Dense-subgraph discovery

Aristides Gionis

Department of Information and Computer Science
Aalto University

aristides.gionis@aalto.fi

School for advanced sciences of Luchon
Network analysis and applications
June 26, 2014
what this lecture is about . . .

given a graph (network)
(social network, biological network, information network, commodity 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

- introduction and motivation
- density functions
- complexity of basic problems
- basic algorithms
- variants of the densest-subgraph problem
- biclique mining, trawling, graph shingling

- finding heavy subgraphs
- centerpiece problems
community detection in graphs and social networks

a small network with clear community structure
community structure vs. dense subgraphs

informal definition of community: a set of vertices
- densely connected to each other, and
- sparsely connected to the rest of the graph

- dense subgraphs: set of vertices with many edges
- no requirement for small cuts
- key primitive for detecting communities, but not identical problem
one motivating application — social piggybacking

[Serafini et al., 2013]

event feeds: majority of activity in social networks
one motivating application — 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
one motivating application — social piggybacking

- **hub optimization** turns out to be a good idea
- depends on finding **dense subgraphs**
other applications of finding dense subgraphs

- communities and spam link farms [Kumar et al., 1999]
- graph visualization [Alvarez-Hamelin et al., 2005]
- real-time story identification [Angel et al., 2012]
- regularoty motif detection in DNA [Fratkin et al., 2006]
- finding correlated genes [Zhang and Horvath, 2005]
- epilepsy prediction [Iasemidis et al., 2003]
- many more ...
notation

- undirected graph $G = (V, E)$ defined with vertex set $V$ and edge set $E \subseteq V \times V$

- degree of a node $u \in V$ is

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

- 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):

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

► edge ratio:

\[ \delta(S) = \frac{|E(S, S)|}{\left(\frac{|S|}{2}\right)} = \frac{|E(S)|}{\left(\frac{|S|}{2}\right)} = \frac{2 |E(S)|}{|S|(|S| - 1)} \]

► triangle density:

\[ t(S) = \frac{|T(S)|}{|S|} \]

► triangle ratio:

\[ \tau(S) = \frac{|T(S)|}{\left(\frac{|S|}{3}\right)} \]
other density measures

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

- **$\alpha$-quasiclique**: the set $S$ has at least $\alpha \left( \frac{|S|}{2} \right)$ edges
  
i.e., $S$ is $\alpha$-quasiclique if $E(S) \geq \alpha \left( \frac{|S|}{2} \right)$
not considered (directly) in this tutorial

- **K-cliques**: a subset of vertices of distance at most $k$ to each other
  - 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
the general densest-subgraph problem

- given an undirected graph $G = (V, E)$
- and a density measure $f : 2^V \rightarrow \mathbb{R}$
- find set of vertices $S \subseteq V$
- that maximizes $f(S)$
complexity of density problems — clique

- find the max-size clique in a graph: \textbf{NP}-hard problem

- strong inapproximability result:

  for any $\epsilon > 0$, there cannot be a polynomial-time algorithm that approximates the maximum clique problem within a factor better than $\mathcal{O}(n^{1-\epsilon})$, unless $P = \text{NP}$

[Håstad, 1997]
finding dense subgraphs – which measure?

- find large cliques...
  - NP-hard problem
  - too strict requirement

- find $S$ that maximizes edge ratio $\delta(S) = |E(S)| / (\frac{|S|^2}{2})$
  - ill-defined problem ... pick a single edge
  - will consider later

- find $S$ that maximizes edge density $d(S) = 2 |E(S)| / |S|$
  - study in more detail next ...
The densest-subgraph problem

- Given an undirected graph \( G = (V, E) \)
- Find set of vertices \( S \subseteq V \)
- That maximizes the edge density (average degree)

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

- ... polynomial? ... \textbf{NP}-hard? ... approximations?
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)
Goldberg’s algorithm for densest subgraph

- 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

- transform to a min-cut instance

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

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

- \( S \neq \emptyset \) 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 \( \delta(S) \geq c \)
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 \( \text{NO} \)

...to find the densest subgraph binary search on \( c \)...
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

[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 $G_n, G_{n-1}, \ldots, G_1$
analysis of the greedy algorithm (I)

[Charikar, 2000]

- first, will upper bound the optimal solution
- consider any arbitrary assignment of edges \((u, v)\) to \(u\) or \(v\)

\[
\text{define } \quad \text{in}(u) = \#\{\text{edges assigned to } u\} \quad \text{and} \quad \Delta = \max_{u \in V} \{\text{in}(u)\}
\]

- claim 1: \(\max_{S \subseteq V} \{d(S)\} \leq 2 \Delta\)

proof: consider the set \(S\) that maximizes \(d(S)\)

\[
|e(S)| = \sum_{u \in S} \text{in}(u) \leq |S| \Delta, \quad \text{so} \quad d(S) = \frac{2 |e(S)|}{|S|} \leq 2 \Delta
\]
consider assignment defined \textit{dynamically} during greedy

initially all edges \textit{are unassigned}

in each step, edges \textit{are assigned to the deleted vertex}

in the end, all edges \textit{have been assigned}

let $z$ be the maximum $d(S)$ achieved by greedy

\textbf{claim 2}: $\Delta \leq z$

\textbf{proof}: consider a single iteration of the greedy

$v^*$ is deleted in $S$

\[ \text{in}(v^*) \leq \{\text{average degree in } S\} = d(S) \leq z \]

it holds for all $v^*$, thus $\max_{v \in V}\{\text{in}(v)\} = \Delta \leq z$
analysis of the greedy algorithm (III)

- putting everything together
- claim 1: $\max_{S \subseteq V} \{d(S)\} \leq 2 \Delta$
- claim 2: $\Delta \leq z$, for $z$ the max $d(S)$ achieved by greedy
- it follows
  \[ z \geq \frac{1}{2} d(S^{\text{OPT}}) \]
- 2-approximation algorithm
the greedy algorithm

- factor-2 approximation algorithm
- for a polynomial problem . . . but faster and easier to implement than the exact algorithm

- running time:
  - naive implementation: $O(n^2)$
  - using heaps: $O(m + n \log n)$
  - also possible: $O(m + n)$ (how?)
densest subgraph on directed graphs

[Charikar, 2000]

- dense subgraphs on directed graphs:
  find sets $S, T \subseteq V$ to maximize

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

- problem can be solved exactly in polynomial time using linear programming (LP)
  - solution to LP can be transformed to integral solution of the same value

- greedy 2-approximation algorithm
  - similar “peel off” flavor as for the undirected case
  - iteratively removes min-degree vertices from $S$ or $T$ (depending on a certain condition)
size-constrained densest-subgraph problems

[Khuller and Saha, 2009]

- given an undirected graph $G = (V, E)$
- find set of vertices $S \subseteq V$
  that maximizes degree density $d(S)$
  and $S$ satisfies size constraints

$DkS$ “equality” constraint: $|S| = k$
$DAMkS$ “at most” constraint: $|S| \leq k$
$DALkS$ “at least” constraint: $|S| \geq k$
size-constrained densest-subgraph problems

- what about the complexity of $D_k S$, $D_{AMk} S$, $D_{ALk} S$?
- all $\textbf{NP}$-hard

$D_k S$ approximation guarantee $O(n^\alpha)$, $\alpha < \frac{1}{3}$

$D_{AMk} S$ as hard as $D_k S$

$D_{ALk} S$ factor-2 approximation guarantee

[Feige et al., 2001, Khuller and Saha, 2009]
(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 $k$

gives a $k$-core, as well as a $k$-core shell decomposition

index of a vertex: the iteration id it was deleted

more central vertices have higher index

popular technique in social network analysis

note resemblance with Charikar’s algorithm
recall our density measures

- edge density: \( d(S) = 2 \frac{|E(S)|}{|S|} \)
- edge ratio: \( \delta(S) = \frac{|E(S)|}{\left(\frac{|S|}{2}\right)} \)
- triangle density: \( t(S) = \frac{|T(S)|}{|S|} \)
- triangle ratio: \( \tau(S) = \frac{|T(S)|}{\left(\frac{|S|}{3}\right)} \)
- \( k \)-core: every vertex in \( S \) is connected to at least \( k \) other vertices in \( S \)
- \( \alpha \)-quasiclique: the set \( S \) has at least \( \alpha \left(\frac{|S|}{2}\right) \) edges
optimal quasicliques

- $S$ is $\alpha$-quasiclique if $|E(S)| \geq \alpha \left( \frac{|S|}{2} \right)$
- for $S \subseteq V$ define edge surplus
  
  $$f_a(S) = |E(S)| - \alpha \left( \frac{|S|}{2} \right)$$

the optimal quasiclique problem:

find $S \subseteq V$ that maximizes $f_a(S)$
densest subgraph vs. optimal quasi-clique

<table>
<thead>
<tr>
<th></th>
<th>densest subgraph</th>
<th>optimal quasi-clique</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\frac{</td>
<td>S</td>
</tr>
<tr>
<td>Dolphins</td>
<td>0.32</td>
<td>0.33</td>
</tr>
<tr>
<td>Football</td>
<td>1</td>
<td>0.09</td>
</tr>
<tr>
<td>Jazz</td>
<td>0.50</td>
<td>0.34</td>
</tr>
<tr>
<td>Celeg. N.</td>
<td>0.46</td>
<td>0.13</td>
</tr>
</tbody>
</table>

[Tsourakakis et al., 2013]
generalized edge-surplus framework

- for a set of vertices \( S \) define edge surplus

\[
f(S) = g(|E(S)|) - h(|S|)
\]

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

find \( S^* \) such that

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

- example 1: optimal quasiclques

\[
g(x) = x, \quad h(x) = \alpha \frac{x(x - 1)}{2}
\]
generalized edge-surplus framework

- edge surplus $f(S) = g(|E(S)|) - h(|S|)$

- example 2
  
  $g(x) = h(x) = \log x$

  find $S$ that maximizes $\log \frac{|E(S)|}{|S|}$

  densest-subgraph problem

- example 3
  
  $g(x) = x$, $h(x) = \begin{cases} 0 & \text{if } x = k \\ +\infty & \text{otherwise} \end{cases}$

  $k$-densest-subgraph problem ($DkS$)
generalized edge-surplus framework

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 solvable in polynomial time
algorithms for finding optimal quasicliques

- find $S \subseteq V$ that maximizes $f_a(S) = |E(S)| - \alpha \left(\frac{|S|}{2}\right)$
- approximation algorithms?
- edge surplus function can take negative values
- multiplicative approximation guarantee not meaningful
- can obtain guarantee for a **shifted version**
  but introduces large additive error
- other types of guarantees more appropriate
finding an optimal quasiclique

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]
practical considerations

1. further improve solution of greedy by local search

2. choice of $\alpha$ in practice?
   when confronted with two disconnected components, the measure should pick one of the two, instead of their union translates to $\alpha \geq \frac{1}{3}$
Table 3: Densest subgraphs extracted with Charikar's method vs. optimal quasi-cliques extracted with the proposed algorithm on real-world graphs.

Results on real graphs are shown in Table 3. We compare the densest subgraphs and quasi-cliques produced by the Goldberg's algorithm with those produced by the proposed algorithm using Charikar's method. The table below shows the edge density of densest opt. quasi-clique, angle density of quasi-cliques, and smaller diameter than 20 times, and less than one minute for the largest graphs (e.g., 59.27s for Wikipedia 2006/11).

<table>
<thead>
<tr>
<th>Graph</th>
<th>Densest subgraph</th>
<th>Quasi-clique</th>
<th>Diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-Google</td>
<td>0.33 2</td>
<td>0.19 2</td>
<td>2</td>
</tr>
<tr>
<td>AS-Skitter</td>
<td>0.53 2</td>
<td>0.45 2</td>
<td>4</td>
</tr>
<tr>
<td>Football</td>
<td>0.63 2</td>
<td>0.52 2</td>
<td>4</td>
</tr>
<tr>
<td>Jazz</td>
<td>0.73 2</td>
<td>0.61 2</td>
<td>4</td>
</tr>
<tr>
<td>Celegans M.</td>
<td>0.83 2</td>
<td>0.71 2</td>
<td>4</td>
</tr>
<tr>
<td>Celegans N.</td>
<td>0.93 2</td>
<td>0.89 2</td>
<td>4</td>
</tr>
<tr>
<td>Polbooks</td>
<td>1.03 2</td>
<td>0.95 2</td>
<td>4</td>
</tr>
<tr>
<td>Football</td>
<td>0.83 2</td>
<td>0.71 2</td>
<td>4</td>
</tr>
<tr>
<td>Jazz</td>
<td>0.93 2</td>
<td>0.89 2</td>
<td>4</td>
</tr>
<tr>
<td>Celegans M.</td>
<td>0.63 2</td>
<td>0.52 2</td>
<td>4</td>
</tr>
<tr>
<td>Celegans N.</td>
<td>0.33 2</td>
<td>0.19 2</td>
<td>2</td>
</tr>
</tbody>
</table>

Concerning the efficiency, all algorithms are linear in the number of edges of the graph. Charikar's and the proposed algorithm are somewhat slower than the Goldberg's method. Indeed, the edge density achieved by the proposed algorithms with Charikar's method is one order of magnitude larger than the triangle density of the densest subgraph.

Figure 1: Edge density and diameter of the top-3 densest subgraphs for AS-Skitter and Wikipedia 2006/11.
recall our density measures

- **edge density**: \( d(S) = \frac{2|E(S)|}{|S|} \)
- **edge ratio**: \( \delta(S) = \frac{|E(S)|}{\binom{|S|}{2}} \)
- **triangle density**: \( t(S) = \frac{|T(S)|}{|S|} \)
- **triangle ratio**: \( \tau(S) = \frac{|T(S)|}{\binom{|S|}{3}} \)
- **k-core**: every vertex in \( S \) is connected to at least \( k \) other vertices in \( S \)
- **\( \alpha \)-quasiclique**: the set \( S \) has at least \( \alpha \binom{|S|}{2} \) edges
the triangle-densest-subgraph problem

[Tsourakakis, 2014]

- given an undirected graph $G = (V, E)$
- find set of vertices $S \subseteq V$
- that maximizes the triangle density

\[
  t(S) = \frac{|T(S)|}{|S|}
\]

- ...polynomial? ...\textbf{NP}-hard? ...approximations?
the triangle-densest-subgraph problem

[Tsourakakis, 2014]

- complexity: polynomial
- two exact algorithms

1. transformation to max-flow

   as Goldberg’s algorithm, but more sophisticated construction

   running time: $\mathcal{O}(\ell(m, n) + nT)$

   where $\ell(m, n)$ triangle listing complexity

   (can be $n^3$, $m^{3/2}$, \ldots), and

   $T$ number of triangles in the graph

2. via supermodular function maximization
The triangle-densest-subgraph problem

[Tsourakakis, 2014]

- also adapt Charikar’s greedy algorithm:
  - iteratively remove the vertex that participates in least number of triangles
  - return the graph with maximum triangle density
- provides factor-3 approximation
the triangle-densest-subgraph problem – summary

[Tsourakakis, 2014]

- in practice, as with optimal quasi-cliques, the triangle-densest-subgraph problem provides **high quality** solutions
- small size, dense in all measures, near cliques
- formulation combines **best of both worlds**: polynomial complexity, good quality solutions
- exact algorithms are expensive but greedy is efficient
mining cliques and bi-cliques

- finding **large** cliques is **NP-hard** problem
- same for bi-cliques (cliques in bipartite graphs)
- ok, so what? . . . let’s see if there is something we can do
- **frequent pattern mining** is all about mining large cliques
reminder: frequent pattern mining

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

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

e.g., \{Iceman, Storm\} appear in 60% of issues
reminder: 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
frequent itemsets vs. dense subgraphs

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

(transaction data) ⇔ (binary data) ⇔ (bipartite graphs)
frequent itemsets vs. dense subgraphs

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

.transaction data ⇔ binary data ⇔ bipartite graphs

.frequent itemsets ⇔ bi-cliques
bi-cliques vs. tiles

- quality of itemsets measured by support require frequency ≥ support threshold

- another idea:
  measure itemset quality as \{itemset size\} \times \{support\}

  tile mining
  measure corresponds to the area of the tile
  or equivalently, number of edges of the bi-clique

- algorithmically: not monotone measure, developed branch-and-bound technique to mine all tiles
frequent itemsets vs. dense subgraphs — discussion

- ideas from frequent itemset mining can be used for finding (bi-)cliques

+ wealth of efficient and highly-optimized algorithms typically uses the concept of support

- not easily adapted for near cliques or other dense subgraphs

  paradigm of enumerating all “large enough” cliques, not for finding the maximum clique
application to 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
dense communities contain small cores

[Kumar et al., 1999]
dense communities contain small cores

[Kumar et al., 1999]
finding web communities

trawling algorithm [Kumar et al., 1999]

1. iterative pruning: when searching for \((a, b)\)-cores vertices with outdegree less than \(a\) can be pruned same for vertices with indegree less than \(b\)

2. inclusion-exclusion pruning: exclude a page or output an \((a, b)\)-core

3. enumeration: after pruning graph has been reduced apply exact enumeration, e.g., Apriori
Inclusive-exclusive pruning

- Consider $u$ with outdegree exactly $a$
- Consider neighbors $N(u)$
- If exist $a - 1$ vertices pointing $N(u)$ then output core
  else eliminate $u$
think what trawling achieves:
find \( u_1, \ldots, u_k \) s.t. \( N(u_1), \ldots, N(u_k) \) have large intersection

somewhat easier problem: \( N(u_1), \ldots, N(u_k) \) are similar
measuring set similarity using the Jaccard coefficient

\[
J(A, B) = \frac{|A \cap B|}{|A \cup B|}
\]
locate similar items via locality-sensitive hashing

design a family of hash function, so that similar items have high probability of collision

for sets hashing based on min-wise independent permutations

[Broder et al., 1997]
min-wise independent permutations

- sets over a universe $U$
- measuring set similarity using the Jaccard coefficient
- $\pi : U \rightarrow U$ a random permutation of $U$
- $h(A) = \min\{\pi(x) \mid x \in A\}$
- then
  \[
  \Pr[h(A) = h(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}
  \]

- amplify the probability:
  - concatenate many hashes into sketches
  - repeat many times
  - consider objects similar if they collide in at least one sketch

- min-wise independent functions are expensive in practice, universal hash functions work well
probability amplification

concatenate $k$ hashes, repeat $\ell$ times

$$
\Pr[\text{sketches of } A \text{ and } B \text{ collide}] = 1 - (1 - J(A, B)^k)\ell
$$
discovering heavy subgraphs

- given a graph $G = (V, E, d, w)$
  - with a distance function $d : E \rightarrow \mathbb{R}$ on edges
  - and weights on vertices $w : V \rightarrow \mathbb{R}$

- 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]
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
discover heavy subgraphs

- maximize $Q(S) = \lambda W(S) - D(S)$
- objective can be 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

Figure 5: Top-3 diverse events discovered from Barcelona bicing data using the SDP algorithm.
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 ambiguity
- 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**:
  - **$k_{\text{softAND}}$**: 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 \( j \) to 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
Thus, we define the center-piece subgraph problem, as follows:

Problem 1. Center-Piece Subgraph Discovery (CEPS)

Given: an edge-weighted undirected graph $W$, $Q$ nodes as source queries $Q = \{q_i\}$ ($i=1,\ldots,Q$), the softAND coefficient $k$ and an integer budget $b$ Find: as a connected subgraph $H$ that (a) contains all query nodes (b) at most $b$ other vertices and (c) it maximizes a "goodness" function $g(H)$.

Allowing $Q$ query nodes creates a subtle problem: do we want the qualifying nodes to have strong ties to all the query nodes? to at least one? to at least a few? We handle all of the above cases with our proposed $K_{softAND}$ queries.

Figure 1(a) illustrates the case where we want intermediate nodes with good connections to at least $k=2$ of the query nodes. Notice that the resulting subgraph is much different now: there are two disconnected components, reflecting the two sub-communities (databases/statistics).

The contributions of this work are the following

• The problem definition, for arbitrary number $Q$ of query nodes, with careful handling of a lot of the subtleties.
• The introduction and handling of $K_{softAND}$ queries.
• EXTRACT, an over subgraph extraction algorithm.
• The design of a fast, approximate method, which provides a 6:1 speedup with little loss of accuracy.

The system is operational, with careful design and numerous optimizations, like alternative normalizations of the adjacency matrix, a fast algorithm to compute the scores for $K_{softAND}$ queries.

Our experiments on a large real dataset (DBLP) show that our method returns results that agree with our intuition, and that it can be made fast (a few seconds response time), while retaining most of the accuracy (about 90%).

The rest of the paper is organized as follows: in Section 2, we review some related work; Section 3 provides an overview of the proposed method: CEPS. The goodness calculation is proposed in Section 4 and its variants are presented in the Appendix. The "EXTRACT" algorithm and the speeding up strategy are provided in Section 5 and Section 6, respectively. We present experimental results in Section 7; and conclude the paper in Section 8.

2. RELATED WORK

In recent years, there is increasing research interest in large graph mining, such as pattern and law mining [2][5][7][20], frequent substructure discovery [27], influence propagation [18], community mining [9][11][12] and so on. Here, we make a brief review of the related work, which can be categorized into four groups: 1) measuring the goodness of connection; 2) community mining; 3) random walk and electricity related methods; 4) graph partition.

The goodness of connection. Defining a goodness criterion is the core for center-piece subgraph discovery. The two most natural measures for "good" paths are shortest distance and maximum flow. However, as pointed out in [6], both measurements might fail to capture some preferred characteristics for social network. The goodness function for survivable network [13], which is the count of edge-disjoint or vertex-disjoint paths from source to destination, also fails to adequately model social relationship. A more related distance function is proposed in [19][23]. However, it cannot describe the multi-faceted relationship in social network since center-piece subgraph aims to discover collection of paths rather than a single path.

In [6], the authors propose a delivered current based method. By interpreting the graph as an electric network, applying +1 voltage to one query node and setting the other query node 0 voltage, their method proposes to choose the subgraph which delivers maximum current between the query nodes. In [25], the authors further apply the delivered current based method to multi-relational graph. However, the delivered current criterion can only deal with pairwise source

Research Track Paper

(a) “K_{softAND}query”: $k = 2$

(b) “AND query”
the team-formation problem

[Lappas et al., 2009]

- users in social network have skills
- find a team to accomplish a task, e.g., task $T = \{x, z\}$
the team-formation problem

[Lappas et al., 2009]

- users in social network have skills
- find a team to accomplish a task, e.g., task $T = \{x, z\}$
the team-formation problem

[Lappas et al., 2009]

- users in social network have skills
- find a team to accomplish a task, e.g., task $T = \{x, z\}$
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 \( H \)
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: incremental 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

$f$ 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 $f_i$
conclusions

summary
- many applications finding dense subgraphs
- different density measures
- different problem formulations
- polynomial-time solvable or \textbf{NP}-hard problems
- choice of density measure matters

promising future directions
- room for new concepts
- better algorithms for upper-bound constraints
- top-\textit{k} versions of dense subgraphs
- formulations for enriched graphs (labels or attributes)
- local algorithms
references

Large scale networks fingerprinting and visualization using the \( k \)-core decomposition.
In *NIPS*.

Dense Subgraph Maintenance under Streaming Edge Weight Updates for Real-time Story Identification.
arXiv.org.

Syntactic clustering of the web.

A tight linear time \( (1/2) \)-approximation for unconstrained submodular maximization.
*FOCS*. 


Clique is hard to approximate within \( n^{1-\epsilon} \).
In *Electronic Colloquium on Computational Complexity (ECCC)*.

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

On finding dense subgraphs.
In *ICALP*.

Trawling the Web for emerging cyber-communities.

Finding a team of experts in social networks.
In *KDD*.


