**INF561: Using randomness in algorithms** **Winter 2013**

# Cours 3 — 23rd January

*Enseignant : David Xiao* *Rédacteur : Menglong Fu and Nathan Grosshans*

## 3.1 Introduction

Computational efficiency is all about reducing the amount of resources that are used by an algorithm. These resources can be :

- Time and space, which are the standard resources we focus on when designing algorithms.
- Randomness, whose use we may also want to reduce, as there is a cost to generate random bits.

But how do we actually generate random bits ? In fact, classically, the generation of bits isn't really random, that's why those generators are called "pseudo-random" (a physical measure like time, temperature or quantum information is used). There are two assumptions that are made about randomness in algorithms :

- we have access to a polynomial amount of randomness (if algorithms are in polynomial time);
- we have uniform and random bits.

These are the two main topics in pseudo-randomness, but we will mostly focus on the first point.

## 3.2 Reducing the amount of randomness used

### 3.2.1 Randomness inefficient error reduction

**Théorème 3.1.** *Let $A$ be an algorithm deciding $L$ with error less than or equal to $\frac{1}{3}$, such that for all x, $\Pr_{r\, uniform}[A(x,r) \neq L(x)] \leq \frac{1}{3}$ and $A$ runs in time $\mathcal{O}(T)$. Then, there exists $A'$ deciding L running in time $\mathcal{O}(kT)$ with error less than or equal to $2^{-k}$.*

**Preuve:** The idea is to run $A$ $\mathcal{O}(k)$ times, and then output the majority result.

**Théorème 3.2 (Chernoff-Hoeffding bound).** *Let $X_1, \ldots, X_l$ be independent and identically distributed (i.i.d.) random variables in $\{0,1\}$. Let $\mu = \mathbb{E}[X_i]$, $\varepsilon \in \{0,1\}$. Then, $\Pr[\frac{1}{l}\sum_{i=1}^{l} X_i < \mu - \varepsilon] \leq e^{-2\varepsilon^2 l}$. This inequality is symmetric.*

We will prove Theorem 3.1 using the Chernoff-Hoeffding bound. Let $X_1, \ldots, X_l$ be such that

$$X_i = \begin{cases} 1 & \text{if i}^{\text{th}} \text{ execution correct} \\ 0 & \text{else} \end{cases}$$

Then $\Pr[A'(x, r') \neq L(x)] = \Pr[\frac{1}{l}\sum_{i=1}^{l} X_i < \frac{1}{2}] \leq e^{-2\frac{1}{6^2}l}$. So, if $l = \mathcal{O}(k)$, we have an error less than or equal to $2^{-k}$.

□

The great disadvantage of this error reduction method is that we have a linear increase in the amount of randomness used.

### 3.2.2 Efficient deterministic error reduction

**Théorème 3.3.** *Suppose $A$ decides $L$ in time $T$, with error less than or equal to $\frac{1}{3}$. Then there exists $c > 0$, such that for all $\delta > 0$, there exists $A'$ deciding L in time $\mathcal{O}((\frac{T}{\delta})^c)$ (n being a function of the input size) and with error less than or equal to $\delta$. Furthermore, $A'$ uses no more randomness than $A$.*

To prove this theorem, we have to introduce the notion of expander graphs.

#### Expander graphs: intuitive definition

The intuition is that expander graphs are highly connected and "random-looking", but also sparse (the number of edges $|E|$ is small relative to the number of vertices $|V|$).

**Examples:** Examples of non-expanders are graphs that are made of different parts that are well connected but linked by very few edges (so that the graph in general is not well connected), or complete graphs which are not sparse (the number of edges in it is $|E| = \Omega(|V|^2)$).

Why does graph theory help ? Fix an algorithm $A$ with input $x$. Say $A$ uses $m$ bits of randomness on $x$. Taking $k$ uniform random $m$-bits strings is equivalent to taking a random walk of length $k$ on $K_{2^m}$ (the complete graph on $2^m$ vertices, where each vertex is labelled by an $m$-bits string). The randomness cost of the walk is then $m + (k-1) \cdot m$ ($m = \log_2(\text{degree of } K_{2^m})$). The idea is that reducing the degree is equivalent to reducing the cost (in terms of randomness), but we must make sure that when taking a random walk in the resulting graph, the number of times we hit bad vertices remains roughly the same as in the original graph (a bad vertex is one where the algorithm would make a mistake if it used it as a random string).

**Définition 3.4.** *A graph $G = (V, E)$ is d-regular if every $v \in V$ has exactly $d$ neighbours (assuming that $G$ is undirected and can contain self-loops and multiple edges).*

**Définition 3.5.** *Fix any $G = (V, E)$. For $v \in V$, $N(v) = \{u | (v, u) \in E\}$ is the neighbourhood of $v$. For $S \subseteq V$, $N(S) = \bigcup_{v \in S} N(v)$.*

**Définition 3.6.** *$G = (V, E)$ is a $(d, \alpha)$-vertex expander if :*

- *$G$ is $d$-regular;*
- *for all $S \subseteq V$, $0 < |S| < \frac{|V|}{2}$, $|N(S)| \geq (1 + \alpha) \cdot |S|$.*

*This means that the graph "expands" in terms of number of vertices.*

**Remark:** This definition is intuitively nice, but technically inconvenient.

**Expander graphs: formal definition and properties**

**Définition 3.7.** *Fix $G = (V, E)$, a $d$-regular graph. The normalized adjacency matrix of $G$ is a $|V| \times |V|$ matrix $M$ where $M_{uv} = \frac{\text{number of edges between u and v in G}}{d}$.*

**Remarks:**

1. For all $u \in V$, $\sum_{v \in V} M_{uv} = 1$, and the same for $u$ and $v$ reverted (this is called a doubly stochastic matrix).
2. $M$ is symmetric and real, therefore eigenvalues are real and the associated eigenvectors are orthonormal.
3. As a consequence of the two previous points, the eigenvalues belong to $[-1, 1]$.

**Proposition 3.8.** *For all $d$-regular graph $G$, $1$ is an eigenvalue of $G$.*

**Preuve:** $\overrightarrow{1}$ is eigenvector of $G$ with eigenvalue 1. □

**Proposition 3.9.** *If $G$ is disconnected, then $1$ has multiplicity greater than or equal to $2$ as eigenvalue of $G$.*

**Preuve:** If $G$ is disconnected, it has at least 2 different connected components. Let $S$ and $T$ be those components. Then,

$$\overrightarrow{1_S} = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ and } \overrightarrow{1_T} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

are two different orthogonal eigenvectors of $G$ for 1 as eigenvalue. □

**Proposition 3.10.** *If $1$ has multiplicity greater than or equal to $2$, then $G$ is disconnected.*

**Proposition 3.11.** *If $G$ is bipartite, then $-1$ is eigenvalue of $G$.*

**Preuve:** As $G$ is $d$-regular, we can organise its adjacency matrix such that

$$\left(\begin{array}{c|c} 0 & \cdots \\ \hline \cdots & 0 \end{array}\right) \cdot \begin{pmatrix} -1 \\ \vdots \\ -1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}$$

□

**Proposition 3.12.** *If $-1$ is eigenvalue of $G$, then $G$ is bipartite.*

**Définition 3.13.** $\lambda_2(G) = \max\limits_{\substack{\lambda \neq 1 \\ \lambda \, eigenvalue \, of \, G}} (\lambda)$.

**Définition 3.14.** *$G$ is a $(d, \lambda)$-expander if $\lambda_2(G) \leq \lambda$ and $G$ is $d$-regular. We want $\lambda < 1$ as small as possible.*

**Example:** $\lambda_2(K_n) = 0$, the adjacency matrix of the complete graph on $n$ vertices being

$$\frac{1}{n} \cdot \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$$

**Théorème 3.15.** *$G$ is a $(d, \alpha)$-vertex expander for some $\alpha > 0$ if and only if $G$ is a $(d, \lambda)$-expander for some $\lambda < 1$. Correspondence between $\alpha$ and $\lambda$ is independent of $|G|$.*

**Définition 3.16.** *$\mathcal{G} = \{G_1, G_2, \ldots, G_i, \ldots\}$ is a family of $(d, \lambda)$-expanders if each $G \in \mathcal{G}$ is a $(d, \lambda)$-expander, $\lambda < 1$ and $d$ being constants independent of $|G|$.*

### Efficient construction of expanders and proof of the theorem

Do expanders exist ? Yes, because random $d$-regular graphs are expanders. But what about efficient deterministic constructions ?

**Définition 3.17.** *$\mathcal{G} = \{G_1, G_2, \ldots\}$ is computable in time $t$ ($t : \mathbb{N} \to \mathbb{N}$) if there exists an algorithm $A$ running in time $t$ such that $A(i, u) = \{v | (u, v)$ neighbours in $G_i\}$ (which is a list of size $d$). If $|G_i| > |G_j|$ for all $i > j$, then the input size is in $\mathcal{O}(\log_2(|G_i|))$.*

**Lemme 3.18 (Expander Mixing Lemma).** *If $G$ is a $(d, \lambda)$-expander, then $\forall S, T \subseteq V$*

$$\left| \, |E(S,T)| - \frac{d\,|S|\,|T|}{|V|} \, \right| \leqslant d\lambda\sqrt{|S|\,|T|}$$

*where $E(S,T) \overset{d}{=} \{(u,v) \in E, u \in S, v \in T\}$. In fact $\mathop{\mathbb{E}}\limits_{random\, d-regular\, graph} [E(S,T)] = \dfrac{d\,|S|\,|T|}{n}$.*

**Preuve:** Let $M$ be the adjacency matrix of $G$. Let $\overrightarrow{1_S} = (0, \cdots, 0, \underbrace{1, \cdots, 1}_{S}, 0, \cdots, 0)^T$

First claim, $|E(S,T)| = (\overrightarrow{1_S}|M\overrightarrow{1_T}) \cdot d$, since

$$(\overrightarrow{1_S}|M\overrightarrow{1_T})d = \sum_{u \in S} (\text{\# edges between u and T}) = |E(S,T)|$$

Let $\vec{x_1}, \cdots, \vec{x_n}$ be orthonormal eigenvectors of $M$, with eigenvalues s.t$|\lambda_1| = 1 \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n|$
Let

$$\overrightarrow{1_S} = \sum_{i=1}^{n} \alpha_i \vec{x_i}$$
$$\overrightarrow{1_T} = \sum_{i=1}^{n} \beta_i \vec{x_i}$$

then

$$\begin{aligned}
d(\overrightarrow{1_S}|M\overrightarrow{1_T}) &= d \sum_{i,j=1}^{n} (\alpha_i \vec{x_i}|M\beta_j\vec{x_j}) \\
&= d \sum_{i,j=1}^{n} \alpha_i\beta_j\lambda_j\,(\vec{x_i}|\vec{x_j}) \\
&= d \sum_{i}^{n} \lambda_i\alpha_i\beta_i \\
&= d\left(\lambda_1\alpha_1\beta_1 + \sum_{i=2}^{n} \lambda_i\alpha_i\beta_i\right) \\
&= d\frac{|S|\,|T|}{n} + d\sum_{i=2}^{n} \lambda_i\alpha_i\beta_i
\end{aligned}$$

therefore,

$$\begin{aligned}
\Big|\,|E(S,T)| - \frac{d\,|S|\,|T|}{|V|}\,\Big| &= \left| d \sum_{i=2}^{n} \lambda_i \alpha_i \beta_i \right| \\
&\leqslant d \sum_{i=2}^{n} |\lambda_i|\,|\alpha_i|\,|\beta_i| \\
&\leqslant d\lambda \sum_{i=2}^{n} |\alpha_i|\,|\beta_i| \\
&\leqslant d\lambda \left( \sqrt{\sum_{i=2}^{n} |\alpha_i|^2} \right) \left( \sqrt{\sum_{i=2}^{n} |\beta_i|^2} \right) && \text{(Cauchy-Schwartz inequality)} \\
&\leqslant d\lambda \sqrt{\sum_{i=1}^{n} |\alpha_i|^2} \sqrt{\sum_{i=1}^{n} |\beta_i|^2} \\
&\leqslant d\lambda \sqrt{|S|\,|T|} && \text{as } \vec{x_1}, \cdots, \vec{x_n} \text{ are orthonormal}
\end{aligned}$$

□

**Théorème 3.19.** *If A decides L in time T, then* $\exists\ c$, $\forall \delta$, $\exists\ A'$ *deciding L in time* $\mathcal{O}\left(\left(\frac{T}{\delta}\right)^c\right)$ *with error* $\leqslant \delta$, $A'$ *using no more randomness than A.*

**Preuve:** Assume $\mathcal{G} = \{G_1, \cdots, G_n, \cdots\}$ is a family of $(d, \lambda)$-expanders, which is efficiently computable. Assume $|G_i| = 2^i$.

**Définition 3.20.** *Given G, let* $G^l = (V, E')$, *where* {# *edges between u,v in* $E'$}= {# *path of length l between u,v in E*}.

**Proposition 3.21.** $M\left(G^l\right) = M\left(G\right)^l \Rightarrow \lambda_2(G^l) = \left(\lambda_2(G)\right)^l$ *and* $deg(G^l) = deg(G)^l$.

If $\mathcal{G}$ is computable in time t, then $\mathcal{G}^l = \{G_1^l, G_2^l, \cdots\}$ is computable in time $t' \leqslant 2d^l t$. So if $l$ is not too large, $\mathcal{G}^l$ remains efficient.

$A'$ : on input $x$, suppose A uses $m$ (uniform) random bits.

- Set $l = \frac{\log\left(\frac{12}{\delta}\right)}{2\log\left(\frac{1}{\lambda}\right)}$.
- Pick $u \in G_m^l$ at random.
- Output majority of $\{A\left(x, v\right) : (u, v) \in G_m^l\}$.

Clearly, $A'$ uses $m$ bits of randomness. The running time of $A'$:
since $d^l = d^{\frac{\log\left(\frac{12}{\delta}\right)}{2\log\left(\frac{1}{\lambda}\right)}} = 2^{\frac{\log(d)\cdot\log\left(\frac{12}{\delta}\right)}{2log\left(\frac{1}{\lambda}\right)}} = \mathcal{O}\left(\frac{1}{\delta^c}\right)$
then,

$$\underbrace{2d^l t\left(m\right)}_{time\,to\,compute\,neighbours} + \overbrace{d^l T}^{time\,to\,run\,A\,on\,neighbours} \leqslant \mathcal{O}\left(\frac{T+t\left(m\right)}{\delta^c}\right) \leqslant \mathcal{O}\left(\frac{T+T^{c'}}{\delta^c}\right)$$

Fix $x$, let

$$Bad = \{v \in G_m^l : A(x,v) \neq L\left(x\right)\}$$
$$Bad' = \{v \in G_m^l : A'(x,v) \neq L\left(x\right)\}$$

We know $|Bad| \leqslant \frac{1}{3}\cdot 2^m$. By the Expander Mixing Lemma,

$$E\left(Bad, Bad'\right) - \frac{d^l\left|Bad\right|\left|Bad'\right|}{2^m} \leqslant d^l\lambda^l\sqrt{\left|Bad\right|\left|Bad'\right|} \tag{3.1}$$

$$\left|E\left(Bad, Bad'\right)\right| = \sum_{v\in Bad'}\left|E\left(Bad,\{v\}\right)\right| \geqslant \sum_{v\in Bad'}\frac{d^l}{2} = \left|Bad'\right|\frac{d^l}{2} \tag{3.2}$$

as $v \in Bad'$, it means that at least half of its $d^l$ neighbours in $G_m^l$ make the algorithm $A$ output a wrong answer when they are used as random strings, that is to say that at least half of the neighbours of $v$ belong to $Bad$, and therefore $\left|E\left(Bad,\{v\}\right)\right| \geqslant \frac{d^l}{2}$.

$$\begin{aligned}
(3.1)\,(3.2) \Rightarrow \frac{\left|Bad'\right| d^l}{2} - \frac{d^l\left|Bad\right|\left|Bad'\right|}{2^m} &\leqslant d^l\lambda^l\sqrt{\left|Bad\right|\left|Bad'\right|} \\
\sqrt{\left|Bad'\right|}\left(\frac{1}{2}-\frac{\left|Bad\right|}{2^m}\right) &\leqslant \lambda^l\sqrt{\left|Bad\right|} \\
\left|Bad'\right| &\leqslant \frac{\lambda^{2l}\left|Bad\right|}{\left(\frac{1}{2}-\frac{\left|Bad\right|}{2^m}\right)^2} \\
&\leqslant \frac{\lambda^{2l}\frac{1}{3}2^m}{\left(\frac{1}{6}\right)^2} \\
&\leqslant 12\lambda^{2l}2^m \\
&\leqslant \delta 2^m
\end{aligned}$$

□

**Remark:** It is possible to efficiently reduce error in random algorithm without using additional randomness. Disadvantages in this approach:

- more time used;
- must constructs good expanders;
- $\delta$ must be at least $\frac{1}{polynomial(n)}$.