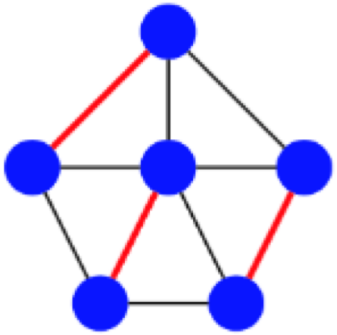
Randomized Composable Coresets for Matchings and Vertex Cover

Sepehr Assadi Sanjeev Khanna

University of Pennsylvania

Matchings in Graphs

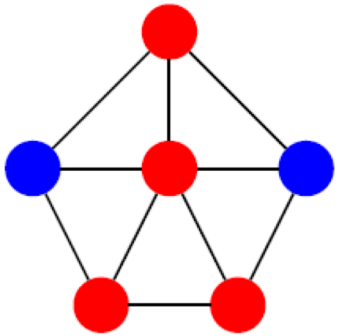
Matching: A set of edges in a graph such that every vertex has at most one edge incident on it.



Maximum Matching Problem: Find a matching with a largest number of edges.

Vertex Cover

Vertex Cover: A set of vertices that contains at least one end-point for every edge in the graph.



Minimum Vertex Cover: Find a vertex cover with a smallest number of vertices.

Matchings and Vertex Cover

- These are extensively studied problems in the classical model where an algorithm has complete access to the input graph.
- As input graphs become very large, a commonly used paradigm for solving graph problems is to distribute the computation over multiple machines.
- In this talk, we will examine both these problems in a simple model of distributed computing called the simultaneous communication model.

A Distributed Computation Model

- The edges of the input graph are distributed across k machines/players.
- There exists an additional party called the coordinator.
- The computation proceeds in rounds.
- In each round, the players simultaneously send a message to the coordinator, the coordinator synthesizes this information, and communicates back to each machine.
- After the last round, the coordinator outputs the answer.

Simultaneous Communication Model

An important and appealing special case is the simultaneous communication model: only one round of communication.

- Each machine sends a summary of its input to the coordinator who then outputs the answer.
- Inherently round-optimal -- the only measure left to optimize is amount of communication from each machine.
- Results in this model have implications for other well-studied models for computing with large data sets: Streaming and Map Reduce.

Optimization Goal

Suppose edges of a graph $G(V, E)$ are partitioned across k machines in an arbitrary manner: machine i has a subgraph $G_i(V, E_i)$.

- Each player's input may have $\Omega\left(\frac{n^2}{k}\right)$ edges.
- Can each player summarize its input using a message of size $\tilde{O}(n)$?

Question: What is the smallest summary size per player that suffices for the coordinator to output a solution?

Commonly used techniques for summarizing the input include linear sketching and composable coresets.

Linear Sketches

- We view the input graph as a vector of edge multiplicities.
- Each machine creates its summary to be a linear projection of its input subgraph.
- Linearity allows the coordinator to combine these summaries to create a linear projection of the entire graph.
- The coordinator then performs an arbitrary computation on the combined linear projection to compute the answer.

Composable Core Sets

- Each machine creates a summary by choosing a suitable subgraph of its input, called a coreset.
- Composability means that the union of the coresets sent by various machines yields a coreset for the union of the subgraphs at various machines.
- The coordinator then performs a computation on the combined coreset to compute the answer.

Some Examples

Graph Connectivity: Each machine can send an $O(n)$ size summary, namely, a spanning forest of its input.

Minimum Spanning Tree: Each machine can send an $O(n)$ size summary, namely, a minimum cost spanning forest of its input.

Cut Sparsifiers: Each machine can send an $\tilde{O}(n)$ size summary, either a cut sparsifier of its input or a suitable linear sketch.

Other examples: Graph Spanners, subgraph counting, etc.

Some Examples

Graph Connectivity: Each machine can send an $O(n)$ size summary, namely, a spanning forest of its input.

Minimum Spanning Tree: Each machine can send an $O(n)$ size summary, namely, a minimum cost spanning forest of its input.

Cut Sparsifiers: Each machine can send an $\tilde{O}(n)$ size summary, either a cut sparsifier of its input or a suitable linear sketch.

Other examples: Graph Spanners, subgraph counting, etc.

Missing from this list: Matchings and Vertex Cover.

What Summary Size Suffices for Matchings and Vertex Cover?

Suppose we wish to compute an α-approximate matching or α-approximate vertex cover of an n–vertex graph. As a function of n and α, what is the size of the message that each player needs to send to the coordinator?

What Summary Size Suffices for Matchings and Vertex Cover?

Suppose we wish to compute an α-approximate matching or α-approximate vertex cover of an n–vertex graph. As a function of n and α, what is the size of the message that each player needs to send to the coordinator?

Theorem 1 [Assadi-Li-K-Yaroslavtsev '16]: Any simultaneous protocol for computing an α-approximate matching requires at least one player to send a message of size $\widetilde{\Omega}(n^2/poly(\alpha))$.

An essentially identical lower bound holds for the vertex cover problem.

Lower Bound Instances

Our starting point is Ruzsa-Szemerédi (RS) graphs.

- Edges in an RS graph can be partitioned into induced matchings of almost linear size.
- Remarkably, one can construct such graphs with $\Omega(n^2)$ edges [Alon, Moitra, Sudakov '12].
- The lower bound is obtained by carefully partitioning edges of modified RS graphs across the machines:
  - Each machine's input contains a special induced matching but all induced matchings look identical.
  - So to send information about the special matching, a machine has to send information about all matchings.
  - Each machine thus needs to communicate almost quadratic amount of information.

Lower Bound Instances

- The lower bound requires both an adversarial graph and a adversarial partitioning of the edges of the graph.
- Can one bypass this intractability result without restricting the class of input graphs?
- In other words, is adversarial partitioning the primary source of intractability?

A Data Oblivious Partitioning Scheme

Yes!

A natural data oblivious partitioning scheme completely bypasses this intractability barrier!

Simply partition the edges of the input graph randomly across the machines.

Each edge is assigned uniformly at random to one of the k machines.

Our Results

Theorem 2 [Assadi-K '17] : If the edges of a graph are randomly partitioned across machines, then each machine can send an $O(n)$ size summary that suffices for the coordinator to compute an $O(1)$–approximate matching.

Our Results

Theorem 2 [Assadi-K '17] : If the edges of a graph are randomly partitioned across machines, then each machine can send an $O(n)$ size summary that suffices for the coordinator to compute an $O(1)$–approximate matching.

Theorem 3 [Assadi-K '17] : If the edges of a graph are randomly partitioned across machines, then each machine can send an $O(n \log n)$ size summary that suffices for the coordinator to compute an $O(\log n)$–approximate vertex cover.

Randomized Composable Coresets

For both matching and vertex cover problems, the summary sent by each machine is a subgraph of the original graph.

So our results show that matching and vertex cover problems admit $\tilde{O}(n)$ size randomized composable coresets that yield $\tilde{O}(1)$-approximation for these problems.

The notion of randomized composable coresets was introduced by [Mirrokni and Zadimoghaddam '15] in the context of distributed submodular maximization.

Plan for the Rest of the Talk

- We will design and analyze a very simple randomized composable coreset for the matching problem.
- We will sketch high-level idea for randomized composable coreset for the vertex cover problem.
- We will then conclude with some subsequent developments.

Randomized Composable Coreset for Matching

Theorem 2 [Assadi-K '17] : If the edges of a graph are randomly partitioned across machines, then each machine can send an $O(n)$ size summary that suffices for the coordinator to compute an $O(1)$–approximate matching.

- Each machine computes a maximum matching of its input and sends it to the coordinator.
- The coordinator then simply outputs a maximum matching of the edges received by it.

Thus the summary is simply a maximum matching in each machine's input graph.

Proof Sketch

- Let $M_1,\ldots,M_k$ be the collection of k maximum matchings computed by the machines.
- Now consider running the greedy algorithm over the edges in $M_1,\ldots,M_k$ in this order.
- For simplicity, assume the optimal matching is of size $\Omega(n)$.

Proof Sketch

- Let $M_1,\ldots,M_k$ be the collection of k maximum matchings computed by the machines.
- Now consider running the greedy algorithm over the edges in $M_1,\ldots,M_k$ in this order.
- For simplicity, assume the optimal matching is of size $\Omega(n)$.

Lemma: At any step $i \in [k]$, either the greedy matching is already of size $\Omega(n)$, or w.h.p. we can increase the size of the greedy matching by adding $\Omega\left(\frac{n}{k}\right)$ edges from M_i.

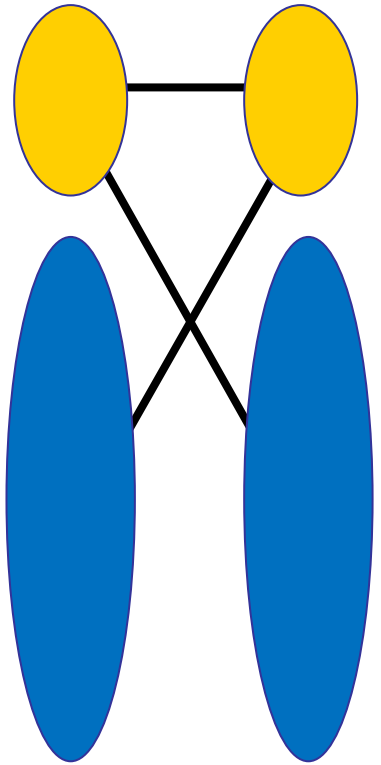
The lemma immediately implies that the matching output by the algorithm has size $\Omega(n)$ w.h.p.

Proof Sketch

G_i = Set of edges in player $P_i's$ input

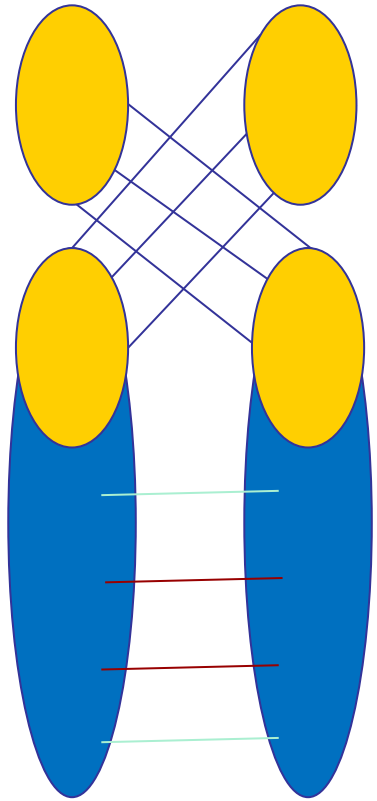
- Suppose the greedy has only matched $o(n)$ vertices so far.
- E_{old} : all edges in G_i that are incident on these already matched vertices.
- μ_{old}: size of a maximum matching in G_i only using edges in E_{old} .

Already Matched Vertices



Proof Sketch

Claim: W.h.p. there exists a matching of size $\mu_{old} + \Omega\left(\frac{n}{k}\right)$ in G_i.



- Fix a matching of size μ_{old}; at most $o(n)$ vertices that were previously unmatched are in this matching.
- So G contains a matching of size $\Omega(n)$ outside of vertices in μ_{old}.
- By random partitioning, w.h.p. $\Omega\left(\frac{n}{k}\right)$ of these edges appear in G_i.
- μ_{old} + these $\Omega\left(\frac{n}{k}\right)$ edges form the desired matching.

Proof Sketch

- There exists a matching of size $\mu_{old} + \Omega\left(\frac{n}{k}\right)$ in G_i.
- Hence any maximum matching in G_i needs to have $\mu_{old} + \Omega\left(\frac{n}{k}\right)$ edges.
- By definition, at most μ_{old} of these edges can be incident on the vertices already matched by the greedy algorithm.

Corollary: Any maximum matching in G_i contains w.h.p. $\Omega\left(\frac{n}{k}\right)$ edges with the property that for each edge, neither end-point was already matched by the greedy algorithm.

Can we improve this further?

Can we obtain even smaller summaries than $O(n)$ size? Afterall, each machine only needs to contribute $O(n/k)$ edges to the maximum matching.

Can we improve this further?

Can we obtain even smaller summaries than $O(n)$ size? Afterall, each machine only needs to contribute $O(n/k)$ edges to the maximum matching.

The answer is no!

Theorem 4 [Assadi-K '17] : Any α-approximation simultaneous distributed protocol for the matching problem requires $\Omega\left(\frac{n}{\alpha^2}\right)$ size messages even when the input is randomly partitioned across the machines.

Randomized Composable Coreset for Vertex Cover

Can we use a minimum vertex cover as a coreset when edges are randomly partitioned?

Randomized Composable Coreset for Vertex Cover

Can we use a minimum vertex cover as a coreset when edges are randomly partitioned?

No!

Consider the star graph on n vertices. This would result in a solution of size $\Omega(n)$ whereas the optimal solution has size 1.

Designing a small coreset for vertex cover is a distinctly harder task because of the strong feasibility constraint – every edge needs to be covered.

Randomized Coreset for Vertex Cover

Our coreset consists of a subset of vertices as well as a subgraph. Each machine computes a coreset by using the following peeling process:

- Iteratively remove high degree vertices and the edges incident on them.
- Vertices removed in each iteration are added to a set that will be in the final solution output by the coordinator.
- Once the remaining graph is sparse (i.e. $O(n \log n)$ edges), we send it along with the set of peeled vertices.

The peeling process was originally introduced by [Parnas and Ron '07] in the context of sublinear time algorithms.

Concluding Remarks

- With adversarial partitioning, the matching and vertex cover problem require essentially quadratic size summaries for any constant factor approximation in the simultaneous communication model.

- With random partitioning, $\tilde{O}(n)$–size summaries suffice for $\tilde{O}(1)$–approximation.

- This also gives a 2-round map-reduce algorithm for computing an $\tilde{O}(1)$–approximate matching or vertex cover using $O(\sqrt{n})$ machines with $O(n^{3/2})$ space per machine.

Concluding Remarks

- Our result may be viewed as showing that the matching and vertex cover problems admit randomized composable coresets.

- Very recently, [Assadi-Bateni-Bernstein-Mirrokni-Stein '17] improved our approximation for matching to $3/2 + \epsilon$, and the approximation for vertex cover to $2 + \epsilon$.

- The paradigm of random partitioning of input graph appears to be a useful starting point for designing communication-efficient distributed algorithms.

Thank you !