The community-search problem and how to plan a successful cocktail party

Mauro Sozio

Max Planck Institute, Germany

Aris Gionis

Yahoo! Research, Barcelona

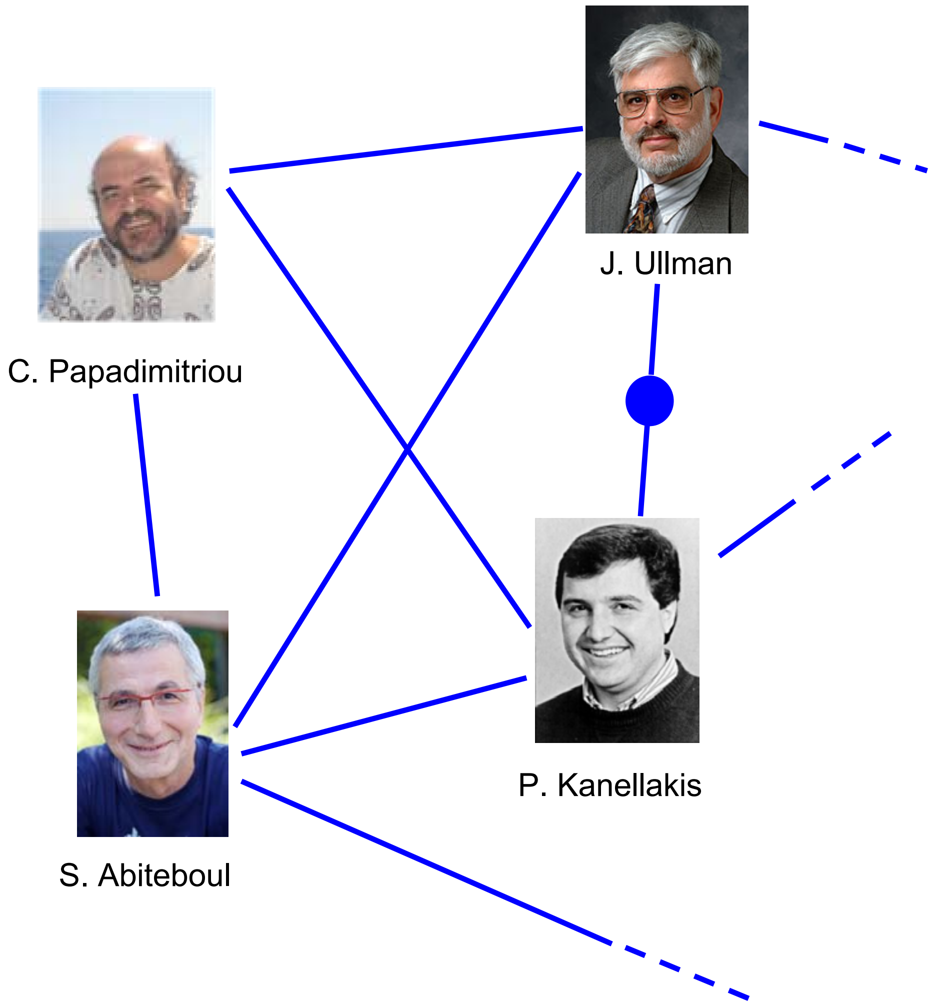
Planning a cocktail party

C. Papadimitriou

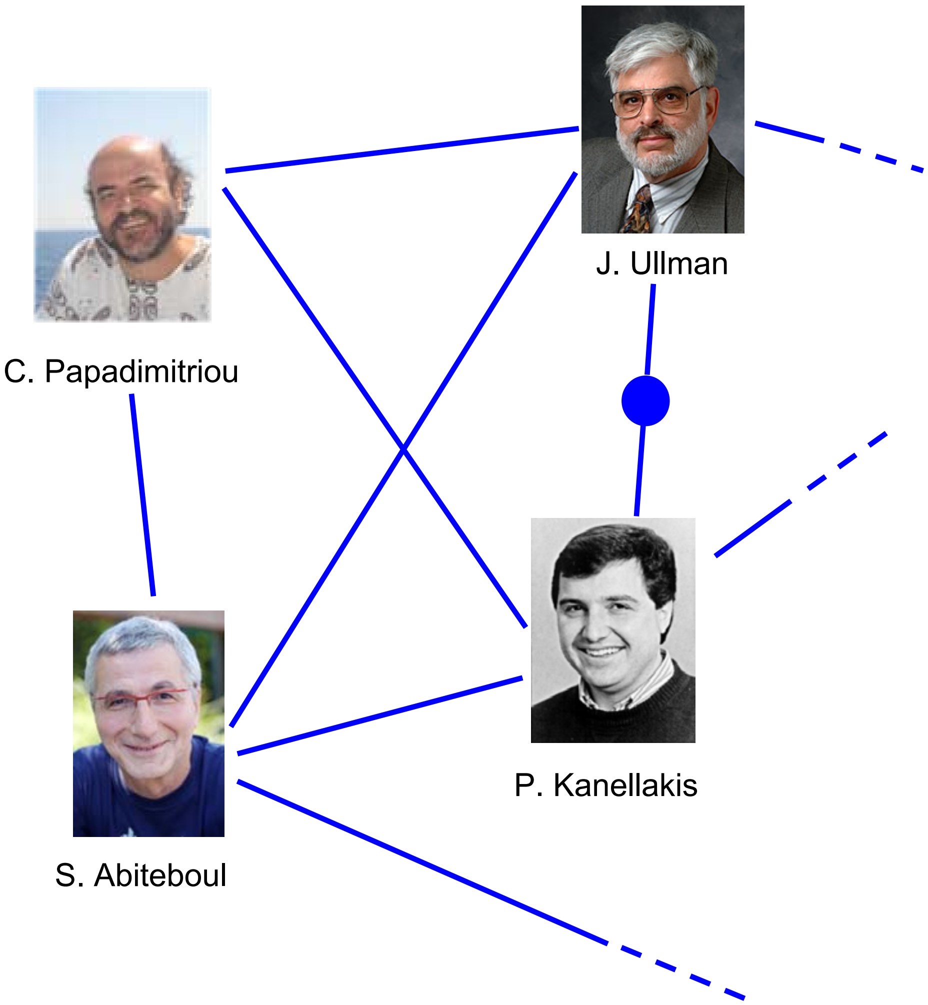
P. Kanellakis

S. Abiteboul

Planning a cocktail party



Planning a cocktail party



Recipe for a successful party: 

- Participants should be “close” to the organizers (e.g. a friend of a friend).
- Everybody should know some of the participants.
- The graph should be connected.
- The number of participants should not be too small but…
- …not too large either!!!
- ….

Not an easy task…

The community-search problem

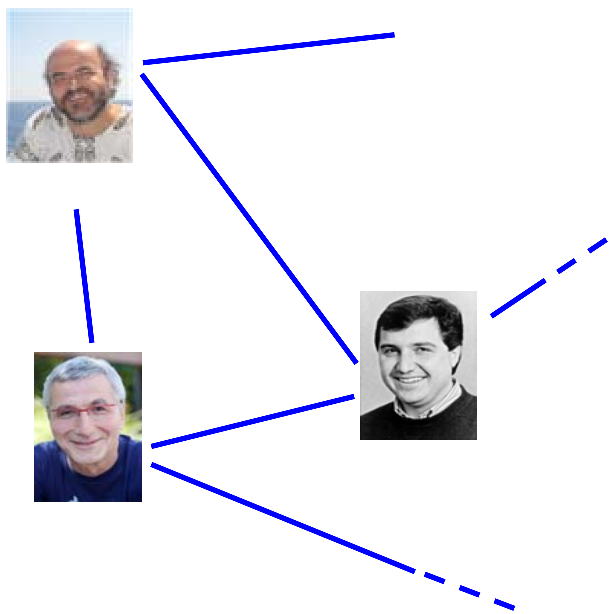
Our problem: find the community that a given set of users belongs to.

Our approach: Given a graph and a set of nodes, find a densely connected subgraph containing the set of users given in input.

The community-search problem

Our problem: find the community that a given set of users belongs to.

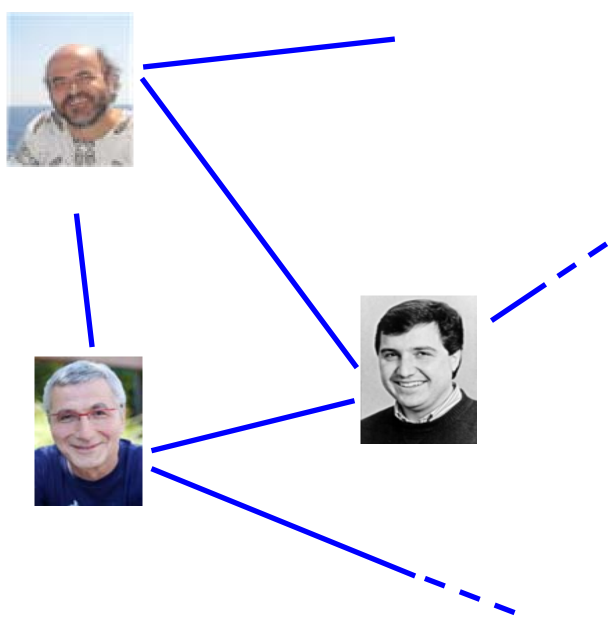
Our approach: Given a graph and a set of nodes, find a densely connected subgraph containing the set of users given in input.



The community-search problem

Our problem: find the community that a given set of users belongs to.

Our approach: Given a graph and a set of nodes, find a densely connected subgraph containing the set of users given in input.

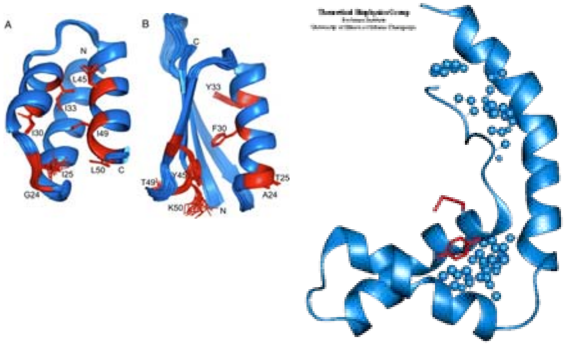


The community-search problem

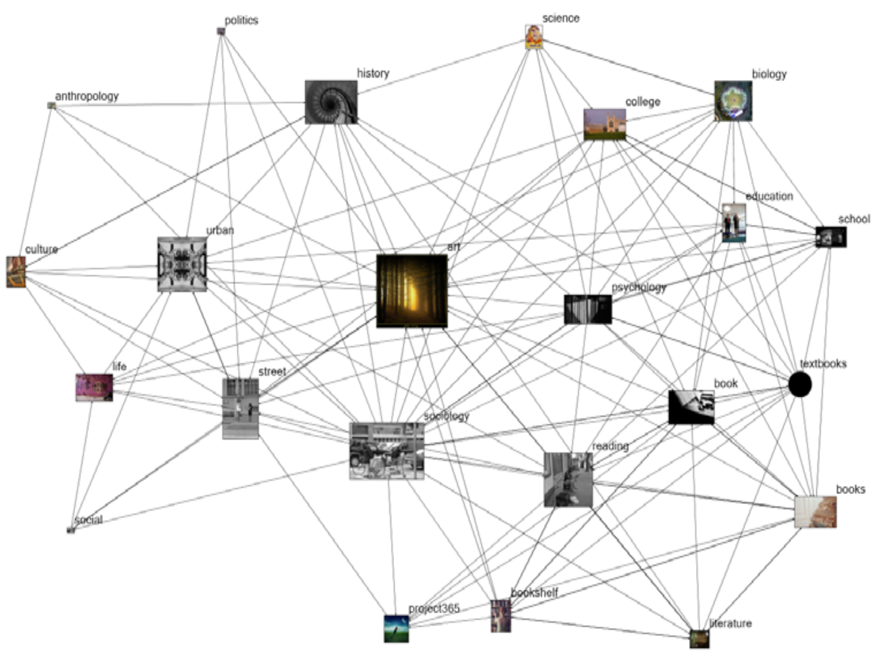
Our problem: find the community that a given set of users belongs to.

Our approach: Given a graph and a set of nodes, find a densely connected subgraph containing the set of users given in input.

Other applications: Tag suggestions, biological data.



Tag suggestion in Flickr



Tags: Dolomites

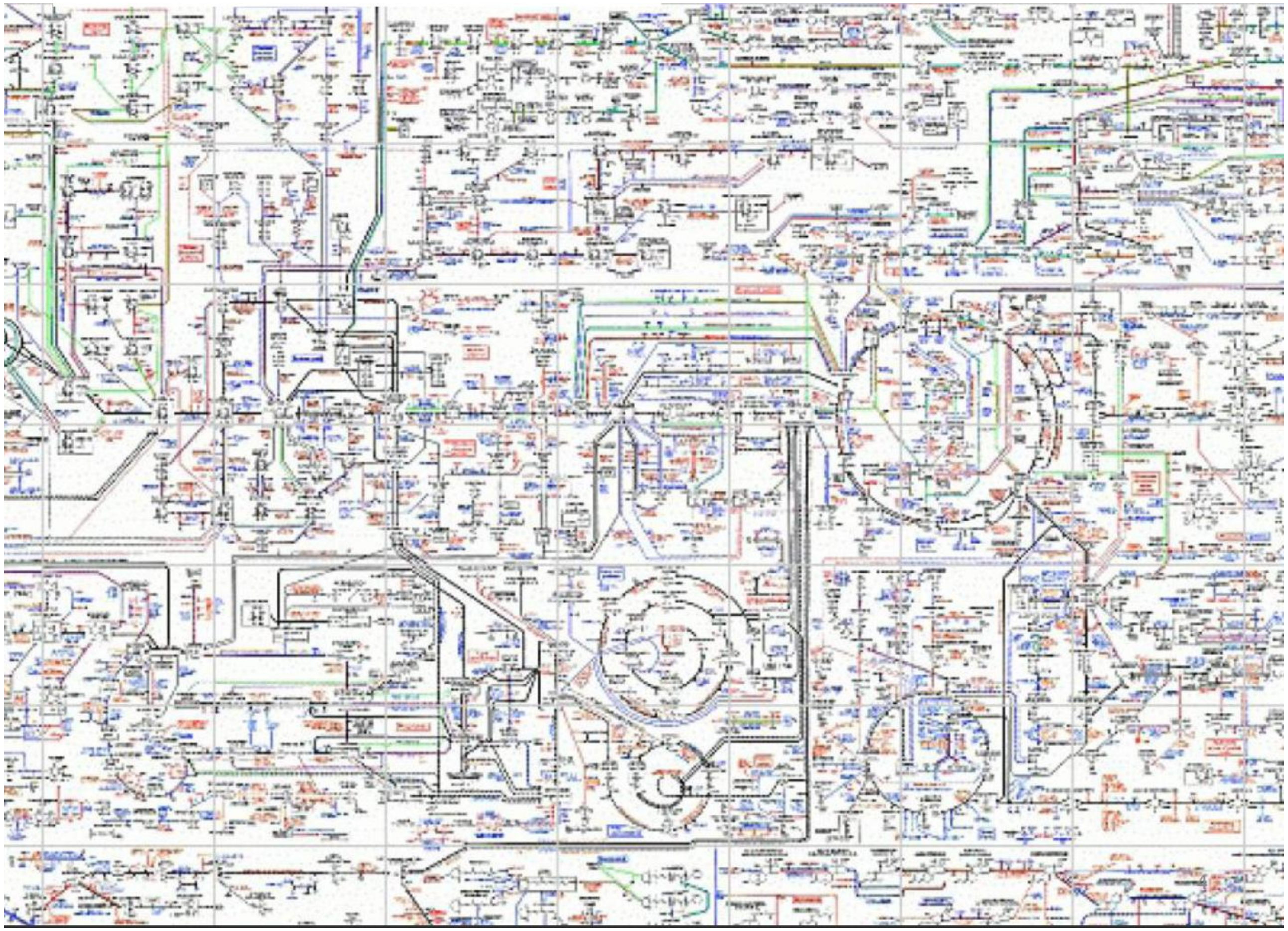
Lake

Sugg.: Mountains

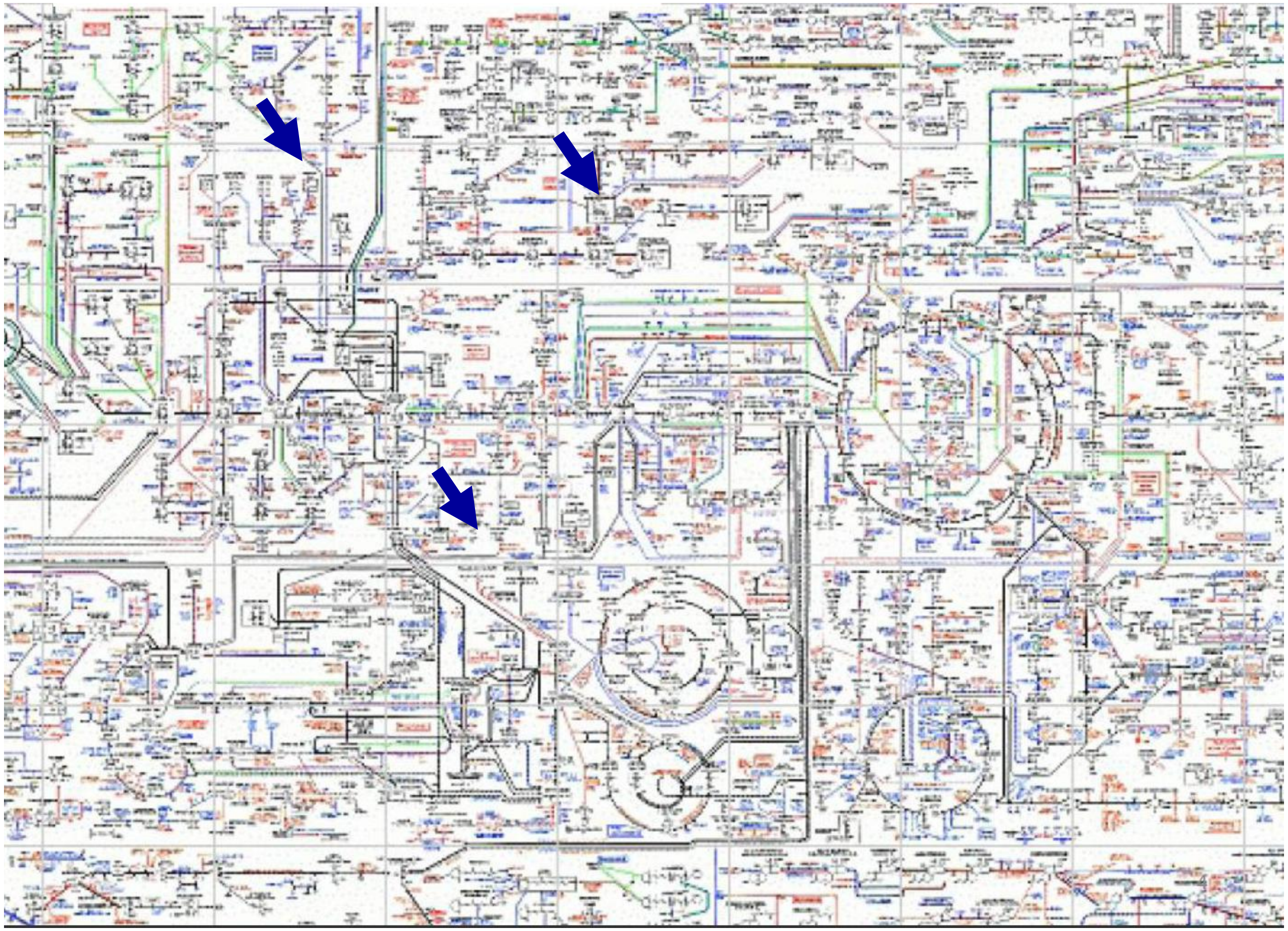
Nature

Landscape

Protein interactions



Protein interactions



Outline

- Introduction
- Related work
- Problem Definition
- Our Algorithms
  - Greedy
  - GreedyDist and GreedyFast
- Evaluation
- Generalization to Monotone Functions
- Conclusions and Future Work

Related Work

Large body of work on finding communities in social networks:

- Agarwal and Kempe (European Physics Journal, 2008)
- S. White and P. Smyth. (SDM, 2005)
- Y. Dourisboure et al. (WWW, 2007)
- D. Gibson, R. Kumar, and A. Tomkins (VLDB, 2005)

Our work: Query-dependent variant of the problem.

Other related work:

- Y. Koren, S. C. North, and C. Volinsky (TKDD, 2007)
- H. Tong and C. Faloutsos (KDD, 2006)
- Lappas et al. (KDD, 2009)
- FOCS, ICALP, APPROX

Outline

- Introduction
- Related work
- Problem Definition
- Our Algorithms
  - Greedy
  - GreedyDist and GreedyFast
- Evaluation
- Generalization to Monotone Functions
- Conclusions and Future Work

Density?

Good properties: small distance, density, connected subgraph

Two definitions of density of a graph

- d(G)=# of edges in G / # of edges in a clique

  Formally, $\frac{m}{n(n-1)/2}$
- D(G)=# of edges in G / # of vertices in G

  Formally, $\frac{m}{n}$ = average degree of G / 2

Fact 1: Computing a subgraph H with maximum density d(H) is NP-Hard (reduction from Max Clique).

Fact 2: Computing a subgraph H with maximum density D(H) can be done in polynomial time but the algorithm is slow.

Another definition of density: minimum node degree.

Density?

Good properties: small distance, density, connected subgraph

Two definitions of density of a graph

- d(G)=# of edges in G / # of edges in a clique

  Formally, $\frac{m}{n(n-1)/2}$

- D(G)=# of edges in G / # of vertices in G

  Formally, $\frac{m}{n}$ = average degree of G / 2

Fact 1: Computing a subgraph H with maximum density d(H) is NP-Hard (reduction from Max Clique).

Fact 2: Computing a subgraph H with maximum density D(H) can be done in polynomial time but the algorithm is slow.

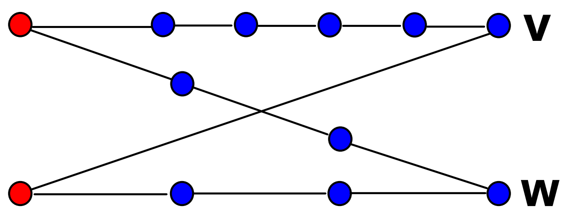
Another definition of density: minimum node degree. ✓

Distance?

Only one query node: d(v,q) = length of the shortest path between *v* and *q*.

Many query nodes:

1. Sum of the distances: Formally, $\sum_{q \in Q} d(q,v)$ but



2. Sum of the squared dist.: Formally $\sum_{q \in Q} d(q,v)^2$. It favors "balanced" scenarios.

Problem definition

- Problem definition: Given an undirected graph $G= (V,E)$, a set of query nodes $Q \subseteq V$, an integer d (distance constraint), we are to find an induced subgraph $H = (V_H, E_H)$ of G, s.t.
  - (i) V_H contains Q;
  - (ii) H is connected;
  - (iii) all nodes in H are at distance at most d from Q;
  - (iv) the minimum degree of H is maximized.

Problem definition

- Problem definition: Given an undirected graph G= (V,E), a set of query nodes $Q \subseteq V$, an integer d (distance constraint), we are to find an induced subgraph $H = (V_H, E_H)$ of G, s.t.
  - (i) V_H contains Q;
  - (ii) H is connected;
  - (iii) all nodes in H are at distance at most d from Q;
  - (iv) the minimum degree of H is maximized.

Good news: There is an optimum greedy algorithm!!!

Outline

- Introduction
- Related work
- Problem Definition
- Our Algorithms
  - Greedy
  - GreedyDist and GreedyFast
- Evaluation
- Generalization to Monotone Functions
- Conclusions and Future Work

Our greedy algorithm

1. Let $G = G_0$.
2. At each step t if there is a node v in G_{t-1} violating the distance constraint, then remove v and all its edges;
3. otherwise remove the node with minimum degree in G_{t-1}.
4. Let G_t the graph so obtained.
5. Among all the graphs $G_0, G_1, \ldots. G_T$ constructed during the execution of the algorithm return the graph G_i
   - containing the query nodes;
   - satisfying the distance constraint;
   - with maximum minimum degree.

Our greedy algorithm

1. Let $G=G_0$.
2. At each step t if there is a node v in G_{t-1} violating the distance constraint, then remove v and all its edges;
3. otherwise remove the node with minimum degree in G_{t-1}.
4. Let G_t the graph so obtained.
5. Among all the graphs $G_0, G_1, \ldots. G_T$ constructed during the execution of the algorithm return the graph G_i
   - containing the query nodes;
   - satisfying the distance constraint;
   - with maximum minimum degree.

Theorem: Our greedy algorithm computes an optimum solution for the community-search problem.

Size Matters!

The size of the community shouldn't be too large:

- If we are to organize a party we might not have place for 1M people.
- Humans should be able to analyze the result.

Bad news: Adding a cardinality constraint on the number of nodes makes the problem NP-Hard (red. from Steiner Tree) but...

Theorem: Let H and H' be two graphs obtained by executing our greedy algorithm with distance constraint d and d', respectively (the other input parameters are the same). Then, $d' \leq d$ implies $|V(H)| \leq |V(H')|$.

GreedyDist

Intuition: Bound the size of the graph by making the distance constraint tighter.

GreedyDist:

- Let k be an upperbound on the number of vertices and let d be a distance constraint.
- While the number of vertices of the computed graph is larger than k
  - Execute Greedy with distance constraint d.
  - Decrease d by one (d--);

GreedyFast

Intuition: Nodes that are far away from the query nodes are most probably not related to them.

GreedyFast:

- Let k be an upperbound on the number of vertices and let d be a distance constraint.
- Preprocessing: consider only the k closest nodes to the query nodes.
- Run Greedy with the subgraph induced by these query nodes, as input

Outline

- Introduction
- Related work
- Problem Definition
- Our Algorithms
  - Greedy
  - GreedyDist and GreedyFast
- Evaluation
- Generalization to Monotone Functions
- Conclusions and Future Work

Evaluation

We evaluate our algorithms on three different datasets:

- DBLP (226k nodes and 1.4M edges);
- Flickr tag graph (38k nodes and 1.3M edges);
- Bio data (16K nodes and 491k nodes).

Queries are generated randomly.

We vary

- Number of query nodes;
- Distance between query nodes;
- Upper bound on the number of nodes.

We measure

- Minimum degree and average degree;
- Size of the output graph;
- Running time.

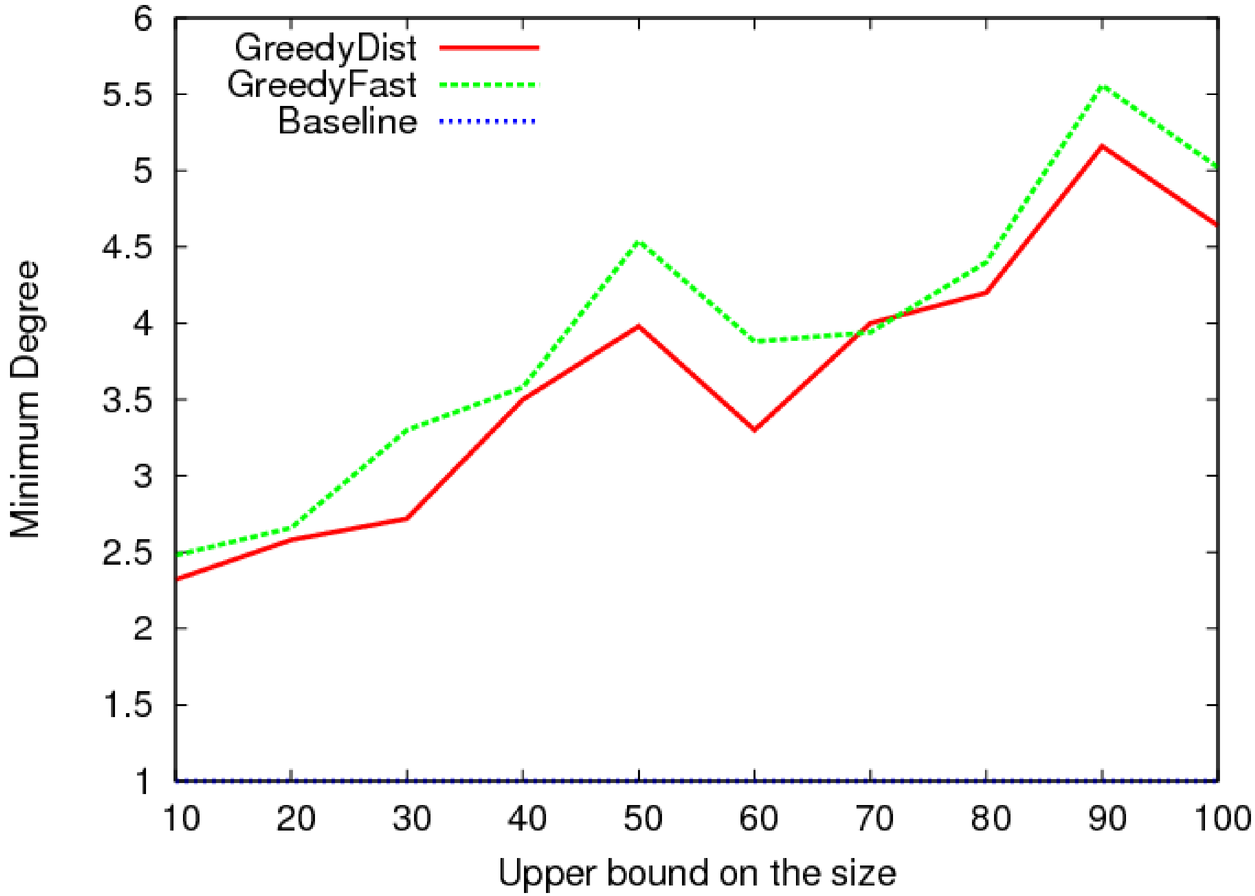
Baseline

We consider an approach where at each step we add one node (in contrast with all previous approaches).

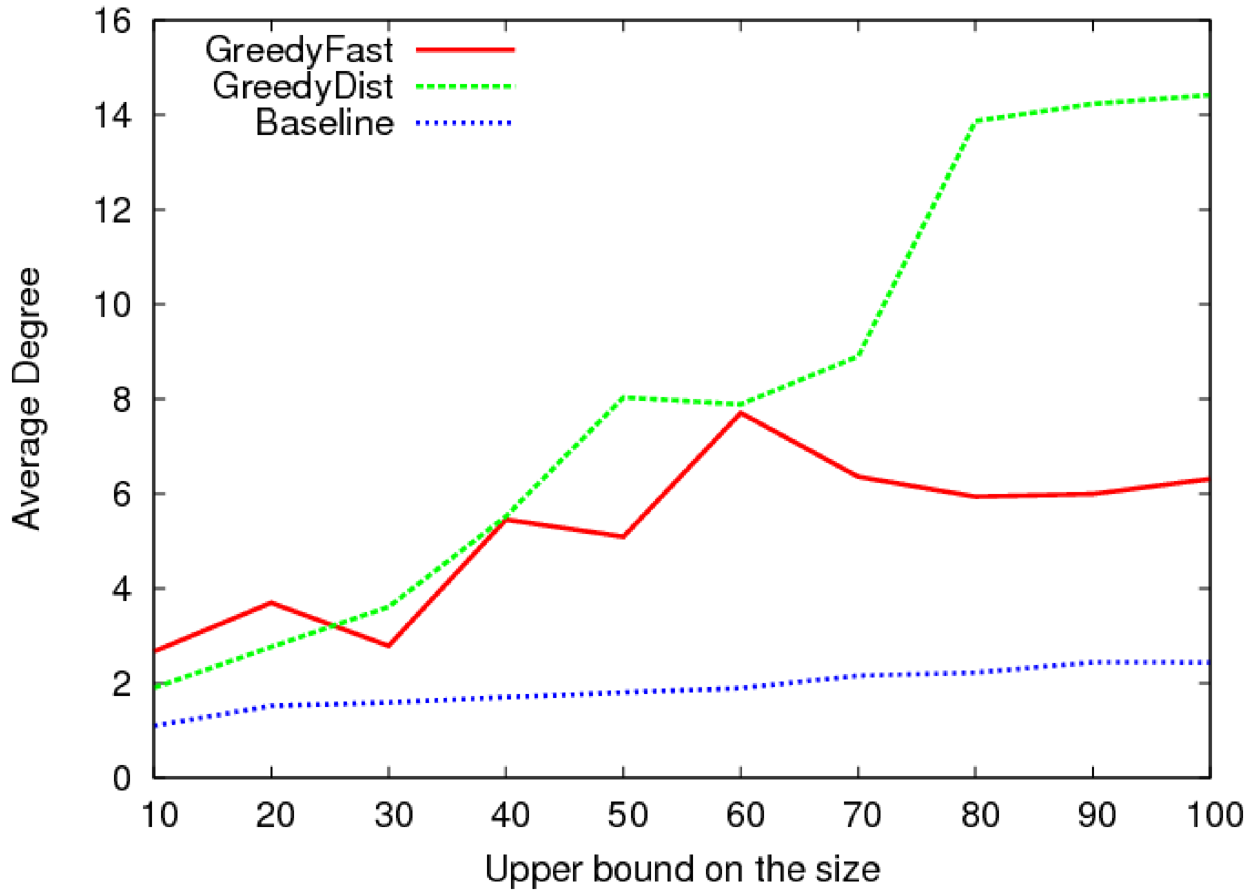
A pseudocode:

1. Connect the query nodes: by means of a Steiner Tree algo. (we use a 2-approximation algorithm for this problem);
2. Let G_t be the graph at step t;
3. Add the node v with maximum degree in $G_t \cup v$;
4. Among all the graph $G_0,\ldots,G_T$ constructed, return the one with maximum minimum degree.

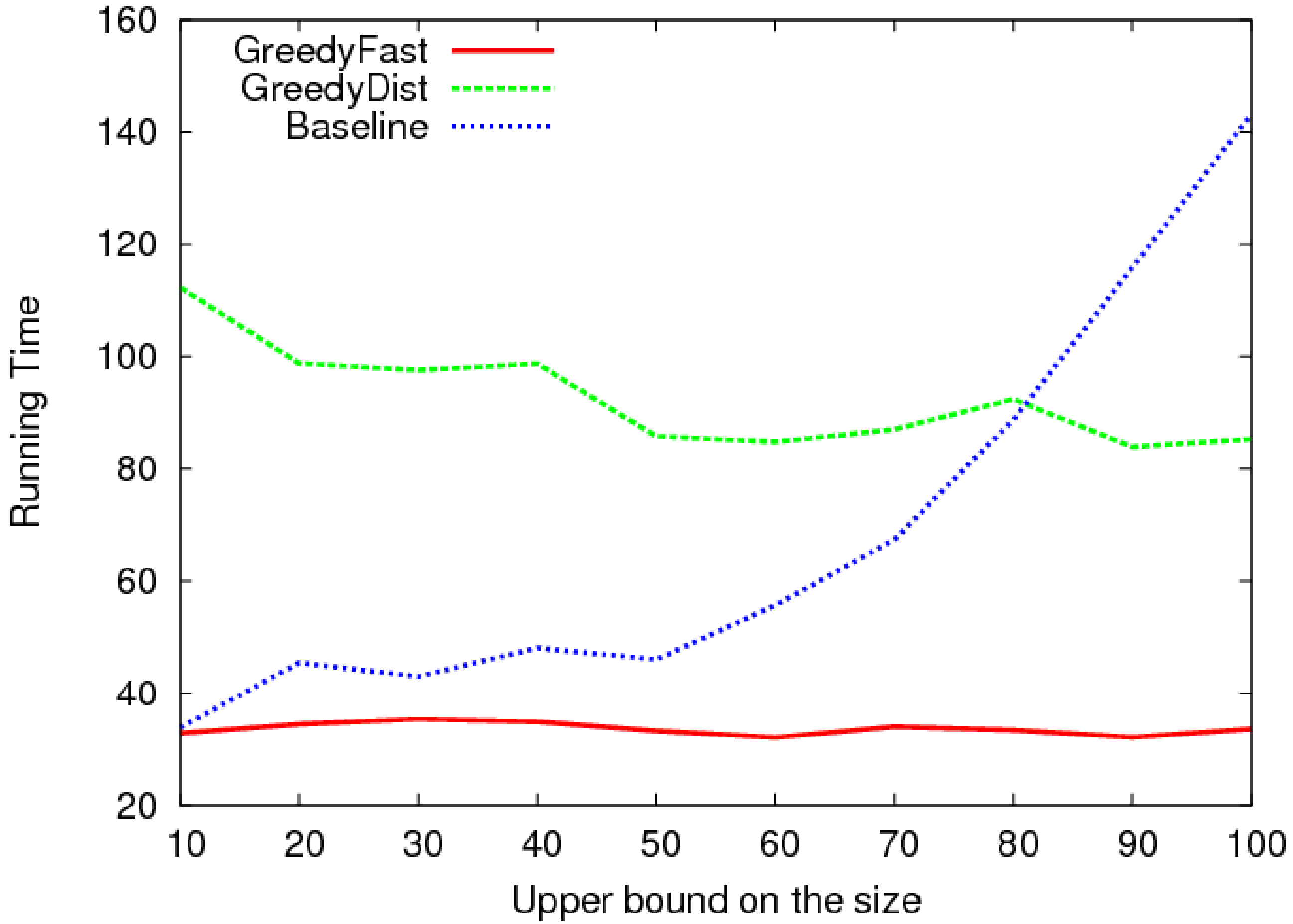
Minimum degree vs Size (Flickr)



Average deg. vs. Size (Flickr)



Running time vs Size (Flickr)



Distance vs Size

Theorem: Let H and H' be two graphs obtained by executing our greedy algorithm with distance constraint d and d', respectively (the other input parameters are the same). Then, $d' \leq d$ implies $|V(H)| \leq |V(H')|$.

	6	9	11	12	14	17	20	27
BIOMINE	71	76	77	867	870	900	923	1394
DBLP	4	9	9	13	14.5	17	21	160
tag	35	248	248	3316	3554	8287	8305	14256

Outline

- Introduction
- Related work
- Problem Definition
- Our Algorithms
  - Greedy
  - GreedyDist and GreedyFast
- Evaluation
- Generalization to Monotone Functions
- Conclusions and Future Work

Generalized Community-Search Problem

Input:

- An undirected graph $G=(V,E)$;
- A set Q of query nodes;
- Integer parameters k,t;
- A set of skills T_v associated to every node v;
- A required set of skills $\bar{T}$.

Goal: Find an induced subgraph H of G s.t.

- G is connected and contains Q;
- The number of vertices of H is $\geq t$;
- The set of skills of H contains $\bar{T}$ ($\cup_{v \in H} T_v \supseteq \bar{T}$);
- Any node is at distance at most k from the query nodes;
- The minimum degree is maximized.

Generalized Community-Search Problem

Input:

- An undirected graph G=(V,E);
- A set Q of query nodes;
- Integer parameters k,t;
- A set of skills T_v associated to every node v;
- A required set of skills $\bar{T}$.

Goal: Find an induced subgraph H of G s.t.

- G is connected and contains Q;
- The number of vertices of H is ≥ t;
- The set of skills of H contains $\bar{T}$ ($\cup_{v \in H} T_v \supseteq \bar{T}$);
- Any node is at distance at most k from the query nodes;
- The minimum degree is maximized.

Monotone functions

Generalized Greedy: Guarantees

Monotone function: f(H) ≤ f(G), if H is a subgraph of G.

Theorem: There is an optimum greedy algorithm for the problem when all constraint are monotone functions.

Running time: Depends on the time to evaluate the function $f_1,\ldots,f_k$, formally $O\left(m + \sum_i n \bullet T_i\right)$ where T_i is the time to evaluate the monotone function f_i

Outline

- Introduction
- Related work
- Problem Definition
- Our Algorithms
  - Greedy
  - GreedyDist and GreedyFast
- Evaluation
- Generalization to Monotone Functions
- Conclusions and Future Work

Conclusions and Future Work

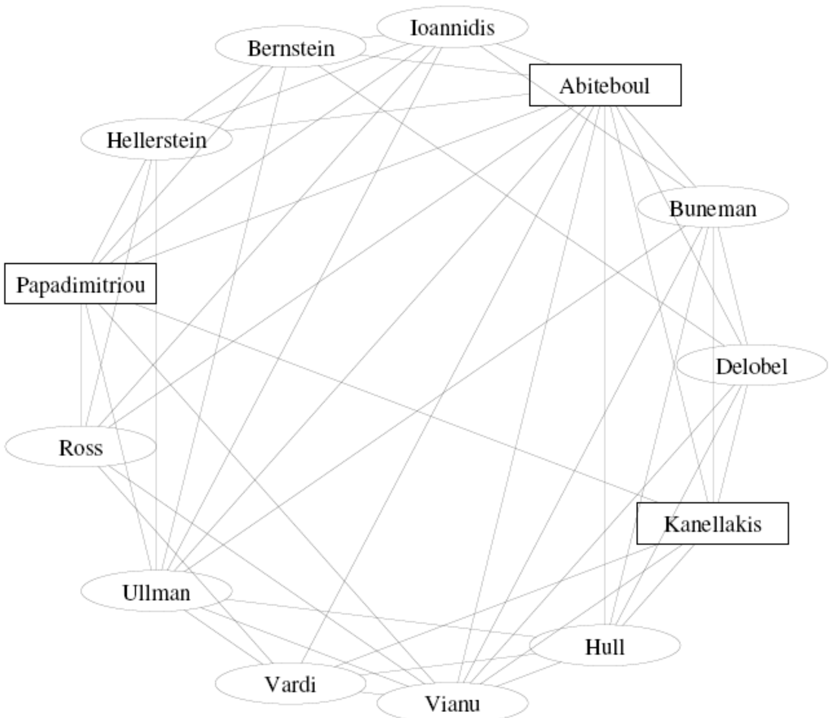
Contributions:

- We proposed a novel combinatorial approach for finding the community of a given set of users in input.
- Distance constraints proved to be effective in limiting the size of the output graph.
- We defined a class of functions that can be optimized efficiently.

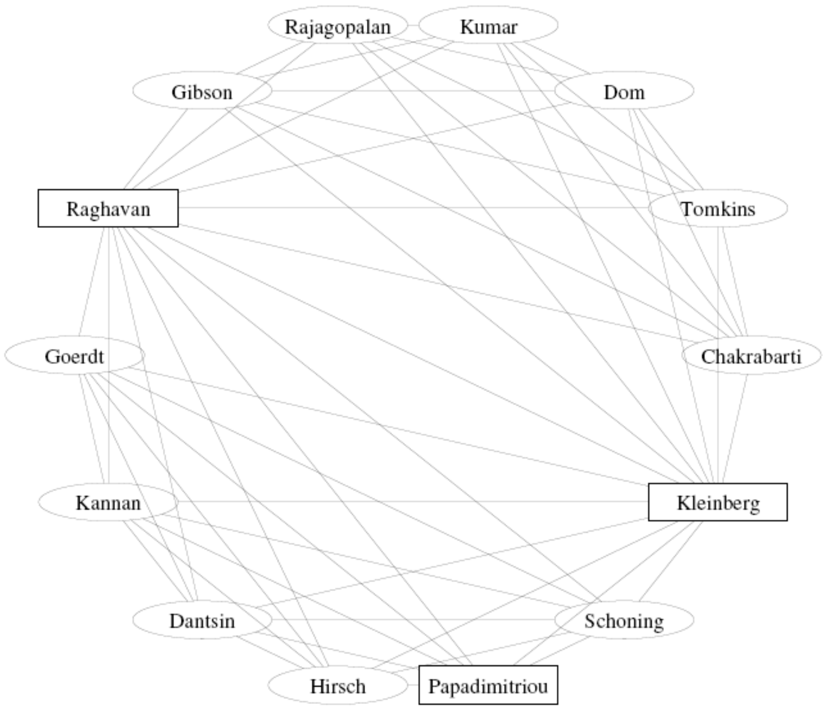
Future work:

- Are there other useful monotone functions?
- Can we find all communities of a given set of users?
- Community search via Map-Reduce?

What about the party?



Community 1: Database



Community2: Algorithms

Thanks