Intelligent Systems

1st Assignment 2018-2019

The first assignment focuses on genetic algorithms and evolutional computing. These techniques can be used to solve many optimization and search problems.

The assignments are to be completed **individually**. Presentations will take place during the lab sessions in the week of **10-16 December 2018**.

The assignment

The goal of the first assignment is to create a maze-solving algorithm using genetic programming. The algorithm should try to find the shortest path from the starting spot in the maze to the ending spot. The assignment data (assignment1.r) contains a number of functions that can be used in the implementation and evaluation of the algorithm. These functions are:

simulateSolution <- function(maze, solution, rows, cols): This function simulates a given solution on a given maze, and should be used in the implementation of the fitness function. Mazes are given as 1-dimensional vectors consisting of '#' for walls, ' ' for empty spaces, 's' for the starting position and 'e' for the ending position. These vectors are treated as 2-dimensional arrays. The *rows* and *cols* parameters specify the dimensions of the maze. Solutions are given as a 1-dimensional vector consisting of 'U' for moving up, 'D' for moving down, 'L' for moving left, and 'R' for moving right. Two examples of mazes and solutions are given in the assignment data.

The function ends the moment a solution reaches the ending spot of the maze. Any further moves are ignored. This function shall be used in the implementation of the fitness function.

moveRight, moveLeft, moveDown, MoveRight: These are helper functions for simulating the solution.

printMaze <- function(maze, rows, cols): This function prints the maze.

With the help of these functions, create a function called geneticAlgorithm that takes a maze, number of rows, number of columns, and other parameters needed for genetic algorithms and generates a path from the starting to the ending position of the maze that is as short as possible using a genetic algorithm.

For the **second part** of the assignment, update the implementation with the following additions:

1. Mazes can now also contain coins, marked with 'c'.

2. Visiting a spot containing a coin is worth 10 points. The coin is then removed from the maze.

Update the genetic algorithm (and the simulateSolution function) in such a way that it finds the path from the starting spot of the maze to the ending spot of the maze that collects as many points as possible, instead of finding the shortest path.

The tasks:

1. Implement the geneticAlgorithm function.
2. Experiment with different parameters (population size, mutation/crossover probability, replacement strategy, fitness function …).
3. Present an evaluation of how different parameters affect the quality of results and the speed of convergence. You can present the results on multiple different mazes to show how the size and complexity of the maze affects the results.
4. Repeat the first three tasks on the second part of the assignment

Grading

The final score will be based on the quality of the implemented algorithm, your exposition and justification of the chosen approach, and your interpretation and presentation of the results. The second part of the assignment (task 4) is not mandatory to obtain a passing grade (6), provided the first part is done well but is required for grades above 8.