Universidad de Puerto Rico
Recinto Universitario de Mayagüez
Departamento de Ingeniería de Eléctrica y Computadora

Artificial Intelligence Final Report

Grupo:
Julibert Díaz Collazo
Luis F. Quiles Ruiz
Manuel E Castañeda Neris

ICOM 5015/CIIC 5015 – 030
Fecha: 15 Mayo, 2020

I. Abstract

The purpose of this project is to compare the efficiency of two searching algorithms, A* and Simulated Annealing with the following objectives in mind: to design agents that maximize their utility, to implement the search algorithms to optimize routes and, compare the performance of both agents based on their travel time. We used csv files to build a non-directed graph representing the locations as nodes. Using a Simple Problem Solving Agent Program, we found that Simulated Annealing solution was consistently and considerably worse than the A* solution.

II. Introduction

Map and GPS services typically incorporate a route optimization element to select the fastest routes to reach the user's selected destination. This element uses traffic, average velocity, distance and other real-time data to search for the fastest travel times and dynamically change routes as the situation changes. There are search algorithms used to obtain routes and estimated travel times were A* and Simulated Annealing, with the goal of finding the fastest route. Once the routes are found, the A* solution will be considered optimal. The Simulated Annealing algorithm solution will then be compared against the A* solution to determine its performance.

III. Theory

The essence of this project is to explore and understand the intrinsics of A* and Simulated Annealing as representatives of heuristic and stochastic algorithms respectively. They employ different strategies and face different challenges when tackling a problem, for this they offer compromises. Stochastic can follow two types: Monte Carlo algorithms, like Simulated Annealing, always finish in bounded time, but don't guarantee an optimal solution, while Las Vegas algorithms aren't necessarily guaranteed to finish in any finite time, but promise to find the optimal solution; and heuristics don't guarantee to find the correct answer but aim for the most optimal. [1]

IV. Agents

An agent can be represented as a function that maps percepts to actions. It acts on an environment, from which it gets its percepts, to act in order to meet some performance measure or goal. The agent in a route optimization problem must search for optimal routes from its initial position to the goal. The environment for this kind of problem is a graph of a map, in which each node represents a location and has some distance-relation to another node. The percepts that the agent can get from this environment are the nodes, or locations of the represented map, and can act on it to change the node, location, or current state of the problem. The performance measure that it will seek to optimize is the estimated travel time of the solution, using an average speed.

A. **Agent Description**: Initially, we planned to use a utility agent but eventually found that, given our specific needs and goals, a simple problem-solving agent worked better. Because we're using the one specifically implemented within the aima-python repository, we follow its definition: *"A simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over.* [2]*"*

- **a)** *Agent type :* Single Agent
- **b)** *Fully observable* : The agent has all data in the CSV.
- **c)** *Sequential* : The next action or state of the agent depends on the previous actions or state reached by the agent.
- **d)** *Static* : The conditions of the state do not change while the agent deliberates.
- **e)** *Discrete* : The agent has only a finite number of options to choose for the next state.
- **f)** *Known* : The agent knows its function, data and goal, it does not need to learn from scratch or from an external source.

The abstract class for the simple problem agent in the aima-python repository was extended and implemented to suit the case for the current graph search problem. Using a given average speed and the problem-converted-map of the region in question, the methods were elaborated in order to execute the search algorithm and examine its performance. Once the problem is formulated, a solution sequence of the traversed nodes is obtained from the search algorithm. This sequence also contains the approximate travel time for the solution, using the average speed approximation, in order to compare the algorithms under the same conditions.

V. Environment

The graph shall be populated with locations, such as cities, with approximate distance information between the locations and GPS coordinate data for each. This data was stored in CSV files and then read to create the graph for the Puerto Rico map. This graph was then used to create the problem for the region, given its initial location and its goal destination. For initial testing purposes the Romania map problem discussed in class and a reduced amount of nodes were used in the graph representation for Puerto Rico. The approximation distance found in the aima-python repository uses distance in kilometers in order to calculate the straight approximation. However, because the Puerto Rico problem uses GPS coordinates, another approximation function was implemented to override the existing one for the Puerto Rico problem in order to accurately calculate the straight line distance between nodes and the goal. Once each algorithm's behavior was validated, the Puerto Rico map graph was expanded with more locations and routing alternatives to gain better performance statistics.

VI. Search algorithms

A. **A* Search:** *"What A* Search Algorithm does is that at each step it picks the node according to a value-'**f**' which is a parameter equal to the sum of two other parameters – '**g**' and '**h**'. At each step it picks the node/cell having the lowest '**f**', and processes that node/cell.*
   a) ***g*** *= the movement cost to move from the starting point to a given node.*
   b) ***h*** *= the estimated movement cost to move from that given node on the grid to the final destination.* ***This is often referred to as the heuristic, which is nothing but a kind of smart guess****.* [3]"

B. **Simulated Annealing:** "*The simulated annealing algorithm was originally inspired from the process of annealing in metal work… In simulated annealing we keep a temperature variable ... We initially set it high and then allow it to slowly 'cool' as the algorithm runs… This gives the algorithm the ability to jump out of any local optimums… allowing the algorithm to gradually focus in on an area of the search space in which hopefully, a close to optimum solution can be found.* [4]" In essence, simulating annealing progressively filters through local maxima testing out options until the agent finds an optimal or close to optimal solution. In this case, the temperature provides a means to determine stochastic elements.

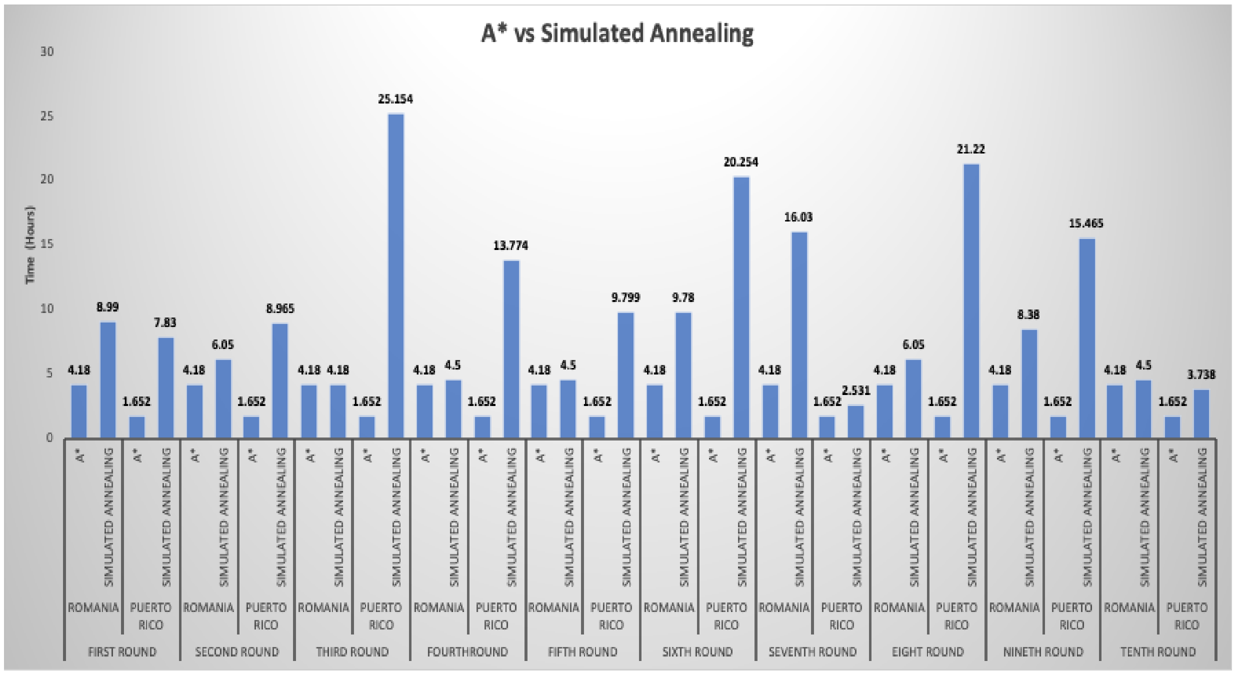
Velocity was taken as an average of typical speeds for the given region. The simulated annealing algorithm incorporates a probability to accept bad choices, simulating the effect that other elements, such as traffic and accidents, can have on expected times. This data was then used in the agent's functions to select the optimal routes. The A* algorithm was used to find the optimal route. The agent used distance between nodes and a straight line approximation to the goal as its evaluation function to determine the best routes to take. Once it found a route with minimum travel distance, a speed approximation was used to determine the route travel time. Validating that the evaluation function was both admissible and consistent was done using functions. Starting the problem at each node of the graph, the distance to the neighboring nodes of the current location and their straight line approximation to the goal was used to validate consistency. In order to validate admissibility, the same approach was taken but each possible path from the current location to the goal was explored to confirm that the straight line approximation was less than the actual possible costl. The agent then searched for a solution using the Simulated Annealing algorithm to optimize the route. The Simulated Annealing agent used the straight line approximation and scheduling function provided in the aima-python repository. Once the approximate travel time for each algorithm was found, the times were used to compare the performance of the agents.

- **Determine Performance**

The A* algorithm uses the distance between locations and their straight line approximation to the goal to find the least expensive choice in terms of distance. Due to its optimal behavior, it was used as the theoretical best route possible for the given problem. The simulated annealing algorithm can be adjusted through its scheduling function in order to make its solutions better, possibly reaching the optimal behavior of A*. However, because it incorporates the probability of accepting bad choices it is difficult to validate this. There, the solution found by the simulated annealing was evaluated based on its error difference with the A* solution.

- **Search Algorithms Comparison**

Figura 1:



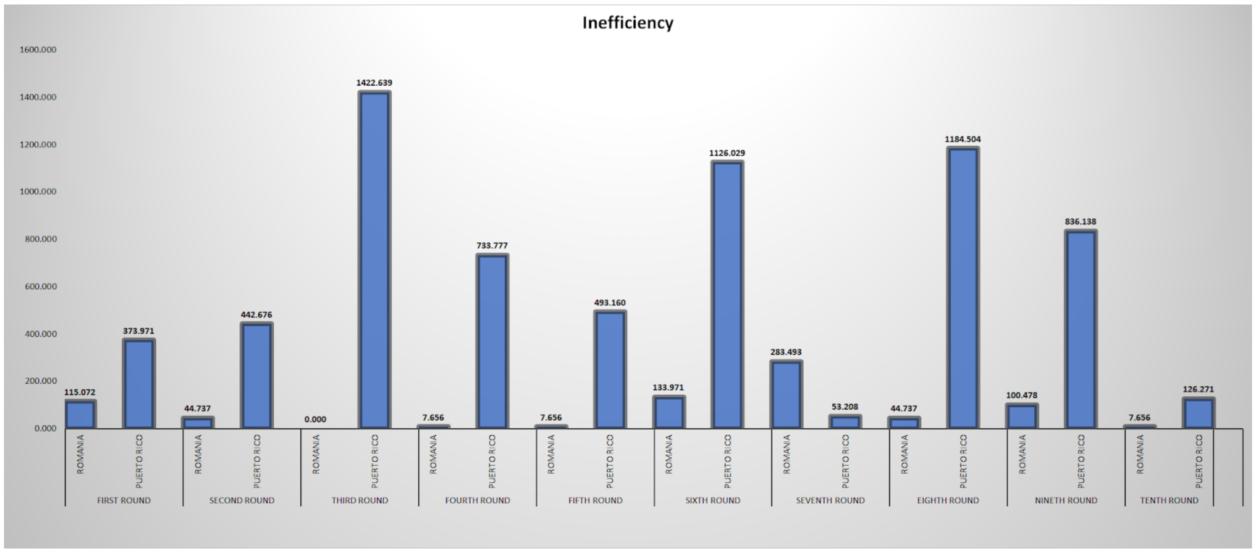
In Figure 1 data for the expected travel times of the solutions found for both the A* search and Simulated Annealing are shown. The data comparison between the A* search algorithm and the Simulated Annealing shows the time the agent took to travel to the desired goal. This comparison was done with the Romania and Puerto Rico map using the same average speed on both of them. The solutions obtained from the A* search algorithm remains consistent, while the resulting path of the Simulated Annealing becomes more random, because of its stochastic factor. This stochastic element is used to display the probabilities that either an accident or traffic can affect the current path during a given moment and how the algorithm manages to find an alternate path. The data shows that, overall, Simulated Annealing result greater time than the A* search, ranging from the same amount to almost four times the amount of time, like in the seventh round.

- **Performance Comparison**

  Performance measures:

  1. *Estimated time*: the approximate time it'll take the agent to reach the destination given the weather and traffic density.
  2. *Velocity*: the rate at which the agent moves from node to node.

Figure 2:



In Figure 2, the data of the percents of inefficiency between the A* Search and the Simulated Annealing are represented in a graph. The data shows the difference in performance efficiency by each case the algorithms were run. As displayed, the Simulated Annealing can occasionally have the same efficiency as the A* Search but it can likewise have an inefficiency percent of 1422 %.

VII. Conclusion

Both agents, while both admissible and consistent, displayed vastly different levels of efficiency given their nature, that is to say, the reason lies in the way they "decide" their next step. The stochastic elements of the simulated annealing algorithm hampered it's efficiency, and while it could stop the agent from getting "stuck" on a dead end or an endless loop, it could still make mistakes. Naturally, the point of this is for the agent to learn, but there's the of very inefficient paths being taken. This could possibly be rectified by adjusting and designing different scheduling functions in order to relate the decrease in temperature to the task or environment at hand. This could adjust the algorithm to make it less probable to accept mistakes and it is a promising area for future work and improvements.

Bibliographic references:

[1] j_random_hacker, “Difference between a stochastic and a heuristic algorithm,” *Stack Overflow*, 01-Oct-1964. [Online]. Available: https://stackoverflow.com/questions/28084941/difference-between-a-stochastic-and-a-heuristic-algorithm. [Accessed: 15-May-2020].

[2] Aimacode, “aimacode/aima-pseudocode,” *GitHub*. [Online]. Available: https://github.com/aimacode/aima-pseudocode/blob/master/md/Simple-Problem-Solving-Agent.md. [Accessed: 12-May-2020].

[3] “A* Search Algorithm,” *GeeksforGeeks*, 07-Sep-2018. [Online]. Available: https://www.geeksforgeeks.org/a-search-algorithm/. [Accessed: 25-Apr-2020].

[4] “Simulated Annealing for beginners,” *The Project Spot*. [Online]. Available: http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6. [Accessed: 25-Apr-2020].

[5] “Agents in Artificial Intelligence,” *GeeksforGeeks*, 06-Aug-2019. [Online]. Available: https://www.geeksforgeeks.org/agents-artificial-intelligence/. [Accessed: 25-Apr-2020].