Connection and Center-Piece Subgraphs

Janne Toivola
jatoivol@cis.hut.fi

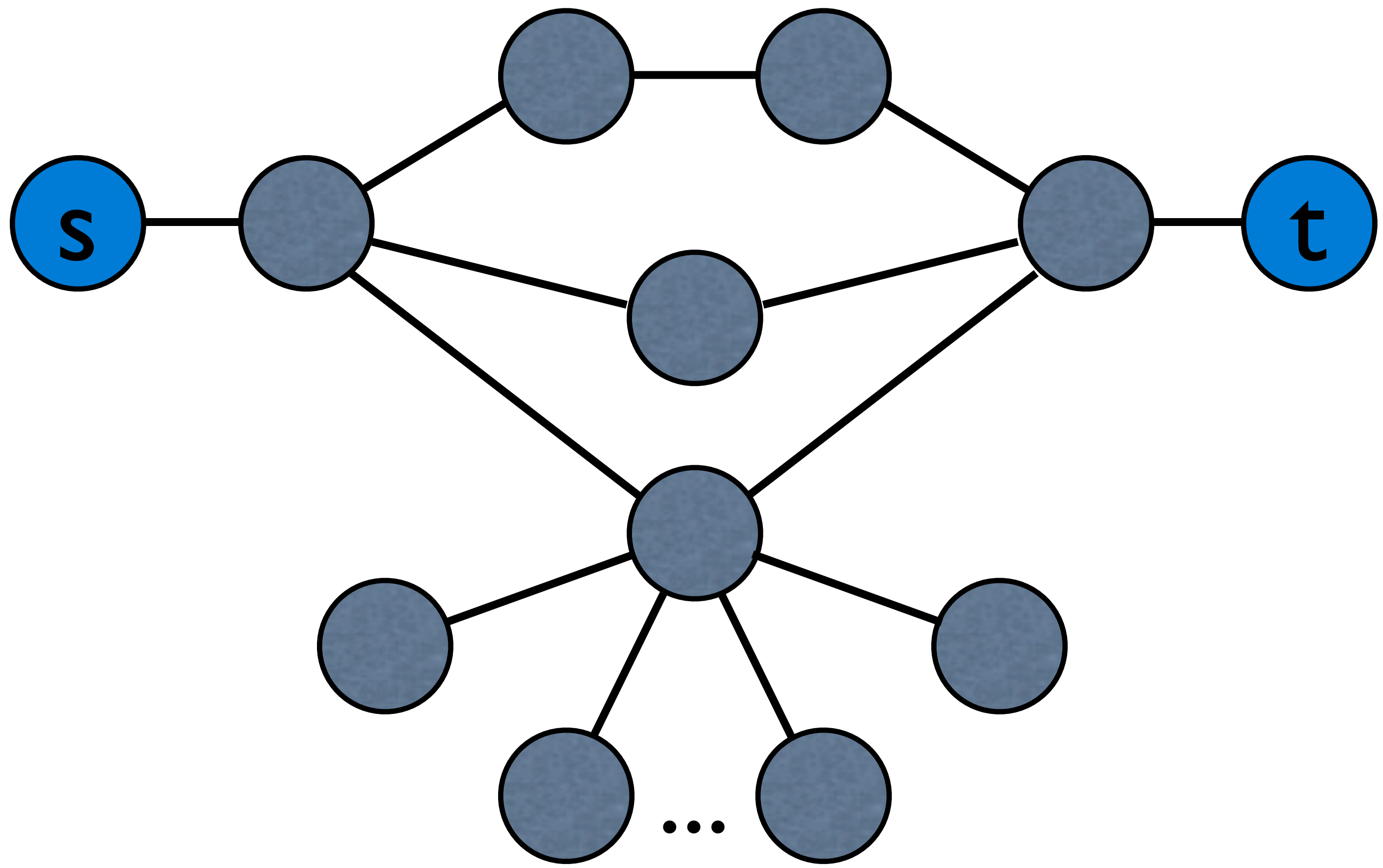
Connection subgraphs

- "Fast Discovery of Connection Subgraphs" by C. Faloutsos, K. McCurley, A. Tomkins
- subgraph showing connections between two given nodes (*s* & *t*) in a larger graph
- interesting to find relationships in vast social networks
- visualization limits manual exploration

Conventional methods

- Connections have been measured before:
- shortest path (Dijkstra, A* etc.)
- maximum flow (Ford-Fulkerson etc.)
- survivable networks: number of edge-disjoint paths
- Not very suitable for social networks

Examples



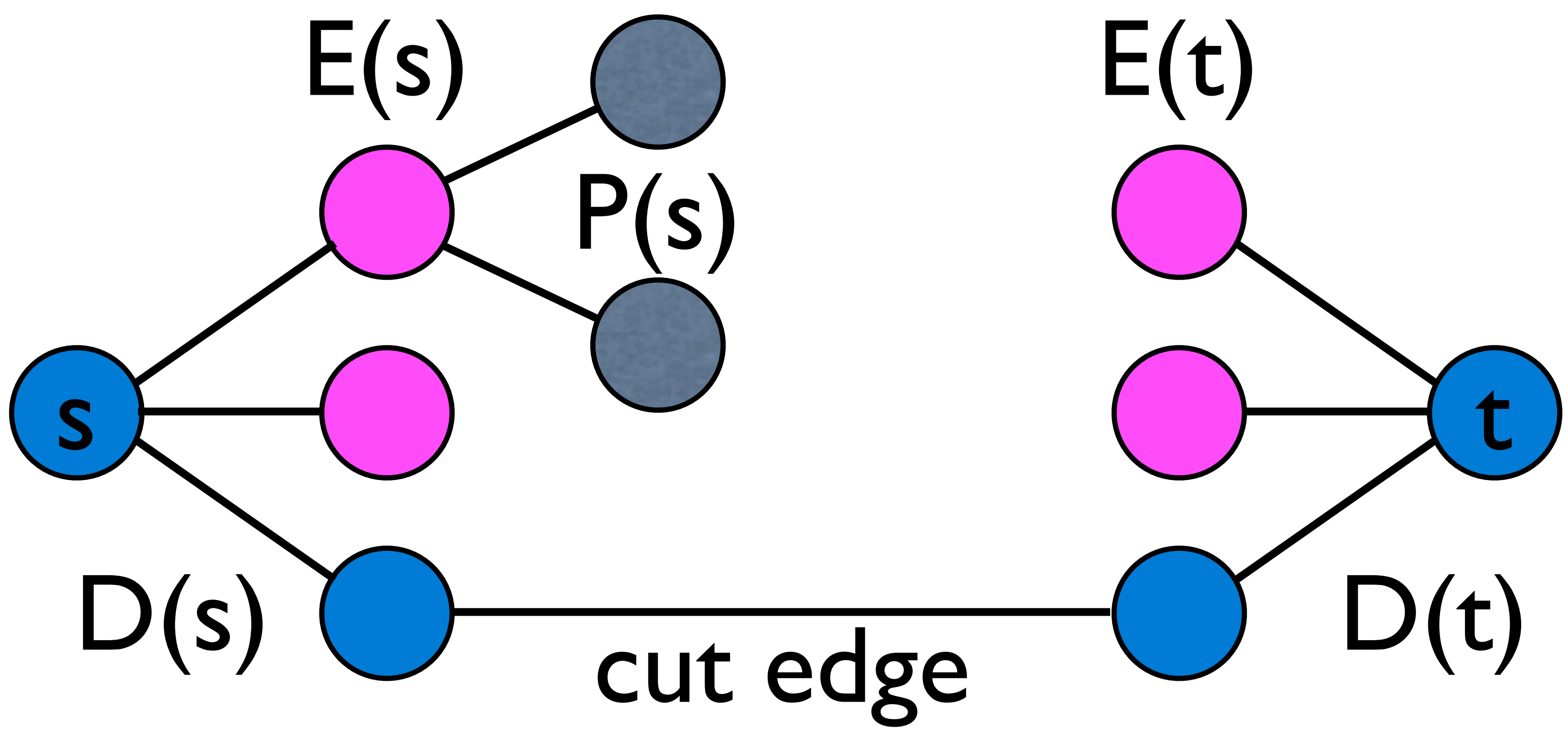
Proposed method

1. Find a smaller *candidate graph* to avoid computational burden
2. Use the electrical concept of *delivered current* to model intuitive connections
3. Find a subgraph of limited size and maximal delivered current for display purposes

1. Candidate graphs

- Quick preprocessing to speed up the rest
- finds most important connections by carefully growing neighborhoods of the query nodes *s* and *t*
- discovered nodes = D(s) and D(t)
- expanded nodes = E(s) and E(t)
- pending nodes = P(s) and P(t)

Neighborhoods



pickHeuristic

- Selects which node to expand next
- take the one with "shortest" path to root
- many ways to define "shortest"
- degree-weighted, count-weighted, and multiplicative properties of length

$$l = deg(u)/C(u,v)$$

$$l = \log(deg^2(u)/C(u,v)^2)$$

stoppingCondition

- Stop when D(s) and D(t) overlap enough
- limit total expansions (disk access)
- limit discovered nodes (memory)
- limit the number of cut edges (connectedness of D(s) and D(t))

2. Electrical model

- Model the network using concepts from electrical circuits
- voltage, current, conductance etc.
- source node *s* has +1 volts, sink *t* has 0V
- weight of edges ~ conductance
- current will flow from source to sink

Other analogues

- Hydraulics : pressurized liquid flowing thru network of pipes of various diameter
- Random walk : a model related to electrons
- find the paths which take random walkers from source to destination

Elementary physics

- Ohm's law: $I(u,v) = C(u,v)(V(u) - V(v))$
- Kirchhoff's 1st law: $\forall v : \sum_u I(u,v) = 0$
- => set of linear equations
- solved in O(n^3)

Network modifications

- Universal sink added to better match the social network domain
- ≈ grounding nodes relative to their degree
- high degree nodes and long paths penalized
- concept of *delivered current* required since part of the total current gets lost

$$\hat{I}(s,u) = I(s,u)$$
$$\hat{I}(s,...,u_i) = \hat{I}(s,...,u_{i-1})\frac{I(u_{i-1},u_i)}{I_{out}(u_{i-1})}$$

3. Display graphs

- Greedy heuristics to find subgraph of given size to maximize delivered current
- Starts with an empty graph and adds paths with highest flow / new node
- Achieved with dynamic programming on topologically sorted (directed) candidate graph

Center-piece subgraphs

- "C-P Subgraphs: Problem Definition and Fast Solutions" by H. Tong, C. Faloutsos
- connection subgraphs had 2 query nodes
- center-piece subgraphs try to describe the community between Q > 2 query nodes
- E.g. find most influential authors related to a set of given researchers in a field

Conventional methods

- Concept of delivered current works only for pairs of nodes
- random walk methods like PageRank etc.
- community detection (remote relations?)
- graph partitioning achieves mostly the opposite thing

Proposed method

- Based on the random walk idea
- random walkers start from each of the query nodes q_i
- steady-state probability score $r(i, j)$ for visiting a certain node j
- score $r(\mathcal{Q}, j)$ for the whole query set $\mathcal{Q}$
- goodness criterion for subgraph $\sum_{j \in \mathcal{H}} r(\mathcal{Q}, j)$

Different queries

- OR, k_softAND, AND
- i.e. how many of the query nodes need to have connections to a target node
- achieved by combining individual scores suitably, e.g. AND: $r(\mathcal{Q}, j) = \prod_{i \in [1, Q]} r(i, j)$
- k_softAND based on meeting probability of k random walkers:

$$r(\mathcal{Q}, j, k) = r(\acute{\mathcal{Q}}, j, k-1) \cdot r(\mathcal{Q}, j) + r(\acute{\mathcal{Q}}, j, k)$$

Solving scores

- Steady-state probabilities become:

$$\mathbf{R} = r(i,j)$$

$$\mathbf{R}^T = c\mathbf{R}^T \times \tilde{\mathbf{W}} + (1-c)\mathbf{E}$$

$$\Rightarrow \mathbf{R}^T = (1-c)(\mathbf{I} - c\tilde{\mathbf{W}})^{-1}\mathbf{E}$$

EXTRACT algorithm

- Like display graph generation: find a small subgraph maximizing score
- Tries to find new key paths from query nodes to most promising destination nodes

$$pd = argmax_{j \neq \mathcal{H}} r(\mathcal{Q}, j)$$

Speeding up

- Graph partitioning used for finding smaller candidate graphs
- select the partitions containing the query nodes

Teh end

- Any questions?
- Thanks for the patience