A Platform for Comparing CDS Discovering Algorithms

M.S. PROJECT REPORT
HUAPING DING
ADVISOR: DR. LI
COMMITTEE MEMBER: DR. HU
OCT. 25TH, 2011, 5PM
14TH CONFERENCE ROOM, 34 PEACHTREE
CS@GSU

Problem

- Connected Dominating Sets (CDS)
  - Important in many applications, e.g., wireless sensor network
- Too many existing algorithms
  - Famous ones: Alzoubi’s, Wu Jie’s, R-value (Dr. Li’s group)
  - Hard to compare their performances as they are implemented in different platforms and languages
- Question:
  - How to provide a research facility to easily simulate, test, and compare various CDS algorithms?

Goal

- Implement a platform with friendly GUI
  - to simulate each step of CDS identification procedure for each CDS algorithm
  - to perform a batch of experiments for multiple algorithms
    - to randomly generate many valid graphs for testing
    - to report the average performance of different algorithms

Background

- Dominating sets:
  - A dominating set for a graph \( G = (V, E) \) is a subset \( D \) of \( V \) such that every vertex not in \( D \) is joined to at least one member of \( D \) by some edge.
- Connected Dominating sets
  - A connected dominating set (CDS) is a subset of the nodes such that it forms a DS and all the nodes in the DS are connected.

Outline

- Problem and Goal
  - Develop a platform for CDS algorithms comparison
- Background
  - DS, CDS, MCDS
- Three Algorithms
- Design Guidelines
  - OS-independent, MVC, OO design
- Design Architecture
  - Packages, UML
- Demo
- Conclusion

NP-complete problem!
Background

One example of CDS application
- CDS is used as a virtual backbone in wireless networks.
- Network broadcasting:
  - Only nodes in CDS relay messages
  - Reduce communication cost
  - Reduce redundant traffic

Three MCDS Algorithms

- Alzoubi et al. Algorithm
- Wu’s Algorithm
- Our R-value Algorithm

Alzoubi’s Algorithm

1. Construct a spanning tree
   - Assign level 0 to root node \( v \)
   - Propagate message and assign level for each node

2. Find MIS nodes in even levels
   - Mark nodes to black based on its rank; others are gray
   - Rank priority: level, id(i)

3. Find connectors in odd levels to form a CDS
   - A connector should connect to 2 unconnected black nodes, mark as blue
   - All black and blue nodes are dominators

Wujie’s Algorithm

Open neighbor set \( N(v) \), close neighbor set \( N[v] \)

1. Marking process
   - All nodes are initially white
   - Check each node \( v \): if two neighbors of \( v \) are unconnected, mark \( v \) black
   - All black nodes form a CDS

2. Pruning process
   - If \( u, v \) are black and \( N(v) \cap N(u) \), unmark \( v \)
   - If \( u, v, w \) are black and \( N(v) \cap N(u) \cap N(w) \), unmark \( v \)

r-CDS algorithm

Definition of r-value and deg
- \( d(v) \) = number of 2-hop neighbors \( - d(v) \)
- \( deg(v) \) = number of neighbors
- Rank priority is triplet \( -r, deg, id(i) \)

1. Marking step
   - Check each node \( v \), mark it black if it has lowest rank among neighbors
   - Mark black node’s neighbors grey

2. Connecting step – find connectors to form CDS
   - Mark a grey node black if it connects 2 unconnected black nodes
   - Mark two grey nodes black if they together connect 2 unconnected black nodes

r-CDS algorithm

3. Pruning step
   - For two black nodes \( u, v \), mark u grey if
     - \( N(u) \cap N(v) \) or
     - \( N(u) = N(v) \) and \( id(u) < id(v) \)
   - Mark = grey if \( u \) has \( v \) black neighbors \( v, w \), such that:
     - \( N(u) = N(v) \cup N(w) \) – \( \{v\} \)
     - \( N(u) \cap N(v) \cap N(w) \) – \( \{u, v, w\} \), and \( id(u) \) is smallest
   - Mark u grey if \( u \) has 3 black neighbors \( x, y, z \), such that:
     - \( N(u) \cap N(x) \cup N(y) \cup N(z) \) – \( \{u\} \)
     - \( N(u) \cap N(x) \cap N(y) \cap N(z) \) – \( \{u, x, y, z\} \), and \( id(u) \) is smallest
Design Requirements

- **Graph Generation**
  - Manual graph construction
  - Automatic graph construction
- **Algorithm**
  - Implementation (initially 3 algorithms)
  - Parameter specification
- **Simulation**
  - Step mode simulation
  - Simulation control: forward/pause/stop
- **Graph Control**
  - Graph Display/Refresh/Clear

Package Information

- **package: algo**
  - Implementation of algorithms, and threads for batch runs
- **package: algo.rule**
  - Implement the Runes pruning for Wu's algo
- **package: model**
  - Implement node, edge, graph, random graph generator
- **package: ui**
  - View and control Swing components
  - AlgoPanel, GraphPanel, ...
- **package: util**
  - Recorder, BatchJobThread, ColorHandler, ElementSet, CurveDataWriter

Design Architecture

UML Diagram (algo)

UML Diagram (algo.rule)

UML Diagram (model)
Goal reached: the platform for research on CDS algorithms has been established

- OO design and MVC architecture has made the platform very robust and extensible
- Good design patterns provide a solid foundation for software development
- Future works: continue adding new CDS algorithms
Thanks for your attention!