Dense Subgraph Discovery (DSD)

Aristides (Aris) Gionis[1]
Charalampos (Babis) E. Tsourakakis[2]

[1]Aalto University, Finland

[2]Harvard University, USA

KDD 2015

Tutorial website

slides and links to relevant papers :

https://densesubgraphdiscovery.wordpress.com/tutorial

can also be found via KDD 2015 website

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 I care?

what does dense mean?

review of main problems, and main algorithms

Outline

- motivating applications
- preliminaries and measures of density
- algorithms for static graphs
- algorithms for dynamic graphs
- problem variants
- conclusions and open problems

Motivating applications

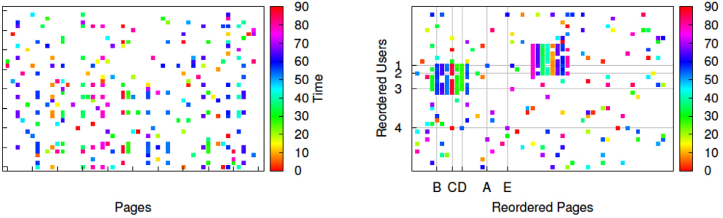
Motivation – correlation mining

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

Motivation – 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]

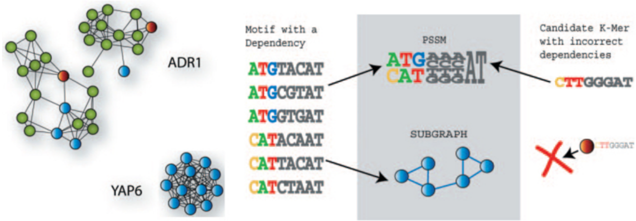
Motivation – e-commerce

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

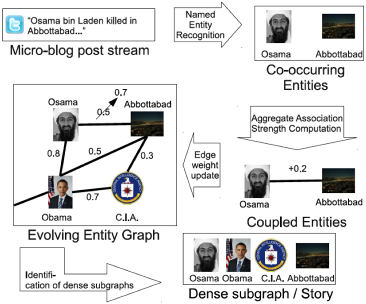
large almost bipartite cliques correspond to sub-markets

Motivation – 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]

Motivation – mining twitter data



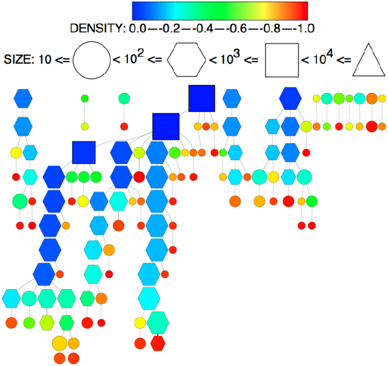
real-time story identification [Angel et al., 2012]

- mining of twitter data
- vertices correspond to entities
- edges correspond to co-occurence of entities
- dense subgraphs capture news stories

Motivation – graph mining

understanding the structure of real-world networks [Sarıyüce et al., 2015]

nucleus decomposition of a graph



(3,4)-nuclei forest for facebook

Motivation – distance queries in graphs

applications :

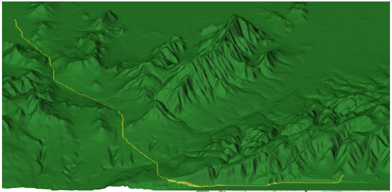
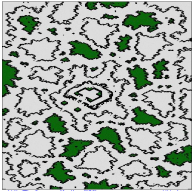
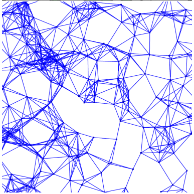
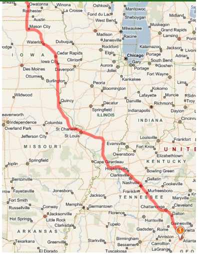
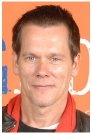
- driving directions
- indoor/terrain navigation
- routing in comm./sensor networks
- moving agents in game maps
- proximity in social/collab. networks

existing solutions :

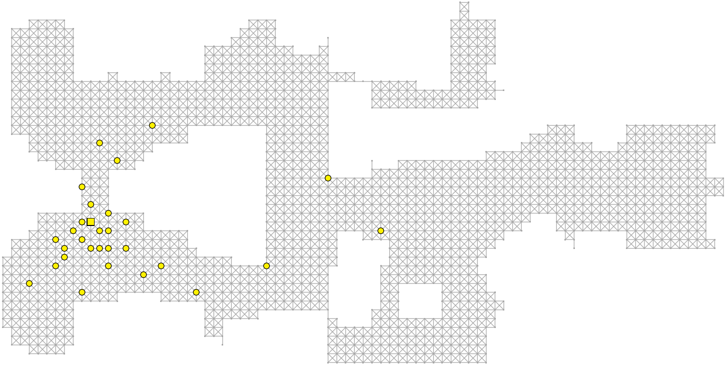
- graph searches are too slow
- fast algorithms are often heuristics
- or tailored to specific graph classes

goals :

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


Motivation – distance queries in graphs

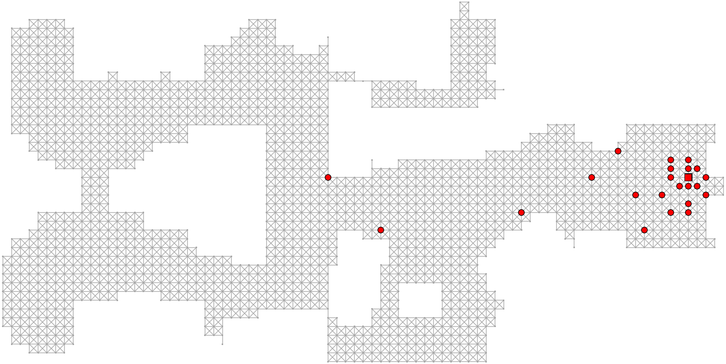


- $L(u) \equiv$ set of pairs $(v, \mathrm{dist}(u, v))$

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

figure from [Delling et al., 2014]

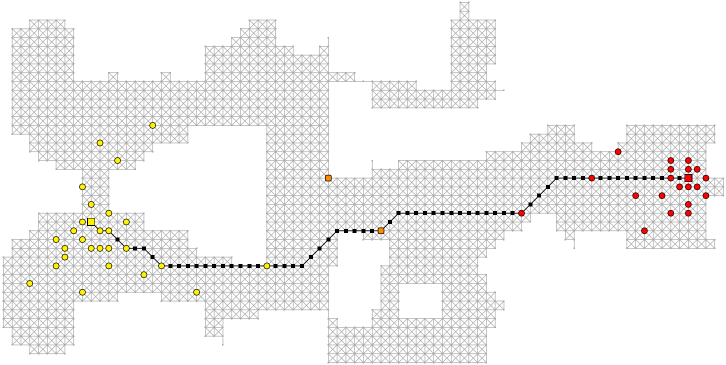
Motivation – distance queries in graphs



- 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]

Motivation – distance queries in graphs



- to answer an s–t query :
  find hub v in $L(s) \cap L(t)$ minimizing $\mathrm{dist}(s, v) + \mathrm{dist}(v, t)$

figure from [Delling et al., 2014]

Motivation – 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 dense-subgraph discovery 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]

Motivation – 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

Motivation – frequent pattern mining

- one of the most well-studied area in data mining
- many efficient algorithms

  Apriori, Eclat, FP-growth, Mafia, ABS, . . .
- main idea: monotonicity

  a subset of a frequent set must be frequent, or

  a superset of an infrequent set must be infrequent
- algorithmically:

  start with small itemsets

  proceed with larger itemset if all subsets are frequent
- enumerate all frequent itemsets

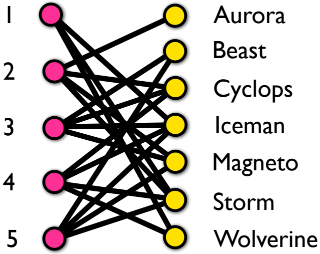
Motivation – frequent itemsets and dense subgraphs

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

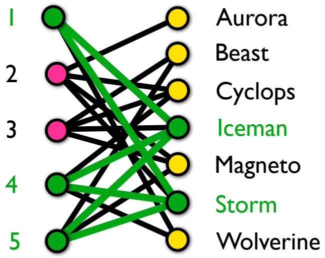
Motivation – frequent itemsets and dense subgraphs

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

⇔

	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

⇔



- transaction data ⇔ binary data ⇔ bipartite graphs
- frequent itemsets ⇔ bi-cliques

Motivation – finding web communities

[Kumar et al., 1999]

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

  e.g., luxury cars and luxury-car aficionados

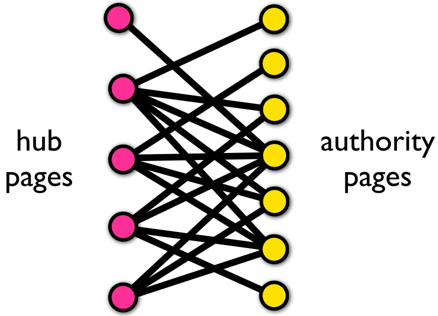
- key observations:

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

Motivation – finding web communities

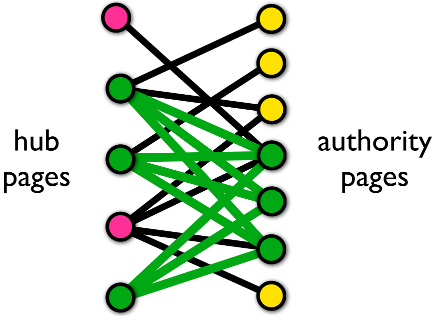
a web community



[Kumar et al., 1999]

Motivation – finding web communities

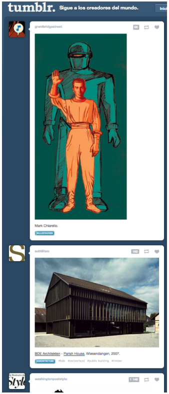
web communities containts small cores



[Kumar et al., 1999]

Motivation – social piggybacking

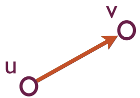
[Gionis et al., 2013]



- event feeds: majority of activity in social networks

Motivation – social piggybacking

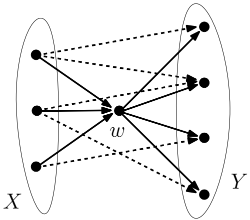
- system throughput proportional to the data transferred between data stores
- feed generation important component to optimize



- primitive operation: transfer data between two data stores
- can be implemented as push or pull strategy
- optimal strategy depends on production and consumption rates of nodes

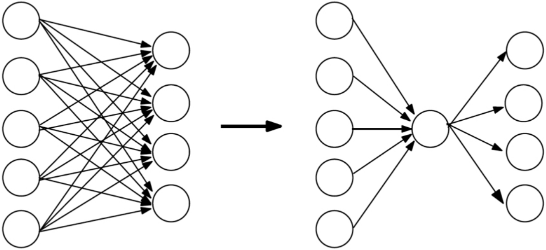
Motivation – social piggybacking

- hub optimization turns out to be a good idea
- depends on finding dense subgraphs



Motivation – 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]



Motivation – 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., 2014a]
- many more

Motivation – 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.

graph stream →

Landscape of related work

- brute force [Johnson and Trick, 1996]
- heuristics [Bomze et al., 1999]
  - spectral algorithms [Alon et al., 1998, McSherry, 2001, Papailiopoulos et al., 2014]
  - belief-propagation methods [Kang et al., 2011]
- enumerating maximal cliques, e.g., [Bron and Kerbosch, 1973, Eppstein et al., 2010, Makino and Uno, 2004]
- NP-hard formulations and various relaxations
  - maximum clique problem [Karp, 1972, Hastad, 1999]
  - k-densest subgraph problem [Bhaskara et al., 2010, Feige et al., 2001]
  - optimal quasi-cliques [Tsourakakis et al., 2013]
- polynomial-time solvable objectives
  - densest subgraph problem [Goldberg, 1984]
  - *"The densest subgraph problem lies at the core of large scale data mining"* [Bahmani et al., 2012]

Preliminaries, 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

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

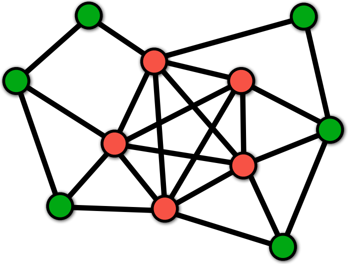
- degree of a node $u \in V$ is $\deg(u) = \deg_V(u)$
- edges between $S \subseteq V$ and $T \subseteq V$ are

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

  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 $G = (V, E)$
- subgraph induced by $S \subseteq V$
- clique: all vertices in S are connected to each other



density measures

- edge density (average degree):

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

(sometimes just drop 2)

- edge ratio:

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

- triangle density:

$$t(S) = \frac{|T(S)|}{|S|}$$

- triangle ratio:

$$\tau(S) = \frac{|T(S)|}{\binom{|S|}{3}}$$

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 $E(S) \geq \alpha\binom{|S|}{2}$

and more

not considered in this tutorial

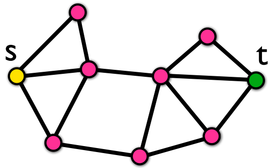
- k-cliques: subset of vertices with pairwise distances at most k
  - distances defined using intermediaries, outside the set
  - not well connected

- k-club: a subgraph of diameter $\leq k$

- k-plex: a subgraph S in which each vertex is connected to at least $|S| - k$ other vertices
  - 1-plex is a clique

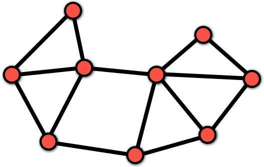
reminder: min-cut and max-cut problems

min-cut problem



- source $s \in V$, destination $t \in V$
- find $S \subseteq V$, s.t.,
- $s \in S$ and $t \in \bar{S}$, and
- minimize $e(S, \bar{S})$

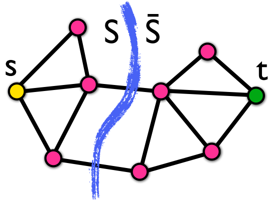
max-cut problem



- find $S \subseteq V$, s.t.,
- maximize $e(S, \bar{S})$

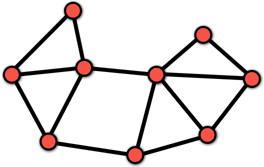
reminder: min-cut and max-cut problems

min-cut problem



- source $s \in V$, destination $t \in V$
- find $S \subseteq V$, s.t.,
- $s \in S$ and $t \in \bar{S}$, and
- minimize $e(S, \bar{S})$
- polynomially-time solvable
- equivalent to max-flow problem

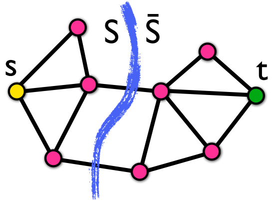
max-cut problem



- find $S \subseteq V$, s.t.,
- maximize $e(S, \bar{S})$

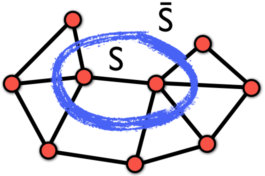
reminder: min-cut and max-cut problems

min-cut problem



- source $s \in V$, destination $t \in V$
- find $S \subseteq V$, s.t.,
- $s \in S$ and $t \in \bar{S}$, and
- minimize $e(S, \bar{S})$
- polynomially-time solvable
- equivalent to max-flow problem

max-cut problem

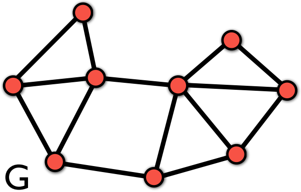


- find $S \subseteq V$, s.t.,
- maximize $e(S, \bar{S})$
- **NP**-hard
- approximation algorithms (0.868 based on SDP)

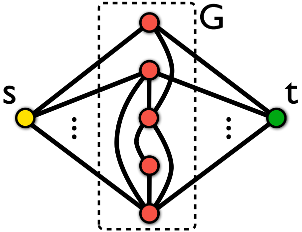
Efficient algorithms for static graphs

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?

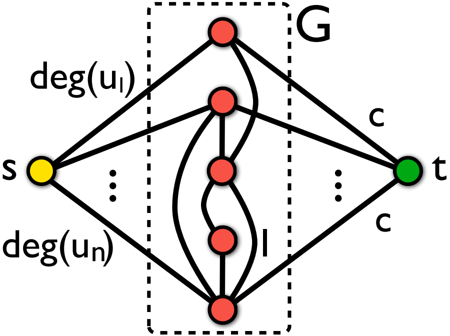
Goldberg's algorithm for densest subgraph

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

$$
\begin{aligned}
\frac{2\,|E(S,S)|}{|S|} &\geq c \\
2\,|E(S,S)| &\geq c|S| \\
\sum_{u \in S} \deg(u) - |E(S,\bar{S})| &\geq c|S| \\
\sum_{u \in S} \deg(u) + \sum_{u \in \bar{S}} \deg(u) - \sum_{u \in \bar{S}} \deg(u) - |E(S,\bar{S})| &\geq c|S| \\
\sum_{u \in \bar{S}} \deg(u) + |E(S,\bar{S})| + c|S| &\leq 2\,|E|
\end{aligned}
$$

Goldberg's algorithm for densest subgraph

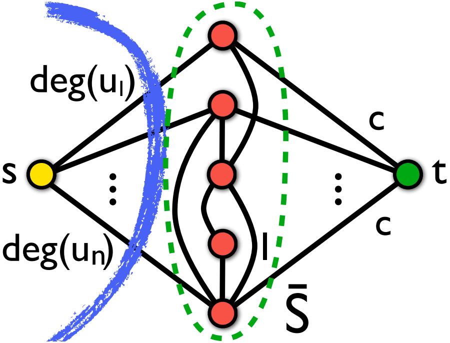
- transformation to min-cut instance



- 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

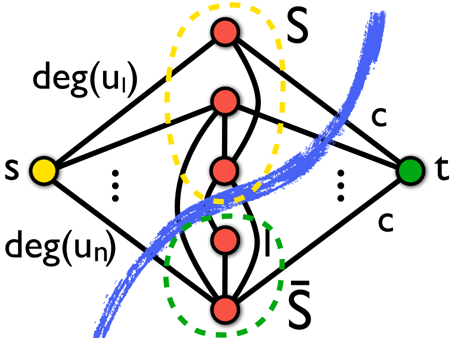
- transform to a min-cut instance



- 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

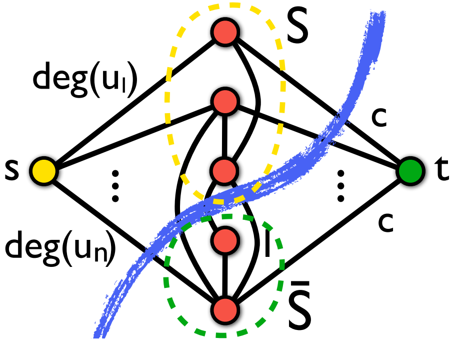
- transform to a min-cut instance



- 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 $d(S) \geq c$

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

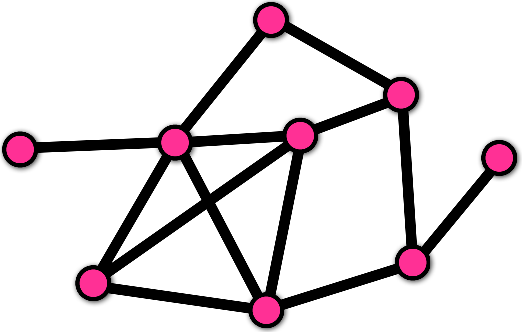
- to find the densest subgraph perform binary search on c
- logarithmic number of min-cut calls
- problem can also be solved with one min-cut call using the parametric max-flow algorithm

densest subgraph problem – discussion

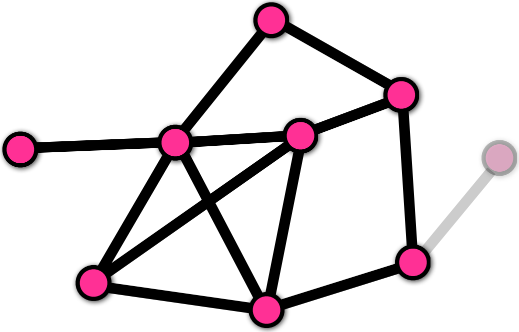
- Goldberg's algorithm polynomial algorithm, but
- $\mathcal{O}(nm)$ time for one min-cut computation
- not scalable for large graphs (millions of vertices / edges)

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

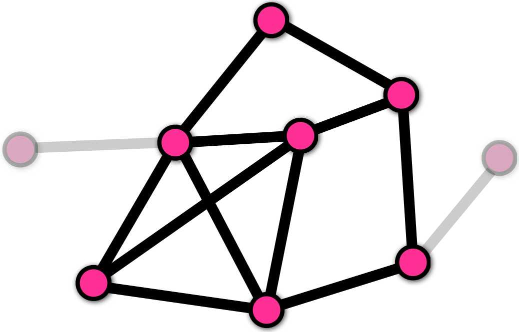
greedy algorithm for densest subgraph — example



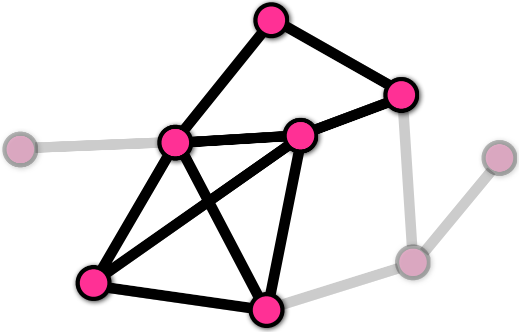
greedy algorithm for densest subgraph — example



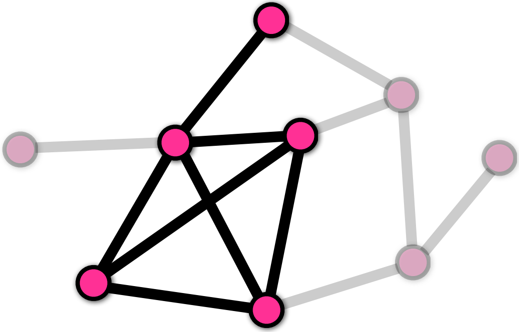
greedy algorithm for densest subgraph — example



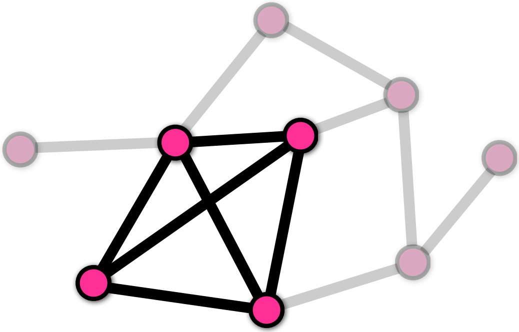
greedy algorithm for densest subgraph — example



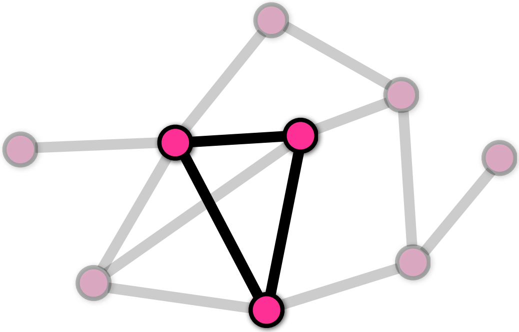
greedy algorithm for densest subgraph — example



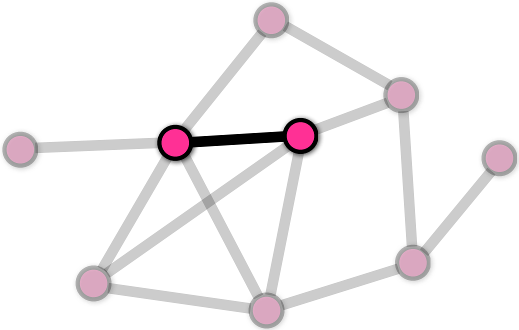
greedy algorithm for densest subgraph — example



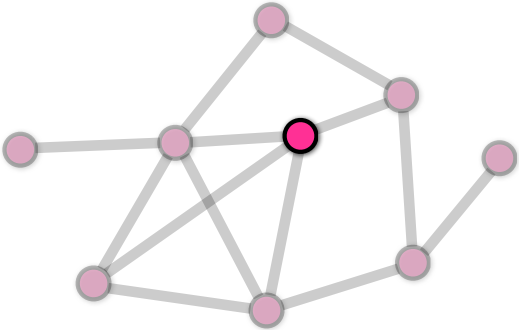
greedy algorithm for densest subgraph — example



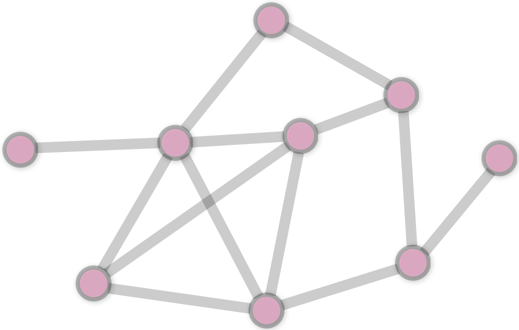
greedy algorithm for densest subgraph — example



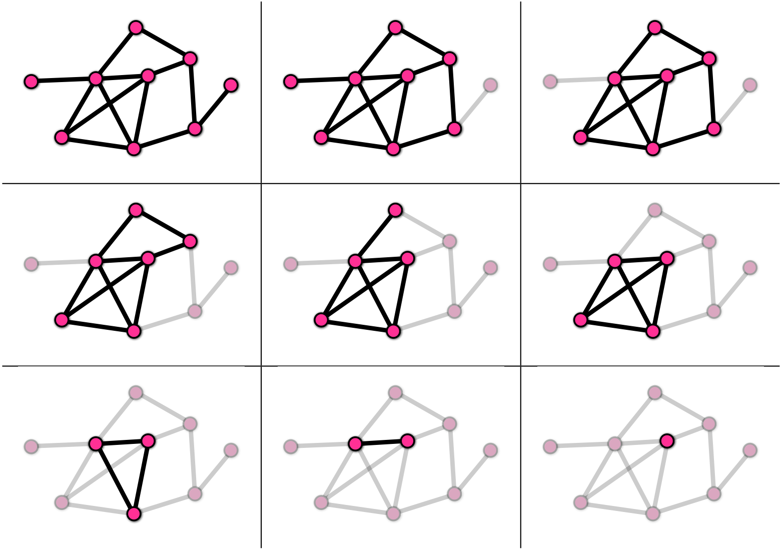
greedy algorithm for densest subgraph — example



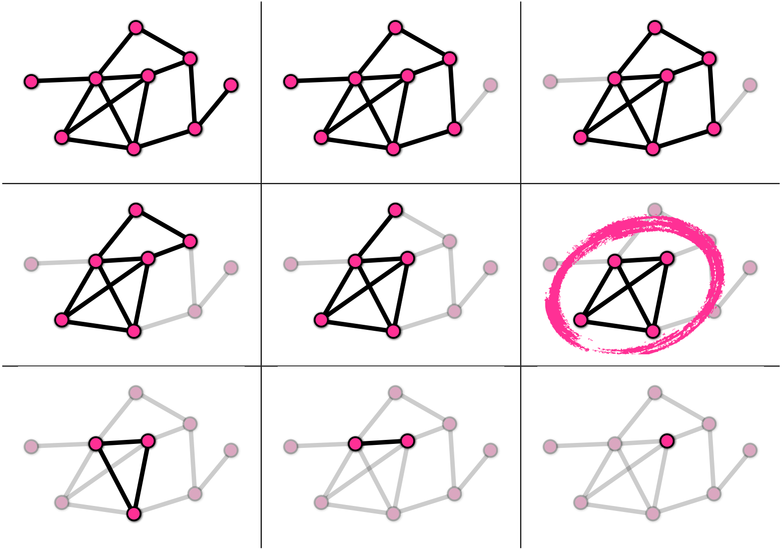
greedy algorithm for densest subgraph — example



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 $d(S^*) = \lambda$ be the maximum degree density
- notice that for all $v \in S^*$ we have $\deg_{S^*}(v) \geq \lambda$
- (why?) by optimality of S^*

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

and thus

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

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 $\geq \lambda |S|/2$
- therefore, greedy returns a solution with degree density at least $\frac{\lambda}{2}$

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

Dense subgraphs on directed graphs – history

- goal: find sets $S, T \subseteq V$ to maximize

$$d(S, T) = \frac{e[S, T]}{\sqrt{|S|\,|T|}}$$

- first introduced in unpublished manuscript [Kannan and Vinay, 1999]
- they provided a $\mathcal{O}(\log n)$-approximation algorithm
- left open the problem complexity

- polynomial-time solution using linear programming (LP) [Charikar, 2000]

Dense subgraphs on directed graphs – history

[Charikar, 2000]

- exact LP-based algorithm
- greedy 2-approximation algorithm running in $\mathcal{O}(n^3 + n^2 m)$

[Khuller and Saha, 2009]

- first max-flow based exact algorithm
- improved running time of the 2-approximation greedy algorithm to $\mathcal{O}(n + m)$!

Directed graphs – algorithms

- reduced problem to $O(n^2)$ LP calls [Charikar, 2000]
- one LP call for each possible ratio $\frac{|S|}{|T|} = c$

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{(i,j)\in E(G)} x_{ij} & \\
\text{such that} & x_{ij} \le s_i, & \text{for all } (i,j) \in E(G) \\
 & x_{ij} \le t_j, & \text{for all } (i,j) \in E(G) \\
 & \displaystyle\sum_i s_i \le \sqrt{c} \;\text{ and }\; \sum_j t_j \le \frac{1}{\sqrt{c}} & \\
 & x_{ij}, s_i, t_j \ge 0 &
\end{array}
$$

Dense subgraphs on directed graphs – greedy

[Charikar, 2000]

input: directed graph $G = (V, E)$, ratio $c = \frac{|S|}{|T|}$

$S \leftarrow V$, $T \leftarrow V$
while both S, T non-empty
  $i_{\min} \leftarrow$ the vertex $i \in S$ that minimizes $|E(\{i\}, T)|$
  $d_S \leftarrow |E(\{i_{\min}\}, T)|$
  $j_{\min} \leftarrow$ the vertex $j \in T$ that minimizes $|E(S, \{j\})|$
  $d_T \leftarrow |E(S, \{j_{min}\})|$
  if $\sqrt{c} d_S \leq \frac{1}{\sqrt{c}} d_T$
    then $S \leftarrow S \setminus \{i_{\min}\}$
    else $ST \leftarrow T \setminus \{j_{\min}\}$

- execute $\mathcal{O}(n^2)$ times; one for each $c = \frac{|S|}{|T|}$
- report best solution
- factor 2 approximation guarantee

Dense subgraphs on directed graphs – greedy

- brute force execution of greedy: $\mathcal{O}(n^2(n+m)) = \mathcal{O}(n^3 + nm))$

[Khuller and Saha, 2009]

- showed that only one execution is needed (instead of $\mathcal{O}(n^2)$)
- total running time $\mathcal{O}(n+m)$

Dense subgraphs on directed graphs – greedy

linear-time greedy [Khuller and Saha, 2009]

definitions:

- let v_i, v_o be the vertices with minimum in- and out-degree
- if $d^-(v_i) \leq d^+(v_o)$ we are in category IN
  otherwise in category OUT

algorithm:

- greedy deletes the minimum-degree vertex
- if in IN, it deletes all incoming edges
- if in OUT, it deletes all outgoing edges
- if the vertex becomes a singleton, it is deleted.
- return the densest subgraph encountered

Dense subgraphs on directed graphs – exact

we wish to answer "are there $S, T \subseteq V$ such that $d(S, T) \geq g$?"

consider

- consider $\alpha = \frac{|S|}{|T|}$ ($\mathcal{O}(n^2)$ possible values)
- network $G' = (\{s, t\} \cup V_1 \cup V_2, E)$, with $V_1 = V_2 = V$

min-cut transformation

- add an edge of capacity m from s to each vertex of V_1 and V_2
- add an edge of capacity $2m + \frac{g}{\sqrt{\alpha}}$ from each vertex of V_1 to t
- add an edge from each vertex j of V_2 to sink t of capacity $2m + \sqrt{\alpha}g - 2\deg(j)$
- for each $(i, j) \in E(G)$, add an edge from $j \in V_2$ to $i \in V_1$ with capacity 2

Dense subgraph problem – summary

- for the degree density measure:
- exact algorithms for undirected and directed graphs
- linear-time 2-approximation achieved by greedy

- how good are these subgraphs?
  study other measures and contrast with degree density

- no control on the size of the subgraph

- what about time-evolving and dynamic graphs?

Edge-surplus framework

introduced by [Tsourakakis et al., 2013]

- for a set of vertices S define edge surplus

$$f(S) = g(e[S]) - h(|S|)$$

where g and h are both strictly increasing

- optimal (g, h)-edge-surplus problem:

  find S^* such that

$$f(S^*) \geq f(S), \quad \text{for all sets } \; S \subseteq S^*$$

Edge-surplus framework

- edge surplus $f(S) = g(e[S]) - h(|S|)$
- example 1

$$g(x) = h(x) = \log x$$

  find S that maximizes $\log \frac{e[S]}{|S|}$

  densest-subgraph problem

- example 2

$$g(x) = x, \quad h(x) = \begin{cases} 0 & \text{if } x = k \\ +\infty & \text{otherwise} \end{cases}$$

  k-densest-subgraph problem

The optimal quasiclique problem

- edge surplus $f(S) = g(e[S]) - h(|S|)$
- consider
$$g(x) = x, \quad h(x) = \alpha \frac{x(x-1)}{2}$$
find S that maximizes $e[S] - \alpha \binom{|S|}{2}$
optimal quasiclique problem [Tsourakakis et al., 2013]
- theorem: let $g(x) = x$ and $h(x) = \alpha x$
we aim to maximize $e(S) - \alpha|S|$
solving $\mathcal{O}(\log n)$ such problems, solves densest subgraph problem

The edge-surplus maximization problem

theorem: let $g(x) = x$ and $h(x)$ concave

then the optimal (g, h)-edge-surplus problem is polynomially-time solvable

proof

$g(x) = x$ is supermodular

if $h(x)$ concave $h(x)$ is submodular

$-h(x)$ is supermodular

$g(x) - h(x)$ is supermodular

maximizing supermodular functions is a polynomial problem

The edge-surplus maximization problem

- poly-time solvable and interesting objectives have linear h
- the optimal quasiclique problem is NP-hard [Tsourakakis, 2014]
- the partitioning version led to a state-of-art streaming balanced graph-partitioning algorithm: Fennel
  - goal: maximize $g(\mathcal{P})$ over all possible k-partitions
  - notice:

$$g(\mathcal{P}) = \underbrace{\sum_i e[S_1]}_{\text{number of edges cut}} - \alpha \underbrace{\sum_i |S_i|^\gamma}_{\text{minimized for balanced partition!}}$$

  - for more details: [Tsourakakis et al., 2014]

Finding optimal quasicliques

adaptation of the greedy algorithm of [Charikar, 2000]

input: undirected graph $G = (V, E)$
output: a quasiclique S
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 subgraph in $G_n, \ldots, G_1$ that maximizes $f(S)$

additive approximation guarantee [Tsourakakis et al., 2013]

Motivating research question

- despite rich landscape of algorithmic tools, until recently, no polynomial algorithm for finding large near-cliques

- can we combine the best of both worlds, namely
  - have poly-time solvable formulation(s) which ...
  - ...consistently succeeds in finding large near-cliques on real-world networks?

- yes! the k-clique densest subgraph problem [Tsourakakis, 2015]

k-clique densest subgraph problem

Definition (k-clique density)

For any $S \subseteq V$ we define its k-clique density $\rho_k(S)$, $k \geq 2$ as $\rho_k(S) = \frac{c_k(S)}{s}$, where $c_k(S)$ is the number of k-cliques induced by S and $s = |S|$

Problem (k-clique DSP)

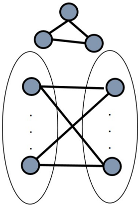
Given $G(V, E)$, find a subset of vertices S^* such that $\rho_k(S^*) = \rho_k^* = \max_{S \subseteq V} \rho_k(S)$

- Notice that the 2-clique DSP is simply the DSP
- We shall refer to the 3-clique DSP as the *triangle densest subgraph problem*

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

Triangle densest subgraph problem

- How different can the densest subgraph be from the triangle densest subgraph?
  In principle, they can be radically different!
  Consider $G = K_{n,n} \cup K_3$



- The interesting question is what happens on real-data
- Can we solve the triangle DSP in polynomial time?
- Can we solve the k-clique DSP in polynomial time?

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*

We will sketch here the idea behind a $O\left(m^{3/2} + \left(nt + \min\left(n, t\right)^3\right) \log n\right)$ algorithm Furthermore,

Theorem

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

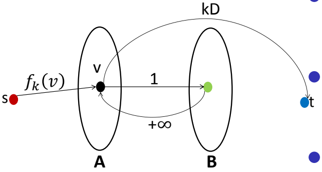
- Even if our construction solves the DSP, Goldberg's algorithm is more efficient

Triangle densest subgraph problem

- Perform binary searches:
  - $\exists S \subseteq V$ such that $t(S) > \alpha|S|$?
- $\mathcal{O}(\log n)$ queries suffice in order to solve the TDSP
  - Any two distinct triangle density values are at least $\mathcal{O}(1/n^2)$ way from each other
  - The optimal density $0 \leq \frac{t}{n} \leq \tau^* \leq \frac{\binom{n}{3}}{n}$
- But what does a binary search correspond to? ...

Triangle densest subgraph problem

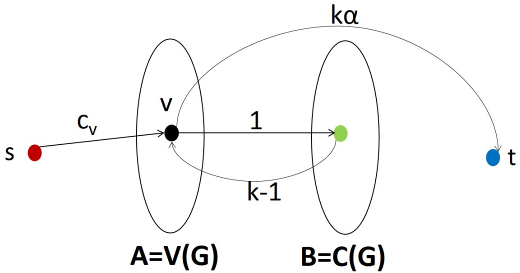
. . . To a maximum flow computation on this network



Construct-Network $(G, \alpha, \mathcal{T}(G))$

- $V(H) \leftarrow \{s\} \cup V(G) \cup \mathcal{T}(G) \cup \{t\}$
- For each vertex $v \in V(G)$ add an arc of capacity 1 to each triangle t_i it participates in
- For each triangle $\Delta = (u, v, w) \in \mathcal{T}(G)$ add arcs to u, v, w of capacity 2
- Add directed arc $(s, v) \in A(H)$ of capacity t_v for each $v \in V(G)$
- Add weighted directed arc $(v, t) \in A(H)$ of capacity 3α for each $v \in V(G)$
- Return network $H(V(H), A(H), w), s, t \in V(H)$

k-clique densest subgraph problem



Triangle densest subgraph problem

Exact-TDS

- List the set of triangles $\mathcal{T}(G)$, $t = |\mathcal{T}(G)|$
- $l \leftarrow \frac{t}{n}, u \leftarrow \frac{(n-1)(n-2)}{6}$
- $S^* \leftarrow \emptyset$
- While$(u \geq l + \frac{1}{n(n-1)})$
  - $\alpha \leftarrow \frac{l+u}{2}$
  - $H_\alpha \leftarrow$ Construct-Network$(G, \alpha, \mathcal{T}(G))$
  - $(S, T) \leftarrow$ minimum st-cut in H_α
  - If($S = \{s\}$), then $u \leftarrow \alpha$
  - otherwise set $S^* \leftarrow \big(S \backslash \{s\}\big) \cap V(G)$ and $l \leftarrow \alpha$
- Return S^*

1. Run time: $O\Big(m^{3/2} + \big(nt + \min(n,t)^3\big) \log n\Big)$
2. Space complexity: $\mathcal{O}(n+t)$. Typically $n \ll t$ on real networks

Triangle densest subgraph problem

1. Set $G_n \leftarrow G$
2. for $k \leftarrow n$ downto 1
   - Let v be the smallest triangle count vertex in G_k
   - $G_{k-1} \leftarrow G_k \backslash \{v\}$
3. Output the triangle densest subgraph among $G_n, G_{n-1}, \ldots, G_1$

- The above peeling algorithm is a 3-approximation algorithm

- The same peeling idea generalizes to the k-clique DSP, providing a k-approximation algorithm

Some experimental findings

Method	Measure	Football
DS	$\frac{\lvert S\rvert}{\lvert V\rvert}(\%)$	100
	2δ	10.6
	f_e	0.094
	3τ	21.12
$\frac{1}{2}$-DS	$\frac{\lvert S\rvert}{\lvert V\rvert}(\%)$	100
	2δ	10.66
	f_e	0.094
	3τ	21.12

Method	Measure	Football
TDS	$\frac{\lvert S\rvert}{\lvert V\rvert}(\%)$	15.7
	2δ	8.22
	f_e	0.48
	3τ	28
$\frac{1}{3}$-TDS	$\frac{\lvert S\rvert}{\lvert V\rvert}(\%)$	15.7
	2δ	8.22
	f_e	0.48
	3τ	28

- Observation 1. Approximate counterparts are close to the optimal exact methods
- Observation 2. The TDS is closer to being a large near-clique compared to the DS

Important remark

- Charikar's algorithm despite being a 2-approximation algorithm performs optimally or close to optimally on real data. This suggests that real-data are "far away" from being adversarial

- Here is one adversarial instance that shows that the 2-approximation is tight

– $G = G_1 \cup G_2$ where $G_1 = K_{d,D}$, G_2 is the disjoint union of D cliques, each of size $d + 1$

– Let $d \ll D$

– How does the Charikar's algorithm perform?

  – Instead of returning the bipartite clique with density $dD/(d + D) \approx d$, it returns a clique of size $d + 1$ with density $d/2$

Computational issues

- The main issue is the size of the bipartite network
  - Both space-wise ...
  - and time-wise, as any max-flow computation depends on its size
- *k*-clique counting is not the main issue. We can count fast based on arboricity based ordering heuristics *k*-cliques efficiently on large networks
  - When the counting part becomes an issue, high-quality approximation algorithms exist, e.g., [Kolountzakis et al., 2012, Tsourakakis et al., 2011, Pagh and Tsourakakis, 2012]

Datasets

Name	n	m
■ Web-Google	875 713	3 852 985
⋆ Epinions	75 877	405 739
⊙ CA-Astro	18 772	198 050
■ Pol-blogs	1 222	16 714
⊙ Email-all	234 352	383 111
⋆ IMDB-B	241 360	530 494
⋆ IMDB-G-B	21 258	42 197

Experimental findings

k-cliques

G	$k=2$		$k=3$		$k=4$		$k=5$	
	f_e	$\|S\|$	f_e	$\|S\|$	f_e	$\|S\|$	f_e	$\|S\|$
$\star$	0.12	1 012	0.26	432	0.40	235	0.50	172
$\odot$	0.11	18 686	0.80	76	0.96	62	0.96	62
$\blacksquare$	0.19	16 714	0.54	102	0.59	92	0.63	84
$\odot$	0.13	553	0.38	167	0.48	122	0.53	104

(p,q)-bicliques

G	$(p,q)=(1,1)$		$(p,q)=(2,2)$		$(p,q)=(3,3)$	
	f_e	$\|S\|$	f_e	$\|S\|$	f_e	$\|S\|$
$\star$	0.001	9 177	0.06	181	0.30	40
$\star$	0.001	6 437	0.41	18	0.43	17

Densest subgraph sparsifiers

Abstraction: We shall abstract both the k-clique DSP and the (p, q)-biclique DSP as a densest subgraph problem in a hypergraph. Let $\mathcal{H}$ be the resulting hypergraph and $\epsilon > 0$ be an accuracy parameter [Mitzenmacher et al., 2015].

Theorem

- *Sample each hyperedge $e \in E_{\mathcal{H}}$ independently with probability $p = \frac{6}{\epsilon^2}\frac{\log n}{D}$*
- *Then, the following statements hold simultaneously with high probability:*
  - *For all $U \subseteq V$ such that $\rho(U) \geq D$, $\tilde{\rho}(U) \geq (1-\epsilon)C \log n$ for any $\epsilon > 0$*
  - *For all $U \subseteq V$ such that $\rho(U) < (1-2\epsilon)D$, $\tilde{\rho}(U) < (1-\epsilon)C \log n$ for any $\epsilon > 0$*

Densest subgraph sparsifiers

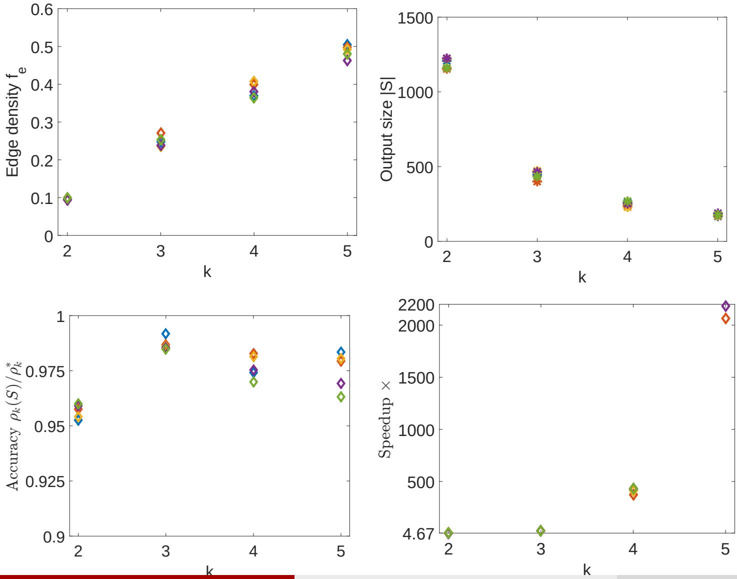
Technical difficulty

- 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

From the previous theorem, we obtain the following corollaries

- $(1+\Theta(\epsilon))$-approximation, expected speedup $\mathcal{O}(\frac{1}{p_D^2})$, expected space reduction is $\mathcal{O}(\frac{1}{p_D})$
- Naturally results in a single pass $(1+\Theta(\epsilon))$-approximation semi-streaming algorithm for a dynamic stream of edges. Same result obtained independently by [Esfandiari et al., 2015, McGregor et al., 2015]

Sampling effect, Epinions network



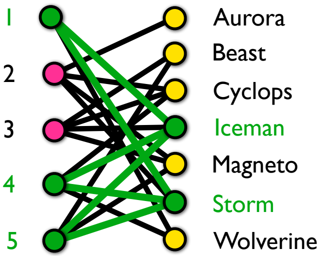
Large Near Bicliques

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

⇔

	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

⇔



- transaction data ⇔ binary data ⇔ bipartite graphs
- frequent itemsets ⇔ bi-cliques

Large Near Bicliques

- We generalize the idea of k-cliques by maximizing the average (p, q)-biclique densities
- For $p = q = 1$ we obtain the well-known densest subgraph problem
- We provide general network construction techniques which can be used to maximize the (p, q)-biclique density for any $p, q = \Theta(1)$
  - Our network construction techniques can be used to maximize densities of other types of subgraphs as well
- We can justify speedups of the order $\mathcal{O}(\rho^{*2} / \log^2 n)$, compared to the exact maximum flow computation based algorithm

Datasets

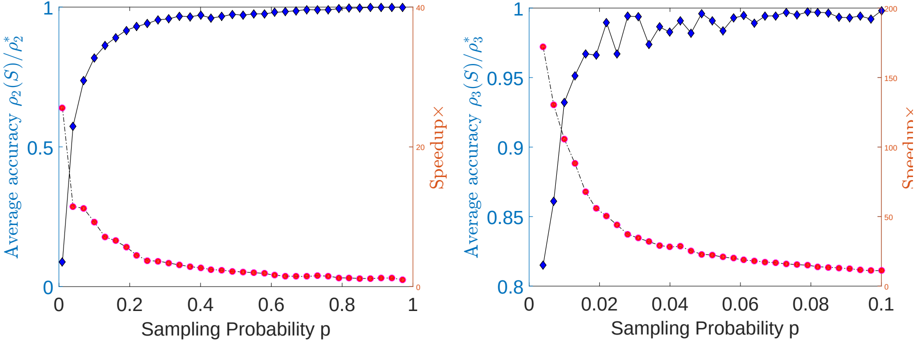
Name	n	m
■ Web-Google	875 713	3 852 985
⋆ Epinions	75 877	405 739
⊙ CA-Astro	18 772	198 050
■ Pol-blogs	1 222	16 714
⊙ Email-all	234 352	383 111
⋆ IMDB-B	241 360	530 494
⋆ IMDB-G-B	21 258	42 197

k-clique and (p, q)-biclique counts and run times

Name	c_3	T	c_4	T	c_5	T
■ Web-Google	11.4M	8.5	32.5M	16.5	82M	36.4
⋆ Epinions	16M	1.6	5.8M	4.8	17.5M	13.4
⊙ CA-Astro	13M	0.6	9.6M	3.94	65M	27.2
■ Pol-blogs	101K	0.05	422K	0.2	1.4M	0.7
⊙ Email-all	383K	0.4	1.1M	0.9	2.7M	1.9

Name	$c_{2,2}$	T	$c_{3,3}$	T
⋆ IMDB-B	691 594	3.6	261 330	3.3
⋆ IMDB-G-B	14 919	0.1	2 288	0.1

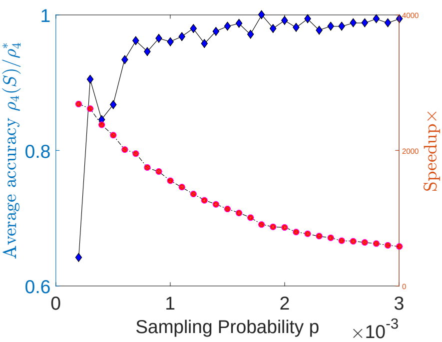
Ranging p, $k = 2, 3$

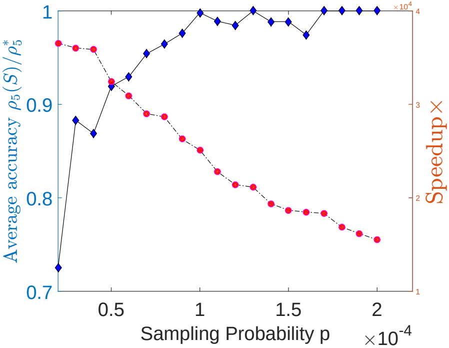


k=2 k=3

Accuracy $\rho_k(S)/\rho_k^*$ and speedup as functions of the sampling probability p for the CA-Astro collaboration network

Ranging p, $k = 4, 5$





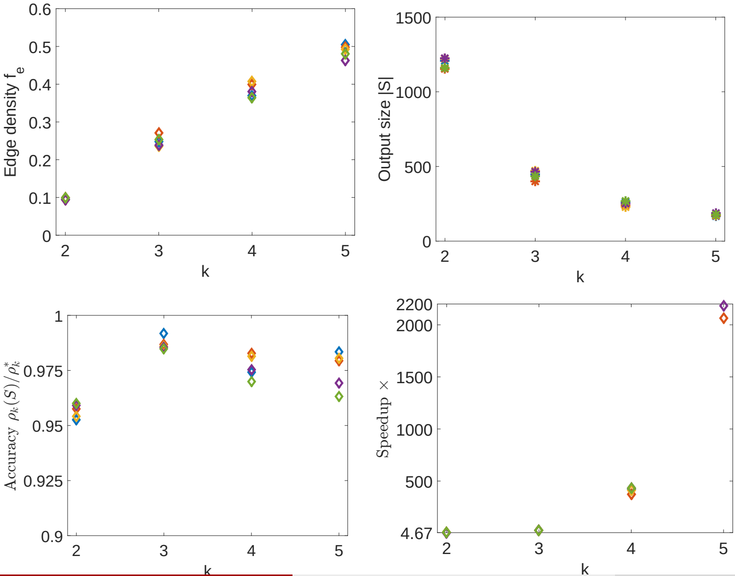
k=4 k=5

Accuracy $\rho_k(S)/\rho_k^*$ and speedup as functions of the sampling probability p for the CA-Astro collaboration network

Observations – Ranging p

- Notice that $\frac{c_k}{n} \leq \rho_k^* \leq \frac{\binom{n}{k}}{n}$
  - We observe that an efficient strategy is to guess a large value of ρ_k^*, i.e., sample with smallest value for p Then, while concentration is not deduced, keep doubling p
- The speedups for $k = 2$ -while valuable- are not impressive as the graphs are pretty sparse to begin with
- However, for $k \geq 3$ the speedups start becoming significant, reaching the order of 4×10^4 for $k = 5$, which achieving excellent accuracies

Sampling effect, Epinions



Accuracies and speedups

- Runtimes (exact), accuracies and speedups (random sampling)
  - Exact: For $k = 2$ the slowest run time was 33.9 secs
  - Sampling: We obtain a speedup of $\approx 3\times$ using sampling Accuracies greater always than 95%
  - Exact: For $k = 5$, the exact algorithm cannot run on one dataset Run times for other datasets, 37 939.6, 2 107.2, 24.04, 52.4
  - Sampling: Speedups range from $410.3\times$ to $77\,288\times$. Accuracies close to 100%
- The results for $k = 3, 4$ interpolate. For the detailed findings, please look at our paper

Effect of hierarchy

k-cliques

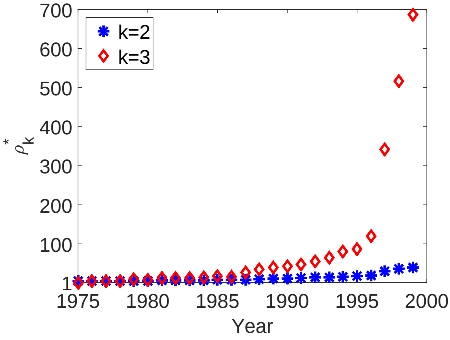
G	$k=2$		$k=3$		$k=4$		$k=5$	
	f_e	$\lvert S\rvert$	f_e	$\lvert S\rvert$	f_e	$\lvert S\rvert$	f_e	$\lvert S\rvert$
$\star$	0.12	1 012	0.26	432	0.40	235	0.50	172
$\odot$	0.11	18 686	0.80	76	0.96	62	0.96	62
$\blacksquare$	0.19	16 714	0.54	102	0.59	92	0.63	84
$\odot$	0.13	553	0.38	167	0.48	122	0.53	104

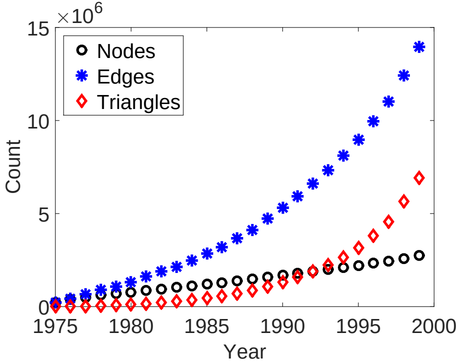
(p,q)-bicliques

G	$(p,q)=(1,1)$		$(p,q)=(2,2)$		$(p,q)=(3,3)$	
	f_e	$\lvert S\rvert$	f_e	$\lvert S\rvert$	f_e	$\lvert S\rvert$
$\star$	0.001	9 177	0.06	181	0.30	40
$\star$	0.001	6 437	0.41	18	0.43	17

Time evolving networks

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



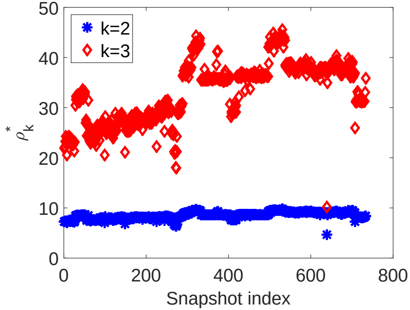


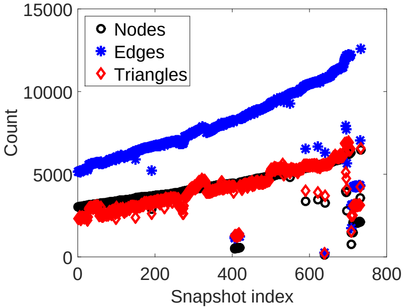
Time evolving networks

- We observe in the left Figure that both ρ_2^* and ρ_3^* exhibit an increasing trend.
- This increasing trend becomes is mild for ρ_3^* up to 1995, but then it takes off
- What makes this finding even more interesting as the number of edges grows faster than the number of triangles
- We are seeing an outlier - the company Allergan, Inc. This company tends to cite all their previous patents with each new patent and creates a dense subregion in the graph

Time evolving networks

Autonomous systems dataset contains 733 daily instances which span an interval of 785 days from November 8 1997 to January 2 2000

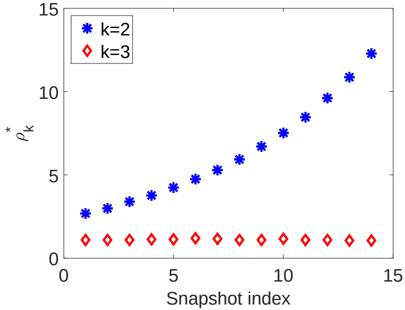


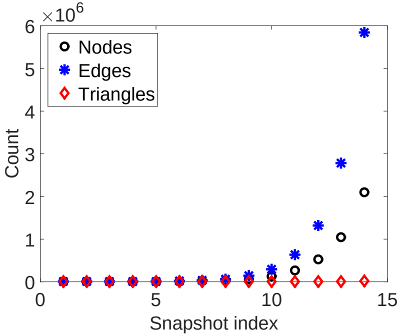


- Despite the average degree increases over time, the optimal density for $k = 2$ remains roughly the same
- The optimal density for $k = 3$ exhibits a mild increasing trend

Time evolving networks

This is how density evolves in stochastic Kronecker graphs with seed matrix [0.9 0.5;0.5 0.2] as we increase the number of nodes as 2^i for $i = 8$ up to $i = 21$





- This and other popular seed matrices can't reproduce what we observe in real-networks with respect to the optimal density

Peeling in batches

The following algorithm due to Bahmani, Vassilvitski and Kumar leads to efficient MapReduce and streaming algorithms [Bahmani et al., 2012]

1. Set $S, \tilde{S} \leftarrow V$
2. **while** $S \neq \emptyset$ **do**
   - $A(S) \leftarrow \{i \in S : D_i(S) \leq 2(1+\epsilon)\rho(S)\}$
   - $S \leftarrow S \backslash A(S)$
   - **if** $\rho(S) \geq \rho(\tilde{S})$ **then** $\tilde{S} \leftarrow S$
3. Return $\tilde{S}$

Peeling in batches

- **Claim**. The previous algorithm achieves a $(2+2\epsilon)$ approximation. Furthermore, it outputs after $\mathcal{O}(\log_{1+\epsilon}(n))$ rounds
- **Proof** .
  - Approximation guarantee: Fix any optimal solution S^*. Consider the first round when a node $v \in S^*$ becomes removed. Let U be the set of vertices at that point. Then, $\rho^* \leq D_v(S^*) \leq D_v(U) \leq (2+2\epsilon)\rho(U)$. QED
  - Number of rounds is $\mathcal{O}(\log_{1+\epsilon}(n))$: The idea is that in each round, we throw away a constant fraction of the vertices $2e(S) > \sum_{v \notin A(S)} D_v(S) > (|S| - |A(S)|)2(1+\epsilon)\rho(S) \rightarrow$ $|A(S)| > \frac{\epsilon}{1+\epsilon}|S| \rightarrow |S| - |A(S)| < \frac{|S|}{1+\epsilon}$

Peeling in batches

Few more remarks

- The previous claim results directly in a $(2+\epsilon)$ approximation algorithm, using $\tilde{O}(n)$ space and $\mathcal{O}(\log n/\epsilon)$
- Similar claim holds for MapReduce. In each round we need to compute degrees and remove $A(S)$
- Many believed that $\mathcal{O}(\log n/\epsilon)$ passes were likely to be necessary
- However, the densest subgraph sparsifier theorem results directly in a **single pass** streaming algorithm that uses $\tilde{O}(n)$ space and provides a $(1+\epsilon)$ approximation guarantee. See also, [Esfandiari et al., 2015, McGregor et al., 2015]

Variations of the DSP

k-densest subgraph	$\delta(S) = \frac{2e[S]}{\lvert S\rvert}, \lvert S\rvert = k$	**NP**-hard
DalkS	$\delta(S) = \frac{2e[S]}{\lvert S\rvert}, \lvert S\rvert \geq k$	**NP**-hard
DamkS	$\delta(S) = \frac{2e[S]}{\lvert S\rvert}, \lvert S\rvert \leq k$	*L*-reduction to DkS

Densest k subgraph problem

- Does not admit a PTAS unless P=NP
- Feige, Peleg and Kortsarz gave a $\mathcal{O}(n^{\frac{1}{3}})$ approximation algorithm [Feige et al., 2001]
- State of the art algorithm due to Bhaskara et al. provides $\mathcal{O}(n^{\frac{1}{4}+\epsilon})$ approximation guarantee for any $\epsilon > 0$ [Bhaskara et al., 2010]
- Closing the gap between lower and upper bounds is a significant problem

DalkS is **NP**-hard

Proof sketch.

- We reduce the DkS to the DalkS. We are given a graph G and a value k we wish to know whether $\exists S \subseteq V$ such that $\rho(S) \geq \lambda, |S| = k$
- Construct $H = K_{n^2} \cup G$ and run DalkS with lower bound on the number of vertices $n^2 + k$
- Turns out that the part of the optimal DalkS solution on H is the answer to DkS

For the details, see [Khuller and Saha, 2009]

2-approximation for DalkS [Khuller and Saha, 2009]

- The algorithm starts with $G_0 \leftarrow G, D_0 \leftarrow \emptyset$
- In the i-th iteration, we compute the densest subgraph H_i from G_{i-1}
- If $|V(D_{i-1})| + |V(H_i)| \geq k$, terminate
- else
  - $D_i \leftarrow D_{i-1} \cup H_i$
  - Remove H_i from G_{i-1}
  - For every $v \in G_{i-1} \backslash H_i$ add a selfloop of weight w_v where $w_v = |N(v) \cap H_i|$
- When the algorithm stops, each D_i is padded with arbitrary vertices to make their size k, let D'_i be the resulting subgraph
- The algorithm returns the subgraph D'_j with maximum density among the D'_is

2-approximation for DalkS – example

Suppose this is the input to the DalkS

- $k = n + \sqrt{2n}$

- $G = H_1 \cup H_2 \cup H_3 \cup H_4$
  - H_1 is a clique on $\sqrt{2n}$ vertices
  - H_2 is a tree on n vertices
  - H_3 is a cycle on n^2 vertices
  - H_4 is a set of n disjoint vertices

2-approximation for DalkS – example

Let's run the 2-approximation algorithm on G

- First we find H_1 as it is the densest subgraph of G
- In the second iteration it will find H_3
- Therefore, the algorithm has two options:
  - Return $H_1 \cup H_3$
  - Append n arbitrary vertices to H_1. These could well be the n isolated vertices
- In both cases the resulting subgraph has density ≈ 1
- However $H_1 \cup H_2$ has density $\frac{2n}{n+\sqrt{2n}} \approx 2$

Some more remarks

- [Andersen and Chellapilla, 2009] proved that an α approximation for DamkS implies a $\mathcal{O}(\alpha^2)$ approximation algorithm for the DkS
- [Khuller and Saha, 2009] improved this, by showing that an α approximation for DamkS implies a 4α approximation algorithm for the DkS
- The algorithmic ideas we showed for undirected case work for DalkS as well

Efficient algorithms for dynamic graphs

Dynamic setting

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

- $\textsc{Initialize}(n)$: Initialize the algorithm with an empty n-node graph.
- $\textsc{Insert}(u, v)$: Insert edge (u, v) to the graph.
- $\textsc{Delete}(u, v)$: Delete edge (u, v) from the graph.
- $\textsc{QueryValue}$: Output a γ-approximate value of $\rho^*(G) = d^*$

Dynamic setting

The performance of a data structure is measured in term of four different metrics.

- Space-complexity: This is given by the total space (in terms of bits) used by the data structure.
- Update-time: This is the time taken to handle an INSERT or DELETE operation.
- Query-time: This is the time taken to handle a QUERYVALUE operation.
- Preprocessing-time: This is the time taken to handle the INITIALIZE operation. Unless explicitly mentioned otherwise, in this paper the preprocessing time will always be $\tilde{\mathcal{O}}(n)$.

Streaming vs. Dynamic efficiency

- Streaming algorithms' community cares primarily about the space efficiency.
- Dynamic algorithms' community care primarily about the update and query times.
- [Bhattacharya et al., 2015] provide the first result that successfully combines both types of efficiencies simultaneously for the densest subgraph problem
  - Research direction: Can we develop similar type of results for other graph theoretic problems?

$(2+\epsilon)$-approximation 1-pass dynamic semi-streaming algorithm

Theorem ([Bhattacharya et al., 2015])

We can process a dynamic stream of updates in the graph G in $\tilde{\mathcal{O}}(n)$ space, and with high probability return a $(2+\mathcal{O}(\epsilon))$-approximation of $d^* = \max_{S\subseteq V} \rho(S)$ at the end of the stream.

- Remark: To obtain both results we introduce the (α, d, L)-decomposition. It generalizes the well-known d-core, namely the (unique) largest induced subgraph with every node having degree at least d.

(α, d, L)-decomposition – Definition

- Fix any $\alpha \geq 1$, $d \geq 0$, and any positive integer L.
- Consider a family of subsets $Z_1 \supseteq \cdots \supseteq Z_L$.
- The tuple $(Z_1, \ldots, Z_L)$ is an (α, d, L)-decomposition of the input graph $G = (V, E)$ iff:
  - $Z_1 = V$ and,
  - for every $i \in [L-1]$, we have

$$Z_{i+1} \supseteq \{v \in Z_i : D_v(Z_i) > \alpha d\}$$

and

$$Z_{i+1} \cap \{v \in Z_i : D_v(Z_i) < d\} = \emptyset.$$

(α, d, L)-decomposition – Key property

Theorem

- *Fix any* $\alpha \geq 1$, $d \geq 0$, $\epsilon \in (0,1)$, $L \leftarrow 2 + \lceil \log_{(1+\epsilon)} n \rceil$.
- *Let* $(Z_1, \ldots, Z_L)$ *be an* (α, d, L)*-decomposition of* $G = (V, E)$.

– *If* $d > 2(1+\epsilon)d^*$, *then* $Z_L = \emptyset$.

– *If* $d < d^*/\alpha$, *then* $Z_L \neq \emptyset$ *and there is an index* $j \in [L]$ *such that* $\rho(Z_j) \geq d/(2(1+\epsilon))$.

Remark 1: A key property of the densest subgraph that prior work [Charikar, 2000] and our work use throughout our work is that $D_v(S^*) \geq d^*$ for any $S^* \subseteq V$ such that $\rho(S^*) = d^*$.
Remark 2: Notice that $\frac{m}{n} \leq d^* < n - 1$.

(α, d, L)-decomposition – Algorithmic aspect

(Rough) Idea of how to turn the previous theorem into an algorithm.

- Discretize the range of d^* as $d_k \leftarrow (1+\epsilon)^{k-1} \cdot \frac{m}{n}$, $k \in [K]$ where $K = \mathcal{O}(\log_{1+\epsilon}(n))$.
- For every $k \in [K]$, construct an (α, d_k, L)-decomposition $(Z_1(k), \ldots, Z_L(k))$, where $L = \mathcal{O}(\log_{1+\epsilon}(n))$.
- Let $k' \leftarrow \max\{k \in [K] : Z_L(k) \neq \emptyset\}$.

Then we have the following guarantees:

1. $d^*/(\alpha(1+\epsilon)) \leq d_{k'} \leq 2(1+\epsilon) \cdot d^*$.
2. There exists an index $j' \in [L]$ such that $\rho(Z_{j'}) \geq d_{k'}/(2(1+\epsilon))$.

$(2+\epsilon)$-approximation 1-pass dynamic semi-streaming algorithm

Our streaming algorithm relies on the fact that if we sample independently each edge with probability (roughly) $\tilde{\mathcal{O}}(\frac{1}{d})$, we can create an (α, d, L)-decomposition whp.

Lemma

Fix a $d > 0$, and let S be a collection of $cm(L-1)\log n/d$ mutually independent simple random samples from the edge-set E of the input graph $G = (V, E)$. With high probability we can construct from S an (α, d, L)-decomposition $(Z_1, \ldots, Z_L)$ of G, using $\tilde{\mathcal{O}}(n)$ bits of space.

$(2+\epsilon)$-approximation 1-pass dynamic semi-streaming algorithm

Emulating Charikar's peeling paradigm.

The algorithm works by partitioning the samples in S evenly among $(L-1)$ groups $\{S_i\}$, $i \in [L-1]$

- Set $Z_1 \leftarrow V$.
- FOR $i = 1$ to $(L-1)$: Set $Z_{i+1} \leftarrow \{v \in Z_i : D_v(Z_i, S_i) \geq (1-\epsilon)\alpha c \log n\}$.

Here, $D_v(Z_i, S_i)$ is the number of neighbors of v in set Z_i connected through the set of edges S_i.

$(2+\epsilon)$-approximation 1-pass dynamic semi-streaming algorithm

- "Guess" the number of edges m.
- For each guess of m, build $\mathcal{O}(\log n/\epsilon)$ $(\alpha, d_k = (1+\epsilon)^{k-1}\frac{m}{n}, L)$-decompositions, one for each density guess d_k. Set $\alpha = \frac{1+\epsilon}{1-\epsilon}$.
- For each guess of d_k maintain a sample S of $cm(L-1)\log n/d_k = \tilde{\mathcal{O}}(n)$ random edges.
- Perform peeling and find k'.

Few remarks.

1. The case of dynamic streams is dealt with by using ℓ_0 samplers [Jowhari et al., 2011].
2. For the dynamic case, we wish to find an α large enough to be lazy enough when we update our data structures, small enough to achieve a good approximation.

Fully dynamic $(4+\epsilon)$-approximation algorithm $\tilde{\mathcal{O}}(n)$ space

Theorem ([Bhattacharya et al., 2015])

- *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* $\tilde{\mathcal{O}}(n)$ *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* $\mathcal{O}(T\,\mathrm{poly}\log n)$. *(Time efficiency)*

Fully dynamic $(4 + \epsilon)$-approximation algorithm $\mathcal{O}(n + m)$ space

- As before, we discretize the range of d^* in the same way, i.e., in powers of $(1 + \epsilon)$ by defining the values $\{d_k\}, k \in [K]$.
- For each d_k we are able to maintain an (α, d_k, L)-decomposition of G in time $\mathcal{O}(L/\epsilon) = \mathcal{O}(\log n/\epsilon^2)$ per edge update.
- The total time for all K decompositions is $\mathcal{O}(\log^2 n/\epsilon^3)$ per update operation.
- Remark: We find an α large enough to be lazy enough, small enough to achieve a good approximation. It turns out using a fine tuned potential function analysis, that for $\alpha = 2 + \Theta(\epsilon)$ we achieve good amortized time and a $(4 + \Theta(\epsilon))$-approximation.

Remark: How to maintain efficiently a random sample of $\tilde{\mathcal{O}}(n)$ edges when the graph changes?

Q1 How do we maintain dynamically the random sample(s) of $\tilde{\mathcal{O}}(n)$ edges?

– If we naively run an ℓ_0 sampler responsible for an edge in the sample for each update, we need $\tilde{\mathcal{O}}(n)$ time per update.

Idea: When an update takes place, only one ℓ_0 sampler needs to be invoked. Let $E = \binom{[n]}{2} \supseteq E^{(t)}$.

- Let $h : E \to [s_k]$ be an ℓ-wise independent hash function
- The i-th "bucket" $Q_i^{(t)}$ is responsible for all edges such that $h(e) = i$, for each $i = 1, \ldots, s_k$. We also run an independent copy of an ℓ_0 sampler.

Few more remarks

- To make Chernoff+union bound work we need $l = \tilde{\mathcal{O}}(n)$. To construct our hash function we invoke the construction due to [Pagh and Pagh, 2008].
- The previous theorem [Bhattacharya et al., 2015] opens the direction towards single-pass semi-streaming algorithms over dynamic streams with polylogarithmic update and query times.
- [Epasto et al., 2015] provided a $(2+\epsilon)$-approximation algorithm, $\mathcal{O}(polylog(n)) = \tilde{\mathcal{O}}(1)$ amortized time per update, $\mathcal{O}(n+m)$ space under the assumption that deletions are *random*.

Problem variants

Problem variants II : top-k dense subgraphs

Top-k dense subgraphs

- in many cases we want to find more than one dense subgraph
- idea: find all dense subgraphs (e.g., denser than a threshold)
- cut enumeration techniques to output all near-optimal dense subgraphs ([Saha et al., 2010])
- in practice, this method suffers from output degeneracies:
  - many subsets of a dense subgraph tend to be near-optimally dense as well

Top-k dense subgraphs

- another approach
  - (i) find a dense subgraph S
  - (ii) remove all vertices and edges of S
  - (iii) iterate
- reported subgraphs are disjoint
- certain degree of overlap can be desirable [Balalau et al., 2015]

Top-k dense subgraphs with limited overlap

problem formulation ([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

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

Top-k dense subgraphs with limited overlap

algorithm MinAndRemove ([Balalau et al., 2015])

input: undirected graph $G = (V, E)$, parameters k and α
output: k subgraphs $G_1, \ldots, G_k$ with overlap at most α

while less than k subgraphs found and G non-empty
    find minimal densest subgraph $G_i = (V_i, E_i)$
    for each $v \in V_i$
        $\Delta_G(v) \leftarrow$ the set of neighbors of v in G
    remove $\lceil (1-\alpha)|V_i| \rceil$ nodes with minimum $|\Delta_G(v) \setminus V_i|$
        and all their edges from G

Top-k dense subgraphs with limited overlap

summary of results ([Balalau et al., 2015])

- MinAndRemove finds optimal solution,
  if this contains disjoint subgraphs
- MinAndRemove works shown to work well in practice
- faster algorithm, at small loss of accuracy

Problem variants III : core decomposition

k-core decomposition

widely used technique for partitioning graphs

$$k\text{-core} = \text{largest subgraph with vertex degrees} \geq k$$

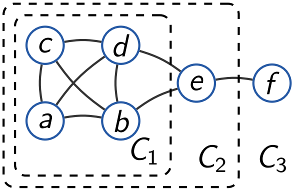
cores form a chain, k-core $\subseteq (k-1)$-core; let

$$k\text{-shell} = \text{vertices in } k\text{-core but not in } (k+1)\text{-core}$$

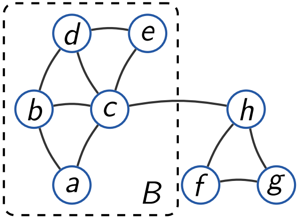
algorithm to find shells:

1. **while** G is not empty
2. $\quad v \leftarrow$ vertex with the smallest degree
3. $\quad$ assign v to k-shell
4. $\quad$ remove v from G

core decomposition and density are not compatible



$d(C_1) = \frac{6}{4} < \frac{8}{5} = d(C_2)$



only one core but

$d(B) = \frac{7}{5} > \frac{11}{8} = d(G)$

density-friendly decomposition

goal:

adapt k-core decomposition for density

obtain a nested sequence of increasingly dense subgraphs

[Tatti and Gionis, 2015]

locally-dense subgraphs

informally,

subgraph H is locally-dense = any subgraph of H is denser than any subgraph outside H

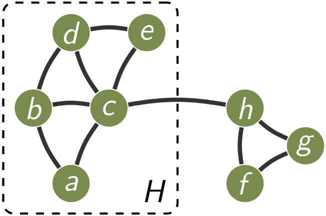
formally, define augmented density

$$d(X, Y) = \frac{|E(X)| + |E(X, Y)|}{|X|}, \quad \text{for} \quad X \cap Y = \emptyset$$

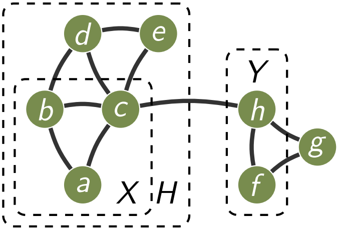
subgraph H is locally-dense if

$$d(X, H \setminus X) > d(Y, H), \quad \text{for any} \quad X \subsetneq H, Y \cap H = \emptyset$$

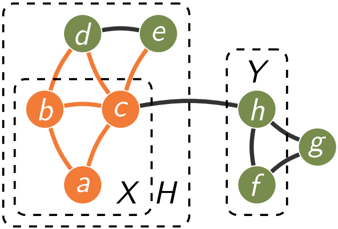
example



example

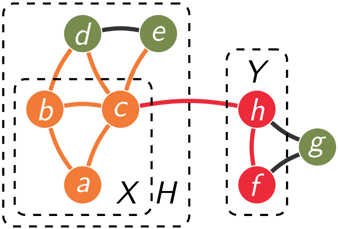


example



$$d(X, H \setminus X) = 6/3$$

example



$$d(X, H \setminus X) = 6/3$$
$$d(Y, H) = 2/2$$

properties

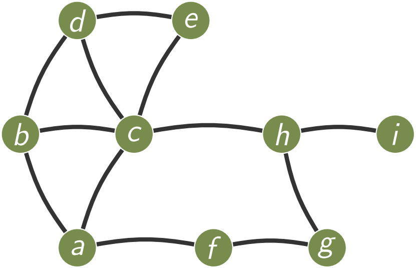
locally-dense subgraphs form a chain

$$\emptyset = B_0 \subsetneq B_1 \subsetneq B_2 \subsetneq \cdots \subsetneq B_k = G$$

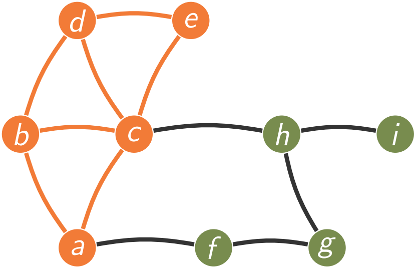
B_i is the densest subgraph containing B_{i-1}

$$\begin{aligned}
B_1 &= \text{densest subgraph} \\
B_2 &= \arg\max_{B \supsetneq B_1} d(B \setminus B_1, B_1) \\
&\dots \\
B_i &= \arg\max_{B \supsetneq B_{i-1}} d(B \setminus B_{i-1}, B_{i-1})
\end{aligned}$$

first approach to compute the subgraphs

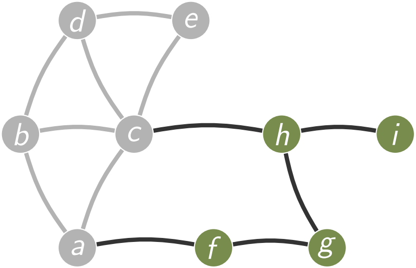


first approach to compute the subgraphs



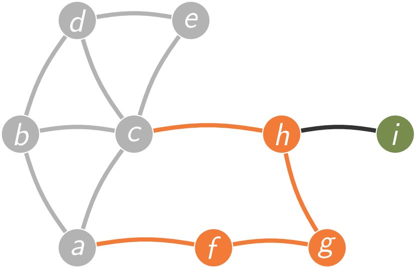
find B_1

first approach to compute the subgraphs



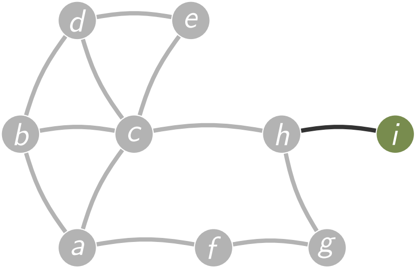
find B_1
delete B_1

first approach to compute the subgraphs



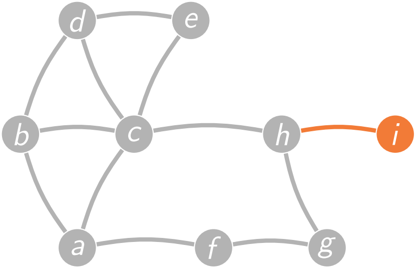
find B_1
delete B_1
find B_2

first approach to compute the subgraphs



find B_1
delete B_1
find B_2
delete B_2

first approach to compute the subgraphs



find B_1
delete B_1
find B_2
delete B_2
find B_3

computing the subgraphs

define

$$F(\alpha) = \arg\max_X |E(X)| - \alpha|X|$$

Goldberg showed that

- $F(\alpha)$ can be solved with a min-cut
- there is α such that $F(\alpha)$ is the densest subgraph

we can show that

- $F(\alpha)$ is locally-dense
- for every B_i there is α such that $B_i = F(\alpha)$

computing the subgraphs

find all B_i by varying α (with divide-and-conquer)

algorithm: $\textsc{Exact}(X, Y)$

1. select α such that $X \subseteq F(\alpha) \subsetneq Y$
2. $Z \leftarrow F(\alpha)$
2. **if** $(Z \neq X)$
3. **output** Z
3. $\textsc{Exact}(X, Z)$
3. $\textsc{Exact}(Z, Y)$

- we need only $2k - 3$ calls of $F(\alpha)$
  (k is the number of locally-dense subgraphs)
- $O(n^2 m)$ total running time, in practice much faster
- $X \subset F(\alpha) \subset Y$ allows optimizations

approximation with profiles

approximation guarantees are tricky:

- algorithm may return different number of subgraphs

define a profile:

$$p(i; \mathcal{B}) = \begin{cases} d(B_1) & \text{if } i \leq |B_1| \\ d(B_2 \setminus B_1, B_1) & \text{if } |B_1| < i \leq |B_2| \\ \dots \end{cases}$$

core decomposition

let $\mathcal{C}$ be the core decomposition

let $\mathcal{B}$ be the optimal locally-dense decomposition

then

$$p(i; \mathcal{C}) \geq p(i; \mathcal{B})/2, \quad \text{for every} \quad i$$

for $i = 1$, this implies

$$d(C_1) \geq d(B_1)/2$$

extending Charikar's algorithm

$C_1 \leftarrow$ densest subgraph of form $v_1, \ldots v_{|C_1|}$

$C_2 \leftarrow$ subgraph maximizing $d(v_1, \ldots v_{|C_2|} \setminus C_1, C_1)$

$C_3 \leftarrow$ subgraph maximizing $d(v_1, \ldots v_{|C_3|} \setminus C_2, C_2)$

...

The graphs C_i

- can be found in $O(n^2)$-time naively
- can be found in $O(n)$-time with PAV algorithm [Ayer et al., 1955]

greedy decomposition

let $\mathcal{C}$ be the greedy decomposition

(found by the extension of Charikar's algorithm)

let $\mathcal{B}$ be the optimal locally-dense decomposition

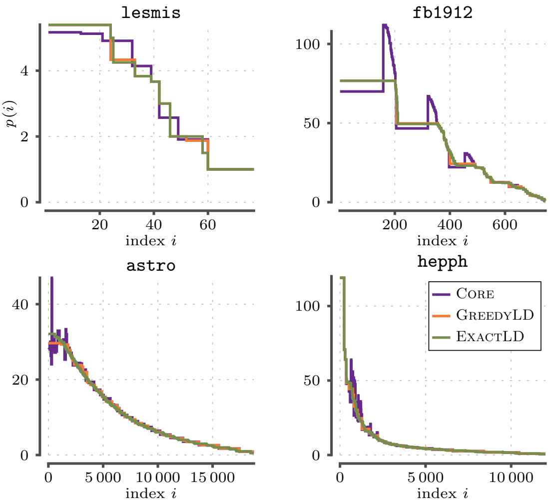
then

$$p(i;\mathcal{C}) \geq p(i;\mathcal{B})/2, \quad \text{for every} \quad i$$

for $i = 1$, this implies

$$d(C_1) \geq d(B_1)/2$$

how well these algorithm perform?



summary (density-friendly decomposition)

- decomposition based on average density
- can be computed exactly in $\mathcal{O}(n^2 m)$ time, faster in practice
- can be $1/2$-approximated in linear time by
  - k-core decomposition
  - greedy algorithm

future work:

- consider different density functions
- control the size of the decomposition

Problem variants IV : community search

community detection problems

- typical problem formulations require non-overlapping and complete partition of the set of vertices
- quite restrictive
- inherently ambiguous: research group vs. bicycling club

- additional information can resolve ambiquity
- community defined by two or more people

the community-search problem

- given graph $G = (V, E)$, and
- given a subset of vertices $Q \subseteq V$ (the query vertices)
- find a community H that contains Q

applications

- find the community of a given set of users (cocktail party)
- recommend tags for an image (tag recommendation)
- form a team to solve a problem (team formation)

center-piece subgraph

[Tong and Faloutsos, 2006]

- given: graph $G = (V, E)$ and set of query vertices $Q \subseteq V$
- find: a connected subgraph H that
  (a) contains Q
  (b) optimizes a goodness function $g(H)$

- main concepts:
- `k_softAND`: a node in H should be well connected to at least k vertices of Q
- $r(i, j)$ goodness score of j wrt $q_i \in Q$
- $r(Q, j)$ goodness score of j wrt Q
- $g(H)$ goodness score of a candidate subgraph H
- $H^* = \arg\max_H g(H)$

center-piece subgraph

[Tong and Faloutsos, 2006]

- $r(i, j)$ goodness score of j wrt $q_i \in Q$

  probability to meet j in a random walk with restart to q_i

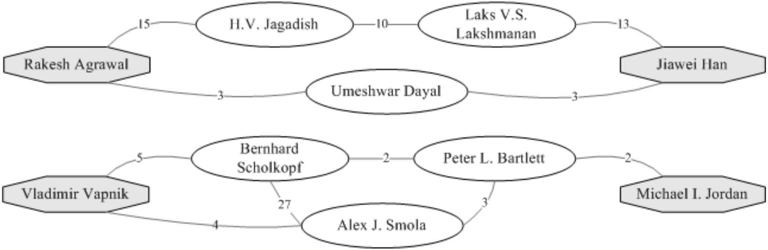
- $r(Q, j)$ goodness score of j wrt Q

  probability to meet j in a random walk with restart to k vertices of Q

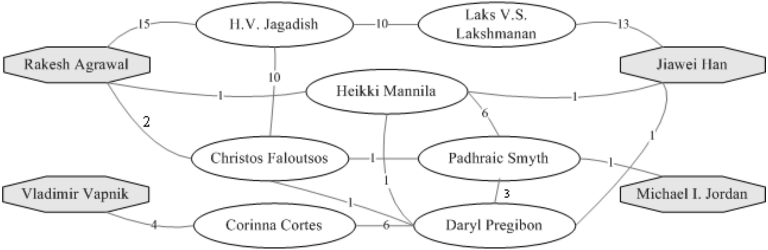
- proposed algorithm:

1. greedy: find a good destination vertex j ito add in H
2. add a path from each of top-k vertices of Q path to j
3. stop when H becomes large enough

center-piece subgraph — example results



(a) "K_softANDquery": $k = 2$



the community-search problem

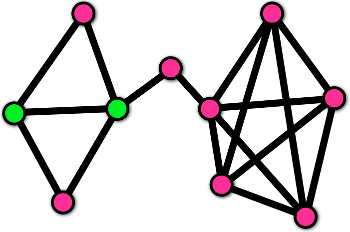
- given: graph $G = (V, E)$ and set of query vertices $Q \subseteq V$
- find: a connected subgraph H that
  - (a) contains Q
  - (b) optimizes a density function $d(H)$
  - (c) possibly other constraints

- density function (b):

  average degree, minimum degree, quasiclique, etc.

  measured on the induced subgraph H

free riders



- remedy 1: use min degree as density function
- remedy 2: use distance constraint

$$d(Q,j) = \sum_{q \in Q} d^2(q_i, j) \leq B$$

the community-search problem

adaptation of the greedy algorithm of [Charikar, 2000]

input: undirected graph $G = (V, E)$, query vertices $Q \subseteq V$
output: connected, dense subgraph H

1 set $G_n \leftarrow G$
2 for $k \leftarrow n$ downto 1
2.1 remove all vertices violating distance constraints
2.2 let v be the smallest degree vertex in G_k among all vertices not in Q
2.3 $G_{k-1} \leftarrow G_k \setminus \{v\}$
2.4 if left only with vertices in Q or disconnected graph, stop
3 output the subgraph in $G_n, \ldots, G_1$ that maximizes $f(H)$

properties of the greedy algorithm

- returns optimal solution if no size constraints
- upper-bound constraints make the problem **NP**-hard (heuristic solution, also adaptation of the greedy)
- generalization for monotone constraints and monotone objective functions

experimental evaluation (qualitative summary)

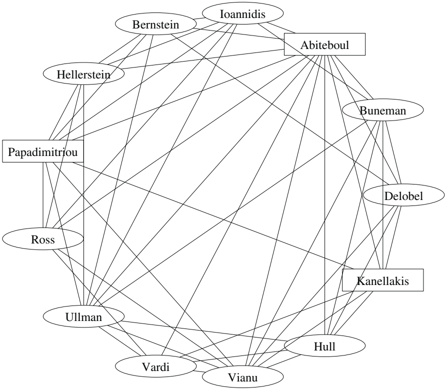
baseline: increamental addition of vertices

- start with a Steiner tree on the query vertices
- greedily add vertices
- return best solution among all solutions constructed

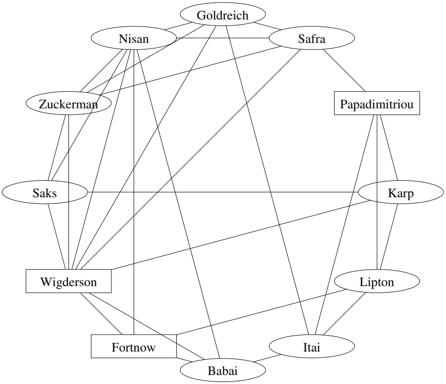
example result in DBLP

- proposed algorithm: min degree $= 3$, avg degree $= 6$
- baseline algorithm: min degree $= 1.5$, avg degree $= 2.5$

the community-search problem — example results



(a) Database theory



(b) Complexity theory

(from [Sozio and Gionis, 2010])

monotone functions

function f is monotone non-increasing if

for every graph G and

for every subgraph H of G it is

$$f(H) \leq f(G)$$

the following functions are monotone non-increasing:

- the query nodes are connected in H (0/1)
- are the nodes in H able to perform a set of tasks?
- upper-bound distance constraint
- lower-bound constraint on the size of H

generalization to monotone functions

generalized community-search problem

given

- a graph $G = (V, E)$
- a node-monotone non-increasing function f
- $f_1, \ldots, f_k$ non-increasing boolean functions

find

- a subgraph H of G
- satisfying $f_1, \ldots, f_k$ and
- maximizing f

generalized greedy

1 set $G_n \leftarrow G$

2 for $k \leftarrow n$ downto 1

2.1 remove all vertices violating any constraint $f_1, \ldots, f_k$

2.2 let v minimizing $f(G_k, v)$

2.3 $G_{k-1} \leftarrow G_k \setminus \{v\}$

3 output the subgraph H in $G_n, \ldots, G_1$ that maximizes $f(H, v)$

generalized greedy

theorem

generalized greedy computes an optimum solution
for the generalized community-search problem

running time

- depends on the time to evaluate the functions $f_1, \ldots, f_k$
- formally $\mathcal{O}(m + \sum_i nT_i)$
- where T_i is the time to evaluate f_i

Problem variants V : heavy subgraphs

discovering heavy subgraphs

- given a graph $G = (V, E, d, w)$

  with a distance function $d : E \to \mathbb{R}$ on edges

  and weights on vertices $w : V \to \mathbb{R}$

- find a subset of vertices $S \subseteq V$

  so that

1. total weight in S is high
2. vertices in S are close to each other

[Rozenshtein et al., 2014a]

discovering heavy subgraphs

- what does total weight and close to each other mean?
- total weight

$$W(S) = \sum_{v \in S} w(v)$$

- close to each other

$$D(S) = \sum_{u \in S} \sum_{v \in S} d(u, v)$$

- want to maximize $W(S)$ and minimize $D(S)$
- maximize

$$Q(S) = \lambda\, W(S) - D(S)$$

applications of discovering heavy subgraphs

- finding events in networks
- vertices correspond to locations
- weights model activity recorded in locations
- distances between locations
- find compact regions (neighborhoods) with high activity

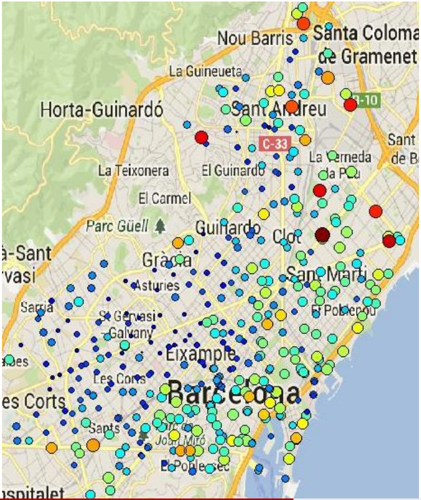
event detection

- sensor networks and traffic measurements

event detection

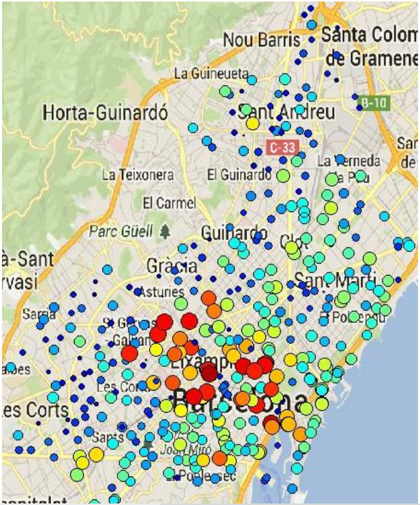
15.11.2012

ordinary day, no events



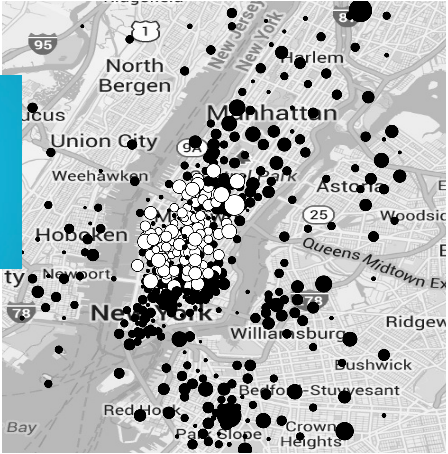
11.09.2012

Catalunya national day



event detection

- location-based social networks



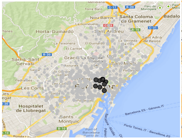
discovering heavy subgraphs

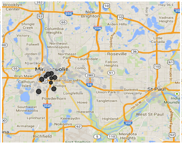
- maximize $Q(S) = \lambda W(S) - D(S)$
- objective can by negative
- add a constant term to ensure non-negativity
- maximize $Q(S) = \lambda W(S) - D(S) + D(V)$

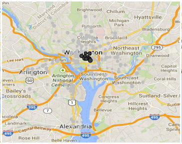
discovering heavy subgraphs

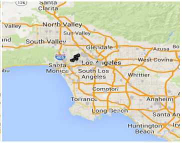
- maximize $Q(S) = \lambda W(S) - D(S) + D(V)$
- objective is submodular (but not monotone)
- can obtain $\frac{1}{2}$-approximation guarantee [Buchbinder et al., 2012]
- problem can be mapped to the max-cut problem which gives 0.868-approximation guarantee [Rozenshtein et al., 2014a]

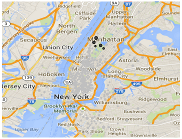
events discovered with bicing and 4square data











(a) Barcelona: 11.09.12 National Day of Catalonia

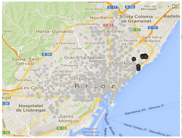
(b) Minneapolis: 4.07.12 Independence Day

(c) Washington, DC: 27.05.13 Memorial Day

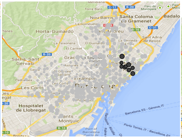
(d) Los Angeles: 31.05.10 Memorial Day

(e) New York: 6.09.10 Labor Day

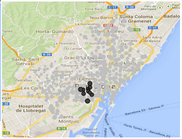
Figure 4: Public holiday city-events discovered using the SDP algorithm.



(a) 01.06.12 Primavera sound music festival



(b) 18.09.12 festival of the Poblenou neighborhood



(c) 31.10.12 Halloween

Problem variants VI :

dense subgraphs in interaction networks

dense subgraphs in interaction networks

- interaction networks : networks with temporal information
  - phonecall networks
  - SMS networks
  - email networks
  - conversation in social-media platforms

- hypothesis : analysis of temporal information
  can reveal hidden structure

[Rozenshtein et al., 2014b]

problem formulation

- given interaction network $G = (V, E)$
- where edges $E = \{(u, v, t)\}$ have time-stamps
- find

  subset of vertices $S \subseteq V$, and

  set T of k time intervals of bounded length

- so that the subgraph induced by S and projected in T is as dense as possible

iterative approach

- decompose the problem in two subproblems
  1. given fixed set of intervals find densest subgraph
  2. given fixed set of vertices find optimal set of intervals
- iterate until convergence

the two subproblems

- subproblem 1 : find optimal vertices given intervals
  - standard densest subgraph problem
  - use the algorithms of Goldberg, or Charikar, etc.

- subproblem 2 : find optimal intervals given vertices
  - NP-hard problem
  - develop greedy heuristic based on
    the generalized maximum coverage problem
    - iteratively add k intervals
    - select a new interval to maximize density per unit of time
  - due to concavity property
    searching the next interval can be done in linear time

sample experimental results — enron email network

dataset

Name	$\lvert V \rvert$	$\lvert \pi(E) \rvert$	$\lvert E \rvert$	$\lvert T \rvert$	$d(\pi(G))$	$d(H)$
Enron	1143	2019	6245	8080	3.53	14.38

dynamic dense subgraphs

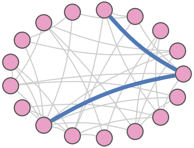
			Community density			Community size		
Dataset	B	K	GA	BA	BASE	GA	BA	BASE
Enron	1	1	6.18	6.18	6.18	11	11	11
		5	10	10.37	6.18	17	16	11
		10	12.2	12.38	6.18	20	21	11
	7	1	6.36	6.36	6.36	11	11	11
		5	11.26	11.23	6.36	19	26	11
		10	13.07	13.07	6.36	28	28	11

sample experimental results — twitter network

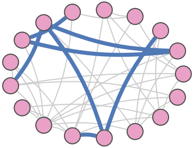
Method	Size	Density	Hashtags
GA	9	4.9	aaltoes, startup, vc, summerofstartups, web, startups, entrepreneur, slush10, skype, funrank, africa, mobile, demoday, design, linkedin, aalto

sample experimental results — facebook network

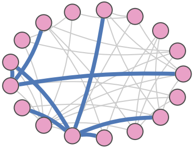
duration
4h 8min



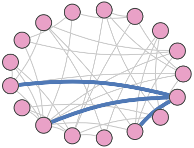
duration
2 days, 2h 32min



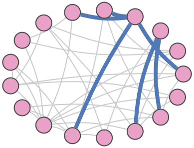
duration
13h 53min



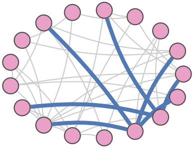
duration
5h 30min



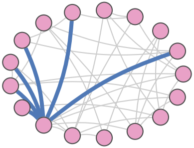
duration
17h 56min



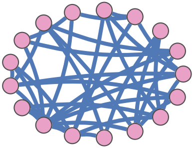
duration
21h 39min



duration
21h 4min



density
4.24



Open problems

Open problems I

- can we improve the $(4 + \epsilon)$ approximation guarantee?
- what about weighted graphs?
- polylogarithmic worst-case update time?
- space- and time-efficient fully dynamic algorithm for other graph problems, e.g., single-source shortest paths?
  - remark: for the connectivity problem, one can combine the space-efficient streaming algorithm of [Ahn et al., 2012] with the fully-dynamic algorithm of [Kapron et al., 2013]

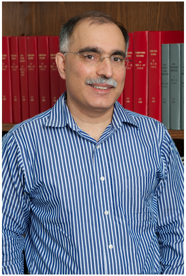
Open problems II

- improve lower bounds for dynamic case [Henzinger et al., 2015]
- for which graph problems does uniform sampling result in high-quality approximation?
  - triangle sparsifiers [Tsourakakis et al., 2011]
  - densest subgraphs [Bhattacharya et al., 2015], [Mitzenmacher et al., 2015]
  - d-max cut, d-sum max clustering [Esfandiari et al., 2015]
  - main difficulty: Chernoff + union bound does not work because of exponential number of bad events

Open problems III

- further study of top-k densest subgraph problem, and
  develop approximation guarantees
- incorporate temporal and/or spatial information
  application: finding local events in social networks
- dense subgraphs with query nodes in graph streams
  preprocessing vs. query-time processing trade-off
- incorporate developed techniques into real-time analytics systems
- deploy existing tools on more real-world applications
  (for code see `https://github.com/tsourolampis`)

Acknowledgements



Shamir Khuller

Renato Werneck

Nikolaj Tatti

references I

Ahn, K. J., Guha, S., and McGregor, A. (2012).
Graph sketches: sparsification, spanners, and subgraphs.
In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14.

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*.

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.

Angel, A., Sarkas, N., Koudas, N., and Srivastava, D. (2012).
Dense subgraph maintenance under streaming edge weight updates for real-time story identification.
Proceedings of the VLDB Endowment, 5(6):574–585.

Ayer, M., Brunk, H. D., Ewing, G. M., Reid, W. T., and Silverman, E. (1955).
An empirical distribution function for sampling with incomplete information.
The Annals of Mathematical Statistics, 26(4):641–647.

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.

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.

references IV

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.

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.

Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. (1999).

The maximum clique problem.

In *Handbook of combinatorial optimization*, pages 1–74. Springer.

references V

Bron, C. and Kerbosch, J. (1973).
Algorithm 457: finding all cliques of an undirected graph.
CACM, 16(9).

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. (2012).
A tight linear time (1/2)-approximation for unconstrained submodular maximization.
In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*.

Charikar, M. (2000).
Greedy approximation algorithms for finding dense components in a graph.
In *APPROX*.

references VI

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.

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.

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.

references VII

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.

Eppstein, D., Löffler, M., and Strash, D. (2010).

Listing all maximal cliques in sparse graphs in near-optimal time.

In *ISAAC*.

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

Applications of uniform sampling: Densest subgraph and beyond.

arXiv preprint arXiv:1506.04505.

references VIII

Feige, U., Kortsarz, G., and Peleg, D. (2001).
The dense k-subgraph problem.
Algorithmica, 29(3).

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.

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.

Goldberg, A. V. (1984).
Finding a maximum density subgraph.
Technical report, University of California at Berkeley.

references IX

Hastad, J. (1999).
Clique is hard to approximate within $n^{1-\epsilon}$.
Acta Mathematica, 182(1).

Henzinger, M., Krinninger, S., Nanongkai, D., and Saranurak, T. (2015).
Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture.

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).

references X

Johnson, D. S. and Trick, M. A. (1996).

Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993, volume 26.

American Mathematical Soc.

Jowhari, H., Saglam, M., and Tardos, G. (2011).

Tight bounds for lp samplers, finding duplicates in streams, and related problems.

In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58.

Kang, U., Chau, D. H., and Faloutsos, C. (2011).

Mining large graphs: Algorithms, inference, and discoveries.

In *International Conference on Data Engineering (ICDE)*, pages 243–254.

references XI

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.

Kapron, B. M., King, V., and Mountjoy, B. (2013).

Dynamic graph connectivity in polylogarithmic worst case time.

In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1131–1142.

references XII

Karande, C., Chellapilla, K., and Andersen, R. (2009).
Speeding up algorithms on compressed web graphs.
Internet Mathematics, 6(3):373–398.

Karp, R. M. (1972).
Reducibility among combinatorial problems.
In Miller, R. and Thatcher, J., editors, *Complexity of Computer Computations*.

Khuller, S. and Saha, B. (2009).
On finding dense subgraphs.
In *ICALP*.

references XIII

Kolountzakis, M. N., Miller, G. L., Peng, R., and Tsourakakis, C. E. (2012).
Efficient triangle counting in large graphs via degree-based vertex partitioning.
Internet Mathematics, 8(1-2):161–185.

Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999).
Trawling the Web for emerging cyber-communities.
Computer Networks, 31(11–16):1481–1493.

Makino, K. and Uno, T. (2004).
New algorithms for enumerating all maximal cliques.
In *Algorithm Theory-SWAT 2004*, pages 260–272. Springer.

references XIV

McGregor, A., Tench, D., Vorotnikova, S., and Vu, H. T. (2015).
Densest subgraph in dynamic graph streams.
arXiv preprint arXiv:1506.04417.

McSherry, F. (2001).
Spectral partitioning of random graphs.
In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 529–537. IEEE.

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.

references XV

Pagh, A. and Pagh, R. (2008).
Uniform hashing in constant time and optimal space.
SIAM J. Comput., 38(1):85–96.

Pagh, R. and Tsourakakis, C. E. (2012).
Colorful triangle counting and a mapreduce implementation.
Information Processing Letters, 112(7):277–281.

Papailiopoulos, D., Mitliagkas, I., Dimakis, A., and Caramanis, C. (2014).
Finding dense subgraphs via low-rank bilinear optimization.
In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1890–1898.

references XVI

Peleg, D. (2000).

Informative labeling schemes for graphs.

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

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

Event detection in activity networks.

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

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

Discovering dynamic communities in interaction networks.

In *Machine Learning and Knowledge Discovery in Databases*.

references XVII

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.

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.

Sozio, M. and Gionis, A. (2010).
The community-search problem and how to plan a successful cocktail party.
In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*.

references XVIII

Tatti, N. and Gionis, A. (2015).
Density-friendly graph decomposition.
In *Proceedings of the 24th International Conference on World Wide Web*.

Thorup, M. (2004).
Compact oracles for reachability and approximate distances in planar digraphs.
Journal of the ACM (JACM), 51(6):993–1024.

Tong, H. and Faloutsos, C. (2006).
Center-piece subgraphs: problem definition and fast solutions.
In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.

references XIX

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.

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., Gkantsidis, C., Radunovic, B., and Vojnovic, M. (2014).
Fennel: Streaming graph partitioning for massive scale graphs.
In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 333–342. ACM.

references XX

Tsourakakis, C. E. (2014).
Mathematical and algorithmic analysis of network and biological data.
arXiv preprint arXiv:1407.0375.

Tsourakakis, C. E., Kolountzakis, M. N., and Miller, G. L. (2011).
Triangle sparsifiers.
J. Graph Algorithms Appl., 15(6):703–726.