Approximating Densest Subgraph in Geometric Intersection Graphs

Sariel Har-Peled[1] Saladi Rahul[2]

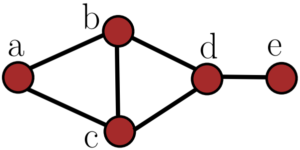
[1]University of Illinois Urbana Champaign (UIUC)

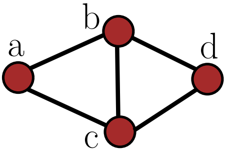
[2]Indian Institute of Science (IISc),

March 4, 2025

Densest Subgraph

Density $\frac{6}{5} = 1.2$





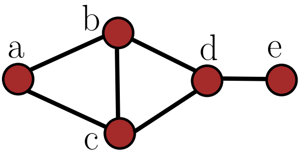
Densest subgraph $\{a, b, c, d\}$

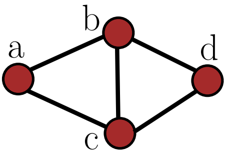
$\frac{5}{4} = 1.25$

- Report the subset of V with the *maximum density*

Densest Subgraph

Density $\frac{6}{5} = 1.2$





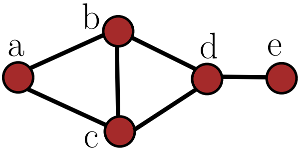
Densest subgraph $\{a, b, c, d\}$

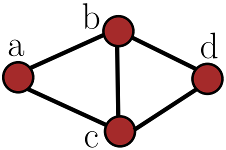
$\frac{5}{4} = 1.25$

- Report the subset of V with the *maximum density*
- Undirected graph $G = (V, E)$
- For any $S \subseteq V$, its density $= \frac{|E_S|}{|S|}$
- Each edge in $E_S \subseteq E$ has both its vertices in S

Densest Subgraph

Density $\frac{6}{5} = 1.2$





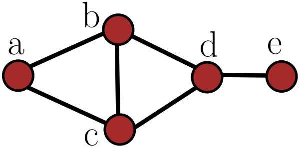
Densest subgraph $\{a, b, c, d\}$

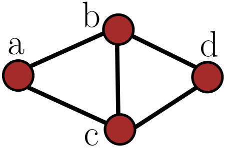
$\frac{5}{4} = 1.25$

- *Many applications...*
  - Mining closely-knit communities
  - Link-spam detection
- *Lot of interest in the theory and applied communities*

Densest Subgraph

Density $\frac{6}{5} = 1.2$





Densest subgraph $\{a, b, c, d\}$

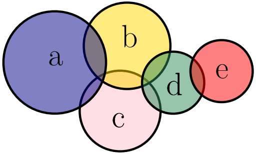
$\frac{5}{4} = 1.25$

- *Many applications...*
  - Mining closely-knit communities
  - Link-spam detection
- *Lot of interest in the theory and applied communities*
- Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. *A survey on the densest subgraph problem and its variants.*

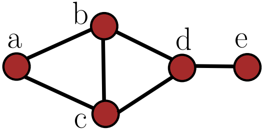
Densest Subgraph

- Can be solved exactly in polynomial time
  - [Goldberg'84], [Gallo, Grigoriadis, Tarjan'89], [Charikar'00], [Khuller, Saha'09]
- 2-approximation algorithm
  - [Asahiro, Iwama, Tamaki and Tokuyama'00]
  - Analyzed by [Charikar'00]
- $(1+\varepsilon)$-approximation algorithm
  - [Bahmani, Goel, Munagala'14]
- *Message:* $\Omega(|E|)$ time taken by all the algorithms

Densest Subgraph in Disk Intersection Graphs

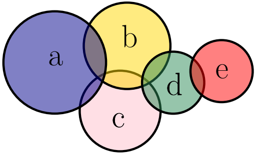


- Collection of n *disks* in the plane
- *Disk Intersection Graph*

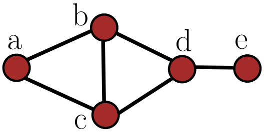


Can near-linear time (in terms of n) algorithms be designed?

Densest Subgraph in Disk Intersection Graphs

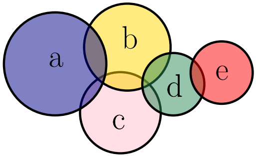


- Collection of n *disks* in the plane
- *Disk Intersection Graph*
- A vertex associated with each disk.

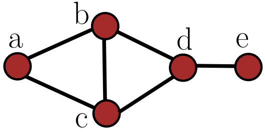


Can near-linear time (in terms of n) algorithms be designed?

Densest Subgraph in Disk Intersection Graphs

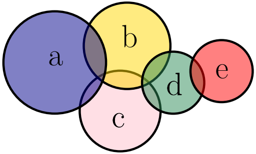


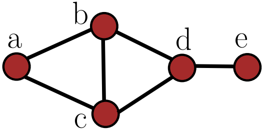
- Collection of n *disks* in the plane
- *Disk Intersection Graph*
- A vertex associated with each disk.
- Two vertices have an edge iff the corresponding disks intersect.



Can near-linear time (in terms of n) algorithms be designed?

Densest Subgraph in Disk Intersection Graphs





- Collection of n *disks* in the plane
- *Disk Intersection Graph*
- A vertex associated with each disk.
- Two vertices have an edge iff the corresponding disks intersect.
- *Implicit* disk intersection graph:
  - Only the disks are given as input
  - Edges are not known *explicitly*

Can near-linear time (in terms of n) algorithms be designed?

Our Results for Disks

Approximation	*Running Time*
$(2+\varepsilon)$	$O\left(\frac{n \log n}{\varepsilon^4}\right) \approx O_\varepsilon(n \log n)$
$(1+\varepsilon)$	$O\left(\frac{n \log^2 n}{\varepsilon^2}\left(\frac{1}{\varepsilon^2} + \log\log n\right)\right) \approx \tilde{O}_\varepsilon(n \log^2 n)$

Can near-linear time (in terms of n) algorithms be designed?

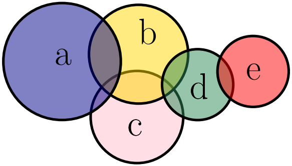
The $(2+\varepsilon)$-approximation algorithm

The $(2+\varepsilon)$-approximation algorithm

- *Idea-1:* Connection with *deepest point*
- *Idea-2:* Efficiently throwing away *low-degree* vertices

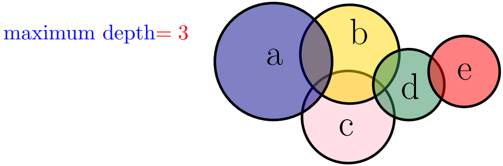
Idea-1: Connection with Deepest Point

maximum depth= 3



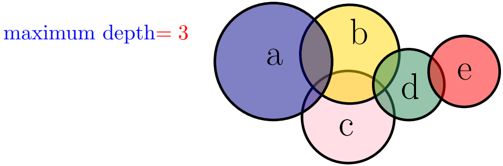
- *Depth* of p is the number of disks containing point p.

Idea-1: Connection with Deepest Point



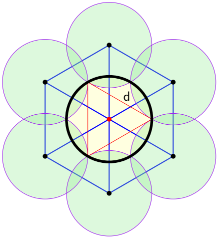
- *Depth* of p is the number of disks containing point p.
- *Deepest* point is maximum depth point.

Idea-1: Connection with Deepest Point



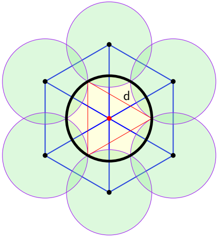
- *Depth* of p is the number of disks containing point p.
- *Deepest* point is maximum depth point.
  - A 1.1-approximation can be computed in $O(n \log n)$ time [Aronov and Har-Peled'08].

Idea-1: Connection with Deepest Point



- *Depth* of p is the number of disks containing point p.
- *Deepest* point is maximum depth point.
  - A 1.1-approximation can be computed in $O(n \log n)$ time [Aronov and Har-Peled'08].

Idea-1: Connection with Deepest Point



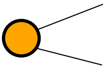
- *Depth* of p is the number of disks containing point p.
- *Deepest* point is maximum depth point.
  - A 1.1-approximation can be computed in $O(n \log n)$ time [Aronov and Har-Peled'08].
- *Lemma:* $\frac{deepest-1}{2} \leq OPT \leq 7 \cdot deepest$.
  - Cute discrete geometry problem.

The $(2+\varepsilon)$-approximation algorithm

- *Idea-1:* Connection with *deepest point*
- *Idea-2:* Efficiently throwing away *low-degree* vertices

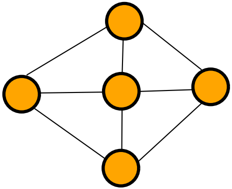
Idea-2: low-degree vertex

degree $< OPT$



low-degree vertex

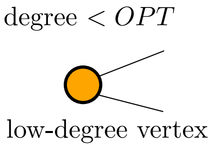
degree $\geq OPT$

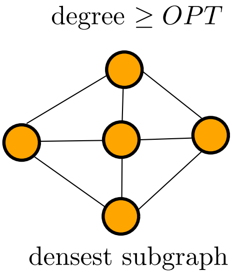


densest subgraph

- *Observation:* None of the vertices in the optimal solution are low degree.
- *Algorithm's goal:* Remove low degree vertices quickly from the graph.

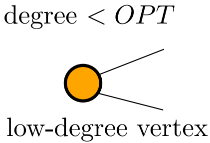
Idea-2: Few low-degree vertices

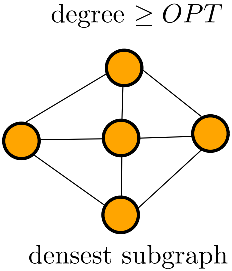




- *Case 1:* # low-degree vertices $\leq \varepsilon n$.
- *Most* vertices in the graph have degree $\geq OPT$.
- Entire graph is a good approximation
  - density $= \frac{\#edges}{n} \geq \frac{OPT \cdot (1-\varepsilon)n}{2n} = \frac{OPT}{2+\varepsilon'}$

Idea-2: Many low-degree vertices





- *Case 2:* # low-degree vertices $\geq \varepsilon n$.
- Need to throw away such vertices.
- *Recurse* on the remaining graph.
- *Luckily*, number of recursive steps $= O(\varepsilon^{-1} \log n)$.

Running time

- Efficient data structure for *(approximate) degree*
  - Preprocessing time $= O(n \log n)$, query time $= O(\log n)$.
  - This is what lets us obtain *near-linear* time algorithm.

Running time

- Efficient data structure for *(approximate) degree*
  - Preprocessing time $= O(n \log n)$, query time $= O(\log n)$.
  - This is what lets us obtain *near-linear* time algorithm.
- $\#$ *guesses* of $OPT = O(1)$.

Running time

- Efficient data structure for *(approximate) degree*
    - Preprocessing time $= O(n \log n)$, query time $= O(\log n)$.
    - This is what lets us obtain *near-linear* time algorithm.
- $\#$ *guesses* of $OPT = O(1)$.
- $\#$ *iterations* $= O(\varepsilon^{-1} \log n)$.

Running time

- Efficient data structure for *(approximate) degree*
  - Preprocessing time $= O(n \log n)$, query time $= O(\log n)$.
  - This is what lets us obtain *near-linear* time algorithm.
- *# guesses* of $OPT = O(1)$.
- *# iterations* $= O(\varepsilon^{-1} \log n)$.
- *Running time* $= O(\sum_i n_i \log n_i) = O_\varepsilon(n \log n)$, since $n_i \leq (1 - \varepsilon) n_{i-1}$. *Optimal!*

The $(1+\varepsilon)$*-approximation algorithm*

The $(1+\varepsilon)$-approximation algorithm

- *Idea-1: Sampling* $O_\varepsilon(n \log n)$ edges suffices
- *Idea-2:* Efficient *data structure* for sampling edges

Idea-1: Near-linear number of edges sampled

- Approximate densest subgraph preserved under *uniform-sampling* of edges
    - Inspired from streaming algorithms

Idea-1: Near-linear number of edges sampled

- Approximate densest subgraph preserved under *uniform-sampling* of edges
  - Inspired from streaming algorithms
- Perform $\Theta_\varepsilon(n \cdot \log n)$ rounds
  - In each round, probability of picking an edge is $\approx \frac{1 \pm \varepsilon}{|E|}$

Idea-1: Near-linear number of edges sampled

- Approximate densest subgraph preserved under *uniform-sampling* of edges
  - Inspired from streaming algorithms
- Perform $\Theta_\varepsilon(n \cdot \log n)$ rounds
  - In each round, probability of picking an edge is $\approx \frac{1 \pm \varepsilon}{|E|}$
- Let E_S be the sampled edges

Idea-1: Near-linear number of edges sampled

- Approximate densest subgraph preserved under *uniform-sampling* of edges
  - Inspired from streaming algorithms
- Perform $\Theta_\varepsilon(n \cdot \log n)$ rounds
  - In each round, probability of picking an edge is $\approx \frac{1 \pm \varepsilon}{|E|}$
- Let E_S be the sampled edges
- Run an approximation algorithm on *sparse graph* (V, E_S)

Idea-2: Efficient Data Structure for Sampling Edges

- *Streaming algos:* Edges are streamed one-by-one.
  - Trivial to sample.

Idea-2: Efficient Data Structure for Sampling Edges

- *Streaming algos:* Edges are streamed one-by-one.
  - Trivial to sample.
- *Implicit* geometric intersection graph
  - Remember edges are not explicitly known!

Idea-2: Efficient Data Structure for Sampling Edges

- *Streaming algos:* Edges are streamed one-by-one.
  - Trivial to sample.
- *Implicit* geometric intersection graph
  - Remember edges are not explicitly known!
- *Key contribution:* Generic technique to use *range reporting* data structures into samples edges (almost)-*uniformly* at random.

Idea-2: Efficient Data Structure for Sampling Edges

- *Streaming algos:* Edges are streamed one-by-one.
  - Trivial to sample.
- *Implicit* geometric intersection graph
  - Remember edges are not explicitly known!
- *Key contribution:* Generic technique to use *range reporting* data structures into samples edges (almost)-*uniformly* at random.
- *Disks:* Preprocessing time $O_\varepsilon(n \log^2 n)$, sample takes $\tilde{O}_\varepsilon(\log n)$ time

Final Comments

Other Geometric Intersection Graphs

- *Prerequisite:* Efficient range reporting data structures.
- Can obtain $(1+\varepsilon)$-approximation and $(2+\varepsilon)$-approximation.
- *Axis-aligned boxes* in d-dimensions
  - $n \log^{O(d)} n$ running time.
- *Balls* in d-dimensions
  - $O(n^{2-\lambda})$ running time for some $\lambda \in (0,1)$.

Open Problems

- *Exact* computation of densest subgraph in *sub-quadratic* time?
  - Unit-disk graphs or Interval graphs?

Open Problems

- *Exact* computation of densest subgraph in *sub-quadratic* time?
  - Unit-disk graphs or Interval graphs?
- *Dynamic* densest subgraph under insertion and deletion of disks
  - Update time: *sub-linear* in n?
  - Several edges are *implicitly* inserted or deleted in each round.

Open Problems

- *Exact* computation of densest subgraph in *sub-quadratic* time?
  - Unit-disk graphs or Interval graphs?
- *Dynamic* densest subgraph under insertion and deletion of disks
  - Update time: *sub-linear* in n?
  - Several edges are *implicitly* inserted or deleted in each round.
- Other variants of densest subgraph in geometric intersection graphs.

Thank You