Allen Dorris

Assignment: Project Report

Due date: April 24, 2017 (Monday) by 11:30 AM

1)

The application replicates an artificial agent, referred to as the ‘character’, attempting to escape a randomly generated maze full of monsters, traps, and food. It successfully escapes the maze when it has reached the ‘goal’. The character has three ‘hearts’ of health. If it moves onto a trap or attacks a monster, it loses one heart of health, and if it eats food, it gains one heart of health. If the character reaches 0 hearts of health, it dies and is unable to search for the exit further.

Problem defined in terms of the five components:

**Initial State:**

The maze is randomly generated; as such, any state where two objects in the maze do not share a location is a valid initial state.

**Actions:**

The character can perform the following actions:

Move(x): Moves the character from its current location to adjacent location x, so long as x is valid.

Attack(x): Character attacks monster x sharing the same position.

Eat(x): Character eats food x sharing the same position.

Die(): Character dies and is unable to continue.

**Transition State:**

RESULT(In(CurrentLocation), Move(NeighboringLocation)) = In(NeighboringLocation)

**Goal Test:**

The character successfully reaches the location of the goal.

RESULT(In(CurrentLocation), Move(GoalLocation)) = In(GoalLocation)

**Path Cost:**

Every action has a path cost of 1.

2i) The Genetic Algorithm moves the character towards the goal state. The heurisitic function used is a slightly modified Euclidean distance function, allowing only 1.50 times more slack in the travel cost to account for objects possibly blocking its path. A chromosome is randomly selected out of the population for use, and crossover/mutation are randomly performed on the selected chromosome.

2ii) Simulated Annealing moves the character towards the goal state. It uses a pure Euclidean distance function to get the best results. The temperature function uses K and K\_Max, where K is the iteration count and K\_Max is 127, the Byte’s maximum value.

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 temperature’s value will continue to change and when the character reaches the goal state signaling the end of the algorithm, the temperature will be 0.

3i) Fitness Function: -1 \* Cost(ChromosomePath, Goal) – (MaximumCost(character, goal) / 100.0f)

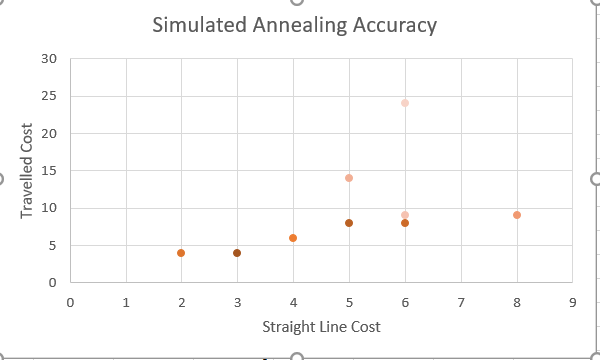
ChromosomePath is the path each chromosome has: 00 is up, 01 is down, 10 is left, 11 is right.

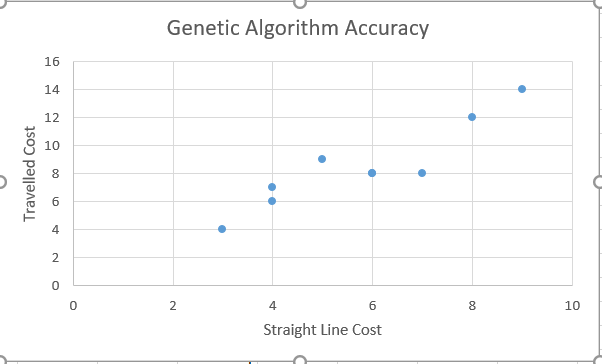
Cost: √ ((goal.row–ChromosomePath.row)2 + (goal.col – ChromosomePath.col)2)

Maximum Cost: √ ((goal.row – character.row)2 + (goal.col – character.col)2)+ 0.50 \* √ ((goal.row – character.row)2 + (goal.col – character.col)2)

3ii) Objective Function: √ ((goal.row – character.row)2 + (goal.col – character.col)2)

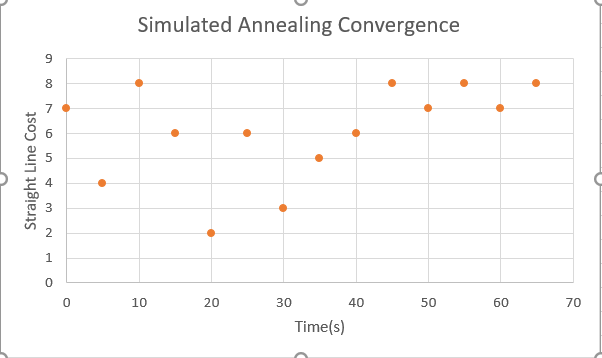
4i)

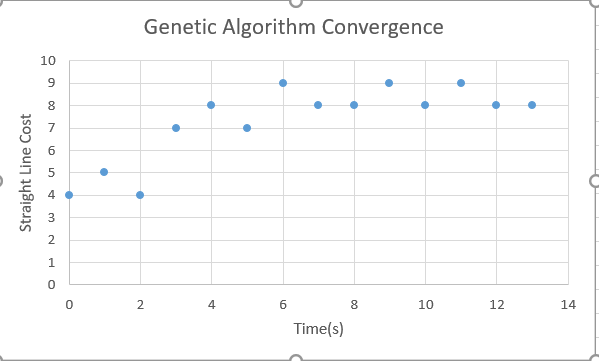




In general, it seems like the Genetic Algorithm has a better accuracy.

4ii)





The Genetic Algorithm converges far faster than Simulated Annealing for this project.