Dense Subgraph Discovery (DSD): Theory and Applications

Charalampos (Babis) E. Tsourakakis[1,2]
Tianyi Chen [1]

[1]Boston University

[2]ISI Fondazione

SDM 2021

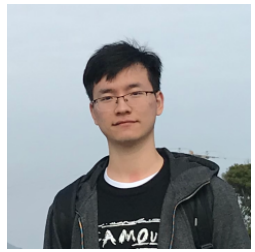
Tutorial

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

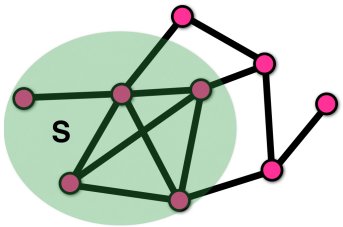
. . . is densely connected

why we care?

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

. . . and its variants

Densest subgraph problem

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

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

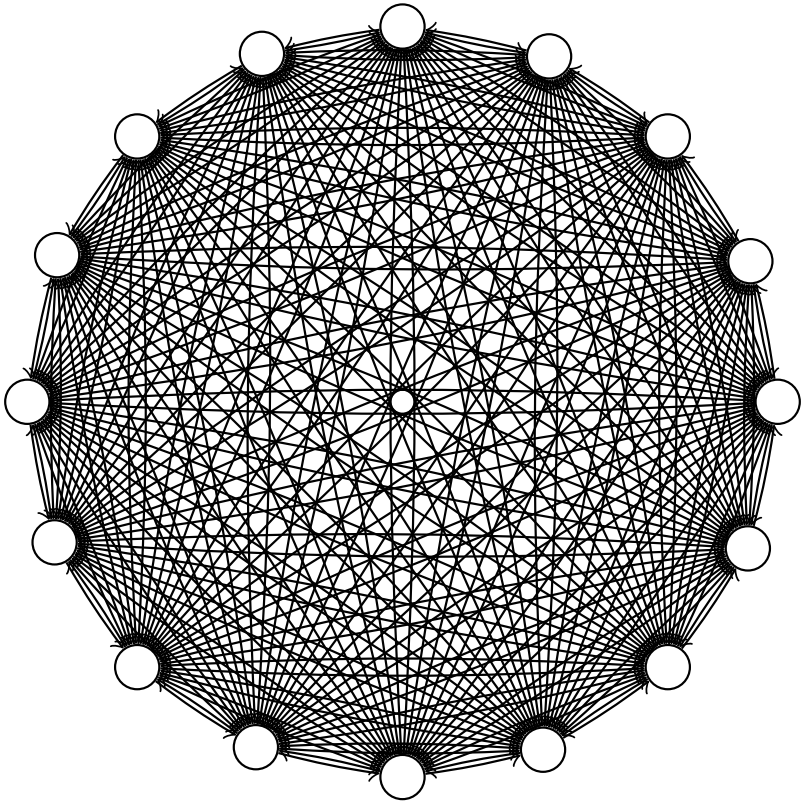
- $\max_{S \subseteq V} \rho(S)$ **Poly-time solvable** (via max flows)

  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), $|S| \leq k$ (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

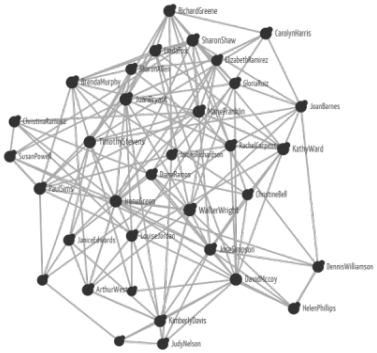
Applications – Correlation mining

correlation mining: a general framework with many applications

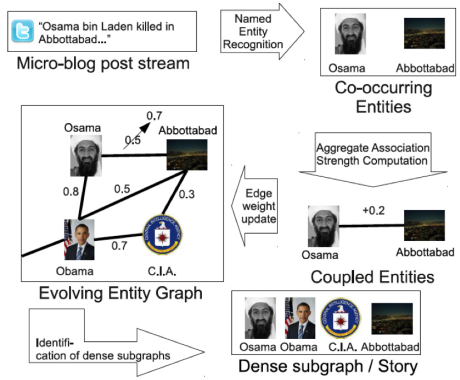
- data is converted into a graph
- vertices correspond to entities
- an edge between two entities denotes strong correlation
  1. stock correlation network: data represent stock timeseries
  2. gene correlation networks: data represent gene expression
- dense subsets of vertices correspond to highly 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

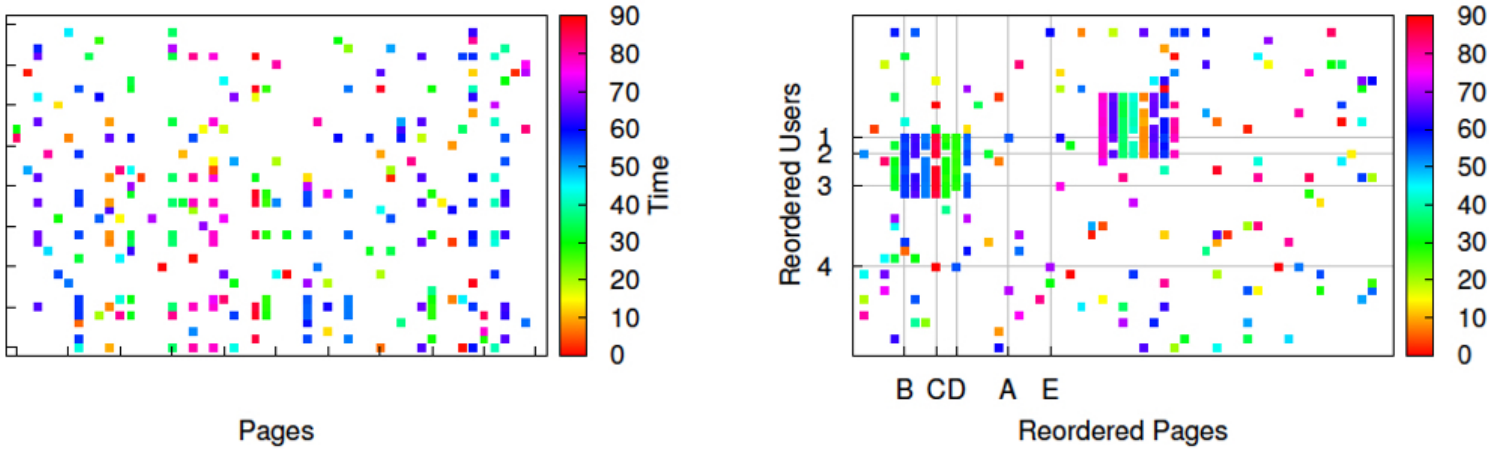


Youtube

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

Applications – fraud detection

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



source: [Beutel et al., 2013]

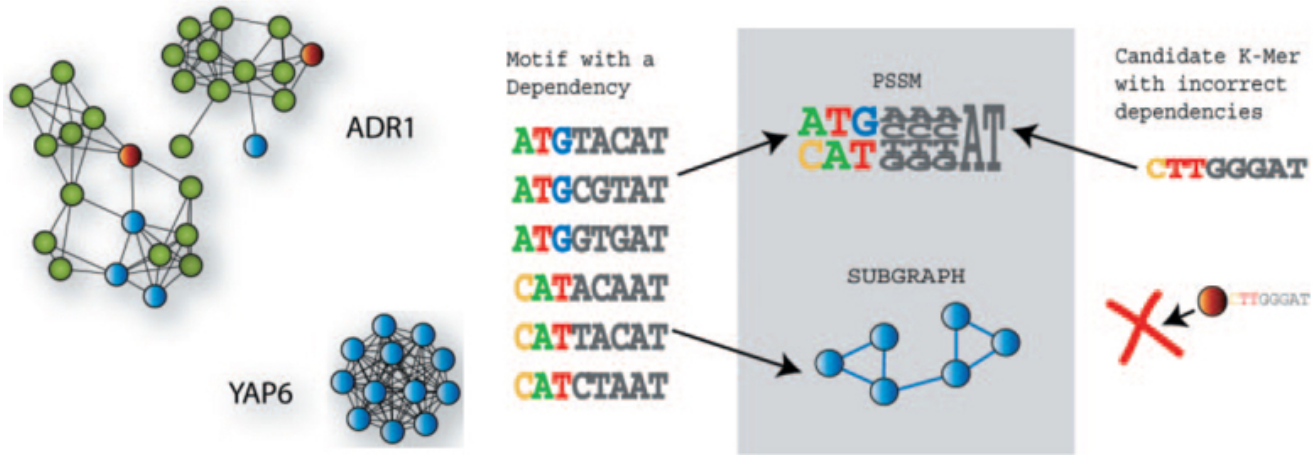
Applications – e-commerce

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 :

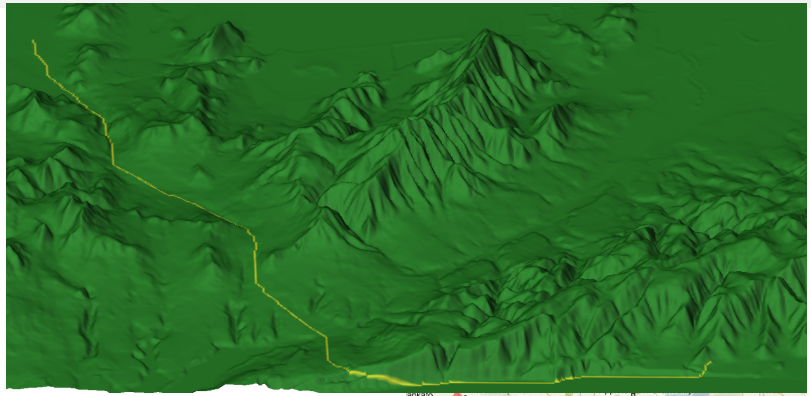
- driving directions
- indoor/terrain navigation
- routing in comm./sensor networks
- moving agents in game maps
- proximity in social/collab. networks

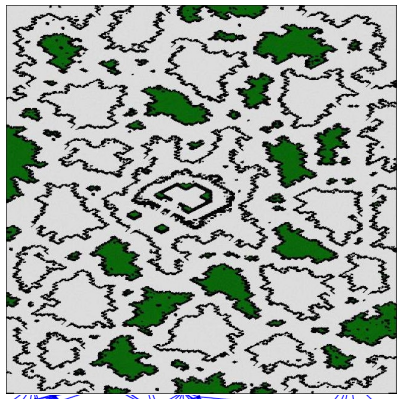
existing solutions :

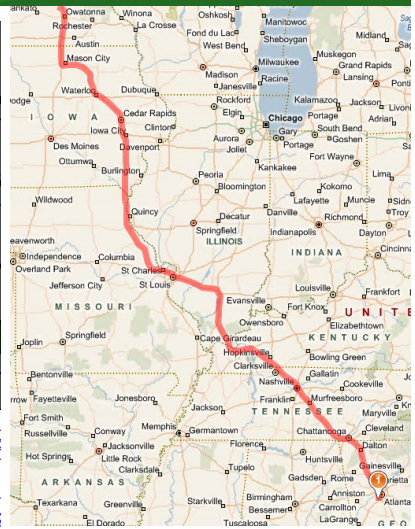
- graph searches are too slow
- fast algorithms are often heuristics
- or tailored to specific graph classes

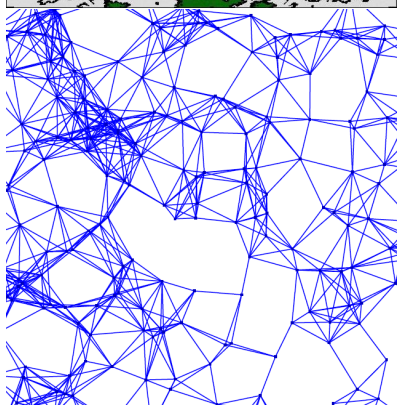
goals :

- fast exact queries
- scalability to large graphs
- wide range of inputs

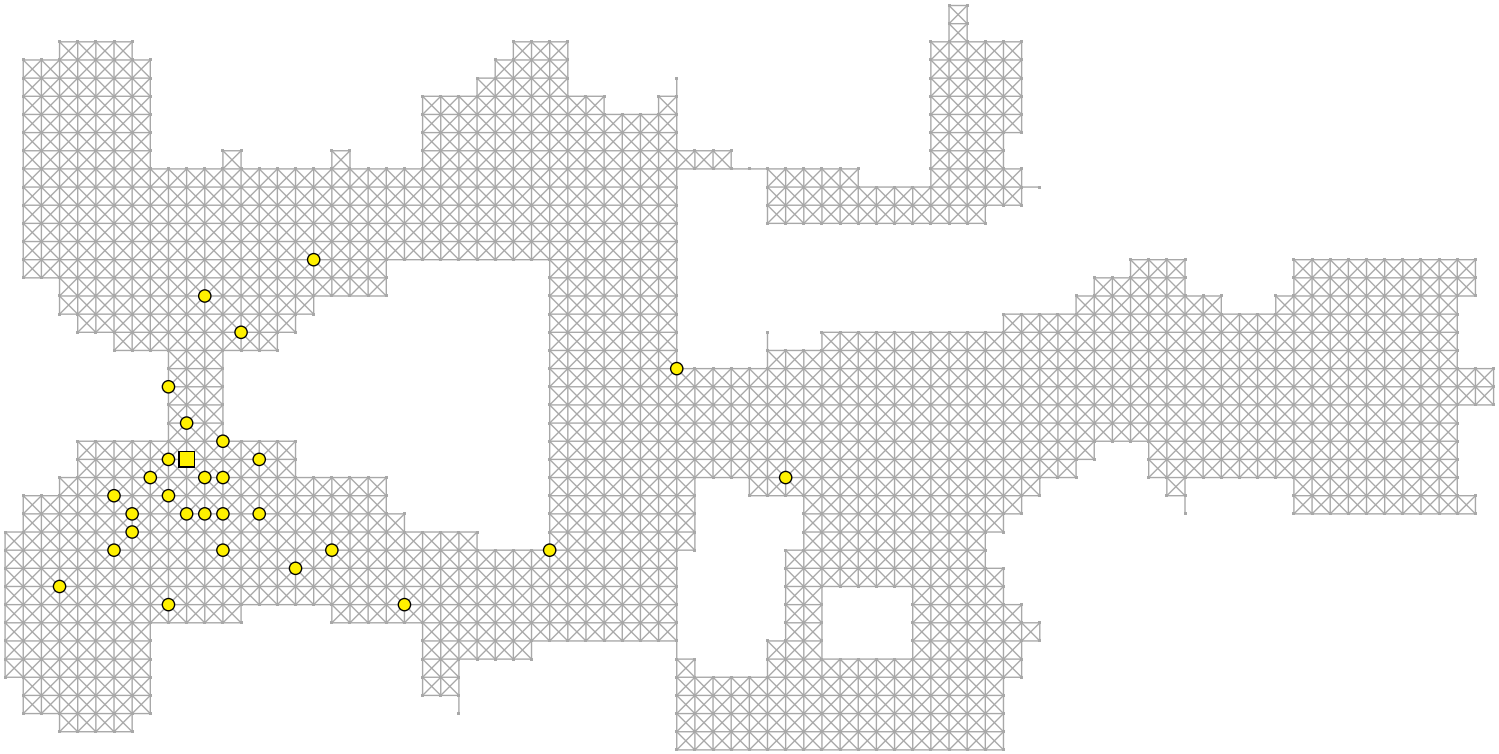








Applications – distance queries in graphs

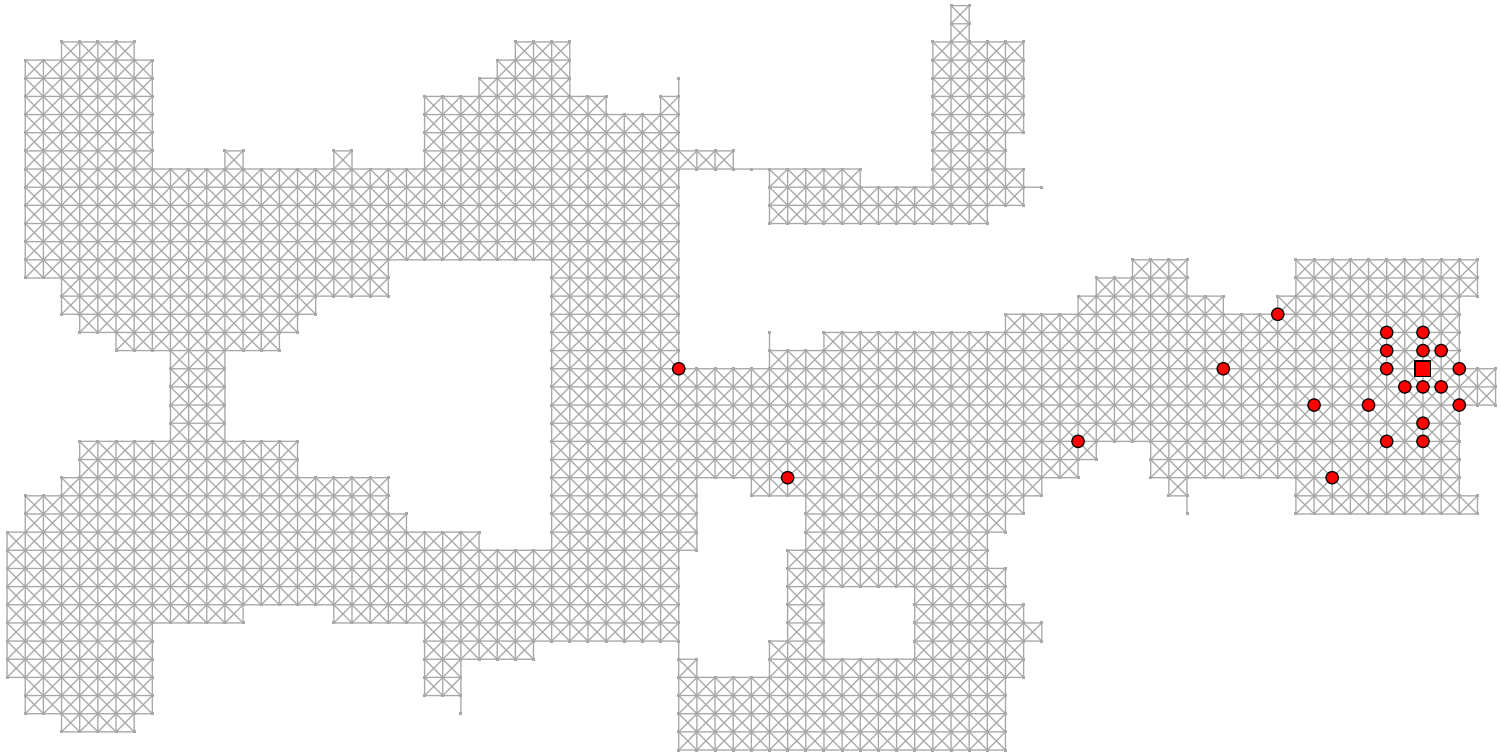


- $L(u) \equiv$ set of pairs $(v, \text{dist}(u, v))$

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

figure from [Delling et al., 2014]

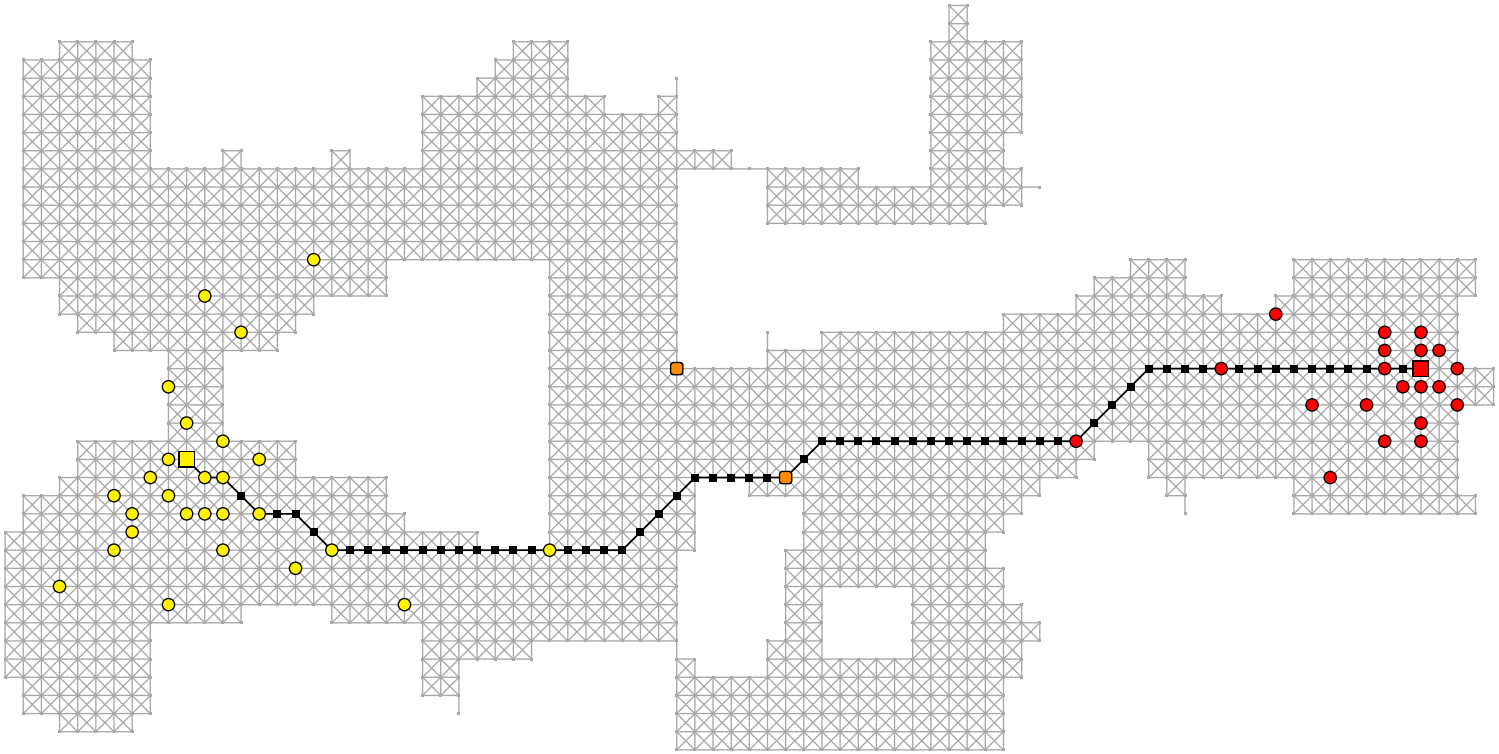
Applications – distance queries in graphs



- 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 v in $L(s) \cap L(t)$ minimizing $\text{dist}(s, v) + \text{dist}(v, t)$

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 θ fraction of the transactions

issue number	heroes
1	Iceman, Storm, Wolverine
2	Aurora, Cyclops, Magneto, Storm
3	Beast, Cyclops, Iceman, Magneto
4	Cyclops, Iceman, Storm, Wolverine
5	Beast, Iceman, Magneto, Storm

e.g., {Iceman, Storm} appear in 60% of issues

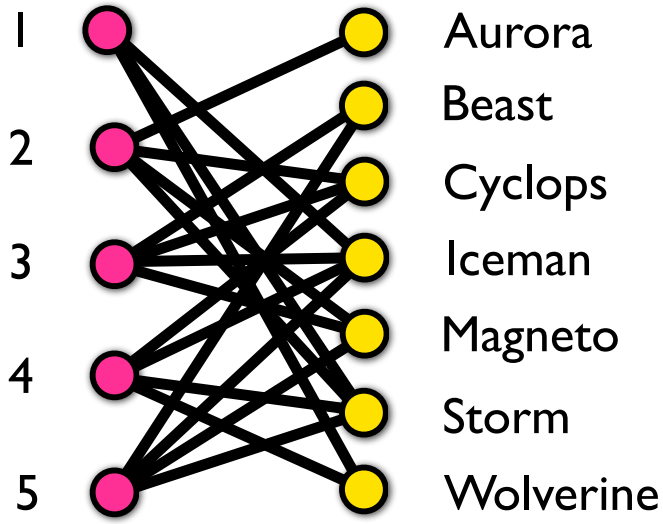
Applications – frequent itemsets and dense subgraphs

id	heroes
1	Iceman, Storm, Wolverine
2	Aurora, Cyclops, Magneto, Storm
3	Beast, Cyclops, Iceman, Magneto
4	Cyclops, Iceman, Storm, Wolverine
5	Beast, Iceman, Magneto, Storm

$\Leftrightarrow$

	A	B	C	I	M	S	W
1	0	0	0	1	0	1	1
2	1	0	1	1	1	0	0
3	0	1	1	1	1	0	0
4	0	0	1	1	0	1	1
5	0	1	0	1	1	1	0

$\Leftrightarrow$



- transaction data $\Leftrightarrow$ binary data $\Leftrightarrow$ bipartite graphs

Frequent pattern mining: Apriori, FP-growth, ...

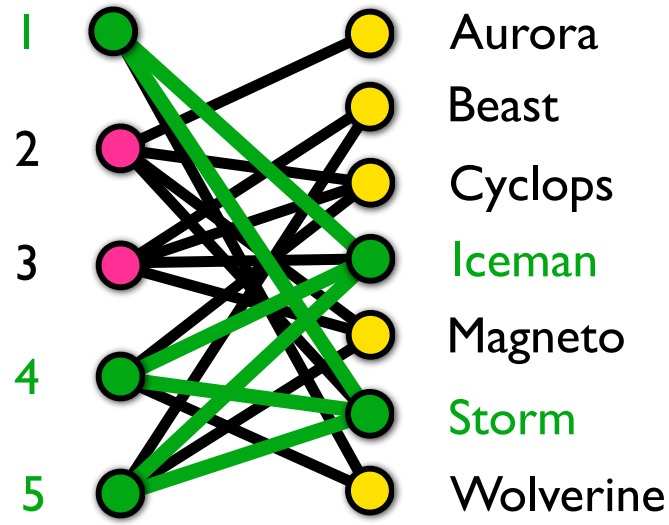
Applications – frequent itemsets and dense subgraphs

id	heroes
1	Iceman, Storm, Wolverine
2	Aurora, Cyclops, Magneto, Storm
3	Beast, Cyclops, Iceman, Magneto
4	Cyclops, Iceman, Storm, Wolverine
5	Beast, Iceman, Magneto, Storm

$\Leftrightarrow$

	A	B	C	I	M	S	W
1	0	0	0	1	0	1	1
2	1	0	1	1	1	0	0
3	0	1	1	1	1	0	0
4	0	0	1	1	0	1	1
5	0	1	0	1	1	1	0

$\Leftrightarrow$



- transaction data $\Leftrightarrow$ binary data $\Leftrightarrow$ bipartite graphs
- frequent itemsets $\Leftrightarrow$ bi-cliques

Applications – finding web communities

[Kumar et al., 1999]

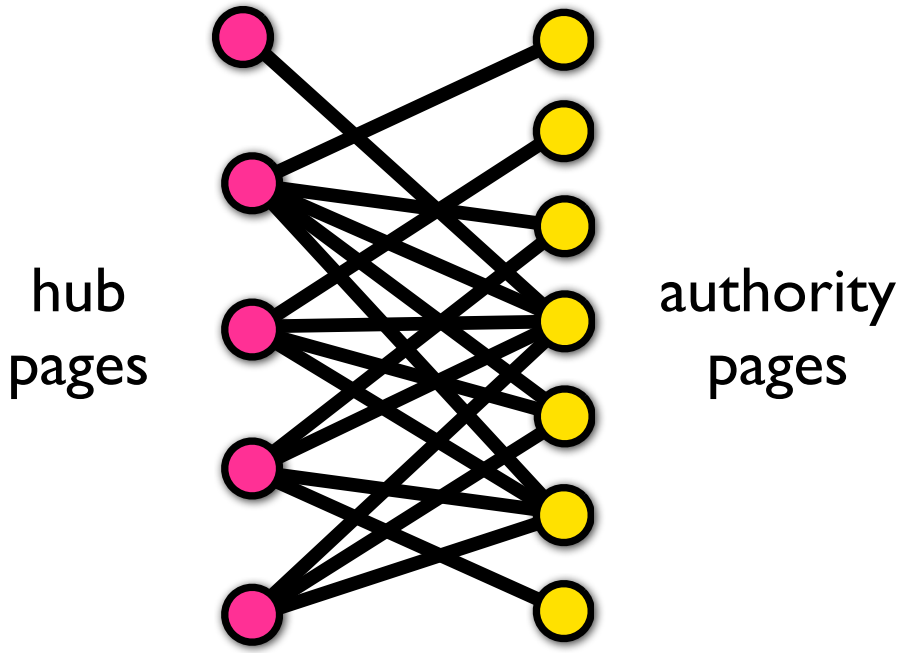
- hypothesis: web communities consist of hub-like pages and authority-like pages
- key observations:

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

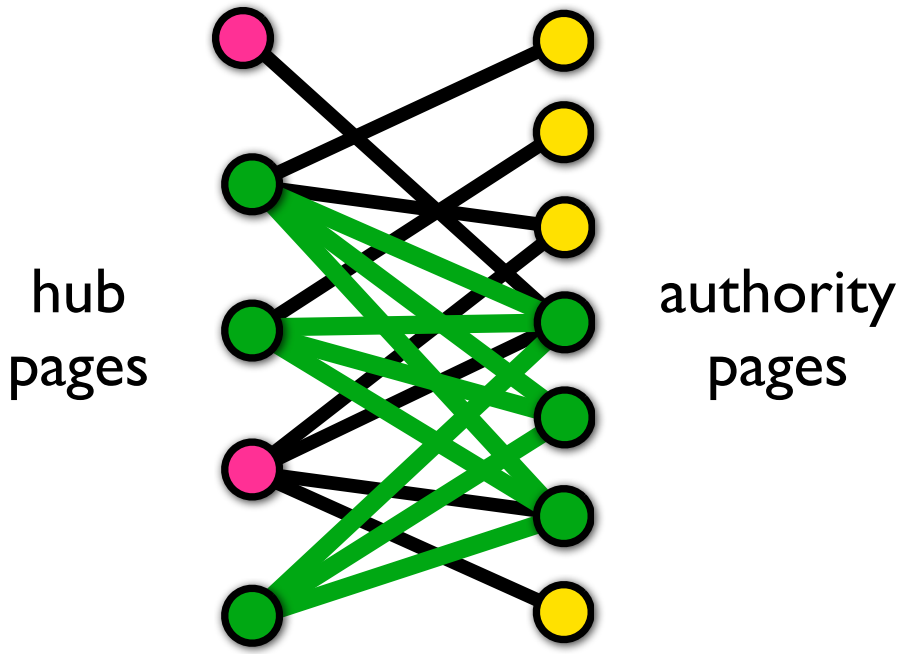
a web community



[Kumar et al., 1999]

Applications – finding web communities

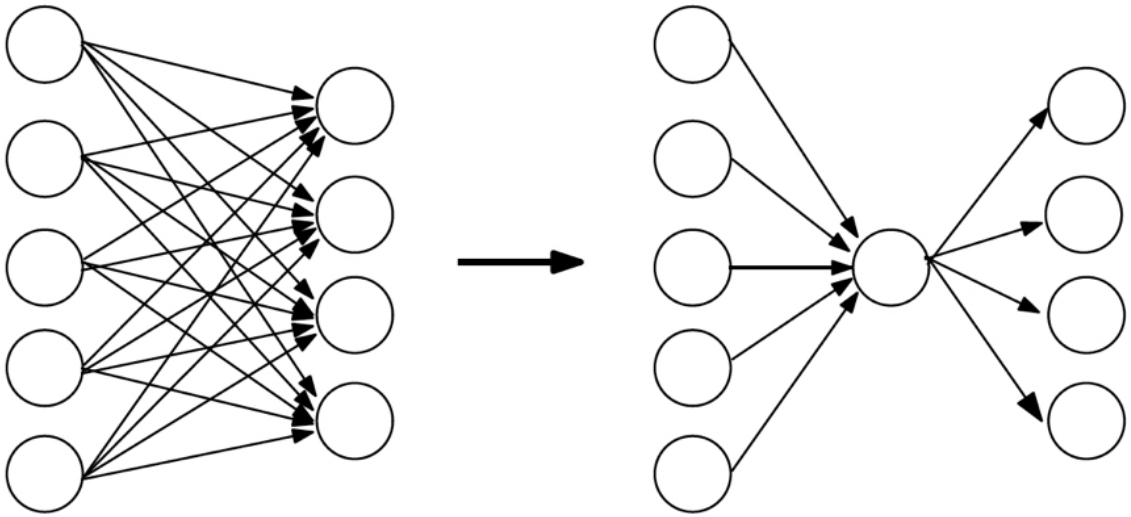
web communities containts small cores



[Kumar et al., 1999]

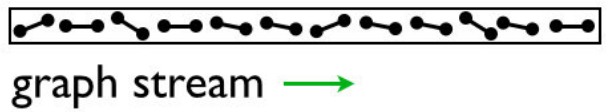
Applications – graph compression

- 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]

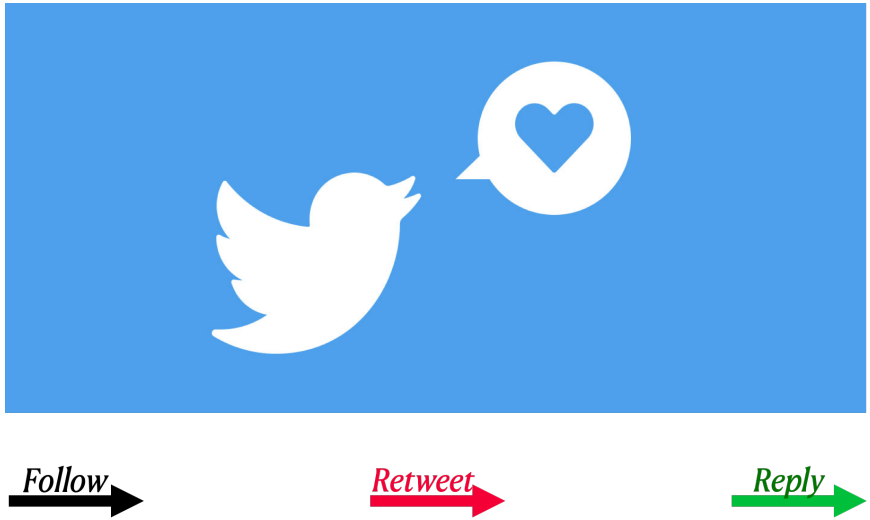


Applications – big and dynamic graphs

- 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.

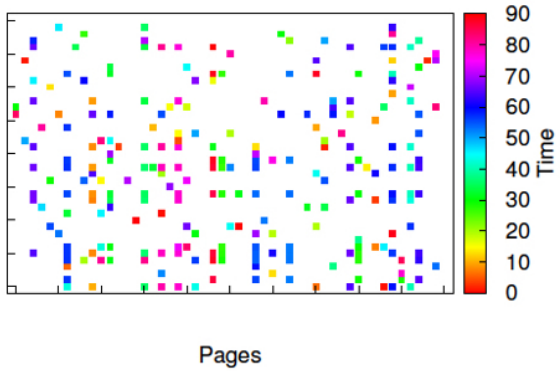


Applications – Exclusion density queries

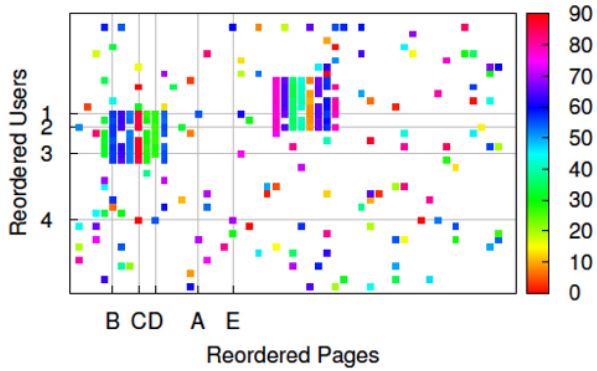


- Twitter users may interact in more than one ways, e.g., FOLLOW, RETWEET, REPLY.
- How can we find a subgraph that contains (i) lots of **reply** interactions, (ii) and no **follow** interactions?
- Such graph queries can be used to mine "interesting" groups of people and interactions, and also study behavioral patterns of Twitter users.

Applications – Anomaly detection using DSD



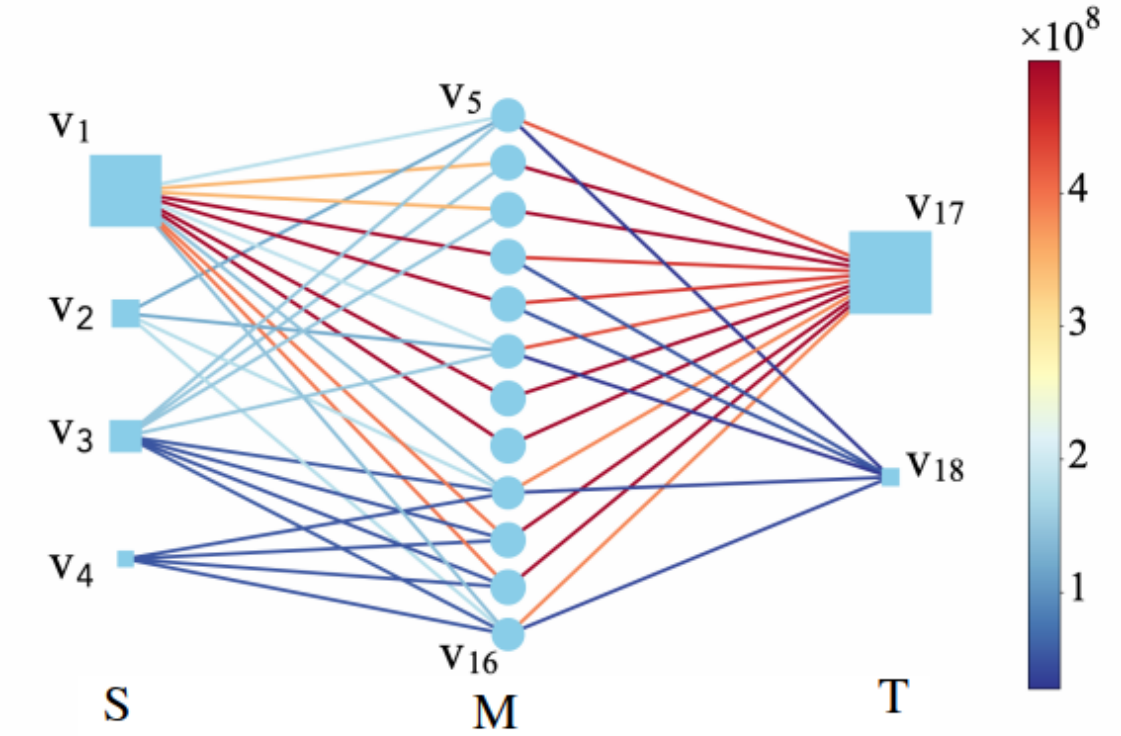
Online social networks



Fraud detection [Hooi et al., 2016]

How do we find graph anomalies using dense subgraph structures?

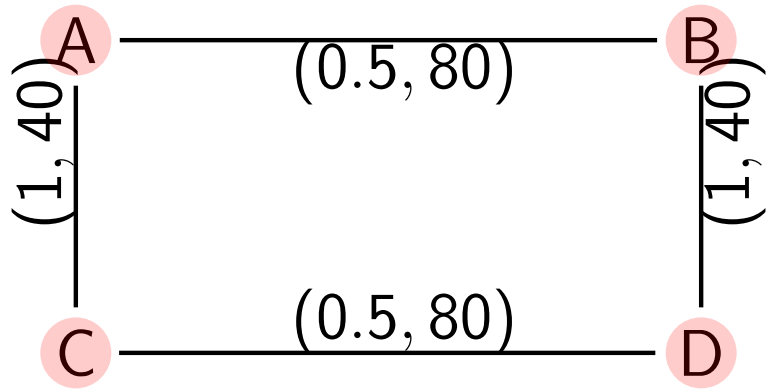
Applications – Anomaly detection using DSD



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

$$\deg_X(u) = |\{v \in X \text{ such that } (u, v) \in E\}|$$

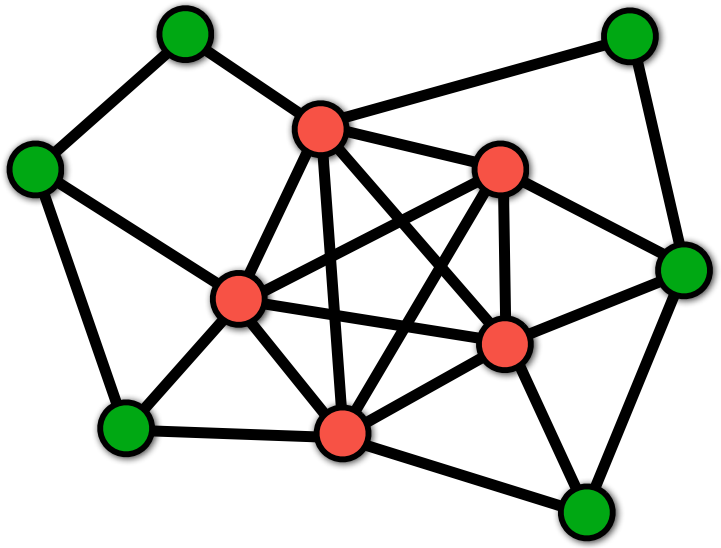
- degree of a node $u \in V$ is $\deg(u) = \deg_V(u)$
- edges between $S \subseteq V$ and $T \subseteq V$ are

$$E(S, T) = \{(u, v) \text{ such that } u \in S \text{ and } v \in T\}$$

  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):

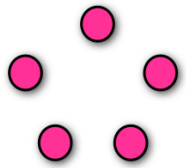
$$d(S) = \frac{2\,|E(S,S)|}{|S|} = \frac{2\,|E(S)|}{|S|}, \rho(S) = \frac{|E(S)|}{|S|}$$

(sometimes just drop 2)

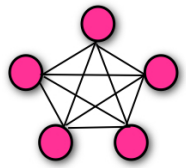
- edge ratio:

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

$$0 \leq \boxed{\delta(S)} \leq 1$$



Independent set



Clique

other density measures

- k-core: every vertex in S is connected to

  at least k other vertices in S

- α-quasiclique: the set S has at least $\alpha \binom{|S|}{2}$ edges

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

- [Kawase and Miyauchi, 2017] f density (generalization of edge density):

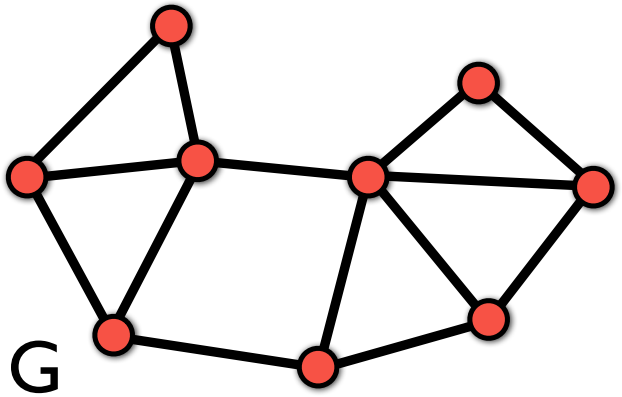
$$d_f(S) = \frac{|E(S)|}{f(|S|)}$$

  e.g., $f(|S|) = |S|, |S|^{1.5}$

Densest subgraph problem: Static graphs

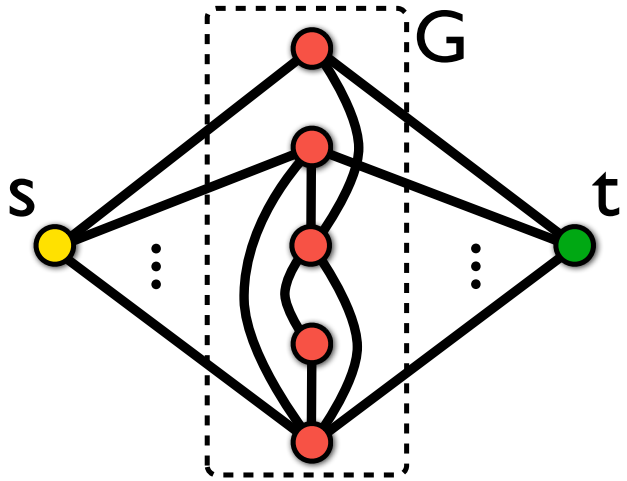
Goldberg's algorithm for densest subgraph

- 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 $d(S) \geq c$?

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

$$2\,|E(S,S)| \geq c|S|$$

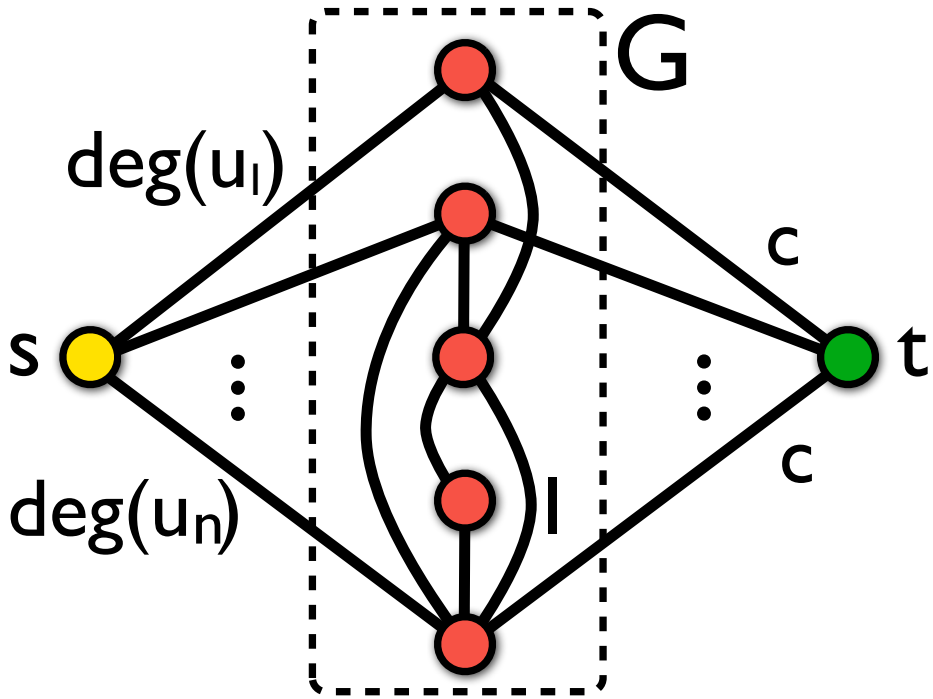
$$\sum_{u \in S} \deg(u) - |E(S,\bar{S})| \geq c|S|$$

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

$$\sum_{u \in \bar{S}} \deg(u) + |E(S,\bar{S})| + c|S| \leq 2\,|E|$$

Goldberg's algorithm for densest subgraph

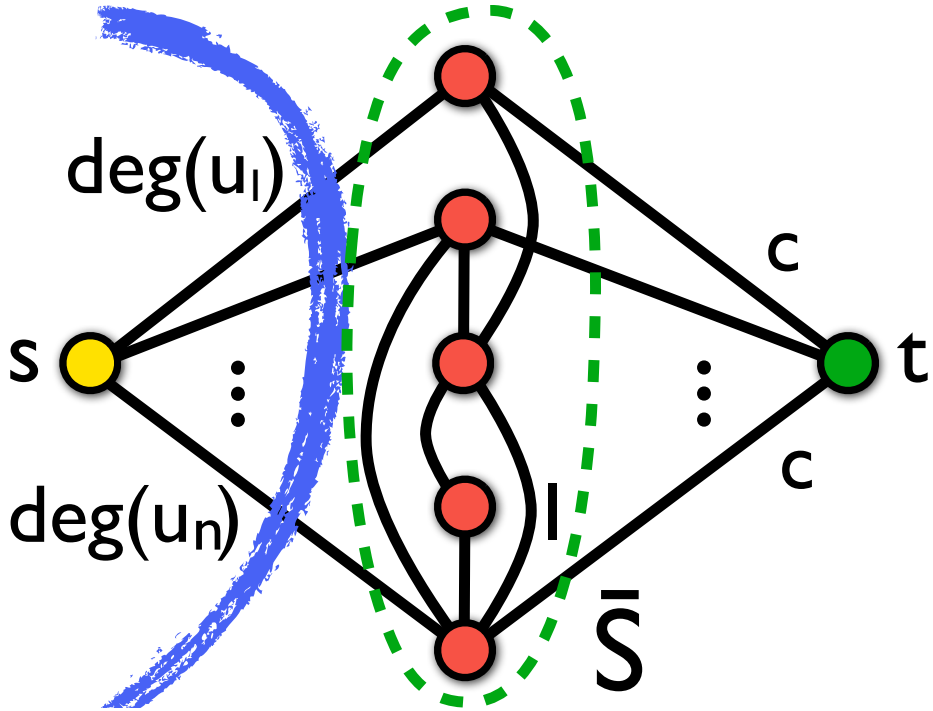
- transformation to min-cut instance



- 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

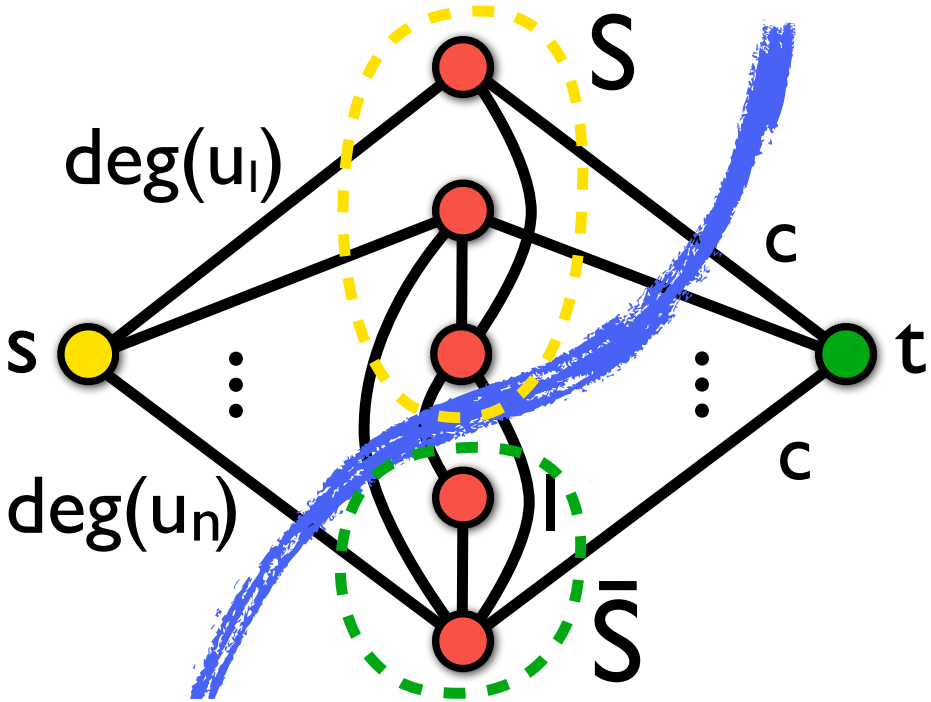
- transform to a min-cut instance



- is there S 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

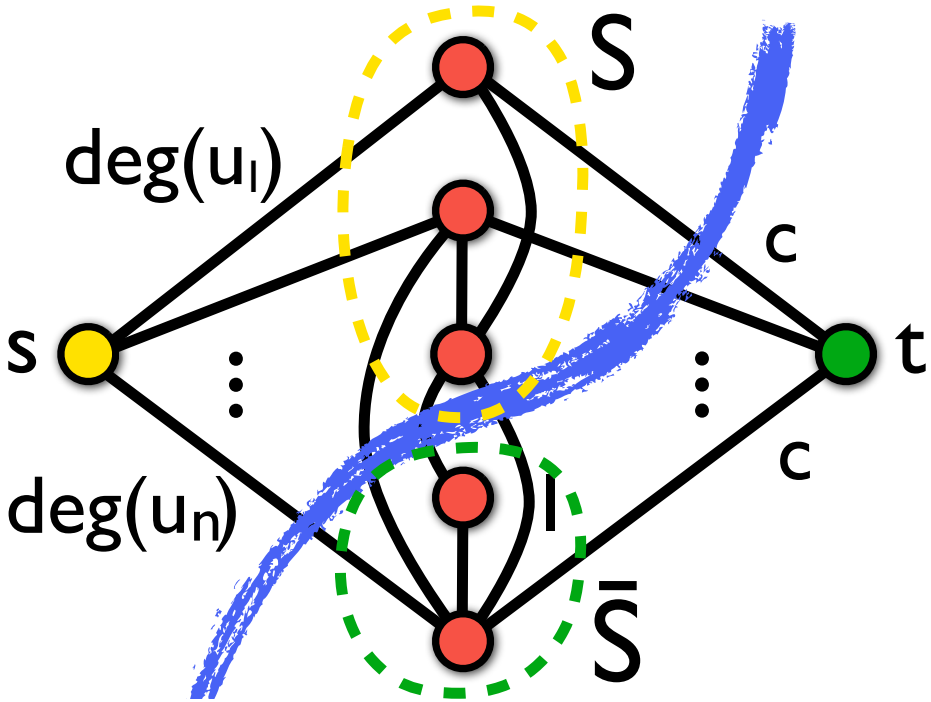
- transform to a min-cut instance



- is there S s.t. $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2\,|E|$?
- $S \neq \emptyset$ gives cut of value $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S|$

Goldberg's algorithm for densest subgraph

- transform to a min-cut instance



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

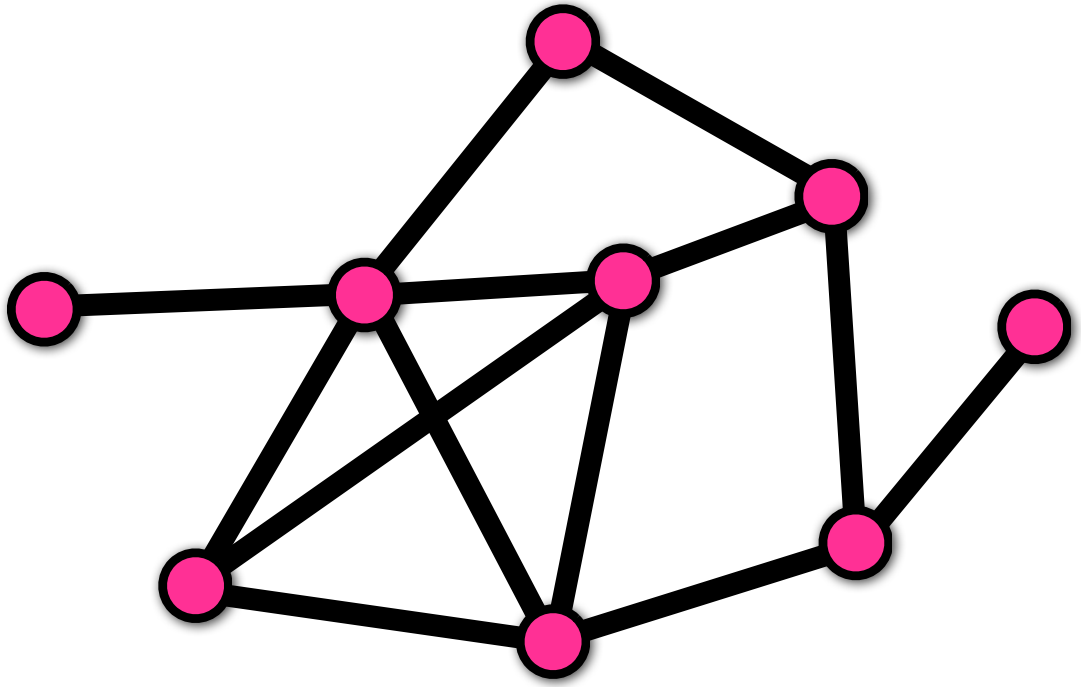
densest subgraph problem – discussion

- Goldberg's algorithm polynomial algorithm, but
- $\mathcal{O}(nm)$ time for one min-cut computation
- not scalable for large graphs (millions of vertices / edges)

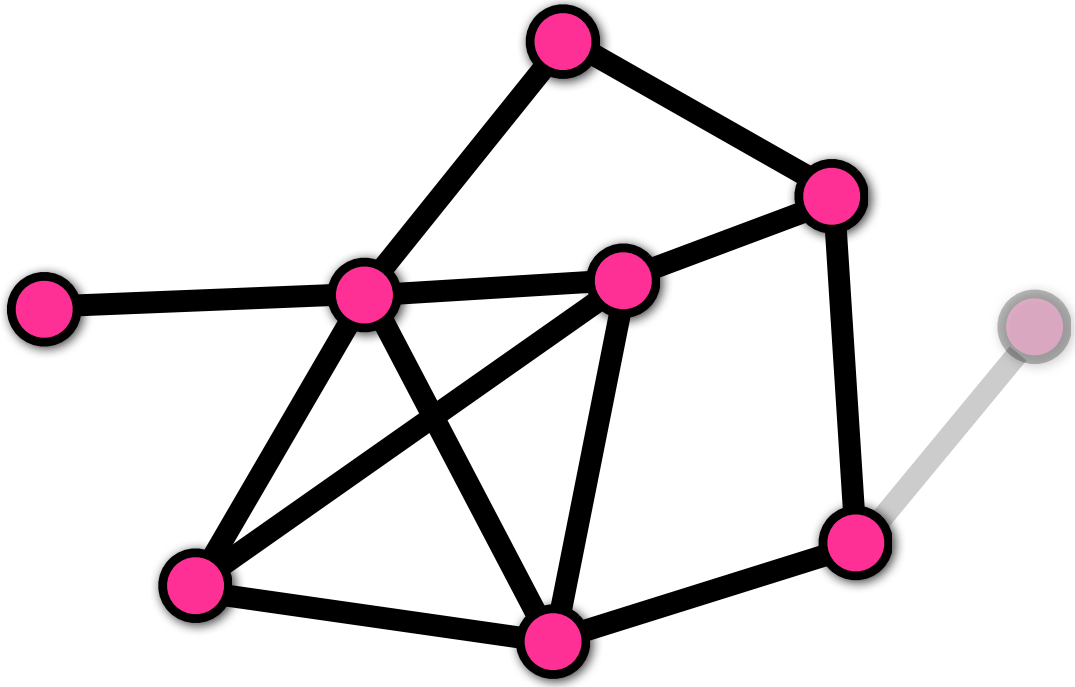
- faster algorithm due to [Charikar, 2000]
- greedy and simple to implement
- approximation algorithm...

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

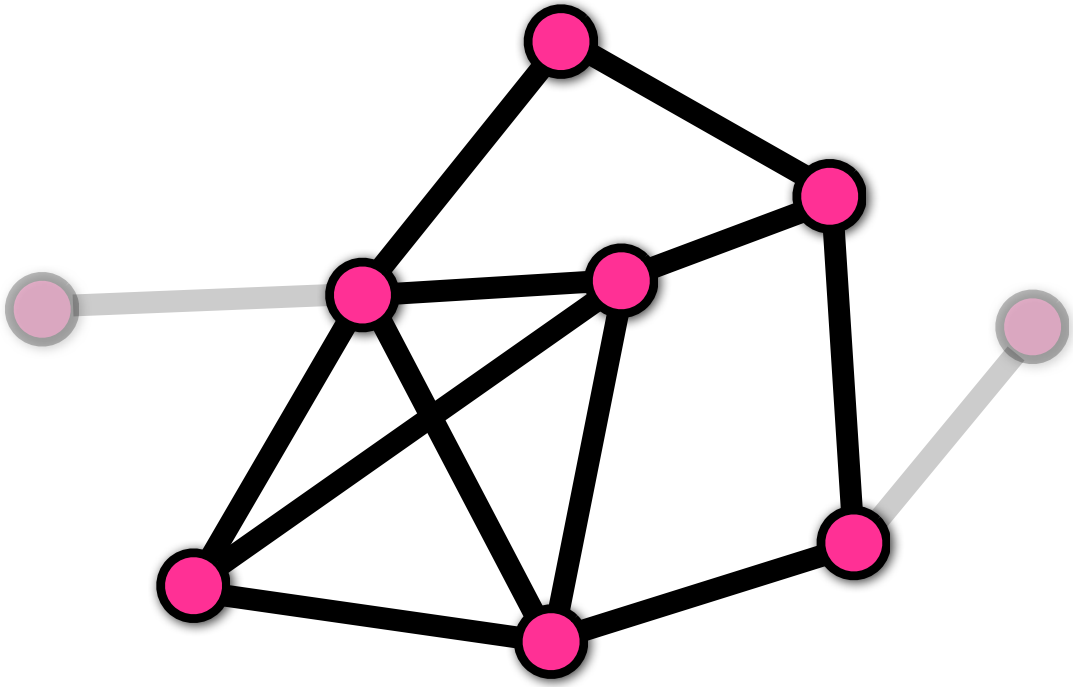
greedy algorithm for densest subgraph — example



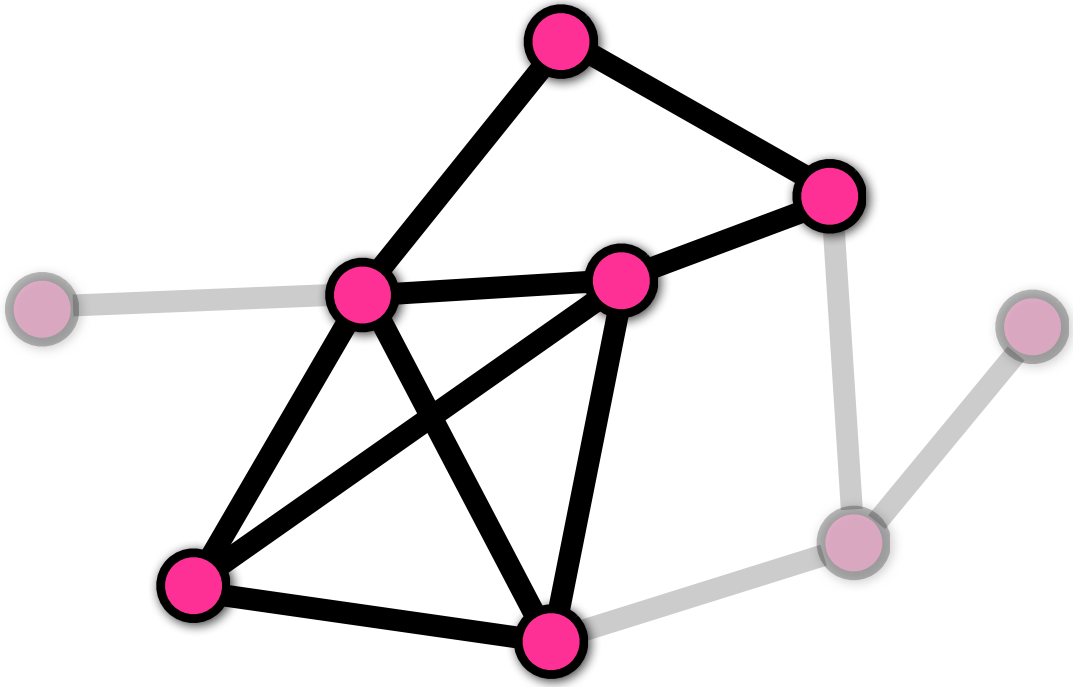
greedy algorithm for densest subgraph — example



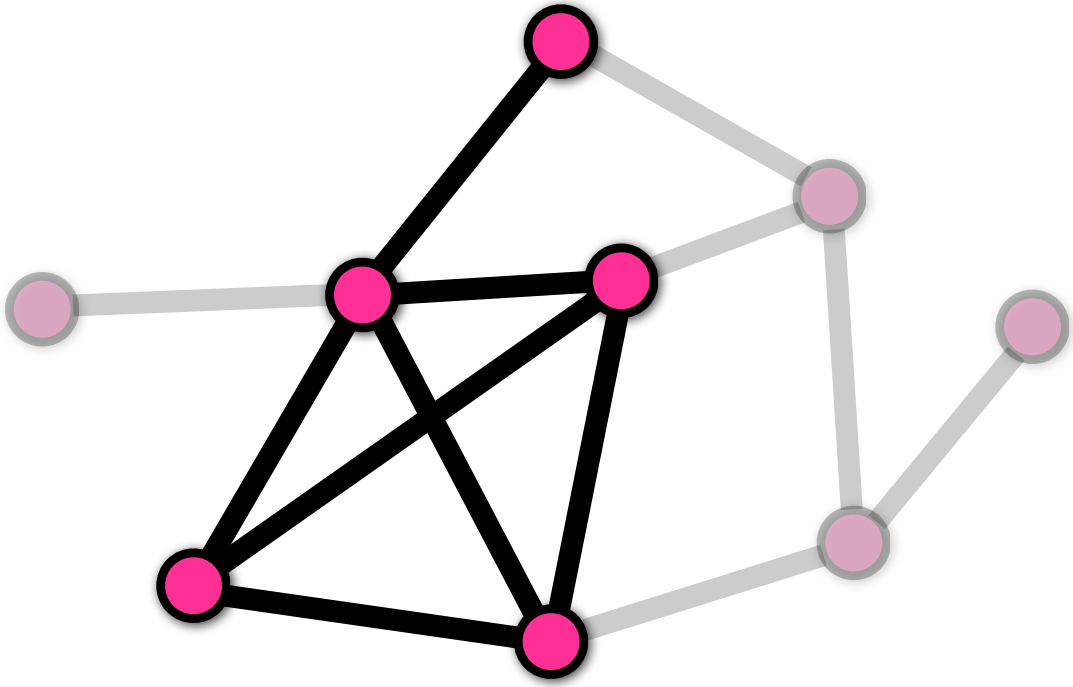
greedy algorithm for densest subgraph — example



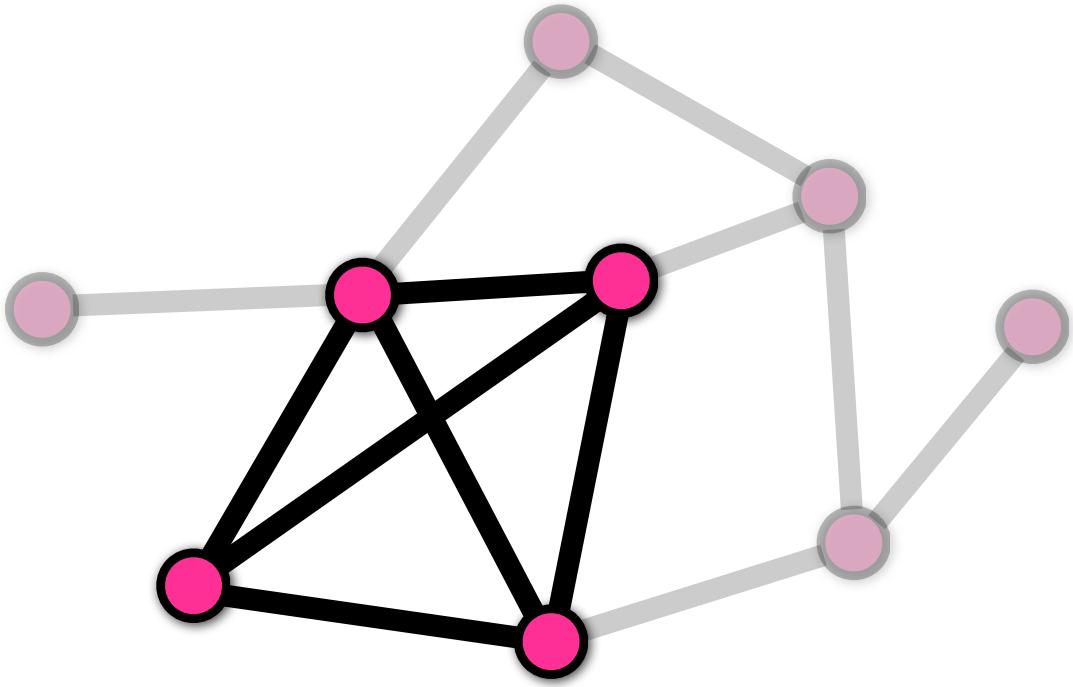
greedy algorithm for densest subgraph — example



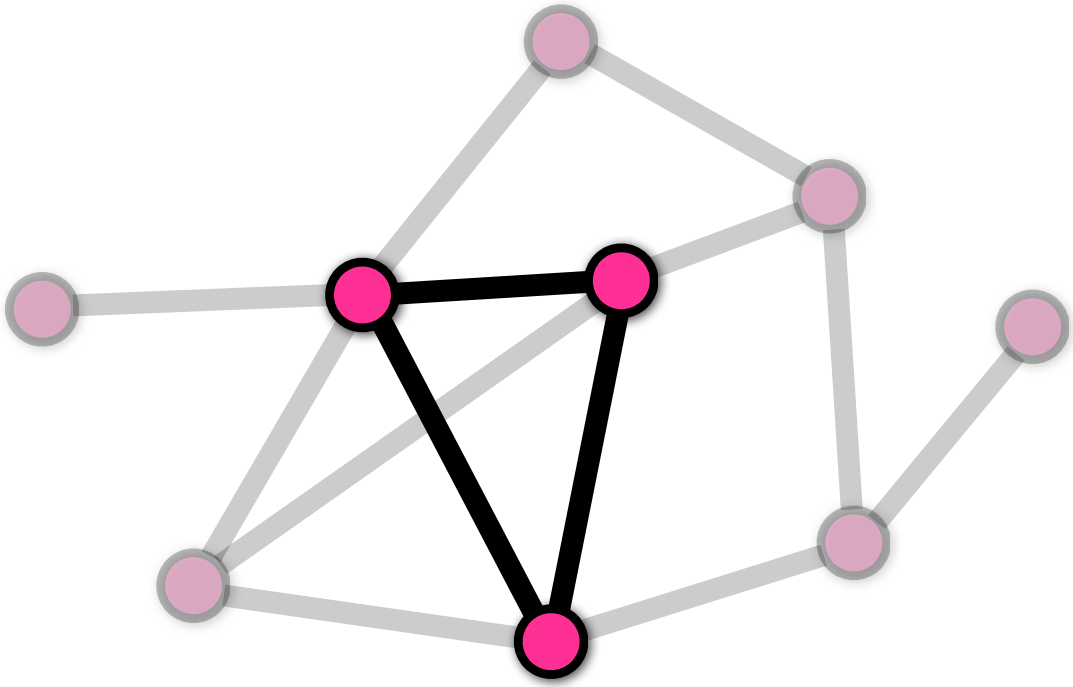
greedy algorithm for densest subgraph — example



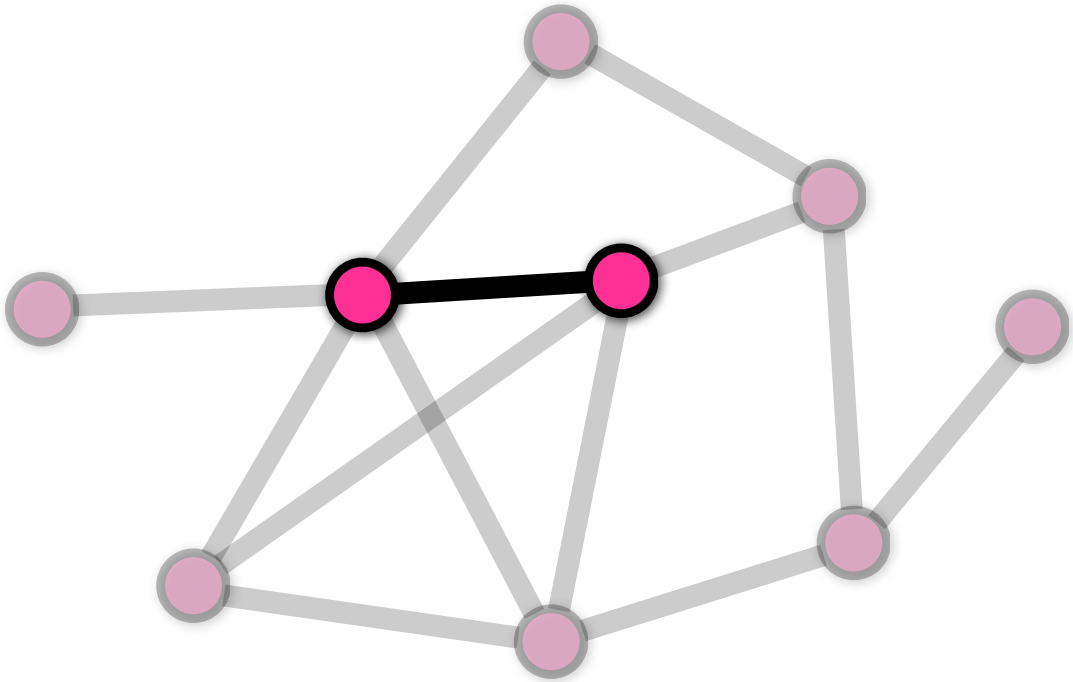
greedy algorithm for densest subgraph — example



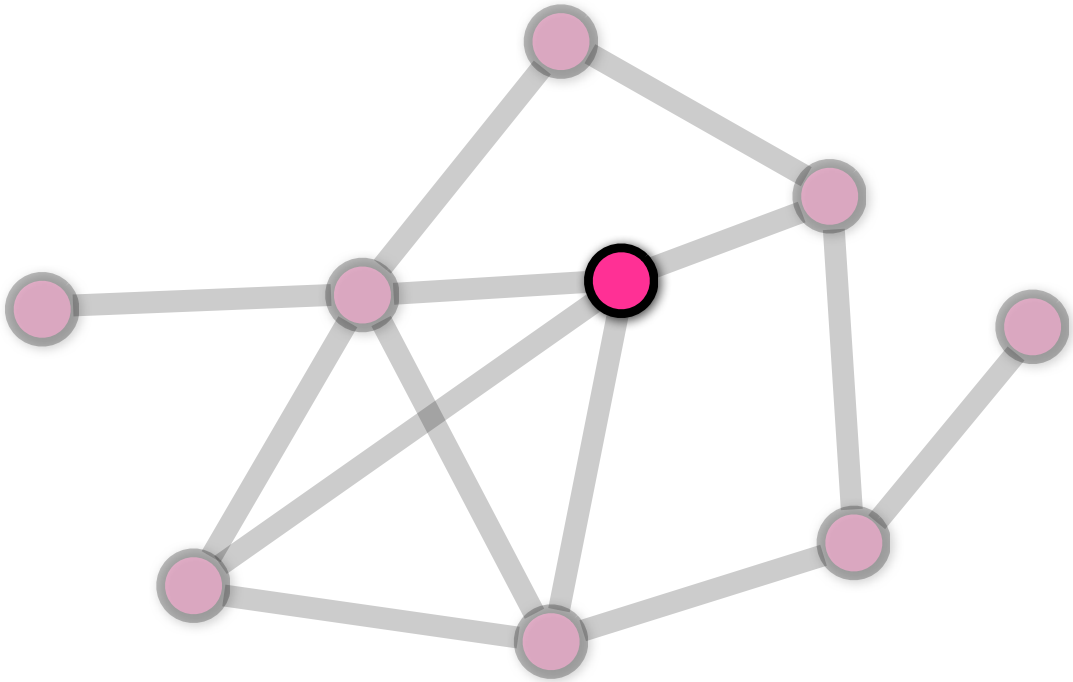
greedy algorithm for densest subgraph — example



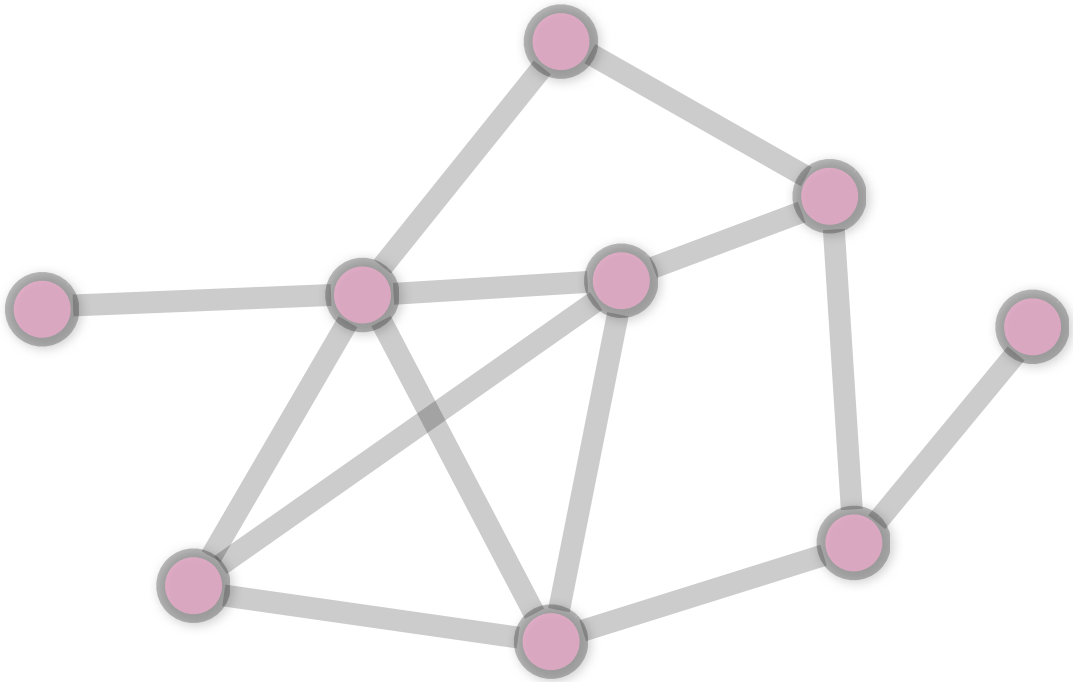
greedy algorithm for densest subgraph — example



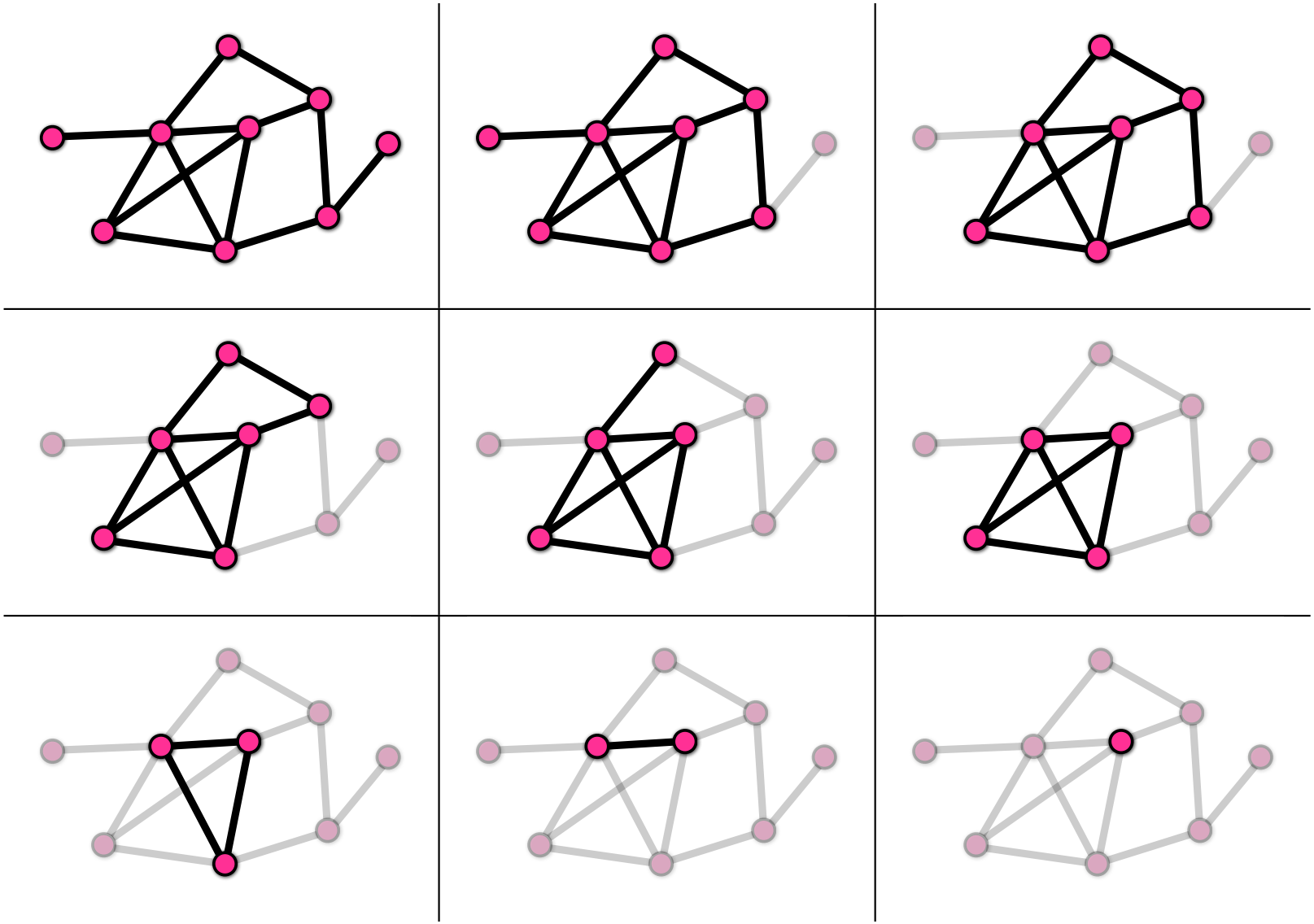
greedy algorithm for densest subgraph — example



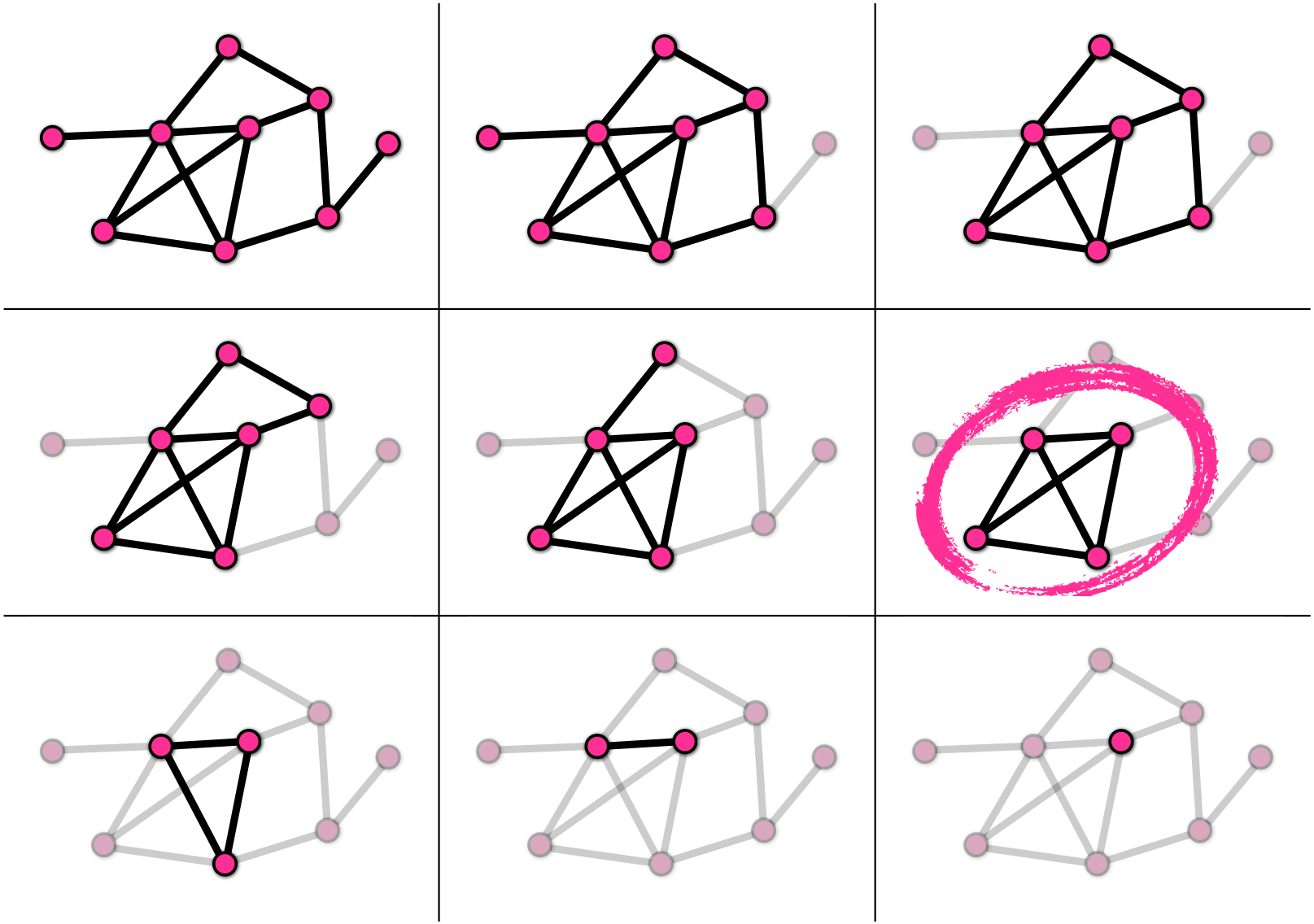
greedy algorithm for densest subgraph — example



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

$$\deg_{S^*}(v) \geq \frac{|e(S^*)|}{|S^*|} = \rho(S^*) = \rho^*$$

proof of 2-approximation guarantee (continued)

([Khuller and Saha, 2009])

- 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 $\mathcal{O}(n+m)$
- for a polynomial problem . . .

  but faster and easier to implement than the exact algorithm

- everything goes through for weighted graphs

  using heaps: $\mathcal{O}(m + n \log n)$

- 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

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

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

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

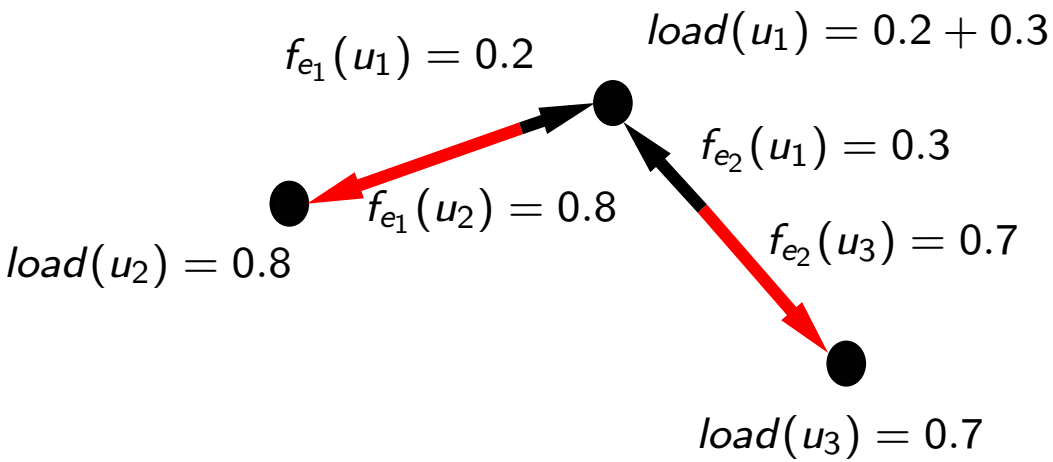
$$\textsc{Dual}(G)$$

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

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

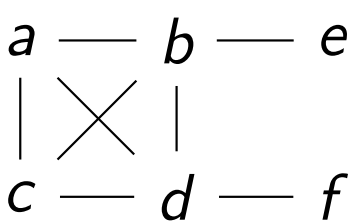


- Each edge pushes its weight (i.e., edge load) to its endpoints.
- The load of each node equals to the sum of all edge weights pushed to it.

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

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

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).

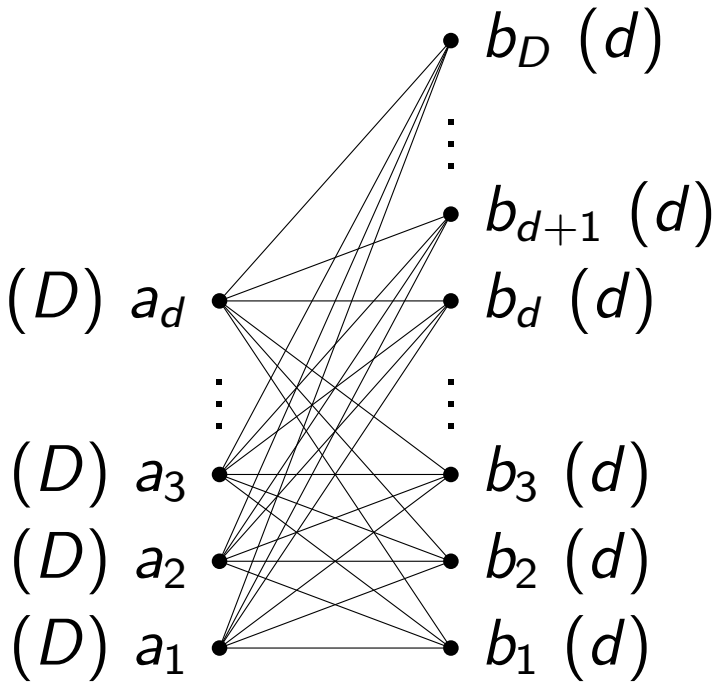
Greedy++ pseudocode – Input $G(V, E), T$

$G_{\text{densest}} \leftarrow G$
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 $\ell_u^{(i-1)} + \deg_H(u)$;
    $\ell_u^{(i)} \leftarrow \ell_u^{(i-1)} + \deg_H(u)$;
    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}

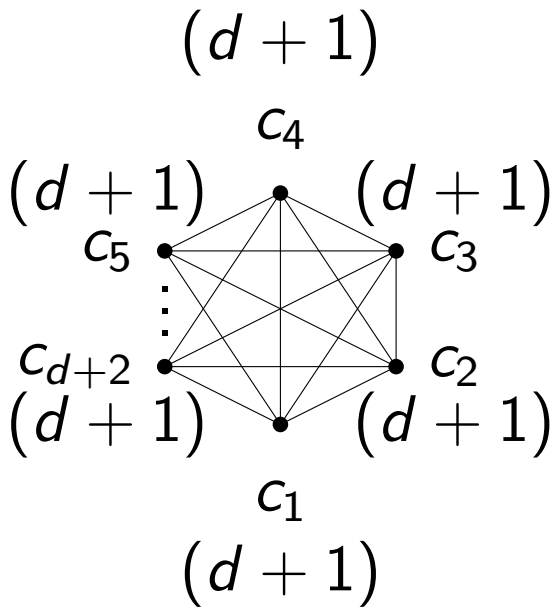
Tight example for Greedy

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

$B = K_{d,D}$



$H_i = K_{d+2}$

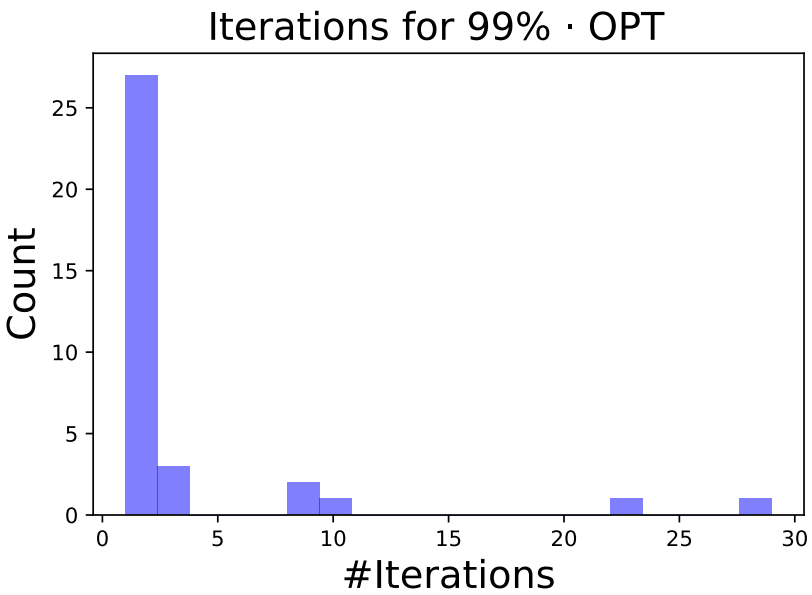


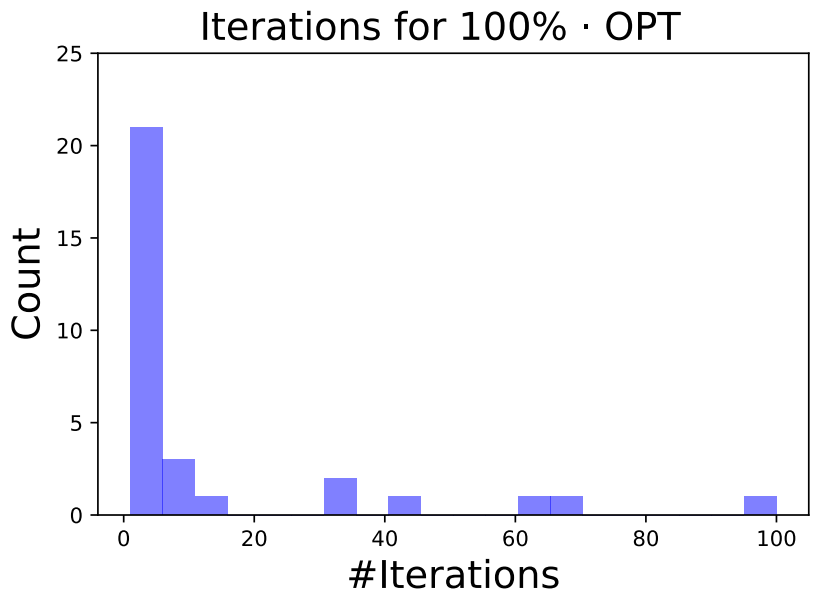
- Charikar gives $\frac{1}{2}$-approximation.
- Flowless with $T = 2$ gives (practically) optimal solution.

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:

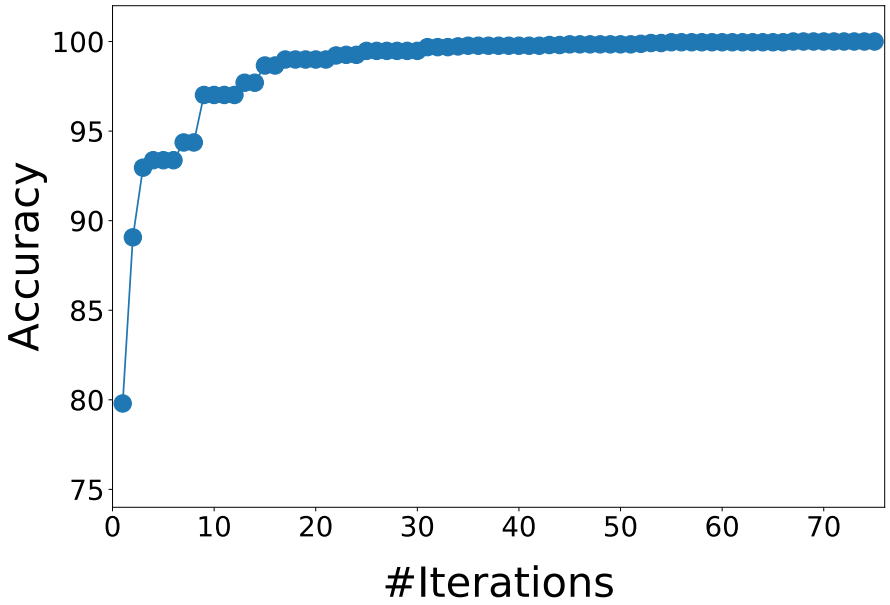


Figure: Amazon co-purchasing network

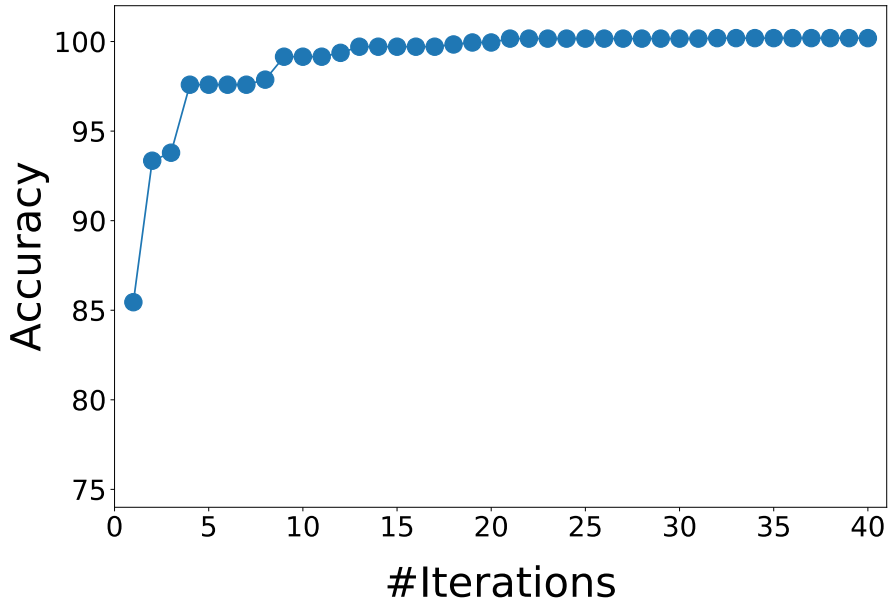


Figure: California road connection network

Experimental results

Scalability and speed of Greedy++:

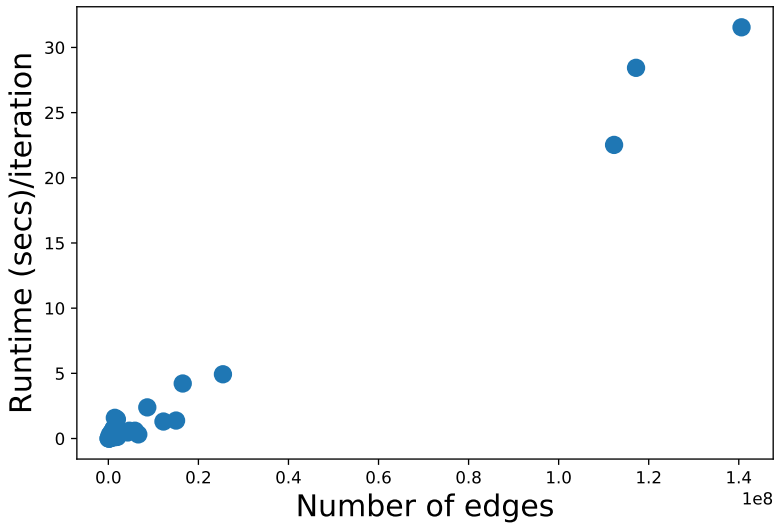


Figure: Runtime in seconds per iteration

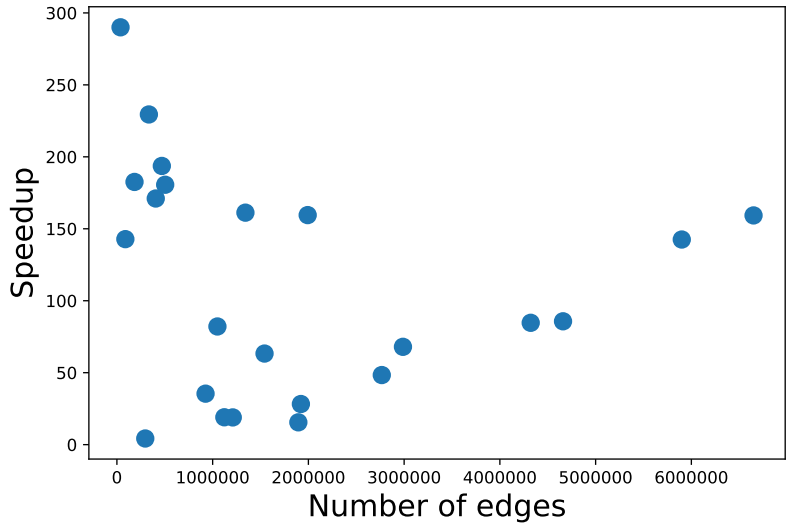


Figure: Speedup over exact algorithm (max-flow based)

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 $144.6\times$ 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.

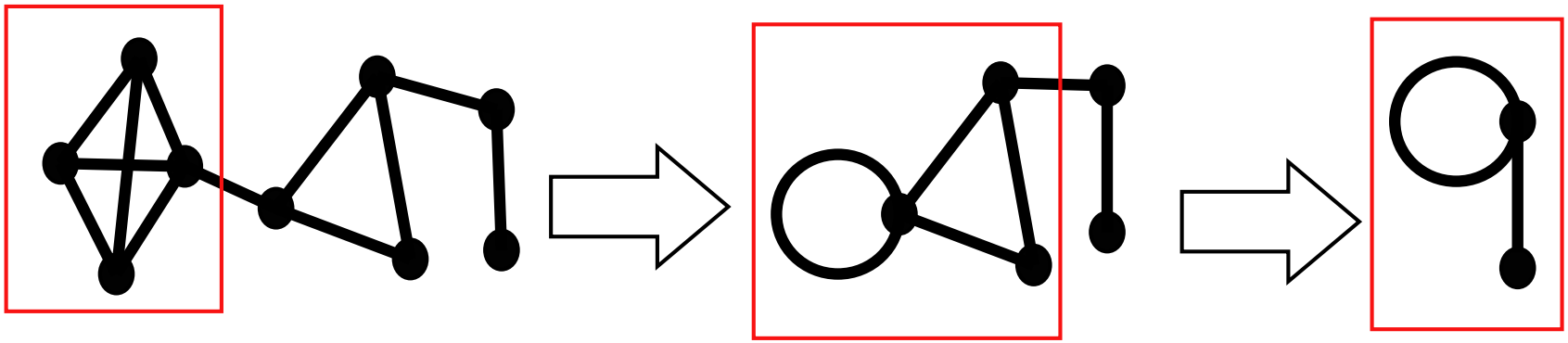
Proposed algorithm: Frank-Wolfe based algorithm

- Large Scale Density-friendly Graph Decomposition via Convex Programming (WWW 2017)

  Danisch-Chan-Sozio

Key Concept - Quotient graph

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



$G_1 = G$ $G_2 = G_1 \backslash S_1$ $G_3 = G_2 \backslash S_2$

Quotient graph of G_1 with respect to node subset S_1 is G_2.
Load vector of G $load^G = [1.5, 1.5, 1.5, 1.5, 1.33, 1.33, 1.33, 1, 1]$.

Dual to Convex Optimization

The $\textsc{Dual}(G)$ is to evenly distribute edge weights to all nodes

$$
\begin{aligned}
\textsc{Dual}(G) \quad \min \quad & D \\
\text{s.t.} \quad & D \geq \sum_{e:u\in e} f_e(u) = load(u), && \forall u \in V \\
& \sum_{u\in e} f_e(u) \geq w_e, && \forall e \in E \\
& f_e(u) \geq 0, && \forall u \in e \in E
\end{aligned} \tag{1}
$$

Consider $Q_G := \sum_{u\in V} load(u)^2$.

Frank-Wolfe Algorithm

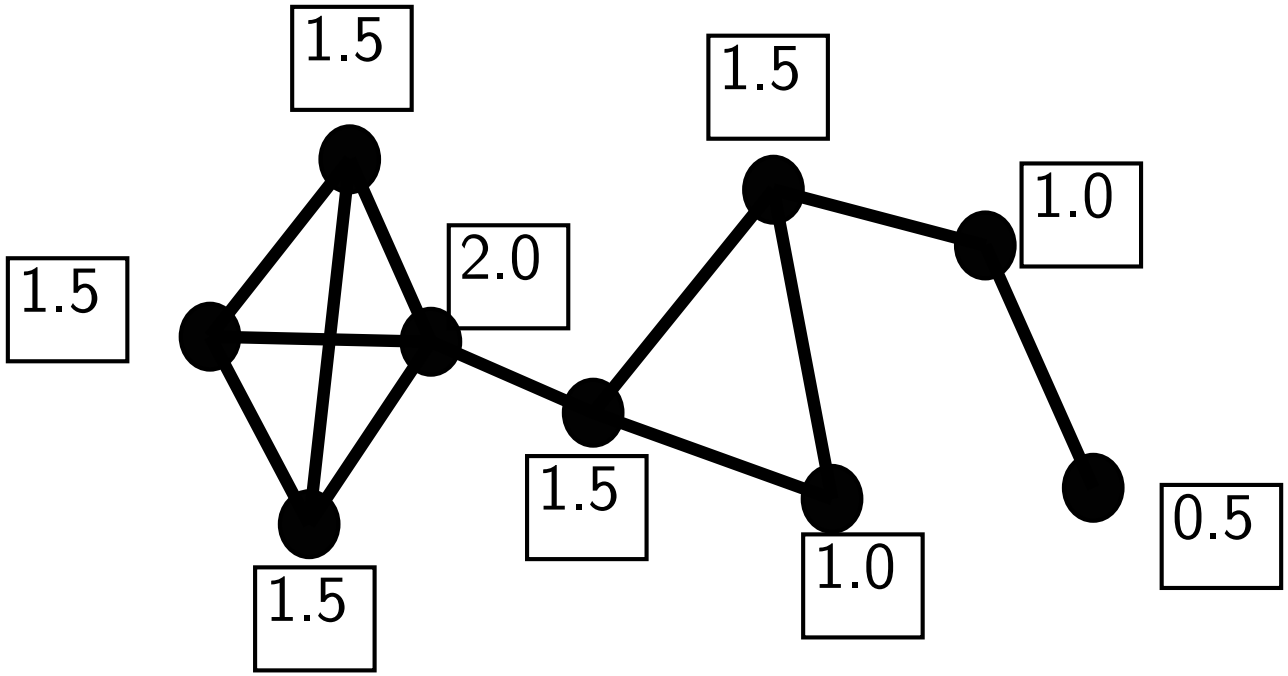
Solve the following convex problem:

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

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

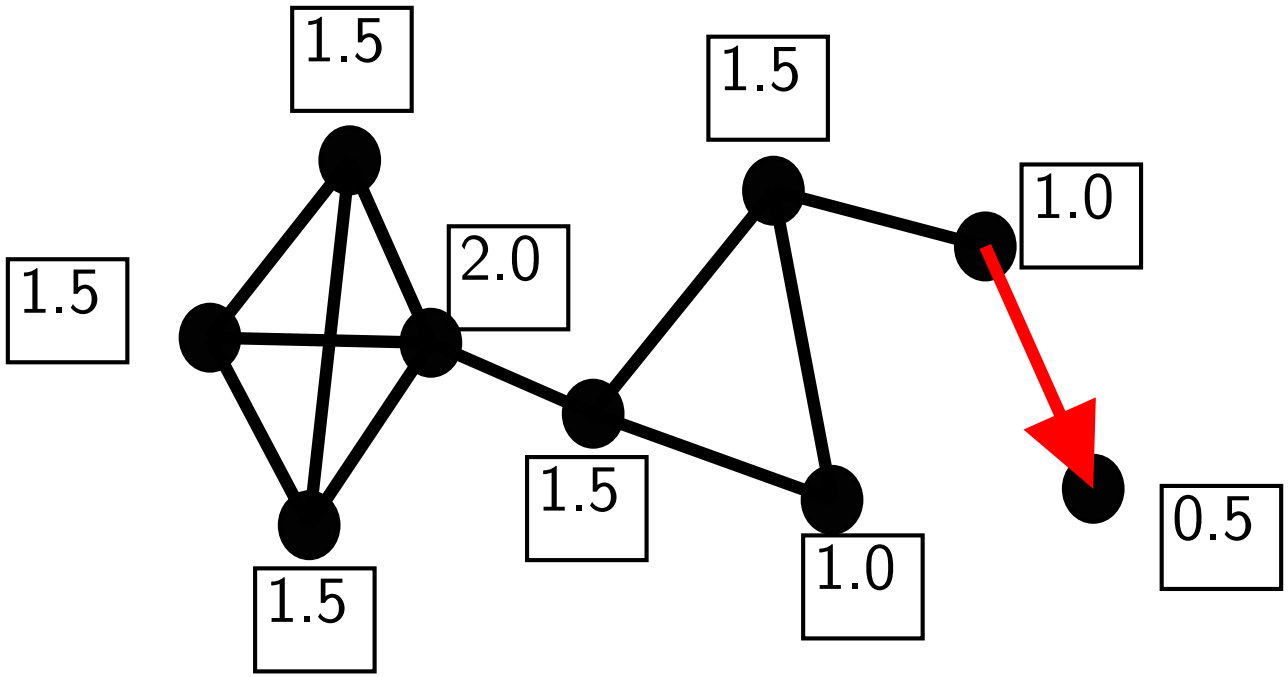
1: Set initial $f^{(0)} \in \mathcal{F}$ arbitrarily
2: **for** each iteration $t = 1, ..., T$ **do**
3: $\quad \gamma_t = \frac{2}{t+2}$
4: $\quad \hat{f} = \arg\min_{f \in \mathcal{F}} < f, \nabla Q_G(f^{(t-1)}) >$
5: $\quad f^{(t)} = (1 - \gamma_t) f^{(t-1)} + \gamma_t \hat{f}$
6: **end for**

Frank-Wolfe Algorithm



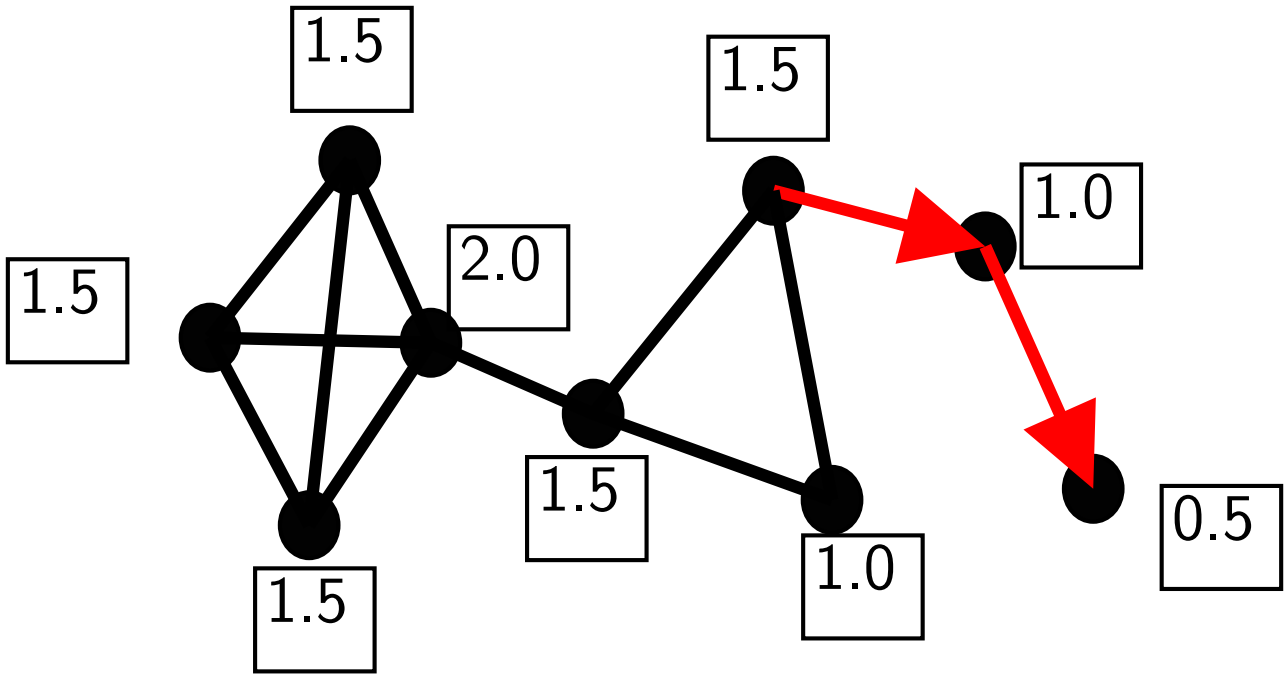
Initialization: Each edge has weight 1, and is distributed evenly to both endpoints. For each node u, $load(u)$ is computed.

Frank-Wolfe Algorithm



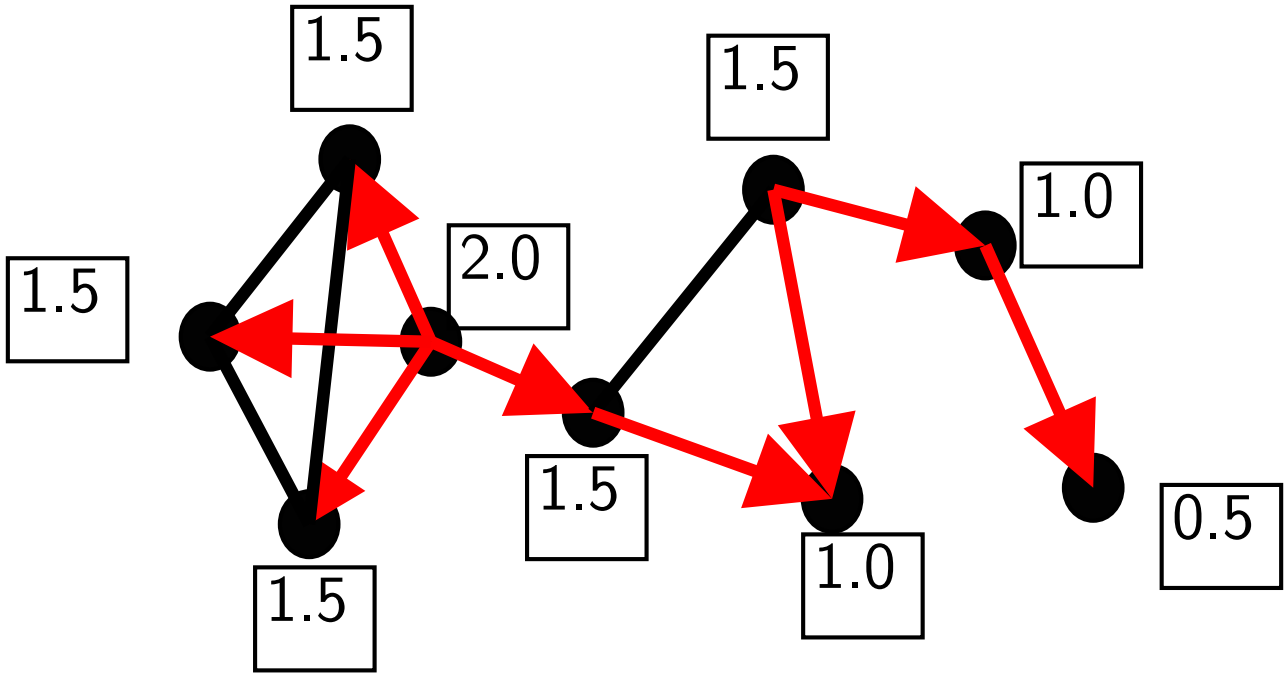
Iteration $t = 1$: Set $\hat{f}_e(u) = 1$ if u has the least $load(u)$ value among endpoints of e; otherwise $\hat{f}_e(u) = 0$.

Frank-Wolfe Algorithm



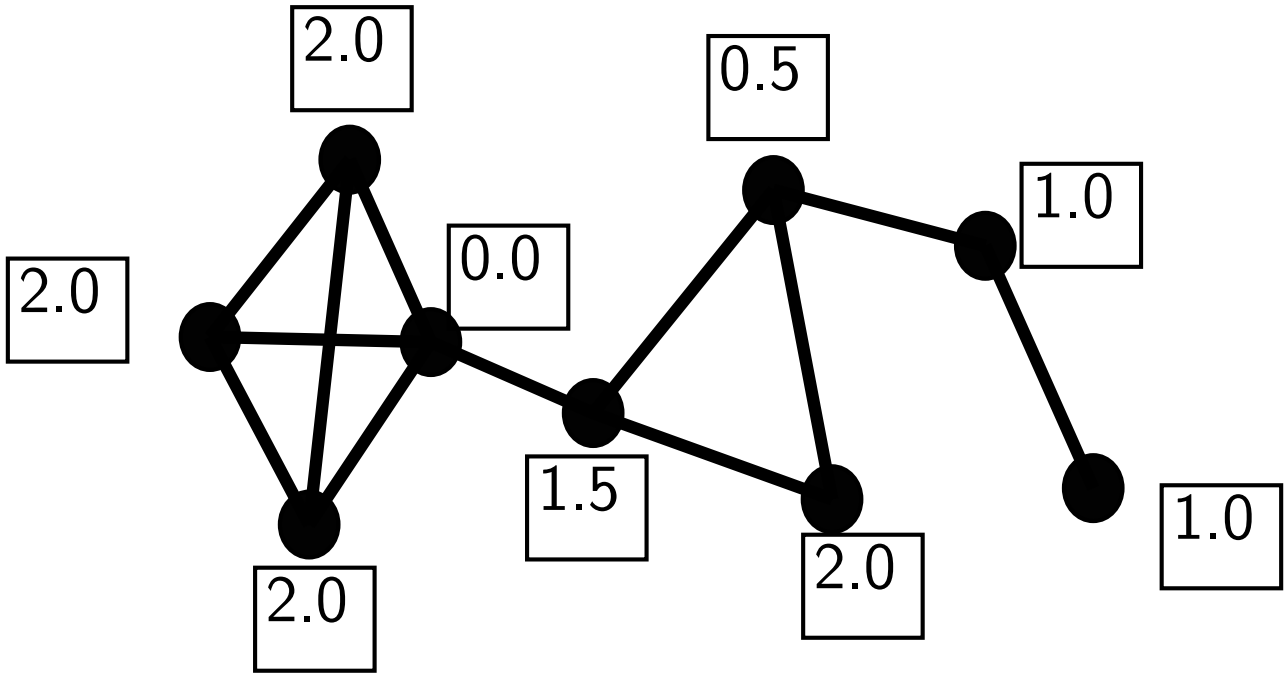
Iteration $t = 1$: Set $\hat{f}_e(u) = 1$ if u has the least $load(u)$ value among endpoints of e; otherwise $\hat{f}_e(u) = 0$.

Frank-Wolfe Algorithm



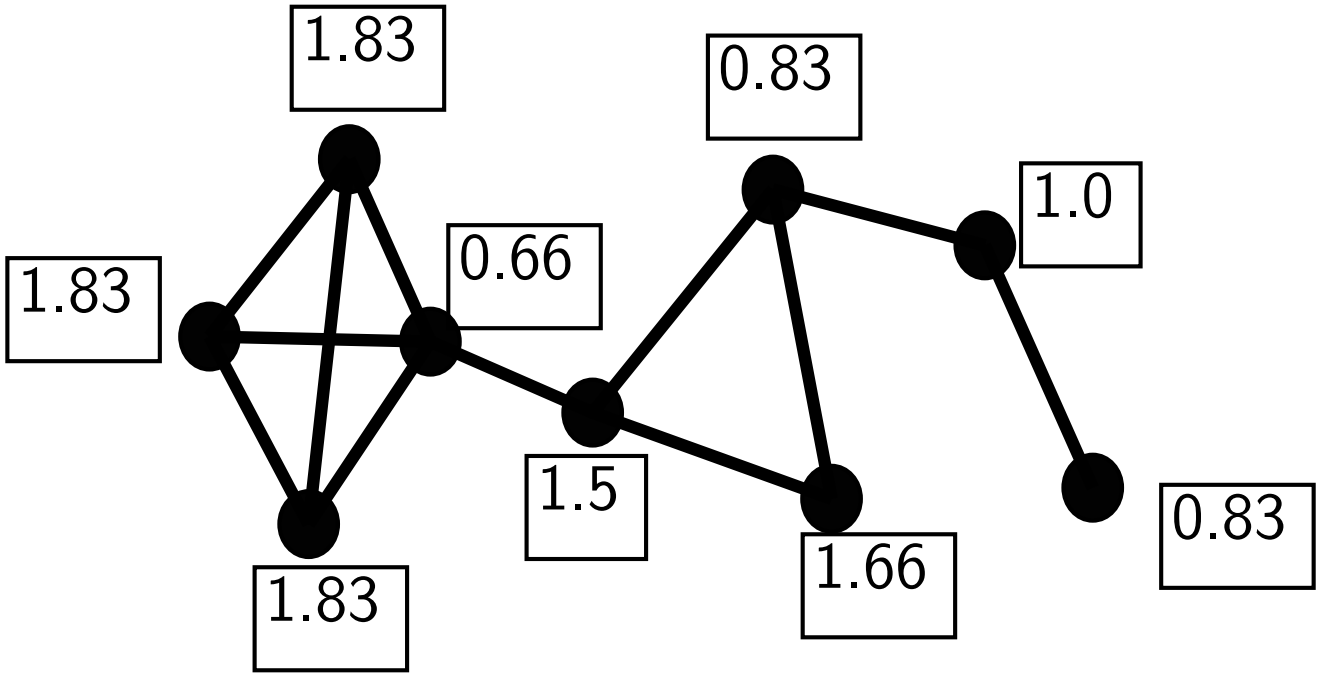
Iteration $t = 1$: Set $\hat{f}_e(u) = 1$ if u has the least $load(u)$ value among endpoints of e; otherwise $\hat{f}_e(u) = 0$.

Frank-Wolfe Algorithm



Iteration $t = 1$: Compute $\hat{load}$ based on $\hat{f}$.

Frank-Wolfe Algorithm



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

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

Loop for T iterations.

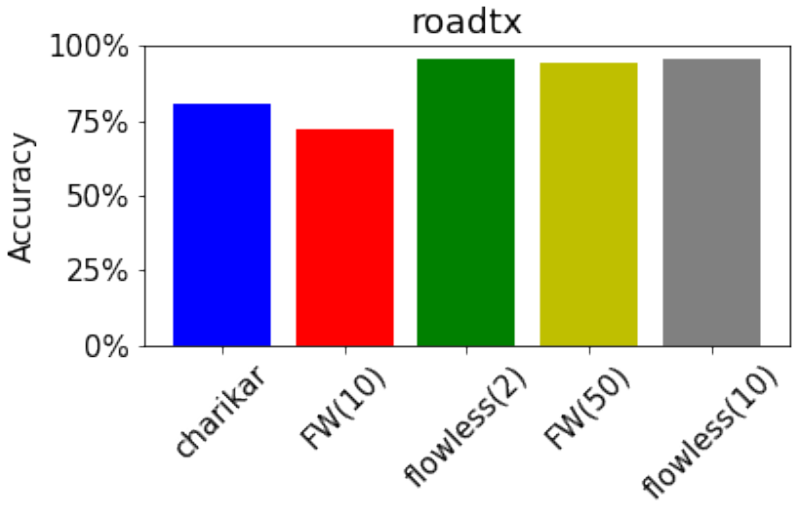
Comparison of methods

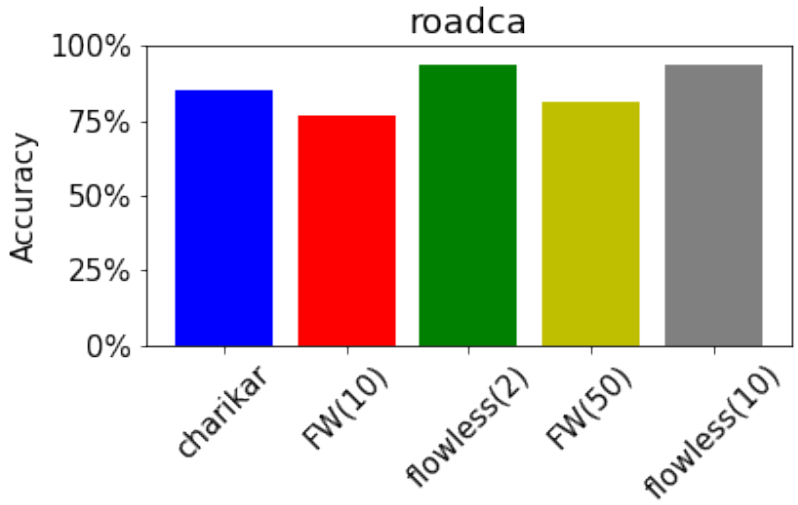
- Exact max-flow algorithm
- Charikar
- Flowless
- Frank-Wolfe based algorithm

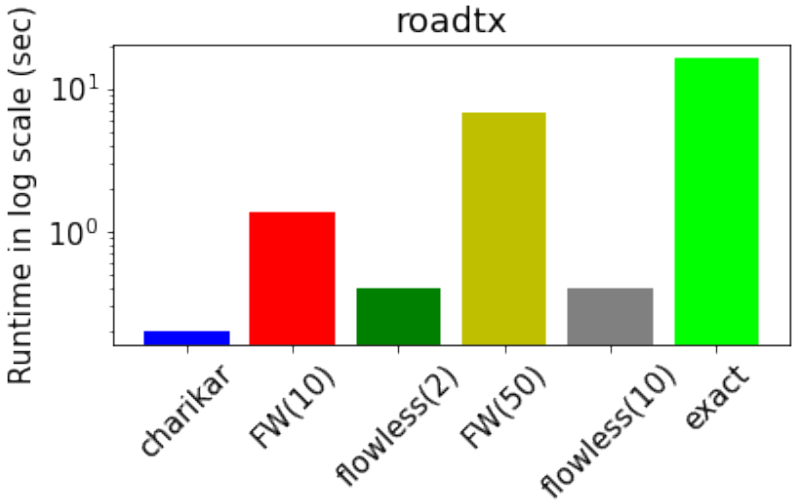
Datasets

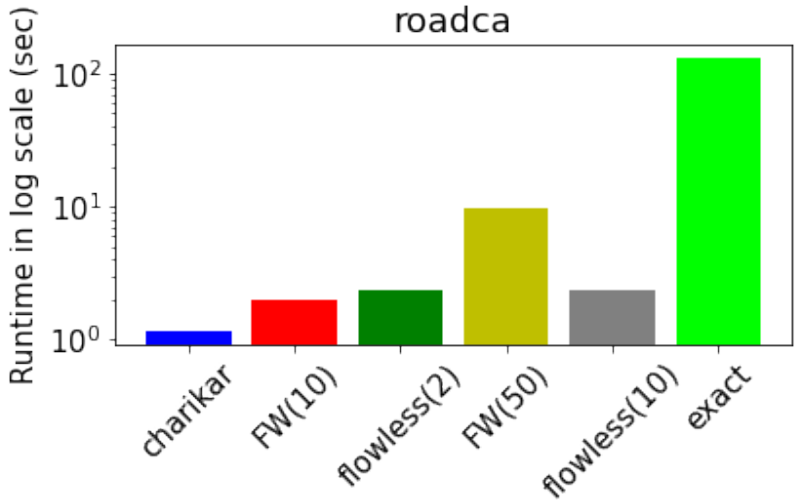
Name	# of nodes	# of edges	type
roadNet-TX	1,379,917	1,921,660	Road
roadNet-CA	1,965,206	2,766,607	Road
roadNet-PA	1,088,092	1,541,898	Road
Enron Email	36,692	183,831	communication
Arxiv HEP-PH	34,546	421,578	citation
Amazon	334,863	925,872	Co-purchasing
web-BerkStan	685,230	7,600,595	Web
LiveJournal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

Results: Accuracy and Run time

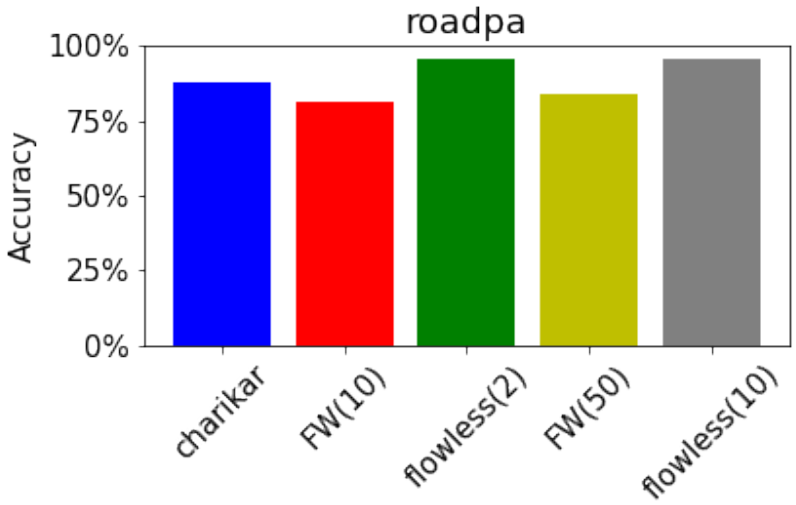


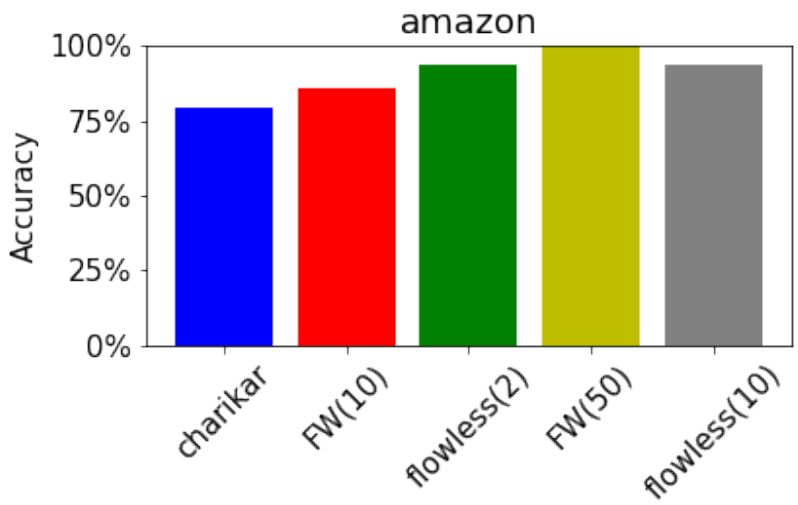


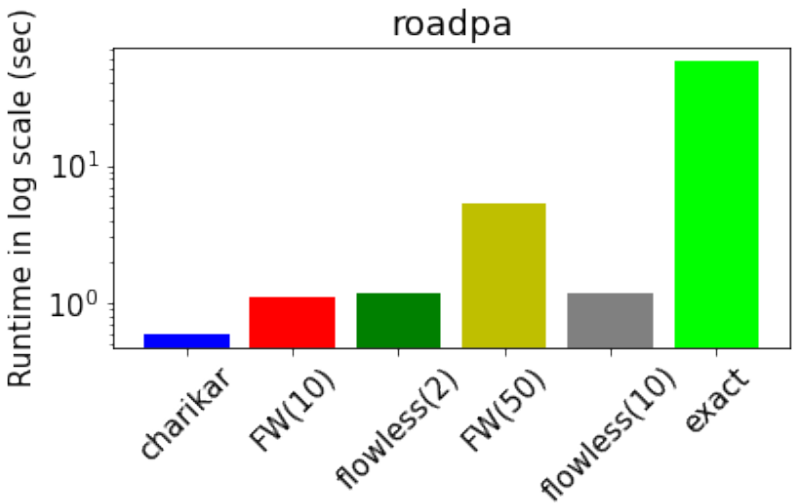


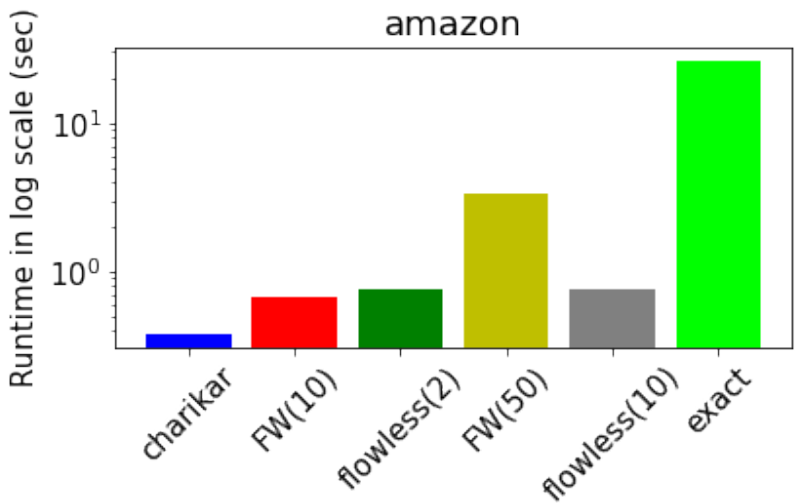


Results: Accuracy and Run time









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 $\ell : E \to \{1, \ldots, L\} = [L]$ 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.

DSP with negative weights – Risk Averse DSD

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., $w^-(e) = \sigma_e^2$.

Hardness and Greedy

Theorem

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

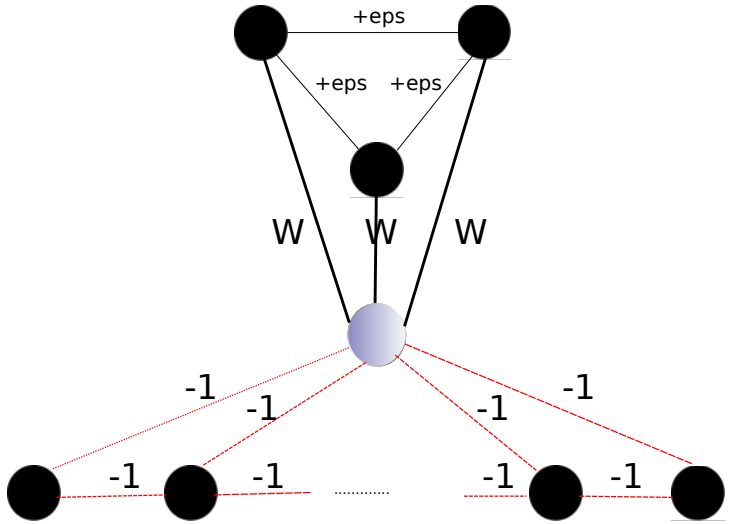
Reduction from MAX-CUT.

Question:: How does Charikar's greedy algorithm perform?

Theorem

Let $G(V, E, w)$, $w : E \to \mathbb{R}$ be an undirected weighted graph with possibly negative weights. If the negative degree $deg^-(u)$ of any node u is upper bounded by Δ, then our Algorithm outputs a set whose density is at least $\frac{\rho^*}{2} - \frac{\Delta}{2}$.

Bad instance



Let $W = \frac{n-4}{3}$. Then, $3W - n < -3$. The degrees of the $n+4$ nodes are as follows:

$$\underbrace{3W - n}_{\text{one node}} < \underbrace{-3}_{n-2 \text{ nodes}} < \underbrace{-2}_{\text{two nodes}} < 0 < \underbrace{2\epsilon + W}_{\text{three nodes}}.$$

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

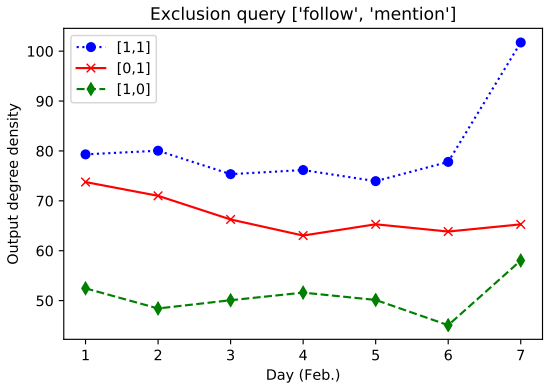
1 set $G_n \leftarrow G$
2 for $k \leftarrow n$ downto 1
2.1 let v be the degree vertex in G_k with smallest score $Cdeg^+(v) - deg^-(v)$
2.2 $G_{k-1} \leftarrow G_k \setminus \{v\}$
3 output the densest subgraph among $G_n, G_{n-1}, \ldots, G_1$

Datasets

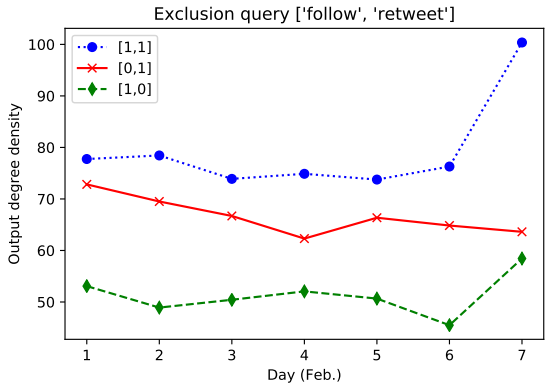
Name	n	m
■ Biogrid	5 640	59 748
■ Collins	1 622	9 074
■ Gavin	1 855	7 669
■ Krogan core	2 708	7 123
■ Krogan extended	3 672	14 317
⊙ TMDB	160 784	883 842
⊙ Twitter (Feb. 1)	621 617	(902 834, 387 597, 222 253, 30 018, 63 062)
⊙ Twitter (Feb. 2)	706 104	(1 002 265, 388 669, 218 901, 29 621, 64 282)
⊙ Twitter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

Experimental findings – Exclusion queries on Twitter

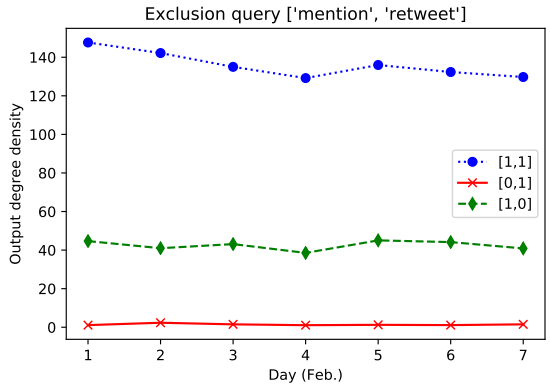
We set $C = 1, W = -\infty$:



(α)



(β)



(γ)

Degree density for three exclusion queries per each pair of interaction types over the period of the first week of February 2018. (α) Follow and mention. (β) Follow and retweet. (γ) Mention and retweet.

Experimental findings - Ranging W, C

C	W	$\lvert S^* \rvert$	$\rho_{\text{retweet}}(S^*)$	$\rho_{\text{reply}}(S^*)$
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

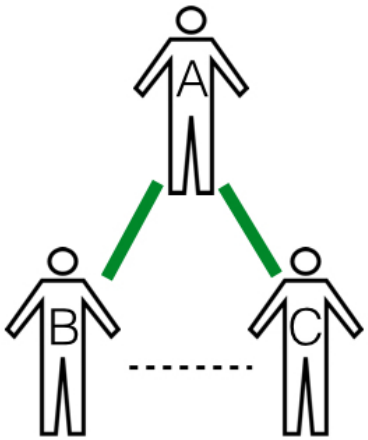
- The k-clique Densest Subgraph Problem (WWW 2015)

  T

- Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling (KDD 2015)

  Mitzenmacher-Pachocki-Peng-**T**-Xu

k-clique densest subgraph problem



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**

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

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

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

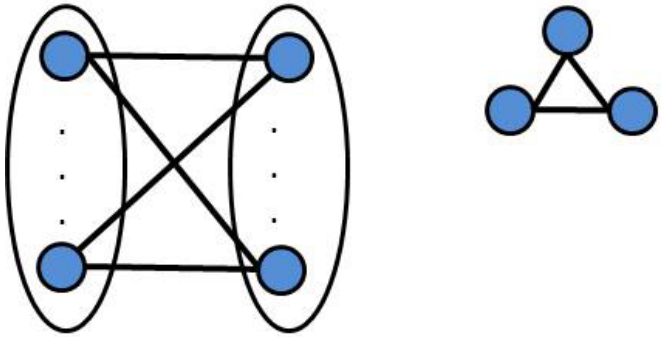
Triangle densest subgraph problem

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.*

Furthermore,

Theorem

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

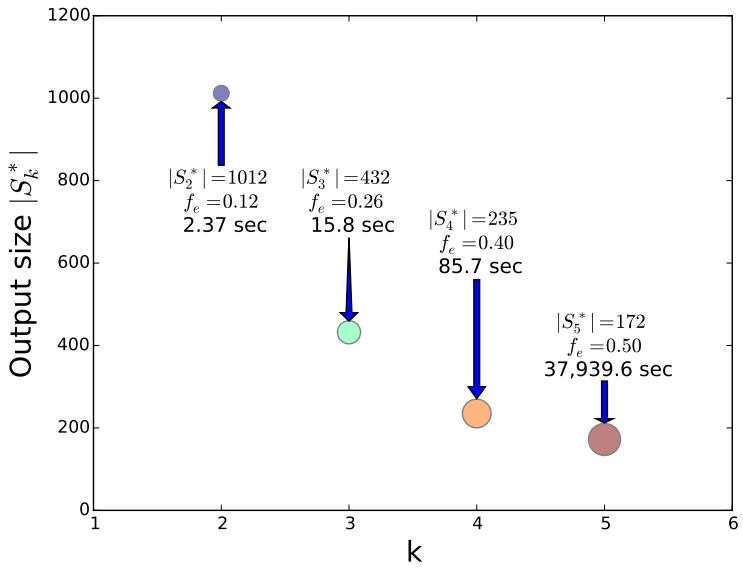
Computation involves

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

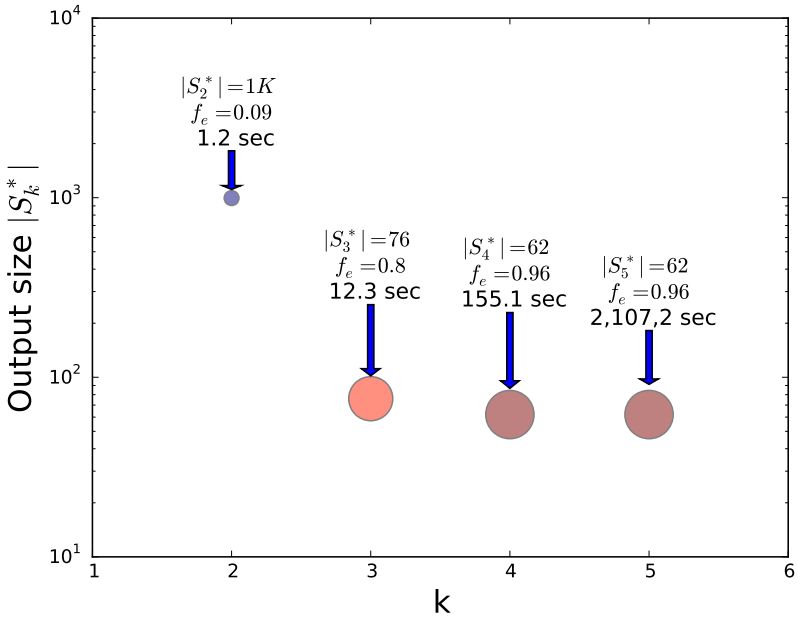
# nodes	75 877
# edges	405 739



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

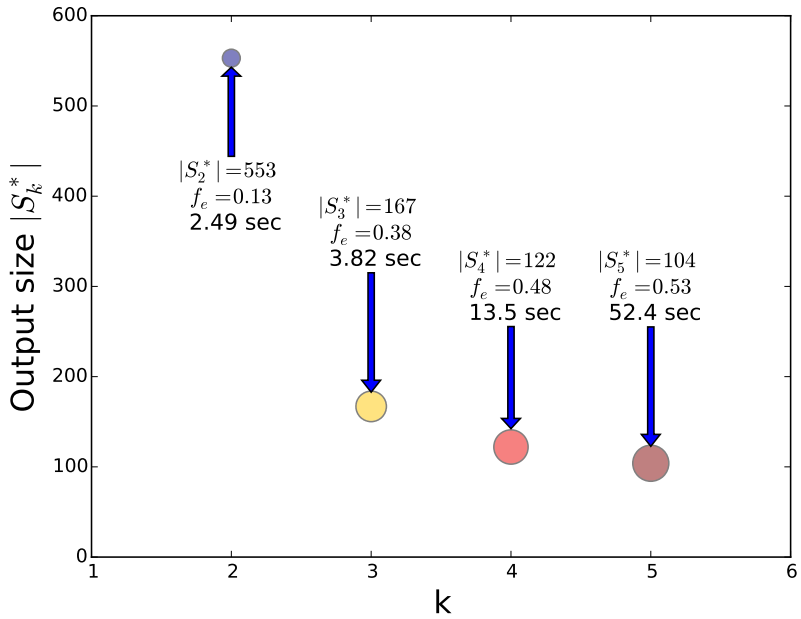
CA-Astro and Email networks

# 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

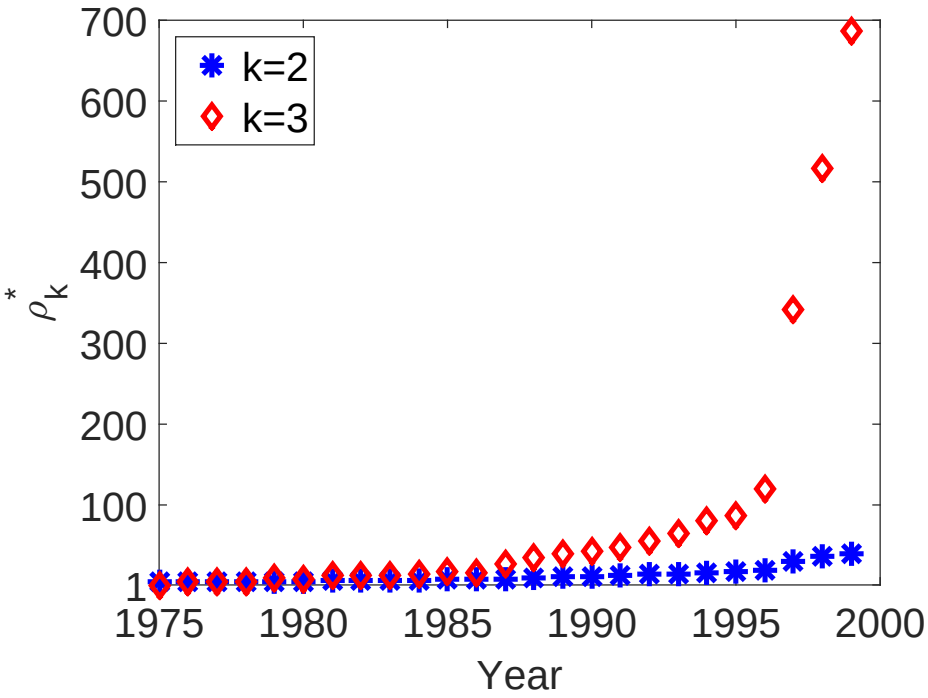
# nodes	234 352
# edges	383 111

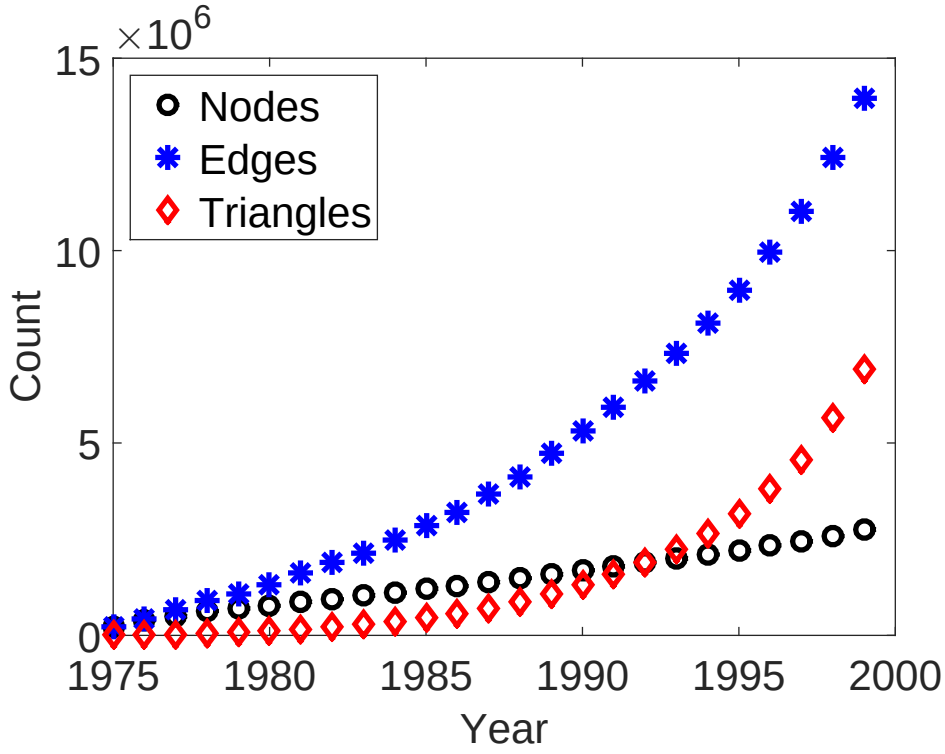


k	c_k	T_k (sec)
3-clique	0.4M	0.4
4-clique	1M	0.9
5-clique	2.6M	1.9

Time evolving networks

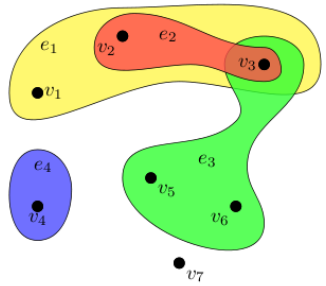
Patents citation network that spans 37 years, specifically from January 1, 1963 to December 30, 1999.





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

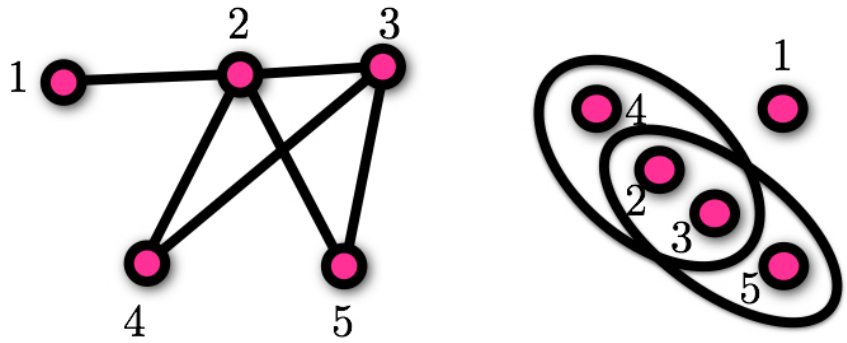
Let $\mathcal{H}(V, E_{\mathcal{H}})$ be a hypergraph, $\epsilon > 0$.

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

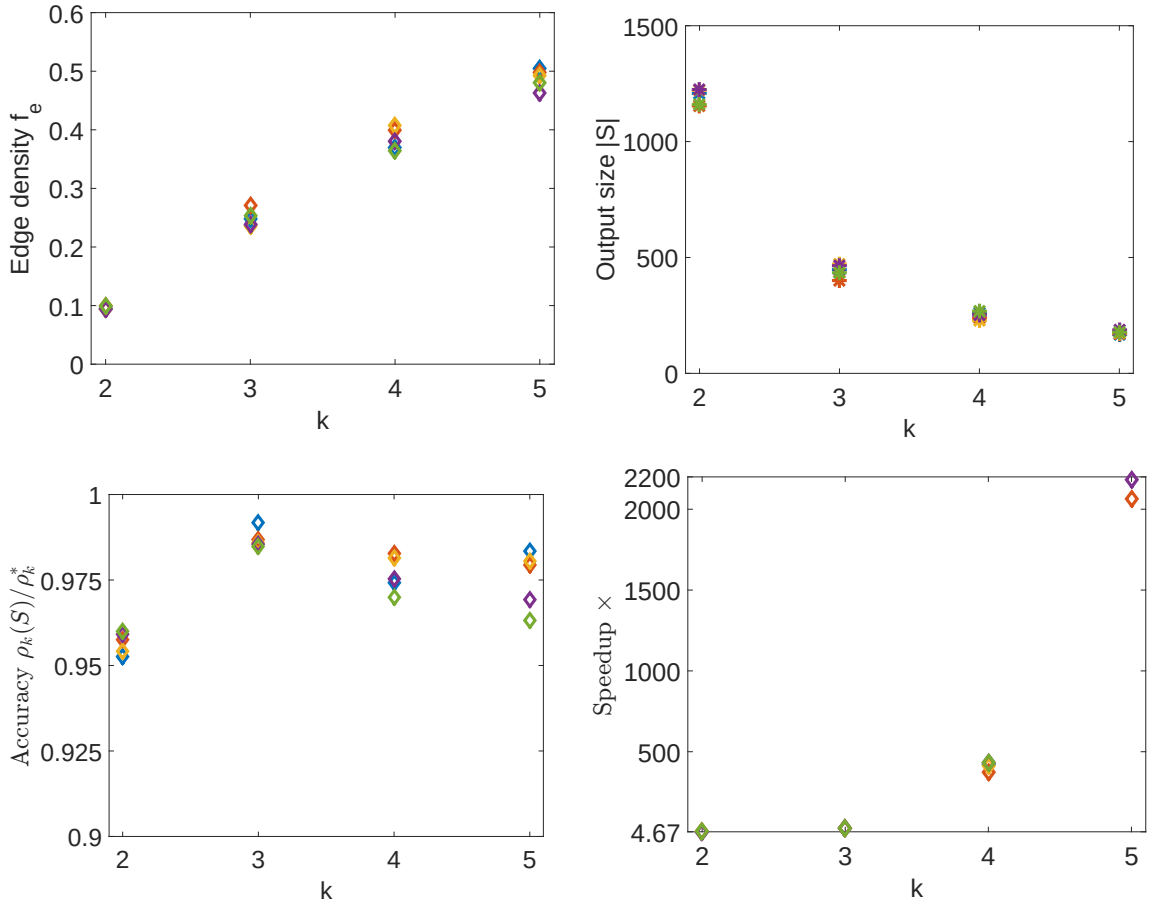
Expected space reduction is $O(\frac{1}{p})$.

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

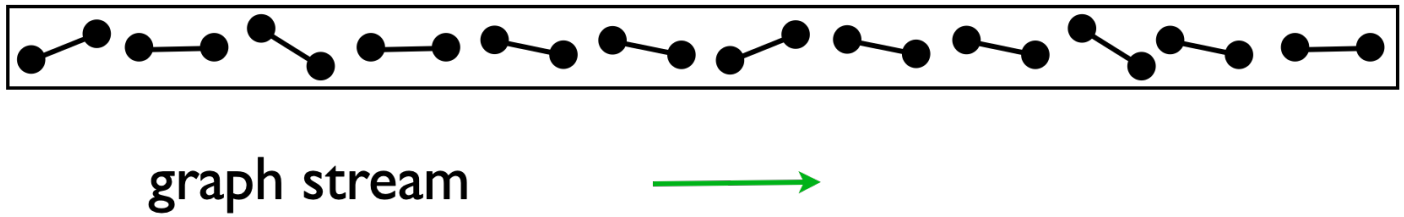


KClist++ a recent algorithm that combines Frank-Wolfe-like algorithm with fast clique enumeration [Sun et al., 2020].

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 $A(S) \leftarrow \{i \in S | deg_S(i) \leq 2(1+\epsilon)\rho(S)\}$
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.

Streaming DSP – State-of-the-art

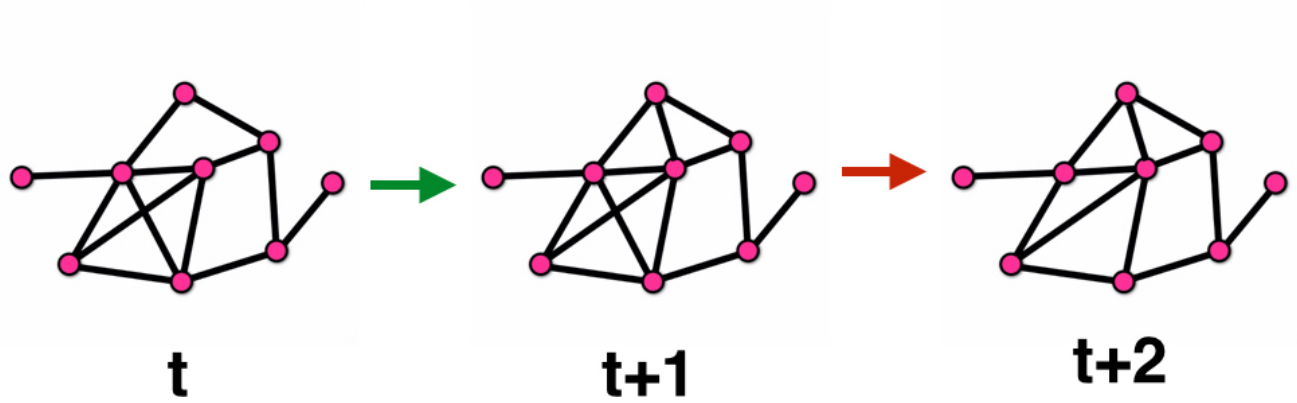
Basic idea: Uniform sampling works, i.e., independently sample each edge in G with probability $\frac{cn \log n}{\epsilon^2 m}$ [Mitzenmacher et al., 2015, Esfandiari et al., 2015, McGregor et al., 2015]

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 γ-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 γ-approximate value of $\rho^*(G) = d^*$

Fully dynamic $(4+\epsilon)$-approximation algorithm $\tilde{O}(n)$ space [Bhattacharya et al., 2015]

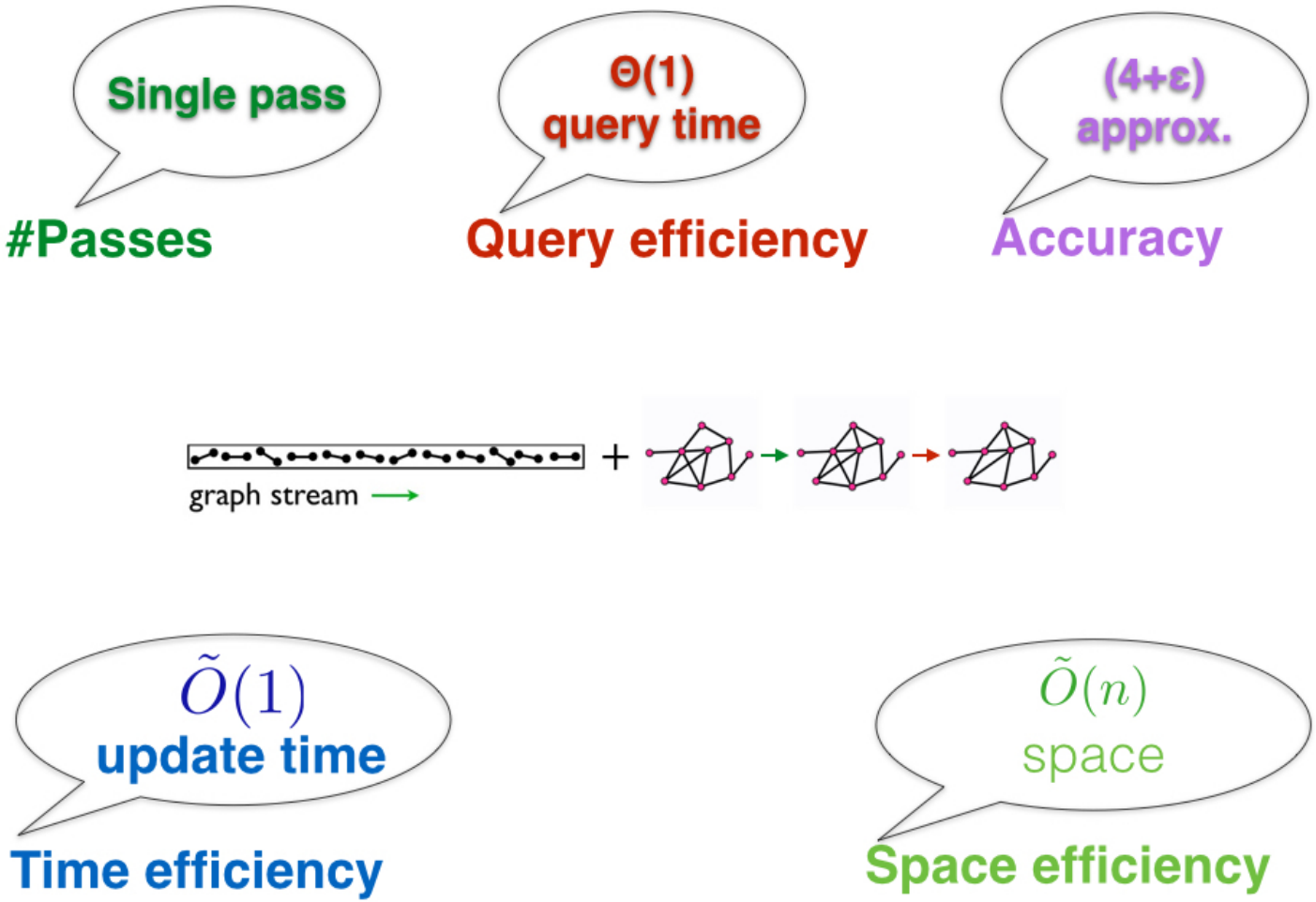
Theorem

- *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 $\textsc{Output}^{(t)}$ at each $t \in [T]$ such that for all $t \in [T]$ whp*

$$\textsc{Opt}^{(t)}/(4+\Theta(\epsilon)) \leq \textsc{Output}^{(t)} \leq \textsc{Opt}^{(t)}.$$

*Also, the total amount of computation performed while processing the first T updates in the dynamic stream is $O(T \ \mathit{poly}\log n)$. (**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 perpective of the DSP!

Sawlani-Wang approach

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

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

Therefore, the following is a local optimality condition :

$$\begin{aligned}
f_e(u) + f_e(v) &= 1 \\
load_u &\leq load_v \\
\forall e = uv \in E, f_e(u) &> 0
\end{aligned}$$

Sawlani-Wang approach

S-W introduce the following notion: η- local optimality condition :

$$\begin{aligned} f_e(u) + f_e(v) &= 1 \\ load_u &\leq load_v + \eta \\ \forall e = uv \in E, f_e(u) &> 0 \end{aligned}$$

The following result:

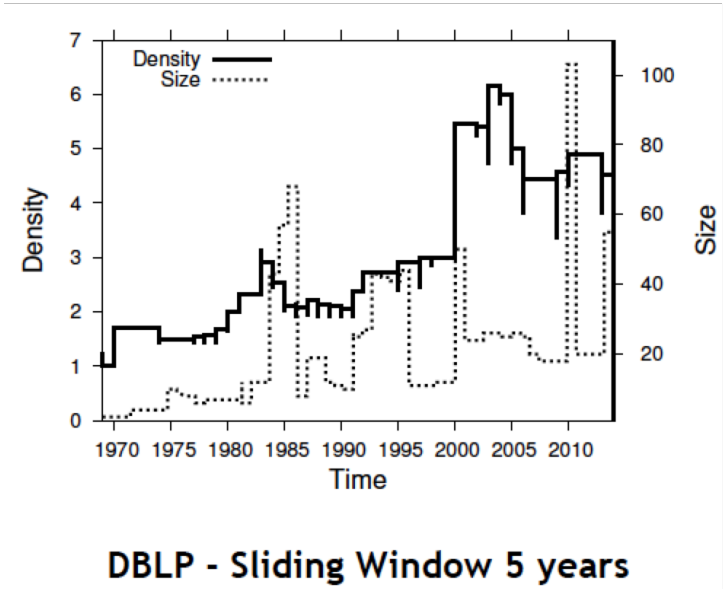
$$\eta - \text{local approx optimality} \Rightarrow \left(1 + \sqrt{\frac{\eta \log n}{OPT}}\right) - \text{global approx}$$

Key idea

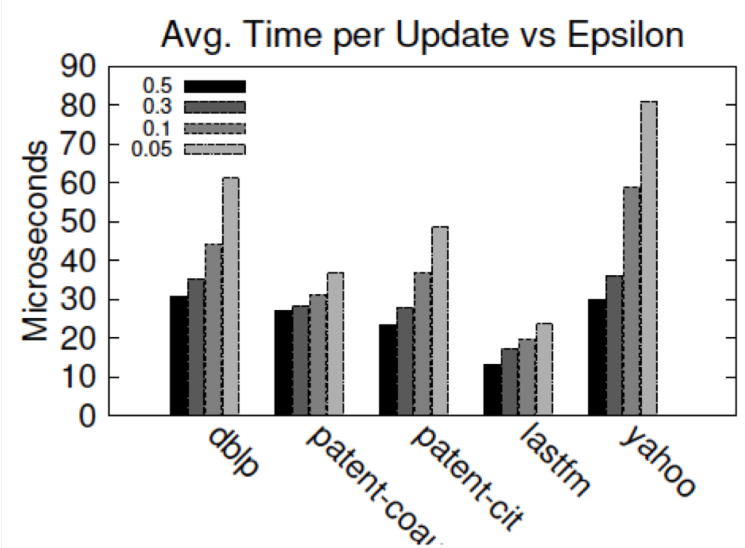
- orient edges so that maximum in-degree is minimized
- "flip" edges till local η-optimality.

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 $(2+\epsilon)$ approx in $O(\log^4(n)/\epsilon^4)$ update time, and linear space.



DBLP - Sliding Window 5 years



Scales much better with Epsilon than worst case.

Source: https://www.epasto.org/papers/www2015-presentation.pdf

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

Algorithm	Time complexity	Technique
[Charikar, 2000]	$\Omega(n^6)$	LP
[Khuller and Saha, 2009]	$O(n^2 t_{max-flow})$	Max flow
[Ma et al., 2020]	$O(k t_{max-flow})$	preprocessing+ Max Flow

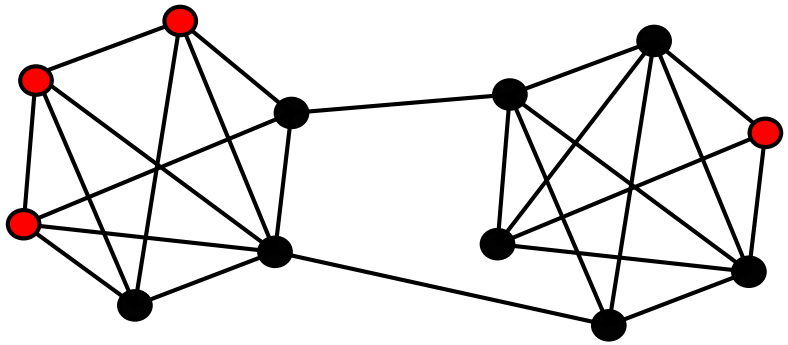
Ma et al. examine k values of possible ratios $\frac{|S|}{|T|}$, and $k \ll n^2$ on real data.

Directed graphs – Approximation Algorithms

Algorithm	Time complexity	Approx.
[Kannan and Vinay, 1999]	$\Omega(sn^3)$	$O(\log n)$
[Khuller and Saha, 2009]	$O(n+m)$	> 2 (paper claimed 2)
Corrected-KS (by [Ma et al., 2020])	$O(n^2(n+m))$	2
[Ma et al., 2020]	$O(\sqrt{m}(n+m))$	2

- 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:

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

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

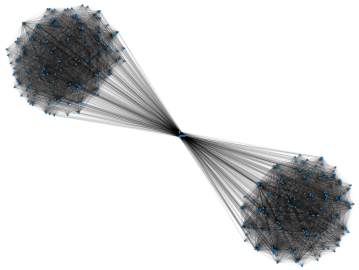
$$f_i = \begin{cases} \frac{1}{\sqrt{n}} & \text{if node } i \text{ is red} \\ -\frac{1}{\sqrt{n}} & \text{if node } i \text{ is blue,} \end{cases}$$

- Fair DSP= DSP+ CONSTRAINT $x^T f = 0$
- Key idea: spectral relaxation:

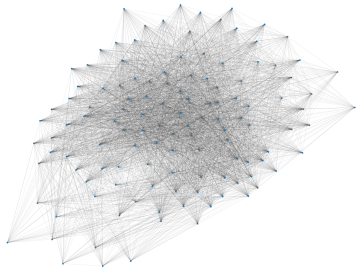
$$\max_{x \in \{0,1\}^n} \frac{2x^T(I - ff^T)A(I - ff^T)x}{x^T x}.$$

- **Solution:** Fair eigenvector

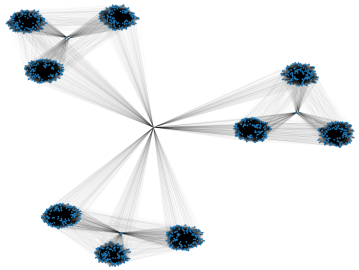
Robustness to vertex/edge failure



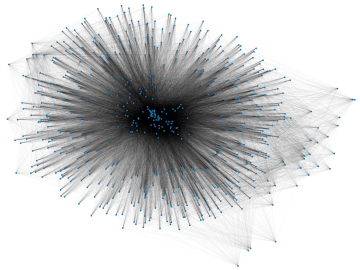
web-BerkStan



web-Google



web-BerkStan



web-Google

- 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 $\rho_G(S) := \dfrac{e(S)}{|S|}$ 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, α-DSLO) ([Balalau et al., 2015])

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

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

subject to

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

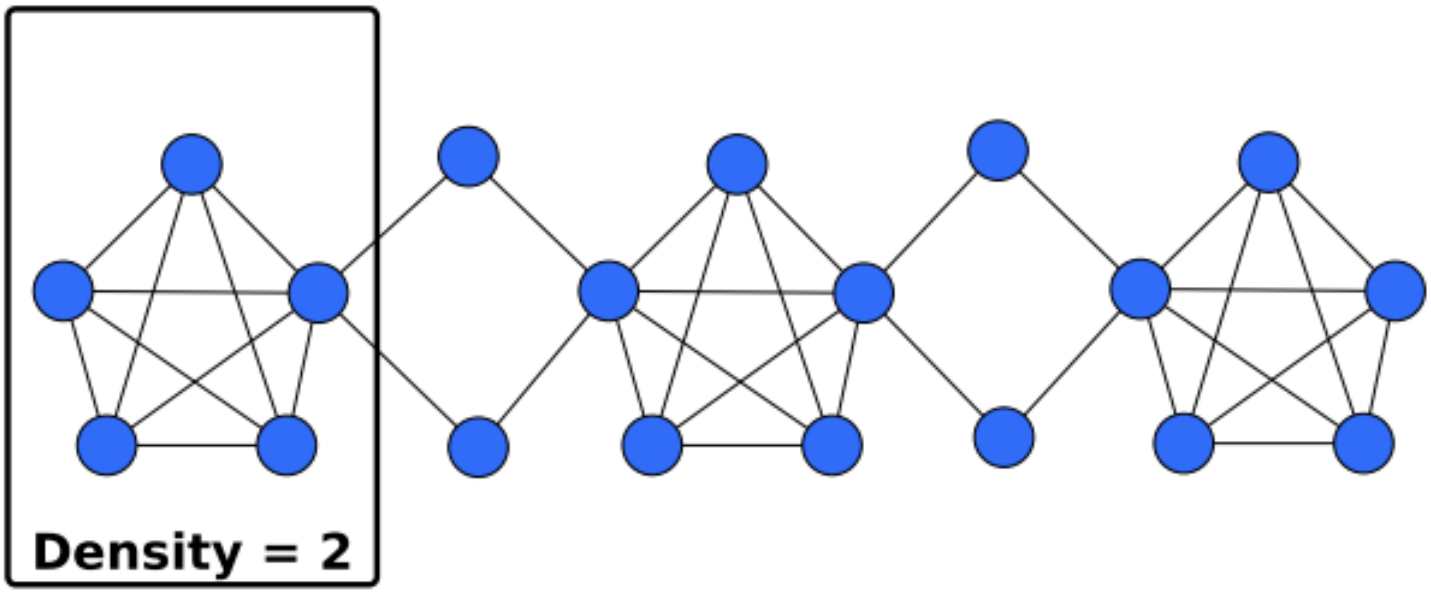
Theorem: The (k, α-DSLO) problem is NP-hard.

MinAndRemove [Balalau et al., 2015]

Example taken from: Balalau et al. slides

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

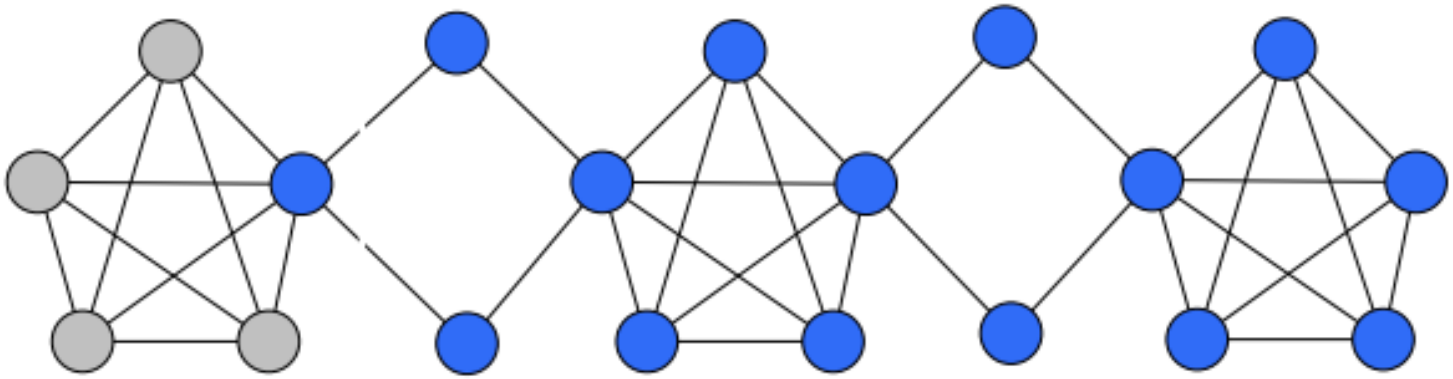
- Find a densest subgraph
- Make it minimal



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

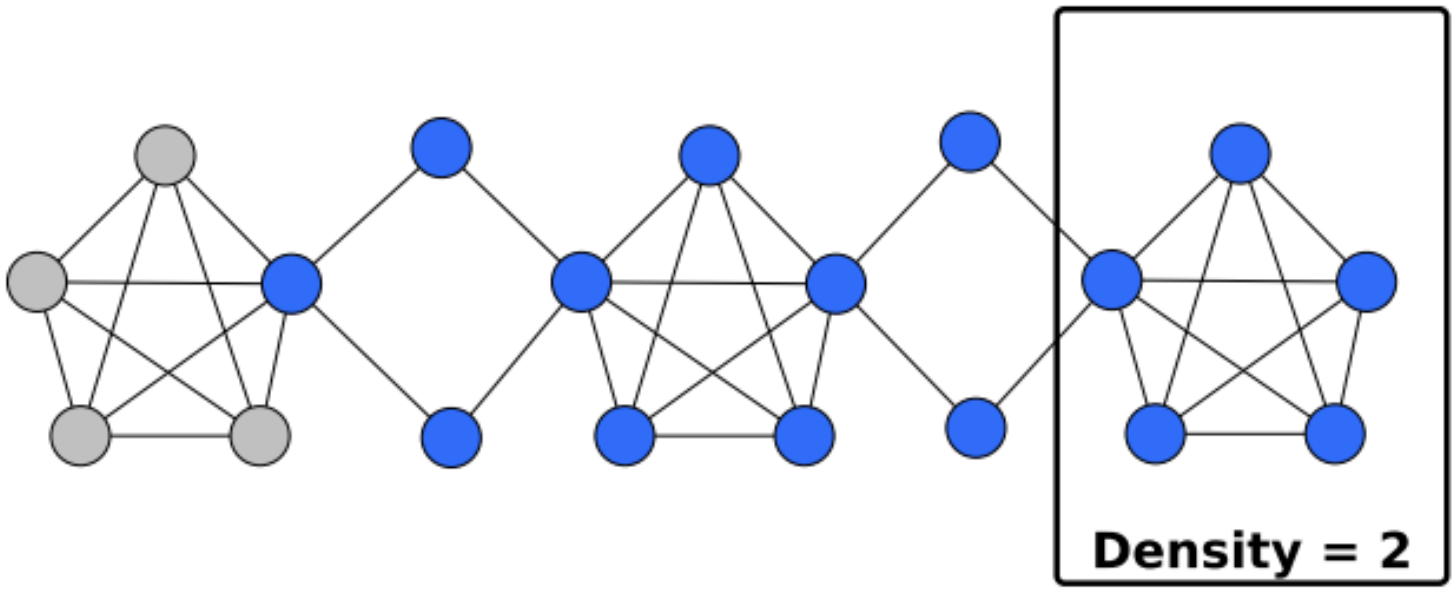


MinAndRemove [Balalau et al., 2015]

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



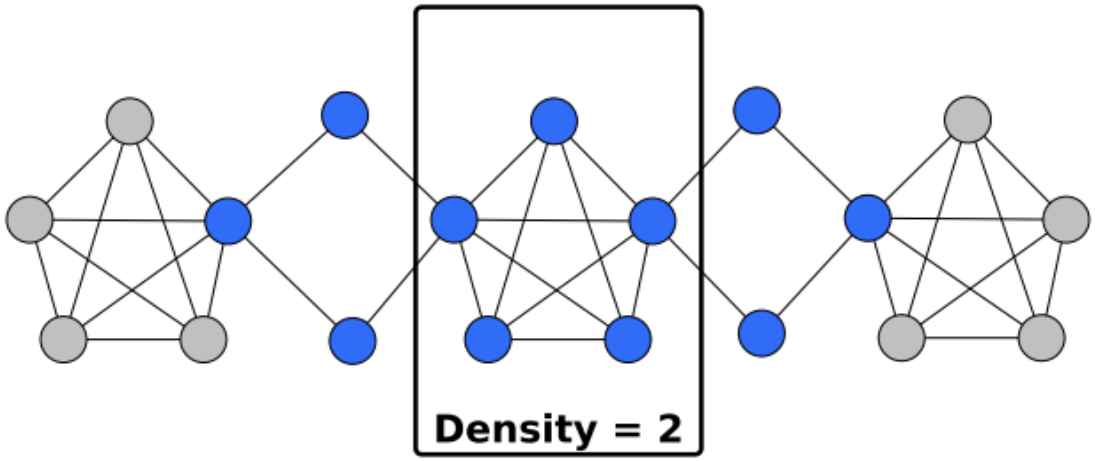
MinAndRemove [Balalau et al., 2015]

Example taken from: Balalau et al. slides

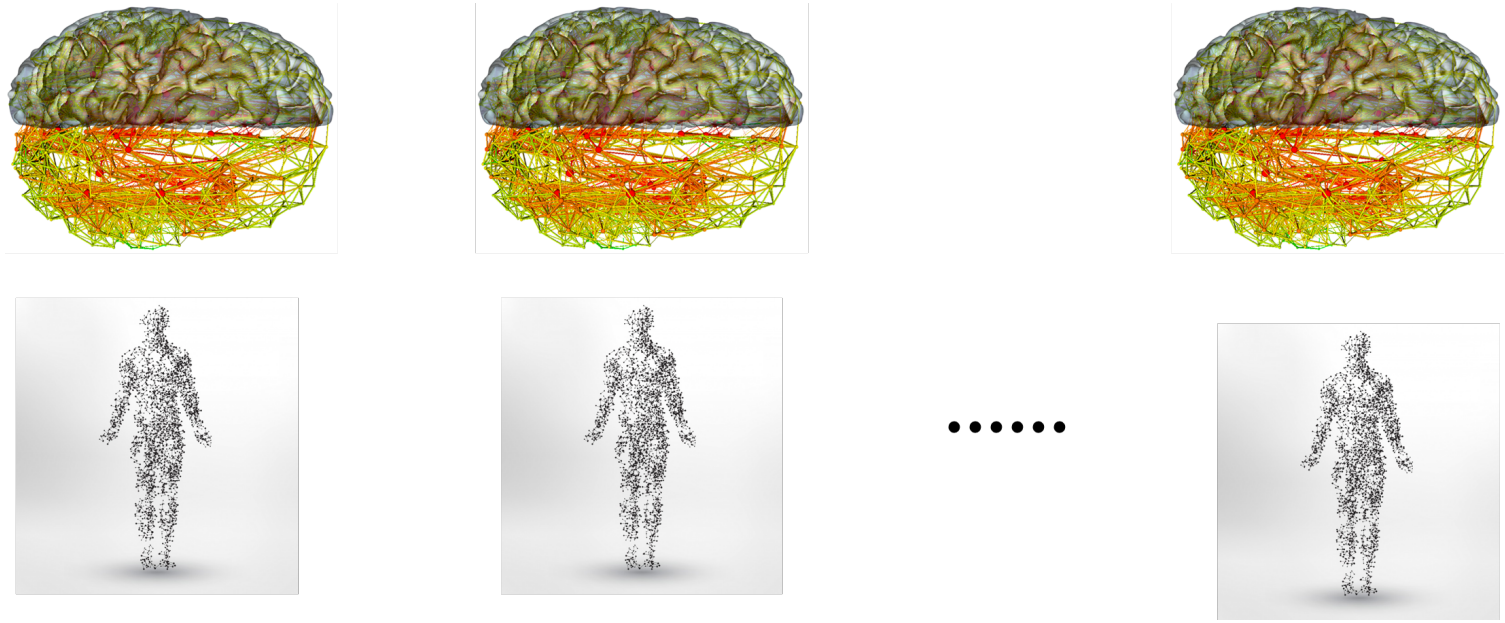
Find $k = 3$ subgraphs that have an overlap of at most $\alpha = 0.25$.

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

No guarantees in the general case.



Common dense subgraph



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

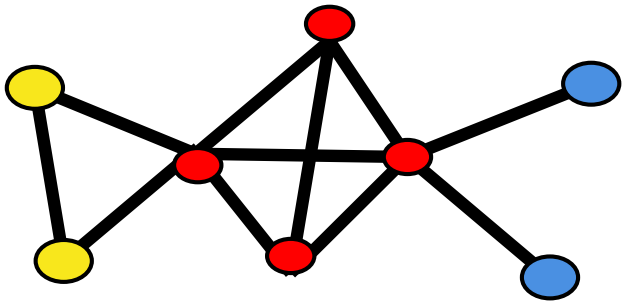
Common dense subgraphs - Results

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

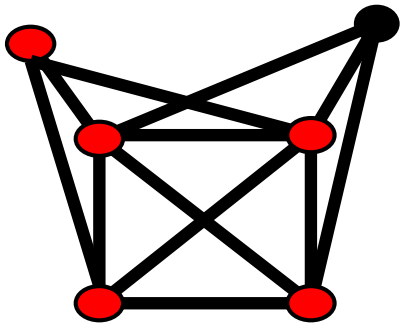
□ [Charikar et al., 2018] Common-DSP-MA (min of average degree) cannot be approximated to within a factor of $2^{\log^{1-\epsilon} n}$ unless **NP**$\subseteq$**DTIME**$(n^{poly \log n})$, via reduction from *MinRep*.

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

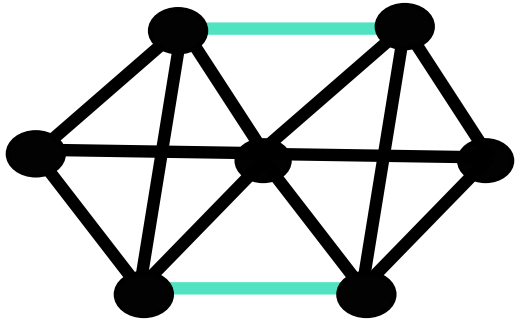
Problem variants



1, 2 and 3 cores



2 plex



3 and 4-truss edges

- 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 $\max_{S \in V} e[S] - \alpha\binom{|S|}{2}$, 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.

More problem variants

- 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

  Flowless is a $1 + O(T^{-1/2})$*-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

Aris Gionis

Atsushi Miyauchi

Questions?

references I

Alon, N., Krivelevich, M., and Sudakov, B. (1998).

Finding a large hidden clique in a random graph.

Random Structures and Algorithms, 13(3-4):457–466.

Alvarez-Hamelin, J. I., Dall'Asta, L., Barrat, A., and Vespignani, A. (2005).

Large scale networks fingerprinting and visualization using the k-core decomposition.

In NIPS.

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.

references II

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

Finding dense subgraphs with size bounds.

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

Arora, S., Barak, B., Brunnermeier, M., and Ge, R. (2011).

Computational complexity and information asymmetry in financial products.

Communications of the ACM, 54(5):101–107.

Arora, S., Hazan, E., and Kale, S. (2012).

The multiplicative weights update method: a meta-algorithm and applications.

Theory of Computing, 8(1):121–164.

references III

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

Densest subgraph in streaming and mapreduce.

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

Balalau, O. D., Bonchi, F., Chan, T. H., Gullo, F., and Sozio, M. (2015).

Finding subgraphs with maximum total density and limited overlap.

In International Conference on Web Search and Data Mining (WSDM), pages 379–388.

Batagelj, V. and Zavernik, M. (2003).

An o(m) algorithm for cores decomposition of networks.

CoRR, cs.DS/0310049.

references IV

Beutel, A., Xu, W., Guruswami, V., Palow, C., and Faloutsos, C. (2013).

Copycatch: stopping group attacks by spotting lockstep behavior in social networks.

In Proceedings of the 22nd international conference on World Wide Web, pages 119–130.

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

references V

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.

Bonchi, F., García-Soriano, D., Miyauchi, A., and Tsourakakis, C. E. (2020).

Finding densest k-connected subgraphs.

arXiv preprint arXiv:2007.01533.

Charikar, M. (2000).

Greedy approximation algorithms for finding dense components in a graph.

In APPROX.

references VI

Charikar, M., Naamad, Y., and Wu, J. (2018).

On finding dense common subgraphs.

ArXiv, abs/1802.06361.

Chen, J. and Saad, Y. (2012).

Dense subgraph extraction with application to community detection.

Knowledge and Data Engineering, IEEE Transactions on, 24(7):1216–1230.

Chen, Y., Fang, Y., Cheng, R., Li, Y., Chen, X., and Zhang, J. (2019).

Exploring communities in large profiled graphs.

IEEE Transactions on Knowledge and Data Engineering, 31(8):1624–1629.

Cohen, E., Halperin, E., Kaplan, H., and Zwick, U. (2003).

Reachability and distance queries via 2-hop labels.

SIAM Journal on Computing, 32(5):1338–1355.

references VII

Cohen, J. (2008).

1 trusses: Cohesive subgraphs for social network analysis.

Delling, D., Goldberg, A. V., Pajor, T., and Werneck, R. (2014).

Robust distance queries on massive networks.

In Algorithms-ESA 2014, pages 321–333. Springer.

Devroye, L., György, A., Lugosi, G., Udina, F., et al. (2011).

High-dimensional random geometric graphs and their clique number.

Electronic Journal of Probability, 16:2481–2508.

Epasto, A., Lattanzi, S., and Sozio, M. (2015).

Efficient densest subgraph computation in evolving graphs.

In Proceedings of the 24th International Conference on World Wide Web, pages 300–310. International World Wide Web Conferences Steering Committee.

references VIII

Esfandiari, H., Hajiaghayi, M., and Woodruff, D. P. (2015).
Applications of uniform sampling: Densest subgraph and beyond.
arXiv preprint arXiv:1506.04505.

Fang, Y., Cheng, R., Chen, Y., Luo, S., and Hu, J. (2017).
Effective and efficient attributed community search.
The VLDB Journal, 26(6):803828.

Fang, Y., Yu, K., Cheng, R., Lakshmanan, L., and Lin, X. (2019).
Efficient algorithms for densest subgraph discovery.
Proc. VLDB Endow., 12:1719–1732.

Feige, U., Kortsarz, G., and Peleg, D. (1999).
The dense k-subgraph problem.
Algorithmica, 29:2001.

references IX

Fratkin, E., Naughton, B. T., Brutlag, D. L., and Batzoglou, S. (2006).

Motifcut: regulatory motifs finding with maximum density subgraphs.

Bioinformatics, 22(14):e150–e157.

Gamarnik, D. and Zadik, I. (2019).

The landscape of the planted clique problem: Dense subgraphs and the overlap gap property.

arXiv preprint arXiv:1904.07174.

Gionis, A., Junqueira, F., Leroy, V., Serafini, M., and Weber, I. (2013).

Piggybacking on social networks.

Proceedings of the VLDB Endowment, 6(6):409–420.

references X

Goldberg, A. V. (1984).

Finding a maximum density subgraph.

Technical report, University of California at Berkeley.

Hastad, J. (1996).

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

In Acta Mathematica, pages 627–636.

Hooi, B., Song, H. A., Beutel, A., Shah, N., Shin, K., and Faloutsos, C. (2016).

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.

references XI

Huang, X., Cheng, H., Qin, L., Tian, W., and Yu, J. X. (2014).

Querying k-truss community in large and dynamic graphs.

In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14, page 13111322, New York, NY, USA. Association for Computing Machinery.

Iasemidis, L. D., Shiau, D.-S., Chaovalitwongse, W. A., Sackellares, J. C., Pardalos, P. M., Principe, J. C., Carney, P. R., Prasad, A., Veeramani, B., and Tsakalis, K. (2003).

Adaptive epileptic seizure prediction system.

IEEE Transactions on Biomedical Engineering, 50(5).

Jethava, V., Martinsson, A., Bhattacharyya, C., and Dubhashi, D. (2013).

Lovász ϑ function, svms and finding dense subgraphs.

The Journal of Machine Learning Research, 14(1):3495–3536.

references XII

Kang, U., Tsourakakis, C. E., and Faloutsos, C. (2009).

Pegasus: A peta-scale graph mining system implementation and observations.

In Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, pages 229–238. IEEE.

Kannan, R. and Vinay, V. (1999).

Analyzing the structure of large graphs.

Rheinische Friedrich-Wilhelms-Universität Bonn.

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

Speeding up algorithms on compressed web graphs.

Internet Mathematics, 6(3):373–398.

references XIII

Kawase, Y. and Miyauchi, A. (2017).

The densest subgraph problem with a convex/concave size function.

Algorithmica, 80.

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

On finding dense subgraphs.

In ICALP.

Konar, A. and Sidiropoulos, N. D. (2020).

Mining large quasi-cliques with quality guarantees from vertex neighborhoods.

Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999).

Trawling the Web for emerging cyber-communities.

Computer Networks, 31(11–16):1481–1493.

references XIV

Kuroki, Y., Miyauchi, A., Honda, J., and Sugiyama, M. (2020).

Online dense subgraph discovery via blurred-graph feedback.

ArXiv, abs/2006.13642.

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.

Ma, C., Fang, Y., Cheng, R., Lakshmanan, L. V., Zhang, W., and Lin, X. (2020).

Efficient algorithms for densest subgraph discovery on large directed graphs.

In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pages 1051–1066.

references XV

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

Densest subgraph in dynamic graph streams.

arXiv preprint arXiv:1506.04417.

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

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

21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Peleg, D. (2000).

Informative labeling schemes for graphs.

In Mathematical Foundations of Computer Science 2000, pages 579–588. Springer.

references XVI

Plotkin, S. A., Shmoys, D. B., and Tardos, É. (1995).

Fast approximation algorithms for fractional packing and covering problems.

Mathematics of Operations Research, 20(2):257–301.

Rozenshtein, P., Anagnostopoulos, A., Gionis, A., and Tatti, N. (2014).

Event detection in activity networks.

In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.

Saha, B., Hoch, A., Khuller, S., Raschid, L., and Zhang, X.-N. (2010).

Dense subgraphs with restrictions and applications to gene annotation graphs.

In Research in Computational Molecular Biology, pages 456–472. Springer.

references XVII

Sarıyüce, A. E. and Pinar, A. (2018).

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.

Sarıyüce, A. E., Seshadhri, C., Pinar, A., and Catalyurek, U. V. (2015).

Finding the hierarchy of dense subgraphs using nucleus decompositions.

In Proceedings of the 24th International Conference on World Wide Web, pages 927–937.

Sawlani, S. and Wang, J. (2020).

Near-optimal fully dynamic densest subgraph.

In Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, pages 181–193.

references XVIII

Semertzidis, K., Pitoura, E., Terzi, E., and Tsaparas, P. (2018).

Finding lasting dense subgraphs.

Data Mining and Knowledge Discovery, pages 1–29.

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.

Thorup, M. (2004).

Compact oracles for reachability and approximate distances in planar digraphs.

Journal of the ACM (JACM), 51(6):993–1024.

references XIX

Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., and Tsiarli, M. (2013).

Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees.

In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 104–112. ACM.

Tsourakakis, C. E., Chen, T., Kakimura, N., and Pachocki, J. (2019).

Novel dense subgraph discovery primitives: Risk aversion and exclusion queries.

In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 378–394. Springer.

Xiao, M., Lin, W., Dai, Y.-S., and Zeng, Y. (2017).

A fast algorithm to compute maximum k-plexes in social network analysis.

In AAAI.

references XX

Zhou, Y., Xu, J., Guo, Z., Xiao, M., and Jin, Y. (2020).

Enumerating maximal k-plexes with worst-case time guarantee.

In AAAI.