Finding Large Independent Sets of Hypergraphs in Parallel

Hadas Shachnai
Department of Computer Science
The Technion IIT
Haifa 32000, Israel
hadas@cs.technion.ac.il

Aravind Srinivasan
Bell Labs, Lucent Technologies
600-700 Mountain Avenue
Murray Hill, NJ 07974, USA
srin@research.bell-labs.com

ABSTRACT

A basic problem in hypergraphs is that of finding a *large* independent set–one of guaranteed size–in a given hypergraph. Understanding the parallel complexity of this and related independent set problems on hypergraphs is a fundamental open issue in parallel computation. Caro and Tuza (*J. Graph Theory*, Vol. 15, pp. 99–107, 1991) have shown a certain lower bound $\alpha_k(H)$ on the size of a maximum independent set in a given k-uniform hypergraph H, and have also presented an efficient sequential algorithm to find an independent set of size $\alpha_k(H)$. They also show that $\alpha_k(H)$ is the size of the maximum independent set for various hypergraph families. Here, we develop the first RNC algorithm to find an independent set of size $\alpha_k(H)$, and also derandomize it for various special cases. We also present lower bounds on independent set size and corresponding RNC algorithms for *non-uniform* hypergraphs.

Keywords

Hypergraphs, Independent Sets, Parallel Algorithms, Randomized Algorithms

1. INTRODUCTION

Finding large/maximal independent sets in (hyper)graphs, defined formally below, is a fundamental problem in parallel combinatorial optimization. An outstanding open question in parallel computation is whether a maximal independent set in a given hypergraph can be found in $(R)NC$ [14]. The work of Karp and Wigderson [16] on finding maximal independent sets in graphs in NC, was a breakthrough that inspired several graph-theoretic NC algorithms, and also led to a rich theory of derandomization. The corresponding problems on hypergraphs have applications, e.g., to feasible communication in channelized cellular telephone systems [20], but seem much harder than in the case of graphs. In this work we develop RNC algorithms for finding large independent sets in hypergraphs, and derandomize these for various special cases. The main RNC algorithm is particularly simple, and basically involves the parallel computation of maxima.

Recall that a *hypergraph* $H = (V, E)$ consists of a vertex set V and a collection E of subsets of V; each element of E is called a (hyper)edge. We will only consider finite hypergraphs here, and will throughout denote the number of vertices and edges in a given hypergraph by n and m respectively. An independent set (IS) in H is a subset S of V that does not contain any edge. S is a *maximal independent set* (MIS) if no proper superset of S is an IS. It is easy to find an MIS sequentially, but efficient parallel algorithms appear much harder. There has been much work on finding MISs in parallel in (hyper)graphs (see, e.g., [1, 4, 7, 9, 17, 15, 16, 18, 21]). Since an MIS can be much smaller than a maximum independent set (as in the case of a star graph), it is also of much interest to find ISs that have a guaranteed size. What is known in this context? Recall that given a hypergraph $H = (V, E)$, the *degree* of a vertex is the number of edges that it lies in; also, H is called k-uniform if all the edges have exactly k elements. Caro and Tuza showed in [6] that for any $k \geq 2$, any k-uniform hypergraph H contains an IS of size at least

$$\alpha_k(H) = \sum_{v \in V} \frac{1}{\binom{d(v) + 1/(k-1)}{d(v)}}; \tag{1}$$

here and from now on, $d(v)$ denotes the degree of $v \in V$. (For any integer $l \geq 0$ and real r, $\binom{r}{l}$ is defined to be $r(r-1)\cdots(r-l+1)/l!$.) They showed that this bound is tight for a large class of hypergraphs and also gave a *sequential* algorithm that finds an IS of size $\alpha_k(H)$ in $O(km+n)$ steps. Note that when $k = 2$, the bound in (1) reduces to

$$\alpha_2(G) = \sum_{v \in V} \frac{1}{d(v)+1}, \tag{2}$$

which is the classical Turán bound for graphs [26]. To give the reader a better feel for (1), we point out that

$$\binom{d(v) + 1/(k-1)}{d(v)} = \Theta((d(v))^{1/(k-1)}). \tag{3}$$

This is relatively easy to derive using the fact that

$$\binom{d(v) + 1/(k-1)}{d(v)} = \prod_{i=1}^{d(v)} (1 + 1/(i(k-1))),$$

SPAA '01 Crete, Greece

and then applying the inequality

$$e^{x-x^2/2} \leq 1 + x \leq e^x,$$

which holds for all $x \geq 0$. (The symbol e denotes the base of the natural logarithm.)

Remark. To handle the case where $d(v) = 0$, the r.h.s. of (3) should actually be $\Theta((d(v)+1)^{1/(k-1)})$. However, since vertices of degree zero are irrelevant and can always be added to our independent set, we will always assume that vertex-degrees are non-zero. Further, if an edge is just a singleton $\{v\}$, then vertex v cannot lie in any independent set. We therefore ignore all vertices that lie in singleton edges; hence, we will take all edge-sizes to be at least 2.

We would like to develop a parallel algorithm that finds an IS of size $\alpha_k(H)$ for k-uniform hypergraphs with any $k \geq 2$. We also aim to find large independent sets in non-uniform hypergraphs.

1.1 Related Work

The following parallel algorithms are known in the case of graphs. Spencer [23] gave an *RNC* algorithm that yields an IS of expected size $\alpha_2(G)$ in graphs G. Goldberg and Spencer [10] presented an *NC* algorithm that finds an IS of size at least $\lceil n^2/(2m+n) \rceil$ in any graph G, where n and m denote the number of vertices and edges in G respectively. This bound equals $\alpha_2(G)$ when G is regular; in all other cases, $\alpha_2(G)$ is higher. (In fact, it is not hard to construct graph families for which $\alpha_2(G) = \Theta(n)$ and $n^2/(2m+n)$ is just $\Theta(1)$: see item (i) in Section 1.2.) Recently, the second author developed an *NC* algorithm that finds an IS of size $(1 - o(1)) \cdot \alpha_2(G)$, where the "$o(1)$" term goes to zero as $n \to \infty$ [24]. Alon, Babai and Itai [1] studied the MIS problem on hypergraphs: they gave an NC algorithm that finds an IS of size $c(n^k/m)^{1/(k-1)}$, $0 < c < 1$, in k-uniform hypergraphs with $k \geq 2$ being any constant. The work of Karger and Koller [13] generalized this to arbitrary k.

1.2 Main Results

We give an RNC algorithm to find an IS of size $\alpha_k(H)$ in k-uniform hypergraphs, and also present related results and extensions. Our main contributions are as follows.

(i) The expected size of the IS produced by our algorithm is larger than the size of the IS found by previous parallel algorithms for this problem. For instance, $\alpha_k(H)$ is always at least as large as the bound $c(n^k/m)^{1/(k-1)}$ of [1, 13]. Furthermore, one can construct families of k-uniform hypergraphs for which $\alpha_k(H)$ is $\Theta(n)$ while $(n^k/m)^{1/(k-1)}$ is $\Theta(1)$. For instance, in the case of constant k, consider hypergraphs H where:

- the vertex set is partitioned into two equal-sized sets V_1 and V_2, and
- H contains all k-sized subsets of V_1 and $n/(2k)$ many pairwise-disjoint k-sized subsets of V_2.

It is easy to check that for such families of hypergraphs, $\alpha_k(H) = \Theta(n)$ and $(n^k/m)^{1/(k-1)} = \Theta(1)$.

(ii) We show how our methods extend to non-uniform hypergraphs: we are not aware of any such bounds or algorithms relating to large independent sets in this non-uniform case.

(iii) Regarding derandomized versions, we have the following for arbitrary hypergraphs. Consider the following three properties (with lg denoting the logarithm to the base 2, as usual):

(P1): all vertex-degrees are constant;

(P2): all the vertex-degrees are at most $O(\lg n)$;

(P3): for each vertex v, there are at most $O(\lg n)$ vertices w such that v and w lie in a common edge.

Then, we show that our algorithm can be made to run in *NC* if (P1) holds, or if both of (P2) and (P3) hold. (Given (P2), property (P3) holds, e.g., when every edge-size is bounded by a constant.)

(iv) All our results extend without any change in the processor or time complexity to their weighted analogs. In the weighted analogs, we are given a non-negative weight for each vertex, and wish to find an IS of large total weight. The only way that we are aware of to extend arbitrary IS algorithms to their weighted analogs is by a suitable (polynomial) blowup in the size of the hypergraph, leading to a loss in efficiency.

One key facilitator of our results is a way of generating random permutations that provides sufficient (stochastic) independence to conduct our analysis; please see Lemma 1. It also leads to algorithms and lower bounds for independent size in non-uniform hypergraphs, as demonstrated by Theorem 3.

2. AN RNC ALGORITHM

2.1 Algorithms and Tools

Spencer's algorithm for graphs [23] is as follows. Randomly permute the vertices; add a vertex $v \in V$ to the IS iff no neighbor of v precedes v in the random order. We show below via an application of the FKG inequality [8] that a natural extension of Spencer's algorithm yields an IS of expected size at least $\alpha_k(H)$, when applied to any k-uniform hypergraph H for any $k \geq 2$. For our purposes, we recall a special case of the FKG inequality; the reader is referred to [3] for more about the inequality. Given a vector $\vec{Y} = (Y_1, Y_2, \ldots, Y_\ell)$ of *independent* random variables $Y_i \in \{0, 1\}$ and an event F that is completely determined by the Y_i's, call F *increasing* iff the following holds: for any $\vec{a}$ such that F holds when $\vec{Y} = \vec{a}$, F also holds when $\vec{Y} = \vec{b}$ for any $\vec{b}$ that co-ordinatewise dominates $\vec{a}$ (i.e., $a_i \leq b_i$ for all i). Then, for any collection of *increasing* events $F_1, F_2, \ldots, F_t$, the FKG inequality shows that

$$\text{Prob}(\bigwedge_{i=1}^{t} F_i) \geq \prod_{i=1}^{t} \text{Prob}(F_i). \tag{4}$$

Consider the following RNC algorithm, $\mathcal{A}_S$, for finding an IS in an arbitrary, not necessarily uniform, hypergraph $H = (V, E)$. Randomly permute the vertices; add a vertex $v \in V$ to the IS iff there is no edge $e \in E$ such that $v \in e$ and v is *last* among the vertices of e in the random order. It is easy to check that we produce a valid IS in this fashion. A specific way of implementing $\mathcal{A}_S$ is given in Figure 1. As will be seen in the proof of Lemma 1, the method of generating random permutations that we adopt, provides sufficient independence to employ tools such as the FKG

Algorithm $\mathcal{A}_S$:
Independently for all $v \in V$ do:
  Sample $X_v \in [0,1)$ using the uniform distribution on $[0,1)$.
Define a permutation π of the vertices in which $\pi(v) < \pi(u)$ *iff* $X_v < X_u$.
$I := \emptyset$;
for all $v \in V$ do
  {
    $j_v = true$.
    For all $e \in E$ such that $v \in e$
      if $\pi(v) = \max_{u \in e} \pi(u)$ then $j_v = false$.
    If j_v then $I := I \cup \{v\}$;
  }
return(I).

Figure 1: The algorithm $\mathcal{A}_S$

inequality. Note that in the special case where H is a graph, $\mathcal{A}_S$ reduces to Spencer's algorithm.

It is readily seen that $\mathcal{A}_S$ can be implemented in RNC. For each edge, we first choose the vertex u in it of highest X_u value; removing duplicates from this multi-set of chosen vertices (e.g., through sorting) yields the set of vertices that will *not* lie in our IS. This is easily done on a CREW PRAM in $O(\lg(m+n))$ steps, using $(\sum_{e_i \in E} |e_i|) \leq mn$ processors. Also, since this algorithm is simple and just uses some basic primitives, it should be easy to implement in other parallel/distributed settings.

2.2 Analysis of the performance of $\mathcal{A}_S$

Given a hypergraph H, denote by B_v the event that vertex v is in the final IS produced by $\mathcal{A}_S$. Our basic tool will be Lemma 1. Before presenting the lemma, we recall that a *linear hypergraph* is one in which every pair of distinct edges share at most one vertex. Linear hypergraphs have been studied in the context of parallel construction of MISs, and NC algorithms are known for the MIS problem on linear hypergraphs [19, 25]. Hence, we also see how well our algorithm does on linear hypergraphs: Lemma 1 also helps provide an exact bound on our algorithm's performance in this case.

LEMMA 1. *Suppose a vertex v lies in edges $e_1, e_2, \ldots, e_t$, whose respective cardinalities are $k_1, k_2, \ldots, k_t$. Then,*

$$Prob(B_v) \geq \int_0^1 [\prod_{i=1}^{t} (1 - x^{k_i - 1})]\, dx.$$

Furthermore, this inequality becomes an equality in the case of linear hypergraphs.

PROOF. Recall the random variables X_u from Figure 1. The main idea behind our proof is that the computations become tractable once we condition on the value of X_v. As we will see, the fact that the X_u's are independent will help us much: this way of introducing independence into a choice of permutations helps us use tools such as the FKG inequality. Let $x \in [0,1)$ be arbitrary, and define, for all $u \neq v$, the random variable $Y_u = 1$ if $X_u > x$, and $Y_u = 0$ otherwise. For each edge e, define the random variable $C(e)$ to be 1 if $\max_{u:u \in e} X_u > x$, and $C(e)$ to be 0 otherwise. Then,

$$\text{Prob}(B_v | X_v = x) = \text{Prob}([\bigwedge_{e: v \in e} C(e)] | X_v = x).$$

Now, even conditional on $X_v = x$, the random variables Y_u are independent with $\text{Prob}(Y_u = 1) = 1 - x$. Also, conditional on $X_v = x$, each $C(e)$ is determined by the values of the Y_u, and is increasing as a function of the Y_u. Thus, by the FKG inequality,

$$\begin{aligned} \text{Prob}(B_v | X_v = x) &\geq \prod_{i=1}^{t} \text{Prob}(C(e_i) | X_v = x) \\ &= \prod_{i=1}^{t} (1 - x^{k_i - 1}). \end{aligned} \tag{5}$$

The first part of the lemma now follows from the fact that

$$\text{Prob}(B_v) = \int_0^1 \text{Prob}(B_v | X_v = x)\, dx.$$

It is also easy to check that the inequality in (5) becomes an equality in the case of linear hypergraphs. Hence, the inequality of this lemma is in fact an equality for linear hypergraphs. ■

Applying the linearity of expectation, we get the following theorem on the expected quality of the IS produced by $\mathcal{A}_S$ on an arbitrary hypergraph:

THEOREM 1. *Suppose we are given an arbitrary hypergraph $H = (V, E)$ with a weight $w_v \geq 0$ for each vertex v. Suppose each vertex v lies in $d(v)$ edges, whose cardinalities are $k_{v,1}, k_{v,2}, \ldots, k_{v,d(v)}$. Then, the expected weight of the IS produced by $\mathcal{A}_S$ is at least*

$$\sum_v (w_v \cdot \int_0^1 [\prod_{i=1}^{d(v)} (1 - x^{k_{v,i} - 1})]\, dx);$$

in the case of linear hypergraphs, this lower bound is an exact bound on the expected weight.

The next theorem considers the performance of $\mathcal{A}_S$ for unweighted uniform hypergraphs, and shows that the expected size of the IS produced is at least as large as $\alpha_k(H)$:

THEOREM 2. *For any $k \geq 2$ and any k-uniform hypergraph H, $\mathcal{A}_S$ finds an IS of expected size at least $\alpha_k(H)$.*

PROOF. We will use the following identity from [11], which holds for any non-negative integer d and any real x that does not lie in the set $\{0, \ldots, -d\}$:

$$\sum_{l=0}^{d} \binom{d}{l} \frac{(-1)^l}{x+l} = \frac{1}{x \binom{d+x}{d}}. \tag{6}$$

Specialized to k-uniform hypergraphs, Lemma 1 shows

that

$$\begin{aligned}\text{Prob}(B_v) &\geq \int_0^1 (1-x^{k-1})^{d(v)}\ dx \\ &= \sum_{l=0}^{d(v)} (-1)^l \binom{d(v)}{l} \int_0^1 x^{(k-1)l}\ dx \\ &= \sum_{l=0}^{d(v)} (-1)^l \binom{d(v)}{l} \frac{1}{1+(k-1)l} \\ &= \binom{d(v)+1/(k-1)}{d(v)}^{-1},\end{aligned}$$

by (6).

Summing over all the vertices and applying the linearity of expectation completes the proof. ■

We remark that any algorithm that works for k-uniform hypergraphs and whose output solution is a non-increasing function of each vertex-degree (as is the function $\alpha_k(H)$), can be immediately extended to give the same guarantee for hypergraphs in which all edges have size *at least* k. We simply replace each edge by an arbitrary subset of it of size k, to achieve this. Thus, our results such as Theorem 2 also hold for hypergraphs with at least k vertices in each edge. However, in various families of non-uniform hypergraphs we can do better than this simple approach, as we demonstrate in Theorem 3.

We need some notation. Suppose each vertex v lies in $D(j,v)$ edges of cardinality $k(j,v)$, for $j = 1,2,\ldots,a(v)$. (In other words, vertex v lies in edges of $a(v)$ different sizes: $k(j,v)$, for $j = 1,2,\ldots,a(v)$. If H is a uniform hypergraph, then $a(v) \equiv 1$.) Define

$$f(v) = \min_{j=1,2,\ldots,a(v)} [D(j,v)]^{-1/(k(j,v)-1)},$$

and let $b(v) = \min_j (k(j,v)-1)$. Then, Theorem 3 shows that given a weight $w(v)$ for each vertex, $\mathcal{A}_S$ produces an IS of expected total weight at least $\Omega(\sum_v (w(v)/a(v)^{1/b(v)}) \cdot f(v))$. We prove this by lower-bounding the quantity guaranteed by Theorem 1.

Our proof of Theorem 3 shows that the quantity guaranteed by Theorem 1 is at most $O(\sum_v w(v) f(v))$; so our "closed-form" bound $\Omega(\sum_v (w(v)/a(v)^{1/b(v)}) \cdot f(v))$ approximates the guarantee of Theorem 1 well when $a(v)$ is "small" or $b(v)$ is "large".

THEOREM 3. *In a hypergraph $H = (V,E)$, let the notation $a(v)$, $b(v)$ and $f(v)$ be as above. Then, given weights $w(v) \geq 0$ for the vertices, the expected weight of the IS produced by $\mathcal{A}_S$ is at least $\Omega(\sum_v (w(v)/a(v)^{1/b(v)}) \cdot f(v))$.*

PROOF. Fix a vertex v. For notational simplicity, let $a = a(v)$, $b = b(v)$, $k(j) = k(j,v)$, and $f = f(v)$. By Theorem 1, it suffices to show that

$$\beta \doteq \int_0^1 [\prod_{j=1}^{a} (1-x^{k(j)-1})^{D(j)}]\ dx \geq \Omega(f/a^{1/b}). \qquad (7)$$

Let $s = s(v) = \text{argmin}_{j=1,2,\ldots,a} [D(j)]^{-1/(k(j)-1)}$. The bound (3) and the proof of Theorem 2 help show that

$$\beta \leq \int_0^1 (1-x^{k(s)-1})^{D(s)}\ dx = \Theta(f);$$

thus, (7) is tight to within a constant factor if, e.g., $a(v)$ is bounded by an absolute constant. (In particular, (7) is a tight bound for hypergraphs of constant maximum degree.)

We now prove (7). Let $t = f/a^{1/b}$, and note that $t \in [0,1]$. Let $\exp(x) \doteq e^x$. We can show that

$$\begin{aligned}\gamma &\doteq \prod_{j=1}^{a} (1-t^{k(j)-1})^{D(j)} \\ &\geq \exp(-O(\sum_{j=1}^{a} D(j) t^{k(j)-1})) \\ &\geq \exp(-O(\sum_{j=1}^{a} D(j) \frac{f^{k(j)-1}}{a})) \quad \text{(by the definition of } b\text{)} \\ &= \exp(-O((1/a) \cdot \sum_{j=1}^{a} D(j) f^{k(j)-1})) \\ &\geq \exp(-O((1/a) \cdot \sum_{j=1}^{a} 1)) \quad \text{(by the definition of } f\text{)} \\ &= \Omega(1).\end{aligned}$$

Thus,

$$\beta \geq \int_0^t [\prod_{j=1}^{a} (1-x^{k(j)-1})^{D(j)}]\ dx \geq \int_0^t \gamma\ dx = t\gamma \geq \Omega(t),$$

establishing (7). ■

3. NC ALGORITHMS

We now explore how one may derandomize our results for various special cases; we will work throughout with the weighted case, and with hypergraphs that need not be uniform. We are able to develop NC versions for two cases via different methods. The unifying idea behind these two results is as follows. Suppose we consider, for any given constant $c > 0$, the family of hypergraphs with maximum degree at most $c \lg n$, where n denotes the number of vertices as usual. The basic observation is that for any vertex v, $\text{Prob}(B_v)$ can be evaluated in NC as follows. Suppose vertex v lies in edges $e_{v,1}, e_{v,2}, \ldots, e_{v,d(v)}$, where $d(v) \leq c \lg n$ by assumption. Let $[k]$ denote the set $\{1,2,\ldots,k\}$. Denote by $C_{v,u}$ the event "$X_v \geq X_u$"; then, by inclusion-exclusion,

$$\begin{aligned}Pr(B_v) &= \sum_{S \subseteq [d(v)]} (-1)^{|S|} \cdot Pr(\bigwedge_{i \in S} [\forall u \in e_{v,i},\ C_{v,u}]) \\ &= \sum_{S \subseteq [d(v)]} (-1)^{|S|} \cdot Pr(\forall u \in [(\bigcup_{i \in S} e_{v,i}) \backslash \{v\}], C_{v,u}) \\ &= \sum_{S \subseteq [d(v)]} (-1)^{|S|} \cdot (1 + |(\bigcup_{i \in S} e_{v,i}) - \{v\}|)^{-1}; \quad (8)\end{aligned}$$

(8) follows from the fact that each X_u, $u \in \bigcup_{i \in S} e_{v,i}$, is equally likely to be $\max\{X_w :\ w \in \bigcup_{i \in S} e_{v,i}\}$. Since there are at most $2^{c \lg n} = n^c$ terms in (8), the sum can be evaluated in NC. We shall briefly sketch how this leads to two types of derandomized versions of our RNC algorithm.

3.1 The Case of Constant Maximum Degree

We now sketch how to find an IS of weight at least $(1-\epsilon)$ times the expected value presented in Theorem 1, for the case where all vertex-degrees are bounded by some constant d; ϵ here denotes a given arbitrary positive constant.

Denote by S_n the set of all permutations of $[n]$. A family of permutations $\mathcal{F} \subseteq S_n$ is defined in [5] to be *approximately min-wise independent with relative error* $\delta > 0$, if for all $X \subseteq [n]$, we have for a randomly chosen permutation $\pi \in F$ that

$$|\text{Prob}(\min\{\pi(X)\} = \pi(x)) - 1/|X|| \leq \delta/|X|.$$

We abbreviate the above property by (n, δ)-amw. We will use the property that for any n and $\delta > 0$, there is an explicitly constructible permutation family $F(n, \delta)$ that satisfies property (n, δ)-amw, and has cardinality $n^{O(\lg(1/\delta))}$ [12]. Let $\delta = O(\epsilon/(d2^d))$ be a suitable constant. In the full version, we shall show that for a randomly chosen permutation π from $F(n, \delta)$ and every vertex v, Prob(B_v) is at least $(1-\epsilon)$ times the value guaranteed by Lemma 1. Thus, by the linearity of expectation, the expected size of an IS produced by using a random element of $F(n, \delta)$ is at least $(1 - \epsilon)$ times the value guaranteed by Theorem 1. (In other words, we show that instead of permuting the vertices completely at random, it suffices to choose a random permutation from the explicit polynomial-sized set $F(n, \delta)$.) We can then apply parallel exhaustive search on the polynomial-sized $F(n, \delta)$ to find a "good" permutation in NC.

The proof ideas involve (3) and the above discussion on the inclusion-exclusion expansion, and will be presented in the full version.

THEOREM 4. *Consider the family of hypergraphs with maximum degree at most d, for any given constant $d > 0$. Then, given any constant $\epsilon > 0$, there is an NC algorithm to find an IS in these hypergraphs of total weight at least $(1 - \epsilon)$ times the expected weight guaranteed by Theorem 1.*

3.2 Logarithmic Degree and Sparse Neighborhoods

We now consider the case of hypergraphs satisfying properties (P2) and (P3) defined in item (iii) of Section 1.2. For such hypergraphs, we now sketch how to find, in NC, an IS of total weight at least $(1 - \epsilon)$ times the expected weight guaranteed by Theorem 1. The parameter ϵ can now in fact be n^{-c} for any given constant $c > 0$. Complete details and proofs will be presented in the full version of this work.

The basic idea here is as follows. Let $t = \lceil 3 \lg n \rceil$. Suppose we choose a random t-bit vector $s_v = s_{v,t-1}s_{v,t-2} \cdots s_{v,0}$ independently for each vertex v, and then define a random permutation of the vertices by sorting them according to the s_v (by interpreting the s_v as integers in $\{0, 1, \ldots, 2^t - 1\}$). It is easy to see that this changes the analysis of Section 2.2 very little, since: (a) with high probability, all the s_v's will be distinct, and (b) if condition (a) holds, then we indeed produce a random permutation of the vertices. Our goal is to make a "good" choice of the vectors s_v deterministically, in NC. To do so, we proceed as follows. Suppose the vertices are numbered $1, 2, \ldots, n$. For $i = 0, 1, \ldots, t-1$, define r_i to be the vector $s_{1,i}s_{2,i} \cdots s_{n,i}$. We aim to make "good" choices for the r_i one-by-one, in the order $i = t-1, t-2, \ldots, 0$, using a basic idea developed in [2]: make these choices one-by-one so that the conditional expectation of the total weight of the final IS falls by only a tiny amount after choosing each additional r_i. (The key difference with the method of conditional probabilities is that we allow the conditional expectation to fall from one iteration to the next, but carefully limit the amount of this fall.) By employing the above-seen inclusion-exclusion expansion to evaluate the conditional expectation of the total weight of the final IS, we show that it suffices to choose each r_i at random from a polynomial-sized *small-bias* space [22], instead of choosing from $\{0, 1\}^n$. Thus, we will be able to choose a "good" value for each additional r_i in NC, via parallel exhaustive search.

THEOREM 5. *Consider the family of hypergraphs satisfying properties (P2) and (P3) of Section 1.2. Then, given any constant $c > 0$ and any $\epsilon \geq n^{-c}$, there is an NC algorithm to find an IS in these hypergraphs of total weight at least $(1 - \epsilon)$ times the expected weight guaranteed by Theorem 1.*

4. DISCUSSION

A question that remains open is to obtain a full derandomization of our RNC algorithms. Any progress on the classical MIS problem on hypergraphs would also be most interesting.

Acknowledgments. We thank Noga Alon for valuable discussions, and for pointing out to us the results by Caro and Tuza. We also thank the SPAA 2001 program committee member(s) and referee(s) for their helpful comments.

5. REFERENCES

[1] N. Alon, L. Babai and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. of Algorithms*, 7, pp. 567–583, 1986.

[2] N. Alon and M. Naor. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16:434–449, 1996.

[3] N. Alon and J. H. Spencer. The Probabilistic Method. Wiley-Interscience, 1992.

[4] P. Beame and M. Luby. Parallel search for maximal independence given minimal dependence. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 212–218, 1990.

[5] A. Z. Broder, M. Charikar, A. M. Frieze and M. Mitzenmacher. Min-wise independent permutations. In *Proc. ACM Symposium on Theory of Computing*, 1998.

[6] Y. Caro and Z. Tuza. Improved lower bounds on k-independence. *J. Graph Theory*, 15, pp. 99–107, 1991.

[7] E. Dahlhaus, M. Karpinski, and P. Kelsen. An efficient parallel algorithm for computing a maximal independent set in a hypergraph of dimension 3. *Information Processing Letters*, 42(6):309–314, 1992.

[8] C. M. Fortuin, J. Ginibre, P. N. Kasteleyn, Correlational Inequalities for Partially Ordered Sets, *Communications of Mathematical Physics*, 22, pp. 89–103, 1971.

[9] M. Goldberg and T. Spencer. A new parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 18:419–427, 1989.

[10] M. Goldberg and T. Spencer. An efficient parallel algorithm that finds independent sets of guaranteed size. *SIAM J. Disc. Math.*, 6:443–459, 1993.

[11] M. Hofri. Analysis of Algorithms. Oxford University Press, 1995.

[12] P. Indyk. A small approximately min-wise independent family of hash functions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 454–456, 1999.

[13] D. R. Karger and D. Koller. (De)randomized construction of small sample spaces in *NC*. *Journal of Computer and System Sciences*, 55 (1997), pp. 402–413.

[14] R. M. Karp and V. Ramachandran. Parallel algorithms for shared memory machines. In *Handbook of Theoretical Computer Science, Volume A*, J. van Leeuwen, Editor, Elsevier, New York, pages 871–941, 1990.

[15] R. M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36:225–253, 1988.

[16] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *J. Assoc. Comput. Mach.*, 32:762–773, 1985.

[17] P. Kelsen. On the parallel complexity of computing a maximal independent set in a hypergraph. In *Proc. ACM Symposium on Theory of Computing*, pages 339–350, 1992.

[18] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15 (1986), pp. 1036–1053.

[19] T. Łuczak and E. Szymańska. A Parallel Randomized Algorithm for Finding a Maximal Independent Set in a Linear Hypergraph. *J. Algorithms*, 25:311–320, 1997.

[20] R. J. McEliece and K. N. Sivarajan. Performance limits for channelized cellular telephone systems. *IEEE Trans. Info. Theory*, 40(1):21–34, 1994.

[21] R. Motwani, P. Raghavan, *Randomized Algorithms*, Cambridge Univ. Press, 1995.

[22] J. Naor and M. Naor. Small–bias probability spaces: efficient constructions and applications. *SIAM Journal on Computing*, 22:838–856, 1993.

[23] J. H. Spencer. The probabilistic lens: Sperner, Turán and Brégman revisited. In *A Tribute to Paul Erdős* (A. Baker, B. Bollobás, A. Hajnal, Eds.), Cambridge Univ. Press, pp. 391–396, 1990.

[24] A. Srinivasan. New approaches to covering and packing problems. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 567–576, 2001.

[25] E. Szymańska. Derandomization of a Parallel MIS Algorithm in a Linear Hypergraph. In *Proc. Fourth International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 39–52, 2000.

[26] P. Turán. On the theory of graphs. *Colloq. Math.*, 3, pp. 19–30, 1954.