Community Search and Cocktail Party Planning

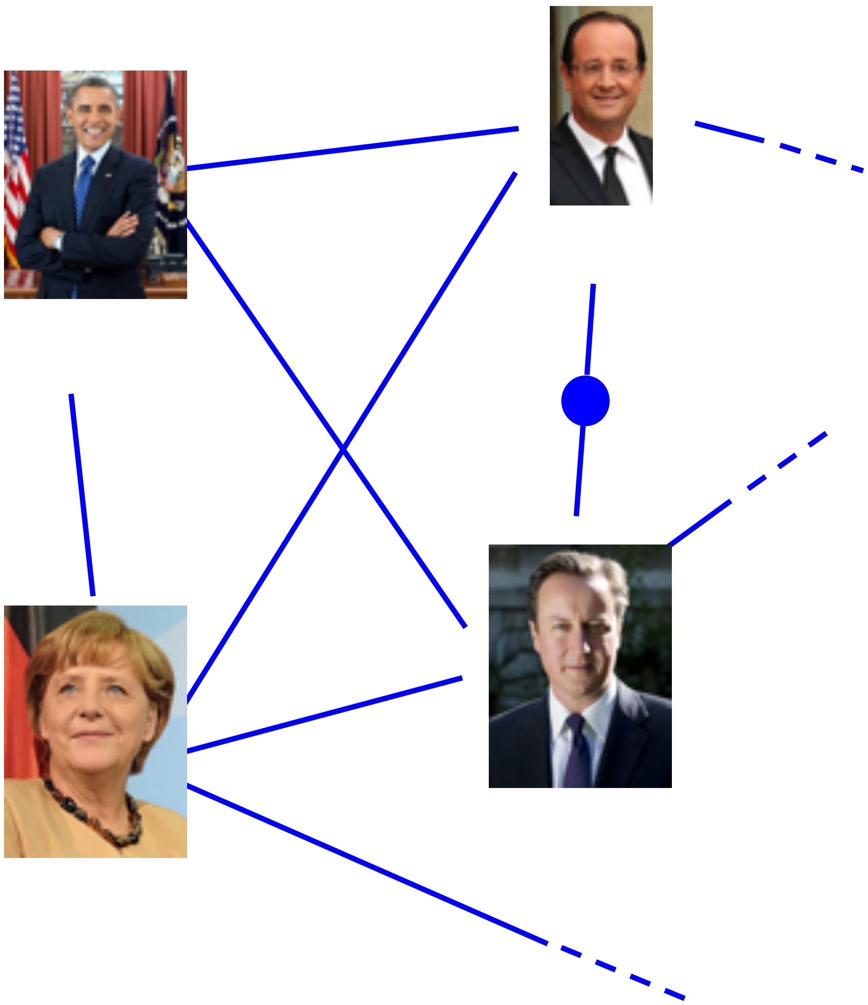
Mauro Sozio and Aris Gionis. The community-search problem and how to plan a successful cocktail party.

KDD 2010.

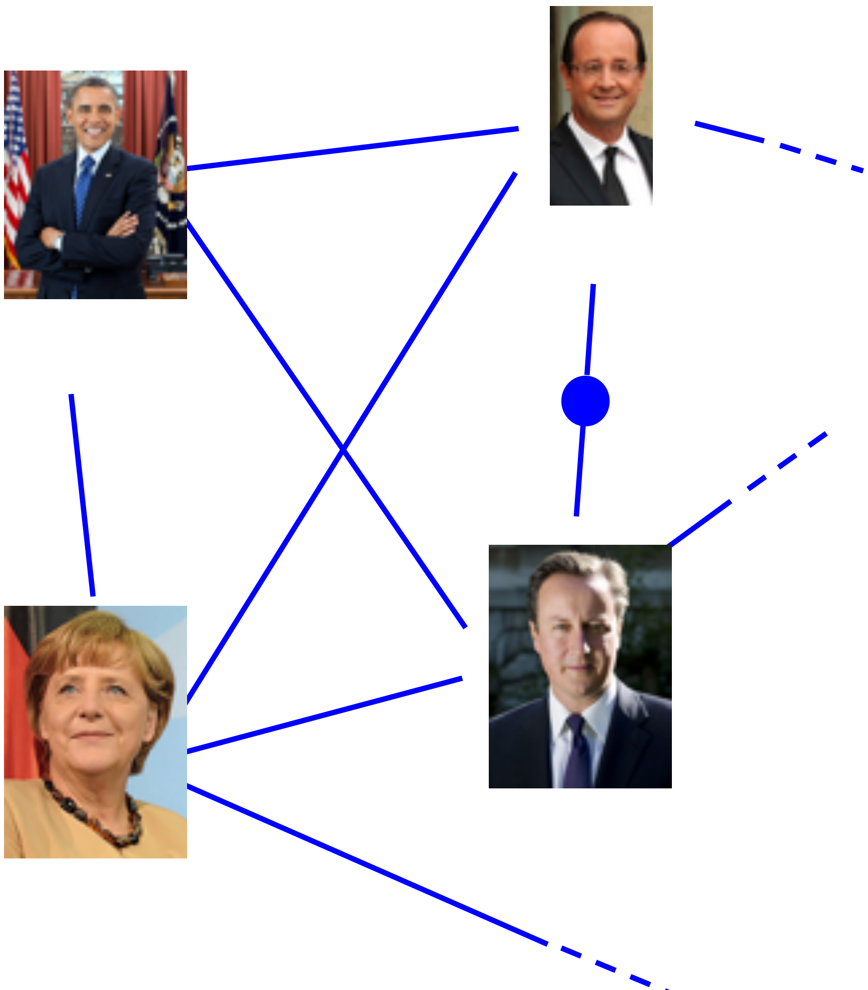
Adapted from the KDD 2010 talk slide of Mauro Sozio.

Planning a cocktail party

Planning a cocktail party



Planning a cocktail party



Recipe for a successful party: 

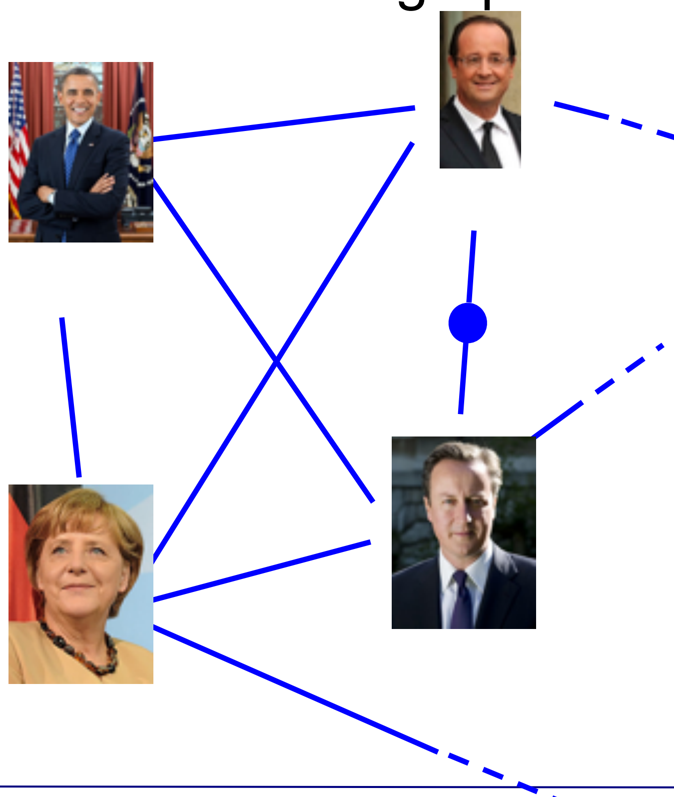
- Participants should be “close” to the organizers (e.g., a friend of a friend).
- Everybody should know sufficiently many in the party (on an average?).
- The graph should be connected.
- The number of participants should not be too small but…
- …not too large either!!!
- ….
- social distance not too large.

Not an easy task…

The community-search problem

- **The problem**: find the community that a given set of users belongs to.

- **Authors' formalization:** Given a graph and a set of nodes, find a densely connected subgraph containing the set of users given in input.

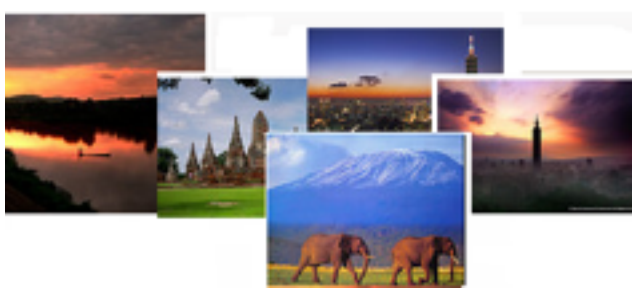


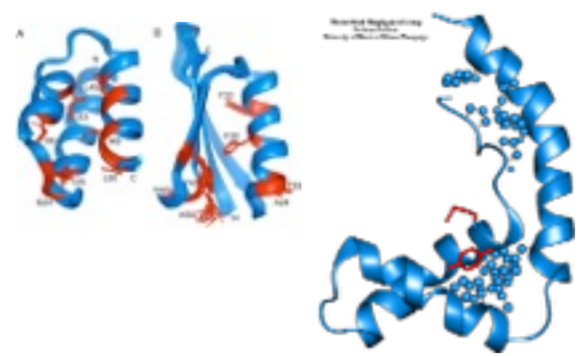
The community-search problem

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

Authors' formalization: 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

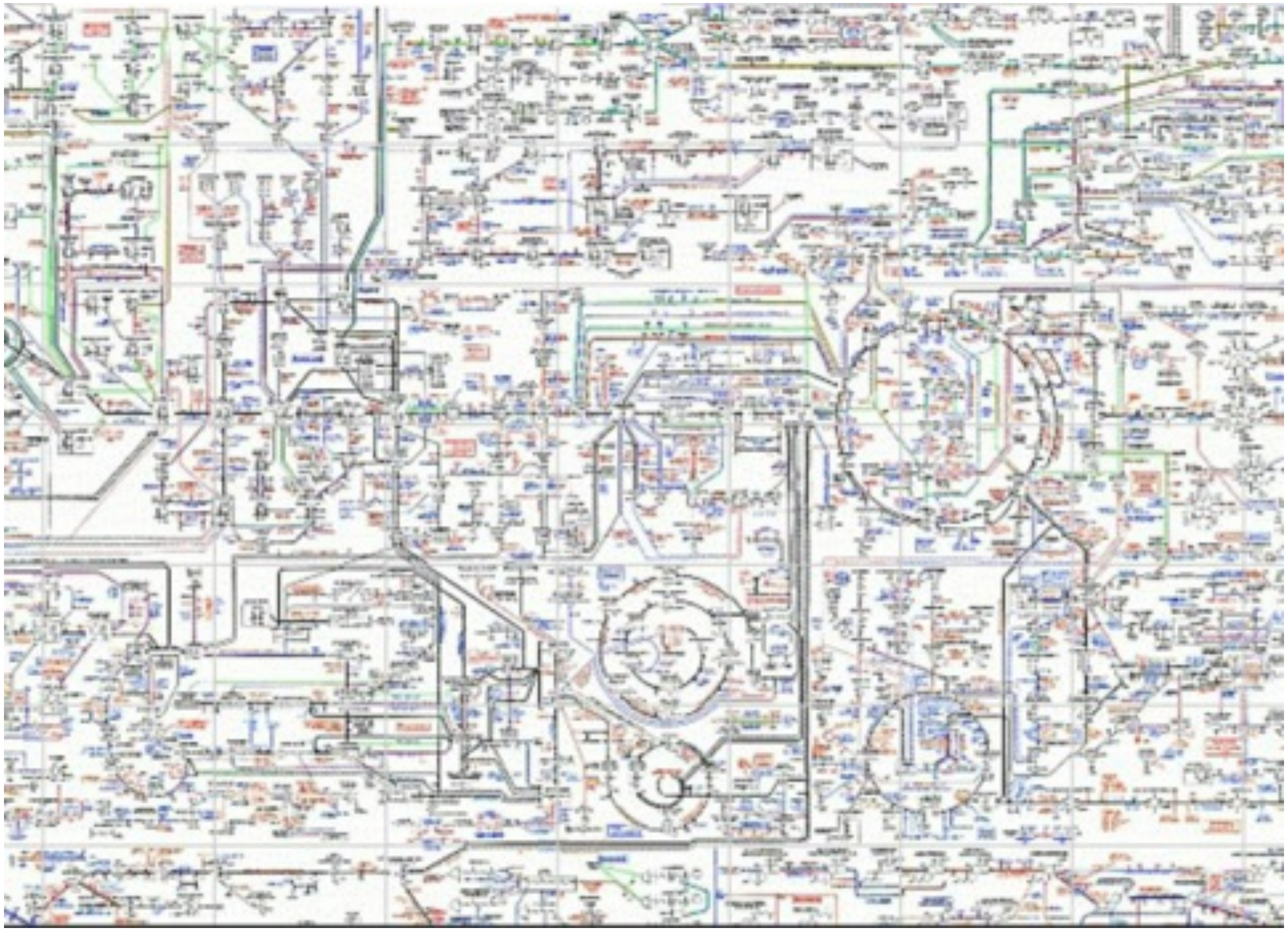
Nature

Landscape

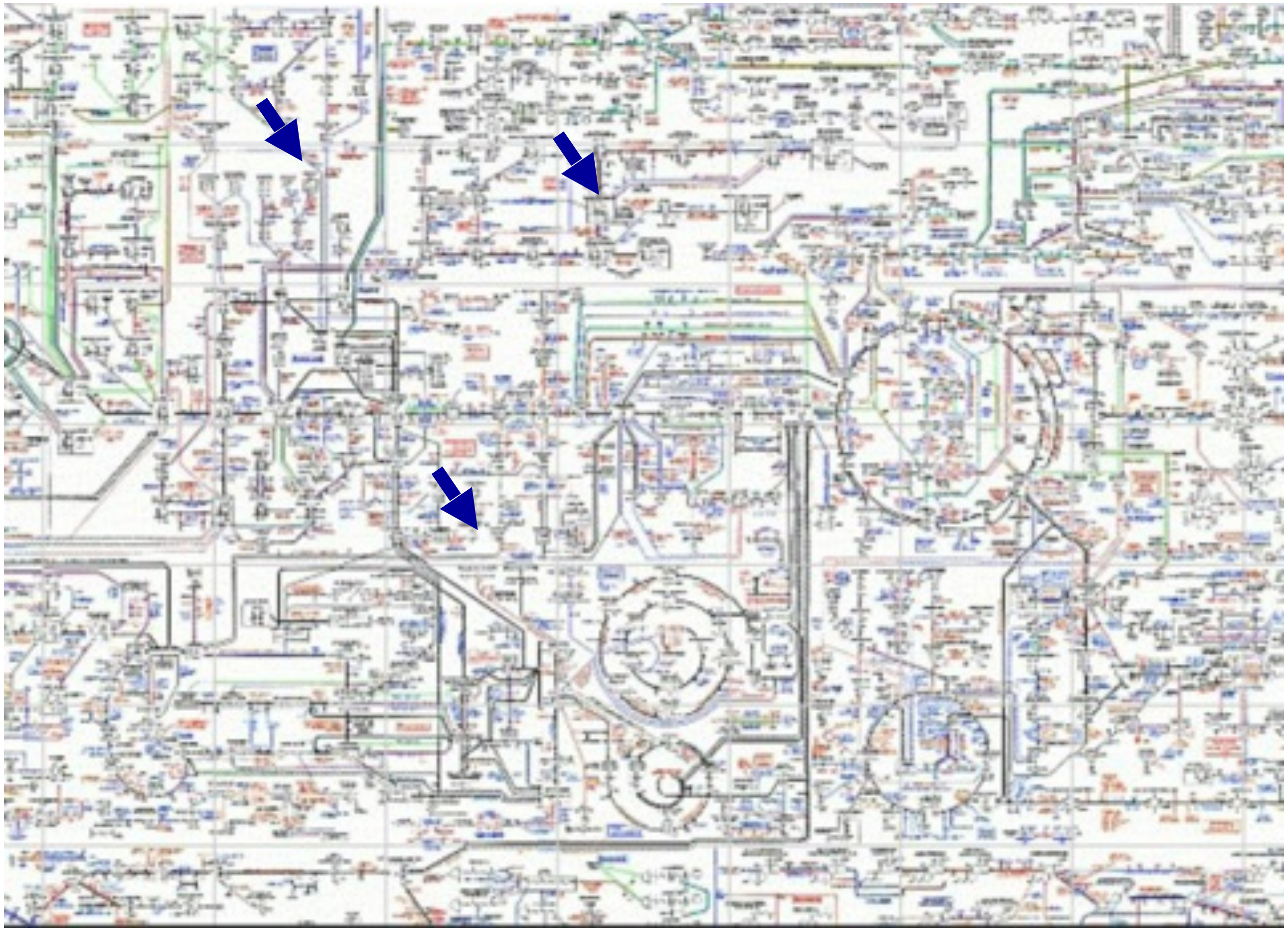
Tag suggestions

- Graph of tags: tags ti and tj connected if they co-occur in many photos.
- given a new photo (or any resource) and initial set of tags, recommend new tags to add.
- tags well connected with one another and the initial set of tags — good candidates.

Protein interactions



Protein interactions



Protein interactions

- Given: Protein-protein interaction network.
- A set of proteins that regulate a gene that a biologist wishes to study.
- what other proteins should she study?
  - those contained in a compact dense subgraph containing the original proteins.

Related 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)

This paper: Query-dependent variant of the problem.

Other related work:

- Y. Koren, S. C. North, and C. Volinsky (TKDD, 2007): cycle-free effective conductance.
- H. Tong and C. Faloutsos (KDD, 2006): random walk based proximity.
- Lappas et al. (KDD, 2009): team formation.
- FOCS, ICALP, APPROX

Problem Definition

Abstract problem definition

- Input: Undirected graph G = (V,E); a query set of nodes Q $\subset$ V and a "goodness" function f that says how good an answer is.
- Find a connected subgraph H = (V_H, E_H) s.t.:
  - $Q \subseteq V_H$ and
  - $f(H)$ is the maximum possible among all connected subgraphs H containing Q.

what are some good choices for f?
want f to capture density.

Some choices of density measure

$n =$ #nodes; $m =$ #edges. Only undirected graphs in this paper.

Good properties: small distance, large density, good connectedness.

Two definitions of density of a graph

- d(G)=# of edges in G / max # possible

  Formally, $m/[n(n-1)/2]$

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

  Formally $\frac{m}{n}$ <— average degree/2.

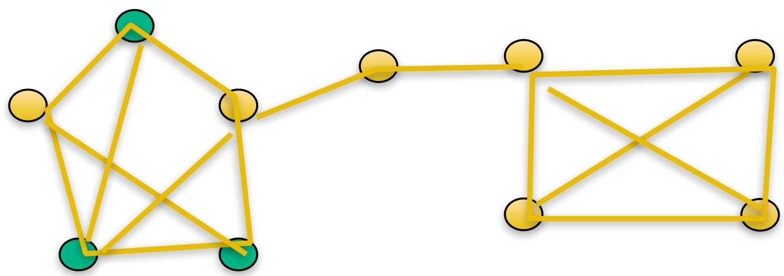
Some choices of density measure

Claim 1: Computing a subgraph H with maximum density d(H) is NP-hard.

Proof Sketch: By reduction from Max Clique.

Some choices of density measure

Fact 2: Computing a subgraph H with maximum density D(H) can be done in polynomial time but avg. degree based f can lead to counterintuitive results.



Free riders problem.

=> choose *minimum* degree instead.
Do any problems persist?

Additionally impose a bound on max. distance of nodes in H to query nodes.

Nothing sacred about squaring distance here.

$$D_Q(H) := \ max_{v \in V_H}(\sum_{q \in Q} d^2(v,q)) \leq \Delta$$

Could use sum instead of max or vice versa.

Final problem definition

- Input: An undirected graph G = (V,E); query nodes $Q \subset V$; distance bound Δ.
- Find a connected subgraph H = (V_H, E_H) s.t.:
  - $Q \subseteq V_H$;
  - $D_Q(H) \leq \Delta$;
  - and f(H) := ***min. degree of H, is maximized.***

Good news: The optimal solution can be found in poly time!

The algorithms

A greedy algorithm

1. Let $G_0 = G.$ fix constraint violations.
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, upon saturation.
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.

- No need to iterate once Q is no longer contained or connected.

A greedy algorithm

1. Let $G_0 = G.$
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, upon saturation.
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: The greedy algorithm computes an optimum solution for the community-search problem.

Optimality of Greedy (w/o distance constraint)

- Let G=G0, G1, …, GT be the series of graphs obtained from G by removing the min. deg. node and its incident edges, until that min. deg. node is in Q or its removal disconnects Q.

- Let G* be an optimal solution.
- Let t be the smallest number for which the min. deg. node v in Gt, is in G*.
- $\Longrightarrow$ G* $\subseteq$ Gt' $\subseteq$ Gt, where Gt' is a connected component of Gt.
- deg_G*(v) <= deg_Gt'(v).
- v is the min. deg. node in Gt and hence of Gt', so Gt' is an optimal solution! QED
- w/o distance constraint, can be implemented in O(n+m) time (see paper).

Optimality — general case

- Paper claims same logic holds for any monotone constraints.
- However, there are some issues to be resolved there.
- Here is the **essence of monotonicity**: G=(V,E) and H=(V',E') an induced subgraph. f maps graphs to reals is monotone if for every graph G and induced subgraph H, $f(H) \leq f(G)$.
- Or f could be monotone non-decreasing instead: $f(H) \geq f(G)$.
- When f is boolean, you get a property (or constraint) instead.
- **Examples:**
  - $D_Q(.) \leq \Delta$, i.e, the max. aggregate distance of any node to the query nodes is bounded, is a monotone constraint.
  - If G satisfies it, so will any induced subgraph containing Q.
  - The distance bound constraint remains monotone if distances to query nodes aggregated using max instead.

Optimality in the general case

- f(G) =1 iff G contains Q and is connected, is monotone. If G fails, so will any induced subgraph.
- Unfort., bound on min. degree (Ex. 2 in paper) is **not** monotone.
- Requiring nodes of a graph to cover a given set of skills (a la Team Formation paper) is monotone.
- See paper for similar def. of node-monotone, a finer grained notion of monotonicity.
- **General Cocktail Party Problem:** Given query nodes Q and graph G, you want to find a connected subgraph H containing Q that maximizes f(.), among all such subgraphs which satisfy given monotone properties: say $\Pi_1, ..., \Pi_k$.
  - paper claims an obvious generalization of greedy for this setting is optimal.

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 an upper bound on the number of nodes makes the problem NP-hard even w/o a distance constraint (reduction from Steiner Tree) but...

Theorem: Let H and H' be two graphs obtained by executing the greedy algorithm with distance constraint Δ and Δ', respectively (the other input parameters are the same).

Then, $\Delta' \leq \Delta$ implies $|V(H')| \leq |V(H)|$.

GreedyDist

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

GreedyDist:

- solve the problem w/o the cardinality constraint on #nodes.
- if size <= bound, report;
- else successively try with tighter distance constraints (can use binary search!).
    - report any small (i.e., size <= bound) connected subgraph containing Q, if found.
    - else report smallest connected subgraph found that contains Q.

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 Δ be a distance constraint (i.e., bound).
- Preprocessing: consider only the k' closest nodes to the query nodes, where k' is the smallest number that ensures the resulting graph is connected and contains k nodes.
- Run Greedy with the subgraph induced by these query nodes, as input

Evaluation

Evaluation

Algorithms evaluated 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.

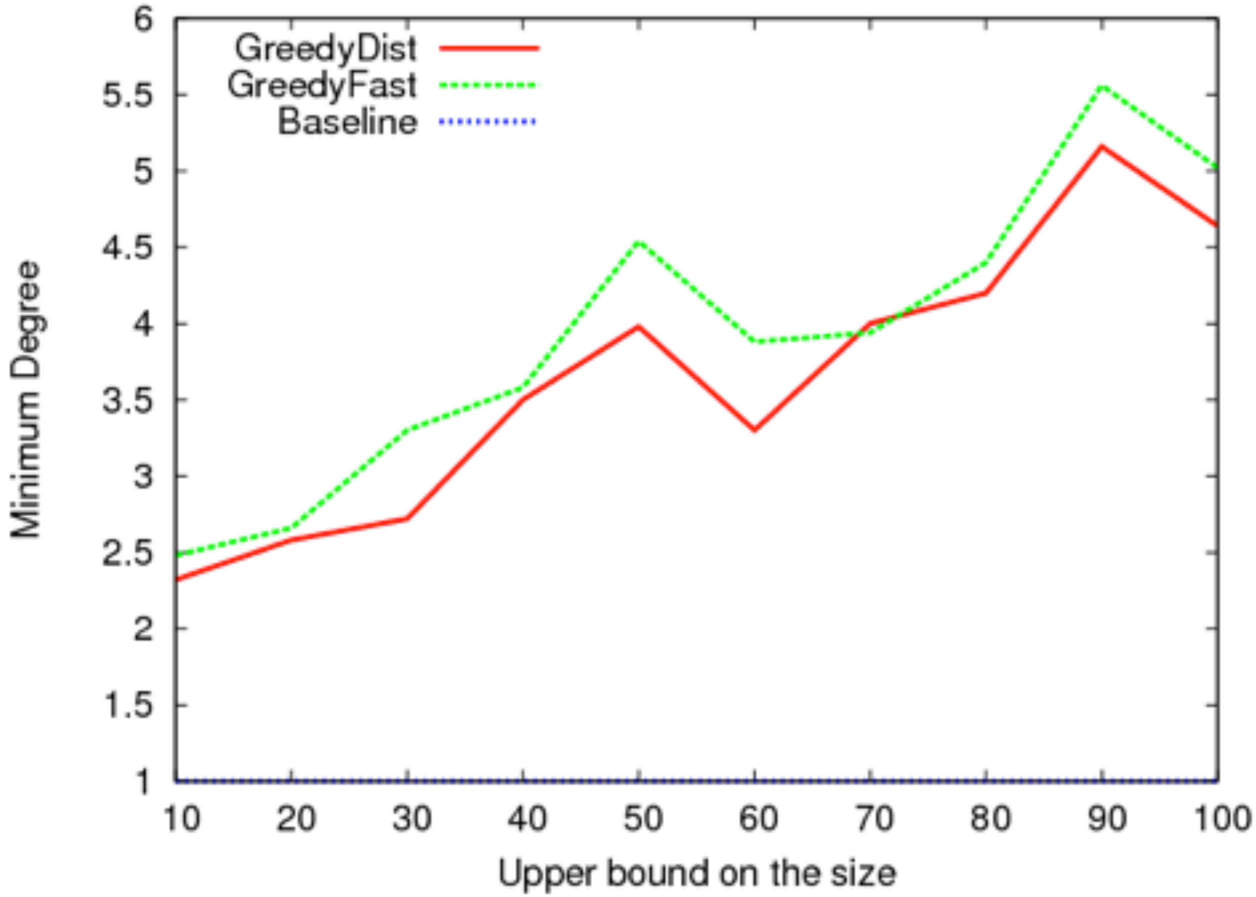
Baseline

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

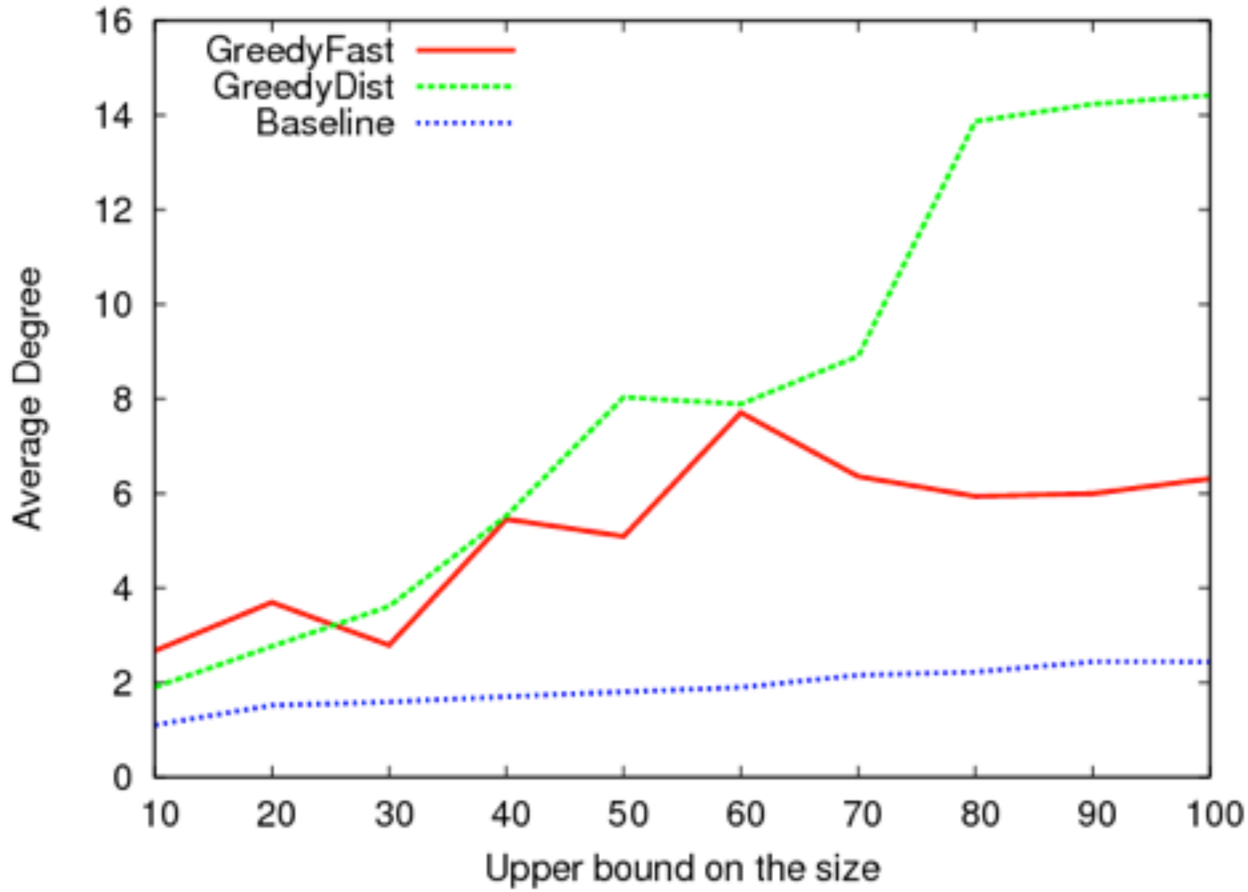
A pseudocode:

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 $G_t \cup v$;
   1. Break ties using distance to Q and further ties arbitrarily.
4. Among all the graph $G_0,\ldots,G_T$ constructed, return the one with maximum minimum degree.

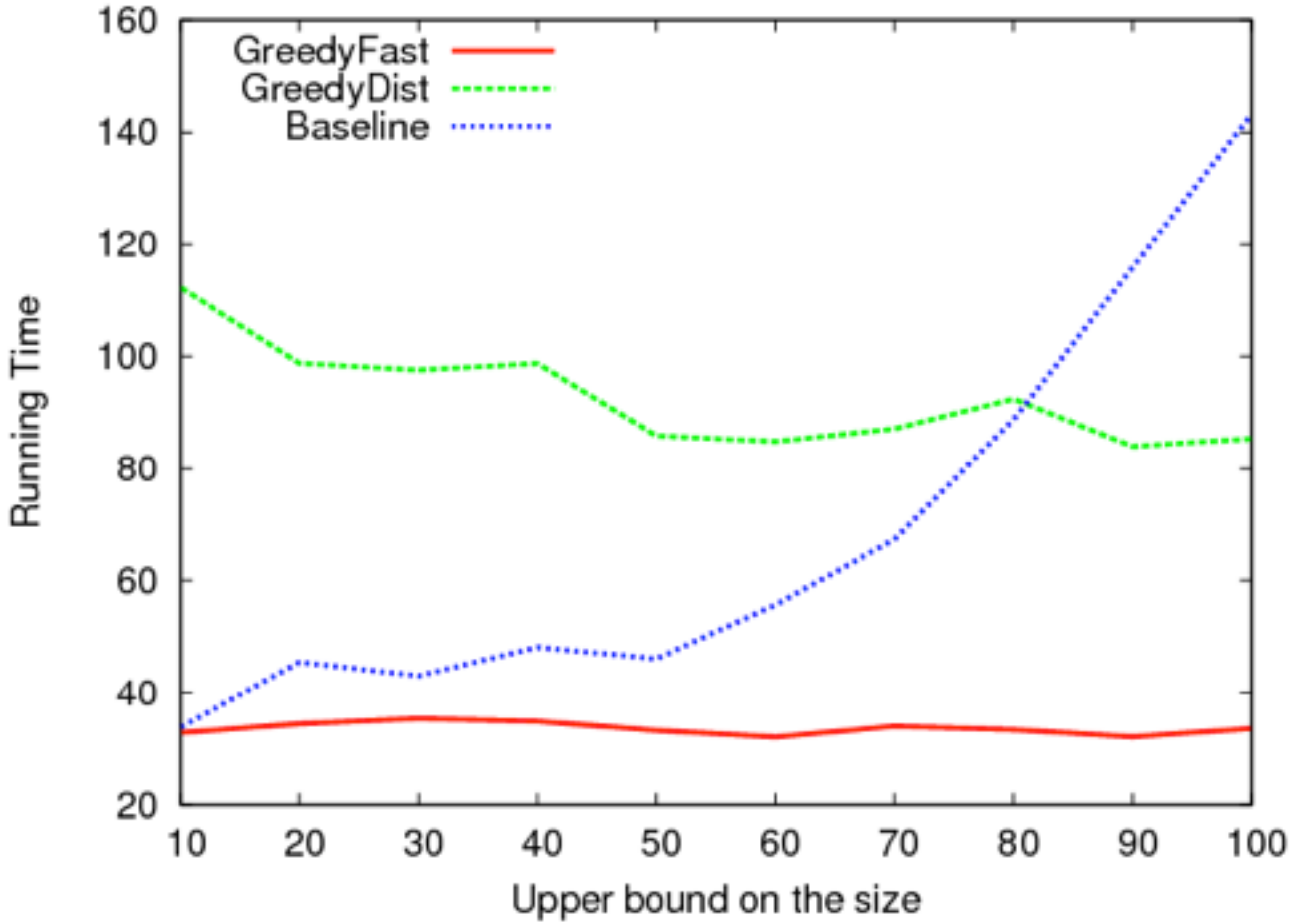
Minimum degree vs Size (Flickr)



Average deg. vs. Size (Flickr)



Running time vs Size (Flickr)



Generalization to monotone functions

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 $\bar{T}$ ($\bigcup_{v \in H} T_v \supseteq \bar{T}$);
- 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* $\bar{T}$ *(* $\bigcup_{v \in H} T_v \supseteq \bar{T}$ *);*
- Any node is at distance at most k from the query nodes;
- **The minimum degree is maximized.**

Monotone functions

The last one is not monotone but poses no problem.
Skill containment — how do you incorporate that in a node elimination paradigm?

Generalized Greedy: Guarantees

Monotone function: $f(H) \le f(G)$, 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, …, 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

Conclusions

Conclusions and Future Work

Contributions:

- 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.
- Defined a class of functions that can be optimized efficiently.

Questions:

- Are there other useful monotone functions?
- Can we find all communities of a given set of users?
- Community search via Map-Reduce?
- What about other dense subgraphs such as k-core, quasi-clique, k-plex, containing given query nodes?