Benjamin Brown

Project Report

4/24/2017

1. (10 points) What is your application problem?
   Antibody moves around the grid using the Simulated Annealing and Genetic algorithm. With these, it will inspect the neighboring grid space for viruses and eliminate them if they exist.

   Initial State:
   Randomly generated location of x and y for the antibody, 4 cells, and a virus within a grid.
   Initial:
   Actions:
   Move(x, y) – Antibody moves to a position on a grid at x, y.
   Inspect(x, y) – Antibody inspects a block on the grid at x, y.
   Kill(x, y) – Antibody kills a virus on the grid at x, y.
   Die() – Virus is killed by the antibody.

   Transition Model:

   If the Antibody is a neighbor of a cell it will inspect it, otherwise it will continue moving towards one.

   Transition: Result(Position(CurrentLocation), Move(NeighboringLocation)) = Position(NeighboringLocation)

   Goal Test:

   The Goal test is determined by if all the cells, including the virus, have been inspected and the virus has been eliminated. When true, the algorithm will end.

   Goal: Empty(CellStack) && Dead(Virus)

   Path Cost:

   The path cost for this implementation is based on the Euclidean distance between the two points on a grid of the initial and goal state.

   Cost = Distance(Location initial, Location final) or

   Cost = Distance(Location initial, Location final) + margin * Distance(Location initial, Location final)

   Where margin is the extra allowed distance to account for obstacles.

2. (15+15 points) Describe:
   i. the representation of your problem that is used to solve the problem using GA.

      The Genetic Algorithm is used to move my Antibody object towards the current cell on a stack. The fitness function is used to calculate an appropriate distance in which the chromosomes can travel with no obstacles using a slightly modified Euclidean distance function. My selection method randomly picks a chromosome out of the population and using it as a parent. While not the most effective, it does work for this problem. Crossover and Mutation are performed randomly on the selected chromosome(s).

Heuristic: -1 * Distance(ChromosomePath, Goal) –
(MaximumDistance(AntibodyLocation, Goal) / 100.0f)

ii. the representation of your problem that is used to solve the problem using SA.

For Simulated Annealing, it does very much the same functionality as the Genetic Algorithm, moving my antibody to the current cell on the stack, but it uses a pure Euclidean distance function to get the best results for the path. My temperature function uses K and K_Max, where K is the iteration count and K_Max is 127 (Byte type's maximum value.) The functions return value will continue to change and when the algorithm is complete (The cells stack is empty), it will return 0.

Heuristic: Distance(initial, goal)

3. (15+15 points) Mathematically define the:
   i. fitness function for your problem for GA.

For Genetic Algorithm my fitness function uses the following algorithm:
Fitness:
-1 * Cost(ChromosomePath, Goal) – MaximumCost(AntibodyLocation, Goal) / 100.0f
Cost:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Maximum Cost:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + 0.35 * \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

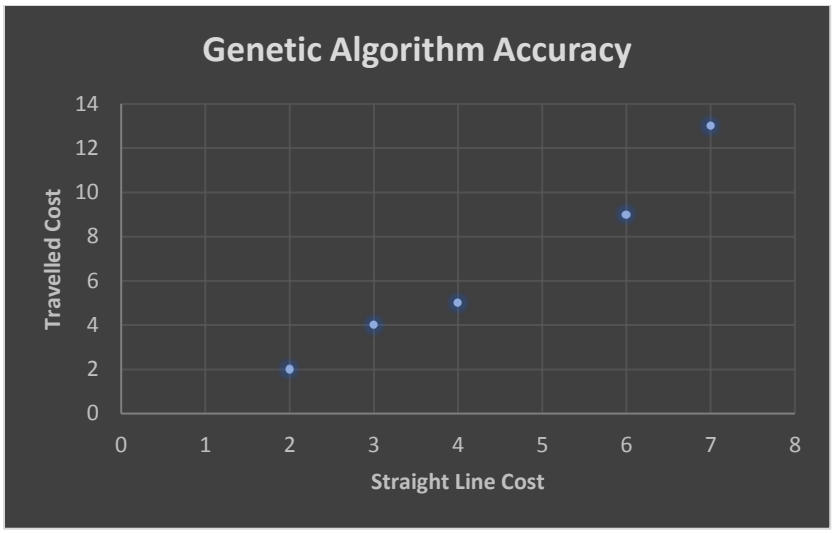
This is because with the genetic algorithm we need to ensure our path accounts for additional barriers that prevent the initial state from taking a direct path to the goal.

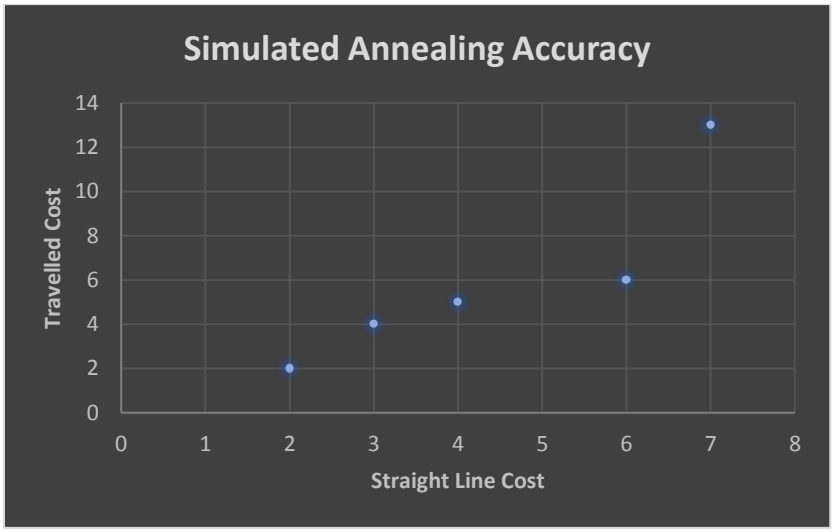
   ii. objective function for your problem for SA.

For Simulated Annealing, it also uses the Euclidean distance formula, but without any extended costs as we are moving on a per cell basis rather than generating an entire path allowing us to account for obstacles during the algorithm runtime.
Cost:

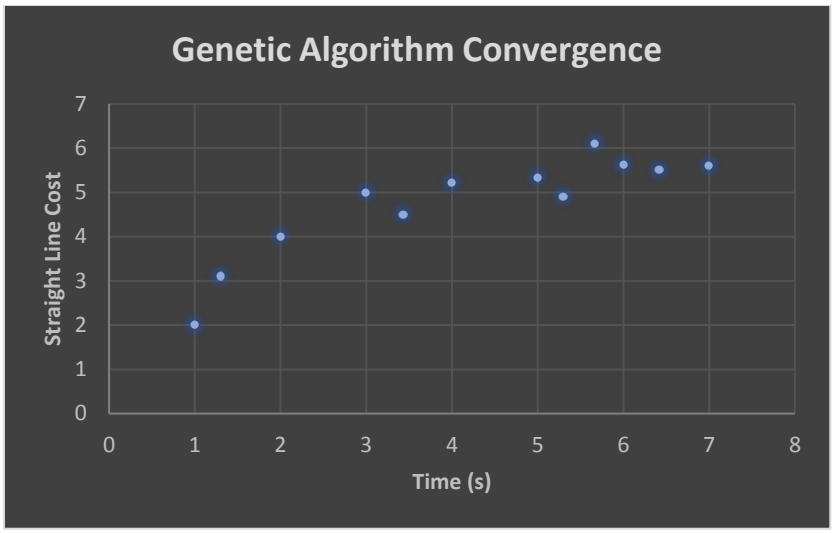
$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

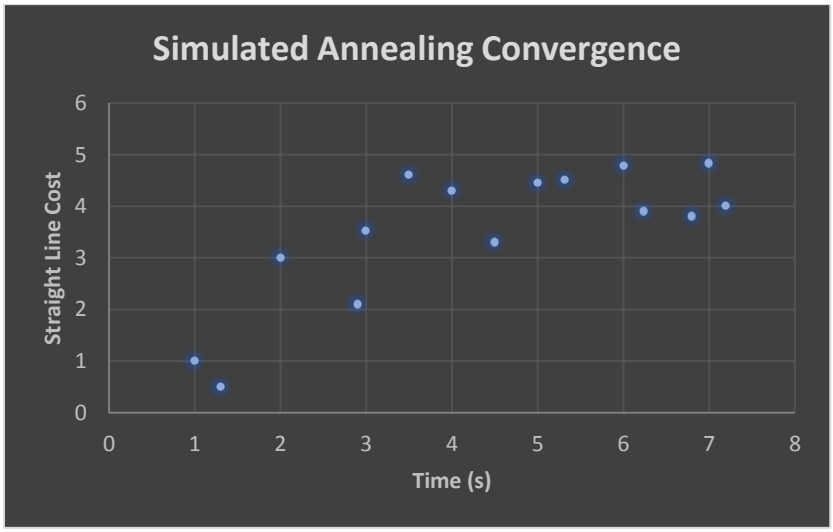
4. (15+15 points) Compare the two algorithms using:
   i. the accuracy of your solution.





ii. the time (in seconds) required to reach convergence. Show convergence plot for each algorithm.


Genetic Algorithm Convergence
Straight Line Cost
Time (s)
0
1
2
3
4
5
6
7
8


Simulated Annealing Convergence
Straight Line Cost
Time (s)
0
1
2
3
4
5
6
7
8