Randomized Composable Core-sets for Distributed Optimization

Vahab Mirrokni

Based on joint work with:
Hossein Bateni, Silvio Lattanzi, Morteza Zadimoghaddam + Aditya Bhaskara (former postdoc), Hossein Esfandiari (intern)
Google NYC Algorithms Research

Market Algorithms (Display Ads & AdWords Opt.)

Large-Scale Graph Mining & Distributed Optimization

Infrastructure Optimization

common expertise: online allocation problems

Tools:
- Clustering, PPR, Graph Building

Tools:
- Balanced Partitioning, Submodular Packing
Large-scale Graph Mining

Develop a *general-purpose library* of graph mining tools for XXXB nodes and XT edges via **MapReduce+Big Table Service (Flume), Pregel, ASYMP**

Goals:

- Develop scalable tools (Ranking, Pairwise Similarity, Overlapping Clustering, Balanced Partitioning, Embedding)
- Compare different algorithms/frameworks
- Help product groups use these tools across Google in a loaded cluster (50+ clients in Search, Ads, Youtube, Maps, Social, Local)
- Fundamental Research (Algorithmic Foundations and Hybrid Algorithms/System Research)
Three most popular techniques applied

1. Local Algorithms: Message Passing/Label Propagation/Local Random Walks
   - e.g., similarity ranking via PPR etc, Connected Components
   - Connected components code that’s 10-50 times faster the state-of-the-art

2. Embedding/Hashing/Sketching Techniques
   - e.g., Linear embedding for balanced graph partitioning to minimize cut
   - Improves the state-of-the-art by 26%. Improved flash bandwidth for search backend by 25%. Paper appeared in WSDM’16.

3. Randomized Composable Core-sets for Distributed Computation: This Talk
Composable Core-Sets for Distributed Optimization

Run ALG in each machine

Machine 1
Machine 2
Machine m

Input Set

T_1
T_2
T_m

Run ALG' on selected items to find the final output set

S_1
S_2
S_m

Selected Items

Output Set

Two rounds of MapReduce
Composable Core-sets

- **Setup**
  - \( P_1, P_2, \ldots, P_m \) are set of points in \( d \)-dimensional space
  - Optimize a function \( f \) over their union \( P \).

- **\( c \)-Composable Core-sets:** Subsets of points \( S_1 \subseteq P_1, S_2 \subseteq P_2, \ldots, S_m \subseteq P_m \) points such that the solution of the union of the core-sets approximates the solution of the point sets.

- Maximization:
  \[
  \frac{1}{c} f_{opt}(P_1 \cup \cdots \cup P_m) \leq f_{opt}(S_1 \cup \cdots \cup S_m) \leq f_{opt}(P_1 \cup \cdots \cup P_m)
  \]

- **Example:** two farthest points
Application in Streaming Computation

- **Streaming Computation:**
  - Processing sequence of \( n \) data elements “on the fly”
  - Limited Storage

- **\( c \)-Composable Core-set of size \( k \)**
  - Chunks of size \( \sqrt{nk} \), thus number of chunks = \( \sqrt{n/k} \)
  - Core-set for each chunk
  - Total Space: \( k\sqrt{n/k} + \sqrt{nk} = O(\sqrt{nk}) \)
  - Approximation Factor: \( c \)
Application in Distributed Computation

- **Streaming Computation**
- **Distributed System:**
  - Each machine holds a block of data.
  - A composable core-set is computed and sent to the server
- **Map-Reduce Model:**
  - One round of Map-Reduce
  - $\sqrt{n/k}$ mappers each getting $\sqrt{nk}$ points
  - Mapper computes a composable core-set of size $k$
  - Will be passed to a single reducer
Overview of recent results

Need to solve (combinatorial) optimization problems on large data

Will cover:
1. Diversity Maximization, PODS’14 by IndykMahdianMahabadiMirrokni
2. Capacitated $\ell_p$ Clustering, NIPS’14 by BateniBhaskaraLattanziMirrokni
3. Submodular Maximization, STOC’15 by MirrokniZadimoghaddam

New work (not covered):
4. Feature Selection (Column Subset Selection), ICML’16 by Alsch
5. Coverage Problems: submitted by BateniEsfandiariMirrokni
First Three Problems Considered

**General:** Find a set $S$ of $k$ items & maximize $f(S)$.

- **Diversity Maximization:** Find a set $S$ of $k$ points and maximize the sum of pairwise distances i.e. $\text{diversity}(S)$.

- **Capacitated/Balanced Clustering:** Find a set $S$ of $k$ centers and cluster nodes around them while minimizing the sum of distances to $S$.

- **Coverage/submodular Maximization:** Find a set $S$ of $k$ items. Maximize submodular function $f(S)$. 
Diversity Maximization Problem

• Given: $n$ points in a metric space
• Find a set $S$ of $k$ points
• Goal: maximize $\text{diversity}(S)$ i.e.
  
  $\text{diversity}(S) = \text{sum of pairwise distances of points in } S.$

• Background: Max Dispersion
  – Halldorson et al, Borodin et al, Abbassi et al
• Useful for feature selection, diverse candidate selection in Search, representative centers...
Local Search for Diversity Maximization [KDD’13]

- Used for sum of pairwise distances
- Algorithm [Abbasi, Mirrokhni, Thakur]
  - Initialize $S$ with an arbitrary set of $k$ points which contains the two farthest points
  - While there exists a swap that improves diversity by a factor of $\left(1 + \frac{\epsilon}{n}\right)$
    » Perform the swap
- For Remote-Clique
  - Number of rounds: $\log_{1+\frac{\epsilon}{n}} k^2 = O\left(\frac{n}{\epsilon \log k}\right)$
  - Approximation factor is constant.
Composable Core-set Results for Diversity Maximization

• Theorem(IndykMahabadiMahdianM.’14): The local search algorithm computes a constant-factor composable core-set for maximizing sum of pairwise distances in 2 rounds:

• Theorem(EpastoM.ZadiMoghaddam’16): A sampling+greedy algorithm computes a randomized 2-approximate composable small-size core-set for diversity maximization in one round.
  • randomized: works under random partitioning
  • small-size: size of core-set is less than k.
Distributed Clustering Problems

**Clustering:** Divide data into groups containing “nearby” points

**Minimize:**
- **k-center:** $\max_i \max_{u \in S_i} d(u, c_i)$
- **k-means:** $\sum_i \sum_{u \in S_i} d(u, c_i)^2$
- **k-median:** $\sum_i \sum_{u \in S_i} d(u, c_i)$

**Metric space** $(d, X)$

**$\alpha$-approximation algorithm:** cost less than $\alpha \cdot \text{OPT}$
Mapping Core-sets for Capacitated Clustering
Capacitated $\ell_p$ clustering

**Problem:** Given $n$ points in a metric space, find $k$ centers and assign points to centers, respecting capacities, to minimize $\ell_p$ norm of the distance vector.

$\rightarrow$ Generalizes balanced $k$-median, $k$-means & $k$-center.
$\rightarrow$ Objective is not minimizing cut size (cf. “balanced partitioning” in the library)

**Theorem:** For any $p$ and $k<\sqrt{n}$, distributed balanced clustering with

- approx ratio: ‘small constant’ * ‘best single machine guarantee’
- # rounds: 2
- memory: $(n/m)^2$ with $m$ machines

$\rightarrow$ Improves [BMVKV’12] and [BEL’13]

(Bateni,Bhaskara,Lattanzi,Mirrokni, NIPS’14)
Empirical study for distributed clustering

Test in terms of **scalability** and **quality of solution**

Two “base” instances & subsamples
- US graph ~30M nodes
- World graph ~500M nodes

<table>
<thead>
<tr>
<th></th>
<th>Size of seq. inst</th>
<th>Increase in OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>1/300</td>
<td>1.52</td>
</tr>
<tr>
<td>World</td>
<td>1/1000</td>
<td>1.58</td>
</tr>
</tbody>
</table>

**Quality**: pessimistic analysis  
**Sublinear** running time **scaling**
Submodular maximization

**Problem**: Given $k$ & submodular function $f$, find set $S$ of size $k$ that maximizes $f(S)$.

Some applications
- Data summarization
- Feature selection
- Exemplar clustering

**Special case**: “coverage maximization”: Given a family of subsets, choose a subfamily of $k$ sets, and maximize cardinality of union.
  - cover various topics/meanings
  - target all kinds of users
Submodular maximization

**Problem**: Given $k$ & submodular function $f$, find set $S$ of size $k$ that maximizes $f(S)$.

Some applications
- Data summarization
- Feature selection
- Exemplar clustering

**Special case**: “coverage maximization”: Given a family of subsets, choose a subfamily of $k$ sets, and maximize cardinality of union.
  - cover various topics/meanings
  - target all kinds of users

[IMMM’14] Bad News: No deterministic composable core-set with approx $\leq \sqrt{k} / \log k$
Submodular maximization

**Problem**: Given $k$ & submodular function $f$, find set $S$ of size $k$ that maximizes $f(S)$.

Some applications
- Data summarization
- Feature selection
- Exemplar clustering

**Special case**: “coverage maximization”: Given a family of subsets, choose a subfamily of $k$ sets, and maximize cardinality of union.
  - cover various topics/meanings
  - target all kinds of users

[IMMM’14] Bad News: No deterministic composable core-set with approx $\leq \sqrt{k}/\log k$

Randomization is necessary and useful:
- Send each set randomly to some machine
- Build a coreset on each machine by greedy algorithm
Randomization to the Rescue

Run GREEDY on each machine

Run GREEDY on selected items to find the final output set

Two rounds of MapReduce
Results for Submodular Maximization: STOC’15

• A class of $0.33$-approximate randomized composable core-sets of size $k$ for non-monotone submodular maximization. For example, Greedy Algorithm.

• Hard to go beyond $\frac{1}{2}$ approximation with size $k$. Impossible to get better than $1-\frac{1}{e}$.

• $0.58$-approximate randomized composable core-set of size $4k$ for monotone $f$. Results in $0.54$-approximate distributed algorithm in two rounds with linear communication complexity.

• For small-size composable core-sets of $k'$ less than $k$: $\sqrt{\frac{k'}{k}}$-approximate randomized composable composable core-set.
Submodular Maximization vs. Coverage Maximization

[MirrokniZadimoghaddam STOC’15]
Randomized composable core-sets work for submodular maximization in Theory.

[Mirzasculeiman et al NIPS’14]
This method works well for submodular maximization in Practice!

Almost solves max. coverage

- But, requires expensive oracle calls
  - Need to send whole “sets” around
- Handles set arrival model, does not handle “element” arrival model.
Sketching Coverage Functions

**Problems**: Given a set system \((n \text{ sets and } m \text{ elements})\),
1. pick \(k\) sets to max. size of union: “\(k\)-coverage”
2. cover *all* elements with least number of sets: “set cover”
3. cover \((1-\lambda)m\) elements with least number of sets: “set cover with outliers”
Sketching Coverage Functions

**Problems**: Given a set system \((n\) sets and \(m\) elements),

1. pick \(k\) sets to max. size of union: “\(k\)-coverage”
2. cover all elements with least number of sets: “set cover”
3. cover \((1-\lambda)m\) elements with least number of sets: “set cover with outliers”

Greedy Algorithm: Pick a subset with the maximum marginal coverage,

- 1-1/e-approx. To \(k\)-coverage, \(\log n\)-approximation for set cover...
- Goal: Achieve good fast approximation with minimum memory footprint
  - Streaming: elements arrive one by one, not sets
  - Distributed: linear communication and memory independent of the size of ground set
Max Coverage

Select k sets to cover the maximum number of elements
Sketching Coverage Functions

Problems: Given a set system \((n \text{ sets and } m \text{ elements})\),
1. pick \(k\) sets to max. size of union: “\(k\)-coverage”
2. cover all elements with least number of sets: “set cover”
3. cover \((1-\lambda)m\) elements with least number of sets: “set cover with outliers”

Results [BateniEsfandiariMirrokni’16]:
- Thm 1: A \((1\pm\varepsilon)\)-approx sketch of coverage function May NOT Help
  - Given an \((1\pm\varepsilon)\)-approx oracle to coverage function, we get \(n^{0.49}\) approximation
Sketching Coverage Functions

**Problems**: Given a set system \((n\) sets and \(m\) elements),
1. pick \(k\) sets to max. size of union: “\(k\)-coverage”
2. cover all elements with least number of sets: “set cover”
3. cover \((1-\lambda)m\) elements with least number of sets: “set cover with outliers”

**Results [BateniEsfandiariMirrokni’16]**:
- **Thm 1**: A \((1\pm \epsilon)\)-approx sketch of coverage function May NOT Help
  - Given an \((1\pm \epsilon)\)-approx oracle to coverage function, we get \(n^{0.49}\) approximation
- **Thm 2**: With some tricks, MinHash-based sketch + proper sampling WORKS
  - Sample elements not sets (different from previous coreset idea)
  - Correlation between samples (MinHash)
  - Cap degrees of elements in the sketch (reduces memory footprint)
Results for coverage problems

From [BEM’15]

- Space indep. of ground set
- Approx ~ best known offline
- “Edge” vs “set” arrival

Previous work:

- [14]=[CW’15]
- [22]=[DIMV’14]
- [24]=[ER’14]
- [31]=[IMV’15]
- [49]=[SG’09]

<table>
<thead>
<tr>
<th>Problem</th>
<th>Credit</th>
<th># passes</th>
<th>Approximation</th>
<th>Space</th>
<th>Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-cover</td>
<td>[49]</td>
<td>1</td>
<td>$1/4$</td>
<td>$O(m)$</td>
<td>set</td>
</tr>
<tr>
<td>k-cover</td>
<td>Here</td>
<td>1</td>
<td>$1 - 1/e - \varepsilon$</td>
<td>$\tilde{O}(n)$</td>
<td>edge</td>
</tr>
<tr>
<td>Set cover w outliers</td>
<td>[24, 14]</td>
<td>$p$</td>
<td>$O(\min(n^{-1/2}, e^{-1}p))$</td>
<td>$\tilde{O}(m)$</td>
<td>set</td>
</tr>
<tr>
<td>Set cover w outliers</td>
<td>Here</td>
<td>1</td>
<td>$(1 + \varepsilon) \log \frac{1}{\varepsilon}$</td>
<td>$\tilde{O}(\lambda(n))$</td>
<td>edge</td>
</tr>
<tr>
<td>Set cover</td>
<td>[14, 49]</td>
<td>$p$</td>
<td>$(p + 1)n^{1/2}$</td>
<td>$\tilde{O}(m)$</td>
<td>set</td>
</tr>
<tr>
<td>Set cover</td>
<td>[22]</td>
<td>$4^k$</td>
<td>$4^k \log n$</td>
<td>$\tilde{O}(nm^{1/k})$</td>
<td>set</td>
</tr>
<tr>
<td>Set cover</td>
<td>[31]</td>
<td>$p$</td>
<td>$O(p \log n)$</td>
<td>$\tilde{O}(nm^{O(\frac{1}{k})})$</td>
<td>set</td>
</tr>
<tr>
<td>Set cover</td>
<td>Here</td>
<td>$p$</td>
<td>$(1 + \varepsilon) \log n$</td>
<td>$\tilde{O}(nm^{O(\frac{1}{k})} + m)$</td>
<td>edge</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>Credit</th>
<th># rounds</th>
<th>Approximation</th>
<th>Load per machine</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-cover</td>
<td>[39]</td>
<td>$O\left(\frac{1}{\varepsilon^2} \log m\right)$</td>
<td>$1 - \frac{1}{\varepsilon} - \varepsilon$</td>
<td>$O(mkn^2)$</td>
<td>submodular functions</td>
</tr>
<tr>
<td>k-cover</td>
<td>[42]</td>
<td>2</td>
<td>0.54</td>
<td>$\max(mk^2, mn/k)$</td>
<td>submodular functions</td>
</tr>
<tr>
<td>k-cover</td>
<td>[19]</td>
<td>$\frac{1}{\varepsilon}$</td>
<td>$1 - \frac{1}{\varepsilon} - \varepsilon$</td>
<td>$\tilde{O}(n + m)$</td>
<td>-</td>
</tr>
<tr>
<td>k-cover</td>
<td>Here</td>
<td>3</td>
<td>$1 - \frac{1}{\varepsilon}$</td>
<td>$\tilde{O}(n + m)$</td>
<td>-</td>
</tr>
<tr>
<td>Set cover w outliers</td>
<td>Here</td>
<td>3</td>
<td>$(1 + \varepsilon) \log \frac{1}{\varepsilon}$</td>
<td>$\tilde{O}(n + m)$</td>
<td>-</td>
</tr>
<tr>
<td>Set cover</td>
<td>[43]</td>
<td>log(nm)</td>
<td>$(1 + \varepsilon) \log n$</td>
<td>$\Omega(mn^{1-\varepsilon})$</td>
<td>Submodular cover</td>
</tr>
<tr>
<td>Set cover</td>
<td>Here</td>
<td>$r$</td>
<td>$(1 + \varepsilon) \log n$</td>
<td>$\tilde{O}(nm^{O(\frac{1}{k})} + m)$</td>
<td>-</td>
</tr>
</tbody>
</table>
Summary: Distributed Algorithms for Five Problems

Defined on a metric space & composable core-sets apply.

1. Diversity Maximization, PODS’14 by IndykMahdianMahabadiM.
2. Capacitated $\ell_p$ Clustering, NIPS’14 by BateniBhaskaraLattanziM.

Beyond Metric Spaces. Only Randomized partitioning apply.

3. Submodular Maximization, STOC’15 by M.Zadimoghaddam
4. Feature Selection (Column Subset Selection), ICML’16 by Alschulter et al.

Needs adaptive sampling/sketching techniques

5. Coverage Problems: by BateniEsfandiarIM → Next year :) Thanks!