Algorithm Frameworks Based on Structure Preserving Sampling

Richard Peng
Georgia Tech
OUTLINE

- Sampling / sparsification
- Sparsifying without construction
- Algorithmic frameworks
RANDOM SAMPLING

Pick a small subset of a collection of many objects

Goal:
- Estimate quantities
- Reduce sizes
- Speed up algorithm

Point sets
Gradients
Graphs
SAMPLING IN ALGORITHMS

Randomized size reductions used in
• Low rank approximations
• Linear system solvers
• Combinatorial optimization

• Compute on the sample
• Need to preserve more structure
PRESERVING GRAPH STRUCTURES

Undirected graph, \( n \) vertices, \( m < n^2 \) edges

Is \( n^2 \) edges (dense) sometimes necessary?

For some information, e.g. connectivity: \(< n \) edges

Preserving more: [Benczur-Karger `96] for ANY \( G \), can sample to get \( H \) with \( O(n\log n) \) edges s.t. \( G \approx H \) on all cuts
HOW TO SAMPLE?

Widely used: uniform sampling

Works well when data is uniform e.g. complete graph

Problem: long path, removing any edge changes connectivity

(can also have both in one graph)

More systematic view of sampling?
ALGEBRAIC REPRESENTATION OF GRAPHS

Edge-vertex incidence matrix:
\[ B_{eu} = -1/1 \text{ if } u \text{ is endpoint of } e \]
\[ 0 \text{ otherwise} \]

- \( x \): 0/1 indicator vector for cut
- \( Bx \): difference across edges
- \( \|Bx\|_2 \): size of cut given by \( x \)
L₂ Row sampling:
Given $\mathbf{B}$ with $m \gg n$, sample a few rows to form $\mathbf{B}'$ s.t. $\|\mathbf{B}\mathbf{x}\|_2 \approx \|\mathbf{B}'\mathbf{x}\|_2 \forall \mathbf{x}$

Note: normally use $\mathbf{A}$ instead of $\mathbf{B}$, $n$ and $d$ instead of $m$ and $n$
UNIFORM SAMPLING

Works well when data is uniform e.g. complete graph

Hard cases:

More general: importance sampling, flip biased coin for each element
For many objectives, exists a set of ‘correct’ probabilities that sum to rank

Sampling by ANY upper bounds \( \times O(\log n) \) gives a good approximation w.h.p.

\( \tau_i = b_i^T (B^T B)^{-1} b_i \)

\( w_i^{2/p} = a_i^T (A^T W^{1-2/p} A)^{-1} a_i \)

L2 : leverage scores / effective resistances
Column subset: robust leverage scores
\( L_p \) norms: Lewis weights
[Spielman-Srivastava `08]:

- Leverage scores are same as weight times effective resistance
- Sampling a graph with probabilities at least $O(\log n)$ times these values give a good spectral approximation
MORE SYSTEMATIC VIEW

[Talagrand `90] “Embedding subspaces of $L_1$ into $L^N_1$” can be interpreted as row-sampling / sparsification

[Cohen-P `15]: for any $1 \leq p \leq 2$, input sparsity time routines for producing $A'$ with about $O(n \log n)$ rows s.t. $\|Ax\|_p \approx \|A'x\|_p \forall x$

Rest of this talk: $p = 2$ case
WHY SPARSIFICATION?

- $\|Bx\|_2 \approx \|B'x\|_2 \forall x$ is same as small relative condition number from numerical analysis
- Easier to generate than verify approximations
- Probabilities computed adaptively, still computing with all of the data

However, most data are sparse to begin with!
OUTLINE

• Sampling / sparsification
• Sparsifying without construction
• Algorithmic frameworks
DENSE OBJECTS

- Matrix powers
- Matrix inverse
- Transitive closures

Cost-prohibitive to compute / store

Directly access sparse approximates?
RANDOM WALKS

\( A \): adjacency matrix of random walk

\( A^k \): \( k \) step random walk

Still a graph, can sparsify!

- Formally, \( I - A = B^T B \)
- \( I - A \) is Gram matrix of \( B \)
LONGER RANDOM WALKS

**A**: one step of random walk

Length 3 path in **A**: u-y-z-v

(part of) edge uv in **A**^3

**A**^3: 3 steps of random walk
Rayleigh’s monotonicity: can use resistance in subgraph as upper bound

Bound $R(u, v)$ using length 3 path in $X$, $u-y-z-v$:

$$w(u, y) \times R(y, z) \times w(z, v)$$

Bound on probability: $w_{uy}^{-1} + w_{yz}^{-1} + w_{zv}^{-1}$
SPARSIFYING $I - A^K$

Repeat $O(m \log n \varepsilon^{-2})$ times:

1. Pick $0 \leq i \leq k - 1$ and edge $e = xy$ at random.
2. Random walk $i$ steps from $x$ to $u$, $k - 1 - i$ steps from $y$ to $v$.
3. Add (rescaled) edge between $uv$ to sparsifier.

Resembles:

- Local clustering
- Approximate triangle counting ($c = 3$)

‘LIGHT WEIGHT’ SPARSIFICATION

• Terrible approximations to the ‘right’ probabilities can still give size reductions
• Such estimates are often easy to compute
• Many routines already compute sparsifiers: e.g. approx. triangle counting via 2-step walks
GAUSSIAN ELIMINATION

Partial state of Gaussian elimination: Schur complement, linear system on a subset of variables,

Alternate view: equivalent circuit on boundaries,

For graphs:
- This is another graph:
- Analog of Y-Δ transform, remove u, then connect all pairs of v ~ u
Degree $d$ vertex: $O(d^2)$ new non-zero entries

Quickly leads to dense matrices (after about 5000 vertices)
WAYS OF REDUCING FILL

- Elimination orderings: $m \sim 10^5$
- Drop entries by magnitude (e.g. ichol from MATLAB): $m \sim 10^6$

[Kyng-Lee-P-Sachdeva-Spielman `15] approx Schur complements in $O(m \log^c n)$ time
HIGH QUALITY SPARSIFICATION

Methods that work in theory:
• Solve linear system in graph Laplacians
• Page rank / local clustering
• Spanners / low diameter clusterings

$$\tau_i = b_i^T (B^T B)^{-1} b_i$$
OUTLINE

- Sampling / sparsification
- Sparsifying without construction
- Algorithmic frameworks
ALGORITHMIC USES OF SAMPLES

\[ \mathbf{B}' \text{ s.t. } \| \mathbf{Bx} \| \approx \| \mathbf{B}' \mathbf{x} \| \quad \forall \mathbf{x} \]

optimization problems with extra conditions on \( \mathbf{x} \) can be solved on \( \mathbf{B}' \)

- Relative error approximation give preconditioners
- Can use iterative methods to reduce error
Study of graph properties via $A^2, A^4, A^8, A^{16}$...

[P-Spielman `14] solving linear systems / computing electrical flows:

$$(I - A)^{-1} = (I + A) (I + A^2) (I + A^4)...$$

[Cheng-Cheng-Liu-P-Teng `15]: sparsified Newton’s method for matrix roots and Gaussian sampling
MATRIX SQUARING

- NC algorithm for shortest path
- Logspace connectivity: [Reingold `02]
- Deterministic squaring: [Rozenman-Vadhan `05]

<table>
<thead>
<tr>
<th>Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
</tr>
<tr>
<td>Until</td>
</tr>
<tr>
<td>Size Reduction</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>Solution transfer</td>
</tr>
</tbody>
</table>
INCORPORATING THE SAMPLING KITCHEN SINK..

[Lee-P-Kyng-Sachedeva-Spielman `15] O(m\log^2 n) time sparse Cholesky / multigrid algorithm by incorporating
• Jacobi iteration on `heavy’ blocks
• Expanders,
CONNECTION LAPLACIANS

Motivated by AC circuits: wires have both magnitude and phase

Blocks of the form:

\[
\begin{bmatrix}
I & U \\
U^T & I
\end{bmatrix}
\]

\[
U: \text{unitary matrix: } U^T U = I
\]
Applications of Connection Laplacians

Applications:
- Cryon-electron microscopy
- Phase retrieval
- Image processing

Many combinatorial graph tools break down due to phase shifts:
cannot turn cycles into paths

Cannot use combinatorial graph sparsification algorithms
SELF-REDUCTION OF SAMPLING

Nystrom method (on matrices):
- Pick random subset of data
- Compute on subset
- Post-process result

\[ \tau_i' = b_i^T (B^T B)^{-1} b_i \]

[Cohen-Lee-Musco-Musco-P-Sidford '15]: random half of the rows give good probabilities for sampling the original

Can recursively use a solver to sparsify itself!
INTUITIONS ABOUT SELF REDUCTION

Hay in a haystack:
• Coherent data
• Half the data still contains same info

Missing: needle in haystack:
• Needles are unique, number bounded by dimension, n
• Fix mistakes via post-process, only $O(n)$ more samples

Motivation: $O(m)$ MST by [Klein-Karger-Tarjan `93]
QUESTIONS

• Do existing algorithms already produce sparsifiers?
• Sparsification packages
• Notions of approximation other than relative condition number?
• Non-linear extensions of multilevel and sparse Cholesky?
• Sparsify + precondition more general linear systems?