Michael Doescher

Computational Geometry

CS/ME 558

Friday December 20, 2013

Final Project

**Strategies for Awakening a Swarm of Robots**

**Abstract**

This problem investigates a collection of robots located at distinct points in a plane. All but one of the robots are initially asleep. The awake robot attempts to wake up all of the robots by moving to their locations and activating their power switch. Once a robot has been awakened it can help wake up other robots. The goal is to find a schedule that permits the robots to be awakened in the minimum amount of time.

**Background**

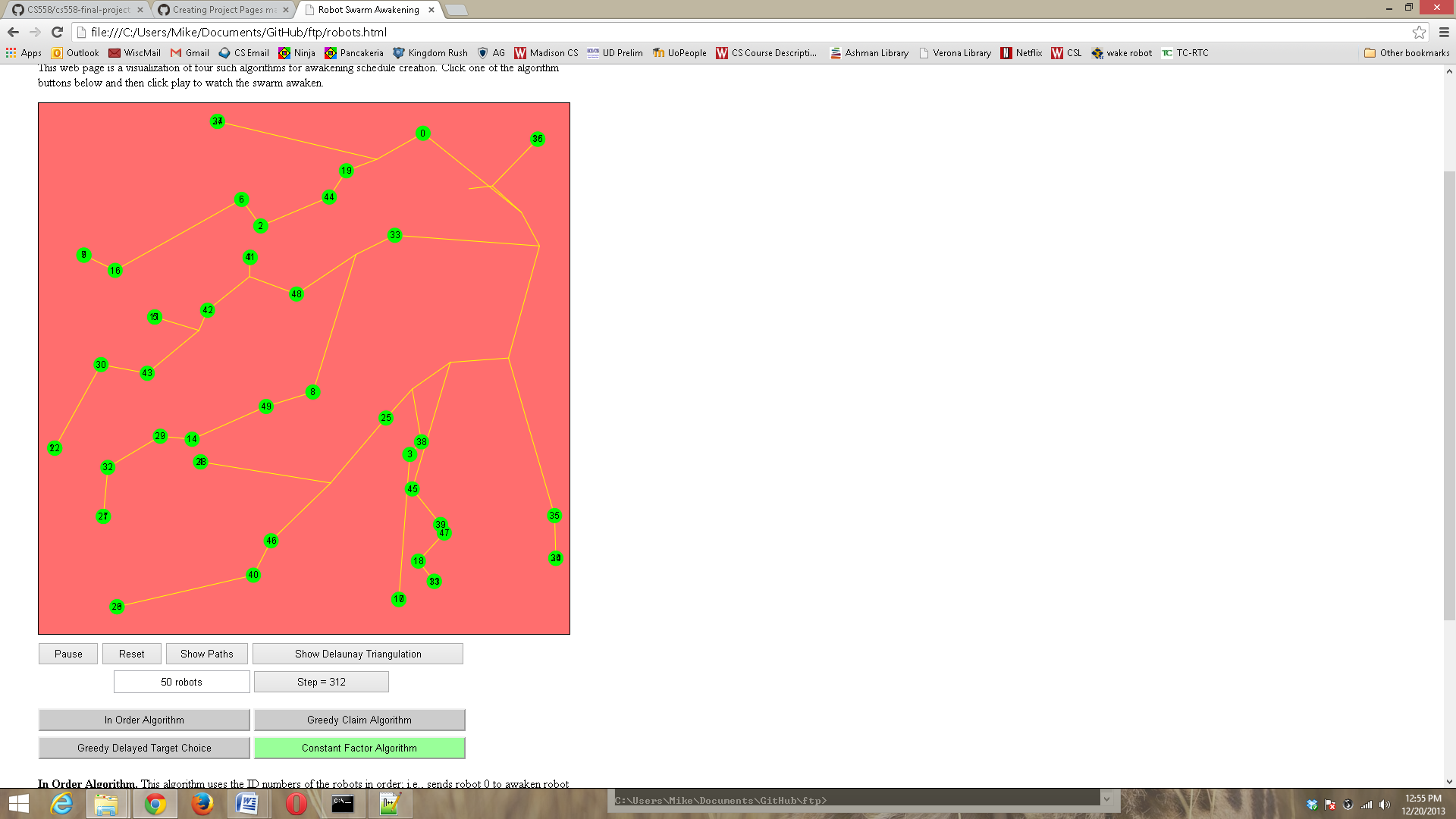
The robot swarm awakening problem, also known as the Freeze Tag Problem, is one of seventy-seven open problems in computational geometry listed at the Open Problems Project by Demain, Mitchell, and O'Rourke.1 The Freeze Tag moniker originates from the children's game freeze-tag: one child chases the other children and tags them to freeze them, then once all children are frozen the game reverses, and the last child runs to each frozen child and unfreezes them; these unfrozen children are allowed to help unfreeze the rest of the children.

The problem has a number of practical applications involving network design and routing; the optimal awakening schedule forms a minimal depth spanning tree with a constraint on the maximum degree of each node.2 Another application involves the dissemination of high risk information: covert agents wait at rendezvous points to meet with another agents and then continue propagating the information to all other agents in the network.2

The problem was first proposed by Arkin et al. in 2002, and they showed the problem to be NP-complete for star graphs, general graphs, and geometric spaces.2 Researchers speculate that this problem is NP-hard in Euclidean space and this problem has in fact been proven to be NP-hard for robots located at the nodes of star graphs in which all nodes are connected only to a central hub.2 In addition to the theoretical work reducing the freeze tag problem from known NP-complete problems, researchers have also investigated several approximation algorithms and explored the complexity of those algorithms including an examination of lower and upper bounds for these algorithms.3,4

**Project**

I developed a simulation to visualize several algorithms for approximating the robot swarm awakening problem for robots in 2D geometric space which demonstrates the relative efficiency of these algorithms by counting how many steps are required to awaken all of the robots. I implemented the In Order, Greedy-Claim, Greedy-Delayed Target Choice, and Arkin's Constant Factor Algorithm.



The visualization shows the positions of all the robots as circles. Initially robots that are sleeping are blue and awake robots are green. The first row of buttons control options for using the simulation. Play/Pause causes the simulation to proceed forward or pause. Reset returns all robots to their initial position and state. Show Paths lets the robot spray the ground with paint as they walk to leave their trail behind. Show Delaunay Triangulation displays the Delaunay Triangulation of the initial positions of the robots. Below that is options for changing the number of robots and the step counter which increments as the simulation runs. Finally buttons for each of the four algorithms appear at the bottom of the screen. Clicking on one of these buttons resets the robots to their initial positions, resets the step counter, and recalculates the schedule.

**Algorithms for the Freeze-Tag Problem**

Algorithms from the literature fall into two categories: heuristic and nondeterministic. Heuristic algorithms include the brute force, naive, greedy3, bang-for-buck3, opposing cone3, and Arkin's constant-factor approximation algorithm2. The nondeterministic algorithms include genetic2, pheromone2, neighborhood hill-climbing4, and random sectors3. Of these, only a few of these algorithms take advantage of the geometry to develop an algorithmic solution to the freeze tag problem.

Brute Force: This algorithm considers all possible awakening schedules to locate the schedule with the minimum wake up time (t\*) for all of the robots. Whenever a robot needs a new target it must consult the schedule which will either return a new target or will instruct the robot to stop. This algorithm must consider all permutations of robots and halting instructions as schedules and so runs in O((n+1)!) time, but does compute the optimal t\*. This algorithm doesn't take advantage of the geometry and is only computationally feasible for uninterestingly small numbers of robots.

In Order: This algorithm takes the list of robots in arbitrary order and simply has the first robot awaken the next robot on the list. Then those robots awaken the next two robots on the list and so on until all robots are awakened. The algorithm runs in constant time and surprisingly the solution has a wake up schedule that bounded by O(log n)t\*.3 To observe this algorithm click the In Order Algorithm button in the simulation.

Greedy-Claim: There are three versions of the greedy algorithm in which a robot awakens the nearest robot. The difference between these deal with how conflicts are resolved as they arise. In the first, when a robot is awakened it claims the nearest unclaimed sleeping robot and then goes to awaken that robot (even if a second robot is later awakened that could reach the claimed robot first (see Greedy-Refresh, and Greedy-Delayed Target Choice below). Greedy algorithms generate wake up schedules that are bounded by O()t\*. A Delaunay triangulation of the robots can be constructed in O(n log n) time and the nearest point must be connected to the current robot by a Delaunay edge. This data structure will permit locating the nearest neighbor in time proportional to the degree of the triangulation O(k) and as robots are awakened they may be deleted from the triangulation in O(k log k).5 The visualization does permit viewing the Delaunay triangulation by clicking on the appropriate button and indeed the nearest robot is along a Delaunay edge and one of the robots will travel along that edge. However, the algorithm in the visualization uses the naive approach to locating the two nearest neighbors.

Greedy-Refresh and Greedy-Delayed Target Choice deal with conflicts that arise after a robot has claimed its nearest neighbor. The refresh method recalculates a matching between all asleep robots and all awake robots at every event when a robot is awakened. Thus, if a robot starts moving toward its nearest neighbor and a robot that is closer is awakened that robot will claim the nearest neighbor and the first robot will chose a new target. The Delayed Target Choice algorithm computes a wake-up schedule by letting the awake robots accumulate movement points without actually moving until an asleep robot is within the radius of their movement range. Then they add that robot to the schedule and teleport to its location. Both of these algorithms belong to the same complexity class as the greedy-claim algorithm. The Greedy-Delayed Target Choice algorithm has been implemented in the visualization.

Bang-for-the-Buck. This algorithm subdivides the space into pie-shaped sectors for each individual robot and the robots calculate the value of a sector as the number of robots within the sector divided by the distance to the closest robot in that sector. This algorithm allows a robot to travel to a more distant robot for a better payoff because more robots are then available to help wake up the others. Robots travel to the sector with the best value and awaken one robot and then travel to the second highest valued sector based on their original starting position.

Opposing-Cones performs well in the special cases where the greedy and bang for the buck algorithms are far from optimal (large groups of robots that are slightly further away or in the opposite direction from other groups of robots). When a robot awakens another there are two robots at the same point in the graph. One robot chooses the nearest neighbor to awaken and the other robot chooses a robot such that the direction it will travel makes an angle greater than 90° to the direction of the first robot.

Arkin's constant-factor approximation algorithm is very similar to the bang-for-the-buck algorithm in that it divides the space into sectors based on angle from the each robot's starting position but does not consider the number of robots in each sector only the distance to the closest robot. This algorithm boasts a constant factor worse than optimal awakening time, O(1)t\*. This algorithm can be implemented with a running time O(K n log n) where K is the number of sectors using a Conical-Voronoi diagram, to locate the nearest point within each sector.2 I wanted to include one of the algorithms that divided the space into pie-shaped sectors and I selected this one as it is slightly easier than Bang-for-the Buck and has a greater foundation in theory. I used eight sectors and calculated the all closest robots in each sector using the naive O(n2) algorithm.

Genetic, pheromone, neighborhood hill-climbing, and random sector algorithms are nondeterministic and do not consider the geometry of the distribution of robots in the plane and I do not plan to consider them further here.

**Final Note**. The visualization is crafted in two parts. A canvas element along with animation subroutines take a schedule and guide the robots through the performance of the schedule. The scheduling algorithm prepares the desired schedule based on which algorithm is selected and returns it as a list to the animation. Numerical differences occasionally result in errors between these two systems and occasionally (particularly for large >1000 robots ) one robot will arrive at its destination out of order which results in every robot from that point on receiving the wrong robot when it queries the schedule.

**References**

1. Demaine, E.D., Mitchell, J.S.B., O'Rourke, J. (**2012**). Open Problems Project. Retrieved from <http://cs.smith.edu/~orourke/TOPP/>.

2. Arkin, E.M., Bender, M.A., Fekete, S.P., Mitchell J.S.B., Skutella, M. (**2002**) The freeze-tag problem: How to wake up a swarm of robots. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms.* ACM Press. Retrieved from <http://www.ams.sunysb.edu/~jsbm/papers/freezetag-soda02.pdf>.

3. Sztainberg, M.O. (**2003**) *Algorithms for Swarm Robotics*. Retrieved from <http://202.28.199.34/multim/3107115.pdf>.

4. Bucantanschi, D., Hoffmann, B., Hutson, K.R., Kretchmar, R.M. (**2006**). A Neighborhood Search Technique for the Freeze Tag Problem. Retrieved from <http://personal.denison.edu/~kretchmar/pubs/AugPathINFORMS06.pdf>.

5. Devillers O. (1998). On Deletion in Delaunay Triangulation. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.646&rep=rep1&type=pdf>.