

## 1 Derandomizing MaxCut algorithm using Pairwise Independent Generator

In the last class, we defined  $(m, t, l)$  pairwise independent generator as a function

$$f : \Sigma^m \rightarrow \Sigma^{tl} \text{ such that}$$

$$\forall i \neq j, \forall \alpha, \beta \in \Sigma^t$$

$$\Pr_{x \in \Sigma^m} [f_i(x) = \alpha \wedge f_j(x) = \beta] = \frac{1}{2^{2t}}$$

We also proved the following theorem.

**Theorem:**  $\forall t, \forall l, \forall m$  with  $2^m > l$  and  $m \geq t$ , there is  $(2m, t, l)$  pairwise independent generator.

Now, we show the following algorithm to derandomize *Max - Cut* algorithm.

1. Input  $G = (V, E)$  where  $|V| = n$
2. Let  $f$  be a  $(2 \log n, 1, n)$  pairwise independent generator.
3. Input  $G = (V, E)$  where  $|V| = n$
4. Randomly pick a  $2 \log n$  bit string (call  $x$ )
5.  $A \leftarrow \phi, B \leftarrow \phi$
6. **for**  $i \leftarrow 1$  to  $n$ 
  - (a) **if**  $f_i(x) = 1$   
 $A \leftarrow A \cup \{v_i\}$
  - (b) **else**  $B \leftarrow B \cup \{v_i\}$
7. Output  $A$  and  $B$

We will now see the expected cut size of this algorithm. Let  $X$  be a random variable that represents the size of the cut. If an edge  $e$  belongs to the cut, let  $X_e$  be a random variable defined as

$$X_e = 1 \text{ if edge } e \in \text{cut}$$

$$= 0 \text{ else}$$

Let  $v_i$  and  $v_j$  be end points of  $e$ .

$$e \in \text{cut} \implies \text{either}$$

$$f_i(x) = 0 \wedge f_j(x) = 1$$

or

$$f_i(x) = 1 \wedge f_j(x) = 0$$

Let  $E_{01}^{ij}$  denote the event  $f_i(x) = 0 \wedge f_j(x) = 1$  and  $E_{10}^{ij}$  denote the event  $f_i(x) = 1 \wedge f_j(x) = 0$ . Since  $f$  is a  $(2 \log n, 1, n)$  pair-wise independent generator, for every  $i$  and  $j$

$$\Pr_{x \in \Sigma^{2 \log n}} [E_{01}^{ij}] = \Pr_{x \in \Sigma^{2 \log n}} [E_{10}^{ij}] = \frac{1}{4}$$

Since events  $E_{01}^{ij}$  and  $E_{10}^{ij}$  are disjoint

$$\Pr_{x \in \Sigma^{2 \log n}} [E_{10}^{ij} \cup E_{01}^{ij}] = 1/2.$$

Hence  $\Pr [e \in \text{cut}] = \frac{1}{2}$ , i.e.  $E[X_e] = \frac{1}{2}$ . Since  $X = \sum_{e \in E} X_e$ ,  $E[X] = \frac{m}{2}$

Note that each  $X_e$  depends only on two vertices  $v_i$  and  $v_j$  only. So, as long as any two vertices  $v_i$  and  $v_j$  are independently placed in  $A$  or  $B$ , we have the expected cut size as  $\frac{m}{2}$  i.e. as long as we have  $n$  bits in which any two of them are independent, this algorithm works. So, all we need is pairwise independence and not complete independence among the bits. This is exactly what pair-wise independent generator is doing.

Now, we know that

$$\Pr[X \geq E[X]] > 0$$

So, if we cycle through all possible random seeds (each of length  $2 \log n$ ), at least one of them will give us a cut size  $\geq \frac{m}{2}$ . Hence, our new deterministic algorithm will be as follows

1.  $Best-A \leftarrow \phi$ ,  $Best-B \leftarrow \phi$
2.  $Best-Cut-Size \leftarrow 0$
3.  $A \leftarrow \phi$ ,  $B \leftarrow \phi$
4.  $Cut-Size \leftarrow 0$
5. Pick a  $(2 \log n, 1, n)$  pairwise independent generator (call  $f$ )
6. Input  $G = (V, E)$  where  $|V| = n$
7. **for each**  $2 \log n$  bit string (call  $x$ )
  - (a) **for**  $i \leftarrow 1$  to  $n$ 
    - i. **if**  $f_i(x) = 1$   
 $A \leftarrow A \cup \{v_i\}$
    - ii. **else**  $B \leftarrow B \cup \{v_i\}$
  - (b) Compute  $Cut-Size$  for  $A$  and  $B$
  - (c) **if**  $Cut-Size > Best-Cut-Size$ 
    - i.  $Best-A \leftarrow A$
    - ii.  $Best-B \leftarrow B$

iii. *Best-Cut-Size*  $\leftarrow$  *Cut-Size*

8. Output *Best-A* and *Best-B*

**Run time complexity:**

The run time complexity of this algorithm is  $O(2^{2\log n}n) = O(n^3)$ .

Another advantage of this algorithm is that it allows parallelism. The block for each of  $2\log n$  strings can be run in parallel.

## 2 Deterministic Amplification

Let  $L \in RP$ .

By definition, we know that

$\exists$  an algorithm  $A$  such that

$$\begin{aligned} x \in L &\implies \Pr[A(x) \text{ accepts}] \geq \frac{1}{2} \\ x \notin L &\implies \Pr[A(x) \text{ accepts}] = 0 \end{aligned}$$

Therefore, error probability  $\leq \frac{1}{2}$

Let  $A$  make use of  $r(n)$  random bits. To reduce the error probability to  $\frac{1}{4}$ , we need  $2 \cdot r(n)$  random bits. This is by running  $A$  twice each time with separate  $r(n)$  random bits and then accepting if at least one of the runs of  $A$  accepts.

$$\begin{aligned} \text{i.e. to reduce error probability} &\leq \frac{1}{4}, \text{ we need } 2 \cdot r(n) \text{ random bits} \\ &\leq \frac{1}{8}, \text{ we need } 3 \cdot r(n) \text{ random bits} \\ &\vdots \\ &\leq \frac{1}{2^n}, \text{ we need } n \cdot r(n) \text{ random bits} \end{aligned}$$

Is it possible to reduce error probability using lesser random bits? For example, can we achieve error probability of  $1/16$  using less than  $4r(n)$  random bits? We will now show that this is possible.

We will make use of  $(2 \cdot r(n), r(n), l)$  pairwise independent generator. Consider the following algorithm to reduce the error probability  $\leq \frac{1}{n}$ .

- 1 Let  $f$  be a  $(2 \cdot r(n), r(n), l)$  pairwise independent generator
- 2 Input  $x$  where  $|x| = n$
- 3 Randomly pick a  $2 \cdot r(n)$  bit string (call  $r$ )
- 4 Compute  $f(r)$  to obtain  $l$  blocks, each of length  $r(n)$
- 5 Run  $A$  using each block as random seed
- 6 Accept iff at least one block causes  $A$  to accept

We will now see that we can apply Chebyshev's inequality to prove the bounds of this algorithm. To apply Chebyshev's inequality, all we need to know is the variance. We need not have completely independent random variables.

Let us define a random variable

$$\begin{aligned} X_i &= 1 \text{ if } A \text{ gives correct answer when it uses } i^{\text{th}} \text{ block as random seed} \\ &= 0 \text{ else} \end{aligned}$$

Since  $f$  is a pair-wise independent generator,  $X_1, X_2, X_3 \dots X_l$  are pairwise independent random variables.

$$\text{Let } X = \sum_{i=1}^l X_i$$

$$\text{We know that } E[X_i] \geq \frac{1}{2}$$

and, since  $X_i$ 's are 0-1 random variables,  $Var(X_i) \leq 1/4$ .

Let  $Z$  be another random variable defined as

$$Z = \frac{X}{l}$$

Since  $E[X] \geq \frac{l}{2}$ ,  $E[Z] \geq \frac{1}{2}$ . Since  $X = \sum_i X_i$  and  $X_i$ 's are pairwise independent,  $Var(X) = \sum_i Var(X_i)$ . Thus

$$\begin{aligned} Var[Z] &= Var\left[\frac{X}{l}\right] \\ &= \frac{Var[X_1] + Var[X_2] + \dots + Var[X_l]}{l^2} \\ &\leq \frac{\frac{1}{4} + \frac{1}{4} + \dots + \frac{1}{4}}{l^2} \\ &\leq \frac{1}{4l} \end{aligned}$$

Now we are ready to calculate the probability that the algorithm is wrong. Observe that if  $x \notin L$ ,  $A$  never goes wrong. Thus we are interested in probability that the algorithm is wrong when  $x \in L$ . This happens when each of  $X_i$ 's are 0.

Now

$$\begin{aligned} \Pr[\text{Algorithm is wrong}] &= \Pr[Z = 0] \\ &\leq \Pr\left[|Z - E[Z]| \geq \frac{1}{2}\right] \end{aligned}$$

By Chebyshev's inequality, we have

$$\Pr[|X - E[X]| \geq \epsilon] \leq \frac{Var[X]}{\epsilon^2}$$

Therefore,

$$\begin{aligned} \Pr[\text{Algorithm is wrong}] &\leq \frac{Var[Z]}{\left(\frac{1}{2}\right)^2} \\ &\leq 4 \cdot \frac{1}{4l} \\ &\leq \frac{1}{l} \end{aligned}$$

Therefore, we can reduce the error probability of a randomized algorithm to  $\leq \frac{1}{l}$  using only  $2 \cdot r(n)$  random bits with running time  $\text{poly}(lr(n))$ . In particular, the error probability can be made  $1/n^k$  in polynomial time. However, note that if we wish to make error probability  $1/2^n$  then this method takes exponential time.