# 0.51-Approximation of Maximum Matching in Sublinear $n^{1.5}$ Time

Sepideh Mahabadi
MSR

Mohammad Roghani
Stanford, MSR intern

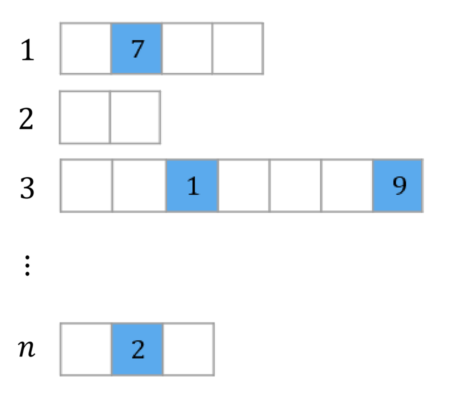
Jakub Tarnawski
MSR

## Sublinear matching

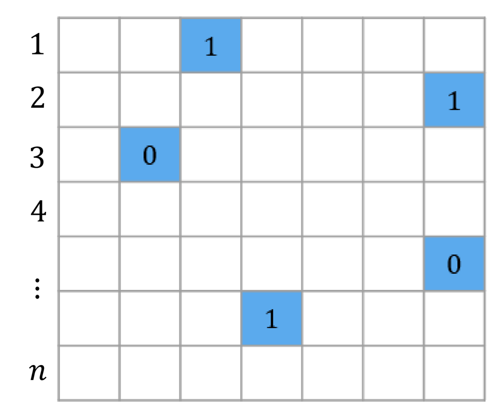
**Given:** query access to undirected graph
**Goal:** estimate size of maximum matching

**Want:**
- Runtime: $n^{2-\Omega(1)}$
- High approximation ratio

Can't **find edges** of a large matching in sublinear time [Parnas, Ron 07]



Adjacency list



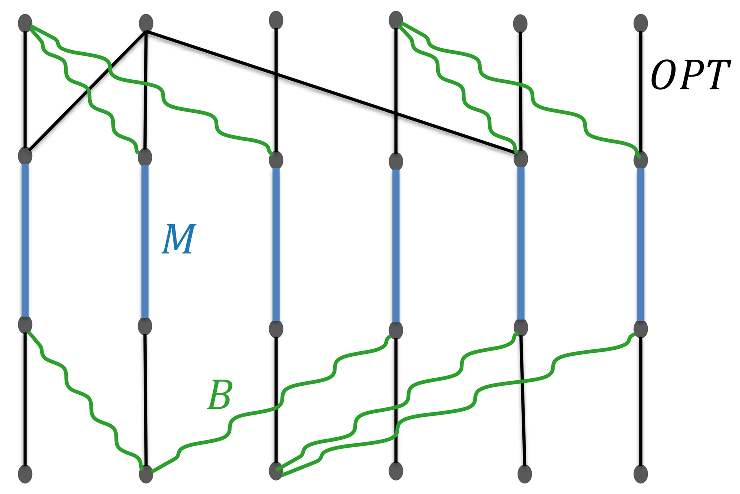
Adjacency matrix

## Two-pass semi-streaming algorithm that we want to simulate...

**Pass 1:** construct maximal matching $M$

**Pass 2: *augment $M$*:** construct maximal $b$-matching $B$ between matched and unmatched vertices. Capacity of matched vertices is 1, capacity of unmatched vertices is $b$.

**Return** maximum matching in $M \cup B$




[Konrad, Naidu 21]

Think $b = 2$

Optimizing $b$, this obtains $(2-\sqrt{2}) \approx 0.585$-approximation

## ...but we can't compute $M$

**Challenge 1:**
we cannot materialize a maximal matching in sublinear time

Idea: we instead use the RGMM oracle of [Behnezhad 21] to ask whether a vertex is matched or not.
This takes time $O(d)$ where $d$ = average degree.

inner oracle

Then, we again use [Behnezhad 21] oracle for the $b$-matching. When seeing edge $e$, to check whether $e$ is between matched and unmatched vertex,
it invokes the inner oracle.

outer oracle

**Challenge 2:**
runtime = $O(d)$ inner oracle calls = $O(d) \cdot O(d) = O(n^2)$

We instead start by **sparsifying** the graph and materialize a matching $M$ that is maximal in the subsampled subgraph:

$M := \emptyset$
For $v \in V$:
  sample $\tilde{O}(\sqrt{n})$ random neighbors
  for sampled $w$:
    add $(v, w)$ to $M$ if possible

takes time $\tilde{O}(n\sqrt{n})$

Now $M$: some *non-maximal* matching in $G$ (with $O(1)$-time access)

Now: unmatched graph $G' = G[V \setminus V(M)]$ has degree $\leq \sqrt{n}$ w.h.p.

Next, we extend $M$ to maximal matching $M \cup M'$ where $M'$ will be accessed via inner oracle.

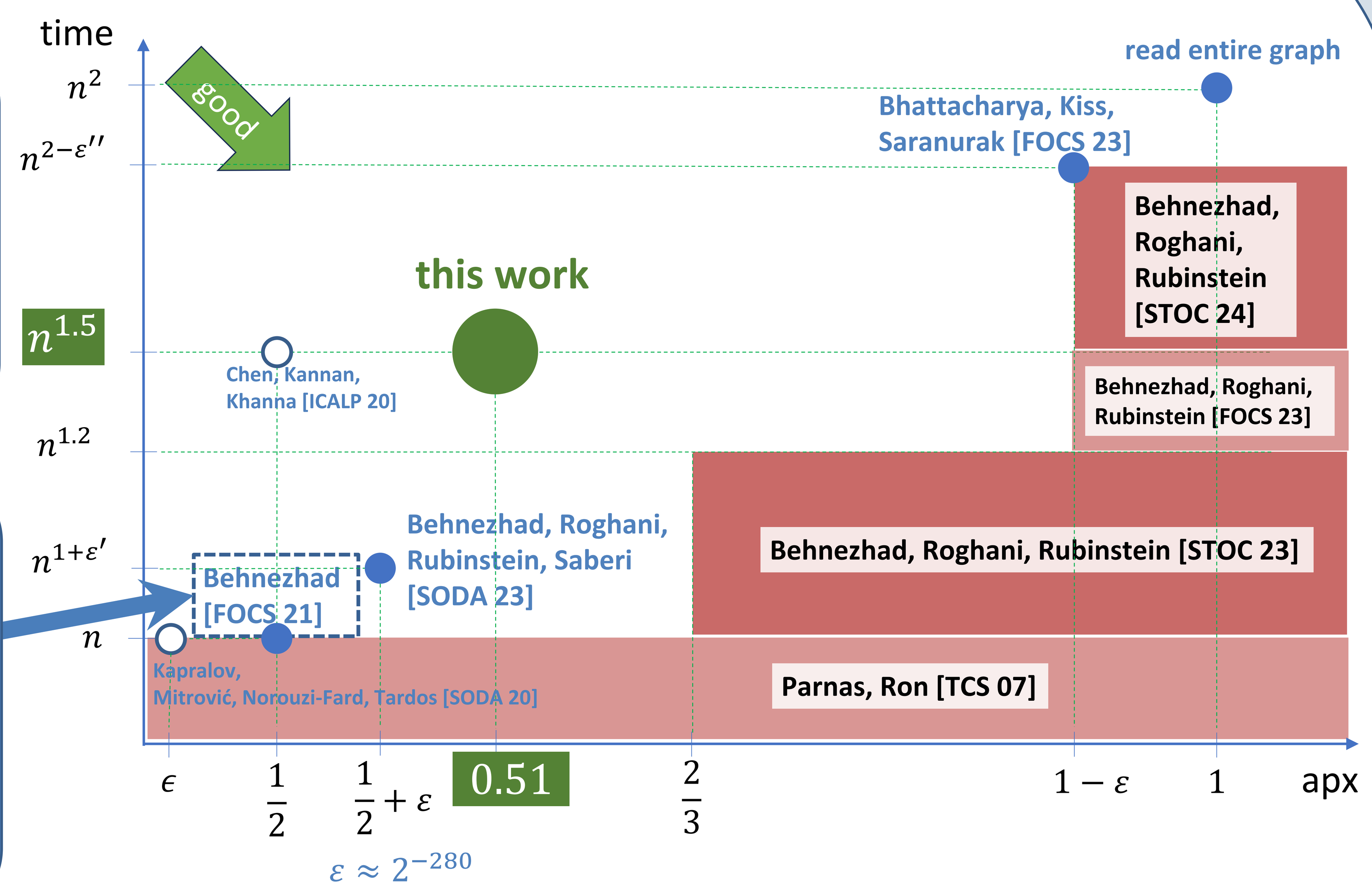
And try to augment $M \cup M'$ with a $b$-matching...

inner oracle

**But still,**
avg degree between $M \cup M'$ and complement can be $O(n)$,
so we would still be at $O(n^2)$ in worst case

## Approximation vs time tradeoff

Prior to our work, best existing algorithms were
- very close to $n^2$ runtime
- or very close to $\frac{1}{2}$-apx




**Crucial tool for us:**

oracle that,
for a vertex $v$,
says whether $v \in M$
in time $O(d)$

$M$: fixed (random greedy) *maximal* matching (RGMM)

**Our result:**
- given adjacency list access, multiplicative 0.5109-apx
- given adjacency matrix access, multiplicative-additive $(0.5109, o(n))$-apx

in time $\tilde{O}(n^{1.5})$ with high probability

## Our algorithm

Take $M$ as-is, and augment only $M'$
(by finding $b$-matching $B_1$)

This way, both oracles operate inside $G'$ which is low-degree

**OR**

Augment only $M$
(by finding b-matching $B_2$)

Only one oracle, because $M$ is materialized (can check if vertex is matched in $O(1)$ time)

For apx ratio, think of case when $M \cup M'$ is only a $\frac{1}{2}$-apx

So runtime of right leg simply $\leq O(n)$

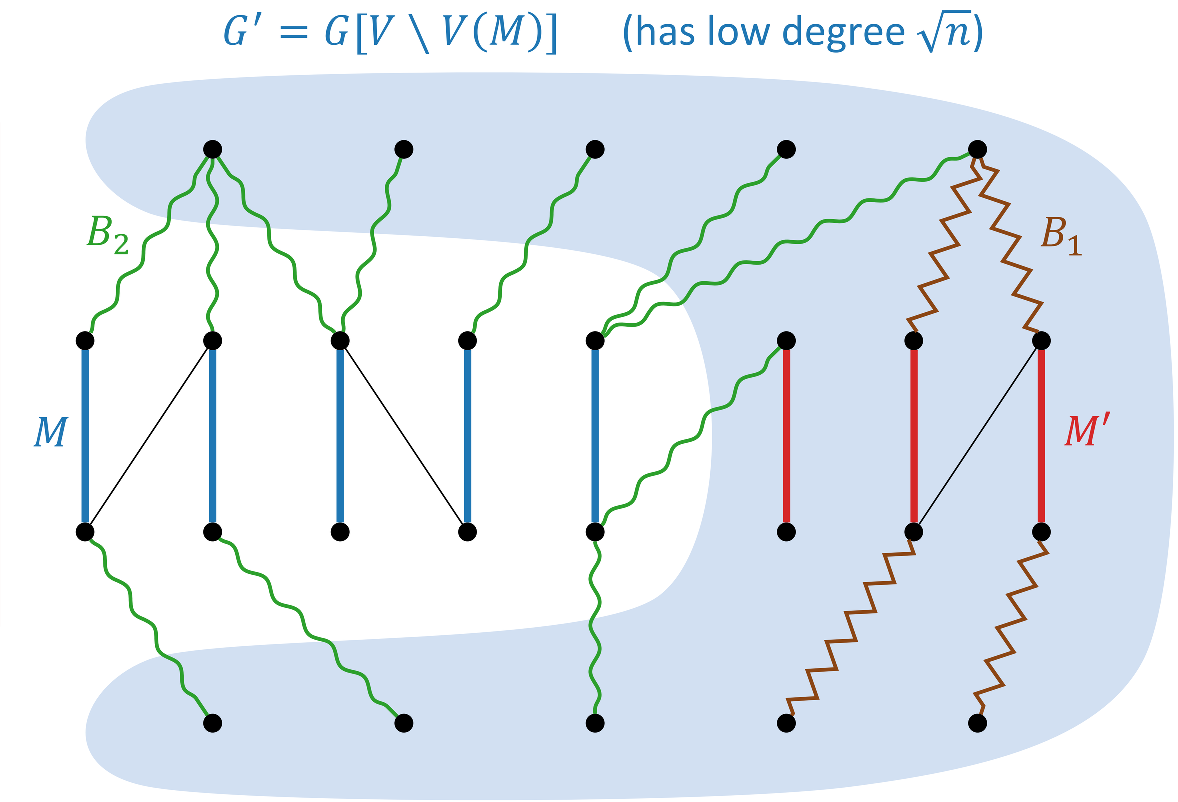
$G' = G[V \setminus V(M)]$ (has low degree $\sqrt{n}$)

Apx ratio:

We get solution size $\approx$ $|M| + (2-\sqrt{2}) \cdot 2|M'|$

Good if $|M'|$ large

Balancing left & right gives 0.5109-apx




Apx ratio:

We get solution size $\approx$ $(2-\sqrt{2}) \cdot 2|M|$

Good if $|M|$ large

Balancing left & right gives 0.5109-apx

Remains to analyze runtime of left leg:

Seemingly,
$O(\sqrt{n})$ outer oracle calls $\times$ $O(\sqrt{n})$ inner oracle calls $= O(n)$ runtime...
if we had access to adjacency list of $G' = G[V \setminus V(M)]$.

**Challenge 3:** But we don't.

Retrieving full adjacency list of a vertex costs $O(n)$.
So we're back to $O(n^2)$?

**Key property 1 of RGMM oracle:**
at each step, it needs a *random* neighbor of some vertex

Expected number of samples from adjacency list of $v$ in $G$ to get neighbor in $G'$ is at most ${}^{n}/_{d_{G'}(v)}$
so if we had all degrees in $G'$ equal to $d$,
then: $\underbrace{O(d)}_{\text{outer}} \cdot \underbrace{O(d)}_{\text{inner}} \cdot \underbrace{O({}^{n}/_{d})}_{\text{overhead}} = O(nd) = O(n\sqrt{n})$, great.

**Challenge 4:** But what if some $d_{G'}(v)$ is small?

(Then overhead is $O(n)$, so we're back to $O(n^2)$?)

**Key property 2 of RGMM oracle:**
it visits every vertex proportionally to its degree

[Mahabadi, Roghani, Vakilian, Tarnawski 25]

So the total runtime finally looks like:

$$\sqrt{n} \cdot \sum_v \frac{d_{G'}(v)}{n} \cdot \frac{n}{d_{G'}(v)} = \sqrt{n} \cdot n$$