Aalto University
School of Science

Algorithmic methods for mining large graphs
Lecure 3 : Finding dense subgraphs

Aristides Gionis

Aalto University

Bertinoro International Spring School 2016
March 7–11, 2016

course agenda

• introduction to graph mining	Tue afternoon
• computing basic graph statistics	Tue afternoon, Wed morning
• finding dense subgraphs	Wed afternoon, Thu morning
• spectral graph analysis	Thu afternoon
• additional topics	Fri morning
– inferring hierarchies in graphs	
– mining dynamic graphs	
– graph sparsifiers	

what this lecture 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 finding dense subgraphs
- problem variants

motivating applications

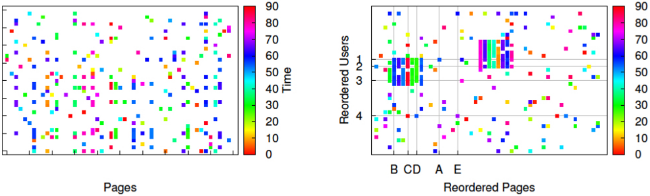
motivation – correlation mining

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]

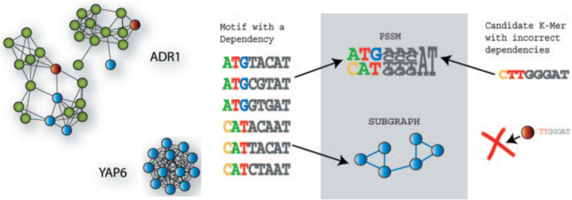
motivation – e-commerce

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

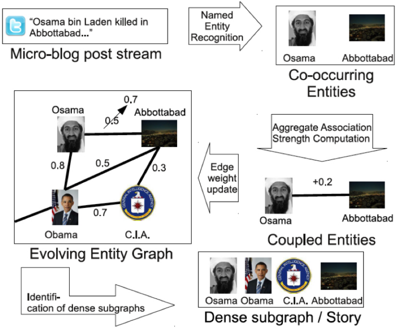
large almost bipartite cliques correspond to sub-markets

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]

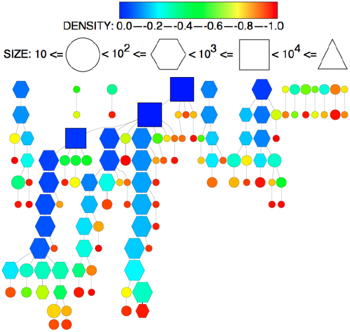
- mining of twitter data
- vertices correspond to entities
- edges correspond to co-occurence of entities
- dense subgraphs capture news stories

motivation – graph mining

understanding the structure of real-world networks

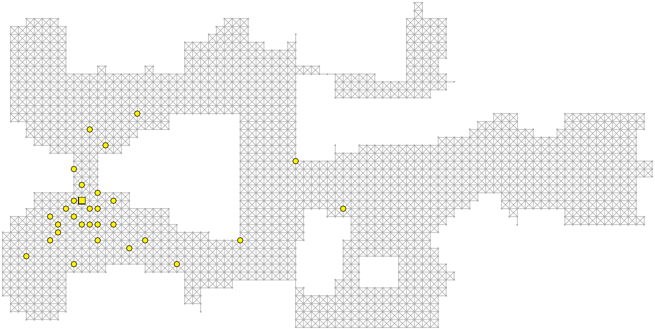
[Sarıyüce et al., 2015]

nucleus decomposition of a graph



(3,4)-nuclei forest for facebook

motivation – distance queries in graphs

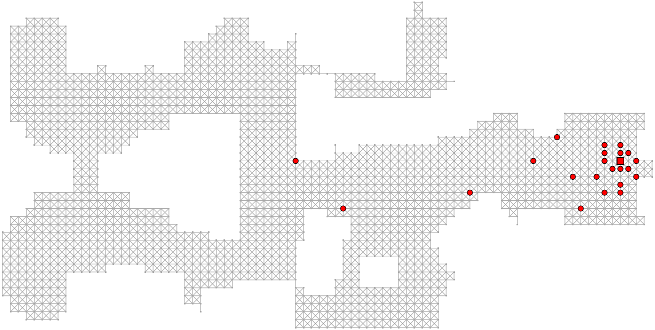


- $L(u) \equiv$ set of pairs $(v, \text{dist}(u, v))$

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

figure from [Delling et al., 2014]

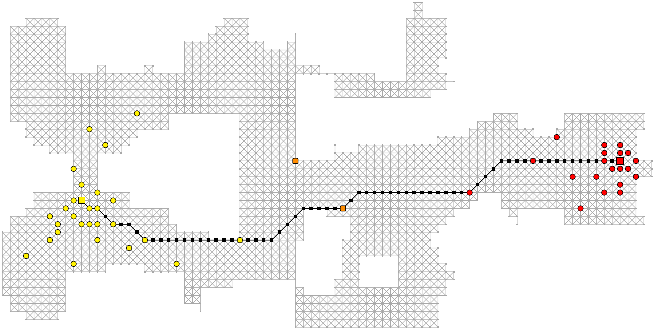
motivation – distance queries in graphs



- 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

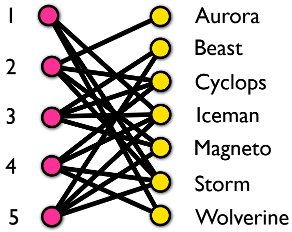
motivation – frequent itemsets and dense subgraphs

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

⇔

	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

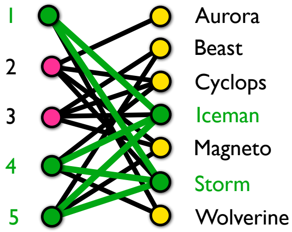
motivation – frequent itemsets and dense subgraphs

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)

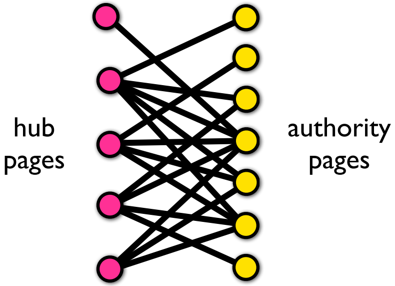
2. consider a web graph with no communities

   then small cores are unlikely

- both observations motivated from theory of random graphs

motivation – finding web communities

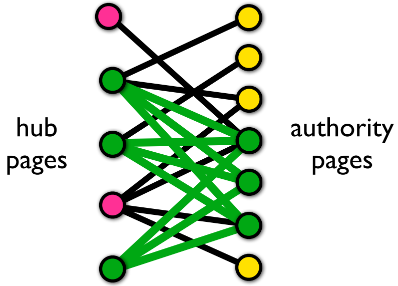
a web community



[Kumar et al., 1999]

motivation – finding web communities

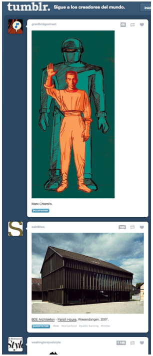
web communities containts small cores

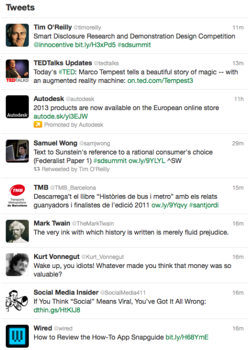


[Kumar et al., 1999]

motivation – social piggybacking

[Gionis et al., 2013]

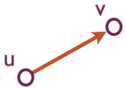




- event feeds: majority of activity in social networks

motivation – social piggybacking

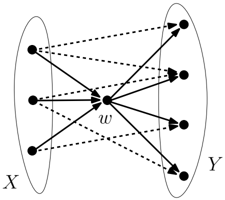
- system throughput proportional to the data transferred between data stores
- feed generation important component to optimize



- 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

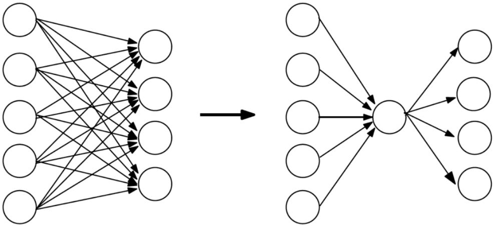
motivation – social piggybacking

- 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., 2014]
- many more

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

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

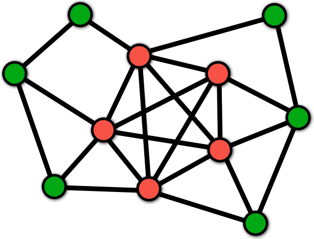
- degree of a node $u \in V$ is $\deg(u) = \deg_V(u)$
- edges between $S \subseteq V$ and $T \subseteq V$ are

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

  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 $E(S, \bar{S}) = E(S, V \setminus S)$
- 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 $E(S) \geq \alpha \binom{|S|}{2}$

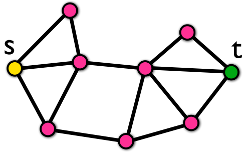
and more

not considered here

- 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

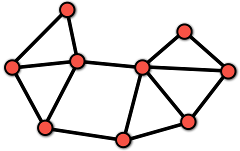
reminder: min-cut and max-cut problems

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 $e(S, \bar{S})$

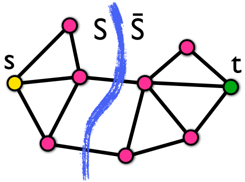
max-cut problem



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

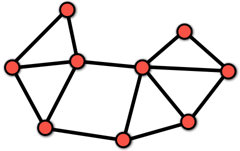
reminder: min-cut and max-cut problems

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

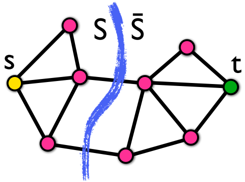
max-cut problem



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

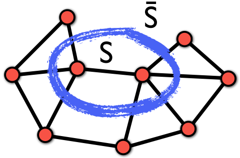
reminder: min-cut and max-cut problems

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

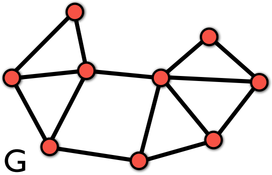


- find $S \subseteq V$, s.t.,
- maximize $e(S, \bar{S})$
- **NP**-hard
- approximation algorithms
  (0.868 based on SDP)

basic algorithms

Goldberg's algorithm for densest subgraph

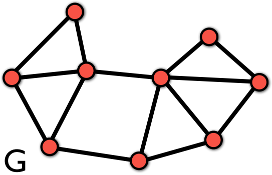
- consider first degree density d



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

Goldberg's algorithm for densest subgraph

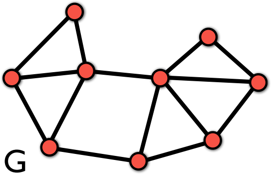
- consider first degree density d



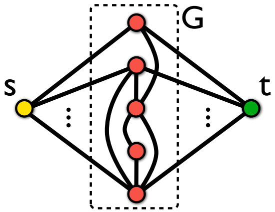
- is there a subgraph S with $d(S) \geq c$?
- transform to a min-cut instance

Goldberg's algorithm for densest subgraph

- consider first degree density d



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



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

Goldberg's algorithm for densest subgraph

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

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

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

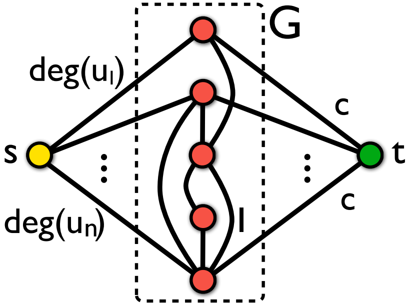
$$\sum_{u \in S} \deg(u) - |E(S,\bar{S})| \geq c|S|$$

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

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

Goldberg's algorithm for densest subgraph

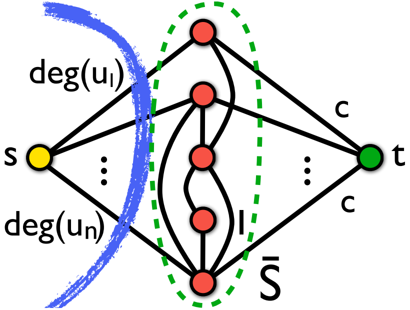
- transformation to min-cut instance



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

Goldberg's algorithm for densest subgraph

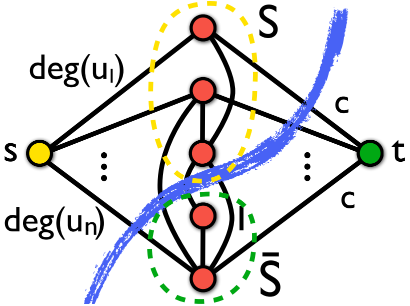
- transform to a min-cut instance



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

Goldberg's algorithm for densest subgraph

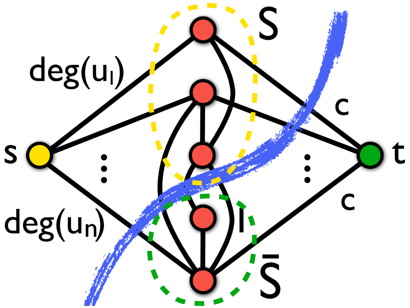
- transform to a min-cut instance



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

Goldberg's algorithm for densest subgraph

- transform to a min-cut instance



- is there S s.t. $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2\,|E|$?
- YES, if min cut achieved for $S \neq \emptyset$

Goldberg's algorithm for densest subgraph

[Goldberg, 1984]

input: undirected graph $G = (V, E)$, number c
output: S, if $d(S) \geq c$
1 transform G into min-cut instance $G' = (V \cup \{s\} \cup \{t\}, E', w')$
2 find min cut $\{s\} \cup S$ on G'
3 if $S \neq \emptyset$ return S
4 else return NO

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

densest subgraph problem – discussion

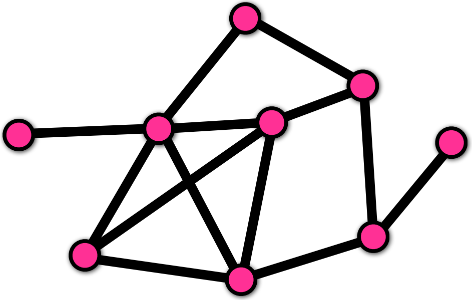
- Goldberg's algorithm polynomial algorithm, but
- $\mathcal{O}(nm)$ time for one min-cut computation
- not scalable for large graphs (millions of vertices / edges)

densest subgraph problem – discussion

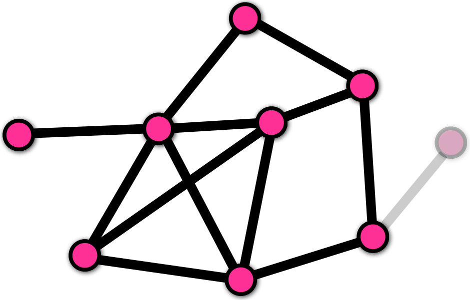
- Goldberg's algorithm polynomial algorithm, but
- $\mathcal{O}(nm)$ time for one min-cut computation
- not scalable for large graphs (millions of vertices / edges)

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

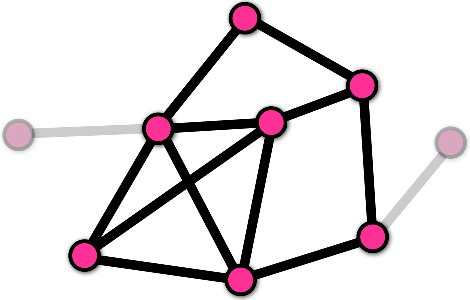
greedy algorithm for densest subgraph — example



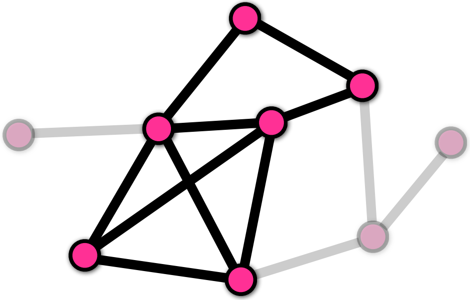
greedy algorithm for densest subgraph — example



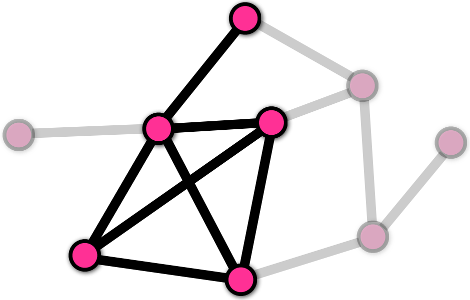
greedy algorithm for densest subgraph — example



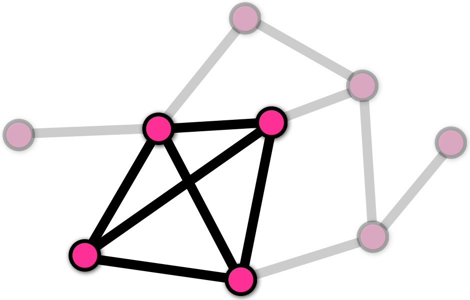
greedy algorithm for densest subgraph — example



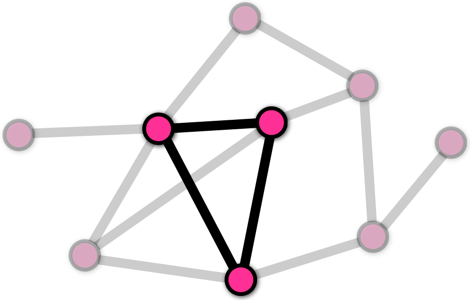
greedy algorithm for densest subgraph — example



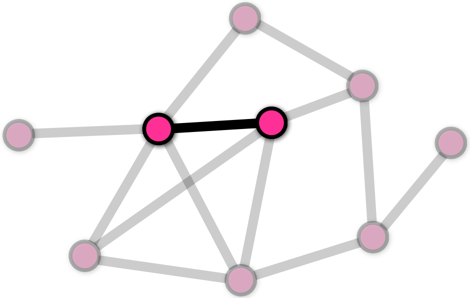
greedy algorithm for densest subgraph — example



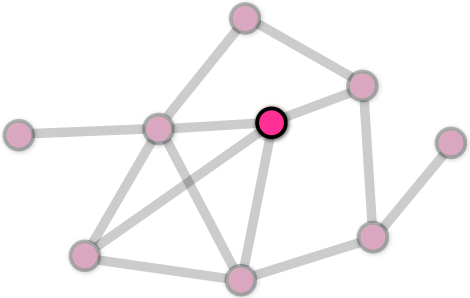
greedy algorithm for densest subgraph — example



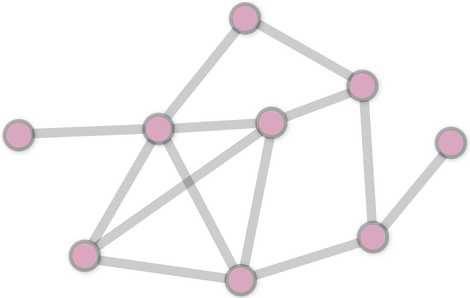
greedy algorithm for densest subgraph — example



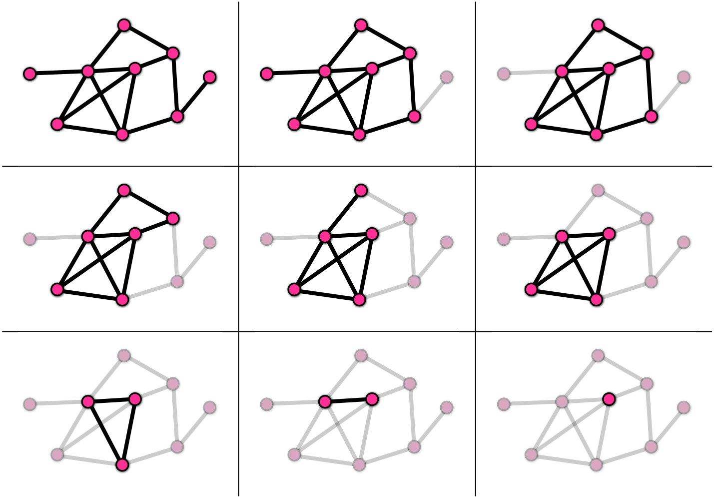
greedy algorithm for densest subgraph — example



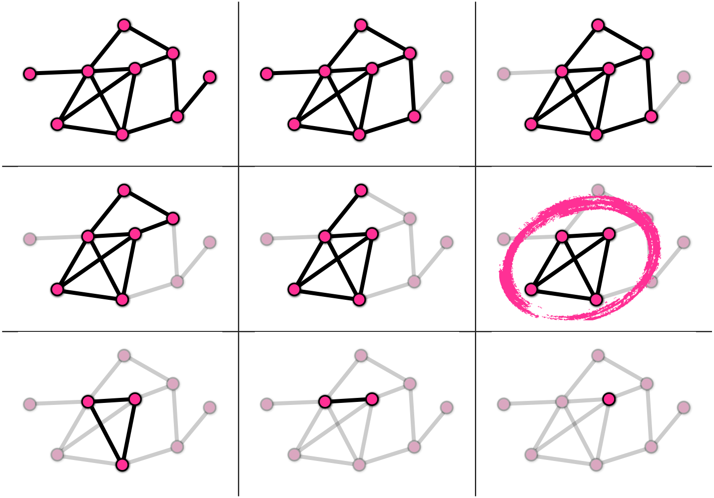
greedy algorithm for densest subgraph — example



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 $\lambda/2$

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 works for weighted graphs

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

- things are not as straightforward for directed graphs

finding dense subgraphs on 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 $\frac{|S|}{|T|} = c$

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

directed graphs – algorithms

[Charikar, 2000]

- for a given value of $\frac{|S|}{|T|} = c$ the $\text{LP}(c)$ has an integral solution

- it can be shown that

$$\max_{S,T \subseteq V} d(S,T) = \max_{c} \text{OPT}(\text{LP}(c))$$

[proof sketch]

1. for $S, T \subseteq V$, with $\frac{|S|}{|T|} = c$ the optimal value of $\text{LP}(c)$ is at least $d(S,T)$
2. given a feasible solution of $\text{LP}(c)$ with value v we can construct $S, T \subseteq V$ such that $d(S,T) \geq v$

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 $T \leftarrow T \setminus \{j_{\min}\}$

- 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 edge of capacity m from s to each vertex of V_1 and V_2
- add edge of capacity $2m + \frac{g}{\sqrt{\alpha}}$ from each vertex of V_1 to t
- add edge from each vertex j of V_2 to sink t of capacity

$$2m + \sqrt{\alpha} g - 2\text{deg}(j)$$

- for each $(i, j) \in E(G)$, add an edge from $j \in V_2$ to $i \in V_1$ with capacity 2

dense subgraphs on directed graphs – exact

- proof of correctness of min-cut algorithm of transformed graph G' follows the argument of Goldberg
- the cut $(\{s\}, \{t, V_1, V_2\})$ has weight $m(|V_1| + |V_2|)$
- thus, min cut has weight at most $m(|V_1| + |V_2|)$
- it can be shown that solution to the min-cut with value smaller than $m(|V_1| + |V_2|)$ corresponds to sets $S \subseteq V_1$, $T \subseteq V_2$ with density $d(S, T)$ greater than g
- densest subgraph can be found with binary search on g
- one min-cut computation suffices (using parametric max-flow algorithm)

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

k-clique densest subgraphs

motivating question

- how to go beyond edge density?
- how to search for large near-cliques

- can we combine the best of both worlds, namely
  - have poly-time solvable formulation(s) which
  - …succeeds in finding large near-cliques?

- 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)

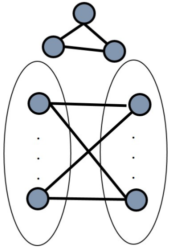
given $G(V, E)$, find a subset of vertices S^* such that $\rho_k(S^*) = \rho_k^* = \max_{S \subseteq V} \rho_k(S)$

- 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?
- an 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 time $\mathcal{O}\left(m^{3/2} + nt + \min(n,t)^3\right)$ *where* t *is the number of triangles in the graph*

Theorem

the k*-clique DSP can be solved in polynomial time for any* $k = \Theta(1)$

- although this construction solves also the (2-clique) DSP Goldberg's algorithm is more efficient

triangle densest subgraph problem

exact algorithm

- once again, follow Goldberg's idea
- perform binary searches:

– is there a set $S \subseteq V$ such that $t(S) > \alpha |S|$?

- $\mathcal{O}(\log n)$ queries suffice to solve TDSP (why?)

– any two distinct triangle density values are at least $\mathcal{O}(1/n^2)$ away from each other

– for the optimal density $0 \leq \frac{t}{n} \leq \tau^* \leq \frac{\binom{n}{3}}{n}$

- but what does a binary search correspond to ? ...

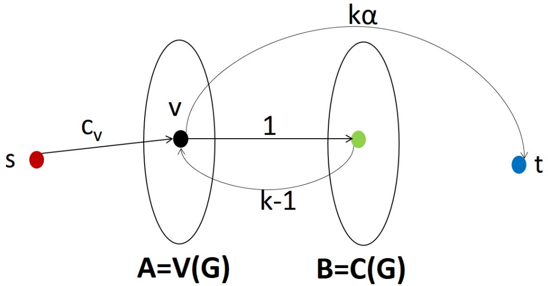
triangle densest subgraph problem

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
- 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

construction for $k = \Theta(1)$



triangle densest subgraph problem

exact algorithm for TDSP

1. list the set of triangles $\mathcal{T}(G)$, $t = |\mathcal{T}(G)|$
2. $l \leftarrow \frac{t}{n}, u \leftarrow \frac{(n-1)(n-2)}{6}$
3. $S^* \leftarrow \emptyset$
4. 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 $S^* \leftarrow \left(S \backslash \{s\}\right) \cap V(G)$ and $l \leftarrow \alpha$
5. return S^*

- run time: $\mathcal{O}\left(m^{3/2} + \left(nt + \min\left(n, t\right)^3\right) \log n\right)$
- space complexity: $\mathcal{O}(n + t)$ (typically $n \ll t$)

triangle densest subgraph problem

greedy works too

1. set $G_n \leftarrow G$
2. for $k \leftarrow n$ downto 1
   - let v be the smallest triangle count vertex in G_k
   - $G_{k-1} \leftarrow G_k \setminus \{v\}$
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	10.66
	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 algorithms find the same solution as optimal exact methods
- observation 2 : the TDS is closer to being a large near-clique compared to the DS

remark

- in many cases, despite being a 2-approximation, the greedy performs optimally or close to optimally
- evidence that real-data are “far away” from adversarial

- however, 2-approximation bound is tight
- consider $G = G_1 \cup G_2$ where $G_1 = K_{d,D}$ and G_2 is the disjoint union of D cliques, each of size $d + 1$
- let $d \ll D$
- how does the greedy algorithm perform?
- optimal is bipartite clique with density $dD/(d + D) \approx d$
- greedy returns a clique of size $d + 1$ with density $d/2$

datasets

non-bipartite

dataset	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

bipartite

dataset	n	m
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
■	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

finding densest subgraphs with map-reduce

peeling in batches

the following algorithm due to Bahmani, Kumar and Vassilvitski 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: previous algorithm is a $2(1+\epsilon)$ approximation
  furthermore, it returns after $\mathcal{O}(\log_{1+\epsilon}(n))$ rounds

- **Proof**

- approximation guarantee

– fix an optimal solution S^*

– consider the first round when a node $v \in S^*$ is removed

– let U be the set of vertices at that point

– then, $\rho^* \leq d_{S^*}(v) \leq d_U(v) \leq (2+2\epsilon)\rho(U)$

- number of rounds is $\mathcal{O}(\log_{1+\epsilon}(n))$

– in each round we throw a constant fraction of the vertices
  $2\,E(S) > \sum_{v \notin A(S)} d_S(v) > (|S| - |A(S)|)2(1+\epsilon)\rho(S)$
  and thus $|A(S)| > \frac{\epsilon}{1+\epsilon}|S|$

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 $\mathbf{P} = \mathbf{NP}$
- Feige et al. 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 a $\mathcal{O}(n^{\frac{1}{4}+\epsilon})$ approximation guarantee for any $\epsilon > 0$ [Bhaskara et al., 2010]
- closing the gap between lower and upper bounds is a significant open problem

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

an alternative density definition

edge-surplus framework

[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
- the partitioning version led to a streaming balanced graph-partitioning algorithm: FENNEL
  - goal: maximize $g(\mathcal{P})$ over all possible k-partitions where

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

  - 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]

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

top-k dense subgraphs with limited overlap

alternative problem formulation

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

$$r(S_1, \ldots, S_k) = \sum_{i=1}^{k} d(S_i) + \lambda \sum_{i,j} \mathrm{dist}(S_i, S_j)$$

- fits the max-sum diversification framework [Borodin et al., 2012]
- possible to obtain an approximation guarantee ($1/10$)

top-k dense subgraphs with limited overlap

- want to maximize

$$r(S_1, \ldots, S_k) = \sum_{i=1}^{k} d(S_i) + \lambda \sum_{i,j} \text{dist}(S_i, S_j)$$

- need to define a distance between subgraphs
- define

$$\text{dist}(S_i, S_j) = \begin{cases} 2 - \frac{|S_i \cap S_j|^2}{|S_i|\,|S_j|} & \text{if } S_i \neq S_j \\ 0 & \text{otherwise} \end{cases}$$

- distance $\text{dist}(S_i, S_j)$ is a metric function
- we can obtain an approximation guarantee ($1/10$)

top-k dense subgraphs with limited overlap

adapting the max-sum diversification framework

Algorithm 1: DOS; Algorithm for finding top-k overlapping densest subgraphs (problem Dense-Overlapping-Subgraphs)

Input: $G = (V, E), \lambda, k$
Output: set of subgraphs $\mathcal{W}$ s.t. $|\mathcal{W}| = k$ and maximizing $r(\mathcal{W})$

$\mathcal{W} \leftarrow \emptyset$;
foreach $i = 1, \ldots, k$ **do** $\mathcal{W} \leftarrow \mathcal{W} \cup \mathsf{Peel}(G, \mathcal{W}, \lambda)$;
return $\mathcal{W}$;

top-k dense subgraphs with limited overlap

adapting the max-sum diversification framework

Algorithm 2: Peel; finds a dense subgraph U of the graph G, overlapping with a collection of previously discovered subgraphs $\mathcal{W}$.

```
Input: G = (V, E), 𝒲, λ
Output: U maximizing χ(U; 𝒲)
V_n ← V;
foreach i = n, ..., 2 do
    v ← arg min_v { deg(v; V_i) − 4λ Σ_{W_j ∋ v} |V_i ∩ W_j| / |W_j| };
    V_{i−1} ← V_i \ {v};
foreach i = 1, ..., n do
    if V_i ∈ 𝒲 then V_i ← Modify(V_i, G, 𝒲, λ);
return arg max_{V_j} {χ(V_j; 𝒲)};
```

top-k dense subgraphs with limited overlap

adapting the max-sum diversification framework

Algorithm 3: Modify; modifies U if $U \in \mathcal{W}$

Input: $U, G, \mathcal{W}, \lambda$
Output: modified U

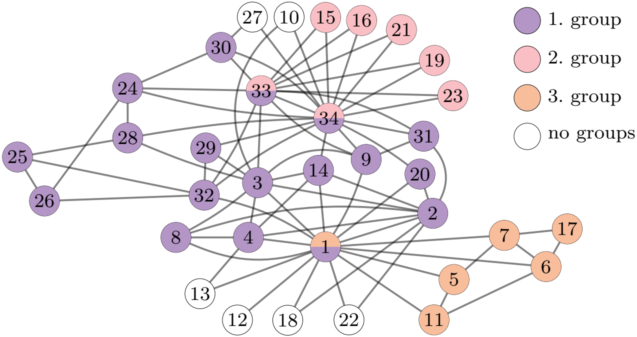
$X \leftarrow \{U \cup \{x\} \mid x \notin U,\ U \cup \{x\} \notin \mathcal{W}\}$;
$Y \leftarrow \{U \setminus \{y\} \mid y \in U,\ U \setminus \{y\} \notin \mathcal{W}\}$;
if $X = \emptyset$ **and** $\mathrm{dens}(U) \leq 5/3$ **then**
    $U \leftarrow$ {a wedge of size 3 not in $\mathcal{W}$};
else
    $U \leftarrow \arg\max_{C \in X \cup Y} \{\chi(C; \mathcal{W})\}$;
return U;

---

top-k dense subgraphs with limited overlap

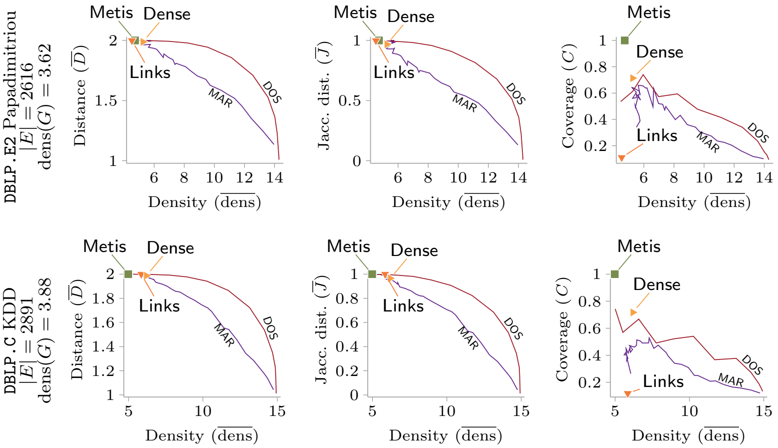
adapting the max-sum diversification framework

example



top-k dense subgraphs with limited overlap

DOS vs. MAR



core decomposition

k-core

- (recall) S is a *k*-core if every vertex in S is connected to at least k other vertices in S
- can be found with the following algorithm:
  1. while (*k*-core property is satisfied)
  2.   remove all vertices with degree less than k
- can also obtain all *k*-cores (for all k)
- all *k*-cores form a nested sequence of subgraphs (*k*-core shell decomposition)
- popular technique in social network analysis
- inner cores : more dense, more central vertices
- note resemblance with Charikar's algorithm

k-core decomposition

widely used technique for partitioning graphs

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

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

k-shell $=$ vertices in k-core but not in $(k+1)$-core

k-core decomposition

widely used technique for partitioning graphs

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

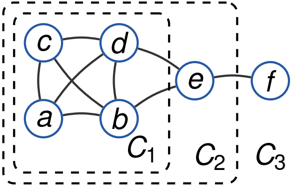
cores form a chain, k-core $\subseteq (k-1)$-core; let

k-shell $=$ vertices in k-core but not in $(k+1)$-core

algorithm to find shells:

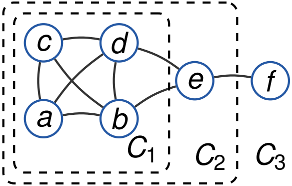
1. **while** G is not empty
2. $\quad v \leftarrow$ vertex with the smallest degree
3. $\quad$ assign v to k-shell
4. $\quad$ remove v from G

core decomposition and density are not compatible

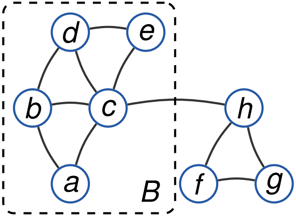


$$d(C_1) = \frac{6}{4} < \frac{8}{5} = d(C_2)$$

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

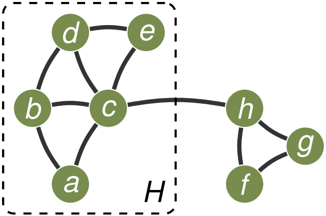
formally, define augmented density

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

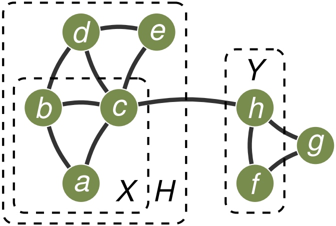
subgraph H is locally-dense if

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

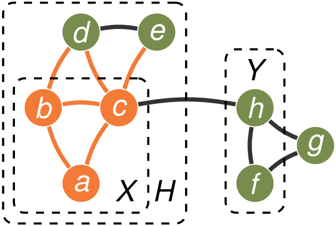
example



example

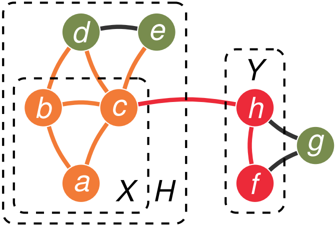


example



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

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}

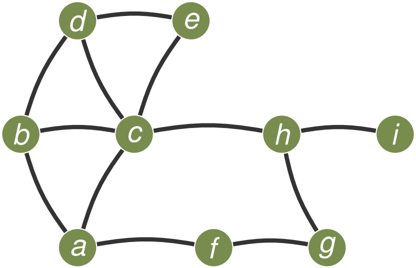
$$B_1 = \text{densest subgraph}$$

$$B_2 = \arg\max_{B \supsetneq B_1} d(B \setminus B_1, B_1)$$

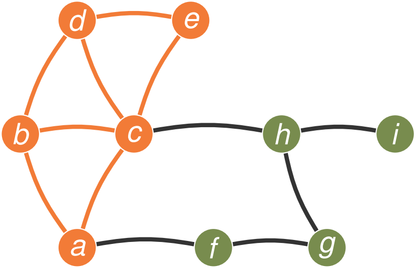
$$\cdots$$

$$B_i = \arg\max_{B \supsetneq B_{i-1}} d(B \setminus B_{i-1}, B_{i-1})$$

first approach to compute the subgraphs

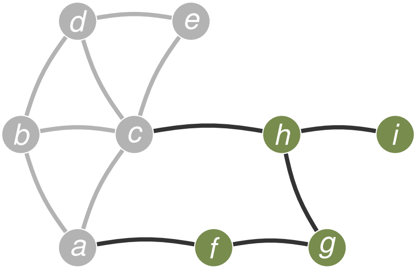


first approach to compute the subgraphs



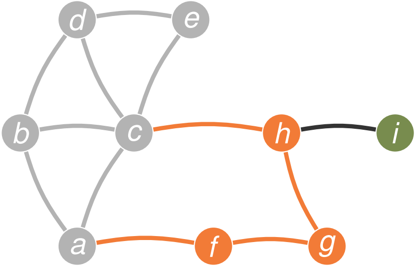
find B_1

first approach to compute the subgraphs



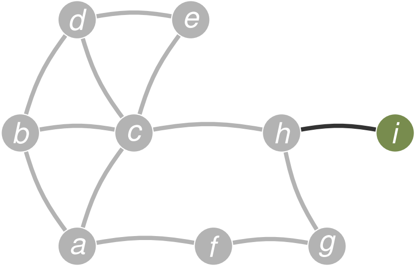
find B_1
delete B_1

first approach to compute the subgraphs



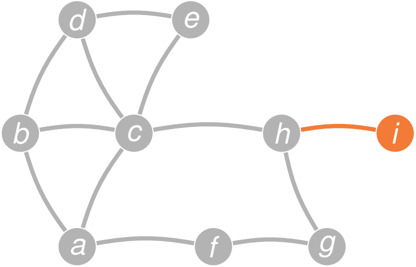
find B_1
delete B_1
find B_2

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

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: 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. EXACT(X, Z)
3. 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:

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

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

$\ldots$

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

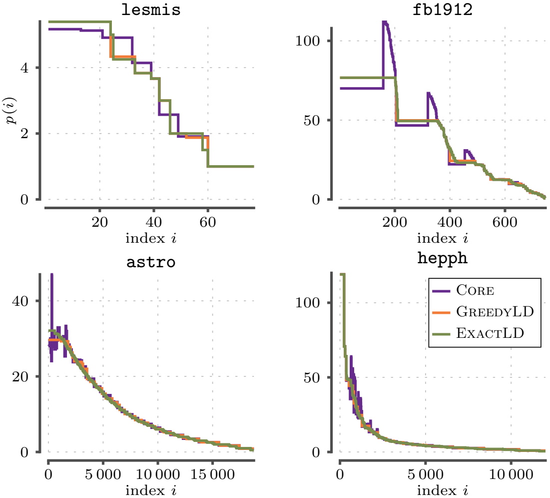
$$p(i;\mathcal{C}) \geq p(i;\mathcal{B})/2, \quad \text{for every } \; i$$

for $i = 1$, this implies

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

experiments

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

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

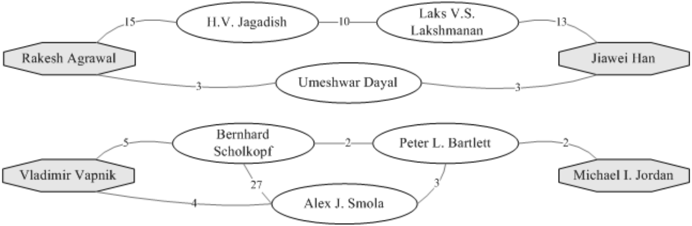
- $r(Q, j)$ goodness score of j wrt Q

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

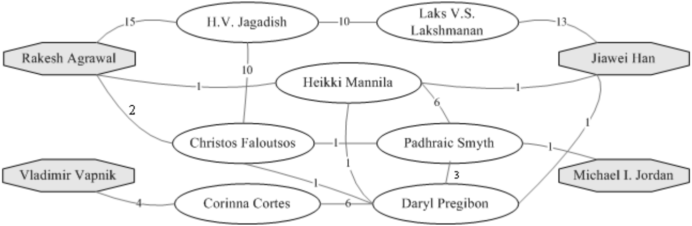
- proposed algorithm:

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”: $k = 2$



(b) “AND query”

[Tong and Faloutsos, 2006]

the community-search problem

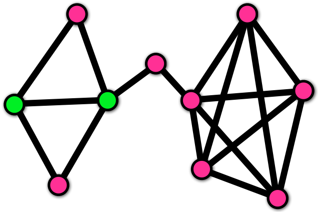
- given: graph $G = (V, E)$ and set of query vertices $Q \subseteq V$
- find: a connected subgraph H that
  (a) contains Q
  (b) optimizes a density function $d(H)$
  (c) possibly other constraints

- 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)

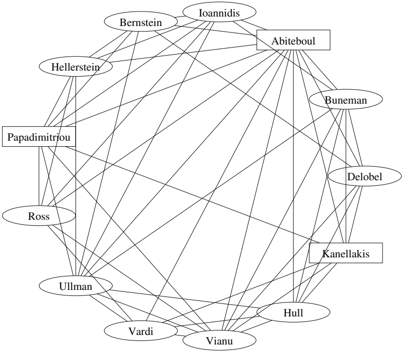
baseline: increamental addition of vertices

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

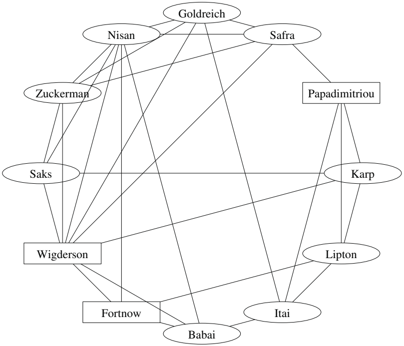
example result in DBLP

- 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

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., 2014]

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

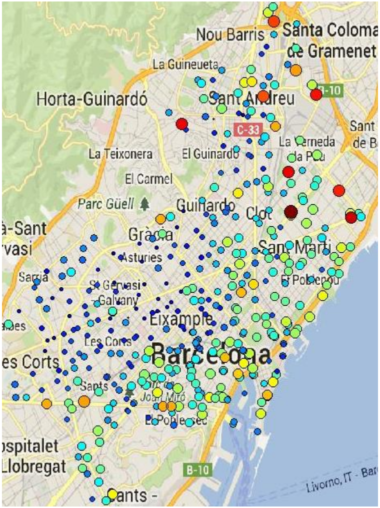
- finding events in networks
- vertices correspond to locations
- weights model activity recorded in locations
- distances between locations
- find compact regions (neighborhoods) with high activity

event detection

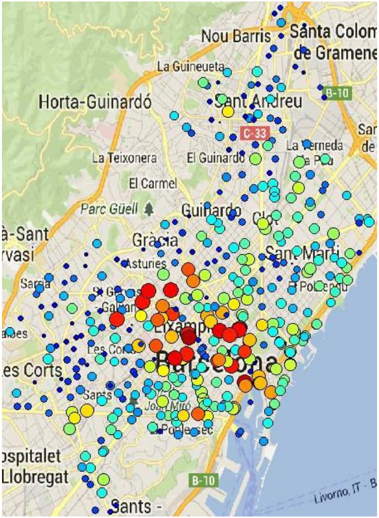
- sensor networks and traffic measurements

event detection

15.11.2012
ordinary day, no events

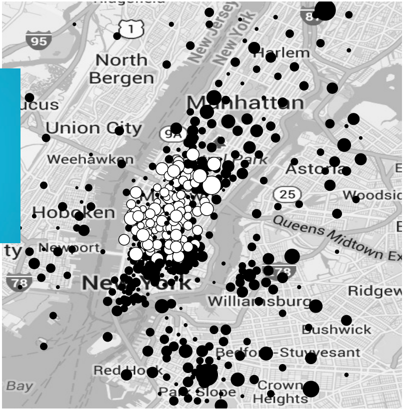


11.09.2012
Catalunya national day



event detection

- location-based social networks



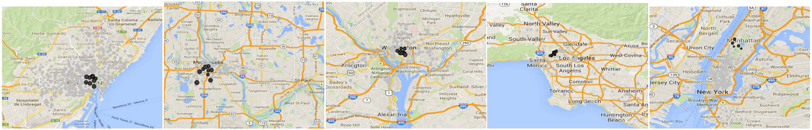
discovering heavy subgraphs

- 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., 2014]

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

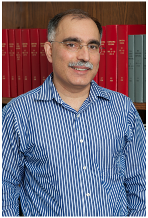
Figure 4: Public holiday city-events discovered using the SDP algorithm.

(a) 01.06.12 Primavera sound music festival
(b) 18.09.12 festival of the Poblenou neighborhood
(c) 31.10.12 Halloween

summary

- the problem of finding dense subgraphs has many different real-world applications
- a number of density measures have been studied
- problem complexity depends on adopted measure
- for some problem formulations there are exact polynomial and faster approximate solution
- a number of different techniques has been used
  min-cut, greedy, submodularity optimization
- many directions and open problems for future work

acknowledgements



Shamir Khuller

Renato Werneck

Nikolaj Tatti

Charalampos Tsourakakis

references

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*.

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.

references (cont.)

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.

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 (cont.)

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.

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.

Borodin, A., Lee, H. C., and Ye, Y. (2012).
Max-sum diversification, monotone submodular functions and dynamic updates.
In *Proceedings of the 31st symposium on Principles of Database Systems*, pages 155–166. ACM.

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

references (cont.)

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*.

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.

references (cont.)

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.

Eppstein, D., Löffler, M., and Strash, D. (2010).
Listing all maximal cliques in sparse graphs in near-optimal time.
In *ISAAC*.

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.

references (cont.)

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

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

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).

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.

references (cont.)

- 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.
- Kang, U., Tsourakakis, C. E., and Faloutsos, C. (2009).
  Pegasus: A peta-scale graph mining system implementation and observations.
  In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 229–238. IEEE.
- Kannan, R. and Vinay, V. (1999).
  Analyzing the structure of large graphs.
  Rheinische Friedrich-Wilhelms-Universität Bonn.
- Karande, C., Chellapilla, K., and Andersen, R. (2009).
  Speeding up algorithms on compressed web graphs.
  Internet Mathematics, 6(3):373–398.

references (cont.)

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*.

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.

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

references (cont.)

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.

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. (2014).
Event detection in activity networks.
In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Saha, B., Hoch, A., Khuller, S., Raschid, L., and Zhang, X.-N. (2010).
Dense subgraphs with restrictions and applications to gene annotation graphs.
In *Research in Computational Molecular Biology*, pages 456–472. Springer.

references (cont.)

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*.

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.

references (cont.)

- 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*.

- 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.