COMP 3203

Assignment #2

Fraser MacQuarrie

100624202

Overview

The included software finds min-sum and min-max solutions to four barrier coverage problems identified in the assignment description, namely:

(1) 1 Dimensional case where the final positions of the sensors are necessarily evenly spaced along the line.
(2) 1 Dimensional case where the final positions of the sensors are not necessarily evenly spaced along the line.
(3) 2 Dimensional case where the area to be protected is a circle and the final position of the sensors is necessarily evenly spaced around the perimeter.
(4) 2 Dimensional case where the area to be protected is a simple polygon and the final position of the sensors is necessarily evenly spaced around the perimeter.

Notes about the source code

The program was written in Java using NetBeans 6.1. The entire project folder has been submitted so that the source code can be compiled. There is also an executable .jar version of the program in the \dist\ folder.

How to use the program

When the program first loads, there is a control area at top and a blank canvas below it. In the control area, enter the desired parameters in the text boxes and click on one of the four simulation buttons to begin a simulation. Once a simulation is started, click on one of the optimization buttons to run the appropriate optimization scenario. Once the scenario is run, the min-sum and min-max information is calculated and displayed as well as whether or not it is feasible to secure the area required with the sensors provided. See figure 1 for a screenshot of the user interface.

Files included in submission

Source Files
The source files are located in \src\

Executable File
An executable .jar file is located in \dist\

Javadoc
The javadoc for this project is located at \dist\javadoc\index.html

Log File
The log file, log.txt, is in the main project folder

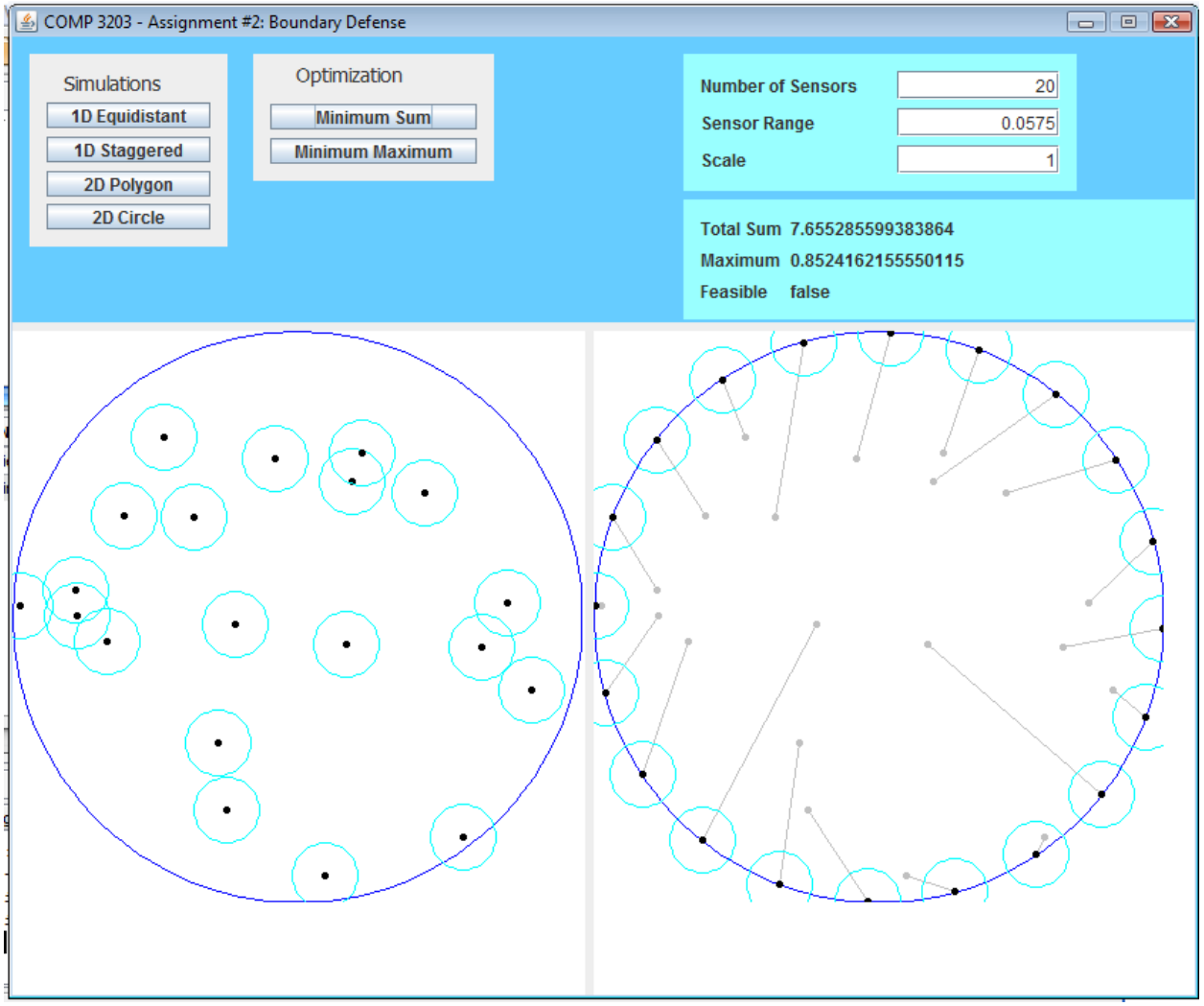

Figure 1: GUI of Project

Algorithms

Algorithm for case (1) to find min-sum

(i) Calculate the positions on the barrier where the sensors should be
(ii) Move the sensors so that there is one sensor at each position
(iii) Iterate pairwise through the sensors, swapping them if doing so leads to further optimization.
(iv) Repeat iterating until one complete pass occurs where no switches take place.

Algorithm for case (1) to find min-max

(i) Calculate the positions on the barrier where the sensors should be
(ii) Move the sensors so that there is one sensor at each position

(iii) Iterate pairwise through the sensors, swapping them if doing so leads to further optimization.
(iv) Repeat iterating until one complete pass occurs where no switches take place.

Algorithm for case (2) to find min-sum and min-max

(i) Calculate the positions on the barrier where the sensors should be
(ii) Move the sensors so that there is one sensor at each position
(iii) Iterate pairwise through the sensors, swapping them if doing so leads to further optimization.
(iv) Repeat iterating until one complete pass occurs where no switches take place.

Algorithm for case (3) and (4) to find min-sum

(i) For each of the sensors do steps (ii) – (vi):
(ii) Calculate the shortest path from the sensor to the barrier and move the sensor along this path until it is at the barrier.
(iii) Use this point to calculate the positions on the barrier where the sensors should be
(iv) Move the sensors so that there is one sensor at each position
(v) Iterate pairwise through the sensors, swapping them if doing so leads to further optimization.
(vi) Repeat iterating until one complete pass occurs where no switches take place.
(vii) Compare the results from each sensor and choose the most efficient solution.

Algorithm for cases (3) and (4) to find min-max

(i) For each of the sensors do steps (ii) – (vi):
(ii) Calculate the shortest path from the sensor to the barrier and move the sensor along this path until it is at the barrier.
(iii) Use this point to calculate the positions on the barrier where the sensors should be
(iv) Move the sensors so that there is one sensor at each position
(v) Iterate pairwise through the sensors, swapping them if doing so leads to further optimization.
(vi) Repeat iterating until one complete pass occurs where no switches take place.
(vii) Compare the results from each sensor and choose the most efficient solution.Class Diagram

Testing

Simulation: Equidistant line

# Sensors	Range	Scale	Feasible	Time
1000	-	-	-	5s
10000	-	-	-	24s
20000	-	-	-	40s
10	0.0575	1	true	-
9	0.0575	1	true	-
8	0.0575	1	false	-

Optimization	# Sensors	Range	Scale	Distance	Maximum
Min-Max	10	0.0575	1	1.21	0.26
Min-Max	20	0.0575	1	1.42	0.20
Min-Max	40	0.0575	1	1.16	0.09
Min-Sum	10	0.0575	1	1.21	0.26
Min-Sum	20	0.0575	1	1.42	0.22
Min-Sum	40	0.0575	1	1.16	0.13

Simulation: Circle

# Sensors	Range	Scale	Feasible	Time
50	-	-	-	6s
75	-	-	-	15s
100	-	-	-	31s
25	0.0575	1	false	-
27	0.0575	1	false	-
28	0.0575	1	true	-

Optimization	# Sensors	Range	Scale	Distance	Maximum
Min-Max	10	0.0575	1	4.44	0.59
Min-Max	20	0.0575	1	10.36	0.88
Min-Max	40	0.0575	1	17.98	0.81
Min-Sum	10	0.0575	1	4.00	0.80
Min-Sum	20	0.0575	1	7.87	1.10
Min-Sum	40	0.0575	1	14.52	1.08

Test Cases

The program has been tested repeatedly to ensure consistency. Through these efforts a rare error was encountered which was traced to a precision error when calculating angles. A fix for this problem has been introduced, and no other errors have been encountered.

Class Diagram