Lab 5 Report & Peer Evaluation

Student Name: Dazhi Li

Student NETID: dazhili

Student ID#: 2400330

# 1. Documentation

This is a personal submission on LAB 5 assignments and peer evaluation on final project.

I coded a new program whose name is OneStepMove as my LAB 5 target.

## Lab 5 graph and running procedure

The graph in my Lab 5 is written and designed on my own as shown in fig1.

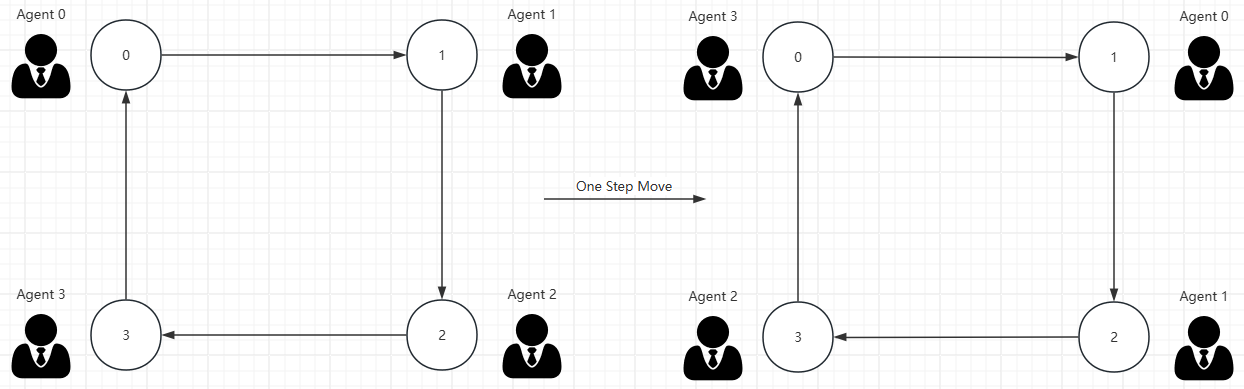


Fig1. Lab5 2D graph and how agent move

Initially, I create new instance of my MoveAgent and spawn the number of agents as the same number of vertices in my graph. By default, each vertex has an agent on it. As my graph is a cycled rectangle, each agent is going to move to the next vertex. That is to say, agent 0 moves to vertex 1, agent 1 moves to vertex 2 and so on.

# 2.Source Code

All source codes are included in the zip file

OneStepMove.java (main)

package edu.uw.bothell.css.dsl.appl.graphs.OneStepMove;

import java.util.List;

import edu.uw.bothell.css.dsl.MASS.Agents;

import edu.uw.bothell.css.dsl.MASS.GraphPlaces;

import edu.uw.bothell.css.dsl.MASS.MASS;

import edu.uw.bothell.css.dsl.MASS.graph.Graph;

import edu.uw.bothell.css.dsl.MASS.graph.transport.VertexModel;

import edu.uw.bothell.css.dsl.MASS.logging.LogLevel;

public class OneStepMove {

    /\*\*

     \* Spawn agent at every vertex and move one step forward

     \*

     \* @param args Input data path

     \*/

    public static void main(String[] args) {

        // Read and validate input parameters

        if (args.length != 1) {

            System.err.println("Usage: java -jar OneSetpMove.jar <dsl\_graph\_file>");

            System.exit(-1);

        }

        String filePath = args[0];

        MASS.setLoggingLevel(LogLevel.DEBUG);

        MASS.init(10000000);

        // loading data from dsl file

        System.out.println("Begin data space generation");

        long begin = System.currentTimeMillis();

        GraphPlaces network = new GraphPlaces(1, NodeGraphMASS.class.getName());

        try {

            network.loadDSLFile(filePath);

        } catch(Exception e) {

            System.err.println("Error reading in graph file: " + e);

            System.exit(3);

        }

        long end = System.currentTimeMillis();

        System.out.println("Import complete\nImport time: " + (end - begin) / 1000.0 + " s");

        // doing one step movement

        run\_one\_step\_move((GraphPlaces) network);

        System.out.println("Finish MASS");

        MASS.finish();

    }

    public static void run\_one\_step\_move(GraphPlaces network) {

        Graph graph = network;

        List<VertexModel> vertices = graph.getGraph().getVertices();

        int nVertices = vertices.size();

        System.out.println("Number of Vertices is : " + nVertices);

        try {

            network.callAll(NodeGraphMASS.init\_);

            // Time measurement starts

            System.out.println("Go! One movement");

            long startTime = System.currentTimeMillis();

            // Instantiate an agent at each node

            Agents crawlers = new Agents(2, MoveAgent.class.getName(), null, network, nVertices);

            crawlers.callAll(MoveAgent.init\_, null);

            int nAgents = crawlers.nAgents();

            System.out.println("\*\*\* Starting Agents = " + nAgents + " \*\*\*\*\*\*\*\*\*\*\*\*");

            // printing status before migration

            int[][] dummyArgs = new int[nAgents][1];

            Object[] status = (Object[]) crawlers.callAll(MoveAgent.show\_location\_, (Object[]) dummyArgs);

            for (int i = 0; i < status.length; i++)

                System.out.println((String) status[i]);

            // Begin first migration

            System.out.println("\*\*\* Begin first migration \*\*\*");

            crawlers.callAll(MoveAgent.one\_step\_move\_, null);

            crawlers.manageAll();

            System.out.println("\*\*\* Movement finished \*\*\*");

            // For stats

            nAgents = crawlers.nAgents();

            System.out.println("\*\*\* Remain Agents = " + nAgents + " \*\*\*\*\*\*\*\*\*\*\*\*");

            // Time measurement ends

            long elapsedTime = System.currentTimeMillis() - startTime;

            System.out.println("Elapsed time = " + elapsedTime);

            // Checking new status after one migration

            status = (Object[]) crawlers.callAll(MoveAgent.show\_location\_, (Object[]) dummyArgs);

            for (int i = 0; i < status.length; i++)

                System.out.println((String) status[i]);

            network.callAll(NodeGraphMASS.display\_);

        } catch (Exception e) {

            System.err.println("Error in execution: " + e.getMessage());

            e.printStackTrace(System.err);

        }

    }

}

MoveAgent.java

package edu.uw.bothell.css.dsl.appl.graphs.OneStepMove;

import edu.uw.bothell.css.dsl.MASS.GraphAgent;

import edu.uw.bothell.css.dsl.MASS.MASS;

public class MoveAgent extends GraphAgent{

    // function identifiers

    public static final int init\_ = 0;

    public static final int one\_step\_move\_ = 1;

    public static final int show\_location\_ = 2;

    public Object callMethod(int functionId, Object argument) {

        switch (functionId) {

            case init\_:

                return init(argument);

            case one\_step\_move\_:

                return migrateRandom(argument);

            case show\_location\_:

                return showLocation(argument);

        }

        return null;

    }

    /\*\*

     \* Is the default constructor.

     \*/

    public MoveAgent() {

        super();

    }

    public MoveAgent(Object arg) {

        super(arg);

        MASS.getLogger().debug("agent(" + getAgentId() + ") was born.");

    }

    public Object init(Object arg) {

        return null;

    }

    public Object showLocation(Object arg) {

        return "Now agent(" + getAgentId() + ") is @: place index =" + getPlace().getIndex()[0];

    }

}

NodeGraphMass.java

package edu.uw.bothell.css.dsl.appl.graphs.OneStepMove;

import edu.uw.bothell.css.dsl.MASS.MASSBase;

import edu.uw.bothell.css.dsl.MASS.VertexPlace;

import java.util.Arrays;

import java.util.stream.Collectors;

public class NodeGraphMASS extends VertexPlace {

    // function identifiers

    public static final int init\_ = 0;

    public static final int display\_ = 1;

    public Object callMethod(int functionId, Object argument) {

        switch (functionId) {

            case init\_:

                return init(argument);

            case display\_:

                return display();

        }

        return null;

    }

    private Object display() {

        try {

            String neighborString = Arrays.stream(getNeighbors())

                    .map(String::valueOf)

                    .collect(Collectors.joining(", "));

            MASSBase.getLogger().debug(getIndex().toString() + ": " + neighborString);

        } catch (Exception e) {

            MASSBase.getLogger().error("Exception displaying vertex", e);

        }

        return null;

    }

    /\*\*

     \* Is the default constructor.

     \*/

    public NodeGraphMASS() {

        super();

    }

    public NodeGraphMASS(Object arg) {

        super(arg);

    }

    public Object init(Object arg) {

        return null;

    }

}

# 3.Execution Result

## 3.1 Running any MASS program result (Triangle Counting)

Example Triangle Counting execution result is shown in fig2.

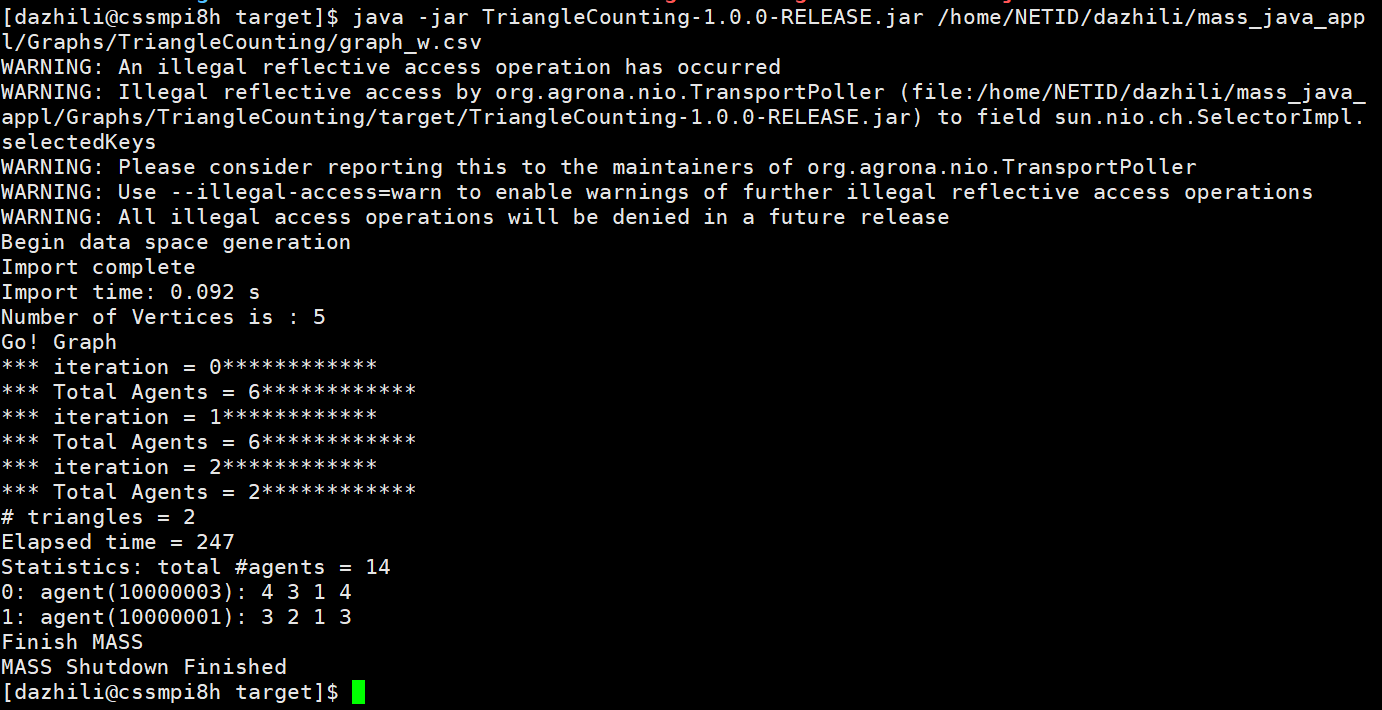


Fig2. Triangle Counting Result

## 3.2 LAB5 OneStepMove program result

The following fig3. shows the running result of my OneStepMove program. As every agent is spawn on one vertex and move forward to the next vertex.

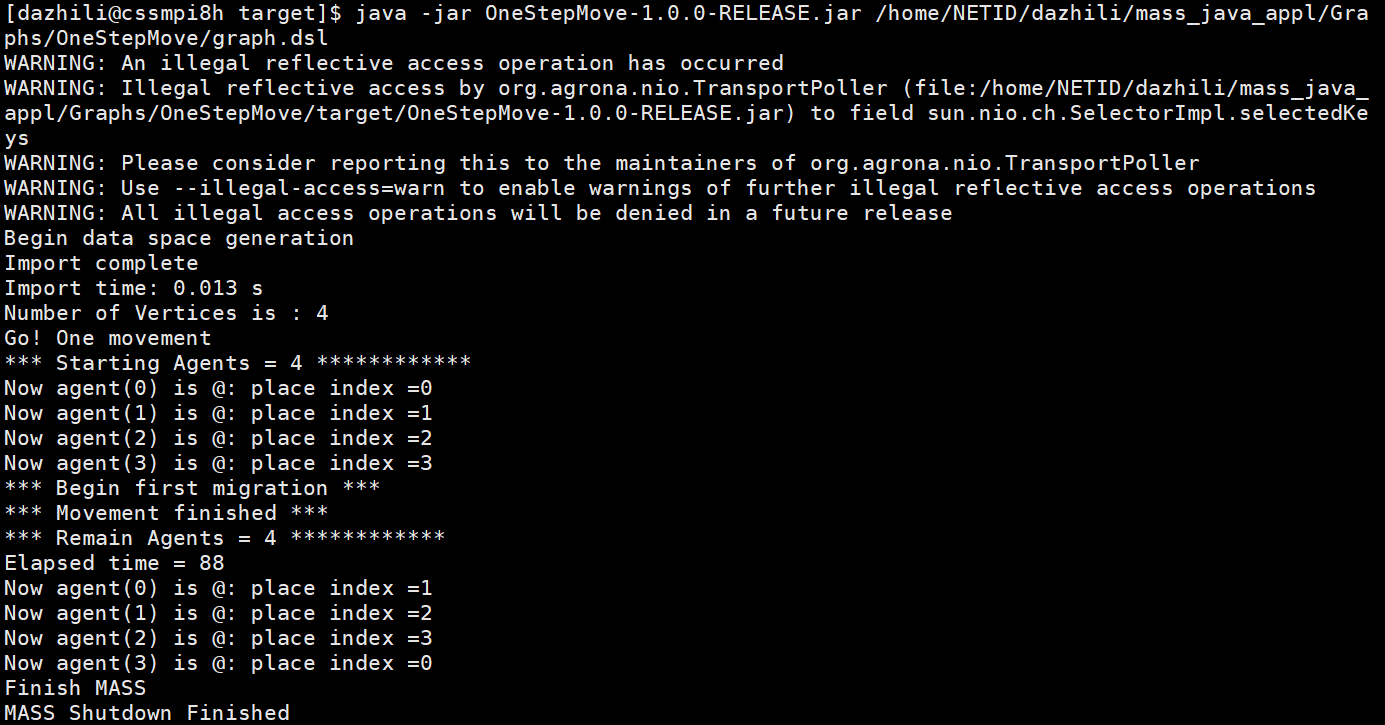


Fig3. One Step Move Result

# 4.Discussions

For Lab5, MASS application shows an intuitive way to do computation on a graph. Especially triangle counting is pretty impressive for me to see how easy it is to accomplish this job.

# 5.Final Project Peer Evaluation (Group 6)

Fred Xu (10/10): Fred Xu is a very good teammate in our final project. He came up with a schedule of our project. And he is very likely to learn new things during our discovery.

Jeremy Gao (10/10): Jeremy Gao did a good job in our project, handling the hardest part of MapReduce. A teammate who undertakes risks of difficult to parallel our topic.

Letitia Su (9/10): Letitia Su is a good teammate but kind of selfish on dataset selecting and considering too much on her own benefit. However, she still contributes efforts in our project. The deducted point is for absence in our online meeting which delays a bit of our progress.