Aalto University
School of Science

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

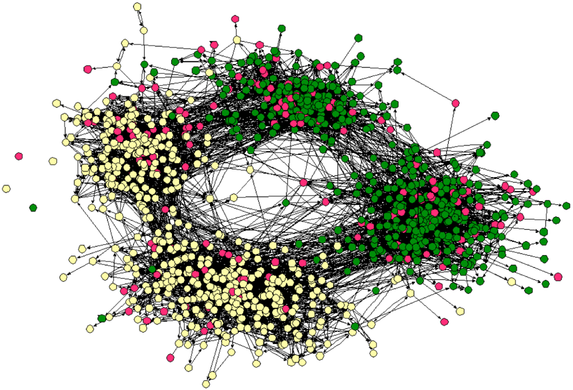
today

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

tomorrow

- finding heavy subgraphs
- centerpiece problems

community detection in graphs and social networks

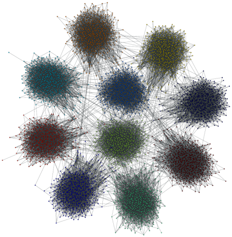


a small network with clear community structure

community structure vs. dense subgraphs

informal definition of community : a set of vertices

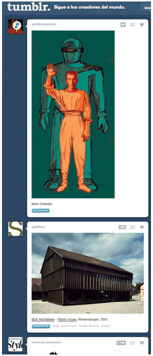
- densely connected to each other, and
- sparsely connected to the rest of the graph

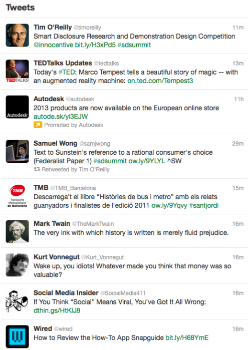


- dense subgraphs: set of vertices with many edges
- no requirement for small cuts
- key primitive for detecting communities,
  but not identical problem

one motivating application — social piggybacking

[Serafini et al., 2013]

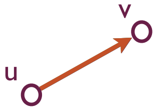




- event feeds: majority of activity in social networks

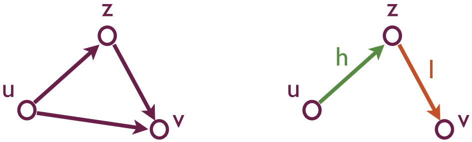
one motivating application — social piggybacking

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

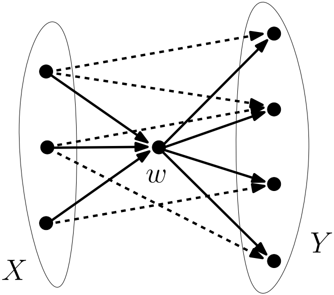


- primitive operation: transfer data between two data stores
- can be implemented as push or pull strategy
- optimal strategy depends on production and consumption rates of nodes

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

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

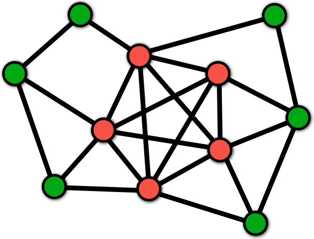
- edges between $S \subseteq V$ and $T \subseteq V$ are

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

  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:

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

- 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 (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 $G = (V, E)$

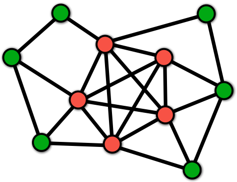
  and a density measure $f : 2^V \to \mathbb{R}$

- find set of vertices $S \subseteq V$

  that maximizes $f(S)$

complexity of density problems — clique

- find the max-size clique in a graph: **NP**-hard problem



- strong innaproximability 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 $\mathbf{P} = \mathbf{NP}$

[Håstad, 1997]

finding dense subgraphs – which measure?

- find large cliques...

  NP-hard problem

  too strict requirement

- find S that maximizes edge ratio $\delta(S) = |E(S)|/\binom{|S|}{2}$

  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

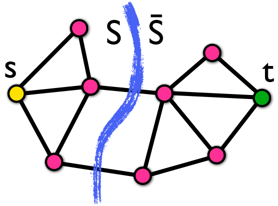
- given an undirected graph $G = (V, E)$
- find set of vertices $S \subseteq V$
- that maximizes the edge density (average degree)

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

- ... polynomial ? ... **NP**-hard ? ... approximations ?

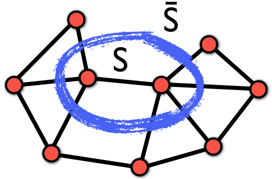
reminder: min-cut and max-cut problems

min-cut problem



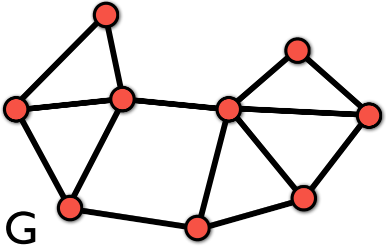
- source $s \in V$, destination $t \in V$
- find $S \subseteq V$, s.t.,
- $s \in S$ and $t \in \bar{S}$, and
- minimize $e(S, \bar{S})$
- polynomially-time solvable
- equivalent to max-flow problem

max-cut problem

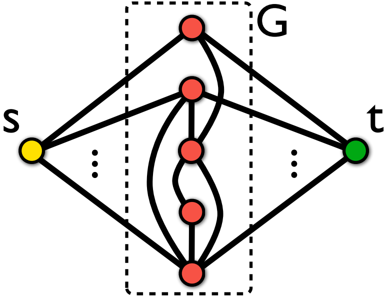


- find $S \subseteq V$, s.t.,
- maximize $e(S, \bar{S})$
- **NP**-hard
- approximation algorithms (0.868 based on SDP)

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?

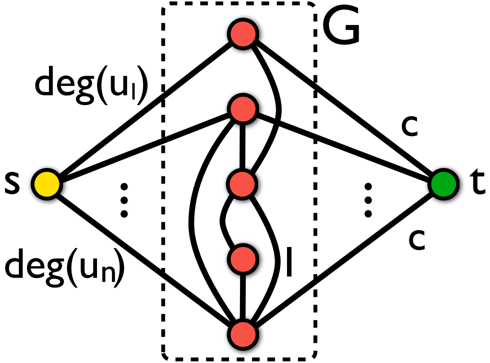
Goldberg's algorithm for densest subgraph

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

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

Goldberg's algorithm for densest subgraph

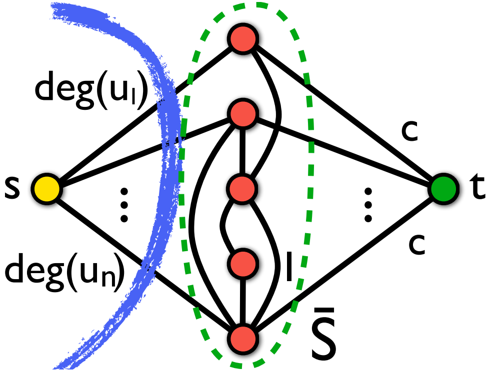
- transform to a min-cut instance



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

Goldberg's algorithm for densest subgraph

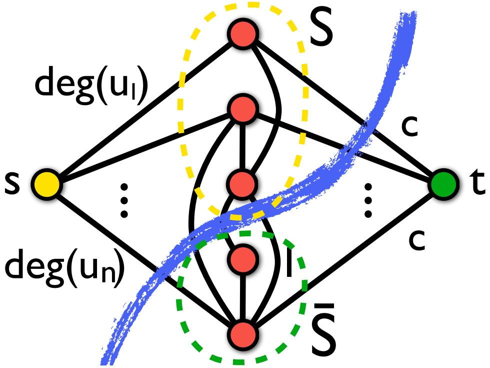
- transform to a min-cut instance



- 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

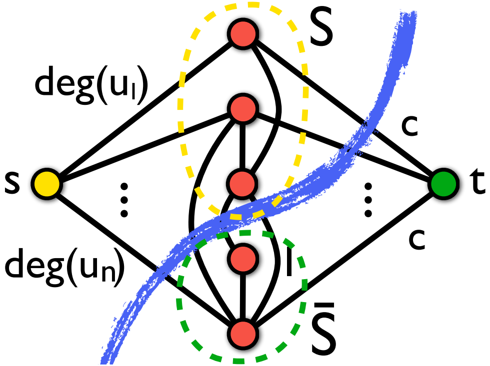
- transform to a min-cut instance



- 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

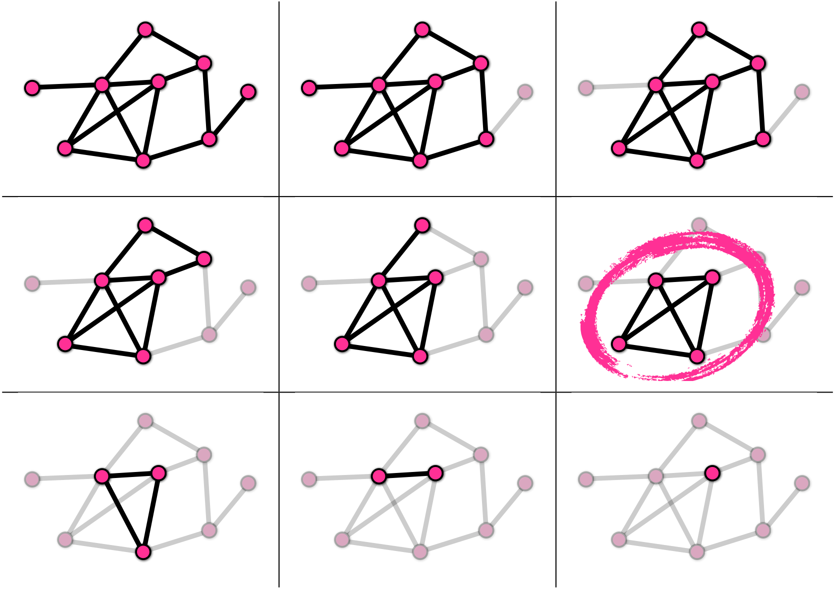
to find the densest subgraph binary search on c

densest subgraph problem – discussion

- Goldberg's algorithm polynomial algorithm, but
- $\mathcal{O}(nm)$ time for one min-cut computation
- not sclable 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

- define

$$\mathrm{in}(u) = \#\{\text{edges assigned to } u\} \quad \text{and} \quad \Delta = \max_{u \in V}\{\mathrm{in}(u)\}$$

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

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

$$|e(S)| = \sum_{u \in S} \mathrm{in}(u) \leq |S|\Delta, \text{ so } d(S) = \frac{2\,|e(S)|}{|S|} \leq 2\,\Delta$$

analysis of the greedy algorithm (II)

- consider assignment defined dynamically during greedy
- initially all edges are unassigned
- in each step, edges are assigned to the deleted vertex
- in the end, all edges have been assigned
- let z be the maximum $d(S)$ achieved by greedy

- claim 2: $\Delta \leq z$

  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

$$z \geq \frac{1}{2} d(S^{\mathrm{OPT}})$$

- 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: $\mathcal{O}(n^2)$

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

  also possible: $\mathcal{O}(m + n)$ (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-constrainted 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

D*k*S "equality" constraint: $|S| = k$

DAM*k*S "at most" constraint: $|S| \leq k$

DAL*k*S "at least" constraint: $|S| \geq k$

size-constrainted densest-subgraph problems

- what about the complexity of DkS, DAMkS, DALkS?
- all **NP**-hard

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

DAMkS as hard as DkS

DALkS factor-2 approximation guarantee

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

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:

```
while (k-core property is satisfied)
    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: $d(S) = 2\,|E(S)|/|S|$
- edge ratio: $\delta(S) = |E(S)|/\binom{|S|}{2}$
- triangle density: $t(S) = |T(S)|/|S|$
- triangle ratio: $\tau(S) = |T(S)|/\binom{|S|}{3}$
- 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

optimal quasicliques

- S is α-quasiclique if $|E(S)| \geq \alpha \binom{|S|}{2}$
- for $S \subseteq V$ define edge surplus

$$f_a(S) = |E(S)| - \alpha \binom{|S|}{2}$$

the optimal quasiclique problem:

find $S \subseteq V$ that maximizes $f_a(S)$

optimal quasicliques in practice

densest subgraph vs. optimal quasiclique

	densest subgraph				optimal quasi-clique			
	$\frac{\lvert S \rvert}{\lvert V \rvert}$	δ	D	τ	$\frac{\lvert S \rvert}{\lvert V \rvert}$	δ	D	τ
Dolphins	0.32	0.33	3	0.04	0.12	0.68	2	0.32
Football	1	0.09	4	0.03	0.10	0.73	2	0.34
Jazz	0.50	0.34	3	0.08	0.15	1	1	1
Celeg. N.	0.46	0.13	3	0.05	0.07	0.61	2	0.26

[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 quasicliques

$$g(x) = x, \quad h(x) = \alpha \, \frac{x(x-1)}{2}$$

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

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

  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 $f_a(S) = |E(S)| - \alpha \binom{|S|}{2}$
- 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

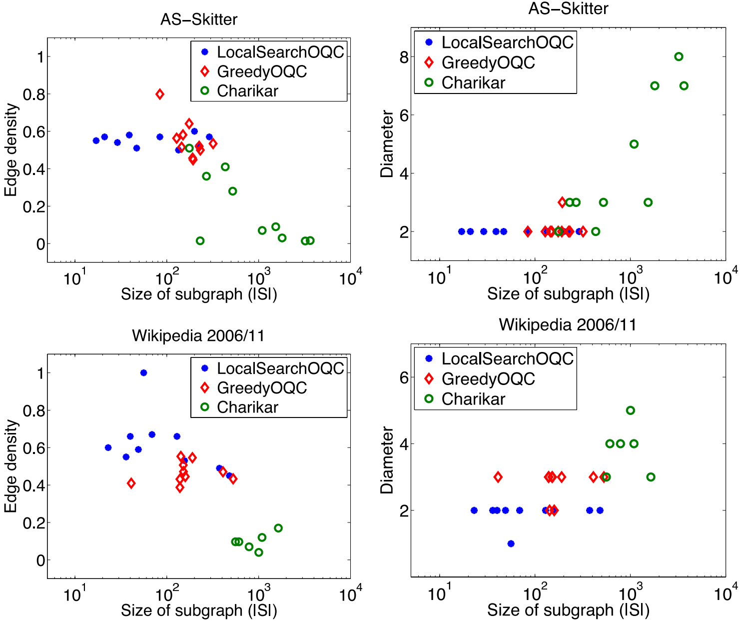
1. further improve solution of greedy by local search

2. choice of α in practice?

   when confronted with two disconnected components, the measure should pick one of the two, instead of their union

   translates to $\alpha \geq \frac{1}{3}$

top-k densest subgraphs and quasicliques



recall our density measures

- edge density: $d(S) = 2\,|E(S)|/|S|$
- edge ratio: $\delta(S) = |E(S)|/\binom{|S|}{2}$
- triangle density: $t(S) = |T(S)|/|S|$
- triangle ratio: $\tau(S) = |T(S)|/\binom{|S|}{3}$
- 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

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 ? ...**NP**-hard ? ...approximations ?

the triangle-densest-subgraph problem

[Tsourakakis, 2014]

- complexity: polynomial
- two exact algorithms

1. tranformation 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 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

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

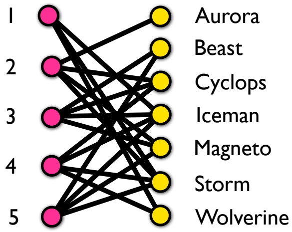
frequent itemsets vs. dense subgraphs

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

$\Leftrightarrow$

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

$\Leftrightarrow$



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

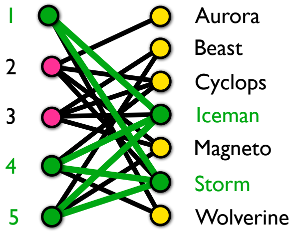
frequent itemsets vs. dense subgraphs

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

⇔



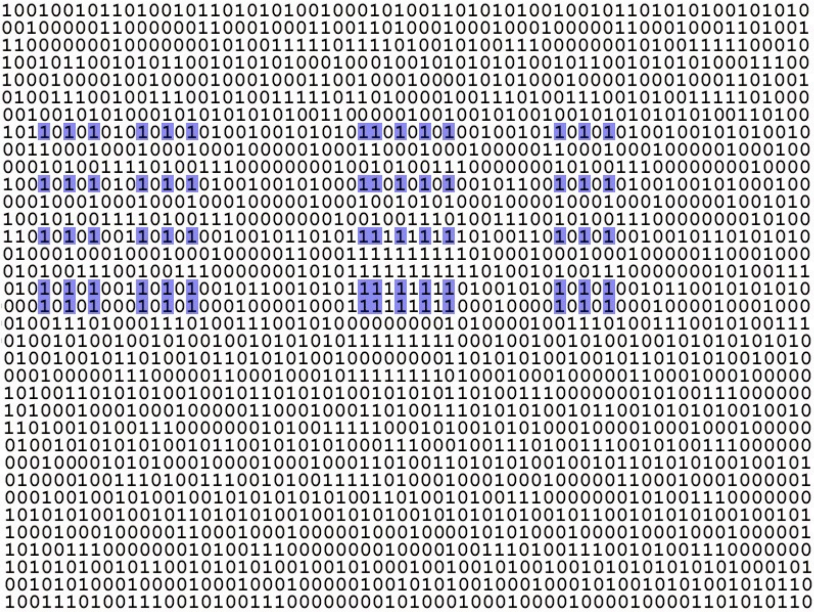
- transaction data ⇔ binary data ⇔ bipartite graphs
- frequent itemsets ⇔ bi-cliques

bi-cliques vs. tiles

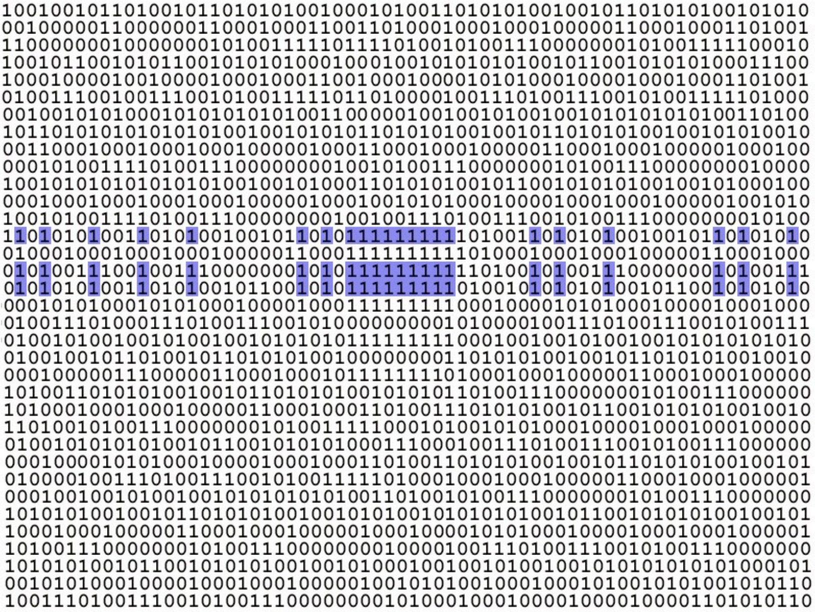
- quality of itemsets measured by support
  require frequency $\geq$ support threshold
- another idea:
  measure itemset quality as {itemset size} $\times$ {support}
  tile mining
  measure corresponds to the area of the tile
  or equivalently, number of edges of the bi-clique
- algorithmically: not monotone measure, developed branch-and-bound technique to mine all tiles

```
1001001011010010110101010010001010011010101001001011010101001010
0010000001100000011000100011001101000100010001000000110001000110
1001110000000010000000010100111110111101001010011100000000101001
1111000101001011001010110010101010001000100101010101001011001010
1010100011100100010000100100001000100011001000100000101010001000
0100010001101001010011100100111001010011111011010000100111010011
1001010011111010000010010101000101010101010011000001001001010010
0101010101001101001011010101010101010100100101010110101010010010
1011010101001011010101001001010100100011000100010001000000100011
0001000100000011000100010000001000100000101001111010011100000000
1001010011100000000101001110000000001000010010101010101010100100
1010001101010100101100101010100100101000100000100010001000100000
1000100101010001000010001000100000010010101001010011110100111000
0000001001001110100111001010011100000000101001101010100110101010
0100101101011111111111010011010101001001011010101001000100010001
0001000001100011111111110100010001000100000011000100001010011100
1001110000000101011111111111010010100111000000001010011101010101
0011010100101100101011111111110100101010101001011001010101000010
1010001010100010000100011111111100010000101010001000010001000010
0111010001110100111001010000000001010000100111010011100101001110
1001010010010100100101010101111111110001001001010010010101010101
0010010111010010111010101001000000000110101010010010111010101001
0010000100000011100000011000100010111111110100010001000000110001
0001000001010011010101001001011010101001010101101001110000000101
0011100000010100010001000100000011000100011010011101010100101100
1010101001001011010010100111000000001010011111000101001010100010
0001000100010000001001010101010100101100101010100011100010011101
0011100101001110000000001000001010100010000100010001000110100110
1010100100101101010100100101000010011101001110010100111110100010
0010001000000110001000100000100010010100100101010101010011010010
1001110000000010100111000000000101010100100101101010100100101010
0101010101001011001010101001001011000100010000001100010001000000
1000100001010100010000100010001000000110100111000000001010011100
000000010000100111010011100101001110000000
```

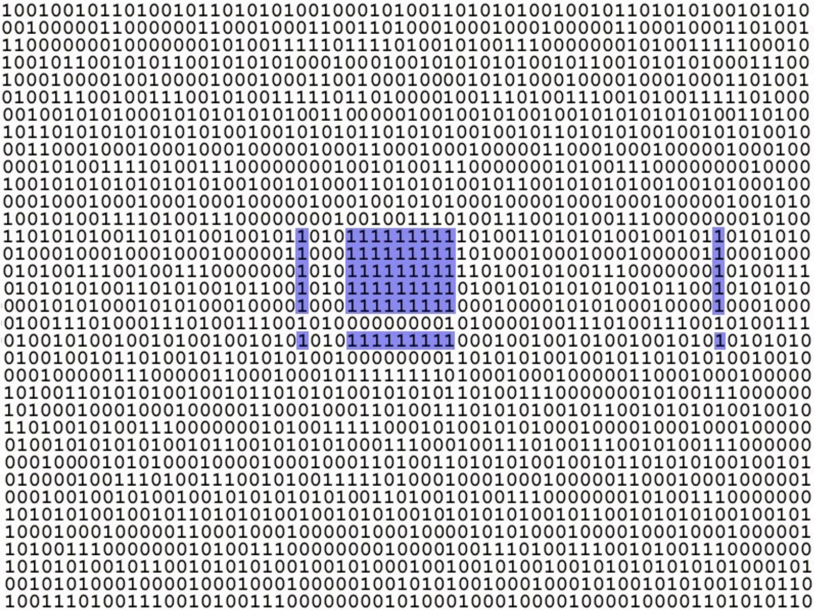
slide by Bart Goethals



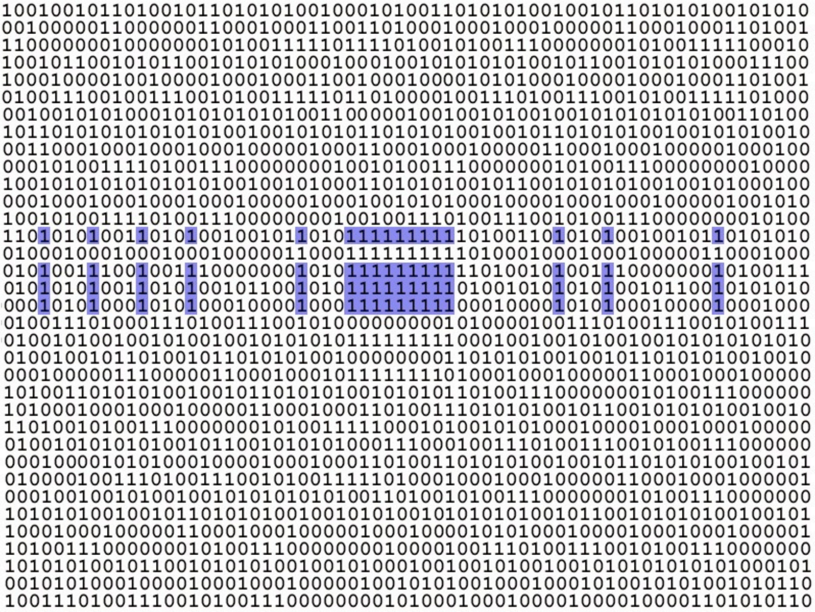
slide by Bart Goethals



slide by Bart Goethals



slide by Bart Goethals



slide by Bart Goethals

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

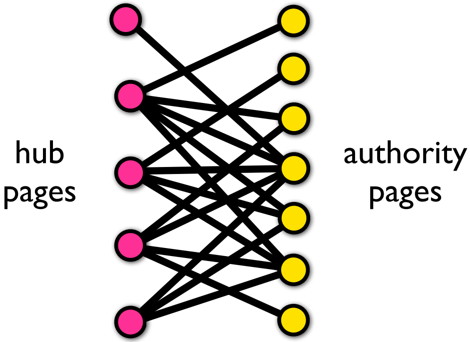
   then G should contain some small core (bi-clique)

2. consider a web graph with no communities

   then small cores are unlikely

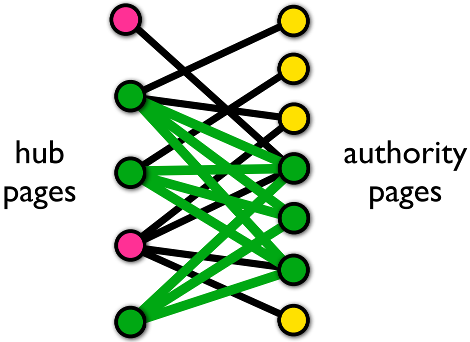
- both observations motivated from theory of random graphs

dense communities contain small cores



[Kumar et al., 1999]

dense communities contain small cores



[Kumar et al., 1999]

finding web communities

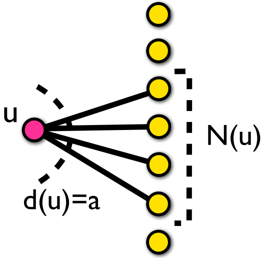
trawling algorithm [Kumar et al., 1999]

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

finding web communities II — graph shingling

[Gibson et al., 2005]

- 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

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

finding web communities II — graph shingling

- 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

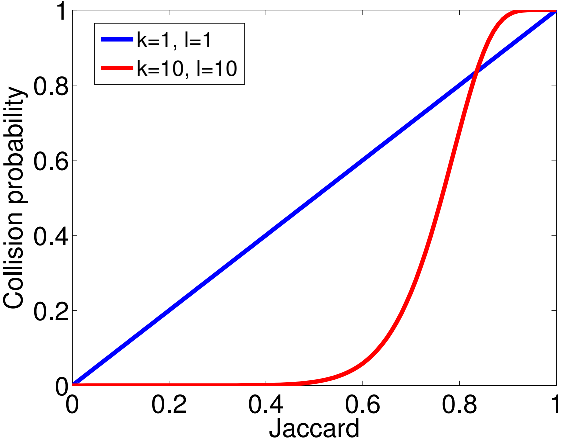
- sets over a universe U
- measuring set similarity using the Jaccard coefficient
- $\pi : U \to U$ a random permutation of U
- $h(A) = \min\{\pi(x) \mid x \in A\}$
- then

$$\Pr[h(A) = h(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- 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 ℓ 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 $d : E \to \mathbb{R}$ 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

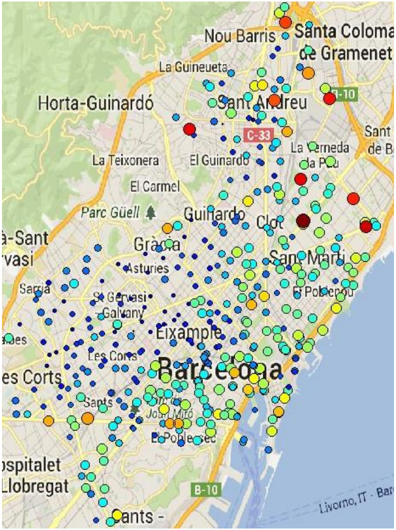
event detection

- sensor networks and traffic measurements

event detection

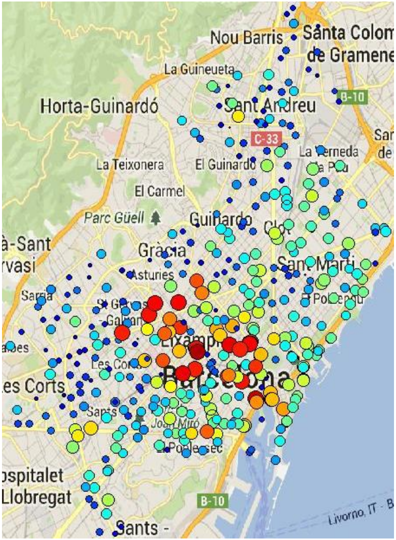
15.11.2012

ordinary day, no events



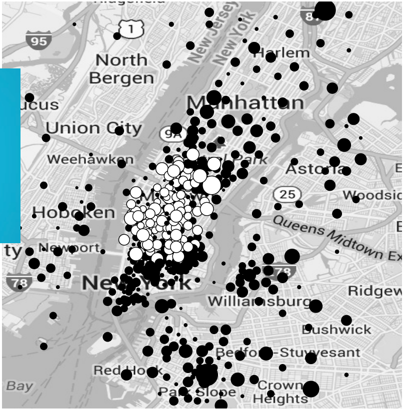
11.09.2012

Catalunya national day



event detection

- location-based social networks



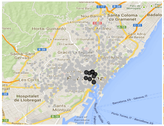
discovering heavy subgraphs

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

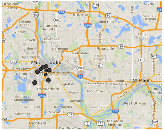
discovering heavy subgraphs

- 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]

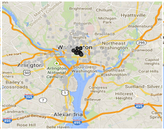
events discovered with bicing and 4square data



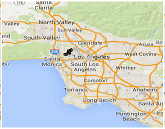
(a) Barcelona: 11.09.12 National Day of Catalonia



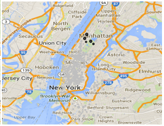
(b) Minneapolis: 4.07.12 Independence Day



(c) Washington, DC: 27.05.13 Memorial Day

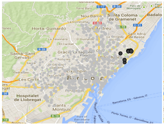


(d) Los Angeles: 31.05.10 Memorial Day

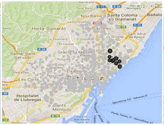


(e) New York: 6.09.10 Labor Day

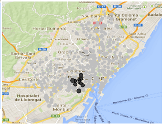
Figure 4: Public holiday city-events discovered using the SDP algorithm.



(a) 01.06.12 Primavera sound music festival



(b) 18.09.12 festival of the Poblenou neighborhood



(c) 31.10.12 Halloween

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

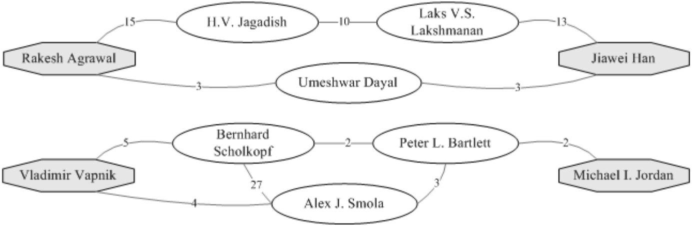
- $r(Q, j)$ goodness score of j wrt Q

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

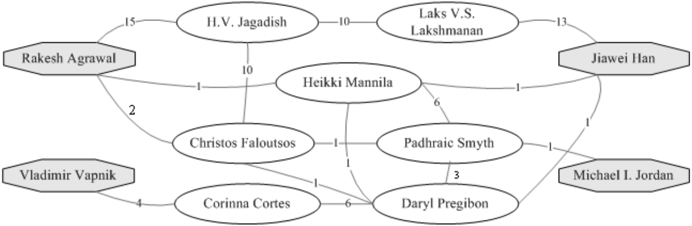
- proposed algorithm:

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



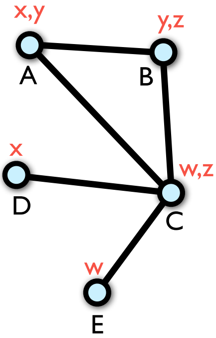
(a) "K_softANDquery": $k = 2$



(b) "AND query"

[Tong and Faloutsos, 2006]

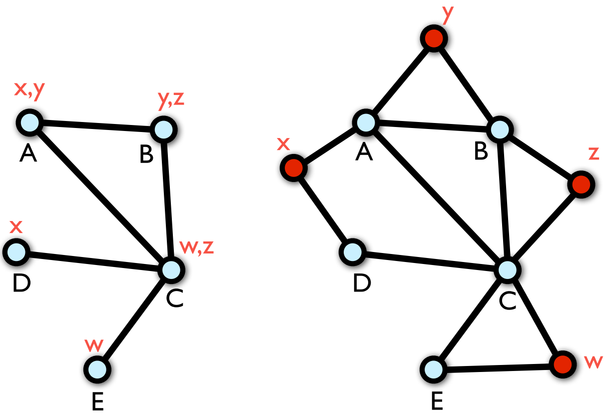
the team-formation problem



[Lappas et al., 2009]

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

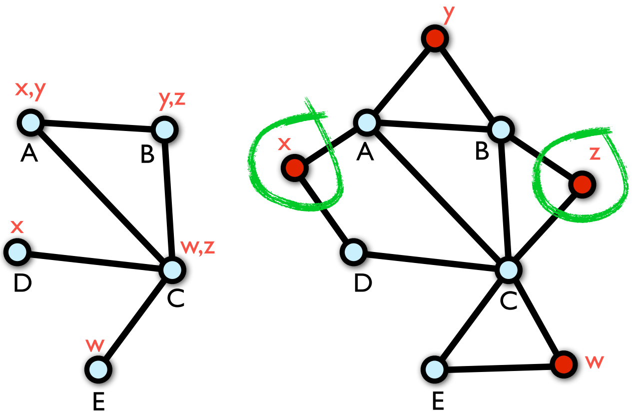
the team-formation problem



[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

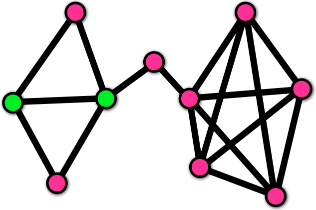
- given: graph $G = (V, E)$ and set of query vertices $Q \subseteq V$
- find: a connected subgraph H that
  (a) contains Q
  (b) optimizes a density function $d(H)$
  (c) possibly other constraints

- density function (b):

  average degree, minimum degree, quasiclique, etc.

  measured on the induced subgraph H

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)

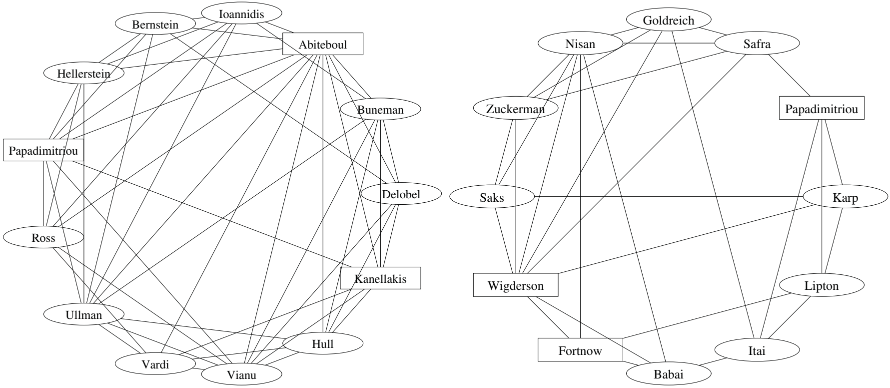
baseline: increamental addition of vertices

- 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

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 **NP**-hard problems
- choice of density measure matters

promising future directions

- room for new concepts
- better algorithms for upper-bound constraints
- top-k versions of dense subgraphs
- formulations for enriched graphs (labels or attributes)
- local algorithms

references

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*.

Angel, A., Koudas, N., Sarkas, N., and Srivastava, D. (2012).
Dense Subgraph Maintenance under Streaming Edge Weight Updates for Real-time Story Identification.
arXiv.org.

Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G. (1997).
Syntactic clustering of the web.
In *Selected papers from the sixth international conference on World Wide Web*, pages 1157–1166, Essex, UK. Elsevier Science Publishers Ltd.

Buchbinder, N., Feldman, M., Naor, J. S., and Schwartz, R. (2012).
A tight linear time (1/2)-approximation for unconstrained submodular maximization.
FOCS.

references (cont.)

- Charikar, M. (2000).
  Greedy approximation algorithms for finding dense components in a graph.
  In *APPROX*.
- Feige, U., Kortsarz, G., and Peleg, D. (2001).
  The dense k-subgraph problem.
  Algorithmica, 29(3):410–421.
- Fratkin, E., Naughton, B. T., Brutlag, D. L., and Batzoglou, S. (2006).
  MotifCut: regulatory motifs finding with maximum density subgraphs.
  Bioinformatics, 22(14).
- Gibson, D., Kumar, R., and Tomkins, A. (2005).
  Discovering large dense subgraphs in massive graphs.
  In *Proceedings of the 31st international conference on Very large data bases*, pages 721–732. VLDB Endowment.
- Goldberg, A. V. (1984).
  Finding a maximum density subgraph.
  Technical report.

references (cont.)

Håstad, J. (1997).
Clique is hard to approximate within $n^{1-\epsilon}$.
In *Electronic Colloquium on Computational Complexity (ECCC)*.

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).

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.

Lappas, T., Liu, K., and Terzi, E. (2009).
Finding a team of experts in social networks.
In *KDD*.

references (cont.)

Rozenshtein, P., Anagnostopoulos, A., Gionis, A., and Tatti, N. (2014).
Event detection in activity networks.
In *KDD*.

Serafini, M., Gionis, A., Junqueira, F., Leroy, V., and Weber, I. (2013).
Piggybacking on Social Networks.
PVLDB, pages 1–12.

Sozio, M. and Gionis, A. (2010).
The community-search problem and how to plan a successful cocktail party.
In *KDD*.

Tong, H. and Faloutsos, C. (2006).
Center-piece subgraphs: problem definition and fast solutions.
In *KDD*.

Tsourakakis, C. (2014).
A Novel Approach to Finding Near-Cliques: The Triangle-Densest Subgraph Problem.
arXiv.org.

references (cont.)

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 *KDD*.

Zhang, B. and Horvath, S. (2005).
A general framework for weighted gene co-expression network analysis.
Statistical applications in genetics and molecular biology, 4(1):1128.