Dense Subgraph Discovery (DSD): Theory and Applications

Charalampos (Babis) E. Tsourakakis$^{1,2}$
Tianyi Chen $^1$

$^1$Boston University

$^2$ISI Fondazione

SDM 2021
Tutorial

slides and code:

https://tinyurl.com/sdm21dsd

tutors:

Charalampos Tsourakakis

Tianyi Chen

Updated version of KDD 2015 tutorial by Aris Gionis-CT
What this tutorial is about . . .

given a graph (network), static or dynamic
(social network, biological network, information network, . . . )

find a subgraph that . . .

. . . has many edges

. . . is densely connected

why we care?

**Key Focus** on the **Densest Subgraph Problem (DSP)** . . .

. . . and its variants
Densest subgraph problem

Degree density: \( \rho(S) = \frac{e(S)}{|S|} \). E.g., \( \rho(S) = \frac{7}{5} \)

- \( \max_{S \subseteq V} \rho(S) \) **Poly-time solvable** (via max flows)

Also, there exists a 2-approximation algorithm which uses linear space \( O(n + m) \) and runs in linear time \( O(n + m) \) due to [Charikar, 2000]

“The densest subgraph problem (DSP) lies at the core of large scale data mining” [Bahmani et al., 2012]

- \( \max_{S \subseteq V} \rho(S) \) subject to **cardinality constraints** becomes computationally hard, e.g., \( |S| = k \) (DkS), \( |S| \leq k \) (DalkS), \( |S| \geq k \) (DamkS) [Bhaskara et al., 2010, Khuller and Saha, 2009, Andersen and Chellapilla, 2009]
Prototypical DSD: Max-Clique Problem

- highly inapproximable problem [Hastad, 1996]
Tutorial Outline

- Motivating applications
- Measures of density
- **DSP:** Static case
- **DSP:** Dynamic case
- **DSP, DSD** problem variants
- Conclusion
Motivating applications
correlation mining: a general framework with many applications

- data is converted into a graph
- vertices correspond to entities
- an edge between two entities denotes strong correlation
  1. stock correlation network: data represent stock timeseries
  2. gene correlation networks: data represent gene expression
- dense subsets of vertices correspond to highly correlated entities
- applications:
  1. analysis of stock market dynamics
  2. detecting co-expression modules/protein complexes
Applications – Large-Near Clique Extraction

Who-calls-whom

Twitter

Youtube

<table>
<thead>
<tr>
<th>Security</th>
<th>Social Media</th>
<th>Topic Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Humans</td>
<td>Videos</td>
</tr>
<tr>
<td>E</td>
<td>Phone call</td>
<td>Co-watched</td>
</tr>
<tr>
<td>Large Near</td>
<td>Anomaly</td>
<td>Thematic</td>
</tr>
<tr>
<td>clique</td>
<td>Thematic coherence</td>
<td>coherence</td>
</tr>
</tbody>
</table>

Dense Subgraph Discovery (DSD): Theory at SDM 2021
Applications – fraud detection

- dense bipartite subgraphs in page-like data reveal attempts to inflate page-like counts
  [Beutel et al., 2013]
Applications – e-commerce

e-commerce

- weighted bipartite graph $G(A \cup Q, E, w)$
- set $A$ corresponds to advertisers
- set $Q$ corresponds to queries
- each edge $(a, q)$ has weight $w(a, q)$ equal to the amount of money advertiser $a$ is willing to spend on query $q$

large almost bipartite cliques correspond to sub-markets
Applications – bioinformatics

- DNA motif detection [Fratkin et al., 2006]
  - vertices correspond to $k$-mers
  - edges represent nucleotide similarities between $k$-mers
- gene correlation analysis
- detect complex annotation patterns from gene annotation data [Saha et al., 2010]
Applications – distance queries in graphs

applications:
- driving directions
- indoor/terrain navigation
- routing in comm./sensor networks
- moving agents in game maps
- proximity in social/collab. networks

existing solutions:
- graph searches are too slow
- fast algorithms are often heuristics
- or tailored to specific graph classes

goals:
- fast exact queries
- scalability to large graphs
- wide range of inputs
Applications – distance queries in graphs

- $L(u) \equiv \text{set of pairs } (v, \text{dist}(u, v))$

$L(u)$ is the label of $u$; each $v$ is a hub for $u$.

figure from [Delling et al., 2014]
Applications – distance queries in graphs

- **preprocessing**: compute a label set for every vertex
- **cover property**: for all \( s, t \) intersection \( L(s) \cap L(t) \) must hit an \( s-t \) shortest path

figure from [Delling et al., 2014]
Applications – distance queries in graphs

- to answer an \( s-t \) query:
  find hub \( \nu \) in \( L(s) \cap L(t) \) minimizing \( \text{dist}(s, \nu) + \text{dist}(\nu, t) \)

figure from [Delling et al., 2014]
Applications – distance queries in graphs

hub label queries are trivial to implement:

- entries sorted by hub id
- linear sweep to find matches
- access to only two contiguous blocks (cache-friendly)

method is practical if labels sets are small

- can we find small labels sets?
- 2-hop labeling algorithm relies on DSP to find such label sets (!) [Cohen et al., 2003]
- state-of-art 2-hop labeling scheme: [Delling et al., 2014]
- more work on the topic: [Peleg, 2000, Thorup, 2004]
Applications – frequent pattern mining

• given a set of transactions over items
• find item sets that occur together in a $\theta$ fraction of the transactions

<table>
<thead>
<tr>
<th>issue number</th>
<th>heroes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Iceman, Storm, Wolverine</td>
</tr>
<tr>
<td>2</td>
<td>Aurora, Cyclops, Magneto, Storm</td>
</tr>
<tr>
<td>3</td>
<td>Beast, Cyclops, Iceman, Magneto</td>
</tr>
<tr>
<td>4</td>
<td>Cyclops, Iceman, Storm, Wolverine</td>
</tr>
<tr>
<td>5</td>
<td>Beast, Iceman, Magneto, Storm</td>
</tr>
</tbody>
</table>

e.g., \{Iceman, Storm\} appear in 60% of issues
Applications – frequent itemsets and dense subgraphs

<table>
<thead>
<tr>
<th>id</th>
<th>heroes</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>I</th>
<th>M</th>
<th>S</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Iceman, Storm, Wolverine</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Aurora, Cyclops, Magneto, Storm</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Beast, Cyclops, Iceman, Magneto</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Cyclops, Iceman, Storm, Wolverine</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Beast, Iceman, Magneto, Storm</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **transaction data** ⇔ **binary data** ⇔ **bipartite graphs**

**Frequent pattern mining:** Apriori, FP-growth, . . .
## Applications – frequent itemsets and dense subgraphs

<table>
<thead>
<tr>
<th>id</th>
<th>heroes</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>I</th>
<th>M</th>
<th>S</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Iceman, Storm, Wolverine</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Aurora, Cyclops, Magneto, Storm</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Beast, Cyclops, Iceman, Magneto</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Cyclops, Iceman, Storm, Wolverine</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Beast, Iceman, Magneto, Storm</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

• **Transaction data** ⇔ **Binary data** ⇔ **Bipartite graphs**
• **Frequent itemsets** ⇔ **Bi-cliques**
Applications – finding web communities

[Kumar et al., 1999]

• hypothesis: web communities consist of hub-like pages and authority-like pages

• key observations:

1. let $G = (U, V, E)$ be a dense web community then $G$ should contain some small core (bi-clique)

2. consider a web graph with no communities then small cores are unlikely

• both observations motivated from theory of random graphs
Applications – finding web communities

a web community

[Kumar et al., 1999]
Applications – finding web communities

web communities contains small cores

[Dense Subgraph Discovery (DSD): Theory and Applications, Kumar et al., 1999]
Applications – graph compression

- **compress web graphs** by finding and compressing bi-cliques [Karande et al., 2009]
- **many graph mining tasks** that can be formulated as matrix-vector multiplication, are more efficient on the compressed graph [Kang et al., 2009]
Applications – big and dynamic graphs

- **size** of graphs increases
  - e.g., in 2012, Facebook reported more than 1 billion users and 140 billion friend connections

- **graphs** change constantly
  - e.g., in Facebook friendships are created and deleted all the time

- need to design efficient algorithms on new computational models that handle **large-scale processing**
  - map-reduce, streaming models, etc.
Applications – Exclusion density queries

- Twitter users may interact in more than one ways, e.g., FOLLOW, RETWEET, REPLY.

- How can we find a subgraph that contains (i) lots of reply interactions, (ii) and no follow interactions?

- Such graph queries can be used to mine “interesting” groups of people and interactions, and also study behavioral patterns of Twitter users.
Applications – Anomaly detection using DSD

Online social networks
Fraud detection [Hooi et al., 2016]

How do we find graph anomalies using dense subgraph structures?
Money laundering transfers in a bank. Edge color and node size indicate the amount of money transferred. [Li et al., 2020]
Applications – Risk aversion

Which maximum matching should we choose?

risk averse DSP

How can we find dense subgraph structures with high expected reward, and low risk?
More applications

- graph visualization [Alvarez-Hamelin et al., 2005]
- community detection [Chen and Saad, 2012]
- epilepsy prediction [Iasemidis et al., 2003]
- event detection in activity networks [Rozenshtein et al., 2014]
- tampered financial derivatives [Arora et al., 2011]
- social piggypacking [Gionis et al., 2013]
- dependency testing [Devroye et al., 2011]
- ...
Measures of density
notation

- graph \( G = (V, E) \) with vertices \( V \) and edges \( E \subseteq V \times V \)
- degree of a node \( u \in V \) with respect to \( X \subseteq V \) is
  \[
  \deg_X(u) = |\{v \in X \text{ such that } (u, v) \in E\}|
  \]
- degree of a node \( u \in V \) is \( \deg(u) = \deg_V(u) \)
- edges between \( S \subseteq V \) and \( T \subseteq V \) are
  \[
  E(S, T) = \{(u, v) \text{ such that } u \in S \text{ and } v \in T\}
  \]
  use shorthand \( E(S) \) for \( E(S, S) \)
- graph cut is defined by a subset of vertices \( S \subseteq V \)
- edges of a graph cut \( S \subseteq V \) are \( E(S, \bar{S}) \)
- induced subgraph by \( S \subseteq V \) is \( G(S) = (S, E(S)) \)
- triangles:
  \[
  T(S) = \{(u, v, w) \mid (u, v), (u, w), (v, w) \in E(S)\}
  \]
density measures

- undirected graph \( G = (V, E), n = |V|, m = |E| \)
- subgraph induced by \( S \subseteq V \)
- clique: all vertices in \( S \) are connected to each other
density measures

- **degree density** (average degree):

  \[
  d(S) = \frac{2|E(S, S)|}{|S|} = 2\frac{|E(S)|}{|S|}, \quad \rho(S) = \frac{|E(S)|}{|S|}
  \]

  (sometimes just drop 2)

- **edge ratio**:

  \[
  \delta(S) = \frac{|E(S, S)|}{\binom{|S|}{2}} = \frac{|E(S)|}{\binom{|S|}{2}} = \frac{2e(S)}{|S||S| - 1} \leq 1
  \]

  Independent set

  Clique
other density measures

• **k-core**: every vertex in $S$ is connected to at least $k$ other vertices in $S$

• **$\alpha$-quasiclique**: the set $S$ has at least $\alpha \left( \frac{|S|}{2} \right)$ edges

  i.e., $S$ is $\alpha$-quasiclique if $E(S) \geq \alpha \left( \frac{|S|}{2} \right)$

• [Kawase and Miyauchi, 2017]  **$f$ density** (generalization of edge density):

  $$d_f(S) = \frac{|E(S)|}{f(|S|)}$$

  e.g., $f(|S|) = |S|, |S|^{1.5}$
Densest subgraph problem: Static graphs
Goldberg’s algorithm for densest subgraph

- Consider first degree density $d$

- Is there a subgraph $S$ with $d(S) \geq c$?

- Transform to a min-cut instance

- On the transformed instance:
  - Is there a cut smaller than a certain value?
Goldberg’s algorithm for densest subgraph

is there $S$ with $d(S) \geq c$?

\[
\frac{2 |E(S, S)|}{|S|} \geq c
\]

\[
2 |E(S, S)| \geq c |S|
\]

\[
\sum_{u \in S} \deg(u) - |E(S, \bar{S})| \geq c |S|
\]

\[
\sum_{u \in S} \deg(u) + \sum_{u \in \bar{S}} \deg(u) - \sum_{u \in \bar{S}} \deg(u) - |E(S, \bar{S})| \geq c |S|
\]

\[
\sum_{u \in \bar{S}} \deg(u) + |E(S, \bar{S})| + c |S| \leq 2 |E|
\]
Goldberg’s algorithm for densest subgraph

• transformation to min-cut instance

• is there $S$ s.t. $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|$?
Goldberg’s algorithm for densest subgraph

- transform to a **min-cut** instance

\[ \text{is there } S \text{ s.t. } \sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E| ? \]

- a cut of value \(2|E|\) always exists, for \(S = \emptyset\)
Goldberg’s algorithm for densest subgraph

• transform to a min-cut instance

• is there $S$ s.t. $\sum_{u \in S} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|$ ?

• $S \neq \emptyset$ gives cut of value $\sum_{u \in S} \deg(u) + |e(S, \bar{S})| + c|S|$
Goldberg’s algorithm for densest subgraph

- transform to a \textit{min-cut} instance

\[ \sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|? \]

- is there \( S \) s.t. \( \sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|? \)
- \textbf{YES}, if min cut achieved for \( S \neq \emptyset \)
Goldberg’s algorithm for densest subgraph

[Goldberg, 1984]

**Input:** undirected graph $G = (V, E)$, number $c$

**Output:** $S$, if $d(S) \geq c$

1. transform $G$ into min-cut instance $G' = (V \cup \{s\} \cup \{t\}, E', w')$
2. find min cut $\{s\} \cup S$ on $G'$
3. if $S \neq \emptyset$ return $S$
4. else return NO

- to find the densest subgraph perform binary search on $c$
- logarithmic number of min-cut calls
- problem can also be solved with one min-cut call using the parametric max-flow algorithm
• Goldberg’s algorithm polynomial algorithm, but
• $O(nm)$ time for one min-cut computation
• not scalable for large graphs (millions of vertices / edges)

• faster algorithm due to [Charikar, 2000]
• greedy and simple to implement
• approximation algorithm . . .

• . . . and a new family of greedy algorithms.
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph — example
greedy algorithm for densest subgraph

[Charikar, 2000]

**input:** undirected graph \( G = (V, E) \)
**output:** \( S \), a dense subgraph of \( G \)

1. set \( G_n \leftarrow G \)
2. for \( k \leftarrow n \) downto 1
   2.1 let \( v \) be the smallest degree vertex in \( G_k \)
   2.2 \( G_{k-1} \leftarrow G_k \setminus \{v\} \)
3. output the densest subgraph among \( G_n, G_{n-1}, \ldots, G_1 \)
proof of 2-approximation guarantee

a neat argument due to [Khuller and Saha, 2009]

• let $S^*$ be the vertices of the optimal subgraph
• let $\rho(S^*) = \rho^*$ be the maximum degree density
• notice that for all $v \in S^*$ we have $\deg_{S^*}(v) \geq \rho^*$
• (why?) by optimality of $S^*$

\[
\frac{|e(S^*)|}{|S^*|} \geq \frac{|e(S^*)| - \deg_{S^*}(v)}{|S^*| - 1}
\]

and thus

\[
\deg_{S^*}(v) \geq \frac{|e(S^*)|}{|S^*|} = \rho(S^*) = \rho^*
\]
consider greedy when the first vertex $v \in S^* \subseteq V$ is removed

let $S$ be the set of vertices, just before removing $v$

total number of edges before removing $v$ is $\geq \rho^*|S|/2$

therefore, greedy returns a solution with degree density at least $\frac{\rho^*}{2}$

QED
the greedy algorithm

• factor-2 approximation algorithm
• runs in linear time $O(n + m)$
• for a polynomial problem . . .
  but faster and easier to implement than the exact algorithm
• everything goes through for weighted graphs
  using heaps: $O(m + n \log n)$
• things are not as straightforward for directed graphs

See Variants part.
Summary: Greedy Algorithm for Densest Subgraph

Input: Undirected graph $G = (V, E)$
Output: $S$, a dense subgraph of $G$

1. Set $G_n \leftarrow G$
2. For $k \leftarrow n$ downto 1
   2.1 Let $v$ be the smallest degree vertex in $G_k$
   2.2 $G_{k-1} \leftarrow G_k \setminus \{v\}$
3. Output the densest subgraph among $G_n, G_{n-1}, \ldots, G_1$

Theorem

Charikar’s algorithm is a $\frac{1}{2}$-approximation algorithm for DSP.

Question: Can we design an algorithm that combines the best of both worlds, i.e., optimality of maximum flows, and practicality of greedy?
Proposed algorithm: Greedy++

- **Flowless: Extracting Densest Subgraphs Without Flow Computations** (WWW 2020)
  Boob-Gao-Peng-Sawlani-T-Wang-Wang
LP formulation

\[ \text{maximize} \sum_{e \in E} y_e \]
\[ \text{subject to} \quad y_e \leq x_u, x_v, \quad \forall e = uv \in E \]
\[ \sum_{v \in V} x_v \leq 1, \]
\[ y_e \geq 0, x_v \geq 0, \quad \forall e \in E, \forall v \in V \]

Theorem [Charikar '00]: OPT of this LP = $\rho_G^*$
Dual of the LP

**Primal** ($G$)

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} y_e \\
\text{s. to} & \quad y_e \leq x_u, x_v, \\
& \quad \sum_{v \in V} x_v \leq 1, \\
& \quad y_e \geq 0, x_v \geq 0,
\end{align*}
\]

**Dual** ($G$)

\[
\begin{align*}
\text{min} & \quad D \\
\text{s. to} & \quad f_e(u) + f_e(v) \geq 1, \\
& \quad \sum_{e \ni v} f_e(v) \leq D, \\
& \quad f_e(v) \geq 0,
\end{align*}
\]
Dual LP

\[ \text{minimize } D \]
\[ \text{subject to } f_e(u) + f_e(v) \geq 1, \quad \forall e = uv \in E \]
\[ \sum_{e \ni v} f_e(v) \leq D, \quad \forall v \in V \]
\[ f_e(u) \geq 0, f_e(v) \geq 0, \quad \forall e = uv \in E \]

Find the smallest $D$ such that:

\[ f_e(u) + f_e(v) = 1 \]
\[ \sum_{e \ni v} f_e(v) \leq D \]
Visualizing node loads

- Each edge pushes its weight (i.e., edge load) to its endpoints.
- The load of each node equals to the sum of all edge weights pushed to it.

\[
\text{load}(v) = \sum_{e \ni v} f_e(v).
\]
Alternate visualization of dual

- Each edge has load = 1
- Assign this load fractionally to both end points
- **Minimize maximum load**

![Graph Diagram]

Optimal assignment:
- assign $be$ to $e$; $df$ to $f$
- all other edges equally to end points
- max load = 1.5
Observe greedy as a dual solution

- Dual problem: assign edges (fractionally) to endpoints
- Here, deleting a vertex $u = assigning$ all its incident edges to $u$
- Each edge $uv$ is assigned completely either to $u$ or $v$
- This can clearly be suboptimal in terms of dual solution

**Greedy++: improves the load balancing of Greedy**

Our algorithm is based on a multiplicative weights update method (MWU) [Arora et al., 2012, Plotkin et al., 1995] inspired algorithm that discards width modulation
Greedy++ - our algorithm

1. Run several iterations of greedy

2. First iteration $\rightarrow$ regular greedy algorithm

3. Update vertex degrees $\leftarrow$ degree $+$ load assigned previously

4. Run greedy with new “degrees” and repeat

Gives a far more balanced dual solution (recall: our aim is to minimize max load).
Greedy++ pseudocode – Input $G(V, E), T$

\[ G_{\text{densest}} \leftarrow G \]

Initialize the vertex load vector $\ell^{(0)} \leftarrow 0 \in \mathbb{Z}^n$;

for $i : 1 \rightarrow T$ do

\[ H \leftarrow G; \]

while $H \neq \emptyset$ do

Find the vertex $u \in H$ with minimum $\ell^{(i-1)}_u + \deg_H(u)$;

\[ \ell^{(i)}_u \leftarrow \ell^{(i-1)}_u + \deg_H(u); \]

Remove $u$ and all its adjacent edges $uv$ from $H$;

if $\rho(H) > \rho(G_{\text{densest}})$ then

\[ G_{\text{densest}} \leftarrow H \]

end if

end while

end for

Return $G_{\text{densest}}$
Tight example for Greedy

Our graph $G = B + \text{many copies of } H_i$

$B = K_{d,D}$

$H_i = K_{d+2}$

- Charikar gives $\frac{1}{2}$-approximation.
- Flowless with $T = 2$ gives (practically) optimal solution.
Large collection of datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>$n$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-trackers [23]</td>
<td>40 421 974</td>
<td>140 613 762</td>
</tr>
<tr>
<td>orkut [23]</td>
<td>3 072 441</td>
<td>117 184 899</td>
</tr>
<tr>
<td>livejournal-affiliations [23]</td>
<td>10 690 276</td>
<td>112 307 385</td>
</tr>
<tr>
<td>wiki-topcats</td>
<td>1 791 489</td>
<td>25 447 873</td>
</tr>
<tr>
<td>cit-Patents</td>
<td>3 774 768</td>
<td>16 518 948</td>
</tr>
<tr>
<td>actor-collaborations [23]</td>
<td>382 219</td>
<td>15 038 083</td>
</tr>
<tr>
<td>ego-gplus</td>
<td>107 614</td>
<td>12 238 285</td>
</tr>
<tr>
<td>dblp-author</td>
<td>5 425 963</td>
<td>8 649 016</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>685 230</td>
<td>6 649 470</td>
</tr>
<tr>
<td>flickr [37]</td>
<td>80 513</td>
<td>5 899 882</td>
</tr>
<tr>
<td>wiki-Talk</td>
<td>2 394 385</td>
<td>4 659 565</td>
</tr>
<tr>
<td>web-Google</td>
<td>875 713</td>
<td>4 322 051</td>
</tr>
<tr>
<td>com-youtube</td>
<td>1 134 890</td>
<td>2 987 624</td>
</tr>
<tr>
<td>roadNet-CA</td>
<td>1 965 206</td>
<td>2 766 607</td>
</tr>
<tr>
<td>web-Stanford</td>
<td>281 903</td>
<td>1 992 636</td>
</tr>
<tr>
<td>roadNet-TX</td>
<td>1 379 917</td>
<td>1 921 660</td>
</tr>
<tr>
<td>roadNet-PA</td>
<td>1 088 092</td>
<td>1 548 988</td>
</tr>
<tr>
<td>Ego-twitter</td>
<td>81 306</td>
<td>1 342 296</td>
</tr>
<tr>
<td>com-dblp</td>
<td>317 080</td>
<td>1 049 866</td>
</tr>
<tr>
<td>com-Amazon</td>
<td>334 863</td>
<td>925 872</td>
</tr>
<tr>
<td>soc-slashdot0902</td>
<td>82 168</td>
<td>504 230</td>
</tr>
<tr>
<td>soc-slashdot0811</td>
<td>77 360</td>
<td>469 180</td>
</tr>
<tr>
<td>soc-Epinions</td>
<td>75 879</td>
<td>405 740</td>
</tr>
<tr>
<td>blogcatalog [37]</td>
<td>10 312</td>
<td>333 983</td>
</tr>
<tr>
<td>email-Enron</td>
<td>36 692</td>
<td>183 831</td>
</tr>
<tr>
<td>ego-facebook</td>
<td>4 039</td>
<td>88 234</td>
</tr>
<tr>
<td>ppi [31]</td>
<td>3 890</td>
<td>37 845</td>
</tr>
<tr>
<td>twitter-retweet [30]</td>
<td>316 662</td>
<td>1 122 070</td>
</tr>
<tr>
<td>twitter-favorite [30]</td>
<td>226 516</td>
<td>1 210 041</td>
</tr>
<tr>
<td>twitter-mention [30]</td>
<td>571 157</td>
<td>1 895 094</td>
</tr>
<tr>
<td>twitter-reply [30]</td>
<td>196 697</td>
<td>296 194</td>
</tr>
<tr>
<td>soc-sign-slashdot081106</td>
<td>77 350</td>
<td>468 554</td>
</tr>
<tr>
<td>soc-sign-slashdot090216</td>
<td>81 867</td>
<td>497 672</td>
</tr>
<tr>
<td>soc-sign-slashdot090221</td>
<td>82 140</td>
<td>500 481</td>
</tr>
<tr>
<td>soc-sign-epinions</td>
<td>131 828</td>
<td>711 210</td>
</tr>
</tbody>
</table>
Number of iterations $T$ required to obtain $f\%$ accuracy

- Histograms of number of iterations to reach 99% of the optimum degree density (left), and the optimum degree density (right)
- **Remark**: typically, within less than 5 iterations we reach a near-optimal solution.
Experimental results

Convergence of Greedy++ on some large networks:

Figure: Amazon co-purchasing network

Figure: California road connection network
Experimental results

Scalability and speed of Greedy++:

**Figure:** Runtime in seconds per iteration

**Figure:** Speedup over exact algorithm (max-flow based)
Some additional remarks

1. When we are able to run the exact algorithm (for graphs with more than 8M edges, the maximum flow code crashes) on our machine, the average speedup that our algorithm provides to reach the optimum is $144.6 \times$ on average, with a standard deviation equal to 57.4. The smallest speedup observed was $67.9 \times$, and the largest speedup $290 \times$.

2. The maximum number of iterations needed to reach 90% of the optimum is at most 3, i.e., by running two more passes compared to Charikar’s algorithm, we are able to boost the accuracy by 10%.

3. Our algorithm GReedy++ when given enough number of iterations always finds the optimal value, and the densest subgraph.
Proposed algorithm: Frank-Wolfe based algorithm

- Large Scale Density-friendly Graph Decomposition via Convex Programming (WWW 2017)
  Danisch-Chan-Sozio
Key Concept - Quotient graph

$S_1, \ deg_1(S_1) = 1.5 \quad S_2, \ deg_2(S_2) = 1.33 \quad S_3, \ deg_3(S_3) = 1$

$G_1 = G \quad G_2 = G_1 \setminus S_1 \quad G_3 = G_2 \setminus S_2$

*Quotient graph* of $G_1$ with respect to node subset $S_1$ is $G_2$.

*Load vector* of $G$ load$^G = [1.5, 1.5, 1.5, 1.5, 1.33, 1.33, 1.33, 1, 1]$. 
The $\text{Dual}(G)$ is to evenly distribute edge weights to all nodes

$$
\text{Dual}(G) \quad \min \quad D \\
\text{s.t.} \quad D \geq \sum_{e:u \in e} f_e(u) = \text{load}(u), \quad \forall u \in V \\
\sum_{u \in e} f_e(u) \geq w_e, \quad \forall e \in E \\
f_e(u) \geq 0, \quad \forall u \in e \in E
$$

Consider $Q_G := \sum_{u \in V} \text{load}(u)^2$. 
Frank-Wolfe Algorithm

Solve the following convex problem:

$$\min_{\text{valid } f} Q_G(f) = \sum_{u \in V} \left( \sum_{e:u \in e} f_e(u) \right)^2$$

Algorithm 1  Frank-Wolfe Algorithm on function $Q_G$ and feasible set $\mathcal{F}$ that satisfies edge weight constraint.

1: Set initial $f^{(0)} \in \mathcal{F}$ arbitrarily
2: for each iteration $t = 1, \ldots, T$ do
3: \quad $\gamma_t = \frac{2}{t+2}$
4: \quad $\hat{f} = \arg \min_{f \in \mathcal{F}} \langle f, \nabla Q_G(f^{(t-1)}) \rangle$
5: \quad $f^{(t)} = (1 - \gamma_t)f^{(t-1)} + \gamma_t \hat{f}$
6: end for
Initialization: Each edge has weight 1, and is distributed evenly to both endpoints. For each node $u$, $load(u)$ is computed.
Frank-Wolfe Algorithm

Iteration $t = 1$: Set $\hat{f}_e(u) = 1$ if $u$ has the least $load(u)$ value among endpoints of $e$; otherwise $\hat{f}_e(u) = 0$. 
Iteration $t = 1$: Set $\hat{f}_e(u) = 1$ if $u$ has the least $\text{load}(u)$ value among endpoints of $e$; otherwise $\hat{f}_e(u) = 0$. 
Frank-Wolfe Algorithm

Iteration $t = 1$: Set $\hat{f}_e(u) = 1$ if $u$ has the least $\text{load}(u)$ value among endpoints of $e$; otherwise $\hat{f}_e(u) = 0$. 
Iteration $t = 1$: Compute load based on $\hat{f}$. 
Frank-Wolfe Algorithm

Iteration $t = 1$: Set $\gamma_t = \frac{2}{t+2}$

Compute $load^{(t)} = (1 - \gamma_t)load^{(t-1)} + \gamma_t \hat{load}$

Loop for $T$ iterations.
Comparison of methods

- Exact max-flow algorithm
- Charikar
- Flowless
- Frank-Wolfe based algorithm
## Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th># of nodes</th>
<th># of edges</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>roadNet-TX</td>
<td>1,379,917</td>
<td>1,921,660</td>
<td>Road</td>
</tr>
<tr>
<td>roadNet-CA</td>
<td>1,965,206</td>
<td>2,766,607</td>
<td>Road</td>
</tr>
<tr>
<td>roadNet-PA</td>
<td>1,088,092</td>
<td>1,541,898</td>
<td>Road</td>
</tr>
<tr>
<td>Enron Email</td>
<td>36,692</td>
<td>183,831</td>
<td>communication</td>
</tr>
<tr>
<td>Arxiv HEP-PH</td>
<td>34,546</td>
<td>421,578</td>
<td>citation</td>
</tr>
<tr>
<td><strong>Amazon</strong></td>
<td>334,863</td>
<td>925,872</td>
<td>Co-purchasing</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>685,230</td>
<td>7,600,595</td>
<td>Web</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>3,997,962</td>
<td>34,681,189</td>
<td>social</td>
</tr>
<tr>
<td>Youtube</td>
<td>1,134,890</td>
<td>2,987,624</td>
<td>social</td>
</tr>
</tbody>
</table>
Results: Accuracy and Run time

![Accuracy Graph for roadtx](image1)

![Accuracy Graph for roadca](image2)

![Runtime Graph for roadtx](image3)

![Runtime Graph for roadca](image4)
Results: Accuracy and Run time

Dense Subgraph Discovery (DSD): Theory and Applications

SDM 2021 90 / 162
DSP with negative weights

Novel Dense Subgraph Discovery Primitives: Risk Aversion and Exclusion Queries (ECML-PKDD 2019)

T-C-Kakimura-Pachocki

- Exclusion DSD queries
- Risk-averse DSD
DSP with negative weights – Exclusion Queries

Problem

Given a multigraph $G(V, E, \ell)$, where $\ell : E \rightarrow \{1, \ldots, L\} = [L]$ is the labeling function, and $L$ is the number of types of interactions, and an input set $\mathcal{I} \subseteq [L]$ of interactions, how do we find a set of nodes $S$ that (i) induces a dense subgraph, and (ii) does not induce any edge $e$ such that $\ell(e) \in \mathcal{I}$?

Application: Given the daily Twitter interactions, find a dense subgraph in follows and quotes but with no replies.

Approach: Use negative weights (e.g., $-\infty$) for the excluded edge types.
Intuitively, our goal is to find a subgraph $G[S]$ induced by $S \subseteq V$ such that:

1. Its average expected reward $\frac{\sum_{e \in E(S)} w_e}{|S|}$ is large.
2. The associated average risk is low $\frac{\sum_{e \in E(S)} \sigma_e^2}{|S|}$.

We approach the problem as follows:

- For each edge we create two edges:
  1. A positive edge with weight equal to the expected reward, i.e., $w^+(e) = \mu_e$
  2. A negative edge with weight equal to the opposite of the risk of the edge, i.e., $w^-(e) = \sigma_e^2$. 

DSP with negative weights – Risk Averse DSD
Hardness and Greedy

**Theorem**

*The DSP on graphs with negative weights is NP-hard.*

**Reduction from MAX-CUT.**

**Question:** How does Charikar’s greedy algorithm perform?

**Theorem**

Let $G(V, E, w)$, $w : E \rightarrow \mathbb{R}$ be an undirected weighted graph with possibly negative weights. If the negative degree $\text{deg}^-(u)$ of any node $u$ is upper bounded by $\Delta$, then our Algorithm outputs a set whose density is at least $\frac{\rho^*}{2} - \frac{\Delta}{2}$. 
Let $W = \frac{n-4}{3}$. Then, $3W - n < -3$. The degrees of the $n + 4$ nodes are as follows:

- **one node**
  \[3W - n < -3\]
- **$n-2$ nodes**
  \[-2 < 0\]
- **two nodes**
  \[2\epsilon + W\]
A heuristic Greedy

- Run the following greedy for various $C$ values.
- Output the densest subgraph among the densest

[Tsourakakis et al., 2019]

---

**input:** $C > 0$, undirected graph with negative weights $G = (V, E, w)$

**output:** $S$, a dense subgraph of $G$

1. set $G_n \leftarrow G$
2. for $k \leftarrow n$ downto 1
   1.1 let $v$ be the degree vertex in $G_k$
      with smallest score $C_{deg^+}(v) - deg^-(v)$
   1.2 $G_{k-1} \leftarrow G_k \setminus \{v\}$
3. output the densest subgraph among $G_n, G_{n-1}, \ldots, G_1$
## Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>$n$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biogrid</td>
<td>5 640</td>
<td>59 748</td>
</tr>
<tr>
<td>Collins</td>
<td>1 622</td>
<td>9 074</td>
</tr>
<tr>
<td>Gavin</td>
<td>1 855</td>
<td>7 669</td>
</tr>
<tr>
<td>Krogan core</td>
<td>2 708</td>
<td>7 123</td>
</tr>
<tr>
<td>Krogan extended</td>
<td>3 672</td>
<td>14 317</td>
</tr>
<tr>
<td>TMDB</td>
<td>160 784</td>
<td>883 842</td>
</tr>
<tr>
<td>Twitter (Feb. 1)</td>
<td>621 617</td>
<td>(902 834, 387 597, 222 253, 30 018, 63 062)</td>
</tr>
<tr>
<td>Twitter (Feb. 2)</td>
<td>706 104</td>
<td>(1 002 265, 388 669, 218 901, 29 621, 64 282)</td>
</tr>
<tr>
<td>Twitter (Feb. 3)</td>
<td>651 109</td>
<td>(1 010 002, 373 889, 218 717, 27 805, 59 503)</td>
</tr>
<tr>
<td>Twitter (Feb. 4)</td>
<td>528 594</td>
<td>(865 019, 435 536, 269 750, 32 584, 71 802)</td>
</tr>
<tr>
<td>Twitter (Feb. 5)</td>
<td>631 697</td>
<td>(999 961, 396 223, 233 464, 30 937, 66 968)</td>
</tr>
<tr>
<td>Twitter (Feb. 6)</td>
<td>732 852</td>
<td>(941 353, 407 834, 239 486, 31 853, 67 374)</td>
</tr>
<tr>
<td>Twitter (Feb. 7)</td>
<td>742 566</td>
<td>(1 129 011, 406 852, 236 121, 30 815, 68 093)</td>
</tr>
</tbody>
</table>
Experimental findings – Exclusion queries on Twitter

We set $C = 1$, $W = -\infty$:

Degree density for three exclusion queries per each pair of interaction types over the period of the first week of February 2018. ($\alpha$) Follow and mention. ($\beta$) Follow and retweet. ($\gamma$) Mention and retweet.
Experimental findings - Ranging $W, C$

| $C$  | $W$    | $|S^*|$ | $\rho_{\text{retweet}}(S^*)$ | $\rho_{\text{reply}}(S^*)$ |
|------|--------|---------|-------------------------------|-----------------------------|
| 0.1  | 1      | 296     | 63.44                         | -0.75                       |
|      | 5      | 99      | 45.67                         | -0.01                       |
|      | 200 000| 200     | 30.37                         | 0                           |
| 1    | 1      | 346     | 72.70                         | -2.75                       |
|      | 5      | 319     | 68.70                         | -1.29                       |
|      | 200 000| 200     | 30.38                         | 0                           |
| 10   | 1      | 351     | 73.10                         | -3.31                       |
|      | 5      | 351     | 73.10                         | -3.31                       |
|      | 200 000| 200     | 30.37                         | 0                           |

Exploring the effect of the negative weight $-W$ on the excluded edge types for various $C$ values.
Motif-aware DSP

- The k-clique Densest Subgraph Problem (WWW 2015)
- Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling (KDD 2015)
  Mitzenmacher-Pachocki-Peng-T-Xu
**$k$-clique densest subgraph problem**

For any $S \subseteq V$ let

$$c_k(S) = \# \text{ $k$-cliques induced by } S.$$ 

Define *$k$-clique density*

$$\rho_k(S) = \frac{c_k(S)}{s}, \quad k \geq 2, \ s = |S|$$

Solve the *$k$-clique DSP*

$$\rho_k(S^*) = \rho_k^* = \max_{S \subseteq V} \rho_k(S)$$

E.g. $c_2(\triangle) = 3$
Triangle densest subgraph problem

We shall refer to the 3-clique DSP as the triangle densest subgraph problem.

$$\max_{S \subseteq V} \tau(S) = \frac{t(S)}{s}$$

How different can the densest subgraph be from the triangle densest subgraph? Radically different, e.g., $G = K_{n,n} \cup K_3$.

What happens on real-data? Can we solve the triangle DSP in polynomial time? The $k$-clique DSP?
Triangle densest subgraph problem

**Theorem**

There exists an algorithm which solves the TDSP and runs in $O\left(\frac{m^3}{2} + nt + \min(n, t)^3\right)$ time.

Furthermore,

**Theorem**

We can solve the $k$-clique DSP in polynomial time for any $k = \Theta(1)$.

**Computation** involves

- Enumerate the set $C_k$ of $k$-cliques in $G$
- Maximum flow on an appropriate network $\mathcal{N}(\{s, t\} \cup V \cup C_k, A)$
Epinions social network

Notation note: Here $f_e(S) = \frac{e(S)}{|S| \choose 2}$

<table>
<thead>
<tr>
<th># nodes</th>
<th>75,877</th>
</tr>
</thead>
<tbody>
<tr>
<td># edges</td>
<td>405,739</td>
</tr>
</tbody>
</table>

Output size $|S^*_k|$ for various $k$:

- $|S^*_1| = 1012, f_e = 0.12, 2.37$ sec
- $|S^*_2| = 432, f_e = 0.26, 15.8$ sec
- $|S^*_3| = 235, f_e = 0.40, 85.7$ sec
- $|S^*_4| = 172, f_e = 0.50, 37,939.6$ sec

<table>
<thead>
<tr>
<th>$k$</th>
<th>$c_k$</th>
<th>$T_k$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-clique</td>
<td>1.6M</td>
<td>1.6</td>
</tr>
<tr>
<td>4-clique</td>
<td>5.8M</td>
<td>4.8</td>
</tr>
<tr>
<td>5-clique</td>
<td>17.4M</td>
<td>13.4</td>
</tr>
</tbody>
</table>

Dense Subgraph Discovery (DSD): Theory and Applications
CA-Astro and Email networks

<table>
<thead>
<tr>
<th># nodes</th>
<th>18 772</th>
</tr>
</thead>
<tbody>
<tr>
<td># edges</td>
<td>198 050</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># nodes</th>
<th>234 352</th>
</tr>
</thead>
<tbody>
<tr>
<td># edges</td>
<td>383 111</td>
</tr>
</tbody>
</table>

Output size $|S_k^*|$ vs. $k$

- $|S_1^*| = 1K$, $f_e = 0.09$, 1.2 sec
- $|S_2^*| = 76$, $f_e = 0.8$, 12.3 sec
- $|S_3^*| = 62$, $f_e = 0.96$, 155.1 sec
- $|S_4^*| = 62$, $f_e = 0.96$, 2,107.2 sec
- $|S_5^*| = 76$, $f_e = 0.8$, 12.3 sec
- $|S_6^*| = 62$, $f_e = 0.96$, 155.1 sec

<table>
<thead>
<tr>
<th>k</th>
<th>$c_k$</th>
<th>$T_k$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-clique</td>
<td>1.4M</td>
<td>0.6</td>
</tr>
<tr>
<td>4-clique</td>
<td>9.5M</td>
<td>3.9</td>
</tr>
<tr>
<td>5-clique</td>
<td>65M</td>
<td>27.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>$c_k$</th>
<th>$T_k$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-clique</td>
<td>0.4M</td>
<td>0.4</td>
</tr>
<tr>
<td>4-clique</td>
<td>1M</td>
<td>0.9</td>
</tr>
<tr>
<td>5-clique</td>
<td>2.6M</td>
<td>1.9</td>
</tr>
</tbody>
</table>
**Time evolving networks**

**Patents citation network** that spans 37 years, specifically from January 1, 1963 to December 30, 1999.
Densest subgraph sparsifiers

Definition: “A hypergraph is a generalization of a graph in which an edge can connect any number of vertices.”

Densest subgraph sparsifier theorem

Let $\mathcal{H}(V, E_\mathcal{H})$ be a hypergraph, $\epsilon > 0$.
Let $E' \subseteq E_\mathcal{H}$ be a sample of $\frac{6n \log n}{\epsilon^2}$ edges chosen uniformly at random.
Solving the DSP on $E'$ results in a $(1 + \epsilon)$ approximation to $\rho^*$ whp.
Densest subgraph sparsifiers

Some hypergraphs of interest

Technical difficulty.
Notice that taking Chernoff bounds and a union bound does not work since by Chernoff the failure probability is $1/poly(n)$ whereas there exists an exponential number of potential bad events.
Densest subgraph sparsifiers

**Corollary:** Single-pass, \((1 + \epsilon)\) semi-streaming algorithm! Just keep \(O(n \log n / \epsilon^2)\) edges.

Same sparsification result, with efficient semi-streaming implementation [Esfandiari et al., 2015, McGregor et al., 2015]

Let \(p = \frac{6n \log n}{|E_H| \epsilon^2}\).

**Expected space** reduction is \(O\left(\frac{1}{p}\right)\).

**Expected speedup** for maximum flow computation \(O\left(\frac{1}{p^2}\right)\)

<table>
<thead>
<tr>
<th>(k)</th>
<th>Avg. Speedup</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3\times</td>
<td>(\geq 95%)</td>
</tr>
<tr>
<td>3</td>
<td>23.8\times</td>
<td>(\geq 98%)</td>
</tr>
<tr>
<td>4</td>
<td>302 \times</td>
<td>(\geq 99%)</td>
</tr>
<tr>
<td>5</td>
<td>24,000\times</td>
<td>(\approx 100%)</td>
</tr>
</tbody>
</table>
Zooming-in on Epinions network

Epinions graph, $n = 75,877$, $m = 405,739$

Edge density $f_e$

Output size $|S|$

Accuracy $\rho_k(S)/\hat{\rho}_k$

Speedup $\times$

**KCl@st++** a recent algorithm that combines Frank-Wolfe-like algorithm with fast clique enumeration [Sun et al., 2020].
Densest subgraph problem: Dynamic graphs
Graph Semi-Streaming model

Model

- Stream of edges (possibly adversarial order; insert only vs dynamic/strict turnstile)
- Space constraint: $\tilde{O}(n)$ space available
- As few passes as possible
- Approximation guarantees
Streaming DSP – BKV algorithm

input: undirected graph $G = (V, E), \epsilon > 0$

output: $\tilde{S} \subseteq V$

1. $S, \tilde{S} \leftarrow V$
2. while $S \neq \emptyset$ do:
   2.1 $A(S) \leftarrow \{i \in S | \text{deg}_S(i) \leq 2(1 + \epsilon)\rho(S)\}$
   2.2 $S \leftarrow S \setminus A(S)$
   2.3 if $\rho(S) > \rho(\tilde{S})$:
      2.4 $\tilde{S} \leftarrow S$
3. output $\tilde{S}$

- The above algorithm is due to Bahmani, Kumar, Vassilvitski [Bahmani et al., 2012]
- Space required $\tilde{O}(n)$ (keeping degrees), $(2 + 2\epsilon)$ approx guarantee, $O(\log n/\epsilon)$ passes over the edge stream.
Basic idea: Uniform sampling works, i.e., independently sample each edge in $G$ with probability $\frac{cn\log n}{\epsilon^2 m}$ [Mitzenmacher et al., 2015, Esfandiari et al., 2015, McGregor et al., 2015]

Theorem ([Esfandiari et al., 2015, McGregor et al., 2015])

There exists a single pass semi-streaming algorithm that gives a $(1 + \epsilon)$ approximation to the DSP using $\tilde{O}(n)$ space with high probability.
Dynamic networks

Model:

- Update time
- Query time
- Approximation guarantees
- Linear space allowed $O(n + m)$

Remark: The goal of a dynamic graph algorithm is to support query and update operations as quickly as possible, instead of computing from scratch (e.g., linear time update)!
Dynamic DSP [Bhattacharya et al., 2015]

We say that an algorithm is a fully-dynamic $\gamma$-approximation algorithm for the densest subgraph problem if it can process the following operations.

- **Initialize**($n$): Initialize the algorithm with an empty $n$-node graph.
- **Insert**($u, v$): Insert edge $(u, v)$ to the graph.
- **Delete**($u, v$): Delete edge $(u, v)$ from the graph.
- **QueryValue**: Output a $\gamma$-approximate value of $\rho^*(G) = d^*$
Fully dynamic $(4 + \epsilon)$-approximation algorithm
\(\tilde{O}(n)\) space [Bhattacharya et al., 2015]

**Theorem**

- Let \(\epsilon \in (0, 1), \lambda > 1\) constant and \(T = \lceil n^\lambda \rceil\).
- There is an algorithm that processes the first \(T\) updates in the dynamic stream such that:
  - It uses \(\tilde{O}(n)\) space (**Space efficiency**)
  - It maintains a value \(\text{OUTPUT}^{(t)}\) at each \(t \in [T]\) such that for all \(t \in [T]\) \(\text{whp}\)

\[
\frac{\text{OPT}^{(t)}}{4 + \Theta(\epsilon)} \leq \text{OUTPUT}^{(t)} \leq \text{OPT}^{(t)}.
\]

Also, the total amount of computation performed while processing the first \(T\) updates in the dynamic stream is \(O(T \ poly \log n)\). (**Time efficiency**)
Accurate near-real time graph analytics with sublinear space [Bhattacharya et al., 2015]
Dynamic DSP – State-of-the-art

**Theorem ([Sawlani and Wang, 2020])**

There exists a deterministic fully dynamic

- (1 + $\epsilon$)-approximation algorithm,
- $O(\log n/\epsilon)$ worst-case query time
- $O(\log^5 n \cdot \epsilon^{-7})$ worst-case update times of per edge insertion or deletion,
- that can output the approximate DS in $O(|DS| + \log n)$.

**Remark:** Basic idea relies on the load balancing perspective of the DSP!
Sawlani-Wang approach

\[ \text{\textbf{Dual}(G)} \]

minimize \[ \max \text{load}_v \]
subject to \[ f_e(u) + f_e(v) = 1, \quad \forall e = uv \in E \]
\[ \text{load}_v = \sum_{e \ni v} f_e(v), \quad \forall v \in V \]
\[ f_e(u) \geq 0, f_e(v) \geq 0, \quad \forall e = uv \in E \]

Therefore, the following is a local optimality condition:

\[ f_e(u) + f_e(v) = 1 \]
\[ \text{load}_u \leq \text{load}_v \]
\[ \forall e = uv \in E, f_e(u) > 0 \]
Sawlani-Wang approach

S-W introduce the following notion: \( \eta \)-local optimality condition:

\[
\begin{align*}
    f_e(u) + f_e(v) &= 1 \\
    \text{load}_u &\leq \text{load}_v + \eta \\
    \forall e = uv \in E, f_e(u) &> 0
\end{align*}
\]

The following result:

\( \eta \)-local approx optimality \( \Rightarrow \left( 1 + \sqrt{\frac{\eta \log n}{\text{OPT}}} \right) \) - global approx

Key idea

- orient edges so that maximum in-degree is minimized
- “flip” edges till local \( \eta \)-optimality.
Epasto et al. [Epasto et al., 2015]

- Epasto, Lattanzi, and Sozio have studied the dynamic DSP assuming arbitrary insertions, and random deletions.
- They obtain weaker theoretical results \((2 + \epsilon) \text{ approx in } O(\log^4(n)/\epsilon^4)\) update time, and linear space.

DSP and DSD variants
Directed DSP – Exact algorithms

- directed graph $G = (V, E)$
- $S, T \subseteq V$ (not necessarily disjoint)
- degree density:

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

- **Directed-DSP**: $\max_{S, T \subseteq V} \rho(S, T)$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time complexity</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Charikar, 2000]</td>
<td>$\Omega(n^6)$</td>
<td>LP</td>
</tr>
<tr>
<td>[Khuller and Saha, 2009]</td>
<td>$O(n^2 t_{max-flow})$</td>
<td>Max flow preprocessing+</td>
</tr>
<tr>
<td>[Ma et al., 2020]</td>
<td>$O(kt_{max-flow})$</td>
<td>Max Flow</td>
</tr>
</tbody>
</table>

Ma et al. examine $k$ values of possible ratios $\frac{|S|}{|T|}$, and $k \ll n^2$ on real data.
### Directed graphs – Approximation Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time complexity</th>
<th>Approx.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Kannan and Vinay, 1999]</td>
<td>$\Omega(sn^3)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>[Khuller and Saha, 2009]</td>
<td>$O(n + m)$</td>
<td>$&gt; 2$ (paper claimed 2)</td>
</tr>
<tr>
<td>Corrected-KS</td>
<td>$O(n^2(n + m))$</td>
<td>2</td>
</tr>
<tr>
<td>(by [Ma et al., 2020])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Ma et al., 2020]</td>
<td>$O(\sqrt{m(n + m)})$</td>
<td>2</td>
</tr>
</tbody>
</table>

- The Kannan-Vinay algorithm is based on SVD; $s$ is the size sample of rows from the adjacency matrix.

- Despite being an $O(\log n)$-approximation, the algorithm has found recently applications on fair dense subgraphs [Anagnostopoulos et al., 2020].
Fair dense subgraphs

Each node has a color, **red** and **black**.

The densest subgraph problem can be expressed as:

$$\max_{x \in \{0,1\}^n} \frac{x^T Ax}{x^T x}.$$  

The **fair** DSP asks for the densest subgraph such that:

$$\sum_{\text{node } i \text{ is red}} x_i = \sum_{\text{node } i \text{ is blue}} x_i$$
Fair dense subgraphs

• define the (unit 2-norm) vector

\[ f_i = \begin{cases} 
\frac{1}{\sqrt{n}} & \text{if node } i \text{ is red} \\
-\frac{1}{\sqrt{n}} & \text{if node } i \text{ is blue,}
\end{cases} \]

• Fair DSP = DSP + CONSTRAINT \( x^T f = 0 \)

• Key idea: spectral relaxation:

\[
\max_{x \in \{0,1\}^n} \frac{2x^T (I - ff^T)A(I - ff^T)x}{x^Tx}.
\]

• Solution: Fair eigenvector
Robustness to vertex/edge failure

Densest subgraphs may **not be robust** to vertex/edge failure
Densest $k$-connected subgraphs

Bonchi, García-Soriano, Miyauchi, and T. introduce the following problem [Bonchi et al., 2020]:

**Problem (Densest $k$-vertex-connected subgraph)**

**Input:** $G = (V, E)$ and $k \in \mathbb{Z}_{>0}$

**Output:** $S \subseteq V$ that maximizes $\rho_G(S) := \frac{e(S)}{|S|}$ under $\kappa(G[S]) \geq k$

Here, the *vertex connectivity* of $G$, denoted by $\kappa(G)$, is the smallest cardinality of a vertex separator of $G$ if $G$ is not a clique and $|V| - 1$ otherwise

- BGMT provide an algorithmic understanding of Mader’s theorem, bicriteria approximation algorithm, and a $\frac{19}{6}$-approximation algorithm.
Overlapping dense subgraphs with limited overlap
[Balalau et al., 2015]

problem formulation ($k$, $\alpha$-DSLO) ([Balalau et al., 2015])

- given graph $G = (V, E)$, and parameters $k$ and $\alpha$
- find $k$ subgraphs $S_1, \ldots, S_k$
- in order to maximize

$$\sum_{i=1}^{k} d(S_i)$$

subject to

$$\frac{|S_i \cap S_j|}{|S_i \cup S_j|} \leq \alpha, \text{ for all } 1 \leq i < j \leq k$$

Theorem: The ($k$, $\alpha$-DSLO) problem is NP-hard.
**MinAndRemove** [Balalau et al., 2015]

*Example taken from:* Balalau et al. slides

Find $k = 3$ subgraphs that have an overlap of at most $\alpha = 0.25$.

- Find a densest subgraph
- Make it minimal
MinAndRemove [Balalau et al., 2015]

Find \( k = 3 \) subgraphs that have an overlap of at most \( \alpha = 0.25 \).

- Find a densest subgraph
- Make it minimal
- Remove 75% of the subgraph’s nodes
MinAndRemove [Balalau et al., 2015]

Example taken from: Balalau et al. slides

Find $k = 3$ subgraphs that have an overlap of at most $\alpha = 0.25$.

- Find a densest subgraph
- Make it minimal
- Remove 75% of the subgraph’s nodes
- Iterate

![Diagram of a network with one highlighted subgraph and its density indicated as 2.](image)
MinAndRemove [Balalau et al., 2015]

Example taken from: Balalau et al. slides

Find \( k = 3 \) subgraphs that have an overlap of at most \( \alpha = 0.25 \).

- Find a densest subgraph
- Make it minimal
- Remove 75% of the subgraph’s nodes
- Iterate

No guarantees in the general case.

![Diagram of a network with labeled nodes and edges with Density = 2]
Question: Given a collection of graphs \( \{ G_1, \ldots, G_k \} \) on the same set of nodes \( V \), does there exist a set of nodes \( S \subseteq V \) such that \( G_i[S] \) is dense for all \( i \)?

[Charikar et al., 2018, Semertzidis et al., 2018, Jethava et al., 2013]
Common dense subgraph

[Semertzidis et al., 2018]

- (Common-DSP-MM) \( \min_{i \in [k]} \min\text{-deg}(G_i[S]) \)
- (Common-DSP-MA) \( \min_{i \in [k]} \frac{|E(G_i(S))|}{|S|} \)
- (Common-DSP-AM) \( \sum_{i \in [k]} \min\text{-deg}(G_i[S]) \)
- (Common-DSP-AA) Average \( k \) graphs, solve DSP
Common dense subgraphs - Results

- Common-DSP-MM can be solved optimally with greedy algorithm in linear time.
- Common-DSP-AA reduces to densest subgraph problem.

□ [Charikar et al., 2018] Common-DSP-MA (min of average degree) cannot be approximated to within a factor of $2^{\log^{1-\epsilon} n}$ unless $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog} n})$, via reduction from $\text{MinRep}$.

□ Common-DSP-AM (average of min degree) is hard to approximate within a factor of $n^{1-\epsilon}$ via reduction from $\text{maximum independent set}$.
Problem variants

- K-core is solvable in linear time [Batagelj and Zavernik, 2003], and widely used in graph analysis such as community detection [Fang et al., 2017, Chen et al., 2019].
- K-plex, [Xiao et al., 2017] find maximum $k$-plex in $c^n n^{O(1)}$ time with $c < 2$ only depend on $k$; [Zhou et al., 2020] enumerates maximal $k$-plexes.
- K-truss, [Cohen, 2008], under dynamic setting [Huang et al., 2014].
Problem variants - optimal quasi-clique

- The problem is defined as $\max_{S \subseteq V} e[S] - \alpha \left( \frac{|S|}{2} \right)$, where $\alpha \in (0, 1)$ [Tsourakakis et al., 2013].

- Simple greedy peeling algorithm and local search method are given for getting approximate solutions [Tsourakakis et al., 2013].

- Recently, [Konar and Sidiropoulos, 2020] proved heavy-tailed degree distribution and large global clustering coefficient imply the existence of neighborhoods of non-trivial sizes possessing high edge-density.
Bipartite graphs: For $G(L \cup R, E), S \subseteq L, T \subseteq R$:

$$\frac{e(S, T)}{\sqrt{|S||T|}}$$

$(p, q)$-biclique densest subgraph problem
[Mitzenmacher et al., 2015].

biclique dense subgraphs with hierarchical relations
[Sarıyüce and Pinar, 2018].

[Kuroki et al., 2020] Dense subgraph problem with oracle that only return noisy weight sum of edge subset.

...
Some open problems

- **Flowless:** Prove (or disprove) the following conjecture:

  \[ \text{Flowless is a } 1 + O(T^{-1/2})\text{-approximation algorithm.} \]

- **Negative weights.** Improved algorithms for the DSP on graphs with negative weights under reasonable assumptions on the input.

- **Dynamic graphs:** Can we improve the near-optimal algorithm of Sawlani-Wang?

- **Multipartite DSP:** How do we find multipartite dense subgraphs in any \( G \)?

- **Stronger results for fair DSP.**
Acknowledgements

Questions?

Aris Gionis

Atsushi Miyauchi


Finding dense subgraphs with size bounds.

Computational complexity and information asymmetry in financial products.

The multiplicative weights update method: a meta-algorithm and applications.
Densest subgraph in streaming and mapreduce.

Finding subgraphs with maximum total density and limited overlap.
In International Conference on Web Search and Data Mining (WSDM),
pages 379–388.

An o(m) algorithm for cores decomposition of networks.
CoRR, cs.DS/0310049.


Detecting high log-densities: an $o(n^{1/4})$ approximation for densest $k$-subgraph.

In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 201–210. ACM.


Finding a maximum density subgraph.
Technical report, University of California at Berkeley.

Clique is hard to approximate within $n^{(1 - \epsilon)}$.

Fraudar: Bounding graph fraud in the face of camouflage.
In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16, page 895904, New York, NY, USA. Association for Computing Machinery.

Adaptive epileptic seizure prediction system.
IEEE Transactions on Biomedical Engineering, 50(5).


Online dense subgraph discovery via blurred-graph feedback.
ArXiv, abs/2006.13642.

Flowscope: Spotting money laundering based on graphs.

Efficient algorithms for densest subgraph discovery on large directed graphs.


Peeling bipartite networks for dense subgraph discovery.
In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18, page 504512, New York, NY, USA. Association for Computing Machinery.

Finding the hierarchy of dense subgraphs using nucleus decompositions.

Near-optimal fully dynamic densest subgraph.


