lambda(n) \cdot T_j.$$

After some calculation, plugging in the values of q_j and $\lambda(n)$, this can be shown to reduce itself to:

$$2^{d(d+1)} \cdot \sum_{k>j} (\log n)^{2^{k-j+1}+F(j-1)\cdot j - F(k-1)+2} \leq$$
$$\frac{2}{\log n + 2 \log \log n}.$$

It is at this point that the definition of f comes into play. [5] define $f(2) = 7$ and $f(i) = (i-1) \cdot \sum_{j=2}^{i-1} f(i) + 7$ for $i > 2$. We then get that $F(i) = i \cdot F(i-1) + 7$ for $i \geq 2$ and $F(1) = 0$. Notice that this definition of F does not allow us to make the above argument. Consider the case when $k = j + 1$. Then

$$2^{k-j+1} + F(j-1) \cdot j - F(k-1) + 2 = -1$$

and the claim becomes

$$2^{d(d+1)} \leq \frac{\log n}{\log n + 2 \log \log n}.$$

This is not true for the larger value of d we are considering. Notice that this was not an issue in the original analysis, because $2^{d(d+1)}$ was a constant in the case they were considering.

In order for the claim to be true, a different definition of f is required. Specifically, we define the recurrence relationship to be

$$f(i) = (i-1) \cdot \sum_{j=2}^{i-1} f(i) + d^2.$$

In this context, we obtain that $F(i) = i \cdot F(i-1) + d^2$. The claim then becomes:

$$2^{d(d+1)} \cdot \sum_{k>j} (\log n)^{2^{k-j+1}+2-d^2+F(j)-F(k-1)} \leq$$
$$\frac{2}{\log n + 2 \log \log n}.$$

We will now show that the claim is true for this new definition of f.

We will begin by first noticing that, for any j, the highest term in the sum is achieved for $k = j + 1$. Formally:

LEMMA 6. *For any $k > j + 1$ and any $j \geq 2$, we have*

$$2^{k-j+1} + 2 - d^2 + F(j) - F(k-1) \leq 6 - d^2.$$

The proof is done by showing that the terms are decreasing as a function of k and therefore, the maximum is achieved for the lowest possible value of k: $j + 1$.

As a consequence, the entire left hand side of the inequality can be upper bounded by

$$2^{d(d+1)} \cdot (d-j) \cdot \frac{1}{(\log n)^{d^2-6}}.$$

By taking $2^{d(d+1)} \leq e^{d(d+1)}$ and $d - j < \log \log n$, it would be enough to show

$$e^{d(d+1)} \cdot \frac{1}{(\log n)^{d^2-6}} \leq \frac{1}{\log^2 n}.$$

In other words, we can show that

$$d(d+1) \leq (\log \log n) \cdot (d^2 - 8).$$

One can check that this inequality holds for $d < \frac{\log^{(2)} n}{4 \log^{(3)} n}$.

At this point, we have shown that the total increase in $d_{j-|x|}(x, H_s)$ during those stages is upper bounded with high probability by $\lambda(n) \cdot T_j$. Moreover, notice that after q_j stages, $\Delta_j(H)$ will not exceed $T_j \cdot \frac{1+3\lambda(n)}{2}$ and hence, after q_d stages, we have that, with high probability,

$$v_j(H_{s_1}) \leq T_j \cdot \frac{1+3\lambda(n)}{2}$$

for any $2 \leq j \leq d$ and $s_0 \leq s_1 \leq s_0 + (\log n)^{q_j}$. In fact, going back to the start of the algorithm, $v_2(H)$ is reduced by a constant factor, with high probability, every q_d stages. Hence, after $O(\log n \cdot q_d)$ stages, we have that $v_2(H) = 0$ and therefore, $V(H) = 0$ and the algorithm terminates.

Now we are left to prove that

$$\log n \cdot q_d \leq (\log n)^{(d+4)!}.$$

Notice that

$$\begin{aligned}
q_d &\leq (log^{(2)} n)^2 \cdot (\log n)^{F(d-1)(d-1)+2} \\
&\leq (\log n)^{F(d-1)(d-1)+3} \\
&\leq (\log n)^{(d+4)!-1}
\end{aligned}$$

where the last inequality can be verified by inductively proving that $F(i) \leq d^2 \cdot (i+2)!$ for all i.

3.2 Larger sampling probability

In the attempt to provide insight into what happens with BL when we consider larger dimensions, we can show that the original lemmas from [2] hold even when the sampling probability is considerably larger: $p = 1/(5d \cdot \Delta)$ versus $1/(2^{d+1} \cdot \Delta)$.

Consider the extension of Lemma 1 from [2]:

LEMMA 7. *Given a hypergraph $H = (V, E)$ with dimension d, let $p = 1/(a\Delta)$ where $a = 5d$. Then for any set of vertices $X \subseteq V$ with $|x| < d$ such that no $e \subset X$ is an edge, we have that $Pr[E_X | C_X] < 1/2$.*

PROOF. Let $f(x) = Pr[E_X | C_X]$. Notice that $f(X) = Pr[\exists e \in E, e \cap X \neq \emptyset \text{ s.t. } C_{e-X}]$. Consider a vertex $t \notin X$. Then we have

$$\begin{aligned}
f(X \cup \{t\}) &= Pr[\exists e \in E, e \cap (X \cup \{t\}) \neq \emptyset \text{ s.t. } C_{e-(X \cup \{t\})}] \\
&= Pr[(\exists e \in E, e \cap X \neq \emptyset \text{ s.t. } C_{e-X}) \vee \\
&\quad (\exists e \in E, e \cap X = \emptyset, t \in e \text{ s.t. } C_{e-\{t\}})] \\
&\leq Pr[(\exists e \in E, e \cap X \neq \emptyset \text{ s.t. } C_{e-X})] + \\
&\quad Pr[(\exists e \in E, e \cap X = \emptyset, t \in e \text{ s.t. } C_{e-\{t\}})] \\
&\leq f(X) + Pr[(\exists e \in E, e \cap X = \emptyset, t \in e \text{ s.t. } C_{e-\{t\}})] \\
&\leq f(X) + Pr[(\exists e \in E, t \in e \text{ s.t. } C_{e-\{t\}})] \\
&\leq f(X) + f(\{t\}).
\end{aligned}$$

Notice that we can upper bound $f(\{t\})$ using the union bound. Specifically,

$$\begin{aligned}
f(\{t\}) &\leq \sum_{k=1}^{d-1} |N_k(\{t\}, H)| \cdot p^k \leq \sum_{k=1}^{d-1} d_k(\{t\}, H)^k \cdot p^k \\
&\leq \sum_{k=1}^{d-1} \Delta^k(H) \cdot p^k \leq \sum_{k=1}^{d-1} \frac{1}{a^k} < \frac{1}{a-1}
\end{aligned}$$

Overall, we get that

$$f(X) \leq \sum_{t \in X} f(\{t\}) < \frac{|X|}{a-1} < \frac{d}{a-1} \leq \frac{1}{2}.$$

$\square$

This carries over when we consider Lemma 2 in [2]:

LEMMA 8. *Let $a = 5d$. For any set of vertices X and j such that $|X| + j \leq c$, if $d_j(X, H) \geq \epsilon \Delta$, then*

$$Pr[\exists Y \in N_j(X, H) s.t. A_Y] \geq \tfrac{1}{4}(\epsilon/a)^j.$$

PROOF. Notice that we need a lower bound on $Pr[\exists Y \in N_j(X, H)$ s.t. $A_Y]$ and hence, we cannot use the union bound as usual. Instead, we will lower bound this event by another one. Specifically, we will require that there is a *unique* $Y \in N_j(X, H)$ such that A_Y. In this context, we can decompose the event into disjoint elements:

$$Pr[\exists! Y \in N_j(X, H), A_Y] = \sum_{Y \in N_j(X,H)} Pr[A_Y \wedge \neg(\exists Z \in N_j(X, H) \setminus \{Y\}, A_Z)].$$

We also have

$$\begin{aligned}
Pr[A_Y \wedge \neg(\exists Z \in N_j(X, H) \setminus \{Y\}, A_Z)] = \\
Pr[A_Y] \cdot (1 - Pr[(\exists Z \in N_j(X, H) \setminus \{Y\}, A_Z) | A_Y]).
\end{aligned}$$

Now we can apply the union bound and get

$$\begin{aligned}
Pr[\exists Z \in N_j(X, H) \setminus \{Y\}, A_Z | A_Y] \leq \\
\sum_{Z \in N_j(X,H) \setminus \{Y\}} Pr[A_Z | A_Y].
\end{aligned}$$

Putting everything together, we get

$$\begin{aligned}
Pr[\exists Y \in N_j(X, H), A_Y] \geq \\
\sum_{Y \in N_j(X,H)} Pr[A_Y] \cdot (1 - \sum_{Z \in N_j(X,H) \setminus \{Y\}} Pr[A_Z | A_Y]).
\end{aligned}$$

In other words, we are left with showing an upper bound on $\sum_{Z \in N_j(X,H), Z \neq Y} Pr[A_Z | A_Y]$.

Notice the following argument from [2]:

$$\begin{aligned}
Pr[A_Z | A_Y] &= \frac{Pr[A_Y \wedge A_Z]}{Pr[A_Y]} \\
&\leq \frac{Pr[C_Y \wedge C_Z]}{Pr[A_Y]} \\
&\leq 2 \cdot \frac{Pr[C_Y \wedge C_Z]}{Pr[C_Y]} \\
&\leq 2 \cdot Pr[C_Z | C_Y] = 2 \cdot Pr[C_{Z-Y}],
\end{aligned}$$

where the second to last inequality uses the previous lemma in the sense that:

$$\begin{aligned}
\Pr[A_Y] &= \Pr[C_Y \wedge \neg E_Y] \\
&= \Pr[C_Y] \cdot \Pr[\neg E_Y | C_Y] \\
&= \Pr[C_Y] \cdot (1 - \Pr[E_Y | C_Y]) \\
&\geq \frac{1}{2} \Pr[C_Y].
\end{aligned}$$

We therefore get that

$$\begin{aligned}
\sum_{Z \in N_j(X,H), Z \neq Y} \Pr[A_Z | A_Y] &\leq 2 \cdot \sum_{Z \in N_j(X,H), Z \neq Y} \Pr[C_{Z-Y}] \\
&\leq 2 \cdot \sum_{S \subset Y} \sum_{Z \in N_j(X,H), Y \cap Z = S} \Pr[C_{Z-S}] \\
&\leq 2 \cdot \sum_{S \subset Y} (1/a)^{j - |S|} \\
&\leq \frac{1}{2}.
\end{aligned}$$

The last inequality is true because

$$\begin{aligned}
\sum_{S \subset Y} (1/a)^{j - |S|} &= \sum_{l=0}^{j-1} \binom{j}{l} (1/a)^{j-l} \\
&= (1 + 1/a)^j - 1 \\
&\leq (1 + 1/a)^d - 1 \\
&\leq ((1 + 1/a)^a)^{d/a} - 1 \\
&\leq e^{d/a} - 1.
\end{aligned}$$

For $d/a \leq 1/5$ we get that $e^{d/a} - 1 \leq 1/4$ and the previous inequality follows. At this point, we get

$$\sum_{Z \in N_j(X,H), Z \neq Y} \Pr[A_Z | A_Y] \leq 1/2.$$

Overall, we have

$$\begin{aligned}
\Pr[\exists Y \in N_j(X,H), A_Y] &\geq \sum_{Y \in N_j(X,H)} \Pr[A_Y] \cdot (1 - \frac{1}{2}) \\
&\geq \frac{1}{2} \cdot \sum_{Y \in N_j(X,H)} \Pr[C_Y \wedge \neg E_Y] \\
&\geq \frac{1}{2} \cdot \sum_{Y \in N_j(X,H)} \Pr[C_Y] \cdot (1 - \Pr[E_Y | C_Y]) \\
&\geq \frac{1}{2} \cdot \sum_{Y \in N_j(X,H)} \Pr[C_Y] \cdot (1 - \frac{1}{2}) \\
&\geq \frac{1}{4} \sum_{Y \in N_j(X,H)} \Pr[C_Y] \\
&\geq \frac{1}{4} |N_j(X,H)| p^j \\
&\geq \frac{1}{4} d_j(X,H)^j (1/a\Delta)^j \\
&\geq \frac{1}{4} (\epsilon/a)^j.
\end{aligned}$$

$\square$

The next step of the proof that we are going to improve is the bound that Kelsen gives on the maximum potential increase in edges in one round, using the same setting as in the original analysis. We obtain an analogue of **Corollary** 2 in [5]:

COROLLARY 3. *For $X \subseteq V$, and $1 \leq j < k \leq d - |X|$, we have*

$$\Pr[S(X,j,k) > (1 + a_{k-j}\lambda^{k-j}) \cdot (\Delta_{|X|+k}(H))^j] \leq \\ 2e^2 e^{-\lambda} n^{k-j-1},$$

where $a_{k-j} = 8^{k-j}(k-j)!^{1/2}$.

Notice that we upper bounded the term $D(H', w', p)$ by $(\Delta_{|X|+k}(H))^j$, just like [5]. Simple algebra can show that this result follows even for the new value of a. Choosing $\lambda = \Theta(\log^2 n)$, we get that an analogue of Corollary 3 in [5]:

COROLLARY 4. *With high probability, for $2 \leq j \leq d$, the maximum increase in $d_{j-|X|}(X,H)$ for any non-empty $X \subseteq V$ during a single stage of the algorithm is less than:*

$$\sum_{k>j} (\log n)^{2(k-j)} \cdot \Delta_k(H).$$

Notice that the bound of $(\log n)^{2(k-j)}$ is much smaller than the one of $(\log n)^{2^{k-j+1}}$ in [5].

3.3 Discussion

In context of these improvements, the natural next step is to investigate what effect they have on the overall running time of BL. We show that, under the current set up of the potential function, no improvement is possible. Specifically, we show that the function F must be roughly exponential for the argument to follow, despite the obvious improvements.

Notice that in our previous attempt to call BL on a hypergraph with super-constant dimension, the main issue was showing that the increase in q_j rounds was upper bounded by $T_j \cdot \lambda(n)$. Incorporating all of the new improvements, we get that the claim formally looks like:

$$(5d)^d \cdot \sum_{k>j} (\log n)^{2(k-j)+F(j-1)\cdot j - F(k-1)+2} \leq \\ \frac{2}{\log n + 2 \log \log n}.$$

We proved this claim by showing that the largest term in the sum was upper bounded by $\frac{1}{(\log n)^{d^2 - 6}}$. We check the minimal conditions that f must satisfy in order for the new claim to be true by precisely looking at this largest term in the case when $k = j + 1$ for a fixed $2 \leq j \leq d$. Notice that, first of all, this will be the largest term when we allow F to satisfy $F(k-1) > F(j) + 2(k - j - 2)$ for all $k > j$. Given that, the term will be:

$$(\log n)^{4 + F(j-1) \cdot j - F(j)}.$$

Notice that, in order for the claim to be true, this term needs to be smaller than

$$\frac{1}{(5d)^d} \cdot \frac{2}{\log n + 2 \log \log n}.$$

In order for this to happen, we must have that

$$(\log n)^{4 + F(j-1) \cdot j - F(j)} \leq \frac{1}{\log n}.$$

This, in turn, requires that

$$F(j) \geq F(j-1) \cdot j + 5.$$

4. CONCLUSION

In this paper, we build on the RNC algorithm for computing an MIS in constant dimension hypergraphs to get an $n^{o(1)}$ algorithm on general hypergraphs when the number of edges is upper bounded by $n^{\frac{\log^{(2)} n}{8(\log^{(3)} n)^2}}$. In order to perform the analysis, we prove that the subroutine algorithm can be adapted to run on a larger dimension while maintaining an appropriate running time. We also present independent improvements to the analysis of the latter and identify the main bottleneck in the approach that affects the final run-time most significantly. For example, notice that the factor of j in the above expression $(F(j) \geq F(j-1) \cdot j + 5)$ originated from Lemma 3. Specifically, [2] lower bound the probability that $d_j(X) > \epsilon\Delta$ becomes 0 in the next iteration, as a function of $(\epsilon/a)^j$. A refinement of that result could potentially lead to a weaker restriction on F and hence, a smaller running time.

5. REFERENCES

[1] Noga Alon and Joel Spencer. *The Probabilistic Method.* John Wiley, 1992.

[2] P. Beame and M. Luby. Parallel search for maximal independence given minimal dependence. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 212–218. Society for Industrial and Applied Mathematics, 1990.

[3] Richard M. Karp and Vijaya Ramachandran. Parallel algorithms for shared-memory machines. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 869–942. 1990.

[4] R.M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *J. Comput. Syst. Sci.*, 36(2):225–253, 1988.

[5] Pierre Kelsen. On the Parallel Complexity of Computing a Maximal Independent Set in a Hypergraph. *Fourth annual ACM symposium on Theory of computing*, 3:339–350, 1992.

[6] Jeong Han Kim and Van H. Vu. Concentration of multivariate polynomials and its applications. *Combinatorica*, 20(3):417–434, 2000.

[7] Tomasz Luczak and Edyta Szymanska. A parallel randomized algorithm for finding a maximal independent set in a linear hypergraph. *J. Algorithms*, 25(2):311–320, 1997.

[8] Warren Schudy and Maxim Sviridenko. Concentration and moment inequalities for polynomials of independent random variables. In *SODA*, pages 437–446, 2012.

[9] Hadas Shachnai and Aravind Srinivasan. Finding large independent sets in graphs and hypergraphs. *SIAM J. Discrete Math.*, 18(3):488–500, 2004.

APPENDIX

A. BL ALGORITHM

We give the pseudocode of the BL algorithm as initially described in [2].

Algorithm 2 BL

Input: A hypergraph $H = (V, E)$
Output: A maximal independent set $I \subseteq V$.

1: Calculate $\Delta(H)$ as defined in Section 3.
2: Let $d = \max\{|e| : e \in E\}$ and $p = 1/(2^{d+1}\Delta)$.
3: $H' = (V', E') \leftarrow H = (V, E)$.
4: $I \leftarrow \emptyset$.
5: **while** $V' \neq \emptyset$ **do**
6:     Select vertices independently at random with probability p.
7:     Let I' be the collection of such selected vertices.
8:     **for all** $e \in E'$ such that $e \subseteq I'$ **do**
9:         $I' \leftarrow I' \setminus e$.
10:     $I \leftarrow I \cup I'$.
11:     $V' \leftarrow V' \setminus I'$.
12:     **for all** $e \in E'$ **do**
13:         $e \leftarrow e \setminus I'$.
14:     **for all** $e, e' \in E'$ **do**
15:         **if** $e \subseteq e'$ **then**
16:             $E' \leftarrow E' \setminus e$.
17:     **for all** $e = \{v\} \in E'$ **do**
18:         $E' \leftarrow E' \setminus e$.
19:         $V' \leftarrow V' \setminus \{v\}$.
20: Return I.