Randomized Composable Core-sets for Distributed Optimization

Vahab Mirrokni
Google Research, New York

Based on the following papers:

1) **Diversity Maximization @PODS'14: w/ Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian**

2) Balanced Clustering @NIPS'14: w/ Hossein Bateni, Aditya Bhaskara, Silvio Lattanzi

3) Submodular Maximization @STOC'15: w/ Morteza ZadiMoghaddam

Google NYC Large-scale Graph Mining

1. Algorithms/Tools: Ranking, Pairwise Similarity, Graph Clustering, Balanced Partitioning, Embedding…
   - Aim for scale - Solve for XXXB edges
2. Help product groups use our tools e.g.,
   - Ads, Search, Social, YouTube, Maps.
3. Compare MR+DHT, Flume, Pregel, ASYMP:
   - Compare for fault-tolerance and scalability
   - Public/private real data, synthetic data
4. Algorithmic Research:
   - Combined system/algorithms research
   - Streaming & local algorithms
   - Distributed Optimization e.g. core-sets

Outline of this Talk

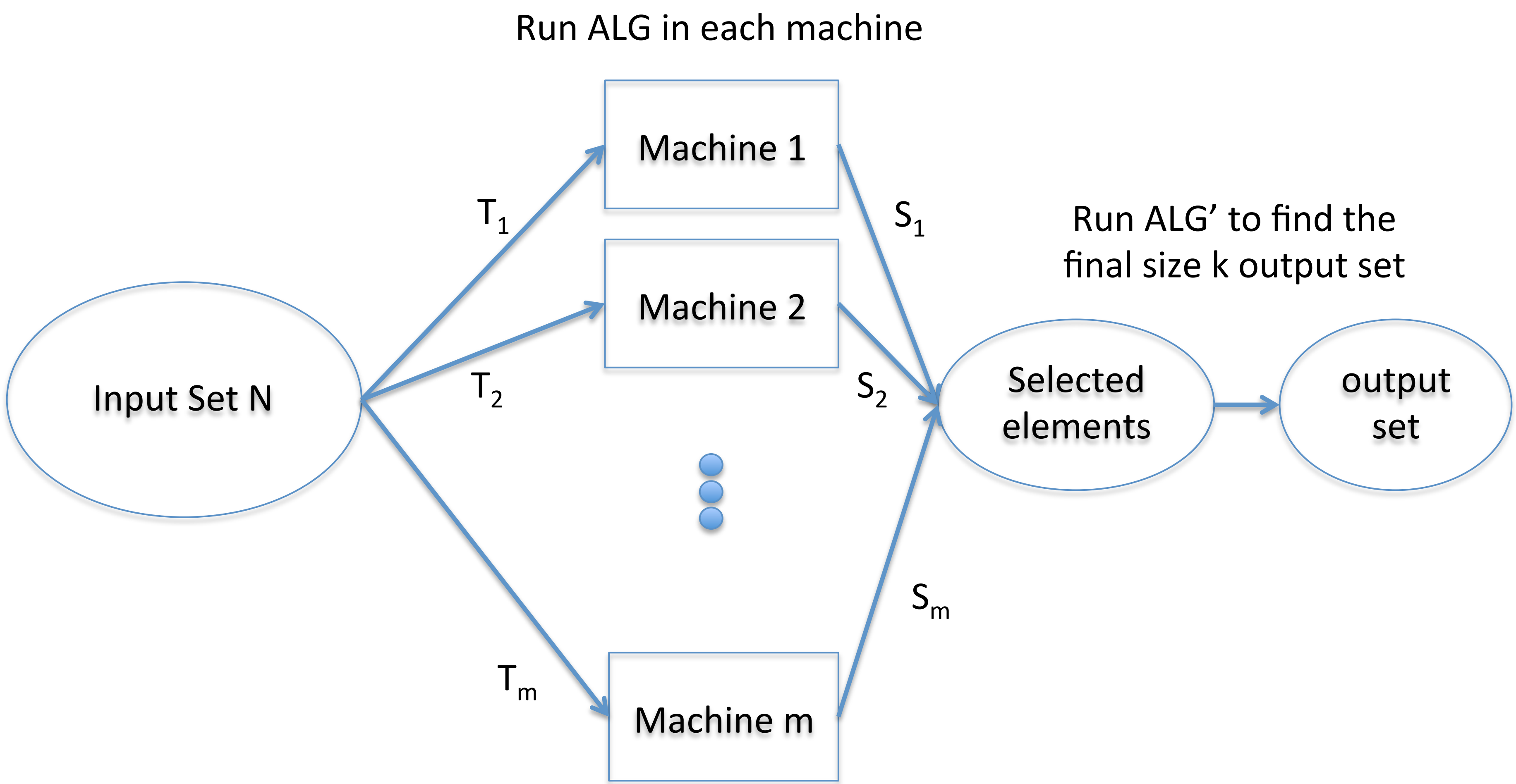
- **Composable Core-sets are useful**
  - Diversity Maximization: Composable Core-sets
  - Clustering Problems: Mapping Core-set
  - Submodular/Coverage Maximization: Randomized Composable Core-sets

- **Large-scale Graph Mining**
  - Modern Graph Algorithms Frameworks:
    - E.g. Connected Components in MR and MR+DHT
    - ASYMP: ASYnchronous Message Passing
  - Problems inspired by specific Applications
    - E.g. Algorithms for public-private graphs

Processing Big Data

- Extract and process a compact representation of data. Examples:
  - Sampling: focus only on a small subset of data
  - Sketching: compute a small summary of data, e.g. mean, variance, …
  - Mergeable Summaries: if multiple summaries can be merged while preserving accuracy [Agarwal et al. 2012].
- Composable core-sets [Indyk et al. 2014]

Distributed Optimization Framework

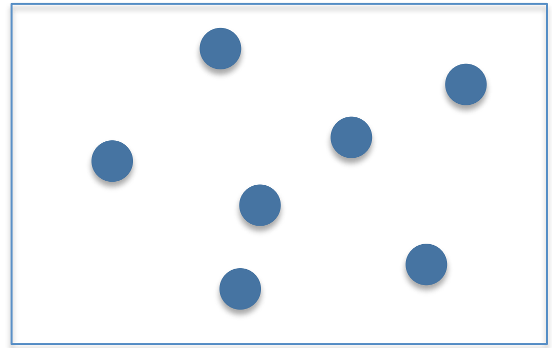


Executive Summary: Composable Core-sets

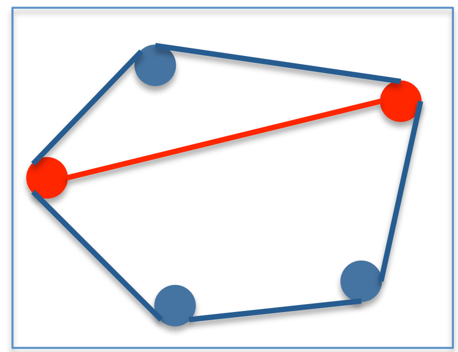
- **Technique for effective distributed algorithm**
  - **One or Two rounds of Computation**
  - **Minimal Communication Complexity**

- **Problems**
  - Diversity Maximization
    - Composable Core-sets
  - Clustering Problems
    - Mapping Core-sets
  - Submodular/Coverage Maximization:
    - Randomized Composable Core-sets

Core-sets



Input: A set of points P
Goal: Optimize some function f
For instance find the **farthest** distance pair of points



Core-set: A subset of points that preserves the optimal solution
For instance Convex hull is a 1-core-set because the farthest pair of points are in the convex hull

In general, we are looking for a **small α-**core-set S,
in other words, a small S with the guarantee f(S) ≥ **α** f(P)

Composable Core-sets

- Partition input into several parts $T_1, T_2, \ldots, T_m$
- In each part, select a subset $S_i \subseteq T_i$
- Take the union of selected sets: $S = S_1 \cup S_2 \cup \ldots \cup S_m$
- Solve the problem on S
- Evaluation: We want set S to represent the original big input well, and preserve the optimum solution approximately.

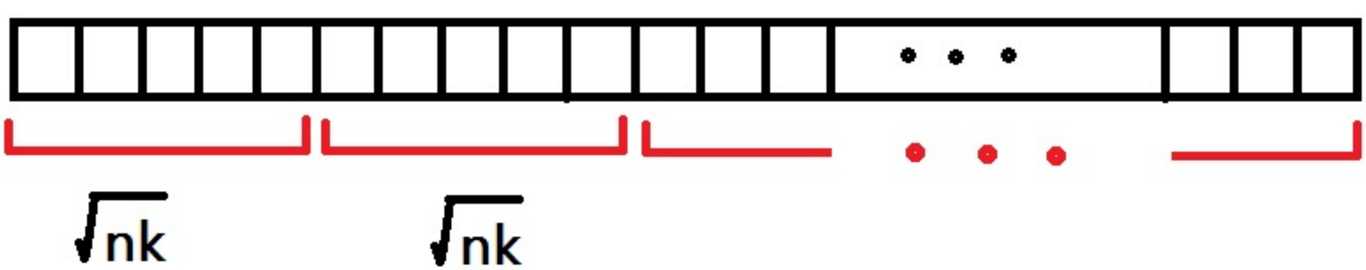
Formal Definition of Composable Core-sets

- Define $f_k(S) \stackrel{\text{def}}{=} \max_{S' \subseteq S, |S'| \leq k} f(S')$, e.g. $f_k(N)$ is the value of the optimum solution.
- ALG(T) is the output of algorithm ALG on input set T. Suppose |ALG(T)| is at most k.
- ALG is α-approximate composable core-set iff for any collection of sets $T_1, T_2, \ldots, T_m$ we have

$$f_k(\mathrm{ALG}(T_1) \cup \ldots \cup \mathrm{ALG}(T_m)) \geq \alpha f_k(T_1 \cup \ldots \cup T_m)$$

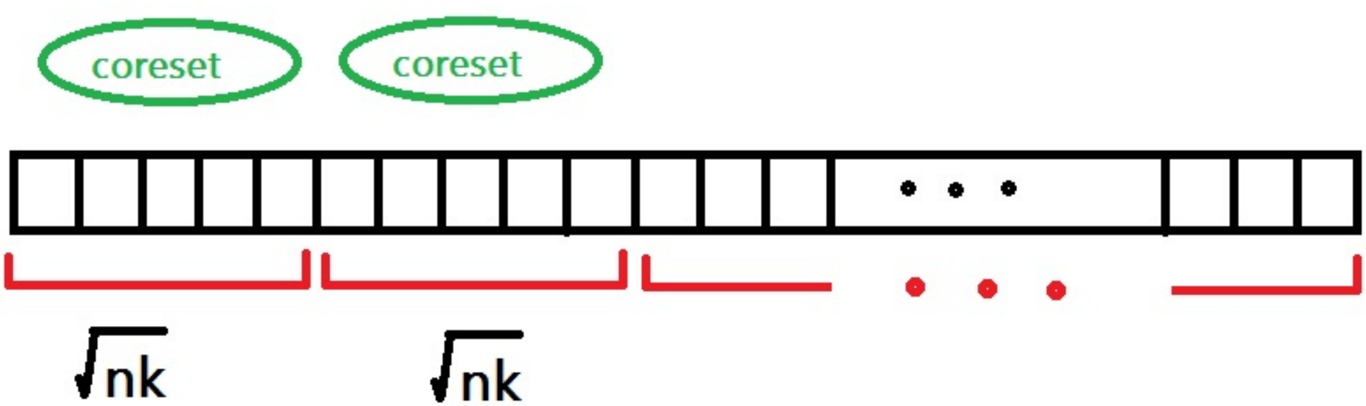
Applications – Streaming Computation

- **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 = $\sqrt{n/k}$



Applications – 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: $k\sqrt{n/k} + \sqrt{nk} = O(\sqrt{nk})$

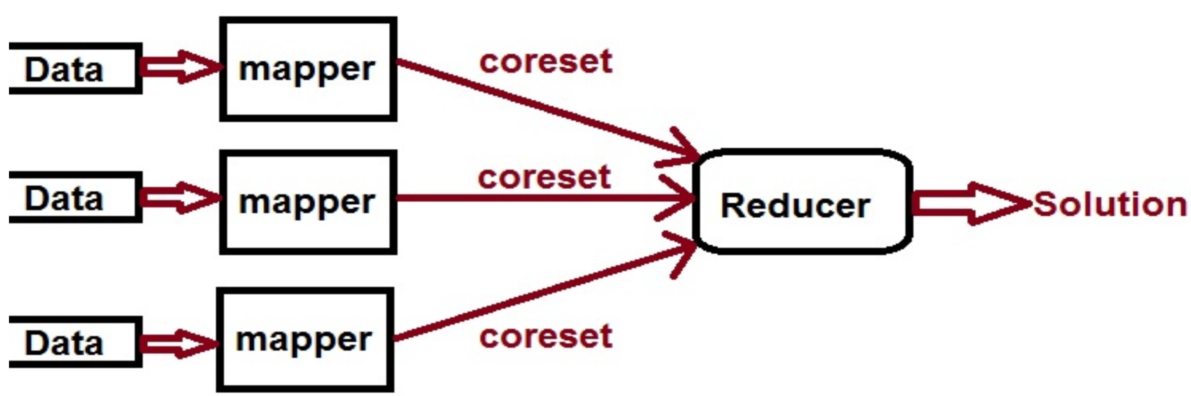


Applications – Distributed Systems

- **Streaming Computation**
- **Distributed System:**
  - Each machine holds a block of data.
  - A composable core-set is computed and sent to the server

Applications – Distributed Systems

- 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



Problems considered

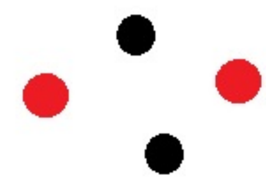
- **Diversity Maximization**: Find a set *S* of *k* points and maximize the sum of pairwise distances i.e. *diversity(S)*.
- **Capacitated/Balanced Clustering**: Find a set S of *k* centers and cluster nodes around them while minimizing the sum of distances to *S*.
- **Coverage/submodular Maximization**: Find a set *S* of *k* items & maximize *f(S)*.

Diversity Maximization Problem

- Given: *n* points in a metric space
- Find a set *S* of *k* points
- Goal:

maximize *diversity(S)* i.e.

diversity(S) = sum of pairwise distances of points in *S*.

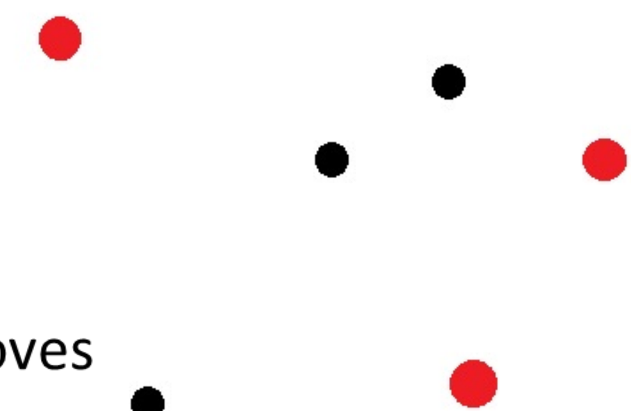


k=4
n = 6

- Background: Max Dispersion
  - Halldorson et al studied 7 variants
  - Recently studied by Borodin et al, Abbassi et al'13.

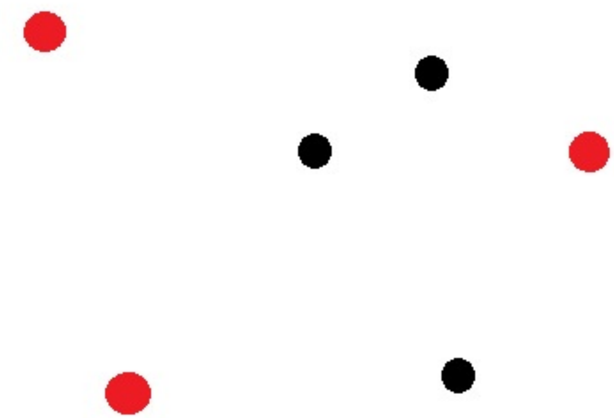
Local Search for Diversity Maximization (KDD'13)

- Used for sum of pairwise distances
- Algorithm [Abbasi, Mirrokni, 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: $\log_{\left\{1+\frac{\epsilon}{n}\right\}} k^2 = O(\frac{n}{\epsilon} \log k)$
  - Approximation factor is constant.



Local Search for Diversity Maximization (KDD'13)

- Used for sum of pairwise distances
- Algorithm [Abbasi, Mirrokni, Thakur]
  - Initialize S with an arbitrary set of k points which contains the two farthest points
  - While there exists a swap that impro diversity by a factor of $\left(1+\frac{\epsilon}{n}\right)$
    - » Perform the swap



- For Remote-Clique
  - Number of rounds: $\log_{\{1+\frac{\epsilon}{n}\}} k^2 = O(\frac{n}{\epsilon}\log k)$
  - Approximation factor is constant.

Composable Core-sets for Diversity Maximization

- Theorem(IndykMahabadiMahdianM.’14): A local search algorithm computes a *constant-factor* composable core-set for maximizing *sum of pairwise distances.*

- Thm(IMMM’14): Greedy Algorithm Computes a 3-composable core-set for maximizing the minimum pairwise distance.

Proof Idea

Let $P_1, \cdots, P_m$ be the set of points , $P = \cup P_i$

$S_1, \cdots, S_m$ be their core-sets, $S = \cup S_i$

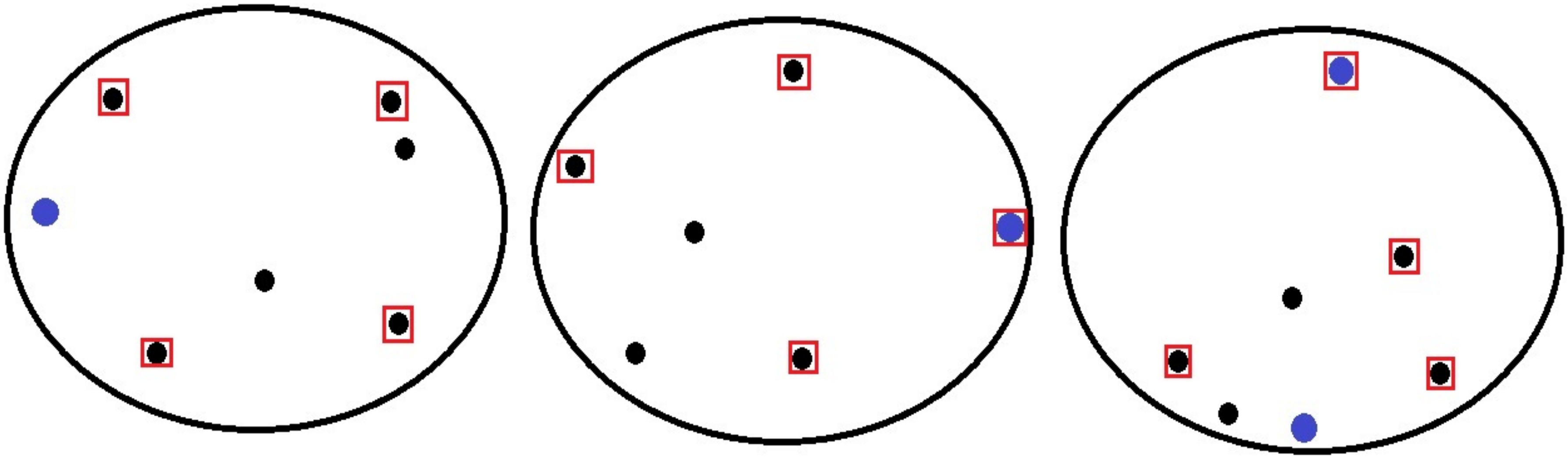
Let $OPT = \{o_1, \cdots, o_k\}$ be the optimal solution

Let r be their maximum diversity , $r = \max_i div(S_i)$,

Goal: $div_k(S) \geq div_k(P)$ / c

Goal: $div_k(S) \geq div(OPT)$ / c

Note: $\mathrm{div_k}(S) \geq r$



Proof Idea

Let $P_1, \cdots, P_m$ be the set of points , $P = \cup P_i$

$S_1, \cdots, S_m$ be their core-sets, $S = \cup S_i$

Let $OPT = \{o_1, \cdots, o_k\}$ be the optimal solution

Let r be their maximum diversity , $r = \max_i div(S_i)$,

Goal: $div_k(S) \geq div_k(P)$ / c

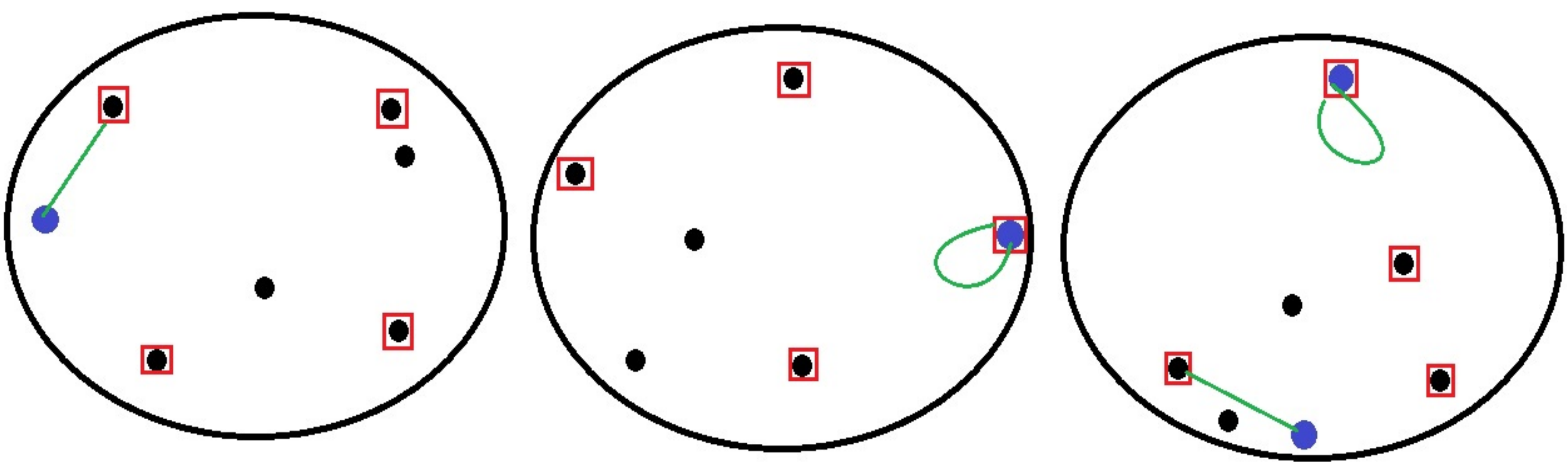
Goal: $div_k(S) \geq div(OPT)$ / c

Note: $\mathrm{div}_k(S) \geq r$

Case 1: one of S_i has diversity as good as the optimum: $r \geq \boldsymbol{O}\big(div(OPT)\big)$

Case 2: : $r \leq \boldsymbol{O}(div(OPT))$

- find a **one-to-one** mapping μ from $OPT = \{o_1, \cdots, o_k\}$ to $S = S_1 \cup \cdots \cup S_m$ s.t.
  $$dist(o_i, \mu(o_i)) \leq \boldsymbol{O}(r)$$
- Replacing o_i with $\mu(o_i)$ has still large diversity
- $div(\{\mu(o_i)\})$ is approximately as good as $div(\{o_i\})$



Proof Idea

Let $P_1, \cdots, P_m$ be the set of points , $P = \cup P_i$

$S_1, \cdots, S_m$ be their core-sets, $S = \cup S_i$

Let $OPT = \{o_1, \cdots, o_k\}$ be the optimal solution

Let r be their maximum diversity , $r = \max_i div(S_i)$,

Goal: $div_k(S) \geq div_k(P)$ / c

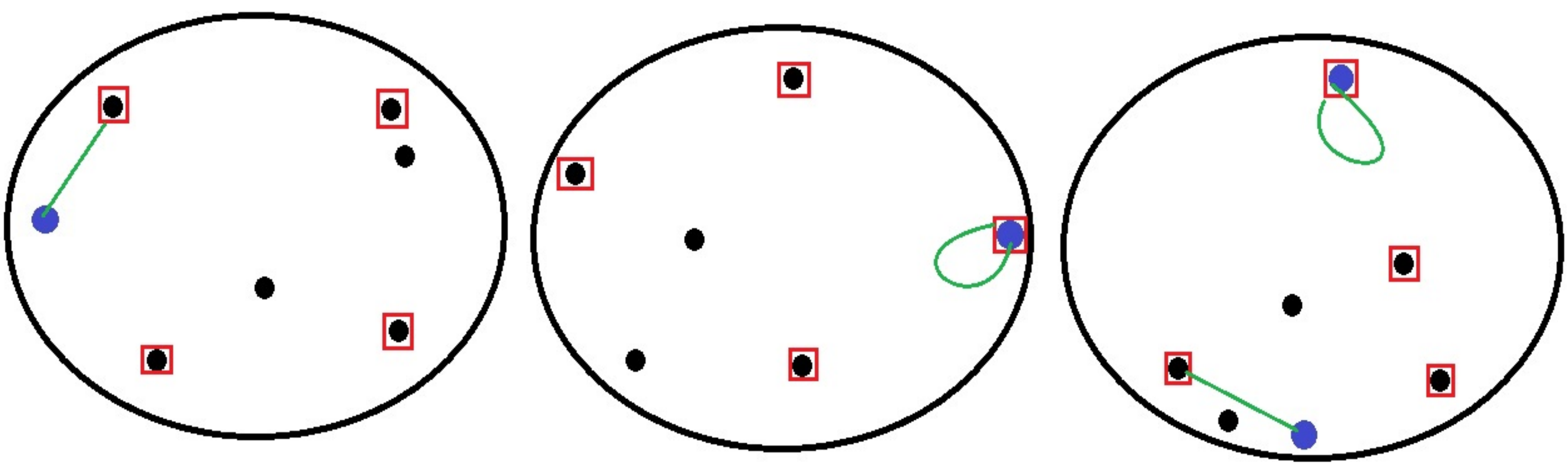
Goal: $div_k(S) \geq div(OPT)$ / c

Note: $\text{div}_\text{k}(S) \geq r$

Case 1: one of S_i has diversity as good as the optimum: $r \geq \boldsymbol{O}\big(div(OPT)\big)$

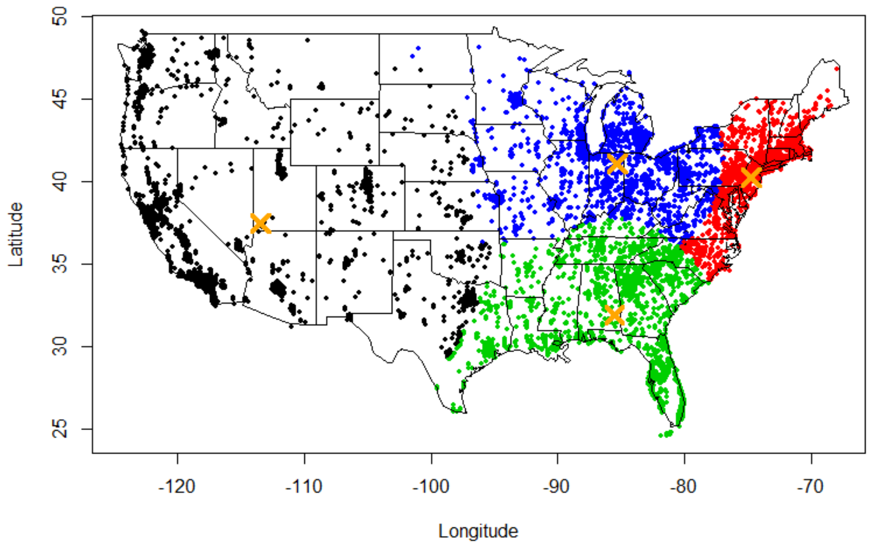
Case 2: : $r \leq \boldsymbol{O}(div(OPT))$

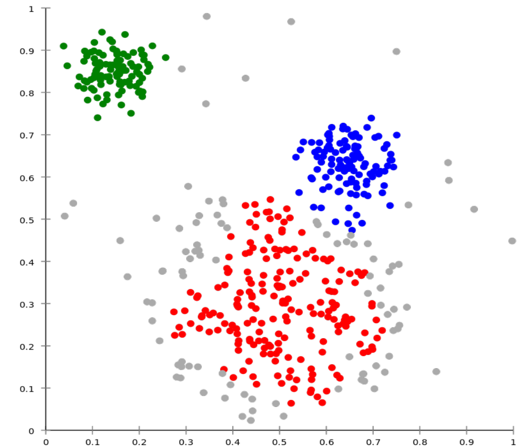
- find a **one-to-one** mapping μ from $OPT = \{o_1, \cdots, o_k\}$ to $S = S_1 \cup \cdots \cup S_m$ s.t.
$$dist(o_i, \mu(o_i)) \leq \boldsymbol{O}(r)$$
- Replacing o_i with $\mu(o_i)$ has still large diversity
- $div(\{\mu(o_i)\})$ is approximately as good as $div(\{o_i\})$



Distributed Clustering

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 : $$\sum_i \sum_{u \in S_i} d(u, c_i)$$

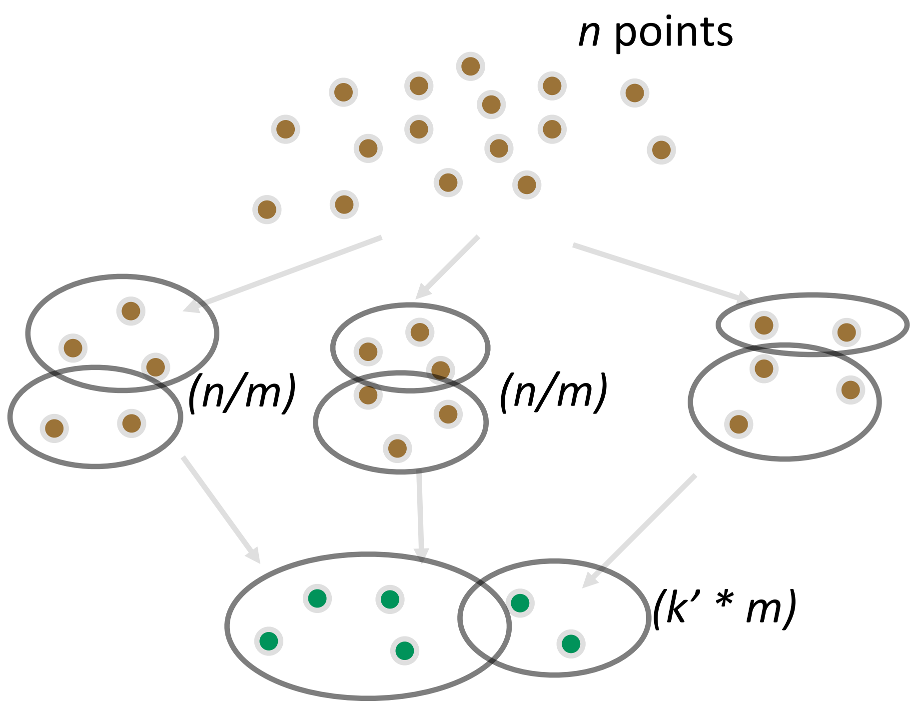
Metric space *(d, X)*

α-approximation algorithm: cost less than α*OPT

Clustering via Composable Core-sets

Goal: Find k clusters (and centers) to minimize objective

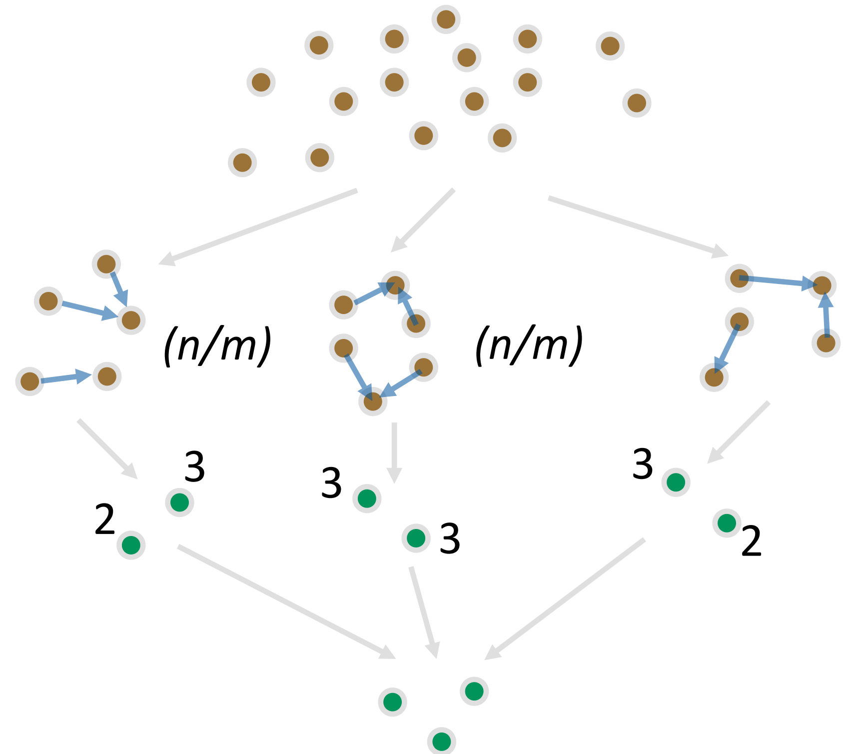
1. partition points into m machines
2. solve on machines separately
3. cluster the centers obtained ($k' * m$)
4. assign points to closest chosen centers



Mapping Core-sets Framework

- **How can we ensure cluster sizes are bounded?**

1. partition points into *m* machines
2. "map" points in machine to a small #points (*k′*)
3. create a "multi-set" instance
4. solve multi-set instance *efficiently*



Balanced/Capacitated Clustering

Theorem(BhaskaraBateniLattanziM. NIPS'14): distributed balanced clustering with

- approx. ratio: (small constant) * (best "single machine" ratio)
- rounds of MapReduce: constant (2)
- memory: ~*(n/m)^2* with *m* machines

Works for all L_p objectives.. (includes k-means, k-median, k-center)

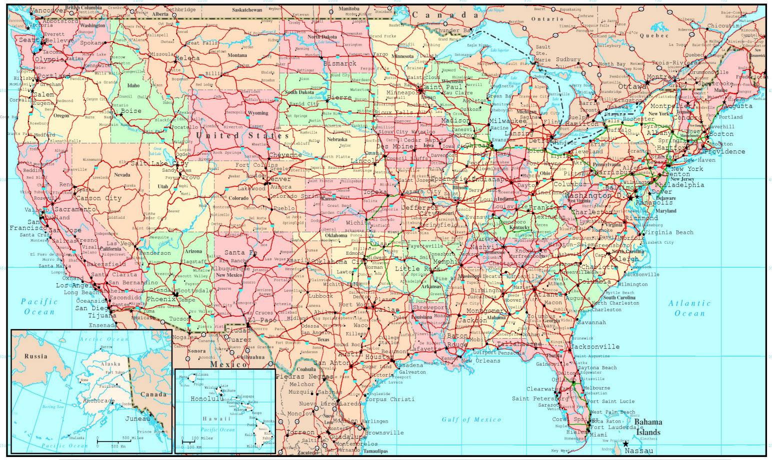
Improving Previous Work

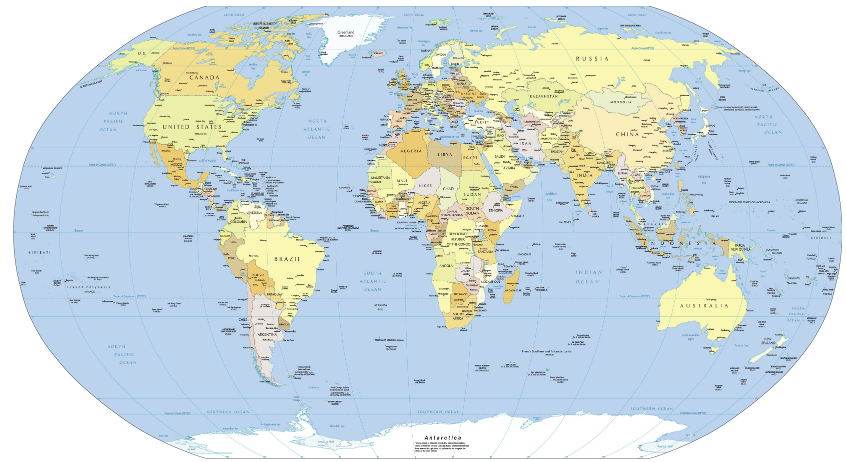
- Bahmani, Kumar, Vassilivitskii, Vattani: Parallel K-means++
- Balcan, Enrich, Liang: Core-sets for k-median and k-center

Experiments

Aim: Test algorithm in terms of (a) scalability, and (b) quality of solution obtained

Setup: Two "base" instances and subsamples (used *k*=1000, #machines = 200)

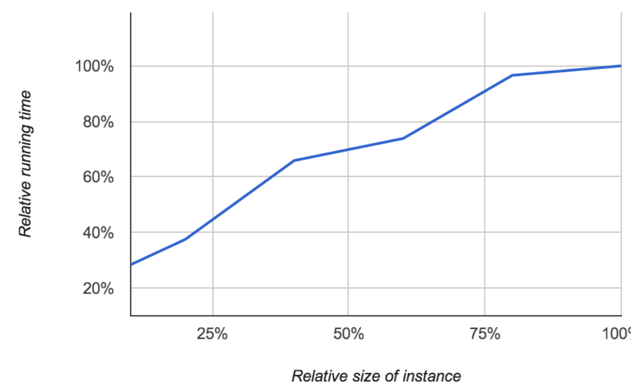




US graph: N = x0 Million
distances: geodesic

World graph: N = x00 Million
distances: geodesic

	size of seq. inst.	increase in OPT
US	1/300	**1.52**
World	1/1000	**1.58**



Accuracy: analysis pessimistic

Scaling: sub-linear

Submodular Functions

- A non-negative set function f defined on subsets of a ground set N, i.e. f: $2^N \longrightarrow R^+ \cup \{0\}$
- f is submodular iff for any two subsets A and B
  - $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$
- Alternative definition: f is submodular iff for any two subsets $A \subseteq B$, and element x:
  - $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$

Coverage/Submodular Maximization

Submodular Maximization:

- Given: k and a submodular function f
- Goal: Find a set S of k elements & maximize $f(S)$.

Max-Coverage (special case):

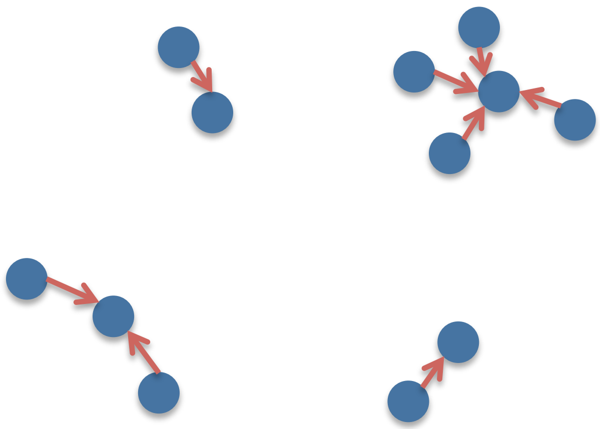
- Given: : k & family of subsets $V_1 \dots V_n$
- Goal: Choose k subsets $V'_1 \dots V'_k$ with the maximum cardinality of union.

Submodular Maximization: Applications

- Many applications for maximizing coverage: Data summarization, data clustering, column selection, diversity maximization in search.
- Machine Learning Applications: Exemplar based clustering, active set selections, graph cuts and others in [Mirzasoleiman, Karbasi, Sarkar, Krause NIPS'13]

Application e.g. Exemplar Sampling

k-median-cost(S) = sum of distances of points
to their closest centers in S



f(S) = k-median-cost(empty set) – k-median-cost(S)
f is a submodular function
Instead of minimizing median cost, maximize f

Bad News!

- Theorem[IndykMahabadiMahdianM PODS'14] There exists no better than $\frac{\log k}{\sqrt{k}}$ approximate composable core-set for submodular maximization.

Submodular Maximization: Related Work

Submodular/coverage maximization in MapReduce:

- ChierchettiKumarTomkins'09: Polylog #rounds
- CoromodeKarloffWirth'10: Better communication in *poly log* # rounds
- Belloch et al'13: $log^2\ n$ #rounds
- KumarMoselyVassilivitskiiVattani (SPAA'13): *log* #rounds or constant #rounds with *log* communication overhead
- Mirzasoleiman, Karbasi, Sarkar, Kraus, NIPS'13: Greedy algorithm works in two rounds (for special submodular functions)

Q: is it possible to solve this in one or two rounds of MapReduce without space/communication overhead?

- IMMM'14 shows that it's not doable via core-sets.

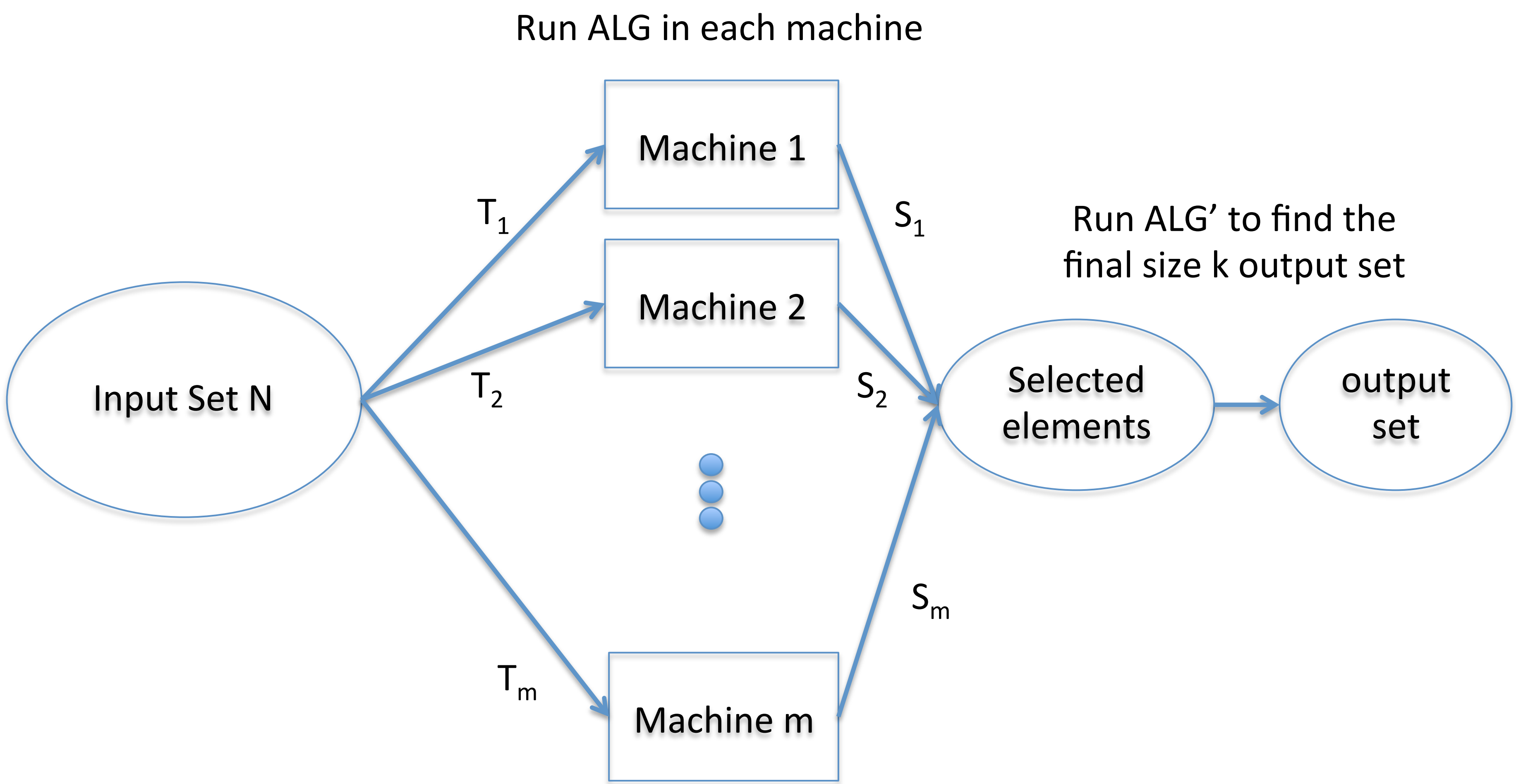
Randomization comes to rescue

- Instead of working with worst case partitioning to sets T_1, T_2, ..., T_m, suppose we have a random partitioning of the input.
- We say alg is α-approximate **randomized** composable core-set iff

$$\mathbb{E}\left[f_k(\mathsf{ALG}(T_1) \cup \ldots \cup \mathsf{ALG}(T_m))\right] \geq \alpha \cdot \mathbb{E}\left[f_k(T_1 \cup \ldots \cup T_m)\right]$$

where the expectation is taken over the random choice of $\{T_1, T_2, \ldots, T_m\}$

General Framework



Good news!
[M. ZadiMoghaddam – STOC'15]

- Theorem [M., ZadiMoghaddam]: There exists a class of O(1)-approximate randomized composable core-sets for monotone and non-monotone submodular maximization.
- In particular, algorithm Greedy is 1/3-approximate randomized core-set for monotone f, and (1/3-1/3m)-approximate for non-monotone f.

Family of β-nice algorithms

- ALG is β-nice if for any set T
  and element $x \in T \setminus ALG(T)$ we have:
  - $ALG(T) = ALG(T\setminus\{x\})$
  - $\Delta(x, ALG(T))$ is at most $\beta f(ALG(T))/k$

  where $\Delta(x, A)$ is the marginal value of adding x to set A, i.e. $\Delta(x, A) = f(A\cup\{x\})-f(A)$
- Theorem: A β-nice algorithm is $(1/(2+\beta))$-approx randomized composable core-sets for monotone f and $((1-1/m)/(2+\beta))$-approx for non-monotone.

Greedy Algorithm

- Given input set T, Greedy returns a size k output set S as follows:
  - Start with an empty set

  For k iterations, find an item $x \in T$ with maximum marginal value to S, $\Delta(x, S)$, and add x to S.
- Remark: Greedy is a 1-nice algorithm.
- In the rest, we analyze algorithm Greedy for a monotone submodular function f.

Analysis

- Let OPT be the subset of size k with maximum value of f.
- Let OPT’ be OPT $\cap$ ($S_1 \cup S_2 \ldots \cup S_m$), and OPT’’ be OPT\OPT’
- We prove that

  $E[\max\{f(OPT'), f(S_1), f(S_2), \ldots, f(S_m)\}] \geq f(OPT)/3$

Linearizing marginal contributions of elements in OPT

- Consider an arbitrary permutation π on elements of OPT
- For each $x \in$ OPT, define OPT^x to be elements of OPT that appear before x in π
- By definition of Δ values, we have:

$$f(OPT) = \sum_{x \in OPT} \Delta(x, OPT^x)$$

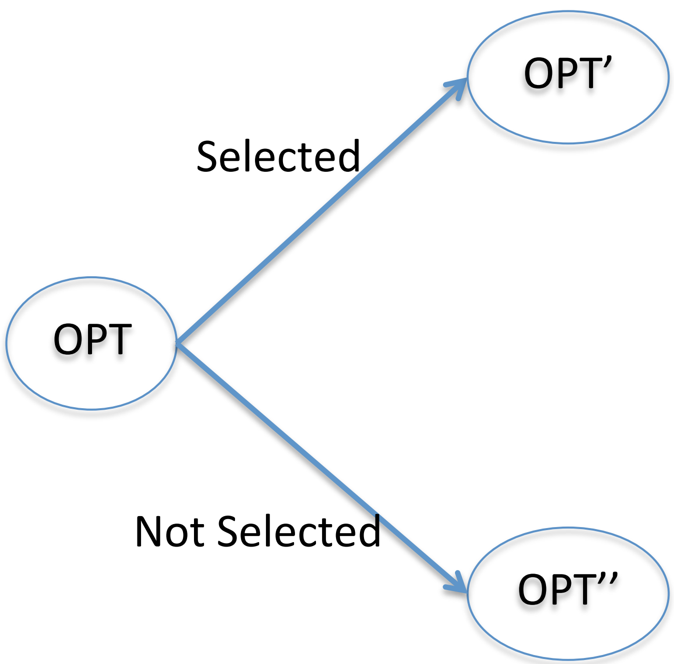
Lower bounding $f(OPT^S)$

- $f(OPT')$ is $\sum_{x \in OPT'} \Delta(x, OPT^x \cap OPT')$
- Using submodularity, we have:

  $\Delta(x, OPT^x \cap OPT') \geq \Delta(x, OPT^x)$
- Therefore: $f(OPT') \geq \sum_{x \in OPT'} \Delta(x, OPT^x)$
- It suffices to upper bound $\sum_{x \in OPT''} \Delta(x, OPT^x)$

Proof Scheme

Goal: Lower bound $\max\{f(OPT'), f(S_1), f(S_2), \ldots, f(S_m)\}$



$\geq \sum_{x \in OPT'} \Delta(x, OPT^x)$

$f(OPT) = \sum_{x \in OPT} \Delta(x, OPT^x)$

Suffices to upper bound $\sum_{x \in OPT''} \Delta(x, OPT^x)$

For each x in $T_i \cap OPT''$: $\Delta(x, S_i) \leq f(S_i)/k$

$\sum_{1 \leq i \leq m} \sum_{x \text{ in } OPT'' \cap Ti} \Delta(x, S_i) \leq \max_i \{f(S_i)\}$

How large can $\Delta(x, OPT^x) - \Delta(x, S_i)$ be?

Upper bounding Δ reductions

$$\Delta(x, OPT^x) - \Delta(x, S_i) \quad \leq \quad \Delta(x, OPT^x) - \Delta(x, OPT^x \cup S_i)$$

$$\sum_{x \text{ in } OPT} \Delta(x, OPT^x) - \Delta(x, OPT^x \cup S_i) = f(OPT) - (f(OPT \cup S_i) - f(S_i)) \leq f(S_i)$$

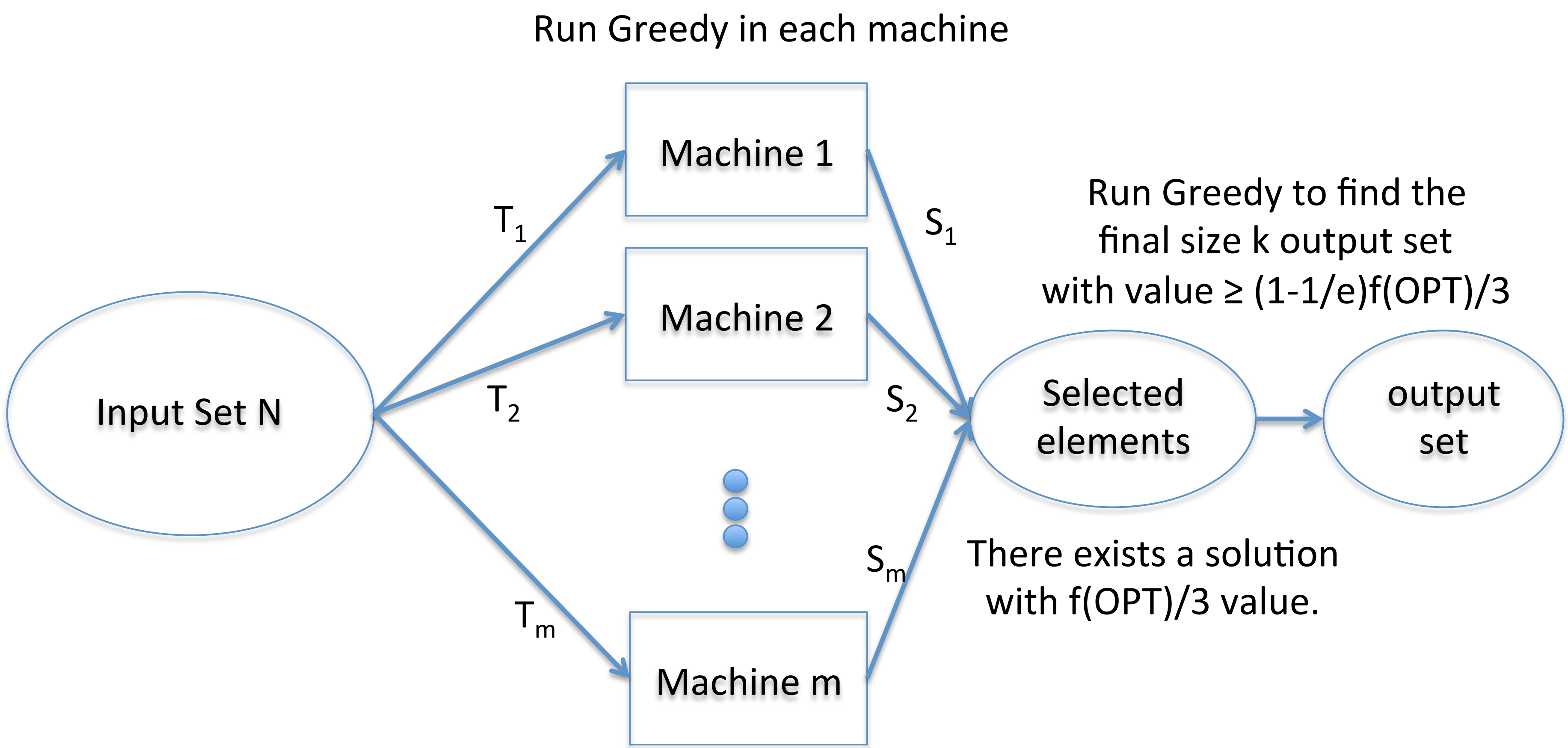
in worst case: $\sum_{1 \leq i \leq m} \sum_{x \text{ in } OPT'' \cap Ti} \Delta(x, OPT^x) - \Delta(x, S_i) \leq \sum_{1 \leq i \leq m} f(S_i)$

in expectation: $\sum_{1 \leq i \leq m} \sum_{x \text{ in } OPT'' \cap Ti} \Delta(x, OPT^x) - \Delta(x, S_i) \leq \sum_{1 \leq i \leq m} f(S_i)/m$

Conclusion: $E[f(OPT')] \geq f(OPT) - \max_i \{f(S_i)\} - \text{Average}_i \{f(S_i)\}$

Greedy is a 1/3-approximate randomized core-set

Distributed Approximation Factor



Take the maximum of $\max_i \{f(S_i)\}$ and Greedy($S_1 \cup S_2 \cup \ldots \cup S_m$) to achieve 0.27 approximation factor

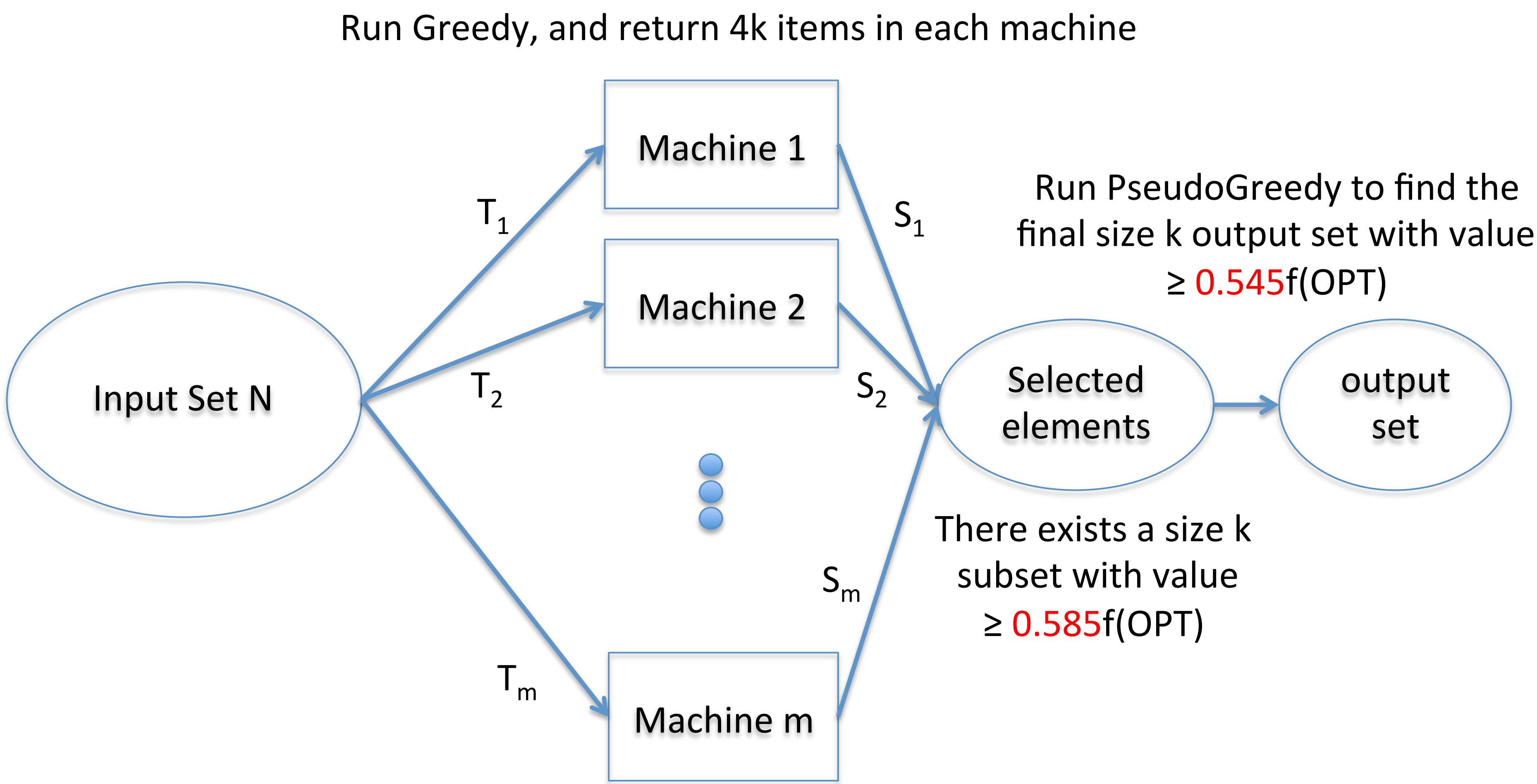
Improving Approximation Factors for Monotone Submodular Functions?

- Hardness Result [M, ZadiMoghaddam]: With output sizes ($|S_i|$) $\leq k$, Greedy, and locally optimum algorithms are not better than ½ approximate randomized core-sets.
- Can we increase the output sizes and get better results?

Summary of Results [M. ZadiMoghaddam – STOC'15]

1. A class of 0.33-approximate randomized composable core-sets of size k for non-monotone submodular maximization.
2. Hard to go beyond ½ approximation with size k. Impossible to get better than 1-1/e.
3. 0.58-approximate randomized composable core-set of size 4k for monotone f. Results in 0.54-approximate distributed algorithm.
4. For small-size composable core-sets of k' less than k: *sqrt{k'/k}*-approximate randomized composable core-set.

Improved Distributed Approximation Factor



$(2-\sqrt{2})$-approximate Randomized Core-set

- Positive Result [M, ZadiMoghaddam]: If we increase the output sizes to be 4k, Greedy will be $(2-\sqrt{2})-o(1) \geq 0.585$-approximate randomized core-set for a monotone submodular function.
- Remark: In this result, we send each item to C random machines instead of one. As a result, the approximation factors are reduced by a $O(\ln(C)/C)$ term.

Algorithm PseudoGreedy

- Forall $1 \le K_2 \le k$
  - Set $K' := K_2/4$
  - Set $K_1 := k - K_2$
  - Partition the first $8K'$ items of S_1 into sets $\{A_1, \ldots, A_8\}$
  - For each $L \subseteq \{1, \ldots, 8\}$
    - Let S' be union of A_i where i is in L
    - Among selected items, insert $K_1 + (4 - |L|)K'$ items to S' greedily
  - If $(f(S') > f(S))$ then $S := S'$
- Return S

Small-size core-sets

- So far we have discussed core-sets of size k for problems with output size of k. What if k is too large and we need a core-set of size k' which is less than k?

- Problem: (Randomized) Composable core-sets for small-size core-sets for diversity and submodular maximization.

Small-size core-sets: Some results

- Problem: (Randomized) Composable core-sets for small-size core-sets for diversity and submodular maximization.

- Theorem (M.ZadiMoghaddam): There exists *a sqrt{k'/k}*-approximate randomized composable core-set for coverage and submodular maximization of size *k'*. For non-randomized core-sets there is a hardness result of *k'/k*.

Summary: Composable Core-sets

- Composable core-set framework
  - Divide data into *m* parts (at random)
  - Solve independently for each part
  - Combine solutions and solve on the union of these solutions
- Also works for ***streaming*** and nearest neighbor search
- Solves diversity maximization and Balanced clustering (k-center, k-median and k-means)
- Coverage and Submodular maximization
  - Impossible for non-randomized composable core-set but solved via randomized core-sets
- Apply to other ML & Graph algorithmic problems: Edges are partitioned into m parts or edges arrive in a stream (e.g. random order)
  - Maximum and Minimum and Weighted Matching Cut Problems
  - Correlation Clustering
  - ML problems: Subset column selection

Google NYC Large-scale Graph Mining

1. Algorithms/Tools: Ranking, Pairwise Similarity, Graph Clustering, Balanced Partitioning, Embedding…
   - Aim for scale - Solve for XXXB edges
2. Help product groups use our tools e.g.,
   - Ads, Search, Social, YouTube, Maps.
3. Compare MR+DHT, Flume, Pregel, ASYMP:
   - Compare for fault-tolerance and scalability
   - Public/private real data, synthetic data
4. Algorithmic Research:
   - Combined system/algorithms research
   - Streaming & local algorithms
   - Distributed Optimization e.g. core-sets

Examples of Research done’14 & ’15

Algorithms Research, e.g.

- MapReduce/Streaming Algorithmics: Minimize #rounds
  - Randomized core-sets for distributed computation …
- Local clustering beyond Cheeger’s Inequality (ICML’13)
- Reduce & Aggregate for Personalized Search @WWW’14
- Graph Alignment @VLDB’14
- Fast algorithms for Public/Private Graphs @KDD’15

Combined system + algorithms research:

- Algorithmic models for MR+DHT, ASYMP
- ASYMP: New graph mining framework
  - Based on “ASYnchronous Message Passing”
  - Compare with MR, Pregel
  - Study its fault-tolerance, and scalability

Graph Mining Frameworks

Applying various frameworks to graph algorithmic problems

- **Iterative MapReduce (Flume):**
  - More widely fault-tolerant available tool
  - Can be optimized with algorithmic tricks
- **Iterative. MapReduce + DHT Service (Flume):**
  - Better speed compared to MR
- **Pregel:**
  - Good for synch. computation w/ many rounds
- **ASYMP (ASYnchronous Message-Passing):**
  - More scalable/More efficient use of CPU
  - Asych. self-stabilizing algorithms

e.g. Connected Components

- Connected Components in MR & MR+DHT
  - Simple, local algorithms with O($\log^2 n$) round complexity
  - Communication efficient (#edges non-increasing)
- Use Distributed HashTable Service (DHT) to improve # rounds to O~(log n) [from ~20 to ~5]
- Data: Graphs with ~XT edges. Public data with 10B edges
- Results:
  - MapReduce: 10-20 times faster than Hash-to-Min
  - MR+DHT: 20-40 times faster than Hash-to-Min
  - ASYMP: A simple algorithm in ASYMP: 25-55 times faster than Hash-to-Min

KiverisLattanziM.RastogiVassilivitskii: SOCC'14:

ASYMP: Graph Processing via ASYnchronous Message Passing

- ASYMP: New graph mining framework
- Compare with MapReduce, Pregel
  - Computation does not happen in a synchronize number of rounds
  - Fault-tolerance implementation is also asynchronous
- More efficient use of CPU cycles
- We study its fault-tolerance and scalability
- Impressive performance: Simple implementations of connected component

Ongoing work joint with Fleury and Lattanzi

Algorithms for Public/Private Graphs

- Given: a public graph *$G(V, E)$*
- Each node v also has a set of private edges G_v not known to the rest of nodes
- Problem: Solve for each node v on G_v, e.g.
  - For each v, compute similar nodes to v in G_v : e.g, topK nodes based on #common neighbors or PPR
  - For each v, compute the cluster that v belongs to in G_v
- Goal: Solve the problem for G first. Then for each v, post-process in time proportional to $|G_v|$

KDD'15: Chierchetti-Epasto-Kumar-Lattanzi-M.

Concluding Remarks

- **Composable Core-sets are useful**
  - Diversity Maximization: Composable Core-sets
  - Clustering Problems: Mapping Core-set
  - Submodular/Coverage Maximization: Randomized Composable Core-sets
- **Large-scale Graph Mining**
  - Modern Graph Algorithms Frameworks:
    - E.g. Connected Components in MR and MR+DHT
    - ASYMP: Asynchronous Message Passing
  - Problems inspired by specific Applications
    - E.g. Algorithms for public-private graphs

Applications of composable core-sets

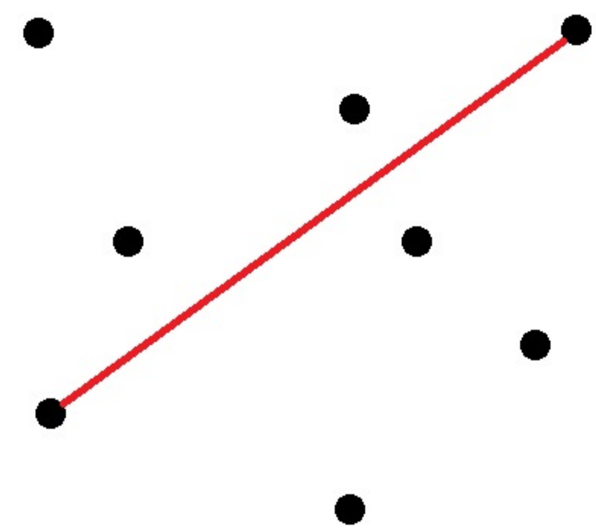
- Distributed Approximation:
  – Distribute input between m machines,
  – ALG selects set $S_i = ALG(T_i)$ in machine $1 \leq i \leq m$,
  – Gather the union of selected items, $S_1 \cup S_2 \cup \ldots \cup S_m$, on a single machine, and select k elements.
- Streaming Models: Partition the sequence of elements, and simulate the above procedure.
- A class of nearest neighbor search problems

Modern Distributed Algorithmics

- **Communication**
  - Can be the overwhelming cost
  - In practice constant factors matter a lot
- **Data Skew:**
  - Most datasets are heavily tailed
  - Naïve data distributions can be disastrous
  - In synchronous environments must wait for slowest shard: "The curse of reducer"
- **Algorithmic techniques:**
  - Embarasssingly parallel may still be slow
  - Techniques to minimize communication & skew

Core-Set Definition

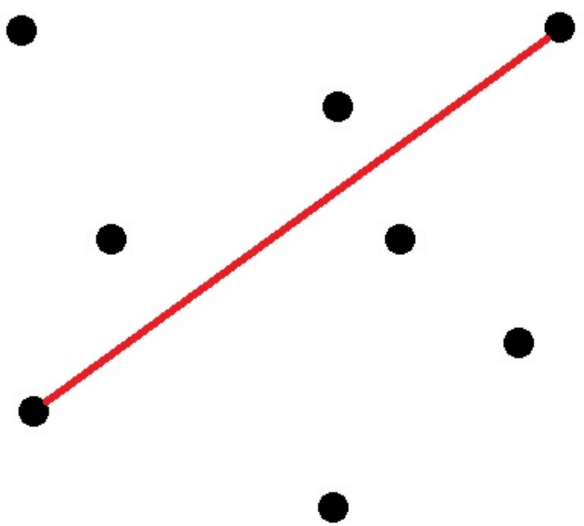
- **Setup**
  - Set of n points $\boldsymbol{P}$ in d-dimensional space
  - Optimize a function f



Core-Set Definition

- **Setup**
  - Set of n points $\boldsymbol{P}$ in d-dimensional space
  - Optimize a function f

- ***c*-Core-set:** Small subset of points $S \subset P$ which suffices to c-approximate the optimal solution
- Maximization: $\frac{f_{opt}(P)}{c} \leq f_{opt}(S) \leq f_{opt}(P)$

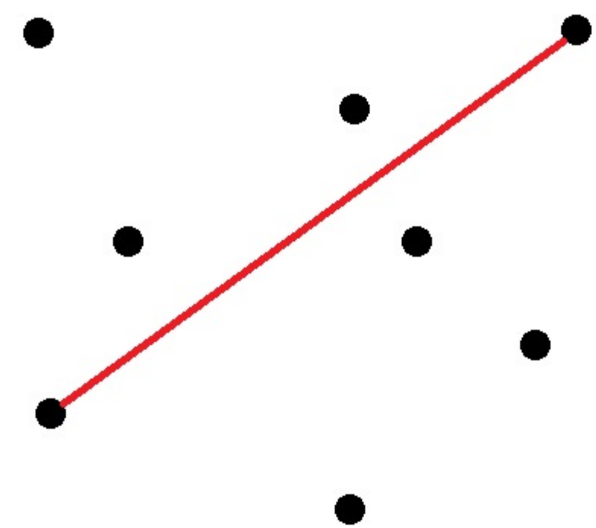


Core-Set Definition

- **Setup**
  - Set of n points $\boldsymbol{P}$ in d-dimensional space
  - Optimize a function f

- ***c*-Core-set:** Small subset of points $S \subset P$ which suffices to c-approximate the optimal solution
- Maximization: $\frac{f_{opt}(P)}{c} \leq f_{opt}(S) \leq f_{opt}(P)$

- **Example**
  - Optimization Function: Distance of the two farthest points

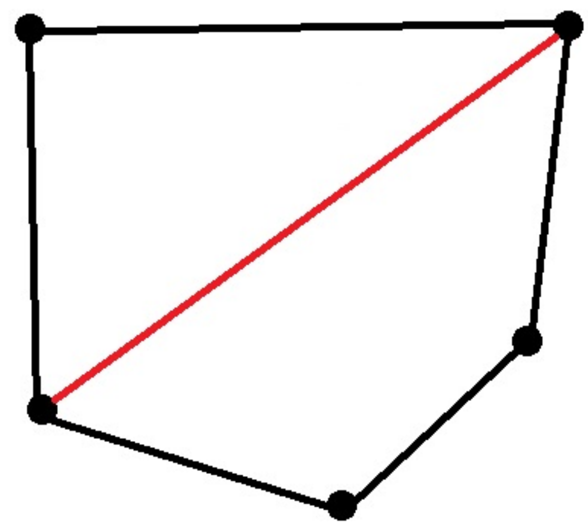


Core-Set Definition

- **Setup**
  - Set of n points $\boldsymbol{P}$ in d-dimensional space
  - Optimize a function f

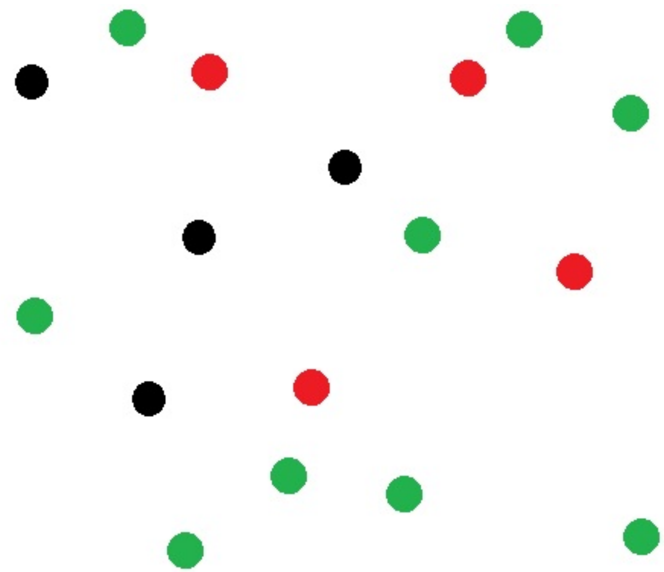
- ***c*-Core-set:** Small subset of points $S \subset P$ which suffices to c-approximate the optimal solution
- Maximization: $\frac{f_{opt}(P)}{c} \leq f_{opt}(S) \leq f_{opt}(P)$

- **Example**
  - Optimization Function: Distance of the two farthest points
  - 1-Core-set: Points on the convex hull.



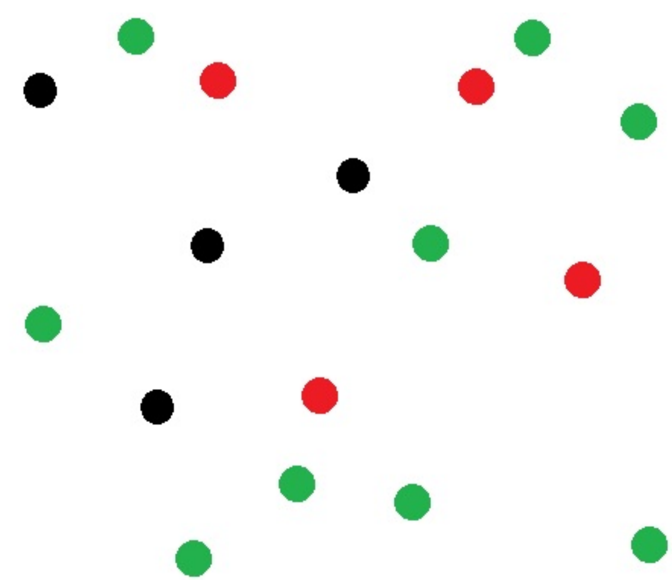
Composable Core-sets

- **Setup**
  - $\boldsymbol{P_1}, \boldsymbol{P_2}, \dots, \boldsymbol{P_m}$ are set of points in d-dimensional space
  - Optimize a function f over their union $\boldsymbol{P}$.



Composable Core-sets

- **Setup**
  - $\boldsymbol{P_1}, \boldsymbol{P_2}, \dots, \boldsymbol{P_m}$ are set of points in d-dimensional space
  - Optimize a function f over their union $\boldsymbol{P}$.

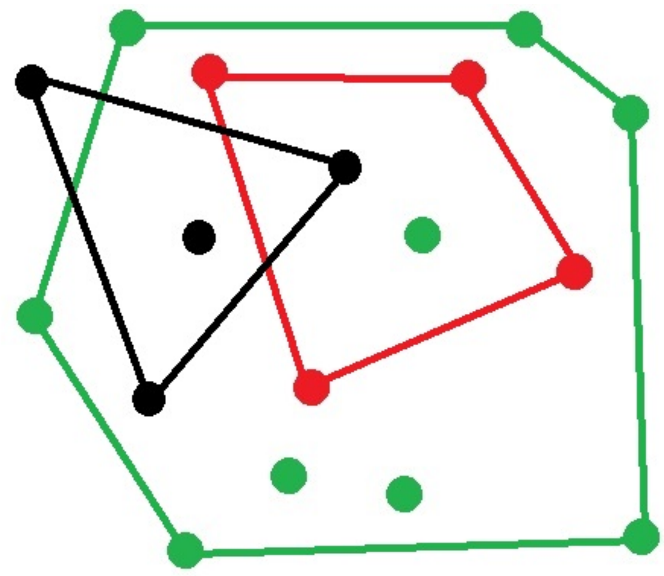


- ***c*-Composable Core-sets:** Subsets of points $S_1 \subset P_1, S_2 \subset P_2, \dots, S_\mathrm{m} \subset P_m$ points such that the solution of the unio of the core-sets approximates the solutic of the point sets.
  - Maximization :

$$\frac{1}{c} f_{opt}(P_1 \cup \cdots \cup P_m) \leq f_{\mathrm{opt}}(S_1 \cup \cdots \cup S_m) \leq f_{opt}(P_1 \cup \cdots \cup P_m)$$

Composable Core-sets

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



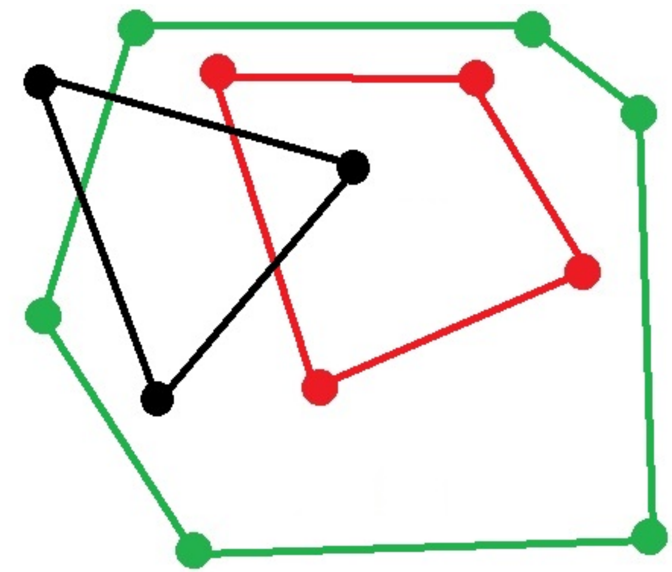
- **c-Composable Core-sets:** Subsets of points $S_1 \subset P_1, S_2 \subset P_2, \ldots, S_{\mathrm{m}} \subset P_m$ points such that the solution of the unic of the core-sets approximates the soluti of the point sets.
  - Maximization :

$$\frac{1}{c} f_{opt}(P_1 \cup \cdots \cup P_m) \leq f_{\mathrm{opt}}(S_1 \cup \cdots \cup S_m) \leq f_{opt}(P_1 \cup \cdots \cup P_m)$$

- **Example:** two farthest points

Composable Core-sets

- **Setup**
  - $\boldsymbol{P_1}, \boldsymbol{P_2}, \dots, \boldsymbol{P_m}$ are set of points in d-dimensional space
  - Optimize a function f over their union $\boldsymbol{P}$.



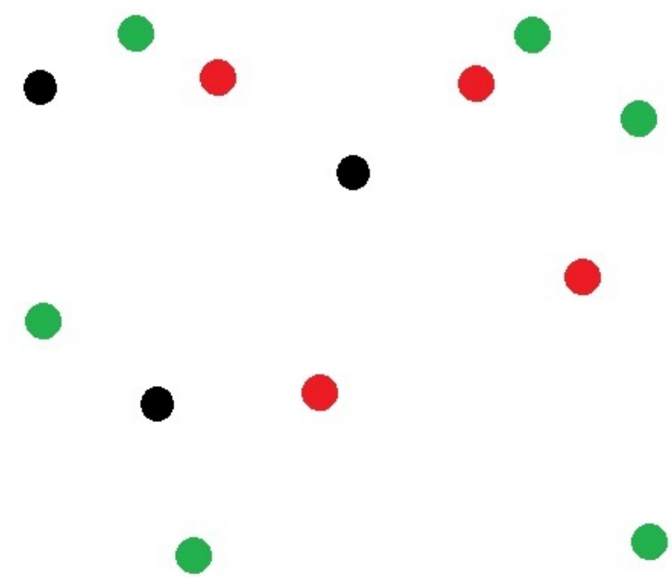
- ***c*-Composable Core-sets:** Subsets of points $S_1 \subset P_1, S_2 \subset P_2, \dots, S_\mathrm{m} \subset P_m$ points such that the solution of the unio of the core-sets approximates the solutic of the point sets.
  - Maximization :

$$\frac{1}{c} f_{opt}(P_1 \cup \dots \cup P_m) \le f_{\mathrm{opt}}(S_1 \cup \dots \cup S_m) \le f_{opt}(P_1 \cup \dots \cup P_m)$$

- **Example:** two farthest points

Composable Core-sets

- **Setup**
  - $\boldsymbol{P_1}, \boldsymbol{P_2}, \dots, \boldsymbol{P_m}$ are set of points in d-dimensional space
  - Optimize a function f over their union $\boldsymbol{P}$.



- ***c*-Composable Core-sets:** Subsets of points $S_1 \subset P_1, S_2 \subset P_2, \dots, S_{\mathrm{m}} \subset P_m$ points such that the solution of the unio of the core-sets approximates the solutic of the point sets.
  - Maximization :

$$\frac{1}{c} f_{opt}(P_1 \cup \cdots \cup P_m) \leq f_{\mathrm{opt}}(\mathrm{S}_1 \cup \cdots \cup S_m) \leq f_{opt}(P_1 \cup \cdots \cup P_m)$$

- **Example:** two farthest points