Dot2Dot

Mining and Visualizing Connection Pathways in Large Information Networks

Leman Akoglu Stony Brook University
Jilles Vreeken University of Antwerp
Hanghang Tong IBM T.J. Watson Research
Polo Chau Georgia Tech
Christos Faloutsos Carnegie Mellon

Dot2Dot is an efficient framework that groups *selected nodes* in a graph and finds *simple connection pathways* among nodes within each group.

Problem

How to make sense of *selected nodes* in a large graph (e.g., anomalies, infected people, activated genes) ?

How are they connected? Are they close by or segregated? How many groups do they form? Are there simple paths to connect nodes in a group? Who are good connectors?

Algorithm

Idea of encoding: We seek to find **easy to "describe"** paths between selected nodes, based on the **Minimum Description Length** principle, so that each **node-2-node** path needs few bits to describe, e.g., avoid high-degree nodes, unless need to visit many of its spokes.

Problem hardness: We show this is an **NP-hard** problem (reduction from the Steiner Tree Problem).

Fast heuristic methods: Our algorithm is based on building **k-level trees** iteratively, where intermediate nodes decrease **encoding cost**, details are in [1].

Interactive Visualization

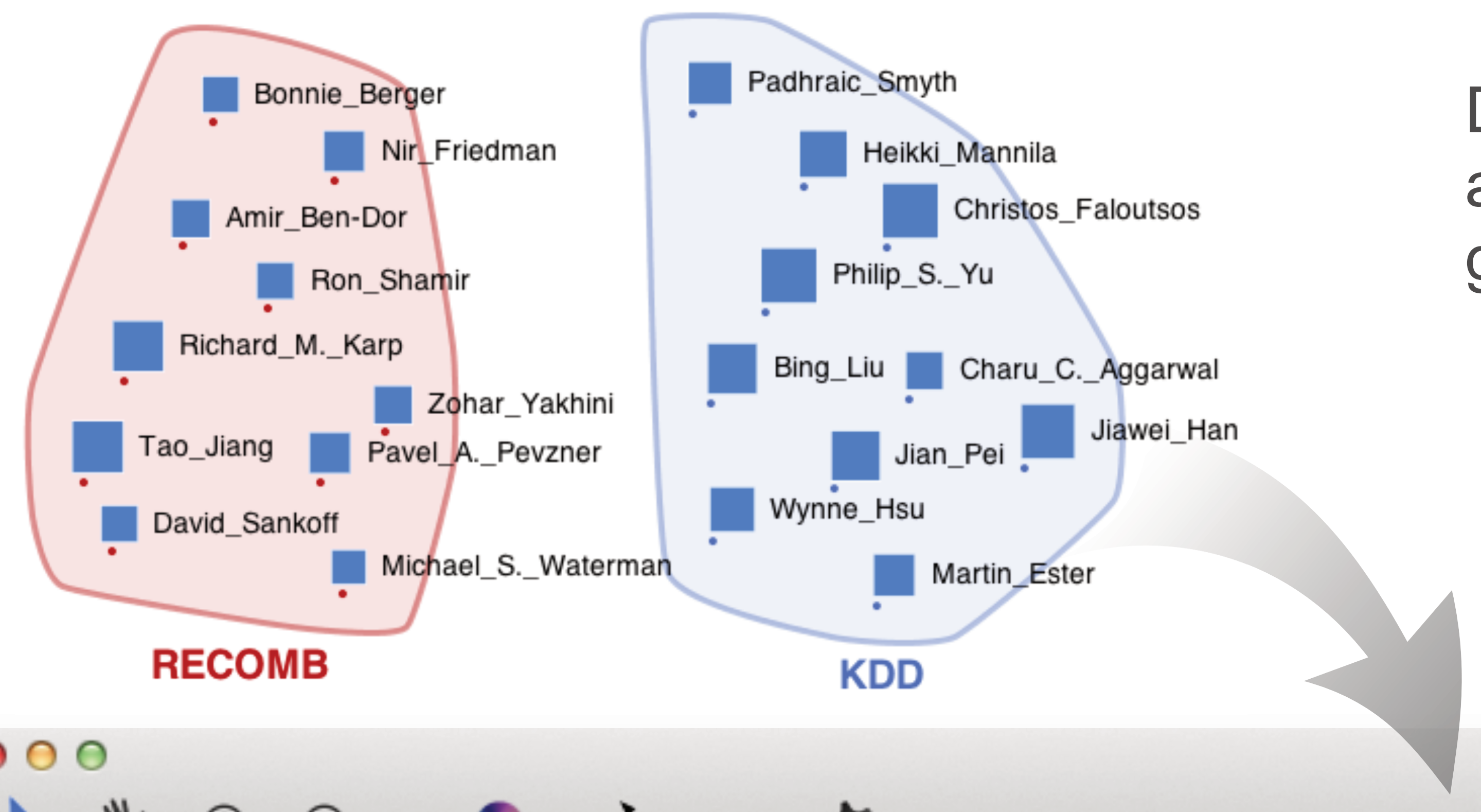
1 **Search. Select.**
Find nodes and drag them into the view.

2 **Select nodes. Go.**
Turn your nodes of interest into squares. Dot2Dot will find simple paths among them.

You can also group nodes visually.

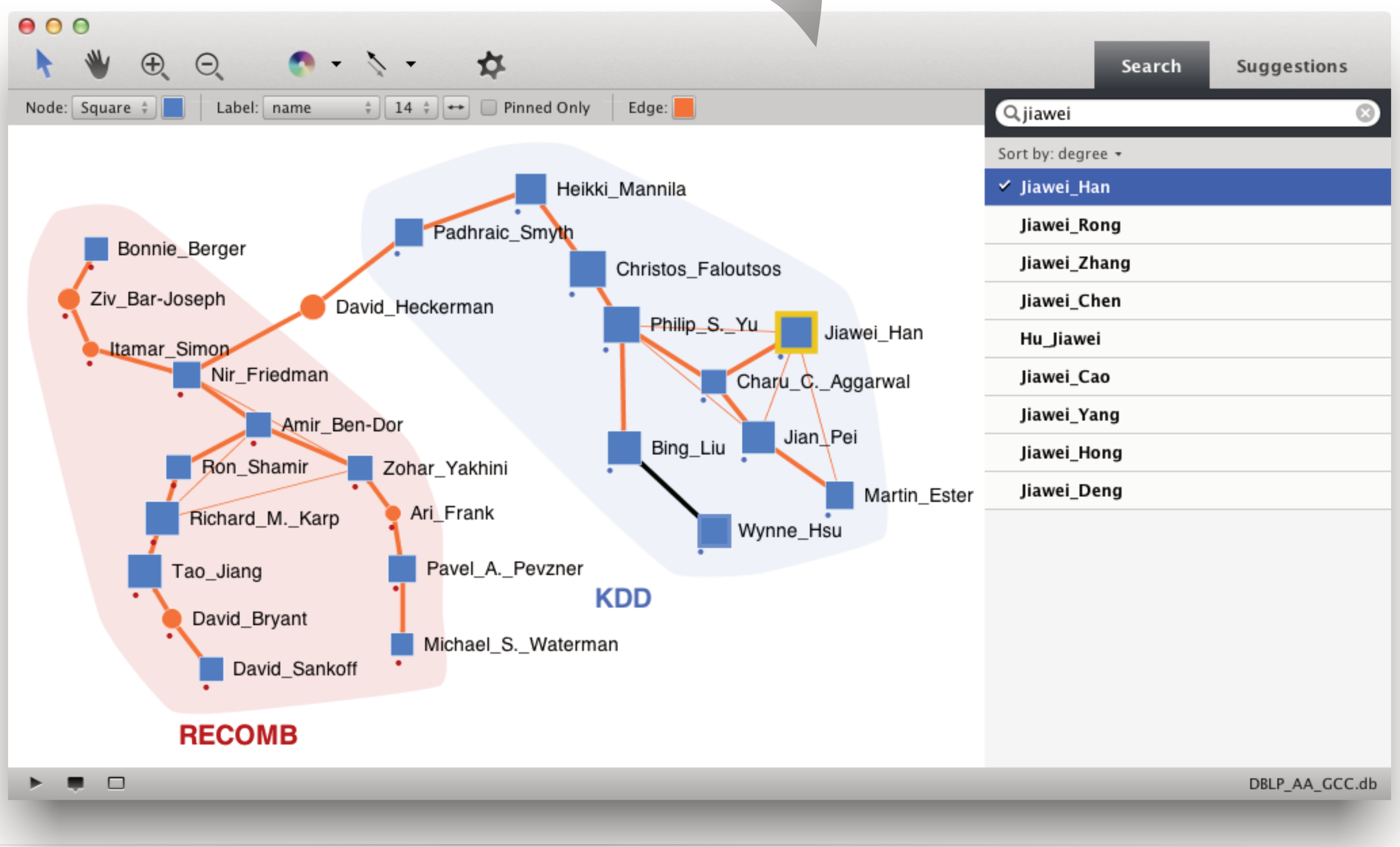
3 **Visualize. Interact.**
Dot2Dot visualizes paths among marked nodes.

You can interact with them: add or delete nodes, mark or unmark them, see their neighbors, and more.



Dot2Dot showing connection pathways among authors from DBLP coauthorship graph (300K nodes, 1M edges).

- **Blue square**: selected nodes
- **Orange circle**: connectors
- **Thick orange edge**: simple path found by Dot2Dot



[1] L. Akoglu, J. Vreeken, H. Tong, D. H. Chau, and C. Faloutsos. Islands and bridges: Making sense of marked nodes in large graphs. Technical Report CMU-CS-12-124, Carnegie Mellon University, 2012.

Visualization implemented in Java, using the JUNG library.
Algorithm written in Matlab 7.10.