# Jcalm 42-AM/2015

## Counting : Algorithms and Complexity

Stéphane Perennes (CNRS)

COATI

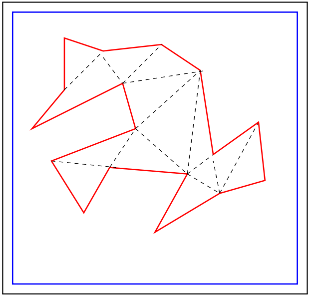
11 mars 2015

# Outline

1. Introduction
   - Why do we need algorithms for that
2. Monte Carlo Method
   - Naive empiral integration is easy . . . deeper than it seems
3. Formalism, $\sharp P$
   - Approx + Randomized + couting + generating
   - The $\sharp P$ Class
   - A Variant of Cook Theorem
   - Toda Theorem Non proof ??
4. Counting Solutions of *Easy Problems* ?
   - Case of trackatable problems
   - Matchings and Permanents
   - Valiant Result about counting Matchings
   - Valiant's reduction :Simulating counting 3Covers
5. Approximate Counting of Matchings (Approx the permanent)
   - The Markov Chain Approach
   - A rapid mixing chain for matchings
   - Chains for Trees
   - Polytope Sampling, Convex Volume integration

# Counting, Generating why ?

- Mesure probabilities, volumes : Area of a shape.
  - Math solution : Triangulate, divide into simplexes decompose into shapes for which we have a formula.
  - High dimension ? $\Rightarrow$ Even a polytope do have exponential number of vertices, non practical.
  - Can we get an efficient algorithm ?



- **Example:** Tree Polytope of $G = (V, E)$

$$w : E \Longrightarrow \mathbf{R}^+$$
$$\forall V' \subset V, w([V', V']) \leq |V'| - 1$$
$$w([V, V]) = |V| - 1$$

Polytope Vertices $=$ The spanning trees

(How $\backslash$ Can) we deal with that ?

# Counting to Evaluate probabilities

**(Discretes|finites Probabilities) $\Longleftrightarrow$ Counting**

## Question

- *Network* $N = (V, E)$, *edges i.i.d failures* $(p = \frac{1}{2})$.
  *Compute* $B(N) = Prob(N$ *gets disconnected* $)$ *?*

$$B(N) = Prob[\cup_{S \in V, S \notin \{\emptyset, V\}} [S, \overline{S}] \text{ fails } ] \sim \cup_{S \in V, S \notin \{\emptyset, V\}} 2^{-|[S, \overline{S}]|}$$

- Random $N$ with a fixed support $\rightarrow$ Probability of a property ?
- Erdũs Reynii : (trivial) Case $N = K_n \rightarrow$ Formulas, theorems ...
- Other Random distributions: degree sequences, Euclidian, planar, whatever ... $\rightarrow$ Again formulas.

# Generating and Sampling

## Typical Questions

- Generate a random tree of $G/$
- Generate a random simple path from $u$ to $v$.
- Generate a random failure scenario .
- Generate complex items for simulation or testing, but in a fair way.
- Make experiment on a random graph that looks like a typical case.

## What about Enumerative Combinatorics?

- Usually about a fixed object ($K_n$ or the Hypercube . . . )
- Fibbonacci, # partitions, "usual stuff" $\rightarrow$ 95 % of the time when we count we build bijections.
- Many (most?) inductive counting argument $\rightarrow$ inductive constructions.
- **Often :** Can count $\rightarrow$ Can Generate.

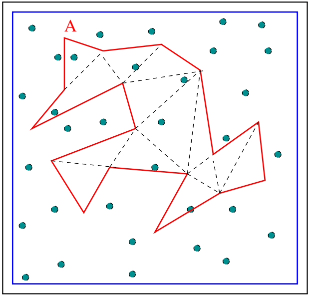
# Maths Sucks method (Naive Monte Carlo)

## Monte Carlo Uber Generator

- Throw random points in the square (we know how to sample it)
- point belongs to $A$ ? $\rightarrow$ returns it.
- Great generator, works for any **NP**problem

## Super Counting Algorithm

- Throw $n$ random points in the square, $a(n)$ points lie in $A$
- Return $Vol(P) = \frac{a(n)}{n} Vol(Square)$




**Works great if $\frac{Vol(A)}{Vol(Square)}$ is not too small (ie polynomial)**

# Math Reason, empirical mean often works . . .

## Chernoff Bound

$Z$ : Sum of $n$ i.i.d Bernouilli variables $X_1, X_2, \ldots X_n$ (random $(A, 1-A)$ biased coins) $\mu = E[Z] = n \cdot A$,

$$Pr[|Z(\omega) - \mu| \geq \delta\mu] \leq 2e^{\delta^2 \mu/3}$$

## Almost good and quite sure ?

- We want $\mu = \frac{1}{k}$
- and $2e^{-\delta^2\mu/3} \leq \frac{1}{2}$

So we want $\mu \geq 3 \ln 4\delta^2$ So

$n \sim$ Who care 1 barbu c'est un barbu et le deuxième momemnt suffit zzz

We need $n$ of order $vol(A)/Vol(Square)$, up to polylog things , indeed we simply need to see events of $A$ happening.

# Some formalism (sorry for that)

## Counting Algo.

Input : A ground set $X$ and some predicate $I(\in \mathbf{P})$

output : an estimation of $\sharp\{x \in X | I(x) = True\} \stackrel{\text{def}}{=} \sharp x$

e.g. (Ham. cycle) : $X = \mathcal{P}(E)$, and $f(x)$ is true if $x$ is a Hamilton cycle

## Random Algorithm

Uses fair independent random bits (don't ask me how we get them)

## Qualities

$$\begin{array}{lll} \text{Approximation} & e^{-\rho}\sharp x \leq output(x) \leq e^{\rho}\sharp x & (\varepsilon - Approx) \\ \text{Success} \geq \tau & Prob[\text{to be } \rho - approximated] \geq \tau & (1) \end{array}$$

## Sampling Algo.

Input : A finite Probability measure $P(X)$, over $X$

output : Algorithm returns $X$ with proba : $P_{Algo}(x)e^{\rho} \leq P(X) \leq P_{Algo}e^{-\rho}$

# Counting $\rightarrow$ Sampling ?

We need a bit more than couting

we Assume that we can still count when we fix a part of the solution

Classical usage of conditional expectations

## Example

- $\mu(G, F_1, F_0) =$ Number of matching in $G$ containing $F_1$ discarding $F_0$
- Nothing more than $\mu(H = f(G, F_1, F_0))$ (here $H =$ remove from $G$ vertices appearing in $F$ and remove $F_0$ from $E$)

  Algo:
  - Compute $N = \mu(G, \emptyset)$ and $p(e) = \mu(G, \{e\}, \{\})$ and $1 - p(e) = \mu(G, \{\}, \{e\})$
  - Pick $e$ with probability $p(e)$, otherwise discard it
  - Procede inductively, either with $(G, \{e\}, \{\})$ or $(G, \{\}, \{e\})$

  If $\mu()$ is exact $\rightarrow$ Perfect Random Generator of matchings.
- Error $e^{\varepsilon}$ on $\mu \rightarrow$ Drift of $e^{t\varepsilon}$ ($t$ steps) (gen. $exp(\sum_{i=0,\dots t} \varepsilon_i)$)

# Sampling → Counting ?(Prince Albert Revenge)

## Monte Carlo on nested areas (often works)

$\mu(A_n) = \frac{\mu(A_n)}{\mu(A_{n-1})} \ldots \times \frac{\mu(A_1)}{\mu(A_0)}\mu(A_0)$

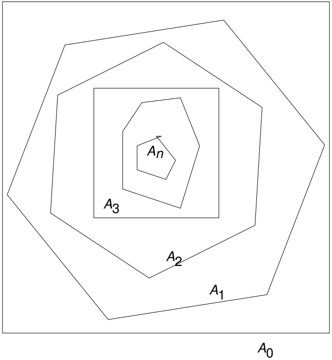
$Pr[A_{i+1}|A_i] = \mu(A_{i+1})/\mu(A_i) = \alpha_i$

$(1+\varepsilon_0)$ approx of $\alpha_i$ takes likes $\frac{1}{\alpha_i \varepsilon_0}$

$n$ steps $\varepsilon = \sum \varepsilon_i$, $\varepsilon_i = \frac{\varepsilon}{n}$

if $\alpha_i \geq \beta \rightarrow$ around $\frac{n^2}{\beta\varepsilon}$

Direct : Pay like $\beta^n$ to observe one $A_0$ in $A_n$




**examples** Matchings (add more and more edges) $\mu(G + \{e\}) \leq 2\mu(G)$, forests, colorings with more than $\Delta$ colors, knapsacks with cost less than $C$ …

# Some formalism : The $\sharp P$ Class

### What contains $\sharp P$ ?

**Unformaly :**

*Any Counting problem that can be associated to successful computations of a Non Deterministic Turing Machine (in Polynomial time)*

### Counting Prob in $\sharp P$

| | | |
|---|---|---|
| Elements of a Set $S(x)$ | Bijection | $\{y \mid TM(x,y)$ says ok $\}$ |
| Elements of a Set $S(x)$ | Bijection | Correct proofs that $(x,y) \in \mathcal{S}$ |

### Example

Ham. Cycle : $x = (V, E)$, $S(x) = \{$Ham. Cycles of $(V, E)\}$, the proof is the cycle itself. For *SAT* where $x$ is the instance (the graph), $y$ is the variable assignement (set of edges) and the machine checks that it works.

# Cook theorem and $\sharp P$-completeness of $3SAT$

## Theorem (Fake)

$$\sharp 3SAT \text{ is } \sharp P - complete.$$

**Proof.** Almost a tautology.

Correctness of a NdetTM computation can be captured by a (big) $3SAT$ formula.

It's Cook's Theorem, mostly says computation is local

| | | |
|---|---|---|
| $3SAT$ variables | bijection | (Random) Choices of the NdetTM |
| $3SAT$ Solutions | bijection | Sucessfull Choices of the NdetTM |

■

## Remarque

*Indeed One says that Cook reduction is parsimonious.*

## Counting Solutions of NP-hard problems ?

- Not really interesting, Almost immediately $\sharp P - complete$
- No approximation theory (deciding 0 or 1 is hard, $\infty$ ratio).
- Easy to amplify the number of solutions (add $k$ fake binary clauses $\times 2^k$)

## Counting exactly is way too strong and complicated

**Theorem (Toda 25-AM/1998)**

*Any problem in the Polynomial hierarchy can be solved using a counter. Fancy Madmen notation is*

$$PH \subset P^{\sharp P}$$

**madness pays off**

Let us be silly and get the godel prize !

# Valiant Vazirani, isolation lemma [11AM]

**Theorem**

*If you can solve problem when they have unique solution you can solve SAT (up to some randomization)*

**Detecting unique solutions**

- 0 solution $\rightarrow$ says 0
- 1 solution $\rightarrow$ says 1
- $>1$ output garbage, anything.

Idea :

- Add linear constraints (see prob. in $Z_2^n$).
- Dichotomy, one contraint $\rightarrow$ Should divide the solution state by 2
- turn linear constraints into extra clauses (silly but needed)

# Isolation Lemma (2)

**Theorem (Isolation)**

- *$S$ any set of $Z_2^n$*
- *pick constraints $H_i = \{x \mid v_i.x = 0\}$ randomly,*
- *let $S_0 = S, S_{i+1} = S_i \cap H_i$.*

*Then with probability $P \geq \frac{1}{4}$ we have $\exists i, |S_i| = 1$.*

So with positive probability once can construct a SAT instance that is stonger (more constrained) than the original one and that admits a single solution.

# Toda proof 3

Sorry No Godel Prize for you !

# Let's try something simpler

## We may still do something for . . .

- simple Path, trees
- Matchings
- Polytopes

## About Matchings ? Fun situation

Counting exactly Matching is $\sharp P$-complete [Valiant 6-AM/79]

One can approx count (and generate) Matchings [Jerrum 21-AM/95]

# Permanent, One factor, Matchings

**Definition (Permanent, determinant)**

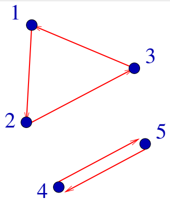
$$Perm(A) = \sum_{\pi \in \mathfrak{S}_n} a_{i,\pi(i)} \quad \Big| \quad Det(A) = \sum_{\pi \in \mathfrak{S}_n} sign(\pi) a_{i,\pi(i)}$$

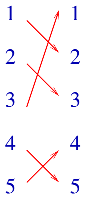
Term of the sum $= 0 \iff$ some edge $(i, \pi(i))$ does not exist.

Term of the sum $= 1$ if all the edges $(i, \pi(i))$ exist

$\Rightarrow$ The permanent counts *One Factors* of $G$

It also counts Matchings in $[G, G]$.




Weigthed version : Instead of 1 we count $\Pi_{e \in F} w(e)$ for a factor (a matching) $F \subset E$

Formal version : Multivariate Generating serie of the Matchings

# Proof Structure

## Proof Organisation

- # Weigthed Matchings $\overset{\text{regular reduction}}{\Longleftrightarrow}$ # Exact Covers by Triples
- # Weighted Perfect Matchings $\rightarrow$ Exact couting for Weigthed Matchings
- Emulating integral weights.
- #Perfect Matchings int. weigths $\rightarrow$ Can count with any weights.

# A small Gadget

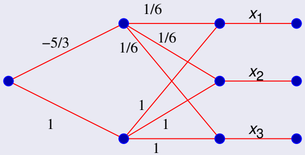
## Main property

$$Perm(G(x_2, x_2, x_3)) = \frac{1 + x_1 x_2 x_3}{3}$$

No term with degrees $1, 2 \rightarrow \forall A \subset \{x_1, x_2, x_3\}$

$\rightarrow$ $\sharp\{M \in Matching | M \cap \{e_1, e_2, e_3\} = A\} = 0$ unless $A = \{x_1, x_2, x_3\}$ or $A = \emptyset$

## The Gadget (uses negative weigth)




**Graph** $G(x_1, x_2, x_3)$

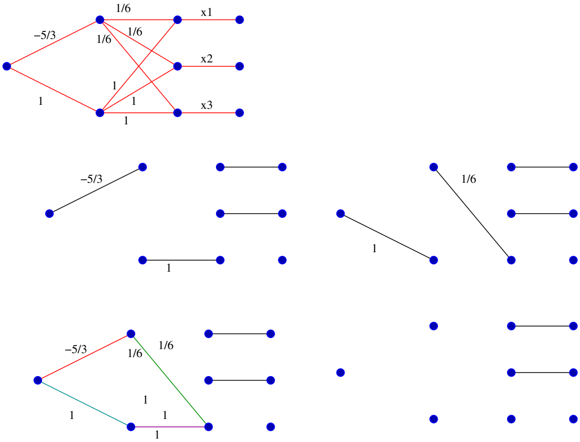
# Just Checking $Perm(G(1,1,0)$.

- 1) $M$ contains two edges, two cases : $\frac{-5}{3} \times 1$ and $\frac{1}{6} \times 1$ (tot. $\frac{-9}{6}$)
- 2) $M$ contains one edge (4 cases) : $-\frac{5}{3} + 1 + 1 + \frac{1}{6}$ (tot. $\frac{3}{6}$)
- 3) $M$ is empty 1 (tot. $+1$)

total contribution is zero

similarly:

$P(1,1,1) = \frac{-5}{3} + 1 + 1 = \frac{1}{3}$



# Consequence

## Property

When we attach the gadget $H$ with its 3 ends to a graph, computing $Perm(G+H)$ compute the "number" of matchings that either contain $\{e_1, e_2, e_3\}$ or do not intersect it.

Gadgets behave like a triple

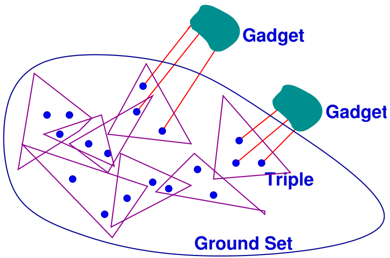
$S$: Instance of cover-with-triples , $3m$ elements (ground set).

Triples and gadgets ( Bijection )

What do we count ? The Exact covers ? No! We count triple-disjoint partial cover

$k$ disjoints triple (+stuff):$\left(\frac{1}{3}\right)^k$

$Perm(H(S)) = \sum \frac{N(k)}{3^k}$, $N(k)$ number of disjoint $k$ covers.



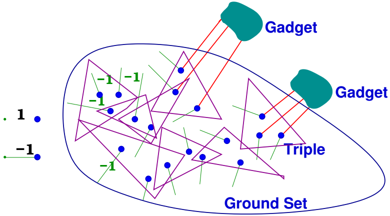
# Getting ride of partial Covers

Add pending leaves to vertices of the ground set.

Edge weight is $-1 \Rightarrow$ Graph $H'(S)$

We count now 0 for a partial Cover.

We still count $\frac{1}{3^m}$ for a perfect cover.

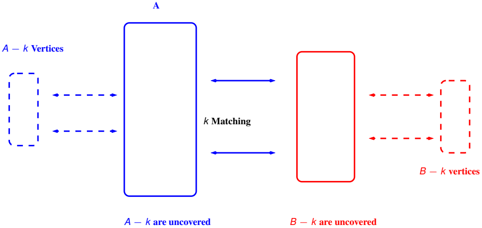



$$Perm(H'(S)) = \frac{\text{number of exact covers}}{3^m}$$

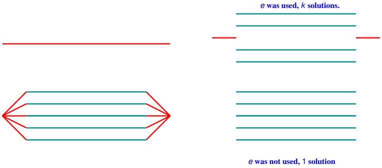
# Some More gadgets

- Matching $\rightarrow$ Perfect Matching

$\forall k$ We count each $k$ matching $(A-k)!(B-k)!$ times




- Simulating Integral weigths




- At the end 4 bad weights $x = \frac{1}{6}, y = \frac{5}{3}, z - 1$ Polynomial on a bounded number ($k = 4$) of variables, degree $n$ (polynomial), $n^4$ coefficients $\rightarrow$ can be computed.

# Even couting Simple walk is Difficult

**Unless $P = NP$**

Assume we can approx. Generate Simple Walk

Amplify probability of long walks

$\rightarrow$ Can Solve Hamilton cycle.

**Proposition**

*Generating unbiased (even very approx) simple walk, or couting them (even very badly) is NP-hard*

Weaker reduction (to $NP$) but similar idea can work for much more problems.

# General framework

## Ideas

- Move randomly in the state space (form Use Markov Chains)
- Ensure that moves are fair (the station. distribution is uniform)
- Want to be random fast (Ensure Rapid-Mixing)

## Theorem (Perron Frobenius + some Folks)

*A stochastic matrix $M$ admits a unique fixed point (eigenvector with eigenvalue 1) and everything else decays fast. i.e if $u.1 = 0$ (noise), then $M^t u \to 0$*

More or less: eigenvalues $1 = \lambda_1, \lambda_2, \dots \lambda_n$

$$|M^t(u) - u_0| \leq (1 - \lambda_2)^t$$

Where $\lambda_2 < 1$ depends on the structure of the chain $M$.
State space $S$ it converges in $\frac{\log |S|}{\log_2(1-\lambda_2(M))}$

# Limitation, Difficulties

Design a Good Enough Chain ...

Prove that it converges fast

No way to compute $\lambda_2$ numerically

State Space is of exponential size

Works only for symmetric chains (but you design it)

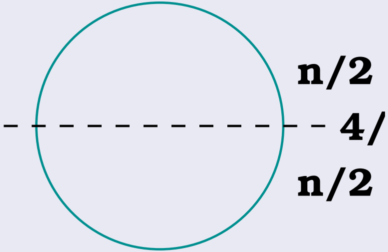
Stupid condition (non bipartite), solved making chain Lazy, loop half of the time

## Typical Fake-chain

- Pick $\frac{|V|}{2}$ edges,
  if they form a matching return it
  else play again
- Mathematically sound, return unbiased matching
- Mixes slowly (loops forever)

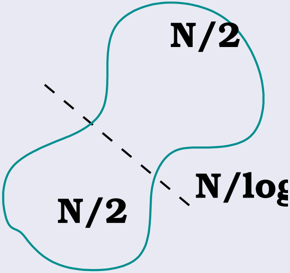
## Bad Guy: Cycle

- Totally random moves, takes $\theta(n^2)$ to get random (unbiased random walk with $t$ steps move away from zero lie $\theta(\sqrt{t})$.
- actual time to mix is $\frac{n^2}{2\pi^2}$.
- Very bad expansion, $\frac{2}{n}$




## Good Girl: De Bruijn

- Binary chains, length $n$, shift and inject a new bit.
- random moves, takes $n$ to exactly anywhere with probably $\frac{1}{2^n}$
- actual time to perfectly mix is $\log_2 |S| = n$.
- Good expansion, $\sim \frac{1}{log_2 n}$



# Why do Liner Algebra Matter

## How does the discrepancy evolve ?

- Look how non uniform is a distribution $X \to \delta(X) = \sum_{e=(u,v)\in E} |X_u - X_v|^2$
- $I$ an Incidence matrix of the graph
- $\delta(X) = |IX|^2 = XI^t IX$
- $\mathcal{L} = II^t$ is the Laplacian of $G$
- $\mathcal{L} = \Delta(G) - M$ ($M$ adjacency matrix, $\Delta$ diagonal of the degrees)
- $G$ is regular : $\mathcal{L} = \Delta Id - II^t$.
- Normalisation : divide by $\Delta$
- $1 - \lambda_2$ is the largest eigenvalue of $\frac{Id-M}{\Delta}$ which is a SDP matrix.

# Link with congested cuts

- $X$ indicator vector for $S$ : $X_u = 1, u \in S, X_u = -1 u \in \overline{S}$
- $\rightarrow I^t IX = 4|[S, \overline{S}]$

**Definition (isoperimetric constant, conductance)**

$$\phi = Min \frac{[S, \overline{S}]}{|S|}$$

**High Conductance = Rpid Mixing**

$$\frac{\phi^2}{2} \leq 1 - \lambda_2 \leq 2\phi \qquad \text{(Cheeger inequality)}$$

To prove rapid-mixing $\rightarrow$ Prove that conductance is high.

Hum ? Need to have an idea of the **??**

still complicated

# The cannonical Path Idea

Define a cannonical path between any pair of states

Hum ? just a routing indeed

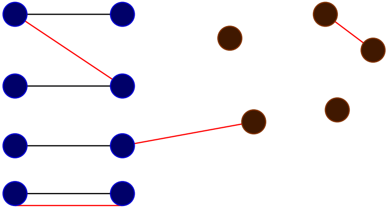
Get low congestion of the edges of $G$

Here low means logarithmic in the state space size $|S|$ (i.e indeed polynomial).

# A "fast" Chain for matchings

**$M$ curent solution, select $e \in E$ Randomly.**

1) No extremity covered $\rightarrow M \cup \{e\}$
2) 1 extremity cov. (by $f$) $\rightarrow M \setminus \{f\} \cup \{e\}$
3) 2 extremities cov., $e \notin M \rightarrow M$
4) $e \in M \rightarrow M \setminus \{e\}$

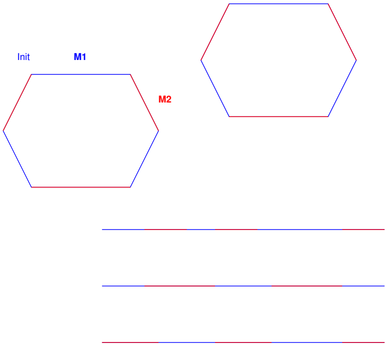


# A good routing

Take the symetric difference of the two matchings.

Order the vertices, induce order on the components

Process by component : for each start from the "first" vertex and do the augmenting path thing.



# Case of Trees

## Cayley formula : labeled trees on $K_n$(comp.graph)

Induction is $\forall e = (u, v) N(G) = N(G \setminus \{e\}) + N(G[u = v])$.
Generalizes as a determinant for general $G$.

## Markov Chain

- Potentialy Rapidly Mixing Chain : Take an edge and flip it (like when you look for the Min Cost Spanning Tree)
- Prob. Mixes fast (need to check)
- But there is Better . . .

## Super Smart Generator

Move in $G$ randomly add edges to your tree unless it makes a cycle.

Mixes perfectly

# Sampling from inside a Polytope (Lovasz & Simonovits)

We are given a "Nice" Polytope (the solutions of a linear program)

### Random Walk inside $P$

discrete : Divide into cells, make a discrete randomwalk.

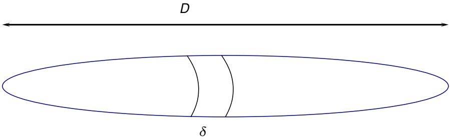
conti: $x \to$ Move randomly inside $B(x, \rho) \cap P$

Complicated :

If $\rho$ big we haven't done anything

$\rho$ small $\to$ No move (mixes slowly)

Continuous space, uniformity ?




Mixing time :

### Poincaré Inequality

Up to some conditon, for a convex body diameter $D$ :

$$Congestion \leq \Theta(\frac{D^2 n}{\delta})$$

# Stuff to Remember (Ubiquitous Take Home Slide)

Very few things we can do practically

Markov chains may work, hard to construct

## More ?

David Aldous Book (future book)
`http://www.stat.berkeley.edu/~aldous/RWG/book.pdf`

Lázló Lovász papers, monographies :
`http://matmod.elte.hu/~lovasz/randwalk-papers.html`

Marc Jerrum & Alistair Sinclair work.

Fan Chung book (spectral graph theory)

Karp & Luby for DNF (easy, just cond. expectation), coupling from the Past (exact simulation),