Principles of Computer Networks
COMP 3203 — Fall 2016
December 13, 2016

Coverage of a Unit Interval with Sensors

Authors:
Christopher McMorran (100968013)
James Mulvenna (100965629)
Jenish Zalavadiya (100910343)

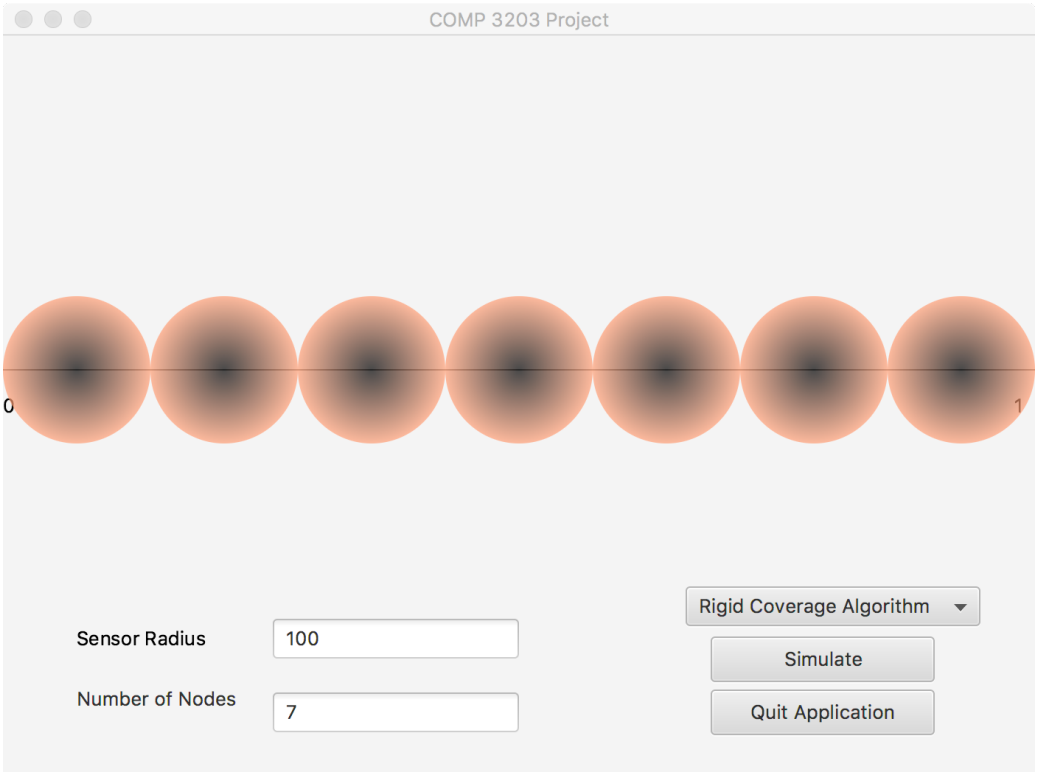


Table of Contents

Program Setup Instructions

The following instructions can be followed to execute the program in a Windows environment. Instructions are provided for running the program without compiling into an executable JAR file, followed by instructions for compiling the source code into an executable JAR file from a standard Carleton University — School of Computer Science, laboratory hosted computer.

Instructions: Without Compiling

1) Place the COMP3203.jar file that was provided alongside this documentation in any directory of your choosing.
2) Open a command prompt (C:\Windows\System32\cmd.exe) and navigate to the directory in which COMP3203.jar exists (cd \path\to\directory).
3) Ensure that you're running a version of Java greater than 1.8.0. This can be done through the following command:

   java -version
4) Execute the following command to run the executable:

   java –jar COMP3203.jar
5) The program should now be displayed. The screen should appear similar to what is shown in **Figure 1.**

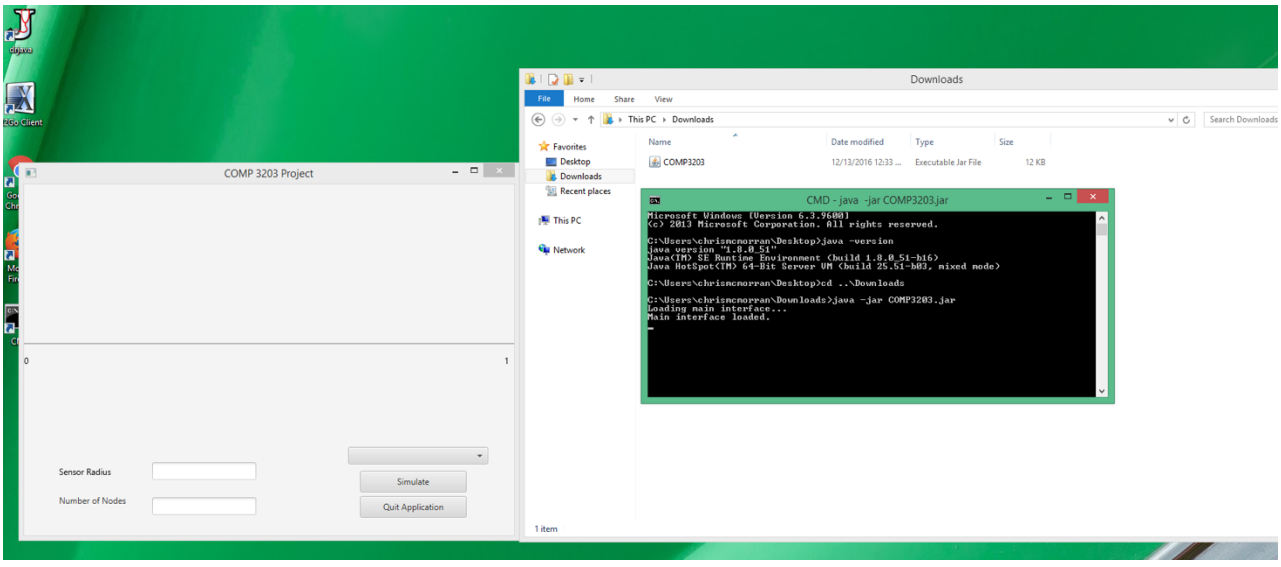


Figure 1

Instructions: With Compiling

1) Open Eclipse, which can be located at *C:\Software\Languages\eclipse-Mars\eclipse.exe*.
2) Navigate to the source code and open it in Eclipse by navigating through *File* → *Import…* → *General* → *Existing Projects into Workspace* → *Browse…* and select the project source code from the directory browsing window. Clicking *Finish* will load the project into Eclipse.
3) Expand the tree alongside Eclipse's *Package Explorer*. Expand the project to select the Main Class. *COMP3203* → *src* → *core* → *Main.java*. Right-click the file *Main.java* and select *Run As* → *1 Java Application*. **Figure 2** displays how this can be achieved.
4) The application should now appear, as shown in **Figure 3**.

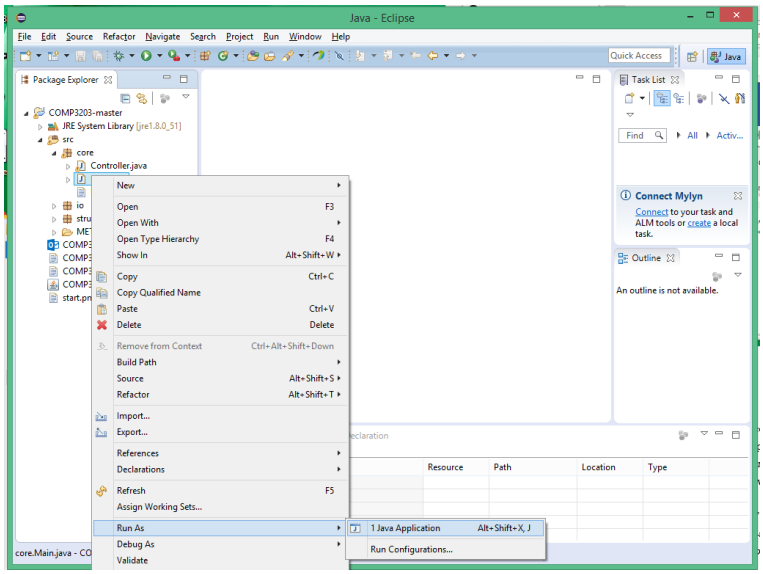


Figure 2

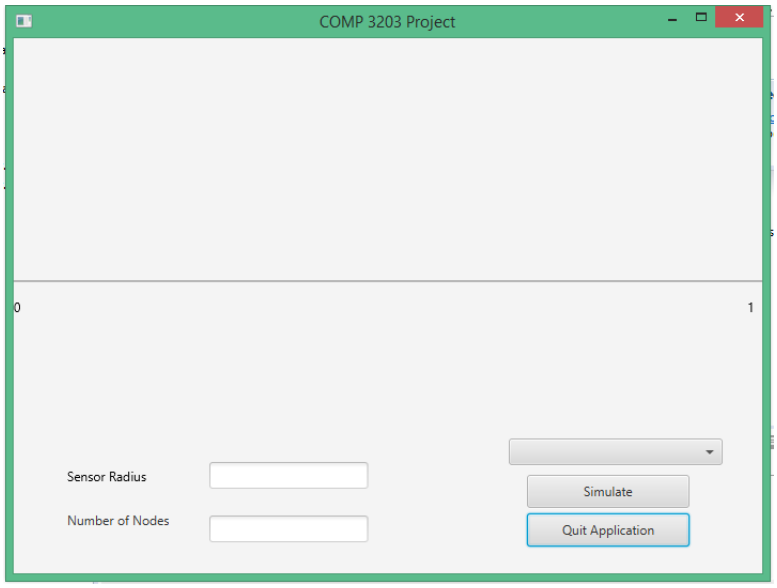


Figure 3

Implementation Details:

Rigid Coverage Algorithm

No overlaps should occur with this algorithm. Sensors are initialized with random X coordinate while the Y coordinate is kept constant which is the same as the unit interval line. The sensor with the smallest random X coordinate takes the left most possible position, the next sensor with smallest random X coordinate takes the second left most position and so on. When the unit interval has been covered with sensors, the remaining sensors leave the unit interval by moving to the right, this will insure no overlaps within the unit interval.

Sensors placed at random initial coordinates:

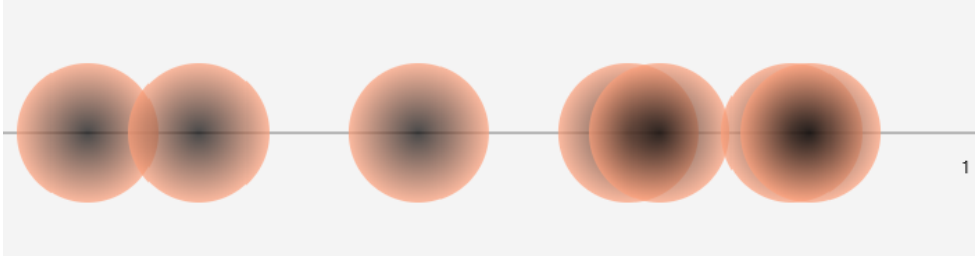


Figure 4

Sensors after displacement:

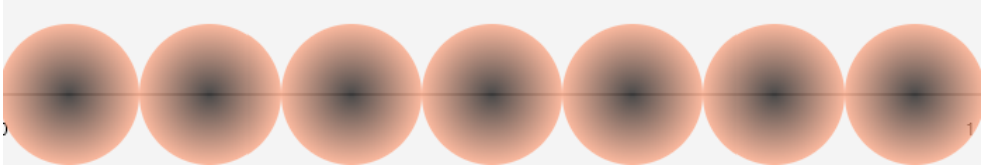


Figure 5

Simple Coverage Algorithm

Overlaps are ignored with this algorithm. Like the rigid coverage algorithm, all sensors are initialized with random X coordinates while the Y is kept constant. Positioning of sensors with this algorithm is also the same as rigid where the sensor with the smallest random X coordinate takes the leftmost position on the unit interval. When the unit interval has been covered, the remaining sensors stay at the positions they were initialized with. This will not remove any overlaps.

Sensors placed at random initial coordinates:

Figure 6

Sensors after displacement:

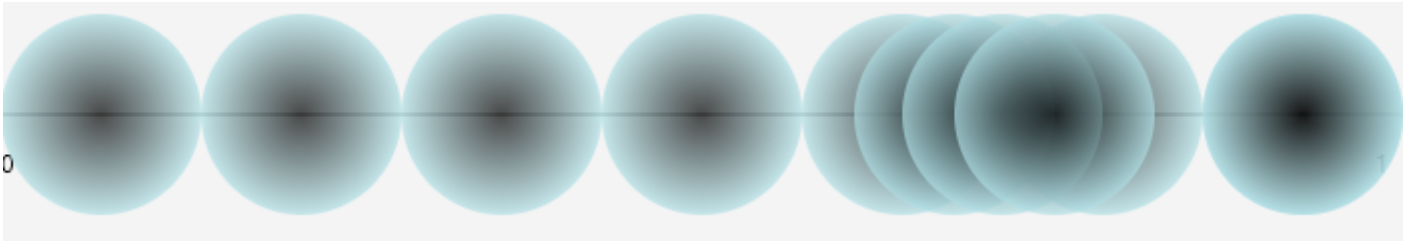


Figure 7

Coverage from Min-Max Y-Coordinate Algorithm

Sensors are initialized with random a random Y coordinate either above or below the unit interval and each respective X coordinate is chosen to fit the unit interval with optimum coverage ranges.

Sensors placed at random initial coordinates:

Figure 8

Sensors after displacement:

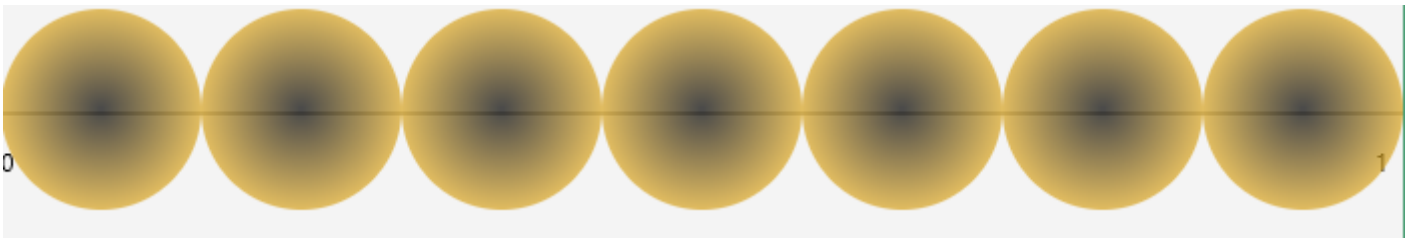
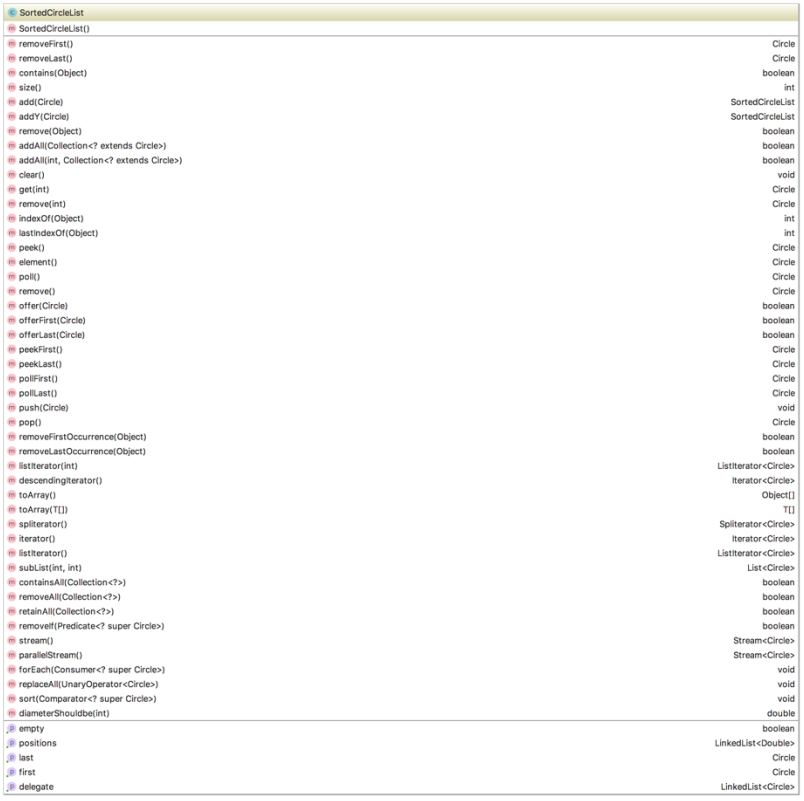
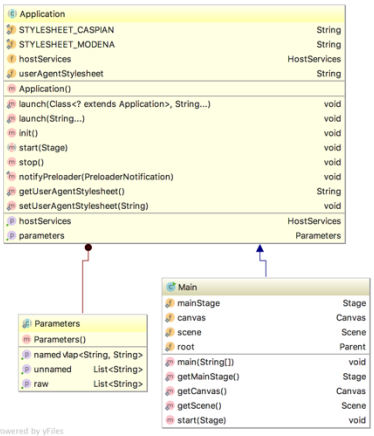


Figure 9

UML Class Diagram





Test Cases

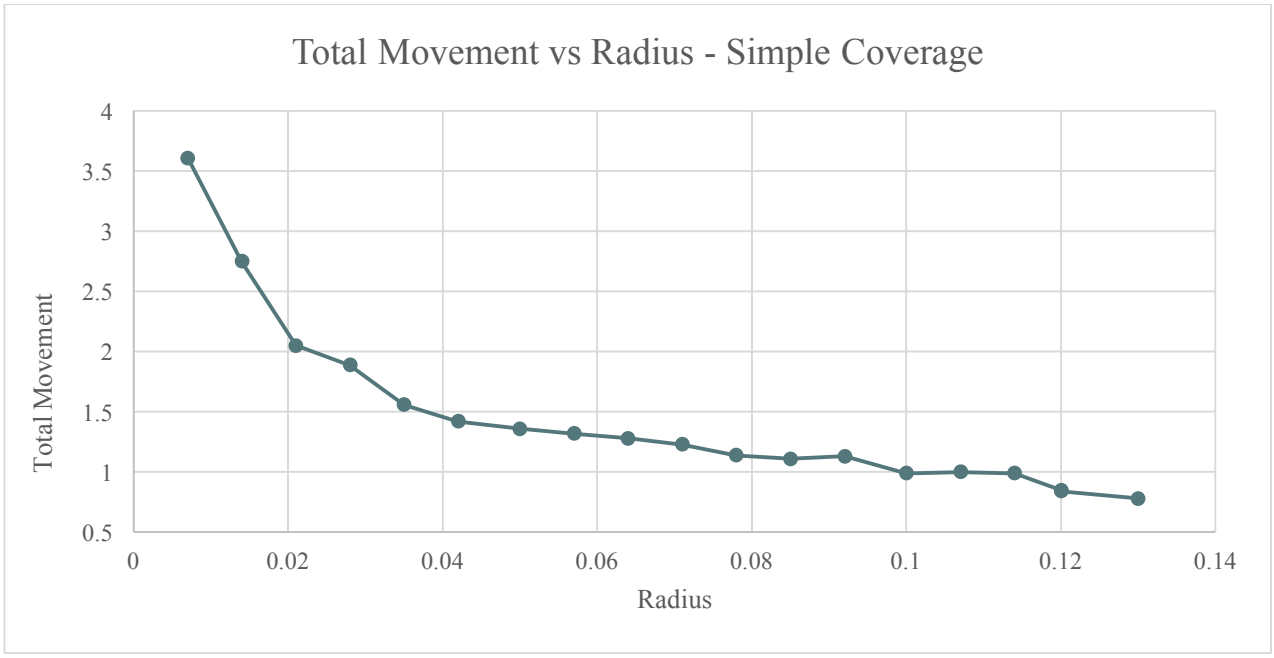
Rigid Coverage Algorithm

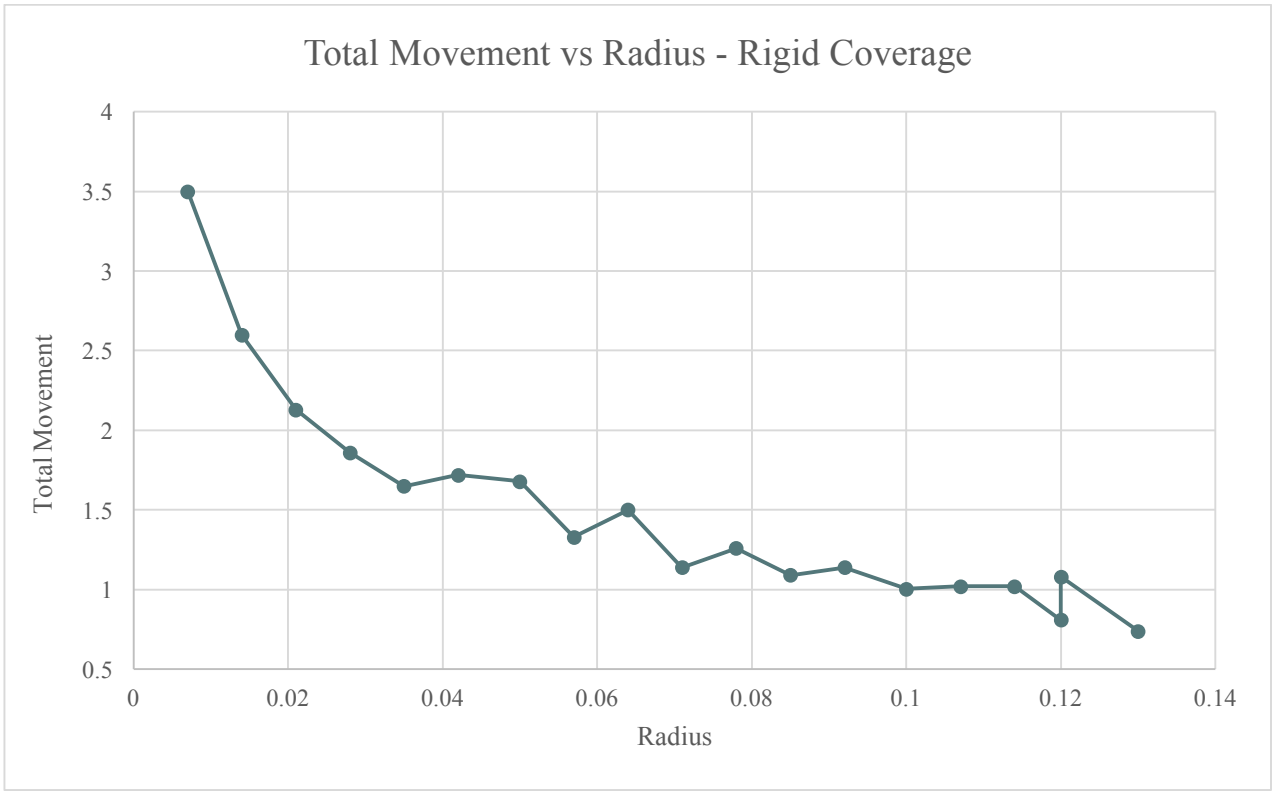
Number of sensors = 8
Radius = 100 (0.14 in relation to unit interval [0,1])
Sensors are added at random positions on the unit interval.
Rigid coverage algorithm is applied to these sensors to optimize the coverage. Rest of the sensors have been moved to the right outside the unit interval, this is done to remove overlaps. Total sum of movements including the sensor that was moved out of unit interval line is equal to 789. In relation to unit interval [0,1], this movement is equal to 1.12.

Simple Coverage Algorithm

Number of sensors = 8
Radius = 100 (0.14 in relation to unit interval [0,1])
Sensors are added at random positions on the unit interval.
Simple coverage algorithm is applied to these sensors to optimize the coverage. Sensors that cause overlaps are ignored. Total sum of movements is equal to 500. In relation to unit interval [0,1], this movement is equal to 0.71.

Graphs


Total Movement vs Radius - Simple Coverage
Total Movement
Radius
4
3.5
3
2.5
2
1.5
1
0.5
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14


Total Movement vs Radius - Rigid Coverage
Total Movement
Radius
4
3.5
3
2.5
2
1.5
1
0.5
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14