

## Derandomization techniques

- ▶ In what follows, “graph” refers to a finite undirected graph.
- ▶ A *cut* of a graph  $G = (V, E)$  is a partition of  $V$  into two disjoint subsets  $V_0$  and  $V_1$ .  
The *weight* of a cut  $(V_0, V_1)$  is the number of edges between  $V_0$  and  $V_1$ , i.e., the weight is

$$|\{ \{v_0, v_1\} \in E : v_0 \in V_0, v_1 \in V_1 \}|.$$

### Algorithm Cut

Input: A graph  $G = (V, E)$  where  $V = \{1, \dots, n\}$ .

Choose random bits  $r_1, \dots, r_n$  by independent tosses  
of a fair coin.

Let  $V_0 = \{i : r_i = 0\}$ .

Let  $V_1 = \{i : r_i = 1\}$ .

Output: The cut  $(V_0, V_1)$ .

## The Algorithm Cut

### Proposition

Let  $G$  be a graph with  $m$  edges. On input  $G$ , the expected weight of the cut returned by Algorithm Cut is  $m/2$ . In particular, the graph  $G$  has a cut with weight at least  $m/2$ .

### Proof

- ▶ Let  $G = (V, E)$  be a graph with  $m$  edges  $e_1, \dots, e_m$ .  
Introduce indicator variables  $\hat{e}_i$  where  $\hat{e}_i = 1$  iff edge  $e_i = \{u_i, v_i\}$  crosses the cut returned by Algorithm Cut.
- ▶ The weight  $\widehat{w}_G$  of the cut returned by Algorithm Cut is just the sum of the  $\hat{e}_i$ , hence by linearity of expectation we have

$$\mathbf{E}[\widehat{w}_G] = \mathbf{E}[\hat{e}_1 + \dots + \hat{e}_m] = \mathbf{E}[\hat{e}_1] + \dots + \mathbf{E}[\hat{e}_m].$$

## Verification of the Algorithm Cut

### Proof (continued)

- ▶ So it suffices to show  $\mathbf{E}[\hat{e}_i] = 1/2$ . Indeed, we have

$$\begin{aligned} \mathbf{E}[\hat{e}_i] &= 1 \cdot \text{Prob}[\hat{e}_i = 1] + 0 \cdot \text{Prob}[\hat{e}_i = 0] \\ &= \text{Prob}[u_i \in V_0 \text{ and } v_i \in V_1] + \text{Prob}[u_i \in V_1 \text{ and } v_i \in V_0] \\ &\stackrel{(*)}{=} \text{Prob}[u_i \in V_0] \text{Prob}[v_i \in V_1] \\ &\quad + \text{Prob}[u_i \in V_1] \text{Prob}[v_i \in V_0] \\ &= \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} \end{aligned}$$

Equation (\*) holds because the assignments of nodes to the two sides of the cut are mutually and hence pairwise independent.

## Verification of the Algorithm Cut

### Proof (continued)

- ▶ If all possible outcomes of the coin tosses resulted in a cut with weight strictly less than  $m/2$ , then  $\mathbf{E}[\widehat{w}_G]$  would be strictly less than  $m/2$ .  
Hence there is a sequence of coin tosses that yield a cut with weight at least  $m/2$ .  
Hence there is a cut with weight at least  $m/2$ .  $\square$

## Derandomization by conditional expectation

- ▶ Let  $G = (V, E)$  be a graph with  $n$  nodes and  $m$  edges and let  $\alpha$  be any word of length  $s \leq n$ .
  - ▶ Let  $\text{Cut}_\alpha$  be the algorithm that works like Algorithm Cut, except that  $r_1 \dots r_s$  is set equal to  $\alpha$ .
  - ▶ Let  $\widehat{w}_G(\alpha)$  be the weight of the random cut returned by Algorithm  $\text{Cut}_\alpha$  on input  $G$ .  
Note that  $\mathbf{E}[\widehat{w}_G(\lambda)] = \mathbf{E}[\widehat{w}_G] = \frac{m}{2}$  ( $\lambda$  is the empty word).
  - ▶ Algorithm  $\text{Cut}_\alpha$ , with probability of  $1/2$  each,
    - sets  $r_{s+1}$  to 0, i.e., works like  $\text{Cut}_{\alpha 0}$ ,
    - sets  $r_{s+1}$  to 1, i.e., works like  $\text{Cut}_{\alpha 1}$ .
- In term of the expected weight of the returned cut, this means

$$\mathbf{E}[\widehat{w}_G(\alpha)] = \frac{1}{2}\mathbf{E}[\widehat{w}_G(\alpha 0)] + \frac{1}{2}\mathbf{E}[\widehat{w}_G(\alpha 1)],$$

Hence at least one of the expected values on the right-hand side is at least as large as  $\mathbf{E}[\widehat{w}_G(\alpha)]$ .

## Derandomization by conditional expectation

- ▶ Is the derandomized version of Cut efficient?  
How complex is it to evaluate Condition (\*)?
- ▶ Consider iteration  $s$  of the for-loop of the algorithm. Assume that  $r_1 \dots r_{s-1}$  have already been defined. Partition  $E$  into four sets  $E_1, E_2, E_3^0$ , and  $E_3^1$  defined by
 
$$\begin{aligned} E_1 &= \{ \{j_1, j_2\} \in E : j_1 < s \text{ and } j_2 < s \} \\ E_2 &= \{ \{j_1, j_2\} \in E : j_1 > s \text{ or } j_2 > s \} \\ E_3^i &= \{ \{j, s\} \in E : j < s \text{ and } r_j = i \}. \end{aligned}$$
- ▶ For the edges in  $E_1$  and  $E_2$ , the choice of  $r_s$  does not matter. The constructed cut will contain from  $E_1$  exactly the edges in

$$E_1' = \{ \{j_1, j_2\} \in E_1 : r_{j_1} \neq r_{j_2} \}.$$

The constructed cut will contain each edge in  $E_2$  with probability  $1/2$ .

## Derandomization by conditional expectation

### Derandomized Algorithm Cut (conditional expectation)

Input: A graph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

For  $s = 1, \dots, n$

If  $\mathbf{E}[\widehat{w}_G(r_1 \dots r_{s-1} 0)] \geq \mathbf{E}[\widehat{w}_G(r_1 \dots r_{s-1} 1)]$  (\*)

then  $r_s = 0$  else  $r_s = 1$ .

Let  $V_0 = \{i : r_i = 0\}$ .

Let  $V_1 = \{i : r_i = 1\}$ .

Output: The cut  $(V_0, V_1)$ .

- ▶ The algorithm returns a cut of weight  $w_G \geq \frac{m}{2}$  because of

$$\begin{aligned} \frac{m}{2} &= \mathbf{E}[\widehat{w}_G(\lambda)] \leq \mathbf{E}[\widehat{w}_G(r_1)] \leq \mathbf{E}[\widehat{w}_G(r_1 r_2)] \leq \dots \\ &\leq \mathbf{E}[\widehat{w}_G(r_1, \dots, r_{n-1})] \leq \mathbf{E}[\widehat{w}_G(r_1, \dots, r_n)] = w_G \end{aligned}$$

## Derandomization by conditional expectation

- ▶ Concerning the sets  $E_3^i$ , the choice of  $r_s$  does matter. The constructed cut contains all edges from  $E_3^{1-r_s}$  but no edge from  $E_3^{r_s}$ , hence has an expected weight of

$$\mathbf{E}[\widehat{w}_G(r_1 \dots r_{s-1} r_s)] = |E_1'| + \frac{1}{2}|E_2| + |E_3^{1-r_s}|.$$

- ▶ The expected weight is maximized by maximizing  $|E_3^{1-r_s}|$ , i.e., by maximizing the number of edges between node  $s$  and the nodes  $1, \dots, s-1$ .
- ▶ This means that we let  $r_s = 0$  in case

$$|\{j < s : \{j, s\} \in E \text{ and } r_j = 1\}|$$

is at most as large as

$$|\{j < s : \{j, s\} \in E \text{ and } r_j = 0\}|,$$

and otherwise, we let  $r_s = 1$ .

## Derandomization by conditional expectation

### Derandomized Algorithm Cut (conditional expectation)

Input: A graph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

For  $s = 1, \dots, n$

If  $|\{j < s : \{j, s\} \in E \text{ and } r_j = 1\}|$   
 $\geq |\{j < s : \{j, s\} \in E \text{ and } r_j = 0\}|$

then  $r_s = 0$  else  $r_s = 1$

Let  $V_0 = \{i : r_i = 0\}$ .

Let  $V_1 = \{i : r_i = 1\}$ .

Output: The cut  $(V_0, V_1)$ .

- ▶ We end up with an extremely simple deterministic algorithm.
- ▶ The randomized version of the algorithm is still used for verifying that the deterministic version works as required.

## Derandomization by pairwise independence

- ▶ *Trivial derandomization* refers to simulating a randomized algorithm for every possible value of its random source.
- ▶ Consider a randomized algorithm that runs for  $t(n)$  steps on all inputs of size  $n$ .  
The algorithm might use up to  $t(n)$  random bits, hence its trivial derandomization runs for about  $2^{t(n)}t(n)$  steps.
- ▶ Now assume that the algorithm uses only  $\log t(n)$  many random bits. Then its trivial derandomization runs for a number of steps that is about

$$2^{\log t(n)}t(n) = t(n) \cdot t(n) = t(n)^2.$$

### Derandomization method of small sample spaces

First transform a randomized algorithm into one that uses only few random bits, then apply trivial derandomization.

## Derandomization by pairwise independence

- ▶ For suitable randomized algorithms, the sample space can be made small by working with  $k$ -wise instead of mutually independent random bits.
- ▶ The verification of Algorithm Cut relied solely on the fact that each bit  $r_1, \dots, r_n$  is uniformly distributed and that these bits are *pairwise* independent.
- ▶ How many mutually independent uniformly distributed random bits are required to specify such  $n$  pairwise independent bits?  
That is, how small can we choose the sample space?

### About $\log n$ bits are sufficient!

This yields a sample space of size about  $n$ .

Trivial derandomization then yields an algorithm that runs in polynomial time.

## Derandomization by pairwise independence

### Construction of $n$ pairwise independent random bits with uniform distribution from a small sample space

For given  $n$ , let  $t = \lceil \log(n+1) \rceil$  and let  $I = \{1, \dots, t\}$ .

Let random bits  $b_1, \dots, b_t$  be obtained by tosses of a fair coin.

Let  $J_1, \dots, J_n$  be pairwise distinct nonempty subsets of  $I$ .

For  $i = 1, \dots, n$  let  $r_i = \bigoplus_{j \in J_i} b_j$ .

In connection with this construction observe that

there are  $2^t - 1 \geq n$  pairwise distinct nonempty subsets of  $I$ ,  
the sample space has a size of  $2^t \leq 2n$ , where equality holds  
in case  $n$  can be written as  $n = 2^k$  for some natural number  $k$ .

## Derandomization by pairwise independence

### Lemma (Verification of the construction)

The random bits  $r_1, \dots, r_n$  obtained by the construction above are pairwise independent and each bit is uniformly distributed.

### Proof.

Fix any  $t \geq 0$  and assume that random bits  $b_1, \dots, b_t$  are obtained by tosses of a fair coin.

Fix any nonempty set  $J \subseteq \{1, \dots, t\}$  and let  $r = \bigoplus_{j \in J} b_j$ .

For any fixed index  $s \in J$  we have,

$$r = \bigoplus_{j \in J} b_j = (\bigoplus_{j \in J \setminus \{s\}} b_j) \oplus b_s.$$

That is, if the  $b_j$  have already been determined for all  $j$  in  $J \setminus \{s\}$ , then depending on the value of  $b_s$ , the value of  $r$  will either agree with or will differ from  $\bigoplus_{j \in J \setminus \{s\}} b_j$ . But  $b_s$  attains its two possible values with probability  $1/2$  each and the  $b_j$  are mutually independent, hence  $r$  will be uniformly distributed in  $\{0, 1\}$ .

## Derandomization by pairwise independence

### Proof (continued).

For any two distinct nonempty subsets  $J$  and  $J'$  of  $\{1, \dots, t\}$ , let

$$r = \bigoplus_{j \in J} b_j \quad \text{and} \quad r' = \bigoplus_{j \in J'} b_j.$$

Since the sets  $J$  and  $J'$  are distinct, there is some index  $s$  that is contained in one of the sets but not in the other; w.l.o.g. we assume  $s \in J$ .

Similarly to the proof of uniformity, we can then argue that

$$r = \bigoplus_{j \in J} b_j = (\bigoplus_{j \in J \setminus \{s\}} b_j) \oplus b_s,$$

and that hence if the  $b_j$  have already been determined for all  $j$  except  $s$ , then depending on the value of  $b_s$ , i.e., with probability  $1/2$ , the value of  $r$  will agree with or will differ from  $r'$ , consequently  $r$  and  $r'$  are mutually independent.

## Derandomization by pairwise independence

### Algorithm Cut<sub>pi</sub> (Using pairwise independent random bits)

Input: A graph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

Let  $t = \lceil \log(n+1) \rceil$ .

Let  $I = \{1, \dots, t\}$ .

Choose random bits  $b_1, \dots, b_t$  by tosses of a fair coin.

Let  $J_1, \dots, J_n$  be pairwise distinct nonempty subsets of  $I$ .

For  $i = 1, \dots, n$ , let  $r_i = \bigoplus_{j \in J_i} b_j$ .

Let  $V_0 = \{i : r_i = 0\}$ .

Let  $V_1 = \{i : r_i = 1\}$ .

Output: The cut  $(V_0, V_1)$ .

- ▶ The expected weight of the cut returned by Algorithm Cut<sub>pi</sub> is  $\frac{m}{2}$ , the analysis is essentially the same as for Algorithm Cut.
- ▶ If we let  $w_1, \dots, w_n$  be the lexicographically least  $n$  words of length  $t$  that differ from  $0^t$ , then we can simply let  $J_i = \{j : w_i^j = 1\}$ , where  $w_i = w_i^1 \dots w_i^t$ ,  $w_i^j \in \{0, 1\}$ .

## Derandomization by pairwise independence

### Derandomized Algorithm Cut (pairwise independence)

Input: A graph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

Let  $t = \lceil \log(n+1) \rceil$  and let  $m = 2^t$ .

Let  $u_1, \dots, u_m$  be the  $m$  words of length  $t$ .

Let  $w_1, \dots, w_n$  be the least  $n$  words of length  $t$  that differ from  $0^t$ .

For  $i = 1, \dots, m$ ,

For  $j = 1, \dots, n$ ,

Let  $r_{i,j} = \bigoplus_{t=1, \dots, t} (u_i^t \wedge w_j^t)$ . ( $\wedge$  is the And-Operator)

For  $j = 1, \dots, n$ ,

Let  $V_0^j = \{i : r_{i,j} = 0\}$ .

Let  $V_1^j = \{i : r_{i,j} = 1\}$ .

Output: A cut  $(V_0^j, V_1^j)$  of maximum weight.

- ▶ The  $(m \times n)$ -matrix  $R$  with entries  $r_{i,j}$  can be viewed as the product of the  $(m \times t)$ -matrix  $U$  with rows  $u_1, \dots, u_m$  and the  $(t \times n)$ -matrix  $W$  with columns  $w_1, \dots, w_n$ .

## Derandomization of Algorithm HyperCut

- ▶ A *hypergraph* is a pair  $(V, E)$  where  $V$  is the set of nodes and  $E$  is a set of subsets of  $V$ .  
The subsets in  $E$  are called *hyperedges*.
- ▶ A hypergraph is *k-regular* if all its hyperedges contain  $k$  nodes.
- ▶ A *cut* of a hypergraph  $G = (V, E)$  is a partition of  $V$  into two disjoint subsets  $V_0$  and  $V_1$ .  
The *weight of a cut*  $(V_0, V_1)$  is the number of hyperedges that intersect both  $V_0$  and  $V_1$ .

### Algorithm HyperCut

Input: A hypergraph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

Choose random bits  $r_1, \dots, r_n$  by independent tosses of a fair coin.

Let  $V_0 = \{i : r_i = 0\}$ .  
Let  $V_1 = \{i : r_i = 1\}$ .

Output: The cut  $(V_0, V_1)$ .

## Derandomization of Algorithm HyperCut

### Proposition

Let  $G$  be a hypergraph that is  $k$ -regular for some  $k \geq 2$  and has  $m$  edges. The expected weight of the cut returned by Algorithm HyperCut on input  $G$  is

$$\left(1 - \frac{2}{2^k}\right) m.$$

In particular, the graph  $G$  has a cut of at least this weight.

### Proof.

For each of the edges  $e_1, \dots, e_m$  of  $G$  introduce an indicator variable  $\hat{e}_i$  where  $\hat{e}_i = 1$  iff edge  $e_i$  crosses the returned cut. Then  $\mathbf{E}[\hat{e}_i] = (1 - \frac{2}{2^k})$ , because the probability that  $e_i$  does not cross the cut, i.e., that  $e_i$  is either contained in  $V_0$  or  $V_1$  is  $2/2^k$ . The weight of the returned cut is the sum over the  $\hat{e}_i$ , hence the proposition follows by linearity of expectation.  $\square$

## Derandomization of Algorithm HyperCut

- ▶ Let  $G = (V, E)$  be a hypergraph with  $n$  nodes and  $m$  edges.
- ▶ Let  $\alpha$  be any word of length  $s \leq n$   
Let  $\text{HyperCut}_\alpha$  be the algorithm that works like Algorithm HyperCut, except that  $r_1 \dots r_s$  is set equal to  $\alpha$ .
- ▶ Let  $\hat{w}_G(\alpha)$  be the weight of the random cut returned by Algorithm  $\text{HyperCut}_\alpha$  on input  $G$ .

$$\mathbf{E}[\hat{w}_G(\lambda)] = \mathbf{E}[\hat{w}_G] = (1 - 2/2^k)m.$$

- ▶ With probability of 1/2 each,  $\text{HyperCut}_\alpha$ 
    - ▶ sets  $r_{s+1}$  to 0, i.e., works like  $\text{HyperCut}_{\alpha 0}$ ,
    - ▶ sets  $r_{s+1}$  to 1, i.e., works like  $\text{HyperCut}_{\alpha 1}$ .
- In term of the expected weight of the returned cut, this means

$$\mathbf{E}[\hat{w}_G(\alpha)] = \frac{1}{2}\mathbf{E}[\hat{w}_G(\alpha 0)] + \frac{1}{2}\mathbf{E}[\hat{w}_G(\alpha 1)],$$

Hence at least one of the expected values on the right-hand side is at least as large as  $\mathbf{E}[\hat{w}_G(\alpha)]$ .

## Derandomization of Algorithm HyperCut

### Derandomized Algorithm HyperCut (Conditional expectation)

Input: A hypergraph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

For  $s = 1, \dots, n$

If  $\mathbf{E}[\hat{w}_G(r_1 \dots r_{s-1} 0)] \geq \mathbf{E}[\hat{w}_G(r_1 \dots r_{s-1} 1)]$  (\*)  
then  $r_s = 0$  else  $r_s = 1$

Let  $V_0 = \{i : r_i = 0\}$ .  
Let  $V_1 = \{i : r_i = 1\}$ .

Output: The cut  $(V_0, V_1)$ .

For the weight  $w_G$  of the cut returned by Algorithm HyperCut on input  $G$  we have

$$\begin{aligned} (1 - \frac{2}{2^k})m = \mathbf{E}[\hat{w}_G(\lambda)] &\leq \mathbf{E}[\hat{w}_G(r_1)] \leq \dots \\ &\dots \leq \mathbf{E}[\hat{w}_G(r_1, \dots, r_{n-1})] \leq \mathbf{E}[\hat{w}_G(r_1, \dots, r_n)] = w_G. \end{aligned}$$

## Derandomization of Algorithm HyperCut

- ▶ Is the derandomized version of HyperCut efficient?  
How complex is it to evaluate Condition (\*)?
- ▶ Let  $\widehat{e}_i(\alpha)$  be the indicator variable for the event that edge  $e_i$  crosses the cut returned by Algorithm HyperCut $_\alpha$ .
- ▶ By linearity of expectation, the expected value  $\mathbf{E}[\widehat{w}_G(r_1 \dots r_{s-1}i)]$  is just the sum over the  $\mathbf{E}[\widehat{e}_i(r_1 \dots r_{s-1}i)]$ .  
The latter expected values are easy to compute.  
Let  $e_i$  contain  $k_s$  small nodes of size at most  $s$  and  $k_l$  large nodes of size strictly larger than  $s$ .  
If there are two small nodes that have already been assigned to different sides of the cut, the expected value is 1.  
Otherwise, all small nodes have been assigned to the same side and the expected value is equal to the probability  $1 - 1/2^{k_l}$  that at least one of the large nodes is assigned to the other side.

## Derandomization of Algorithm HyperCut

- ▶ Recall from the introduction, that random variables  $X_1, \dots, X_n$  are *k-wise independent* if every subset of  $k$  of these variables is mutually independent.
- ▶ The verification of Algorithm HyperCut relied solely on the fact that the bits  $r_1, \dots, r_n$  have been chosen according to the uniform distribution on  $\{0, 1\}$  and that they are *k-wise independent*.
- ▶ The construction of pairwise, i.e., 2-wise independent, random bits in connection with the derandomization of Algorithm Cut can be extended to an, albeit more involved, construction of *k-wise independent* random bits for arbitrarily large  $k$ .
- ▶ In what follows, we present another standard construction of *k-wise independent* random bits that uses algebraic methods.

## Derandomization of Algorithm HyperCut

### Proposition

Let  $p$  be any prime number and let the numbers  $\widehat{a}$  and  $\widehat{b}$  be chosen uniformly and independently from  $\{0, \dots, p-1\}$ .

Then the  $p$  numbers

$$\begin{array}{ll} \widehat{a} \cdot 0 + \widehat{b} & \text{mod } p, \\ \widehat{a} \cdot 1 + \widehat{b} & \text{mod } p, \\ \vdots & \\ \widehat{a} \cdot (p-1) + \widehat{b} & \text{mod } p \end{array}$$

are uniformly distributed in  $\{0, \dots, p-1\}$  and are pairwise independent.

## Derandomization of Algorithm HyperCut

### Proof.

- ▶ The random variable  $\widehat{a}i + \widehat{b}$  is uniformly distributed.  
If  $\widehat{a}i$  is already determined, by choosing  $\widehat{b}$  uniformly,  $\widehat{a}i + \widehat{b}$  assumes any value in  $\{0, \dots, p-1\}$  with probability  $\frac{1}{p}$ .
- ▶ Any random variables  $\widehat{a}i + \widehat{b}$  and  $\widehat{a}j + \widehat{b}$  where  $i \neq j$  are mutually independent.

Fix any numbers  $m_1, m_2$  in  $\{0, \dots, p-1\}$ . Then the system of equations

$$ai + b = m_1 \pmod{p}, \quad aj + b = m_2 \pmod{p}$$

has a unique solution  $(a_0, b_0)$  with  $a_0, b_0$  in  $\{0, \dots, p-1\}$ .

Hence

$$\begin{aligned} & \text{Prob}[\widehat{a}i + \widehat{b} \pmod{p} = m_1, \widehat{a}j + \widehat{b} \pmod{p} = m_2] \\ &= \text{Prob}[\widehat{a} = a_0, \widehat{b} = b_0] = \frac{1}{p^2}. \end{aligned}$$

□

## Derandomization of Algorithm HyperCut

### Algorithm $\text{Cut}_{\text{pi}}$ (Pairwise independence)

Input: A graph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).  
The least prime  $p \geq n$ .

Choose  $\hat{a}$  and  $\hat{b}$  uniformly and independently in  $\{0, \dots, p-1\}$ .

For  $i = 1, \dots, n$ , let  $r_i = (\hat{a}i + \hat{b}) \bmod p$ .

Let  $V_0 = \{i : r_i \text{ is even}\}$ .

Let  $V_1 = \{i : r_i \text{ is odd}\}$ .

Output: The cut  $(V_0, V_1)$ .

- ▶ Again, the expected weight of the cut returned by Algorithm  $\text{Cut}_{\text{pi}}$  is  $\frac{m}{2}$ .
- ▶ The analysis is basically the same as for Algorithm Cut. (We omit some minor technical details that relate to the fact that now the nodes are assigned to the two parts of the cut with probability  $\frac{p-1}{2p}$  and  $\frac{p+1}{2p}$ .)

## Derandomization of Algorithm HyperCut

### Derandomized Algorithm Cut (Pairwise independence)

Input: A graph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

Let  $p$  be the least prime where  $n \leq p$ .

For all pairs  $(a, b)$  in  $\{0, \dots, p-1\}$

For  $i = 1, \dots, n$ , let  $r_i = (ai + b) \bmod p$ .

Let  $V_0^{(a,b)} = \{i : r_i \text{ is even}\}$ .

Let  $V_1^{(a,b)} = \{i : r_i \text{ is odd}\}$ .

Output: A cut  $(V_0^{(a,b)}, V_1^{(a,b)})$  of maximum weight.

- ▶ The algorithm returns a cut of weight at least  $\frac{m}{2}$ .
- ▶ By Bertrand's postulate the least prime  $p > n$  is not larger than  $2n$  and checking any number  $n, \dots, 2n$  for primality requires time polynomial in  $\log n$ .
- ▶ A number in the range between  $n$  and  $2n$  can be specified by at most  $1 + \lceil \log n \rceil$  bits, hence in the algorithm at most  $16n^2$  pairs  $(a, b)$  have to be considered.

## Derandomization of Algorithm HyperCut

### Proposition

Let  $p$  be any prime number and let the numbers  $\hat{a}_0, \dots, \hat{a}_{k-1}$  be chosen uniformly and independently from  $\{0, \dots, p-1\}$ .

Let  $r_1, \dots, r_p$  be equal to the unique sequence of  $p$  numbers in  $\{0, \dots, p-1\}$  such that

$$\hat{a}_{k-1}0^{k-1} + \hat{a}_{k-2}0^{k-2} + \dots + \hat{a}_0 = r_1 \bmod p,$$

$$\hat{a}_{k-1}1^{k-1} + \hat{a}_{k-2}1^{k-2} + \dots + \hat{a}_0 = r_2 \bmod p,$$

$$\hat{a}_{k-1}2^{k-1} + \hat{a}_{k-2}2^{k-2} + \dots + \hat{a}_0 = r_3 \bmod p,$$

$\vdots$

$$\hat{a}_{k-1}(p-2)^{k-1} + \dots + \hat{a}_0 = r_{p-1} \bmod p,$$

$$\hat{a}_{k-1}(p-1)^{k-1} + \dots + \hat{a}_0 = r_p \bmod p.$$

Then the random numbers  $r_1, \dots, r_p$  are uniformly distributed in  $\{0, \dots, p-1\}$  and are  $k$ -wise independent.

## Derandomization of Algorithm HyperCut

### Proof.

Each  $r_i$  is uniformly distributed in  $\{0, \dots, p-1\}$  because assuming that the values of  $\hat{a}_1$  through  $\hat{a}_{k-1}$  have already been determined, the  $p$  equally probable choices for the value of  $\hat{a}_0$  will result in  $p$  pairwise distinct values for  $r_i$ .

In order to demonstrate that the  $r_i$  are  $k$ -wise independent, it suffices to show that for pairwise distinct indices  $i_1, \dots, i_k$  and any numbers  $b_1, \dots, b_k$  where the  $i_j$  and the  $b_j$  are in  $\{1, \dots, p\}$ , we always have

$$\text{Prob}[r_{i_1} = b_1, \dots, r_{i_k} = b_k] = \frac{1}{p^k}.$$

## Derandomization of Algorithm HyperCut

Proof (continued).

By definition of the  $r_j$ , this amounts to show that the system of equations

$$\begin{pmatrix} i_1^{k-1} & i_1^{k-2} & \dots & i_1^0 \\ i_2^{k-1} & i_2^{k-2} & \dots & i_2^0 \\ \vdots & \vdots & \ddots & \vdots \\ i_{k-1}^{k-1} & i_{k-1}^{k-2} & \dots & i_{k-1}^0 \\ i_k^{k-1} & i_k^{k-2} & \dots & i_k^0 \end{pmatrix} \begin{pmatrix} \hat{a}_0 \\ \vdots \\ \hat{a}_{k-1} \\ \hat{a}_k \end{pmatrix} = (b_1, \dots, b_k)$$

has a unique solution (where now the numbers appearing in the system of equations are identified in the obvious way with elements of the field with  $p$  elements).

Observe that the matrix  $M$  in this system of equations is a Vandermonde matrix.

## Derandomization of Algorithm HyperCut

Proof (continued).

For the Vandermonde matrix  $M$ , its determinant is given by

$$\prod_{1 \leq j < j' \leq k} (i_{j'} - i_j).$$

Since the  $i_j$  are pairwise distinct, the determinant differs from 0, hence the matrix is invertible, and one obtains for the  $\hat{a}_j$  the unique solution  $M^{-1}(b_1, \dots, b_k)$ .  $\square$

## Derandomization of Algorithm HyperCut

Algorithm HyperCut ( $k$ -wise independence)

Input: A hypergraph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

Let  $p$  be the least prime where  $n \leq p$ .

Choose  $a_0, \dots, a_{k-1}$  in  $\{0, \dots, p-1\}$  uniformly and independently.

For  $i = 1, \dots, n$ , let

$$r_i = a_{k-1}i^{k-1} + a_{k-2}i^{k-2} + \dots + a_0 \pmod{p}.$$

Let  $V_0 = \{i : r_i \text{ is even}\}$ .

Let  $V_1 = \{i : r_i \text{ is odd}\}$ .

Output: The cut  $(V_0, V_1)$ .

The algorithm returns a cut of expected weight of approximately  $(1 - \frac{2}{2^k})m$ .

## Derandomization of Algorithm HyperCut

Derandomized Algorithm HyperCut ( $k$ -wise independence)

Input: A hypergraph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).

Let  $p$  be the least prime where  $n \leq p$ .

For all  $k$ -tuples  $\vec{a} = (a_0, \dots, a_{k-1})$  over  $\{0, \dots, p-1\}$

For  $i = 1, \dots, n$ , let

$$r_i = a_{k-1}i^{k-1} + a_{k-2}i^{k-2} + \dots + a_0 \pmod{p}.$$

Let  $V_0^{\vec{a}} = \{i : r_i \text{ is even}\}$ .

Let  $V_1^{\vec{a}} = \{i : r_i \text{ is odd}\}$ .

Output: A cut  $(V_0^{\vec{a}}, V_1^{\vec{a}})$  of maximum weight.

The algorithm returns a cut of weight that is approximately at least  $(1 - \frac{2}{2^k})m$ .

## Independent sets in hypergraphs

- ▶ A *hypergraph* is a pair  $(V, E)$  where  $V$  is the set of nodes and  $E$  is a set of subsets of  $V$ . The subsets in  $E$  are called *hyperedges*.

A hypergraph is *3-regular* if all its hyperedges contain exactly 3 nodes.

### Definition

In a hypergraph  $G = (V, E)$ , a subset of  $V$  is called *independent* if it does not contain any edge in  $E$ .

### Theorem

A 3-regular hypergraph with  $n$  nodes and  $m \geq n/3$  hyperedges has an independent set  $U$  where

$$|U| \geq \frac{n\sqrt{n}}{3\sqrt{m}}.$$

## Independent sets in hypergraphs

- ▶ The randomized Algorithm IndependentSet computes an independent set  $U$  of expected size of at least  $|U| \geq \frac{n\sqrt{n}}{3\sqrt{m}}$ .

### Algorithm IndependentSet

Input: A hypergraph  $G = (V, E)$  ( $V = \{1, \dots, n\}$ ).  
A parameter  $p$  with  $0 \leq p \leq 1$ .

Choose random bits  $r_1, \dots, r_n$  by tosses of a biased coin  
such that  $\text{Prob}[r_i = 1] = p$ .

Let  $T = \{i : r_i = 1\}$ .

Let  $Y = \{\min e : e \in E \text{ and } e \subseteq T\}$

$U = T \setminus Y$ .

Output: The set  $U$ .

## Independent sets in hypergraphs

- ▶ The set  $T$  is a candidate for an independent set.
- ▶ The set  $Y$  contains the least node from each hyperedge contained in  $T$ .
- ▶ The set  $U$  is independent.
- ▶  $Y$  is a subset of  $T$ , hence  $|U| = |T| - |Y|$ .

By linearity of expectation, we have

$$\mathbf{E}[|U|] = \mathbf{E}[|T|] - \mathbf{E}[|Y|]$$

- ▶ The expected size of  $T$  is  $\mathbf{E}[|T|] = np$ .

## Independent sets in hypergraphs

- ▶ Let  $D = \{e \in E : e \subseteq T\}$ .  
Then  $|Y| \leq |D|$ , hence  $\mathbf{E}[|Y|] \leq \mathbf{E}[|D|]$ .
- ▶ For each  $e$  in  $E$ , let  $\hat{e}$  be the indicator variable for the event that  $e$  is contained in  $T$ .  
The size of  $D$  is just the sum over the random variables  $\hat{e}$ .  
By linearity of expectation, the expected size of  $D$  is the sum of the expected values  $\mathbf{E}[\hat{e}]$ .  
For any hyperedge  $e$  in  $E$ , by construction

$$\begin{aligned}\mathbf{E}[\hat{e}] &= \text{Prob}[e \subseteq T] \cdot 1 + \text{Prob}[e \not\subseteq T] \cdot 0 \\ &= \text{Prob}[e \subseteq T] = p^3.\end{aligned}$$

$$\mathbf{E}[|D|] = m \cdot p^3.$$

- ▶ In summary, we have

$$\mathbf{E}[|U|] = \mathbf{E}[|T|] - \mathbf{E}[|Y|] \geq \mathbf{E}[|T|] - \mathbf{E}[|D|] = n \cdot p - m \cdot p^3.$$

## Independent sets in hypergraphs

- ▶ We have seen

$$\mathbf{E}[|U|] = \mathbf{E}[|T|] - \mathbf{E}[|Y|] \geq n \cdot p - m \cdot p^3.$$

- ▶ For which  $p \in [0, 1]$  is the term  $n \cdot p - m \cdot p^3$  maximum?  
Assuming  $n \leq 3m$ , the term is maximum for

$$p = \sqrt{\frac{n}{3m}}.$$

Consider the function  $p \mapsto n \cdot p - m \cdot p^3$ .

Its first derivative  $n - 3mp^2$  is 0 for this value of  $p$ , while its second derivative is negative. For  $p = 0$ , the function is 0.

For  $p = 1$ , the function evaluates to  $n - m < 0$ .

- ▶ For  $p = \sqrt{\frac{n}{3m}}$ , we then obtain

$$\begin{aligned} \mathbf{E}[|U|] &\geq n \cdot p - m \cdot p^3 \\ &= \left( \frac{1}{\sqrt{3}} - \frac{1}{3\sqrt{3}} \right) \frac{n\sqrt{n}}{\sqrt{m}} \geq \frac{1}{3} \frac{n\sqrt{n}}{\sqrt{m}} \end{aligned}$$

## Independent sets in hypergraphs

- ▶ By the usual argument, the expected value of  $|U|$  must be attained for some choice of the coin tosses in Algorithm IndependentSet, hence any 3-regular graph with  $n \leq 3m$  contains an independent set of the required size.
- ▶ Algorithm IndependentSet can be derandomized
  - ▶ by the method of conditional expectations, where it is arranged in the derandomized algorithm that the potential function  $\mathbf{E}[|T|] - \mathbf{E}[|D|]$  does not decrease, hence is always at least as large as the initial value  $\frac{1}{3} \frac{n\sqrt{n}}{\sqrt{m}}$ .  
How difficult is it to determine the next random bit?
  - ▶ by the method of small sample spaces, using 3-wise independent random variables that attain the value 1 with probability  $p$ , then applying trivial derandomization.  
How can such 3-wise independent random variables be obtained (where the probability  $p$  for the value 1 may only be approximated)?