Benjamin Brown

COMP720 Project Report

4/26/2017

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.

   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 rectilinear distance between the two points on a grid of the initial and goal state.

   <u>P.E.A.S. Evaluation</u>

   Performance: Rectilinear Distance

   Environment: 2D Coordinate Plane

   Actuators: Killing Virus, Inspecting Cells, Moving

   Sensors: Direct Contact (Neighboring)

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

Genetic Algorithm uses the four cardinal directions stored as a bit string for encoding the path in a population with rectilinear distance as its fitness 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).

Chromosome Representation
00: Up, 01: Down, 10: Left, 11: Right
Ex: [00101101010010] = Up -> Left -> Right -> Down -> Down -> Up -> Left

Hard constraints must be placed to ensure that the chromosome repairs itself.

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

Simulated Annealing's objective function uses the rectilinear distance formula to calculate the best path. Euclidean distance is straight line, therefore for a program such as mine would not suffice as we do not move diagonally. Iterates k times to a limit of k_max where it will halt if a solution is not found.

Temperature
Returns k / k_max or 0 when the stack of cells is empty and the algorithm is complete.

Accepted Probability
Returns $e^{\Delta cost / T}$ where Δcost = current_cost – new_cost and T is the current temperature.

3. (15+15 points) Mathematically define the:
Definitions:
AntibodyLocation(x, y) is the current location of the antibody.
ChromosomePath(x, y) is the location where the chromosome ends at.
Initial(x, y) is the location we begin at.
Goal(x, y) is the ending location to in which we travel towards.

i. fitness function for your problem for GA.

For Genetic Algorithm my fitness function uses the following algorithm:
Cost: Cost(Initial(x_1, y_1), Goal(x_2, y_2) = $|x_2 - x_1| + |y_2 - y_1|$
Fitness: Cost(AntibodyLocation, Goal) - Cost(ChromosomePath, Goal)

ii. objective function for your problem for SA.

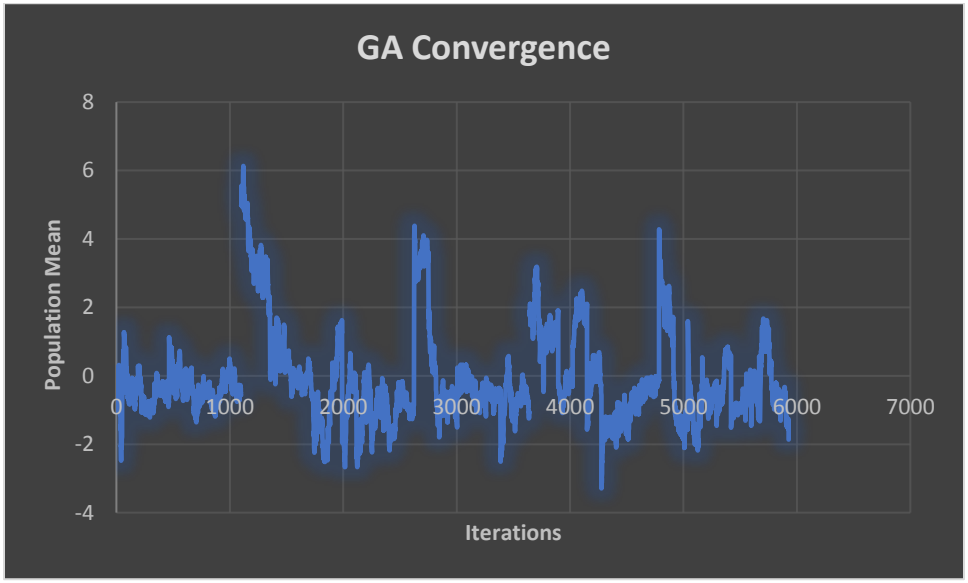
For Simulated Annealing, it also uses the rectilinear distance formula, but without any extended costs as we are moving on a per cell basis rather than generating an entire path allowing us to account for obstacles during the algorithm runtime.
Cost: Cost(Initial(x_1, y_1), Goal(x_2, y_2) = $|x_2 - x_1| + |y_2 - y_1|$

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

   Simulated annealing was about ~99% accurate. It had very few cases where it could not solve the problem. Meanwhile Genetic algorithm had ~93% as it had issues finding a solution when surrounded by 3 or more cells at any given time.

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

   Time taken to converge for GA: 0.4ms $\pm$ 0.1
   Due to random selection, my convergence is skewed. If I had performed another technique such as roulette wheel selection, perhaps it would have improved.



The negative mean is due to the distance often being negative to indicate that a smaller location is more beneficial to take due to the fitness function.

Time taken to converge for SA: 0.2ms $\pm$ 0.1