Network Motifs with MASS

(Midterm Report)

Matt Kipps
10/30/2014
Project Details

Prof Kim:
● Analyze biological data for statistically significant subgraphs (motifs).

Prof Fukuda:
● Evaluate MASS implementation against MapReduce and sequential implementations
What is MASS?

Multi-Agent Spatial Simulation

- designed for simulations
- creates a virtual space over an arbitrary cluster
- abstracts cluster management from programmer
What are Network Motifs?

Statistically significant subgraphs within a network.

Given a motif size $n$,

- Find all subgraphs of order $n$ within the network
- Determine subgraph equivalence and count the frequency (and compare to random network)
- Motifs are those subgraphs which occur most often
Finding Subgraphs with ESU
This project seeks to simulate “crawlers” that move through a network. Replace recursion with spawning new crawlers.

GraphCrawler extends Agent

GraphNode extends Place
Step 0
Initial network

Agents will be searching for a subgraph motif size of 3 nodes.

(For simplicity, this sketch only shows the behavior of the first agent spawned. Another agent will also be spawned at Node 2 and will recover another subgraph not shown here)
**Step 1**
Agent A is spawned at node 1

Agent A has a size of 1, so Agent A will keep searching.

Node 1 has only 1 branch so Agent A will not need to spawn any new Agents.
Step 2
Agent A is now at Node 2, and represents the subgraph of “1,2”

Agent A has a size of 2, so Agent A will keep searching.

Node 2 has 2 branches so Agent A will traverse one, and will create a child clone to traverse the other. This can be seen in the next step.
Step 3
Agent A is now at Node 3, with subgraph of “1,2,3”
Agent B is spawned with subgraph of “1,2” and
directions to move to Node 4

Agent A has a size of 3, so it will terminate itself and deposit the results Node 3.

Note - Agent B must “inherit” the existing subgraph path “1,2” from Agent A.

Notice that Agent B is one step behind Agent A because Agent B must be spawned first and then migrate, before proceeding with the algorithm.
Step 4
Agent A is now terminated, and its subgraph of "1,2,3" is stored at Node 3. Agent B migrated to Node 4, and now has a subgraph of "1,2,4".

Agent B has a size of 3, so now it will also terminate itself and deposit the subgraph results at it’s final node, Node 4.
Step 5
Agent B is now terminated, and its subgraph of “1,2,4” is stored at Node 4.

When the entire network traversal is complete, the MASS-based program collects all data from all network nodes, using the return values through the places callAll() method.
After finding all subgraphs, the subgraphs are collected at the master node.

Then, sequentially:
- the subgraphs are sent to \texttt{labelg} to get canonical labels.
- the canonical labels are counted.
At this point, basic implementation is complete.
Performance
Network size of ~2500 nodes
Analyzing motif of size 5
Parallel Performance Analysis

Combined Execution Time

- Zero Parallelization
- My Program
- Optimal Parallelization

Milliseconds vs. Nodes graph

- X-axis: Nodes (1 to 15)
- Y-axis: Milliseconds (0 to 1,800,000)

Graph showing the combined execution time for different parallelization scenarios.
Next Steps
Conduct Evaluation

- Performance vs MapReduce implementation (identical cluster, possibly Google Cloud)
- Programmability of MASS program vs MapReduce program and sequential program
Prior to conducting evaluation, I will be focusing on optimizing and refining the program.

Primarily:

- Optimizing MASS Agent handling
- Dispersing Agent spawning in algorithm
Agent Population

![Graph showing the population of agents over time, with a peak around the 8th year.](image-url)
One more thing...
One more thing...

Place.java

replaced:

```java
Vector<Agent> agents = new Vector<Agent>();
```

with:

```java
Set<Agent> agents = Collections.synchronizedSet(
    new HashSet<Agent>());
```