Faster Streaming and Scalable Algorithms for Finding Directed Dense Subgraphs in Large Graphs

Slobodan Mitrović, **Theodore Pan**

University of California, Davis

July 2024

Densest subgraph problem

Given an undirected graph G, we define the density given a vertex set S to be

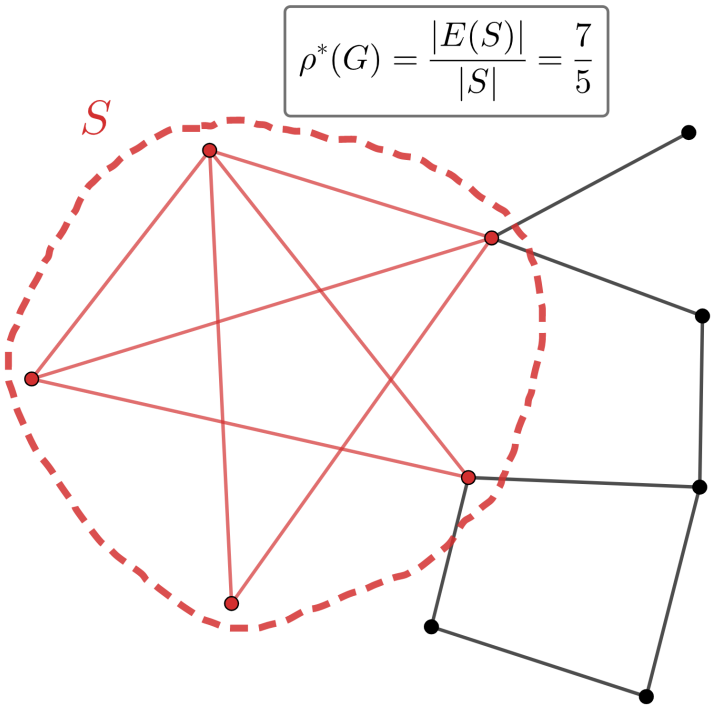
$$\rho(S) = \frac{|E_G(S)|}{|S|}.$$

For a **directed** graph G, we define the density given two vertex sets S, T to be

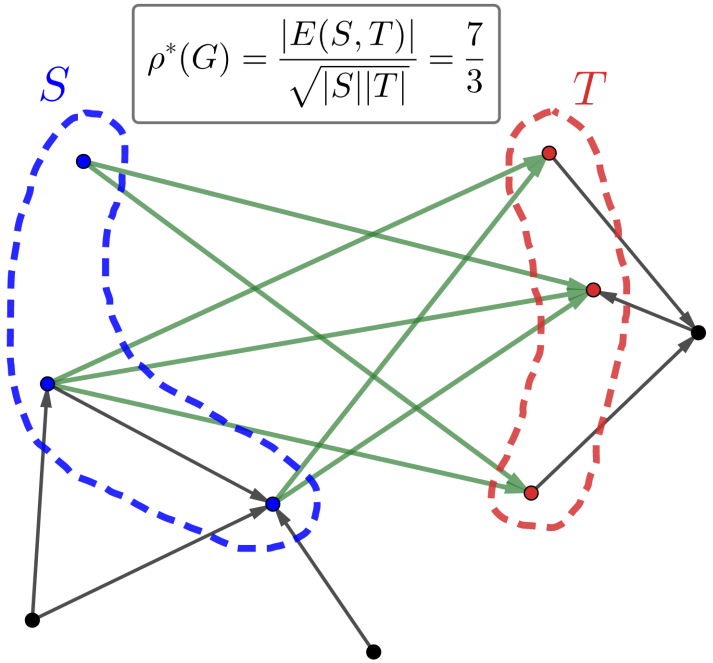
$$\rho(S,T) = \frac{|E_G(S,T)|}{\sqrt{|S||T|}}.$$

The **directed densest subgraph** are sets S^*, T^* that attain the maximum density $\rho^*(G)$. Our algorithm finds a $(2+\epsilon)$-approximation of the densest subgraph.

Densest subgraph problem



(a) Undirected graph

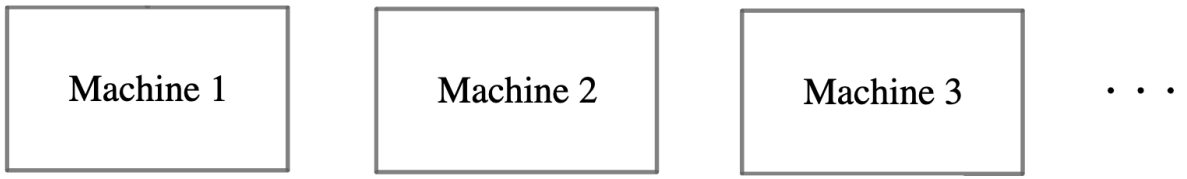


(b) Directed graph

Model definitions

Semi-streaming model: You can store $O(n \operatorname{poly} \log n)$ edges locally.

Massively Parallel Computation (MPC) model: Each machine holds certain amount of memory - for $\delta \in (0, 1)$, sub-linear $O(n^{\delta})$, near-linear $O(n \operatorname{poly} \log n)$, super-linear $O(n^{1+\delta})$.



Contributions

Streaming/Semi-streaming:

Algorithm	Approx factor	Memory	Passes
Bahmani et al. (VLDB 2012)	$(2+\epsilon)$	$O(n)$	$O(\log_{1+\epsilon} n)$
Esfandiari et al. (SPAA 2015)	$(1+\epsilon)$	$O\left(\frac{n\sqrt{n}\log^2(n)}{\epsilon^2}\right)$	1
Our (randomized stream)	$(2+\epsilon)$	$O\left(\frac{n\log^2(n)}{\epsilon^2}\right)$	1

MPC:

Algorithm	Approx factor	Memory	Rounds
Bahmani et al. (WAW 2014)	$(1+\epsilon)$	sub-linear	$O(\log(n)/\epsilon^2)$
Bahmani et al. (VLDB 2012)	$(2+\epsilon)$	sub-linear	$O(\log(n)/\epsilon)$
Our	$(2+\epsilon)$	near-linear	$O(\sqrt{\log(n)})$
Our	$(2+\epsilon)$	super-linear	$O(1)$

Peeling algorithm

Let $c = |S^*|/|T^*|$ where S^*, T^* are vertex sets of a directed densest subgraph.

1. Let $S, T, S^*, T^* \leftarrow V$
2. Sample $O(n \log n)$ edges uniformly at random from $E(S, T)$
3. If $|S|/|T| \geq c$, remove vertices from S that are below $(1+\epsilon)$·average degree
   Else, remove vertices from T that are below $(1+\epsilon)$·average degree
4. Update S^*, T^* with the densest subgraph S, T seen so far
5. Repeat step 2 if S and T are both nonempty

Average degree is calculated from the subgraph consisting of edges from $E(S, T)$ each iteration.

Peeling algorithm

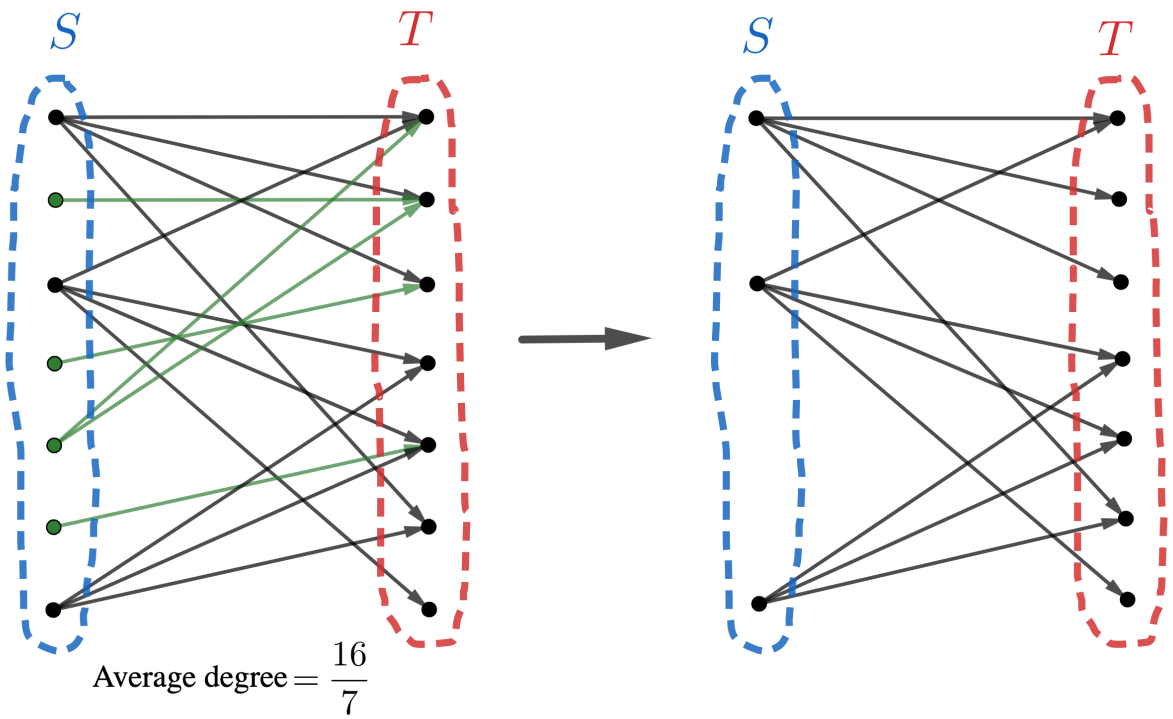


Figure: One iteration of peeling on set S

One-pass streaming algorithm

Our idea is to take multiple samples of the graph throughout one pass, using each of them to apply an iteration of peeling.

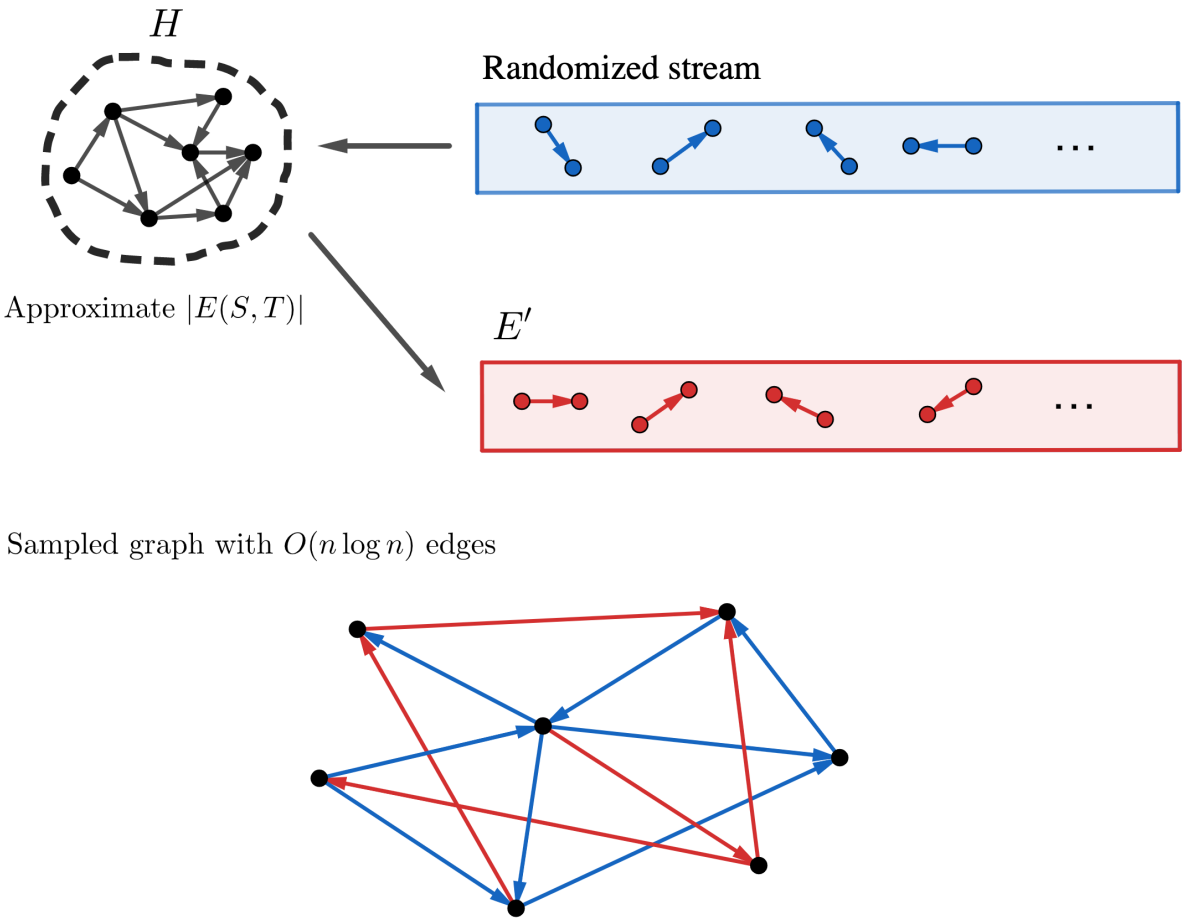


Figure: Sampling routine

Super-linear MPC algorithm

We simulate the streaming algorithm, with the first machine acting as the start of a randomized stream.

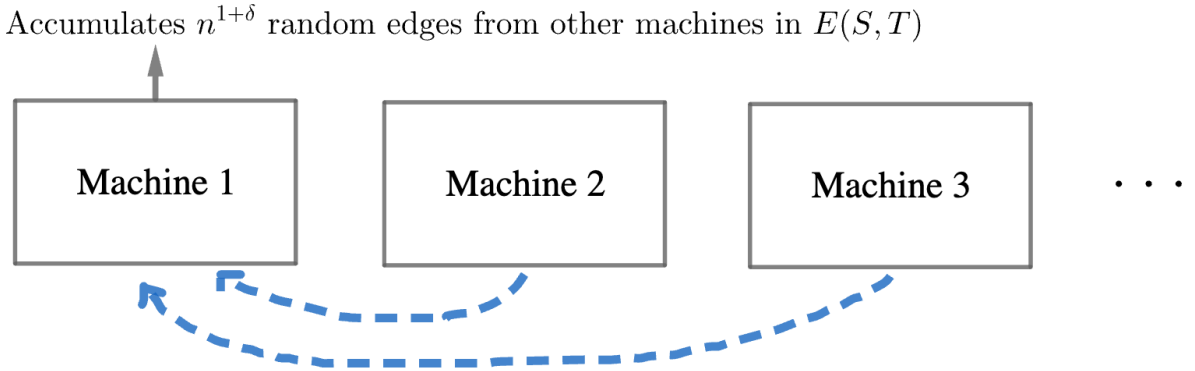


Figure: One round of super-linear MPC algorithm

The algorithm then takes a constant amount of rounds, $O(1/\delta)$ where each machine holds $n^{1+\delta}$ edges.

Near-linear MPC algorithm

We make two observations to adapt our previous algorithm to the near-linear memory regime.

1. We only need to take samples of $O((|S| + |T|) \log n)$ edges rather than $O(n \log n)$ edges.
2. Consecutive iterations of peeling on the same set, either S or T, do not require a fresh sample of edges. This guarantees that $(|S| + |T|)$ goes down by a constant factor each time we take another sample of edges.

Results in a quadratic optimization from previous sub-linear algorithms.

Experimental results

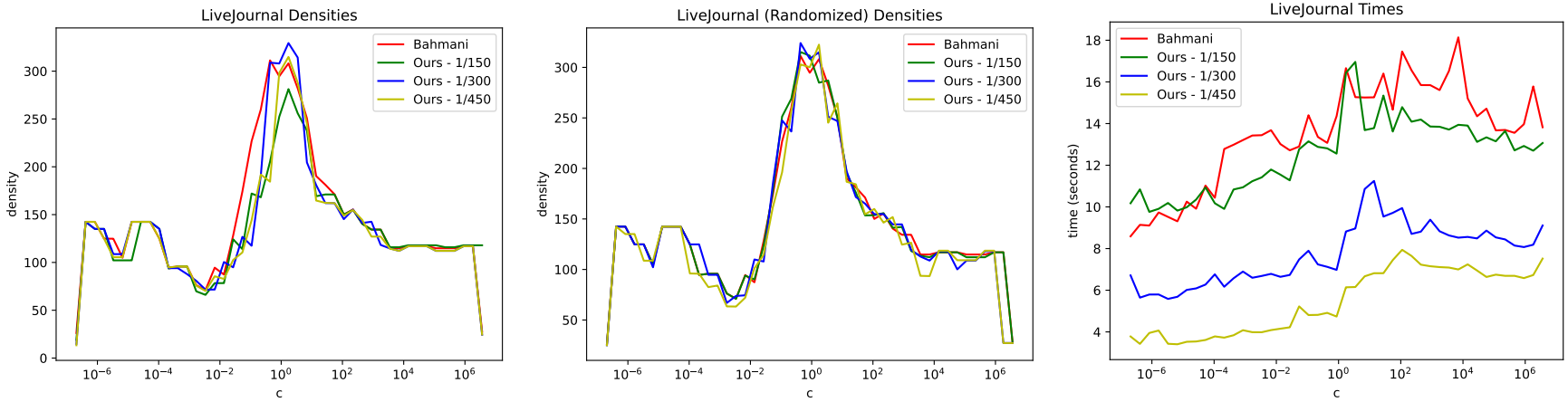


Figure: Density and running-time as a function of c for LiveJournal dataset

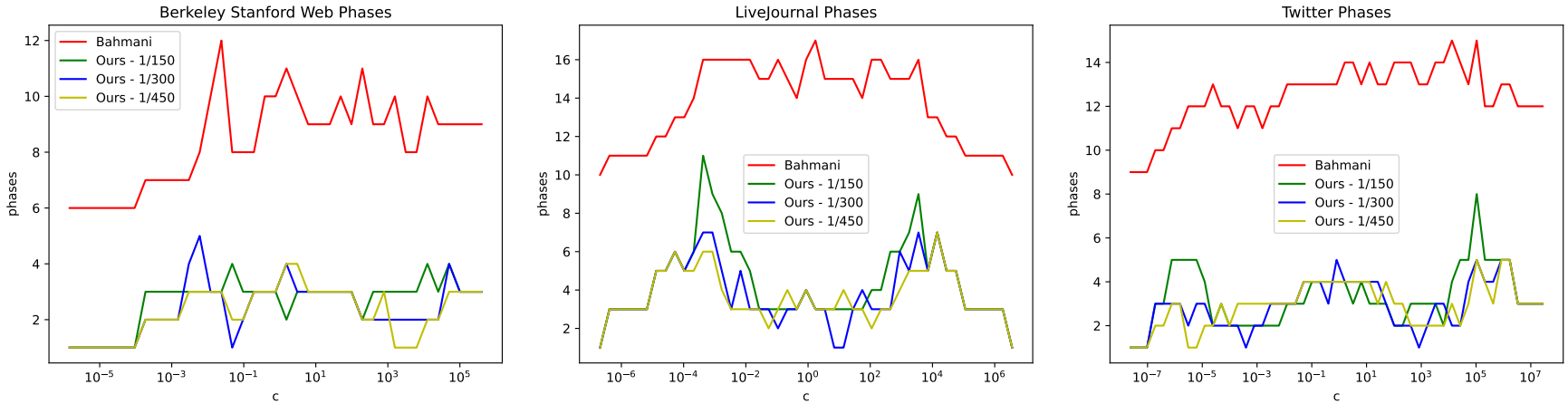


Figure: Number of phases as a function of c for various data sets

Future work

1. We hope to explore the trade-off between the number of passes and the memory requirement of streaming algorithms for this problem.
2. Is there a way to attain a $o(\log n)$ MPC round algorithm for the sub-linear memory regime? How about improving our near-linear MPC algorithm to a $(1+\epsilon)$-approximation?