Cours ENSL: Big Data – Streaming, Sketching, Compression

Olivier Beaumont, Inria Bordeaux Sud-Ouest
Olivier.Beaumont@inria.fr
Introduction
Positionning

- w.r.t. traditional courses on algorithms
  - Exact algorithms for polynomial problems
  - Approximation algorithms for NP-Complete problems
  - Potentially exponential algorithms for difficult problems (going through an ILP for example)

- Here, we will consider extreme contexts
  - not enough space to transmit input data (sketching) or
  - not enough space to store the data stream (streaming)
  - not enough time to use an algorithm other than a linear complexity one

- Compared to the more "classical" context of algorithms:
  - we aim at solving simple problems and
  - we are looking for approximate solutions only because we have very strong time or space constraints.

- Disclaimer: it is not my research topic, but I like to look at the sketching/streaming papers and I am happy to teach it to you!
Application Context 1: Internet of Things (IoT)

- Connected objects, which take measurements
- The goal is to aggregate data.
- Processing can be done either locally, or on their way (fog computing), or in a data center (cloud computing).
- We must be very energy efficient
  - because objects are often embedded without power supply.
- E3nergy cost: Communication is the main source of energy consumption, followed by memory movements (from storage), followed by computations (which are inexpensive)
- A good solution is to do as many local computations as possible!
  - but it is known to be difficult (distributed algorithms)
  - especially when the complexity is not linear (e.g. think about quadratic complexity)
- Solution:
  - compress information locally (and on the fly)
  - only send the summaries; summaries must contain enough information!
Application Context 2: Datacenters

- Aggregate construction
- except the network (we can have several levels + infiniband), everything is "linear"
- the distance between certain nodes/data is very large but a strong proximity with certain data stored on disk
- with 1,000 nodes with 1TB of disk and a link at 400 MB/s, we have 1 PB and 400 GB/s (higher than with a HPC system)
- provided the data is loaded locally!
- for 25 TF/s ($10^3$25GFs seti@home) in total, ratio 60 (HPC system 40 000)
- in practice, dedicated to linear algorithms and very inefficient for other classes.
- In both contexts, there is a strong need to have data driven algorithms (where placement is imposed by data) whose complexity is linear
Sketching – Streaming
• large volume of data generated in a distributed way
  • to be processed locally and compressed before transmission.

• Types of compression?
  • lossless compression
  • compression with losses
  • compression with losses, but controlled tightly controlled loss for a specific function (sketching)

• + we are going to do compression on the fly (streaming)
On-the-fly compression dedicated to a function \( f \)

- Easy problems?
  - examples: \( \min, \max, \sum, \) mean value median?
  - Constraint: linearize the computations (later on plagiarism detection)

- How?
  - The solution is often to switch to \textit{randomized approximation algorithms}. 
Compression associated to a specific function $f$

- More formally, given $f$, we want to compress the data $X$ but still be able to compute $\approx f(X)$.
- Sketching: we are looking for $C_f$ and $g$ such that
  - the storage space $C_f(X)$ is small (compression)
  - from $f(X)$, we can recover $f(X)$, i.e., $g(C_f(X)) \approx f(X)$
- Streaming: additional difficulty, the update is performed on the fly.
  - we cannot compute $C_f(X \cup \{y\})$ from $X \cup \{y\}$
  - since we cannot store $X \cup \{y\}$
  - so we need another function $h$ such that $h(C_f(X), \{y\}) = C_f(X \cup \{y\})$
- and one last difficulty:
  - very often, it is impossible to do in deterministic and exact / deterministic and approximate
  - but only with a randomized and approximation algorithm.
- How to write this?
  - We are looking for an estimator $Z$ such that for given $\alpha$ and $\epsilon$
  - $Pr(|Z - f(X)| \geq \epsilon f(X)) \leq \alpha$. How to read this?
    - the probability of making a mistake by a ratio greater than $\epsilon$ (as small as you want)
    - is smaller than $\alpha$ (as small as you want)
Example: count the number of visits / packets

- **Context**
  - a sensor/router sees packets / visits passing through,....
  - you just want to maintain elementary statistics (number of visits, number of visits over the last 1 hour, standard deviations)
  - Here, we simply want to count the number of visits

- What storage is necessary if we have \( n \) visits? \( \log n \) bits. Why ? Pigeonhole principle. If we have strictly less than \( \log n \) bits, then we have two events (among the \( n \)) that will be coded in the same way.

- What happens if we only allow an approximate answer (say, to a factor of \( \rho < 2 \))? you need at least \( \log \log n \) bits. Why ? sketch of the proof: if we use \( t < \log \log n \) bits, then we will be able to distinguish less than \( \log n \) different groups and you can estimate how many groups are needed to count \{0\}, \{0, 1\}, \{0, 1, 2\}, \{0, 1, ..., 7\}.

- We will look for a randomized and approximated solution
  - Let us set \( \alpha \) and \( \epsilon \)
  - we are looking for an algorithm that computes \( \hat{n} \), an approximation of \( n \)
  - that only uses \( K \log \log n \) bits storage
  - and such that \( Pr(|\hat{n} - n| \geq \epsilon n) \leq \alpha \)
  - \( K \) must be a constant...not necessarily a small constant for now!
Crash Course in probabilities

- $Z$ random variable with positive values
- $E(Z)$ is the expectation of $Z$
- definitions and properties?
  - $E(Z) = \int \lambda P(Z = \lambda) d\lambda$ or $E(Z) = \sum_j jP(Z = j)$
  - $E(Z) = \int P(Z \geq \lambda) d\lambda$ or $E(Z) = \sum_j P(Z \geq j)$
  - $E(aX + bY) = aE(X) + bE(Y)$
  - total probabilities (with conditioning) $E(Z) = \sum_j E(Z|Y = j)P(Y = j)$
- To measure the distance from $Z$ to $E(Z)$, we use the variance $V(Z)$
  - Definition?
  - $V(Z) = E((Z - E(Z))^2) = E(Z^2) - E(Z)^2$
  - Properties:
    - $V(aZ) = a^2V(Z)$
    - In general, $V(X + Y) \neq V(X) + V(Y)$ (but it is true if $X$ and $Y$ are independent random variables)
- How to measure the difference between $Z$ to $E(Z)$?
  1. Markov: $Pr(Z \geq \lambda) \leq E(Z)/\lambda$
  2. Chebyshev: $Pr(|Z - E(Z)| \geq \lambda E(Z)) \leq \frac{V(Z)}{\lambda^2 E(Z)^2}$
  3. Chernoff: If $Z_1, \ldots, Z_n$ are Independent Bernouilli rv with $p_i \in [0,1]$ and $Z = \sum Z_i$, then $Pr(|Z - E(Z)| \geq \lambda E(Z)) \leq 2 \exp\left(\frac{-\lambda^2 E(Z)}{3}\right)$.
Morris Algorithm: Counting the number of events

- Step 1: Find an estimator $Z$
  - $Z$ must be small (of order of $\log \log n$)
  - we need to define an additional function $g$
  - such that $E(g(Z)) = n$

- Morris algorithm
  - $Z \rightarrow 0$
  - At each event, $Z \rightarrow Z + 1$ with probability $1/2^Z$
  - When queried, return $f(Z) = 2^Z - 1$

- What is the space complexity to implement Morris’ algorithm?
- What is the time complexity in the worst case? What is the expected complexity of a step?
- Prove the correctness: $E(2^{Z_n} - 1) = n$ (note $Z_n$ the random variable that denotes $Z$ after $n$ events) Hint: by induction, assuming that $E(2^{X_n}) = n + 1$ and showing that $E(2^{X_{n+1}}) = n + 2$
- How to find a probabilistic guarantee of the type $Pr(|f(X_n) = \bar{n} - n| \geq \epsilon n) \leq \alpha$? Hint Prove $E(2^{2X_n}) = 3/2n^2 + 3/2n + 1$.
- Conclusion? Is this unexpected?
From Morris to Morris+ and Morris+++ 

- 2nd step: How to get a useful bound?
- Objective: to reduce the variance (expectation is what we want). How to do it?
  - Classic idea: do the same experience many times and average them
- Morris algorithm +
  - Morris is used to compute independent $Z^1_n, Z^2_n, \ldots, Z^K_n$
  - On demand, compute $Y_n = \sum_i Z^i_n$ return $f(Y_n) = 2^{Y_n} - 1$
- Questions:
  - Which space complexity to implement Morris+'s algorithm?
  - What time complexity?
  - Establish the correctness: $E(2^{X_n} - 1) = n$
  - What is the new guarantee obtained with Chebyshev? How many counters should be maintained?
- How can we do even better?
  - Morris+++ = Morris+(1/3) and median
  - proof with Chernoff: If $Z_1, \ldots, Z_n$ are Independent Bernouilli rv with $p_i \in [0,1]$ and $Z = \sum Z_i$, then 
    $Pr(|Z - E(Z)| \geq \lambda E(Z)) \leq 2 \exp(-\lambda^2 E(Z)/3)$. 

2nd example: how to count the number of unique visitors

Context

• It is assumed that visitors are identified by their address \((i_k \in [1, n])\)
• We observe a flow of \(m\) visits \(i_1, \ldots, i_m\) with \(i_k \in [1, n]\)
• How many different visitors ?
• Deterministic and trivial algorithms:
  • if \(n\) is small, if \(n\) is big… and in front of what?
  • solution in \(n: n\) bit array
  • solution in \(m \log n\): we keep the whole stream!
• We will see a bit later
  • that we cannot do better with exact and deterministic algorithms
  • that we cannot do better with approximated and deterministic algorithms
• How to do if you cannot store \(n\) bits
  • but only \(O(\log^k n)\) for a certain \(k\)?
• we will see that it is again possible by using both randomization and approximation.
• and that no deterministic exact or deterministic approximation can do it with this space constraint.
We will start with an idealized algorithm (which cannot be implemented in practice).

- Let us choose a random $h$ function from $[1, n]$ to $[0, 1]$
- Why idealized?
  - Problem 1: to store such a random function, you must define the images for in each of the $n$ points... at least $\Omega(n)$ bits
  - Problem 2: and in addition we would have to store real values!
  - We will come back to these two problems in a moment....
  - Let us assume for now that storing such a function costs $\Theta(1)$
- How do you keep track of the number of unique visitors?
- We will keep $Z \longrightarrow \min_{i \in \text{stream}} h(i)$. Intuition?
  - If you see the same visitor $k$ times, it won’t change $Z$
  - If we see $t$ different visitors, then the values taken by $h$ split $[0, 1]$ in $t + 1$ intervals...and all should have the same size in expectation... and this size is $\frac{1}{t+1}$ including the first !
- so you should return $\frac{1}{Z} - 1$ !
Proof of correctness

- Let’s prove that \( E(Z) = \frac{1}{t+1} \).
- \( E(Z) = \int_{0}^{+\infty} P(Z \geq \lambda) d\lambda \).
  - Show that \( E(Z) = \frac{1}{t+1} \)
  - How to continue? by calculating the variance and applying Chebychev
  - Prove that \( E(Z^2) = \frac{2}{(t+1)(t+2)} \)
  - There is still one foolishness not to be said.... \( E(1/Z) \neq 1/E(Z) \)
  - Intuition: if we can control closely \( Z \) and \( \frac{1}{t+1} \), \( 1/Z - 1 \) will be close to \( t \)

- FM+
  - Let us maintain \( q = \frac{1}{\epsilon^2 \eta} \) FM instances.
  - \( Z_i \) is the value produced by \( FM_i \)
  - What to return? \( Y = \frac{1}{(\sum_1^q Z_i)/q} - 1 \)
  - \( E(\frac{\sum_1^q Z_i}{q}) = \frac{1}{t+1} \)
  - \( V(\frac{\sum_1^q Z_i}{q}) = \frac{t}{q(t+1)^2(t+2)} < \frac{E(Z)^2}{q} \)
  - Claim 1: \( P(lY - \frac{1}{t+1} l \geq \frac{\epsilon}{t+1}) \leq \eta \)
  - Claim 2: \( P(l 1/Y - 1 - tl \geq \Theta(\epsilon)t) \leq \eta \)

- FM++
  - choose \( \eta = \frac{1}{3} \) adapt \( \epsilon \), instantiate \( K \) copies of \( Y Y_1, \ldots, Y_K \)
  - output median\{ \frac{1}{Y_i} \} Ok for \( K = \lceil 36 \log(\frac{1}{\delta}) \rceil \)
Toward a Non Idealized Version. A crucial tool: hashing functions

- We used the set of all possible functions (too large set, to large. storage for one function)
- To make it practical, we will consider a large (not too large) family of functions $H$ from $[1, p] \rightarrow [1, p]$
- How to define the quality of a family $H$?
- Notion of $k$-wise independence
  - $\forall i_1, \ldots, i_k, \forall j_1, \ldots, j_k, i_k \neq i_l$, and if we pick a random $h$ function in $H$, then
  - $P(h(i_1) = j_1$ and $h(i_k) = j_k) = 1/p^k$
  - a larger $k$ provides a "better" family
- Examples:
  1. the set of all functions from $[1, p] \rightarrow [1, p]$ is Ok.
    - What $k$, what storage cost?
    - $f(1) \rightarrow p$ choices, $f(p) \rightarrow p$ choices
    - Problem: expensive, $p \log p$ bits are necessary for one function
  2. with the polynomials $H^k_{\text{poly}}$ of degree $k$ in $F_p$
    - evaluation cost? for degree $k$, $k$ mult & and adds
    - independence? how many polynomials such that $(h(i_1) = j_1$ and $h(i_k) = j_k$
    - exactly one, Lagrange polynomial: $P = \sum_{r=1}^{k} \prod_{l\neq r} (X-i_l) \times j_r$
    - choice? picking a function at random in $H^k_{\text{poly}} \rightarrow$ choose $k + 1$ coefficients.
    - and thus the family $H^k_{\text{poly}}$ is $k$--independent
Non Idealized FM (1)

- **Step 1:** find a $O(1)$-approximation $\tilde{t}$ of $t$ in $O(\log n)$ bits, i.e. a constant $C$ such that $\frac{t}{C} \leq \tilde{t} \leq Ct$ with constant probability (say $\frac{2}{3}$)
  1. Pick $h$ from a 2-wise family from $[n]$ to $[n]$ (works $\forall n$ but complicated, otherwise round to $2^k$, or assume that $n$ is a prime).
  2. Maintain $X = \max_{i \in \text{stream}} \text{lsb}(h(i))$ (lsb: least significant bit)
  3. Output $2^X$

- **Intuition:**
  - $P(\text{lsb}(h(i)) = j) = \frac{1}{2^{j+1}}$, so $E(\{i, \text{lsb}(h(i)) = j\}) = \frac{t}{2^{j+1}}$ and $E(\{i, \text{lsb}(h(i)) > j\}) \approx \frac{t}{2^{j+2}} + \frac{t}{2^{j+3}} + \ldots \approx \frac{t}{2^{j+1}}$.
  - What happens when $j$ is of order $\log t$...
    - there is $\simeq 1$ visitor such that $\text{lsb}(h(i)) = j$
    - there is $\simeq 1$ visitor such that $\text{lsb}(h(i)) > j$
  - Thus, if $j$ is of order $(\log t) - 5$ it is very unlikely ($1/2^5$) that there is no $i$ s.t. $\text{lsb}(h(i)) \geq j$
  - Thus, if $j$ is of order $(\log t) + 5$ it is very unlikely ($1/2^5$) that there is a $i$ s.t. $\text{lsb}(h(i)) \geq j$
  - with good probability, $\tilde{t} = 2^X$ is in $[\frac{t}{C}, Ct]$

- **The proof is very similar to what we have done, with one tricky issue**
  - how to use 2-wise independence?
  - fix $j$, define $Y_i = 1$ iff $\text{lsb}(h(i)) = j$ so that $Z_j = \sum_i Y_i$, then $E(Z_j) = \frac{1}{2^{j+1}}$
  - as usual we need $V(Z_j)$ to control probabilities and $V(Z_j) = E((\sum_i Y_i)^2) - E(\sum_i Y_i)^2 = \sum V(Y_i) + \sum_{i \neq k} E(Y_i Y_j) - E(Y_i)E(Y_j) = \sum V(Y_i)$ because 2-wise independence says that $E(Y_i Y_j) = E(Y_i)E(Y_j)$!
Non Idealized FM (2)

- Playing with constants, let us assume that Step1 provides a 32-approximation with probability $\frac{2}{3}$, then perform $K$ experiments and take the median to have 32-approx with large probability
- To obtain a stronger approximation, we rely on the following technique
  - let us chose $g$ in a 2 wise family from $[n]$ to $[n]$.
    1. Imagine that we consider $\log n$ sets, with $S_j$ contains the elements $i$ of the stream s.t. $\text{lsb}(g(i)) = j$.
    2. we know $\tilde{t}$ (close to $t$), let us denote by $Z$ the size of $S_j$ when $2^{j+1} \approx \tilde{t}\epsilon^2$
    3. and let consider $U = 2^{j+1}Z$ in this case
- $E(U) = 2^{j+1}E(Z) = t$, $V(U_i) = 2^{2j+2} \text{Var}(Z) \leq t2^{2j+1}$
- so that (Chebychev) $P(|U - t| \geq \epsilon t) \leq \frac{t2^{j+1}}{\epsilon^2 t^2} = \frac{2^{j+1} \tilde{t}}{\epsilon^2 \tilde{t} t} \leq C'$
- Then, we use several hashing functions and take the average value to obtain an error with arbitrarily small probability
- Not completely finished ! Is this algorithm implementable this time with small space ?
- No, because $S_0$ is very large for instance ! But the maximum value we are expecting in "interesting" $S_j$ is $\frac{t}{2^{j+1}} = \frac{\tilde{t}}{2^{j+1}} \frac{t}{\tilde{t}} \leq \frac{C}{\epsilon^2}$
- Thus, we can "only" remember the first $\frac{C}{\epsilon^2}$ is each set !
- Overall space complexity ???
• Technique called Geometric sampling

• $n$ elements in the stream, $k \leq n$ distinct elements (with respect to some property)

• Store $\log n$ sub-streams, where $S_0$ stores $1/2$ of the elements (distinct wrt the property), $S_1$ stores $1/4$ of the elements, ... $S_{\log k}$ stores (close to) 1 element, $S_{\log n}$ a priori stores nothing if $k \ll n$

• Suppose that when there are $l$ elements in one of the sets, we can find a good estimation of $k$ where typically $l$ is of order $\frac{1}{\epsilon^2}$

• Then, we bound all the sets to store less than $10l$ elements (they are useless after that)

• if we have a constant approximation of $k$ (obtained elsewhere), then we know in which set we should look at.
Why do we need randomization and approximation?

- Because a deterministic algorithm needs at least $\Omega(n)$ bits
- How to prove this? We assume $n = \Theta(m)$
- Let us consider the state of the memory of the algorithm after seeing $i_1, \ldots, i_m$
- We need to prove that there is enough information in what is stored
- so as to differentiate $2^n$ distinct elements
- Remark: you can add as many computations as you want!
- Input $X$, let us denote by $C_f(X)$ the state on the memory
- What can be computed using $C_f(X)$ (and only $C_f(X)$)?
- we can compute $h(C_f(X))$ and $h(C_f(X), \{y\}) = C_f(X \cup \{y\})$
- do it for all possible $y$ values (visitors)...
- If $y$ was in the stream, then $h(C_f(X), \{y\}) = h(C_f(X))$ otherwise $h(C_f(X), \{y\}) = h(C_f(X)) + 1$
- In $C_f(X)$, there is enough information to distinguish $2^n$ possible vectors (all visitors vectors)
- and thus $n$ bits are needed!
Why do we need randomization and approximation?

- Because a deterministic approximation algorithm (say 1.1-approx) needs at least $\Omega(n)$ bits
- Let us suppose that there exists a collection $C$ of subsets of $n$ such that
  - $|C|$ is large ($\geq \exp(n/10^4)$)
  - $\forall S \in C, |S| = n/100$ (sets are large)
  - $\forall S_1, S_2 \in C^2, |S_1 \cap S_2| \leq n/2000$ (intersections are small)
- General idea
  - Let us assume that we have presented to the algorithm one of the sequences of $C$
  - Then, we can find back which one!
  - just by trying exhaustively all $\#C$ sequences with $C_f(X)$
  - Since we know how to differentiate exponentially many $(\exp(n/10^4))$ elements, we need $\Omega(n)$ bits
- We still need to prove that such a set $C$ exists!
  - $n$ visitors numbered from 1 to $n$ split into $n/100$ packets of 100 visitors
  - In $S_i, \forall i$ we randomly choose one visitor per packet
  - we build $\exp(n/10^4)$ such sets $S_i$.
  - easy: What is their size? $n/100$
  - we need to check that $\forall i, j, i \neq j, |S_i \cap S_j| \leq n/2000$
  - How to do this ?it is enough to prove that the $P(it\ works)$ is $> 0$
  - Why does it work ? $Y_{i,j}$ number of collisions between $S_i$ and $S_j$
  - $E(Y_{i,j})$ ? $Pr(Y_{i,j} > n/2000)$ ? $Pr(\exists i, j t.q. Y_{i,j} > n/2000)$ ?