Algorithmic Primitives for Finding Dense Structures in Rich Graph Data

Charalampos (Babis) E. Tsourakakis

University of Crete & ICS FORTH

SEA 2026

Thank You for the Invitation

Martin Aumüller

Irene Finocchi

Holger Dell

What this talk is about . . .

given a graph (network), static or dynamic

(social, biological, financial , information networks, . . .)

whose nodes & edges carry rich side information

find a subgraph that . . .

. . . is densely connected

. . . and matches what we actually want:

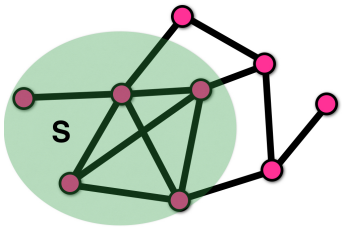
near-clique · diverse · anomalous · aligned with side info

Key Focus: the **Densest Subgraph Problem** (DSP)

Outline

- **Related work**
- **Rich graph data: from applications to formulations**
- **Some Algorithmic Results**
  - Flowless
  - k-clique DSP
  - AntiBenford Subgraphs
- **Conclusion**

Densest subgraph problem

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

$\rho(S) = \frac{7}{5}$

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

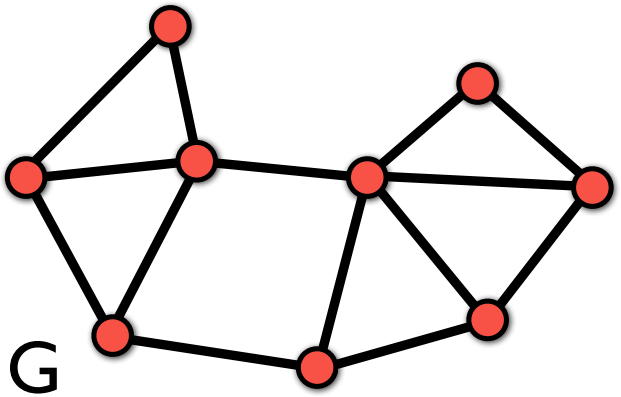
  Also, there exists a 2-approximation algorithm which uses linear space $O(n+m)$ and runs in linear time $O(n+m)$ due to [Charikar, 2000]

  "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), $|S| \geq k$ (DalkS), $|S| \leq k$ (DamkS) [Bhaskara et al., 2010, Khuller and Saha, 2009, Andersen and Chellapilla, 2009]

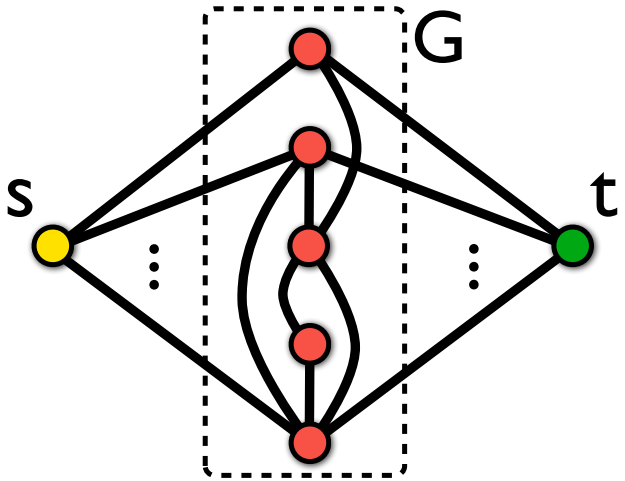
Goldberg's algorithm for densest subgraph

- consider first degree density $d(S) = 2\rho(S)$



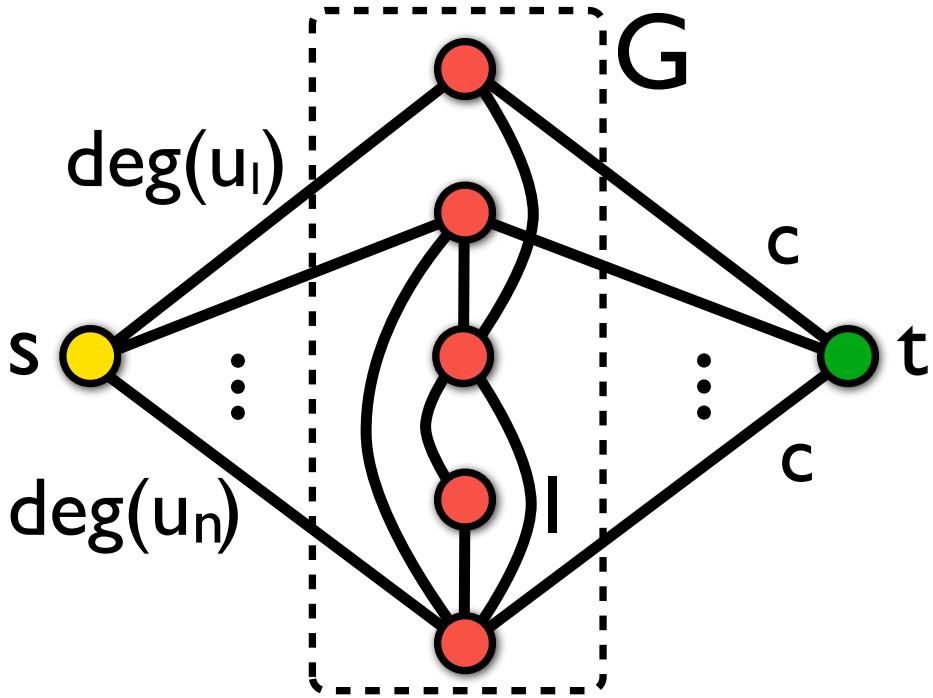
- is there a subgraph S with $d(S) \geq c$?
- transform to a min-cut instance

- on the transformed instance:
- is there a cut smaller than a certain value?



Goldberg's algorithm for densest subgraph

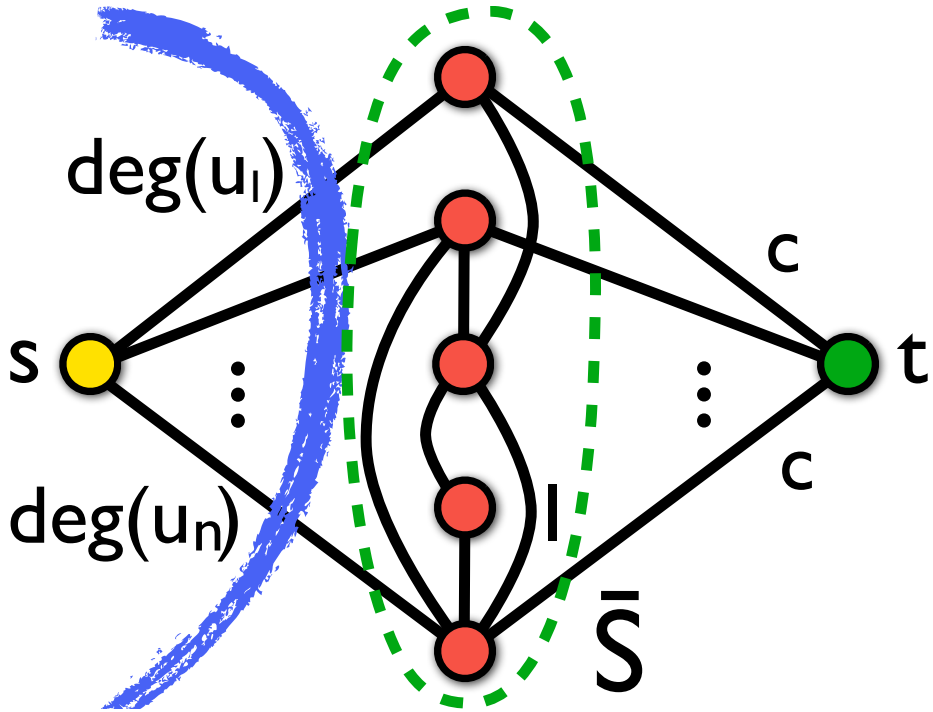
- transform to min-cut instance



- is there S s.t. $d(S) \geq c$?

Goldberg's algorithm for densest subgraph

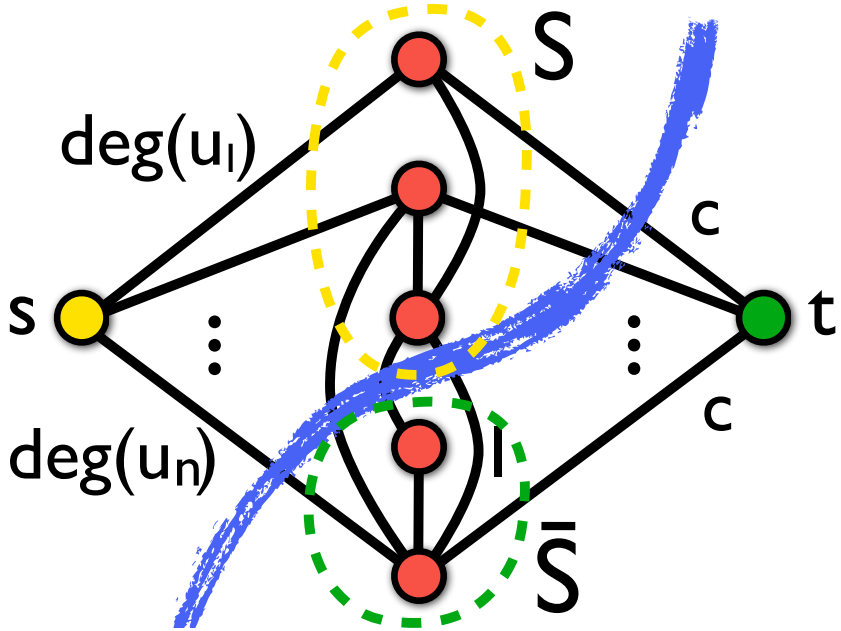
- transform to a min-cut instance



- if not, cut size is $2|E|$.
- a cut of value $2|E|$ always exists for source s alone, i.e., for $S = \emptyset$

Goldberg's algorithm for densest subgraph

- transform to a min-cut instance



- if yes, if min cut achieved for $S \neq \emptyset$
- Specifically, s-side of the cut is $s \cup S$ and this gives a feasible subset of nodes in G

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 [Gallo et al., 1989]

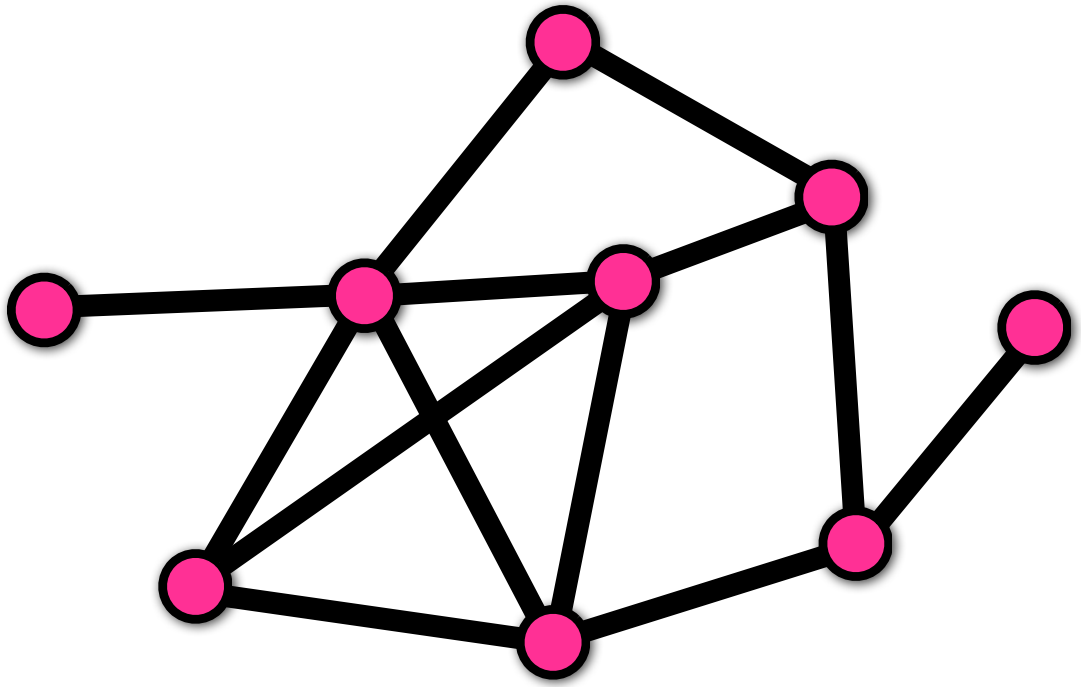
densest subgraph problem – discussion

- Goldberg's algorithm polynomial algorithm, but
- Max flows are (still) expensive in practice despite theoretical progress [Orlin, 2013, Chen et al., 2025a]
- not scalable for large graphs (millions of vertices / edges)

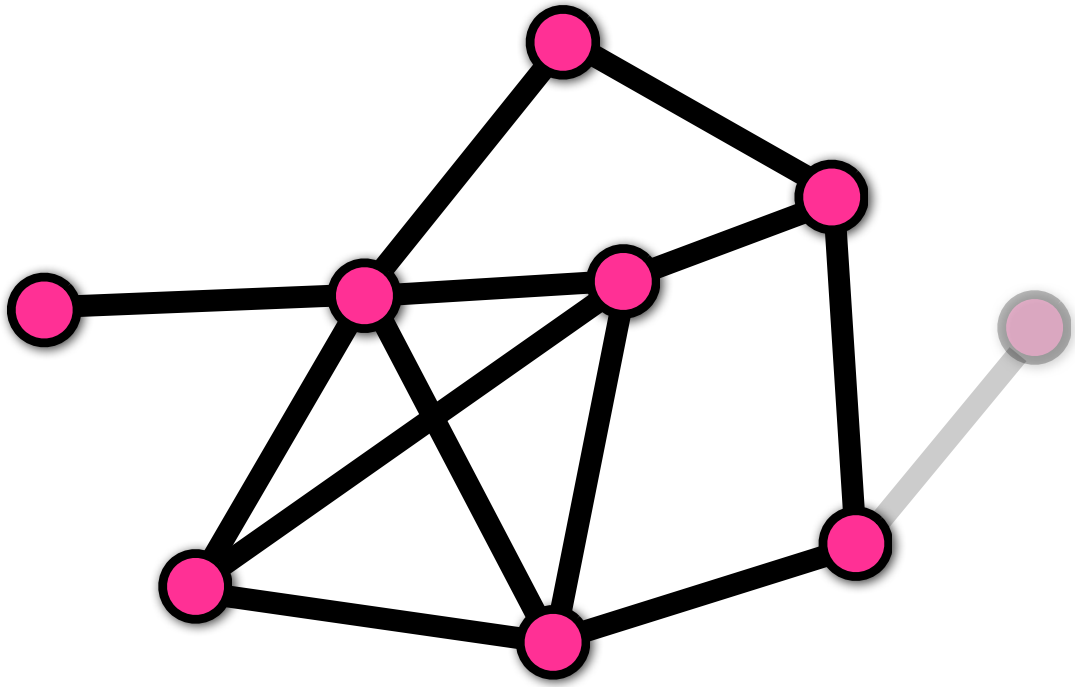
- faster algorithm due to [Charikar, 2000]
- greedy and simple to implement
- approximation algorithm...

- ...and a new family of greedy algorithms.

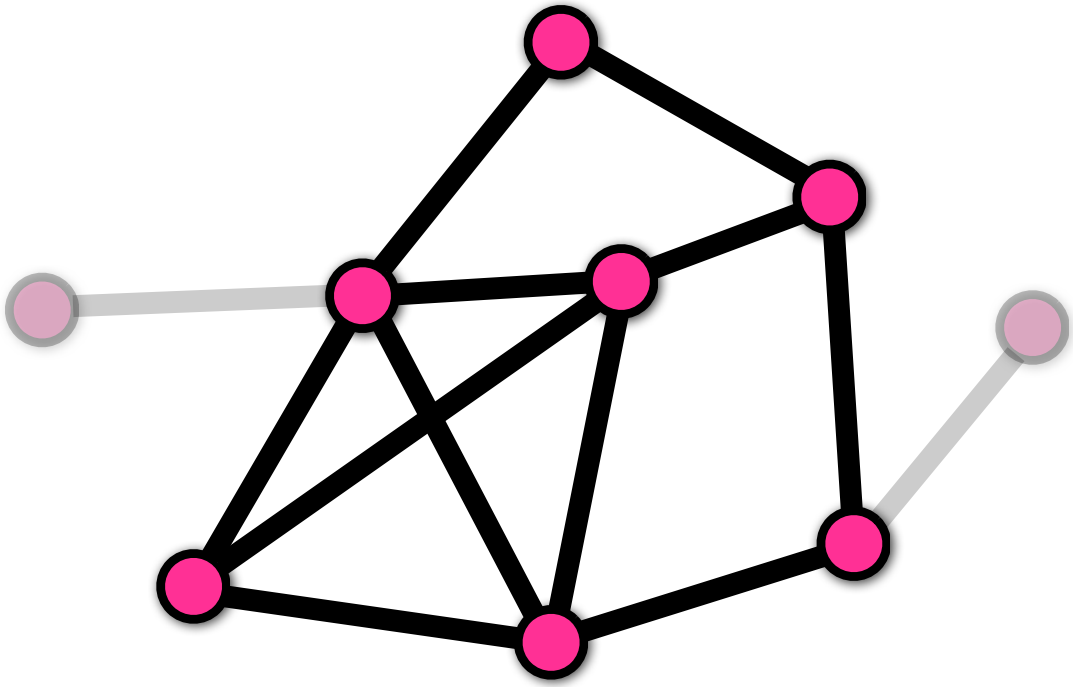
greedy algorithm for densest subgraph — example



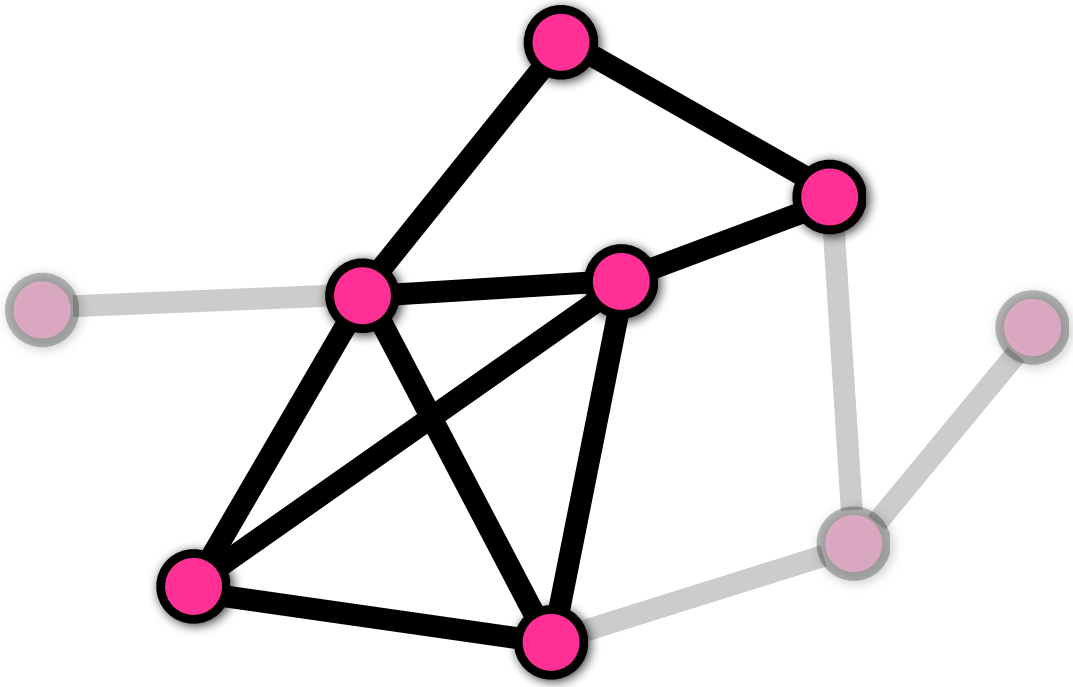
greedy algorithm for densest subgraph — example



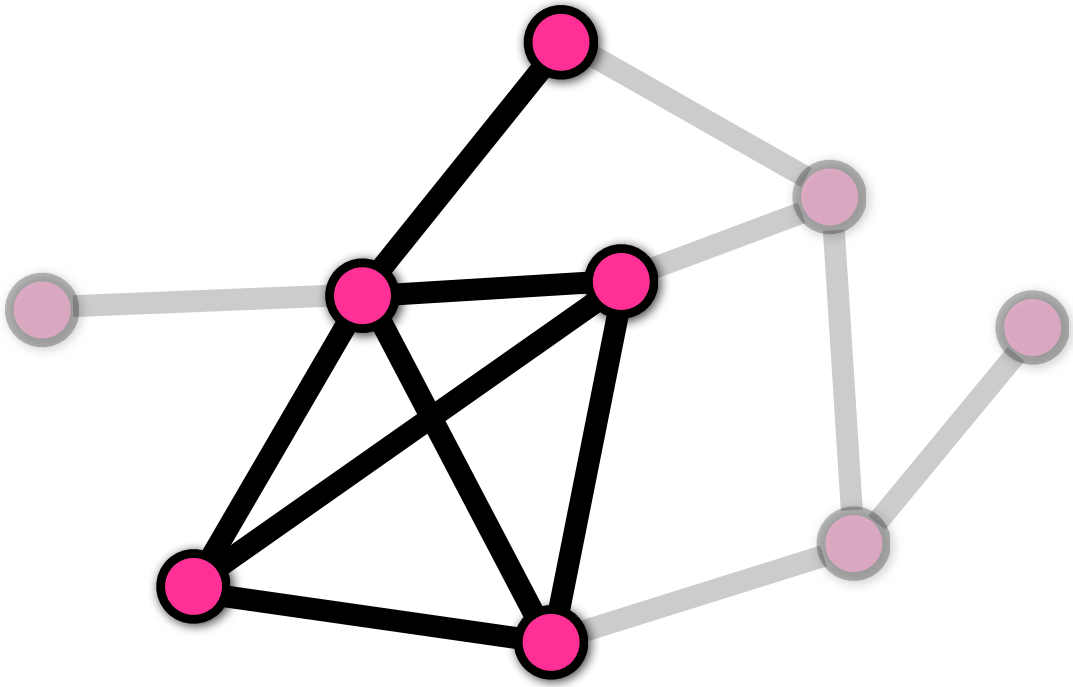
greedy algorithm for densest subgraph — example



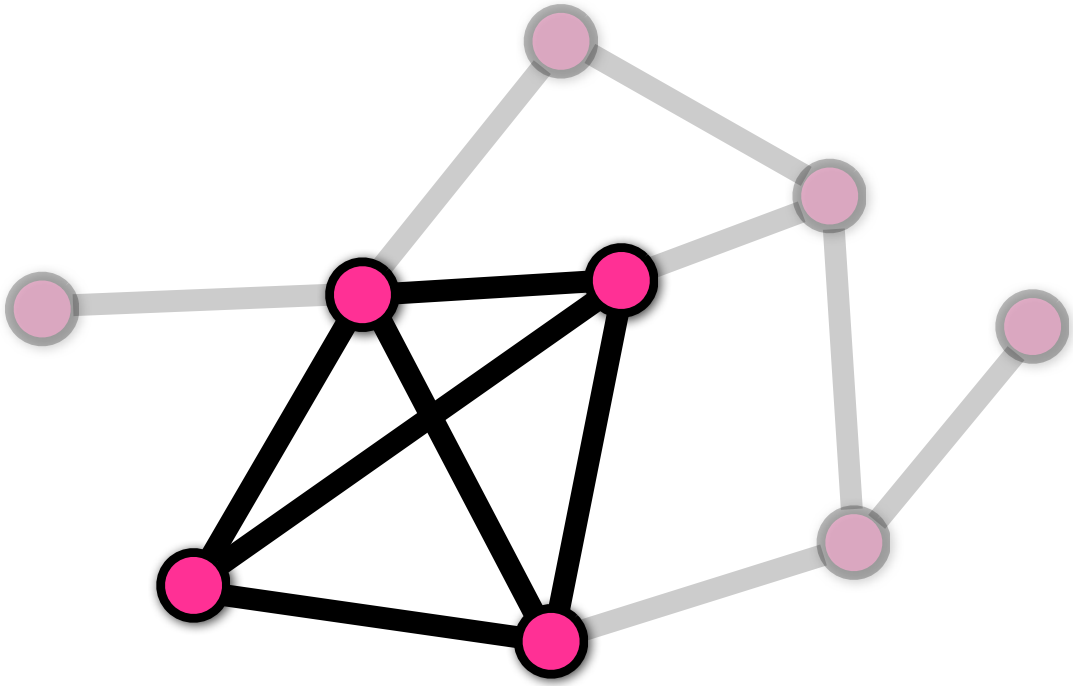
greedy algorithm for densest subgraph — example



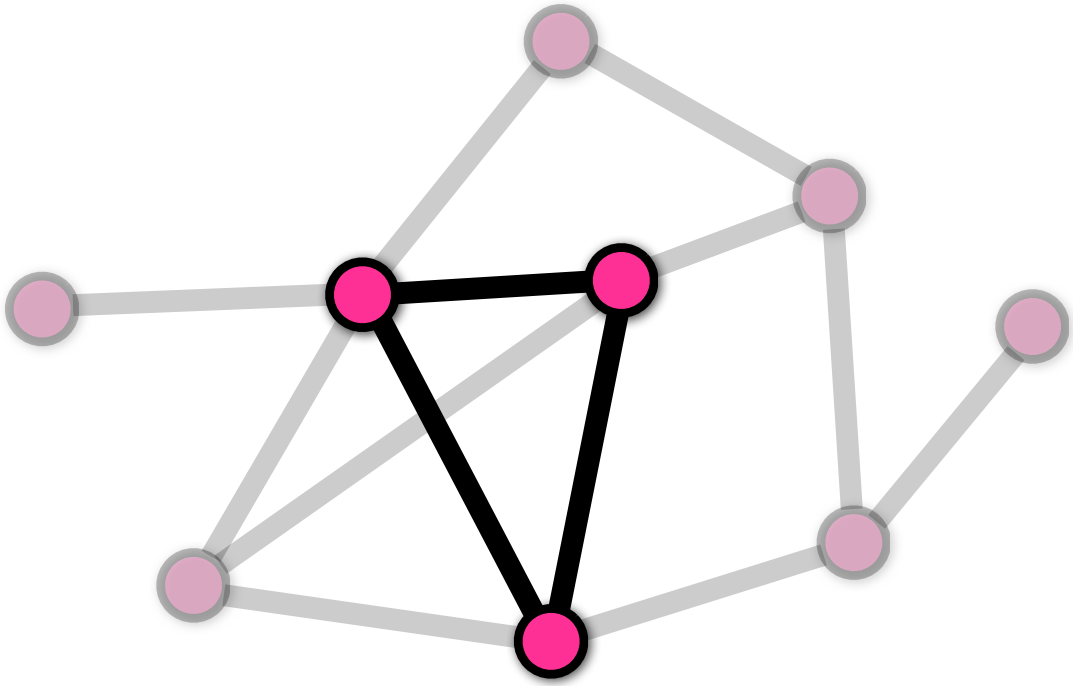
greedy algorithm for densest subgraph — example



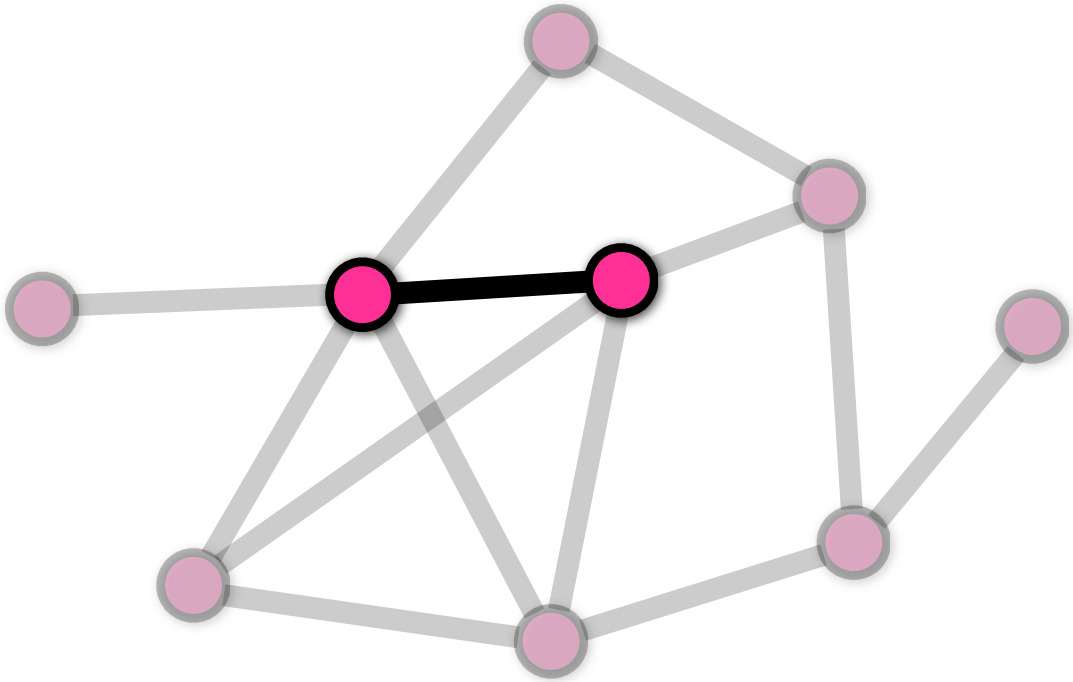
greedy algorithm for densest subgraph — example



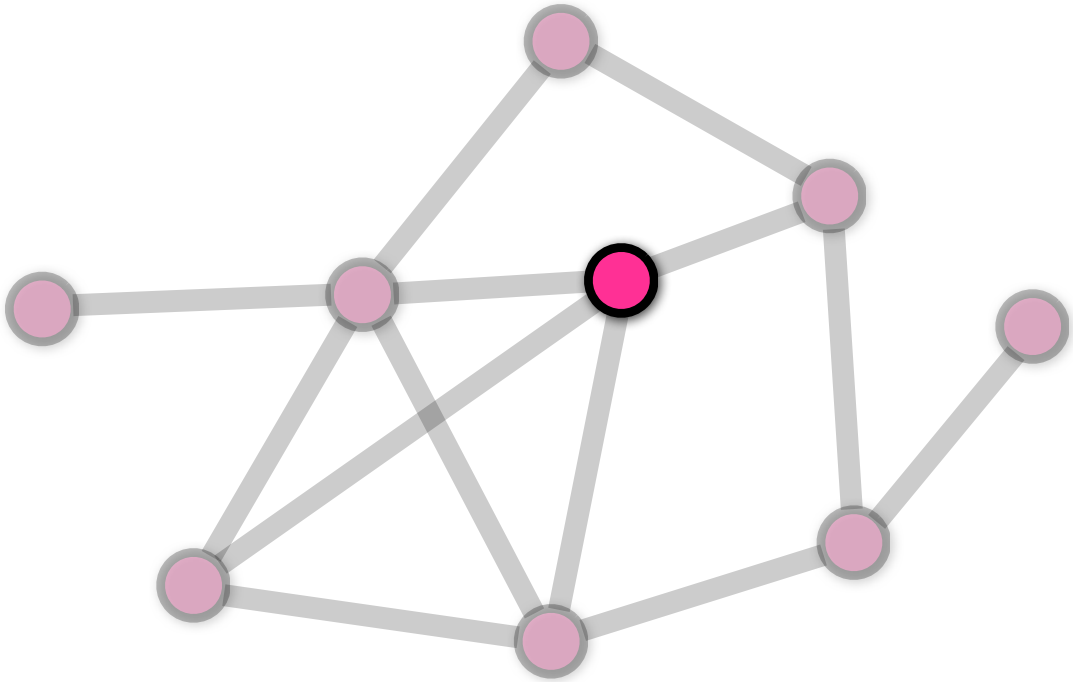
greedy algorithm for densest subgraph — example



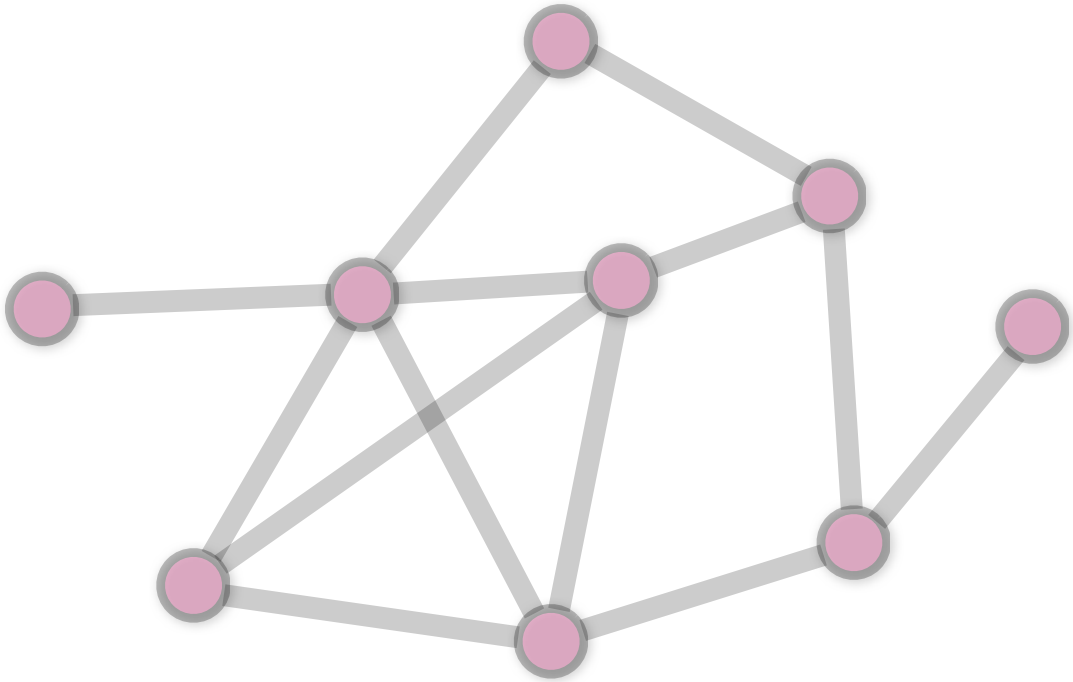
greedy algorithm for densest subgraph — example



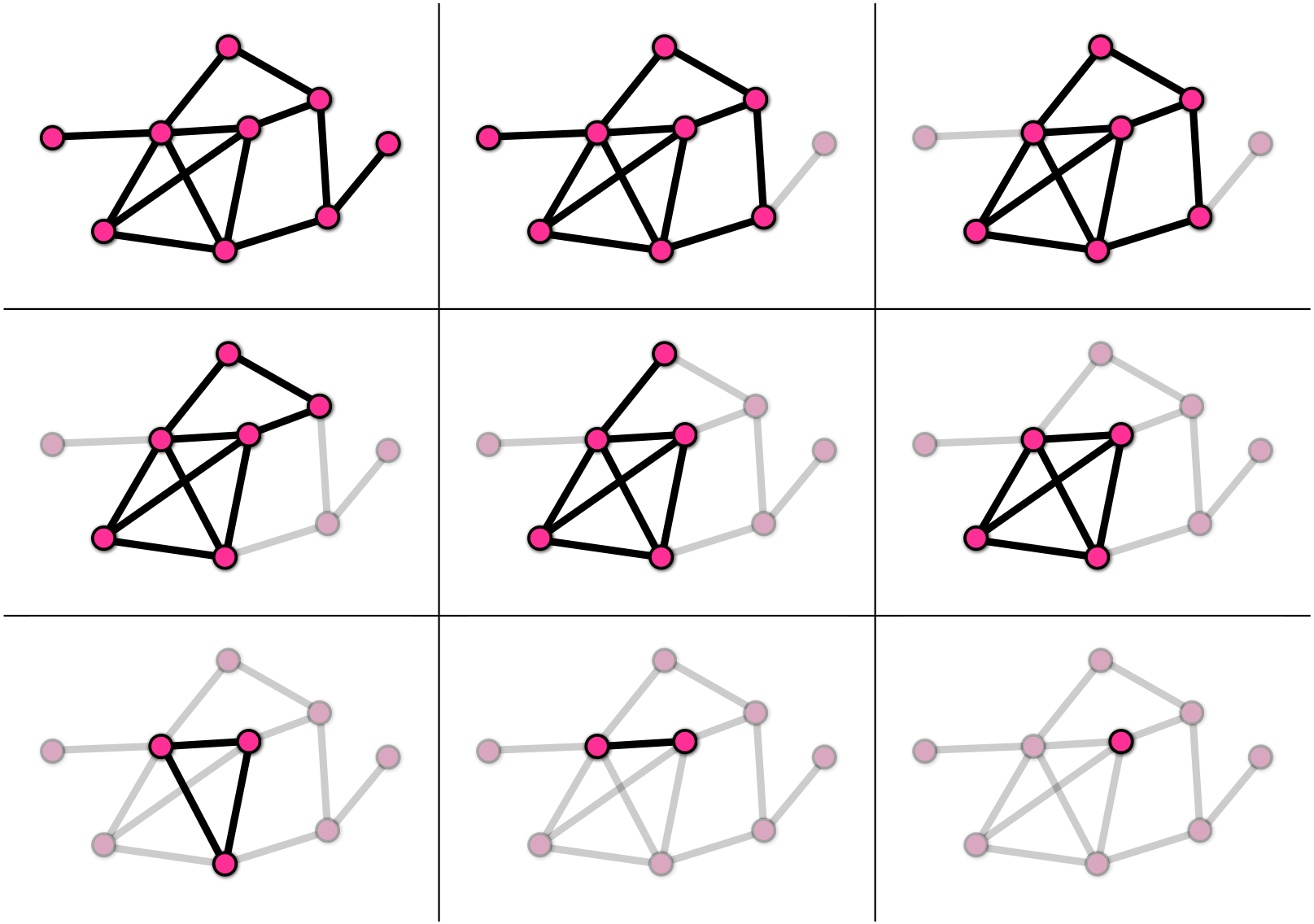
greedy algorithm for densest subgraph — example



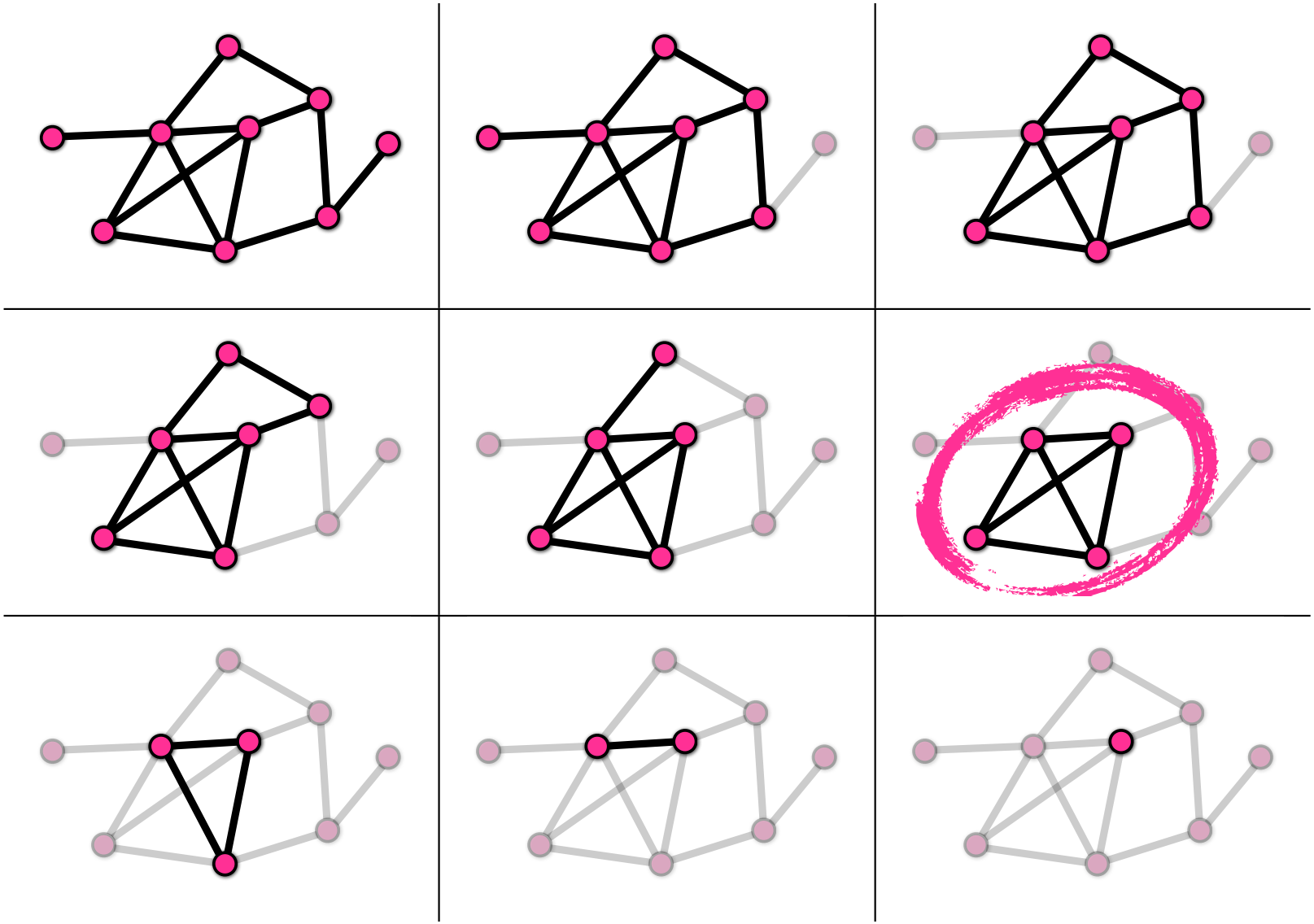
greedy algorithm for densest subgraph — example



greedy algorithm for densest subgraph — example



greedy algorithm for densest subgraph — example



the greedy algorithm

Theorem

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

- runs in linear time $\mathcal{O}(n+m)$ on unweighted graphs
- everything goes through for weighted graphs
  using heaps: $\mathcal{O}(m + n \log n)$

Challenge #1: (Near-) Exact and Practical

Can we combine the optimality guarantees of maximum-flow methods with the scalability and simplicity of greedy peeling?
[Boob et al., 2020]

Concrete Computational Challenges

Challenge #2: Massive Data

Can we solve the densest subgraph problem when the graph is too large to fit in memory and arrives as a stream?[Mitzenmacher et al., 2015]

See also
[Bahmani et al., 2012, Esfandiari et al., 2015, McGregor et al., 2015]

Challenge #3: Evolving Graphs

Can we solve the densest subgraph in the dynamic graph model?
[Bhattacharya et al., 2015]

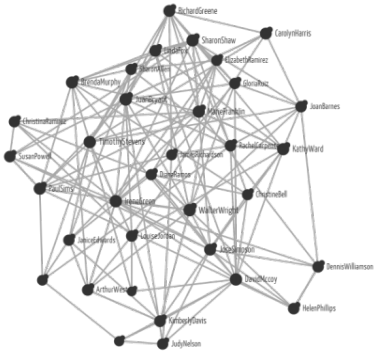
In [STOC '15], we bridged these two paradigms achieving both space efficiency and update times efficiency.

Outline

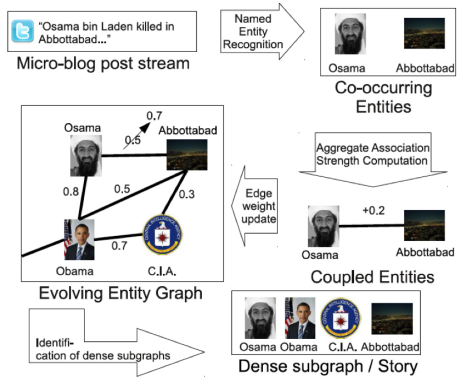
- **Related work**
- **Rich graph data: from applications to formulations**
- **Some Algorithmic Results**
  - Flowless
  - k-clique DSP
  - AntiBenford Subgraphs
- **Conclusion**

Large-Near Clique Extraction

Who-calls-whom



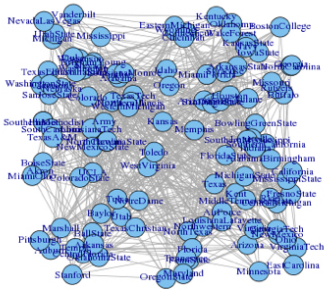
Twitter



Youtube

	Security	**Social Media**	**Topic Clusters**
V	Humans	Entities	Videos
E	Phone call	Co-occurence	Co-watched
Large Near clique	Anomaly	Thematic coherence	Thematic coherence

DSP typically **does not** result in large-near cliques



- Football network with

  $n = 115$ vertices $\leftrightarrow$ teams and $m = 613$ edges $\leftrightarrow$ games
- The optimal solution S^* to the **DSP** is the whole network with resulting degree density $\rho(S^*) = 5.3$ and edge density $f_e(S) = \frac{e(S)}{\binom{|S|}{2}} = 0.094.$
- There exists S' such that $|S'| = 18, f_e(S') = 0.48.$ However $\rho(S') = 4.1.$

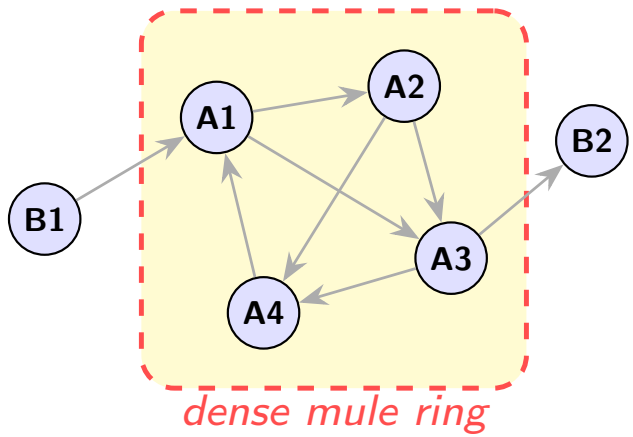
Can we have a principled approach for finding large near-cliques in real-world networks?
[Tsourakakis, 2015, Mitzenmacher et al., 2015]

Transactions are relational data

- Banks log every transfer: time, sender, receiver, amount, ...
- That log is a graph — accounts are nodes; transfers are directed, weighted, timestamped edges.
- Fraud and laundering never show up in one row — they show up as coordinated dense structure (mule rings, smurfing).

time	from		to		amount
09:00	B1	UK	A1	IT	$9,900
09:01	A1	IT	A2	CY	$9,800
09:02	A2	CY	A3	MO	$9,700
09:03	A3	MO	A4	AL	$9,500
09:04	A4	AL	A1	IT	$9,900
09:05	A3	MO	B2	IR	$9,600

graph



all just under the $10k reporting threshold

A single transaction looks normal; the dense subgraph it sits in does not

DSP and Transaction Monitoring [KDD '22]

The Challenge

Traditional AML monitoring relies on threshold-based rules. As a result, **75–99% of generated alerts is often reported to be the false positives**, creating substantial investigation costs and analyst burden.

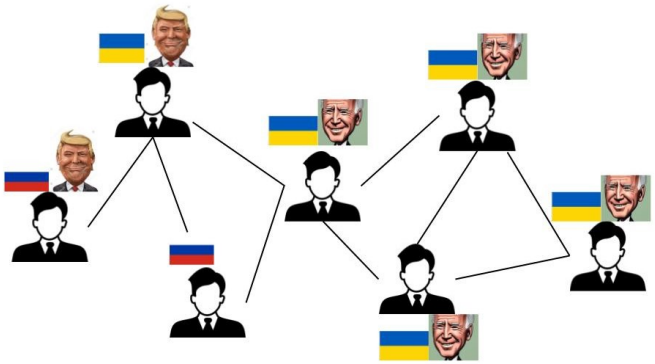
- Global spending on AML compliance is often estimated at \$200–300+ billion annually across financial institutions worldwide.
- Industry reports routinely estimate that 60–90% of AML investigation effort is spent on false positives.

Key Research Question?

How can we leverage rich relational attributes to bias dense subgraph discovery toward true positives in a principled way?

AI gives us advanced signals [WSDM '25]

Rich graph data may come from

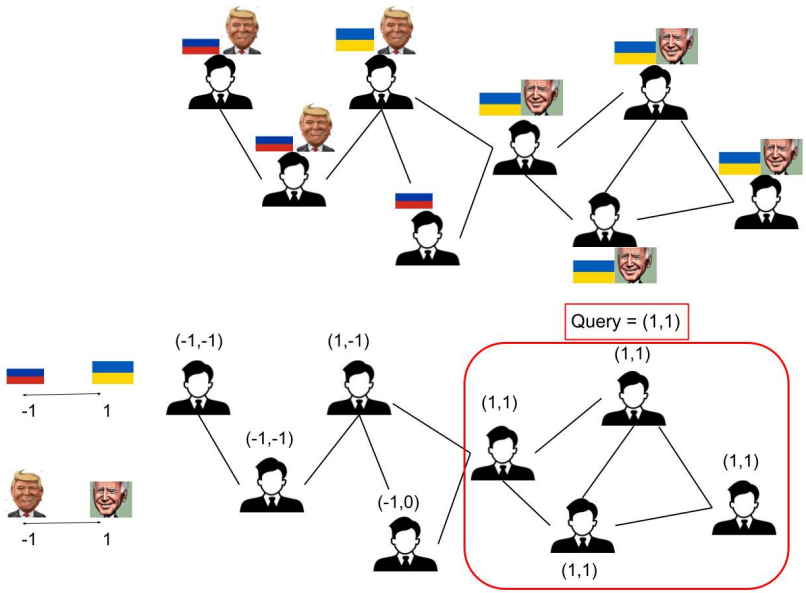


- Mapping text $\rightarrow$ opinions used to be the hard part [Medhat et al., 2014]. With LLMs, it has become much more feasible [Zhang et al., 2024]

Key Research Question?

How can we find dense subgraphs aligned with opinion query vectors? [Chen et al., 2025b]

Dense and like-minded [WSDM '25]



Q-DISCO: Query-Centric Dense Subgraphs [Chen et al., 2025b]:

$$\max \quad \text{Density}(S)$$

$$\text{s.t.} \quad \text{Average Agreement}(S, \mathbf{q}) \geq \theta$$

Dense and like-minded with little info [VLDB '22]

Key Idea

Instead of estimating opinions for everyone, start from a small set of known accounts:

- **Attractors**: trusted representatives of a viewpoint.
- **Repulsers**: accounts expressing the opposite viewpoint.

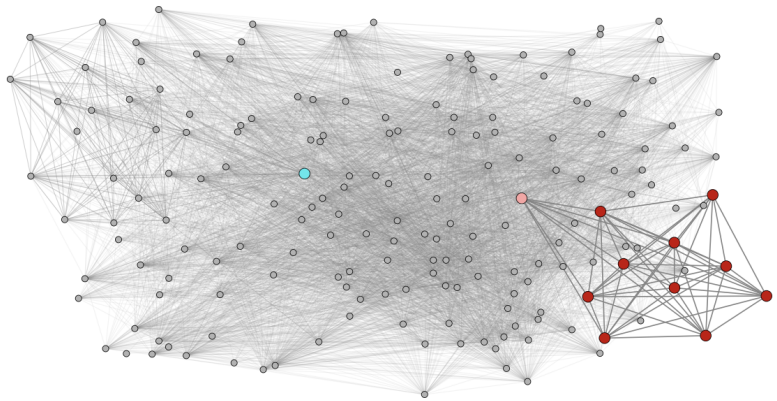
Use the interaction graph to discover groups that are simultaneously

1. densely connected, and
2. strongly aligned with the attractors while avoiding the repulsers.

Dense + Like-minded = Echo Chambers, Extremist Communities, Coordinated Bot Clusters

Dense and like-minded with little info [VLDB '22]

$$\underbrace{\text{Density}}_{\text{graph structure}} + \underbrace{\text{Proximity to attractors}}_{\text{side information}} + \underbrace{\text{Distance to repulsers}}_{\text{opposite viewpoint}}$$

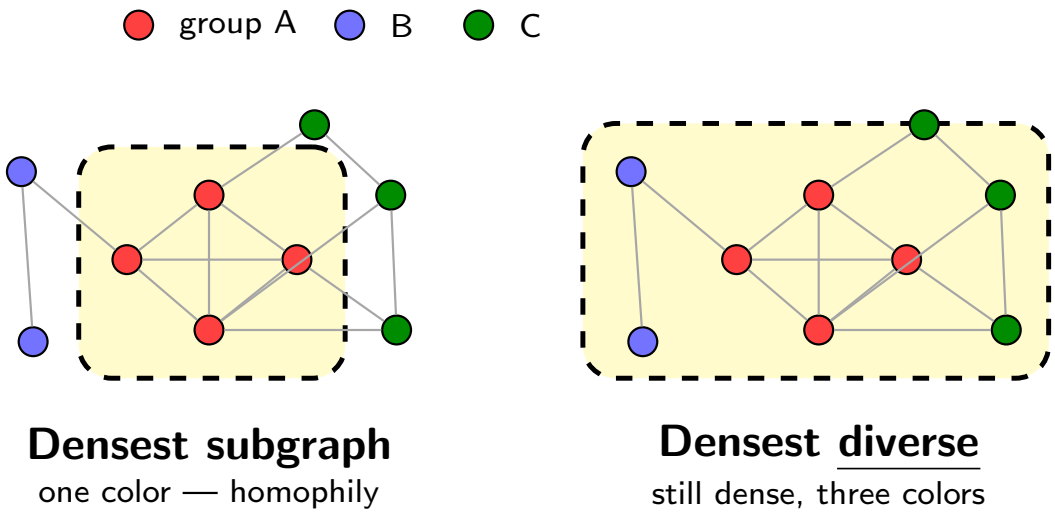


We query for a dense cluster close to Makeleio and far from the Efimerida Syntakton. The method recovers ten Golden Dawn leaders later charged with running a criminal organization.

Diverse and Dense [KDD '23]

- Task: staff a team from a collaboration network where no single group dominates (e.g., gender, religion).

  This may create legal, compliance, or reputational concerns.



Diverse Densest Subgraph [KDD '23]

Given a colored graph $G = (V, E)$ with color function $\ell : V \to C$, define, for any $S \subseteq V$, $c_{\max}(S) = \max_{c \in C} |\{v \in S : \ell(v) = c\}$
i.e., the size of the largest color class in S.
The dominance ratio of S is

$$\alpha(S) = \frac{c_{\max}(S)}{|S|}.$$

Dense and Diverse

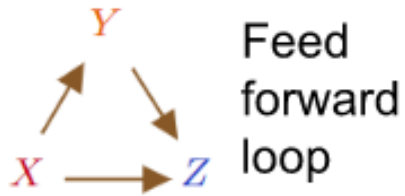
Given an undirected graph $G = (V, E)$, a coloring $\ell : V \to C$, and a parameter $\alpha \in [1/|C|, 1]$ find a subset $S \subseteq V$ that maximizes the degree density subject to $\alpha(S) \leq \alpha$.

Our formulation substantially generalizes the fair densest-subgraph framework of [Anagnostopoulos et al., 2020]

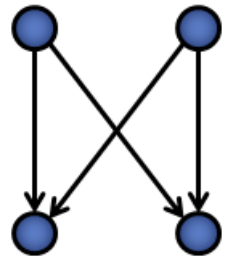
Densest subgraph as an anomaly detection tool in science

Motifs are small subgraph patterns with **statistical significance.**

[Ugander et al., 2013]
Social network



[Milo et al., 2002]
Brain network

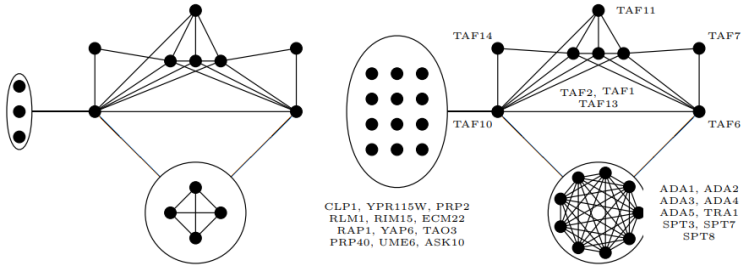


[Lipshtat et al., 2008]
Transcriptional network

Is a certain pattern statistically significant?

Standard approach: count the occurrences, compare to a null model (e.g., Chung-Lu) random graph model and compute z-scores.

Combinatorial Artifact



This motif appears 27,720 times clustered within 29 nodes [Grochow and Kellis, 2007].

$$\binom{12}{3} \times \binom{9}{4} = 27720$$

Independent sets and cliques boost the occurrence of this motif.

- We refer to this phenomenon as Combinatorial Artifact [Pachter,].

Why I read the network nonsense papers

February 12, 2014 in q-bio.GN, reviews, sophistry | Tags: Benjamini-Hochberg, binomial coefficient, FDR, Joshua Grochow, Luke Ward, Manolis Kellis, methodsmatter, network motif | by Lior Pachter

Is a certain pattern statistically significant?

Can we use the densest subgraph problem to detect comb. artifacts?

(f, q)-spanning motif [ECML-PKDD '22]

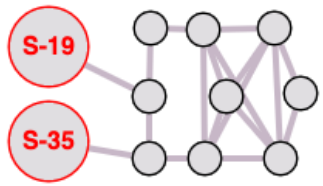
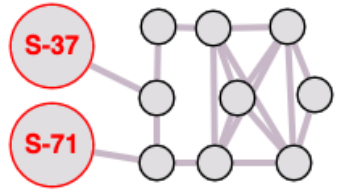
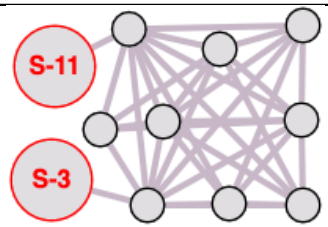
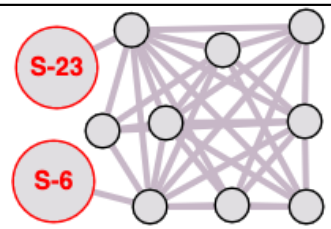
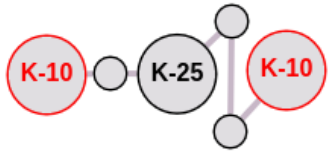
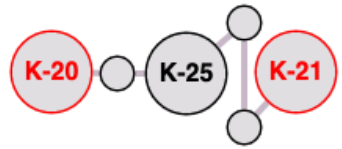
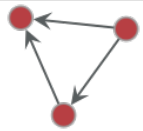
A motif should frequently occur across the whole network and not only in a small portion of the graph!

Definition ((f, q)-spanning)

A motif $\mathcal{M}$ is (f, q)-spanning in graph $G(V, E)$ if there exists a set of nodes $S \subseteq V$ such that $|S| \leq q|V|$ and the induced subgraph $G[S]$ contains an (at least) f-fraction of the occurrences of $\mathcal{M}$ in G.

- Given a graph G, a motif $\mathcal{M}$
  - Fix the value of q, maximizing f is NP-hard with reduction from densest-at-most-k-subgraph problem.
  - Fix the value of f, minimizing q is NP-hard with reduction from the minimum p-union problem.

Detected Combinatorial artifacts [ECML-PKDD '22]

Dataset	Motif	Artifact source	Count	(f, q)
S. cerevisiae - PPI			$\binom{37}{19} \times \binom{71}{35}$	$(1, 0.06)$
C. Elegans-PPI			$\binom{23}{11} \times \binom{6}{3}$	$(1, 0.063)$
hamsterster-social			$\binom{21}{10} \times \binom{20}{10}$	$(1, 0.027)$
C. Elegans-Brain		-	1554	$(0.8, 0.61)$

Outline

- **Related work**
- **Rich graph data: from applications to formulations**
- **Some Algorithmic Results**
- **Conclusion**

Challenge: Can we combine the optimality guarantees of maximum-flow methods with the scalability and simplicity of greedy peeling?

- Near-optimal DSP without Flow Computations (WWW '20)

  Boob-Gao-Peng-Sawlani-**T**-Wang-Wang

  Important follow-up work:
- Densest Subgraph: Supermodularity, Iterative Peeling, and Flow (SODA '22) by Chekuri, Quanrud, Torres [Chekuri et al., 2022]

LP formulation

$$\textsc{Primal}(G)$$

$$\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} y_e \\
\text{subject to} \quad & y_e \leq x_u, x_v, && \forall e = uv \in E \\
& \sum_{v \in V} x_v \leq 1, \\
& y_e \geq 0, x_v \geq 0, && \forall e \in E, \forall v \in V
\end{aligned}$$

Theorem [Charikar '00]: OPT of this LP $= \rho^*_G$

Dual of the LP

$$\textsc{Primal}(G)$$

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

$$\textsc{Dual}(G)$$

$$\begin{aligned} \min \quad & D \\ \text{s. to} \quad & f_e(u) + f_e(v) \geq 1, \\ & \sum_{e \ni v} f_e(v) \leq D, \\ & f_e(v) \geq 0, \end{aligned}$$

Dual LP

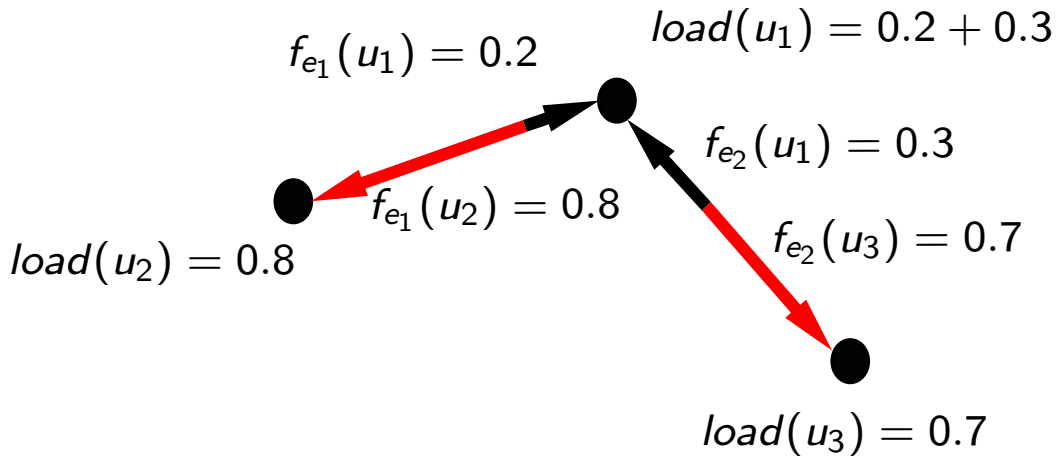
$$\textsc{Dual}(G)$$

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

Find the smallest D such that:

$$\begin{aligned}
f_e(u) + f_e(v) &= 1 \\
\sum_{e \ni v} f_e(v) &\leq D
\end{aligned}$$

Visualizing node loads

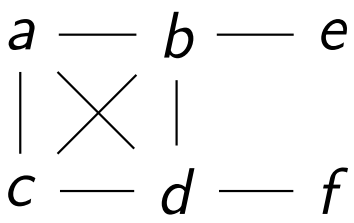


- Each edge pushes its weight (i.e., edge load) to its endpoints.
- The load of each node equals to the sum of all edge weights pushed to it.

$$\text{load}(v) = \sum_{e \ni v} f_e(v).$$

Alternate visualization of dual

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



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

- **Greedy++** can be viewed as repeatedly re-running Charikar while remembering past mistakes.
- Vertices that have accumulated too much load become more expensive, forcing subsequent rounds to explore different dense regions.

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).

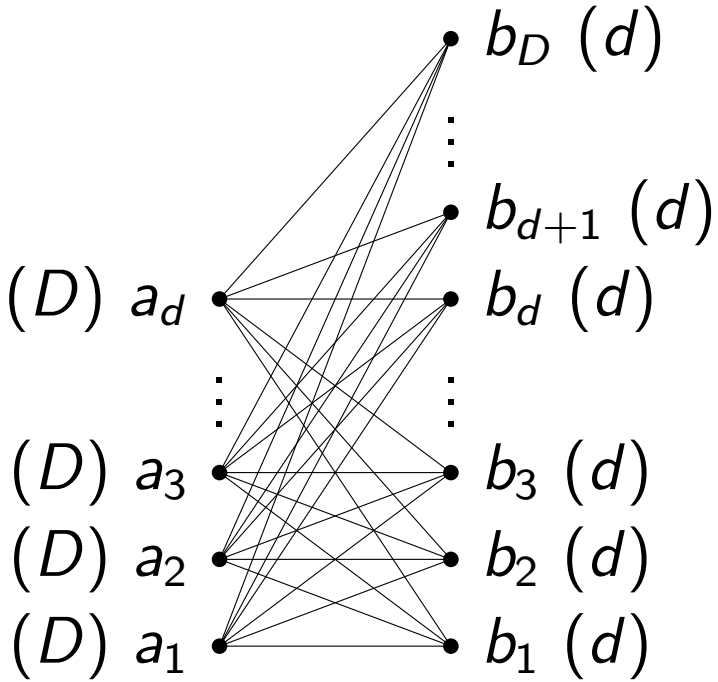
Greedy++ pseudocode – Input $G(V, E), T$

$G_{\text{densest}} \leftarrow G$
Initialize the vertex load vector $\ell^{(0)} \leftarrow 0 \in \mathbb{Z}^n$;
for $i : 1 \to T$ **do**
  $H \leftarrow G$;
  while $H \neq \emptyset$ **do**
    Find the vertex $u \in H$ with minimum $\ell_u^{(i-1)} + \deg_H(u)$;
    $\ell_u^{(i)} \leftarrow \ell_u^{(i-1)} + \deg_H(u)$;
    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_{densest}

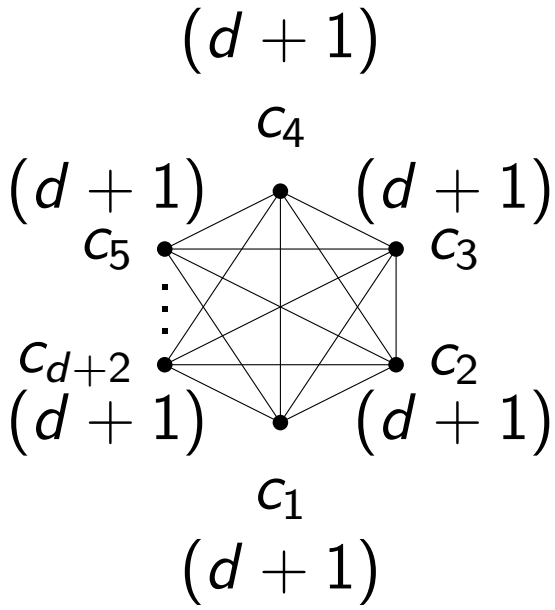
Tight example for Greedy

Our graph $G = B +$ many copies of H_i

$B = K_{d,D}$



$H_i = K_{d+2}$

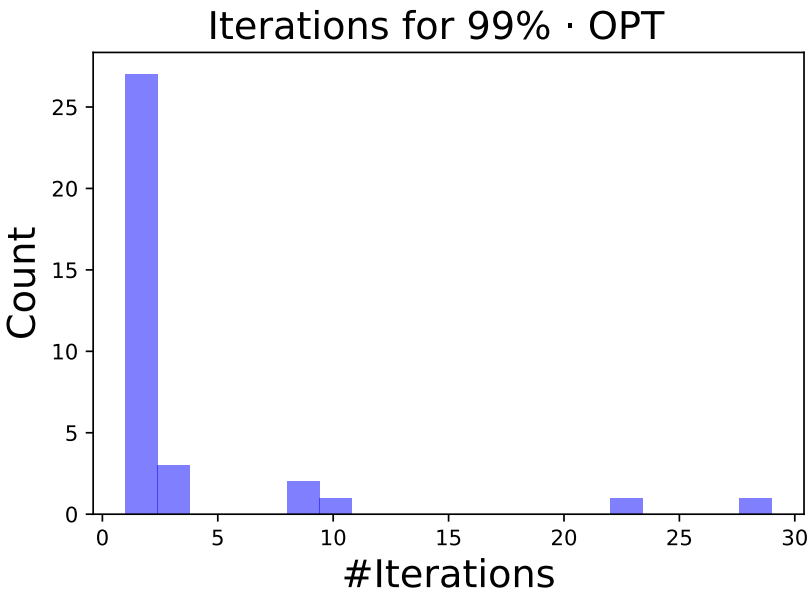


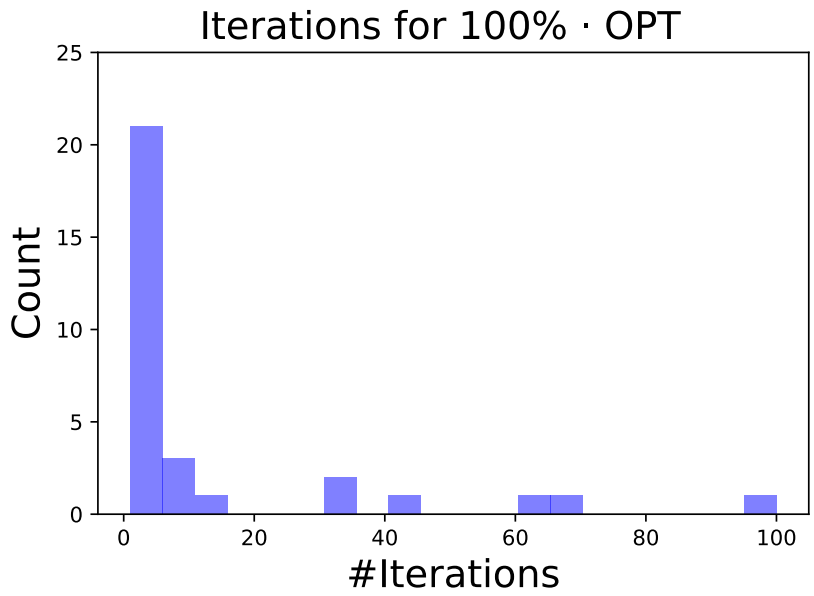
- Charikar gives $\frac{1}{2}$-approximation but Flowless with $T = 2$ is optimal

Large collection of datasets

Name	n	m
web-trackers [23]	40 421 974	140 613 762
orkut [23]	3 072 441	117 184 899
livejournal-affiliations [23]	10 690 276	112 307 385
wiki-topcats	1 791 489	25 447 873
cit-Patents	3 774 768	16 518 948
actor-collaborations [23]	382 219	15 038 083
ego-gplus	107 614	12 238 285
dblp-author	5 425 963	8 649 016
web-BerkStan	685 230	6 649 470
flickr [37]	80 513	5 899 882
wiki-Talk	2 394 385	4 659 565
web-Google	875 713	4 322 051
com-youtube	1 134 890	2 987 624
roadNet-CA	1 965 206	2 766 607
web-Stanford	281 903	1 992 636
roadNet-TX	1 379 917	1 921 660
roadNet-PA	1 088 092	1 54 898
Ego-twitter	81 306	1 342 296
com-dblp	317 080	1 049 866
com-Amazon	334 863	925 872
soc-slashdot0902	82 168	504 230
soc-slashdot0811	77 360	469 180
soc-Epinions	75 879	405 740
blogcatalog [37]	10,312	333 983
email-Enron	36 692	183 831
ego-facebook	4 039	88 234
ppi [31]	3 890	37 845
twitter-retweet [30]	316 662	1 122 070
twitter-favorite [30]	226 516	1 210 041
twitter-mention [30]	571 157	1 895 094
twitter-reply [30]	196 697	296 194
soc-sign-slashdot081106	77 350	468 554
soc-sign-slashdot090216	81 867	497 672
soc-sign-slashdot090221	82 140	500 481
soc-sign-epinions	131 828	711 210

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:

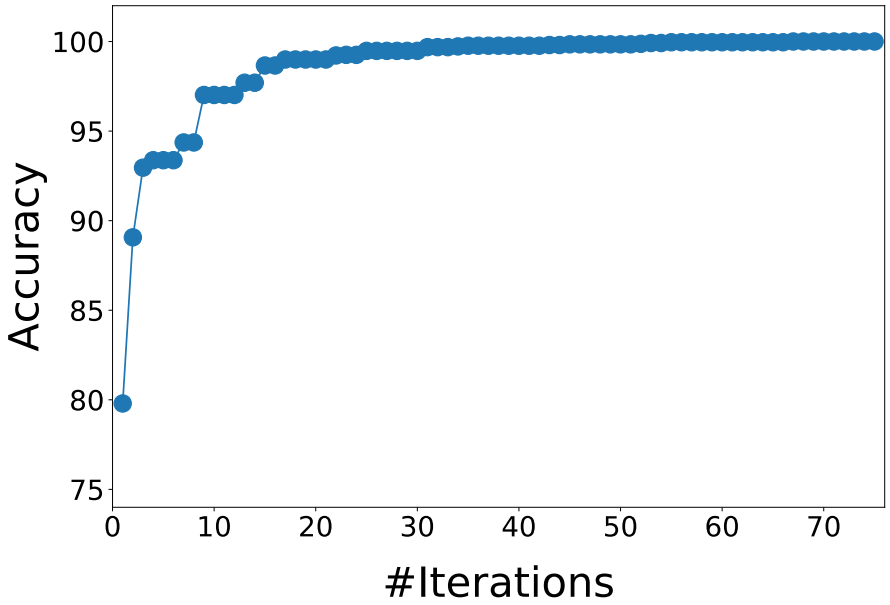


Figure: Amazon co-purchasing network

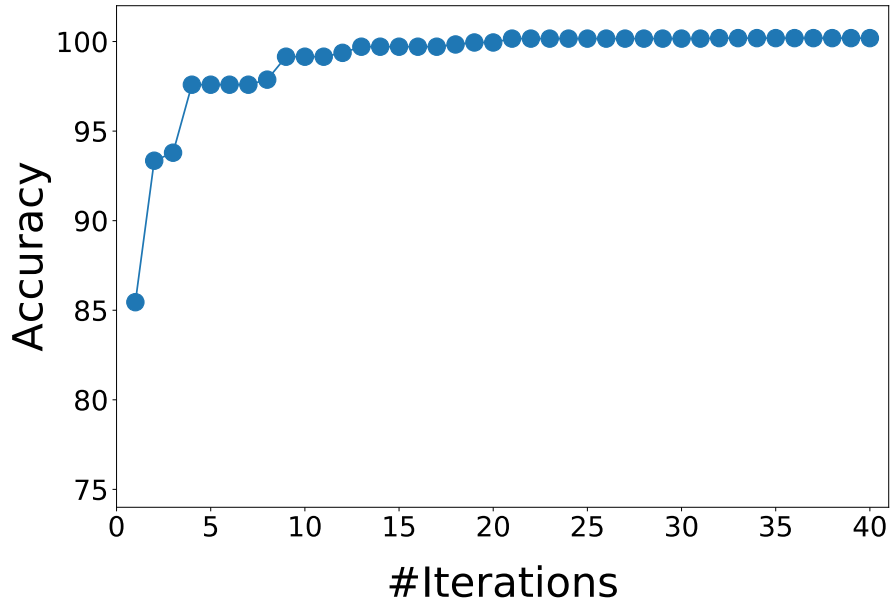


Figure: California road connection network

Experimental results

Scalability and speed of Greedy++:

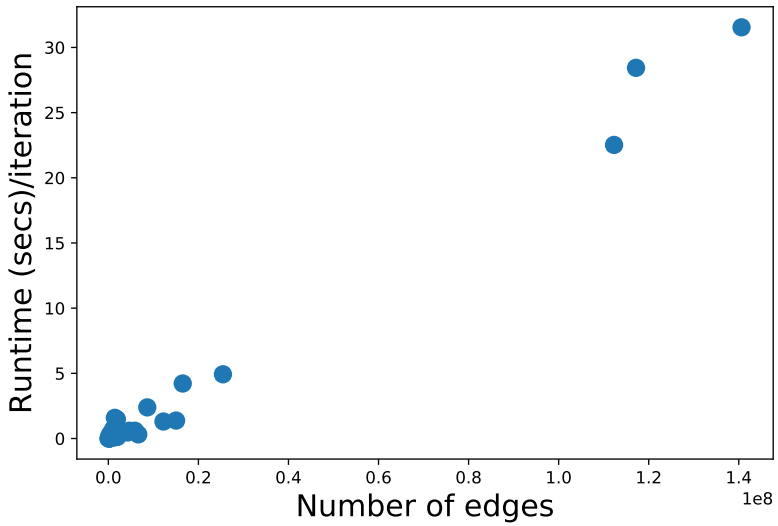


Figure: Runtime in seconds per iteration

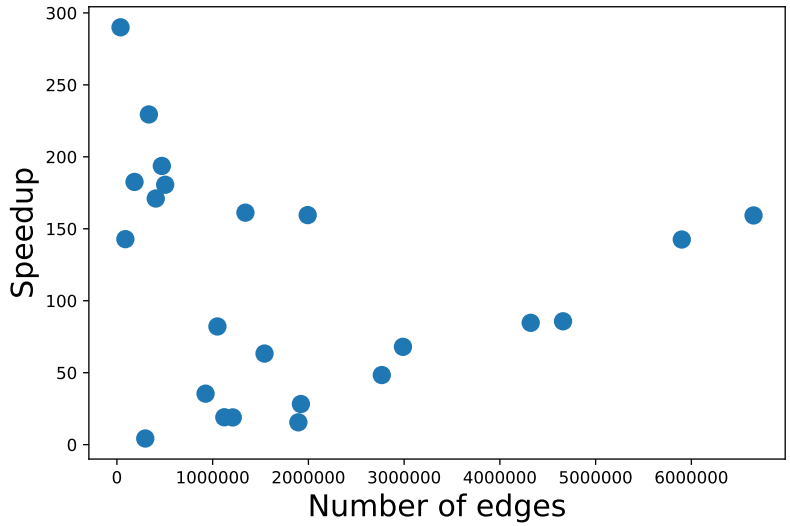


Figure: Speedup over exact algorithm (max-flow based)

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× on average, with a standard deviation equal to 57.4. The smallest speedup observed was 67.9×, and the largest speedup 290×.
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.

SuperGreedy++ [Chekuri et al., 2022]

Chekuri, Quanrud, Torres proved that indeed Greedy++ converges to a $(1-\epsilon)$ approximation for DSP ($\epsilon > 0$) in

$$O\left(\frac{1}{\epsilon^2}\frac{\max deg}{\rho^*}\log n\right)$$

iterations.

Major Conceptual leaps:

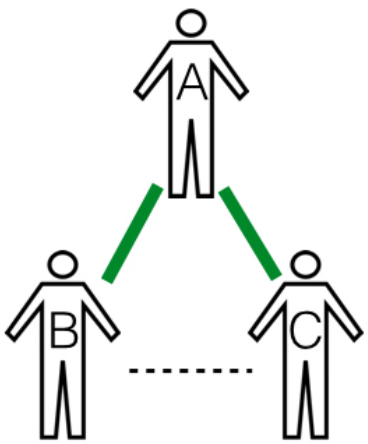
1. By lifting the analysis from graphs to orderings, Greedy++ becomes a multiplicative-weights algorithm over the Lovász-extension LP.
2. Generalization of Greedy++ to SuperGreedy++ for general supermodular optimization.

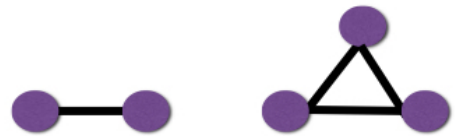
Challenge: Can we have a principled approach for finding large near-cliques in real-world networks?

- The K-clique Densest Subgraph Problem (WWW 2015) **T**

- Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling (KDD '15) by Mitzenmacher, Pachocki, Peng, **T**, Chen Xu

k-clique DSP for large-near cliques





2-clique 3-clique

E.g. c_2()=3

For any $S \subseteq V$ let

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

Define ***k*-clique density**

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

Solve the ***k*-clique DSP**

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

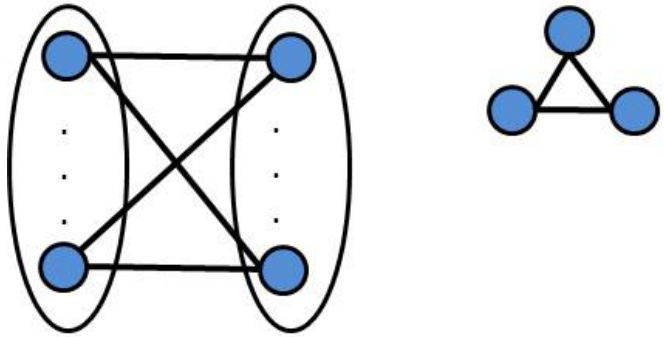
Triangle densest subgraph problem

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 $O\left(m^{3/2} + nt + \min\left(n, t\right)^3\right)$ *time [Ahuja et al., 1994].*

Furthermore,

Theorem

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

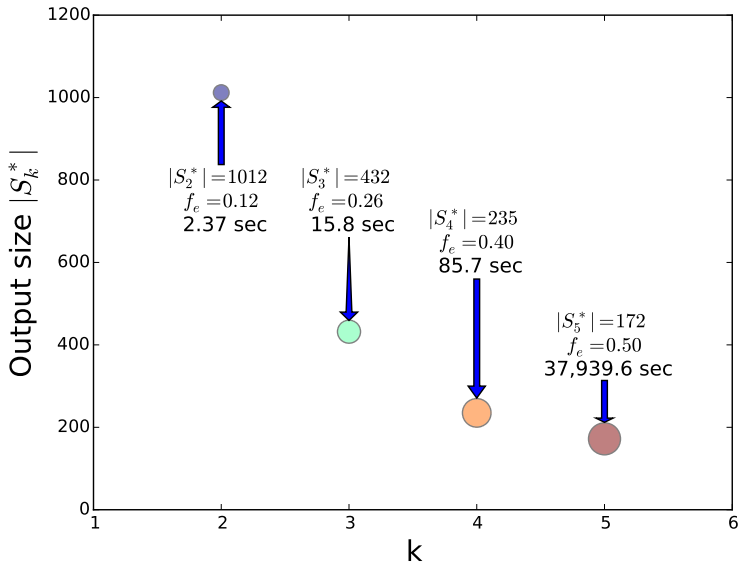
Computation involves

- 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 $f_e(S) = \frac{e(S)}{\binom{|S|}{2}}$

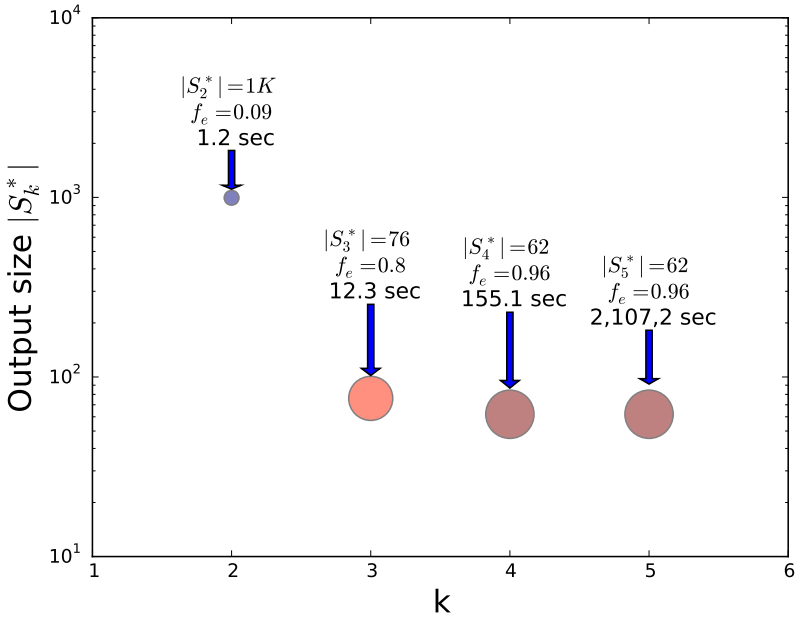
# nodes	75 877
# edges	405 739



k	c_k	T_k (sec)
3-clique	1.6M	1.6
4-clique	5.8M	4.8
5-clique	17.4M	13.4

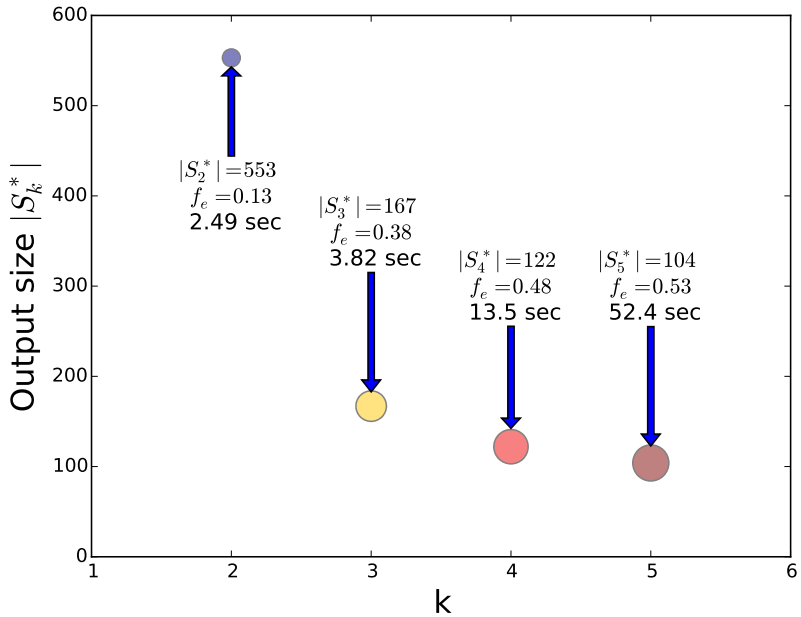
CA-Astro and Email networks

# nodes	18 772
# edges	198 050



k	c_k	T_k (sec)
3-clique	1.4M	0.6
4-clique	9.5M	3.9
5-clique	65M	27.2

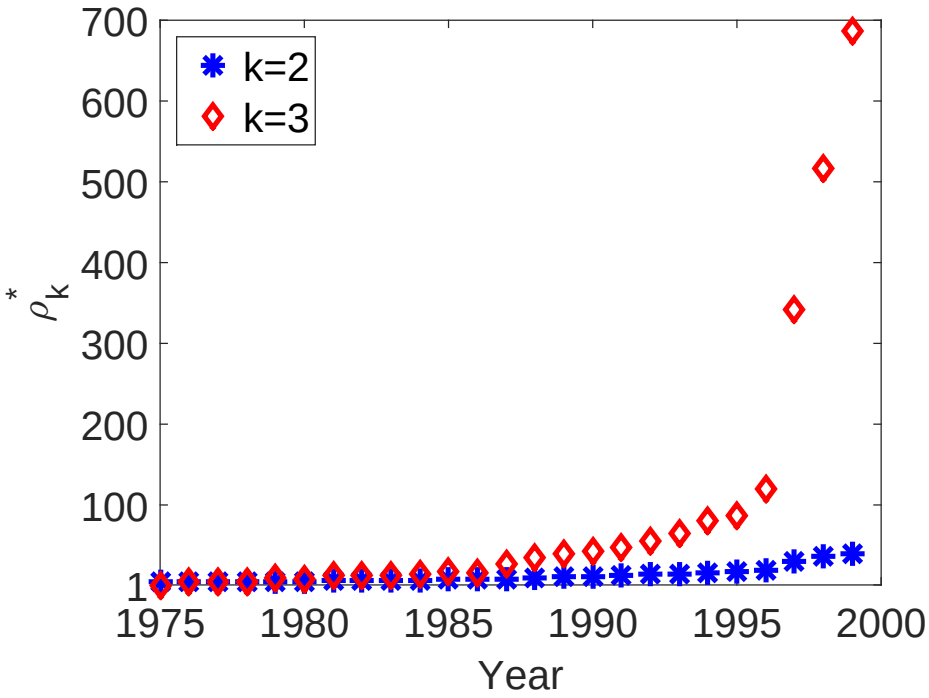
# nodes	234 352
# edges	383 111

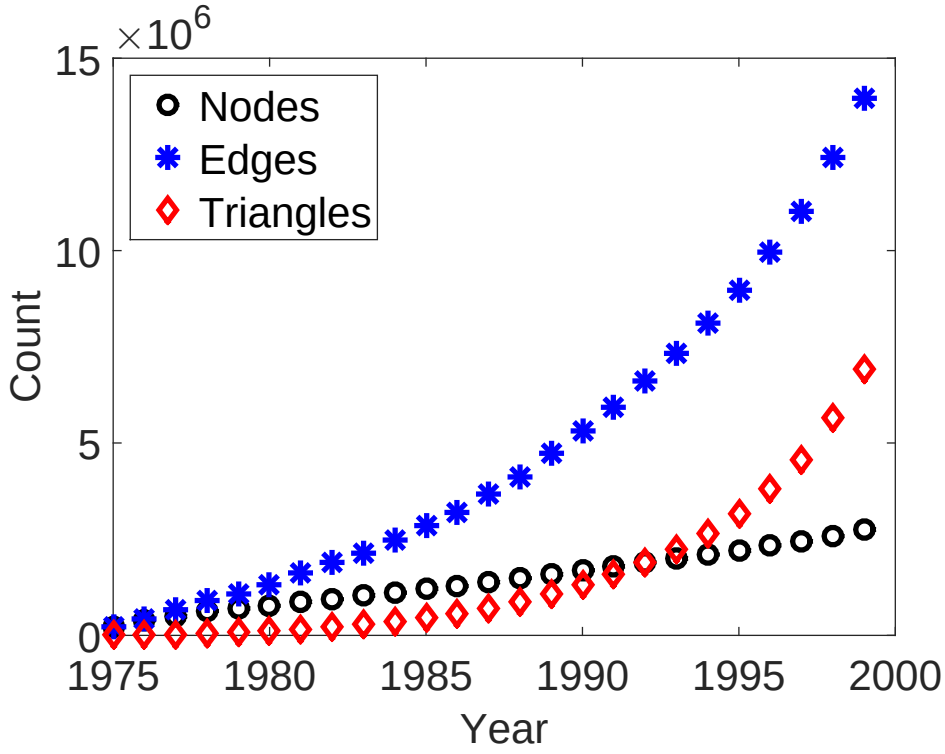


k	c_k	T_k (sec)
3-clique	0.4M	0.4
4-clique	1M	0.9
5-clique	2.6M	1.9

Time evolving networks

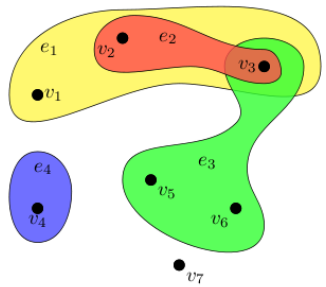
Patents citation network that spans 37 years, specifically from January 1, 1963 to December 30, 1999.





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

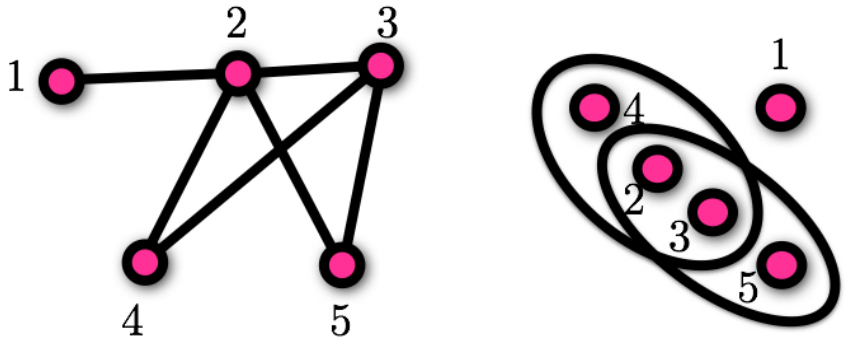
Let $\mathcal{H}(V, E_{\mathcal{H}})$ be a hypergraph, $\epsilon > 0$.

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 ρ^* **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 $p = \frac{6n \log n}{|E_{\mathcal{H}}|\epsilon^2}$.

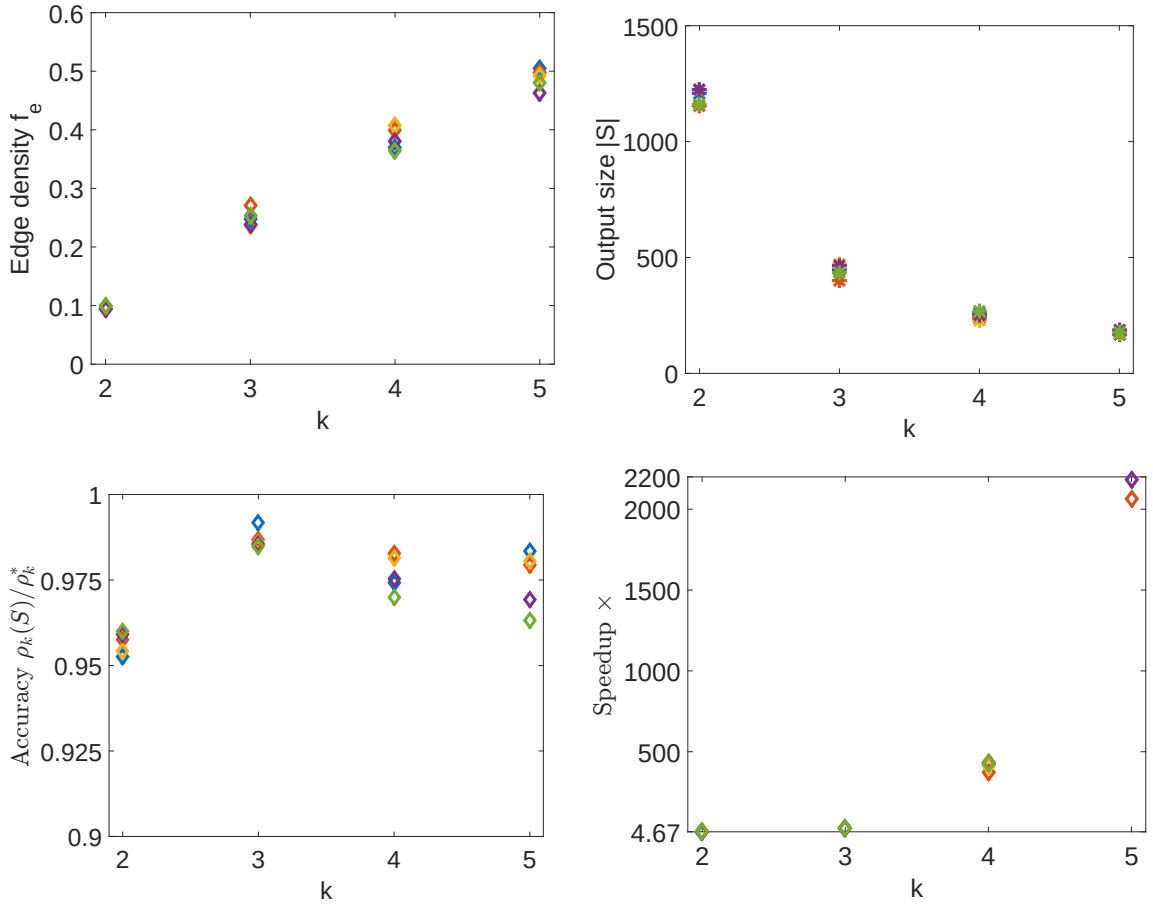
Expected space reduction is $O(\frac{1}{p})$.

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

k	Avg. Speedup	Accuracy
2	3×	$\geq 95\%$
3	23.8×	$\geq 98\%$
4	302 ×	$\geq 99\%$
5	24 000×	$\approx 100\%$

Zooming-in on Epinions network

EPINIONS graph, $n = 75\,877, m = 405\,739$



KClist++ a recent algorithm that combines Frank-Wolfe-like algorithm with fast clique enumeration [Sun et al., 2020].

Challenge: How can we leverage rich relational attributes to bias dense subgraph discovery toward true positives in a principled way?

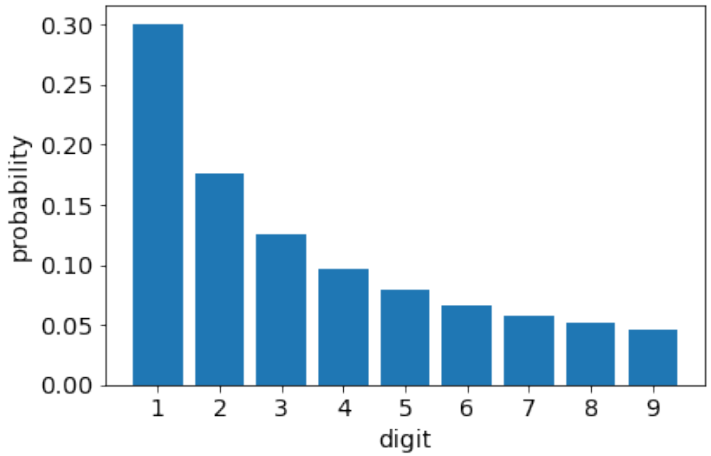
- AntiBenford Subgraphs: Unsupervised Anomaly Detection in Financial Networks (KDD '22)

  Chen-**T**

Benford's law

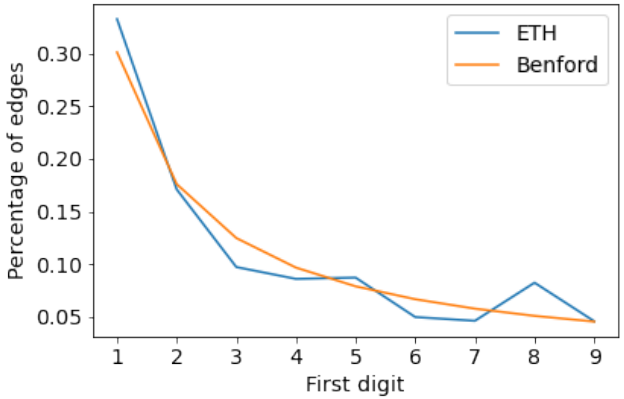
Benford's law states that in many real-world numerical datasets, the leading digit is far from uniform: numbers are much more likely to begin with 1 than with 9 [Benford, 1938, Newcomb, 1881].

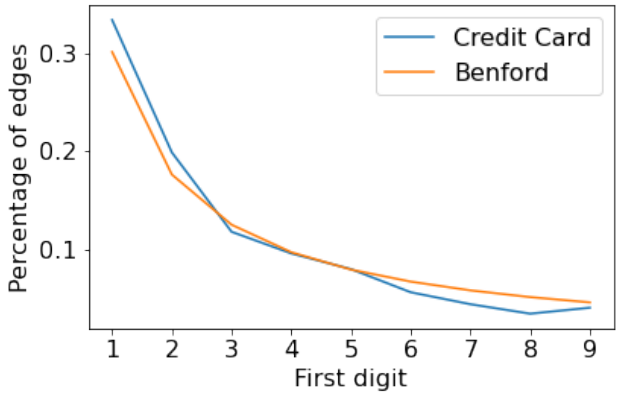
$$P(d) = \log_{10}(1 + \frac{1}{d})$$



It has been used to detect anomalies in: Tax evasion [Nigrini, 2017], Trading [Cerioli et al., 2019], ...

Anti-Benford subgraphs [KDD '22]





- Many naturally occurring financial quantities often approximately follow Benford's law."
  [Kaggle, , Data World,]

Problem formulation (informal): Given a weighted network $G(V, A, w), w : A \to \mathbb{R}^+$, find a subset of nodes $S \subseteq V$ such that:

- it induces on average many edges, and
- whose weights' first digit empirical histogram deviates from Benford's law significantly, in a statistical sense.

Definitions

Hypothesis H_0: Given a weighted transaction network $G(V = [n], E, w)$, we view the first digits of the transaction weights as an i.i.d. samples from Benford's distribution.

χ^2-statistic of a subgraph $G[S]$:

$$\chi^2(S) = \sum_{d=1}^{9} \frac{\left(X_{S,d} - \mathbb{E}(X_{S,d})\right)^2}{\mathbb{E}(X_{S,d})},$$

$X_{S,d}$ is the number of edges in subgraph S whose weight's first digit is d.

Main result

Under the null hypothesis H_0:

Theorem

With high probability for all subgraphs $S \subseteq V$ with **large enough** *edge density, the number of edges $X_{S,d}$ that start with digit d in $G[S]$ is strongly concentrated around its expectation, for all $d = 1, \ldots, 9$.*

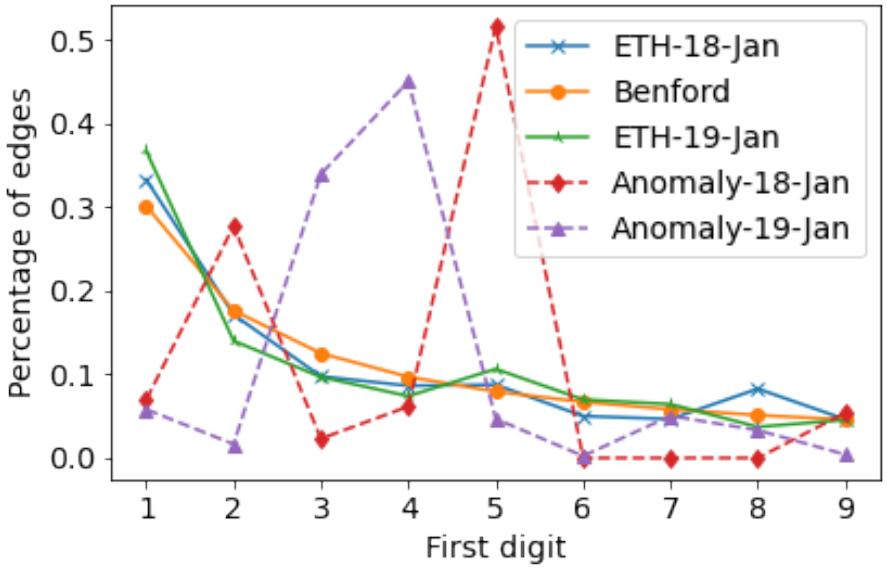
Corollary

We expect to observe subgraphs $S \subseteq V, |S| = \Omega(\log n)$ with $\frac{\chi^2(S)}{|S|} \approx \frac{e(S)}{|S|}$.

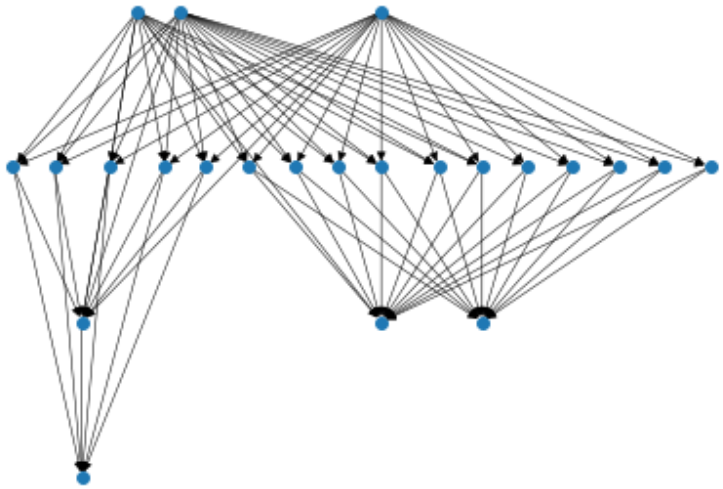
AntiBenford subgraph: is a subset of nodes $S \subseteq V$ such that $\frac{\chi^2(S)}{|S|} \gg \frac{e(S)}{|S|}$.

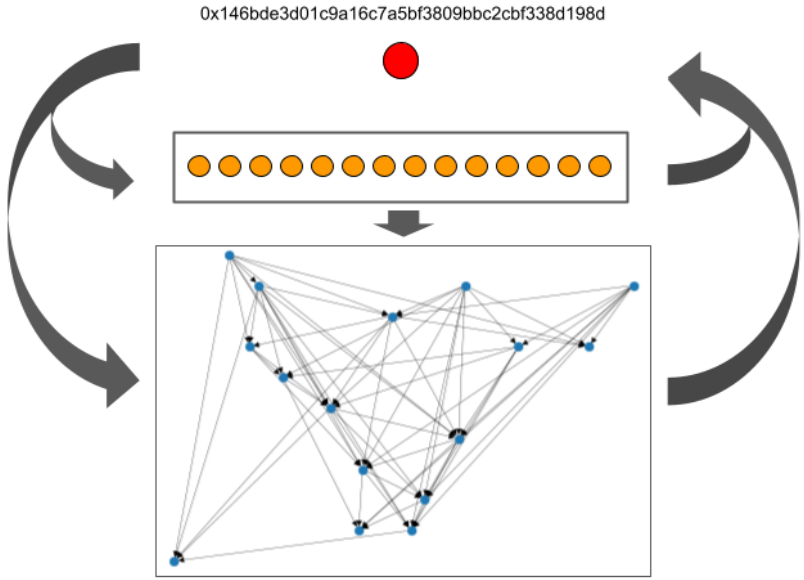
Ethereum Token Transfer Dataset

Name	# nodes	# edges	# transactions
ETH-Jan-18	1 761 571	2 749 707	4 279 799
ETH-Jan-19	2 199 347	3 331 594	6 128 061



Suspicious properties of AntiBenford subgraphs





- High-volume transfers.
- Multipartite structure
- Zero-out middle account [Li et al., 2020]

Outline

- **Related work**
- **Rich graph data: from applications to formulations**
- **Some Algorithmic Results**
  - Flowless
  - k-clique DSP
  - AntiBenford Subgraphs

- **Conclusion**

Takeaways & Open Problems

- Classical challenges in dense subgraph discovery remain open
  - Closing the approximation gap for Densest-k-Subgraph
  - Faster algorithms for streaming, dynamic, and distributed settings
  - Understanding why Greedy++ converges so quickly in practice
- The future lies in richer formulations
  - Side information, opinions, risk, diversity, anomalies, constraints
  - Density is often necessary, but rarely sufficient
- A recurring lesson
  - Algorithm design is very important
  - Choosing the right objective is also very important for applications and may reveal new computational challenges

Thank you! Questions?

references I

Ahuja, R. K., Orlin, J. B., Stein, C., and Tarjan, R. E. (1994).

Improved algorithms for bipartite network flow.

SIAM Journal on Computing, 23(5):906–933.

Anagnostopoulos, A., Becchetti, L., Fazzone, A., Menghini, C., and Schwiegelshohn, C. (2020).

Spectral relaxations and fair densest subgraphs.

In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pages 35–44.

Andersen, R. and Chellapilla, K. (2009).

Finding dense subgraphs with size bounds.

In Algorithms and Models for the Web-Graph, pages 25–37. Springer.

references II

Bahmani, B., Kumar, R., and Vassilvitskii, S. (2012).

Densest subgraph in streaming and mapreduce.

Proceedings of the VLDB Endowment, 5(5):454–465.

Benford, F. (1938).

The law of anomalous numbers.

Proceedings of the American philosophical society, pages 551–572.

Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., and Vijayaraghavan, A. (2010).

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

In Proc. STOC '10, pages 201–210.

references III

Bhattacharya, S., Henzinger, M., Nanongkai, D., and Tsourakakis, C. E. (2015).

Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams.

arXiv preprint arXiv:1504.02268.

Boob, D., Gao, Y., Peng, R., Sawlani, S., Tsourakakis, C., Wang, D., and Wang, J. (2020).

Flowless: extracting densest subgraphs without flow computations.

In Proc. TheWebConf '20, pages 573–583.

Cerioli, A., Barabesi, L., Cerasa, A., Menegatti, M., and Perrotta, D. (2019).

Newcomb–benford law and the detection of frauds in international trade.

Proceedings of the National Academy of Sciences, 116(1):106–115.

references IV

Charikar, M. (2000).

Greedy approximation algorithms for finding dense components in a graph.

In APPROX.

Chekuri, C., Quanrud, K., and Torres, M. R. (2022).

Densest subgraph: Supermodularity, iterative peeling, and flow.

In Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1531–1555. SIAM.

Chen, L., Kyng, R., Liu, Y., Peng, R., Probst Gutenberg, M., and Sachdeva, S. (2025a).

Maximum flow and minimum-cost flow in almost-linear time.

Journal of the ACM, 72(3):1–103.

references V

Chen, T., Miyauchi, A., and Tsourakakis, C. E. (2025b).

Q-disco: Query-centric densest subgraphs in networks with opinion information.

In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, pages 194–203.

Chen, T. and Tsourakakis, C. (2022).

Antibenford subgraphs: Unsupervised anomaly detection in financial networks.

In KDD '22, pages 2762–2770.

Data World, .

Dataworld.

https://data.world/raghu543/credit-card-fraud-data.

references VI

Esfandiari, H., Hajiaghayi, M., and Woodruff, D. P. (2015).

Applications of uniform sampling: Densest subgraph and beyond.

arXiv preprint arXiv:1506.04505.

Gallo, G., Grigoriadis, M. D., and Tarjan, R. E. (1989).

A fast parametric maximum flow algorithm and applications.

SIAM Journal on Computing, 18(1):30–55.

Goldberg, A. V. (1984).

Finding a maximum density subgraph.

Technical report, University of California at Berkeley.

references VII

Grochow, J. A. and Kellis, M. (2007).

Network motif discovery using subgraph enumeration and symmetry-breaking.

In Speed, T. and Huang, H., editors, RECOMB, pages 92–106.

Kaggle, .

Complete live historical ethereum blockchain data (bigquery).

https://www.kaggle.com/bigquery/ethereum-blockchain.

Khuller, S. and Saha, B. (2009).

On finding dense subgraphs.

In ICALP.

references VIII

Li, X., Liu, S., Li, Z., Han, X., Shi, C., Hooi, B., Huang, H., and Cheng, X. (2020).

Flowscope: Spotting money laundering based on graphs.

Proceedings of the AAAI Conference on Artificial Intelligence, 34(04):4731–4738.

Lipshtat, A., Purushothaman, S. P., Iyengar, R., and Ma'ayan, A. (2008).

Functions of bifans in context of multiple regulatory motifs in signaling networks.

Biophysical Journal, 94(7):2566–2579.

McGregor, A., Tench, D., Vorotnikova, S., and Vu, H. T. (2015).

Densest subgraph in dynamic graph streams.

arXiv preprint arXiv:1506.04417.

references IX

Medhat, W., Hassan, A., and Korashy, H. (2014).

Sentiment analysis algorithms and applications: A survey.

Ain Shams engineering journal, 5(4):1093–1113.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002).

Network motifs: Simple building blocks of complex networks.

Science, 298(5594):824–827.

Mitzenmacher, M., Pachocki, J., Peng, R., Tsourakakis, C. E., and Xu, S. C. (2015).

Scalable large near-clique detection in large-scale networks via sampling.

In KDD '15: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 815–824.

references X

Newcomb, S. (1881).

Note on the frequency of use of the different digits in natural numbers.

American Journal of mathematics, 4(1):39–40.

Nigrini, M. (2017).

Audit sampling using benford's law: A review of the literature with some new perspectives.

Journal of Emerging Technologies in Accounting, 14.

Orlin, J. B. (2013).

Max flows in o (nm) time, or better.

In Proceedings of the forty-fifth annual ACM symposium on Theory of computing, pages 765–774.

references XI

Pachter, L.

Why i read the network nonsense papers.

https://liorpachter.wordpress.com/2014/02/12/why-i-read-the-network-nonsense-papers/.

Sun, B., Danisch, M., Chan, T. H., and Sozio, M. (2020).

Kclist++: A simple algorithm for finding k-clique densest subgraphs in large graphs.

Proc. VLDB Endow., 13(10):1628–1640.

Tsourakakis, C. (2015).

The k-clique densest subgraph problem.

In Proceedings of the 24th International Conference on World Wide Web, pages 1122–1132. International World Wide Web Conferences Steering Committee.

references XII

Ugander, J., Backstrom, L., and Kleinberg, J. (2013).

Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections.

In Proc. WWW '13, pages 1307–1318.

Zhang, W., Deng, Y., Liu, B., Pan, S., and Bing, L. (2024).

Sentiment analysis in the era of large language models: A reality check.

In Findings of the Association for Computational Linguistics: NAACL 2024, pages 3881–3906.