Graph Algorithms for Ad Hoc Networks

V. Sunitha
Modelling and Analysis Group of Networks
DA-IICT, Gandhinagar

MAGNeT

Wireless and mobile networks are an excellent playground for graph theoreticians. - R. Wattenhofer

Wireless and mobile networks are an excellent playground for graph theoreticians. - R. Wattenhofer

- Vertex Colouring
- Edge Colouring
- List Colouring
- Independent Set
- Dominating Set
- Graph Spanners - Spanning trees, Steiner trees, Spanning subgraphs
- Random Graphs
- ...

Wireless and mobile networks are an excellent playground for graph theoreticians. - R. Wattenhofer

- Vertex Colouring
- Edge Colouring
- List Colouring
- Independent Set
- Dominating Set
- Graph Spanners - Spanning trees, Steiner trees, Spanning subgraphs
- Random Graphs
- ...

Need of the hour: Distributed dynamic graph algorithms

An approach to solve problems in Ad Hoc Networks

1. Model the network problem as a graph theoretic problem
2. Find an algorithm for the graph theoretic problem
3. Convert to distributed and/or dynamic algorithm

Some graphs that model MANETs

- Communication graph
- Unit disk graph
- Quasi-disk graph
- Random geometric graph
- Interference/conflict graph
- Gabriel graph
- Yao graph
- ...

Topology control and Clustering in MANETs

- Permit each node to communicate with only a selected subset of its neighbours.
- Use a hierarchical structure.

Aim: **To conserve energy**

Clustering based on dominating and/or independent sets

Network requirement	Graph property
For cluster based communication between each pair of nodes, every node should have at least one cluster head in its neighbourhood	Find a (minimum) dominating set
For routing using clusters, the cluster heads must be connected	Find a (minimum) connected dominating set
For cluster based MAC, cluster heads should not be in mutual transmission range	Find a (maximal) independent set (Note: not maximum)
Different nodes have different costs/priorities for serving as a cluster head	Find WCDS

Dominating Sets for Unit Disk Graph

MDS for UDG

1. Each node v randomly selects an ID $ID(v)$ from the interval $[1, \cdots, n^2]$.
2. Nodes exchange $ID(v)$ with all neighbours.
3. Each node v elects the node w in its neighborhood with the largest ID.
4. All nodes that have been elected at least once become dominator.

Dominating Sets for Unit Disk Graph (contd.)

CDS by connecting an MIS for UDG

- The size of any IS of an UDG G is at most 5 times the cardinality of the optimal MDS in G.
- The size of any IS of an UDG G is at most one more than 4 times the cardinality of the optimal MCDS in G.

So, we can use an MIS as an approximation for a MCDS in the UDG.

Assume: Each node has a unique ID.
Each node v having $ID(v)$ waits with its decision until one of the two following conditions hold:

- v has a neighbour node that joins the MIS $\Rightarrow$ v becomes covered.
- v has the largest ID among all uncovered neighbours $\Rightarrow$ v joins the MIS.

Dominating sets for general graphs

Popular centralized approaches for finding CDS

- Grow a specially designed spanning tree and then the internal nodes of the spanning tree are selected as the CDS.
- Find a DS and then connect the DS to a CDS (may be by using a Steiner tree algorithm).

Dominating sets for general graphs (contd.)

Constructing a special spanning tree

Outline: Grow the tree T starting from the vertex of the maximum node degree. At each step, pick a vertex v (or a pair of adjacent vertices u, v) and "scan" it (or them). When scanning a vertex, add edges of T from it to all its neighbours that are currently not in T.

```
1 At the start of the first phase, all nodes are colored white.
2 while there is a white node left do
3     Find the vertex $w$ that has the largest number of white neighbors
      among all vertices and also the pair of adjacent vertices $u$ and $v$ such
      that the union of their white neighbors has the largest size among all
      adjacent vertices $u$ and $v$.
4     Scan vertex $w$ or the pair of vertices $u$ and $v$, whichever will have the
      largest yield. When scanning a vertex, we mark its white neighbors to
      gray and mark itself to black.
5 end while
```

A centralized algorithm for finding WCDS

To construct a WCDS for an arbitrary node-weighted graph G

1. Apply a weighted set-cover approximation algorithm to find a WDS. (The weighted set-cover instance is created by making each vertex an element, and each vertex corresponds to a weighted set that contains all its one-hop neighbours, in addition to itself. The weight of the corresponding set is the weight of this vertex.)
2. Connect the vertices in the DS selected above by using a node-weighted Steiner tree (NST) approximation algorithm (To apply the NST, the weights of all vertices in the DS are made equal to zero.), so that the DS and the Steiner nodes form a CDS.