Challenge 2:
Travelling Salesman Problem

Emilio Castillo [1], Miguel Monterrubio [2] y Eugenio Leal [3]
Intelligent Systems, Computer Department, Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Santa Fe.
[1] A01335139@itesm.mx, [2] A01022153@itesm.mx , [3] A01022983@itesm.mx

Abstract. The following article aims to demonstrate the learning obtained within the Intelligent Systems class, within it the concept of Artificial Intelligence, This article includes an introduction to Travelling Salesman Problem challenge, explanation of our solution and a brief user manual to install and run the code.

Key words: Intelligent Systems, Artificial Intelligence, Travelling Salesman Problem, behavior, environment, agent, python, reactive agent.

1 Introduction

The article was developed with the purpose of accrediting the second partial of the subject "*Intelligent Systems*" taught at the Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Santa Fe.

As a starting point, it is necessary to understand the following concepts: *informed search and uninformed search* and their principal differences.

The informed search aims to reduce the number of searches to be carried out using a smart node selection process (for expansion). This type of search uses heuristics to evaluate the probability that the node is part of the solution.

Compared to the informed search, the uninformed search (also known as brute force) generates a search tree without the use of domain-specific knowledge.

The objective of this experiment and investigation is to implement a Traveling Salesman problem using informed and uninformed methods.
Throughout the article the following will be presented

- Problem Analysis and the explanation of the solutions.
- The computational requirements necessary to run the application.

2 Problem Analysis

2.1 Description

The Travelling Salesman Problem describes a salesman who must travel between N cities. The order in which he does so is something he does not care about, as long as he visits each once during his trip, and finishes where he was at first. Each city is connected to other close by cities, or nodes, by airplanes, or by road or railway. Each of those links between the cities has one or more weights (or the cost) attached. The cost describes how "difficult" it is to traverse this edge on the graph, and may be given, for example, by the cost of an airplane ticket or train ticket, or perhaps by the length of the edge, or time required to complete the traversal. The salesman wants to keep both the travel costs, as well as the distance he travels as low as possible.

The Traveling Salesman Problem is typical of a large class of NP-hard optimization problems that have intrigued mathematicians and computer scientists for years. Most important, it has applications in science and engineering. For example, in the manufacture of a circuit board, it is important to determine the best order in which a laser will drill thousands of holes. An efficient solution to this problem reduces production costs for the manufacturer.

2.2 Our Approach

2.2.1 Jupyter Notebook

The team use Jupyter Notebook for program and show the solutions, for more information and details of the code, please consult the attached file.

2.2.1.1 Generating Cities

The first step is to generate a group of random cities, this group is created in order to demonstrate the function of the soluticions.

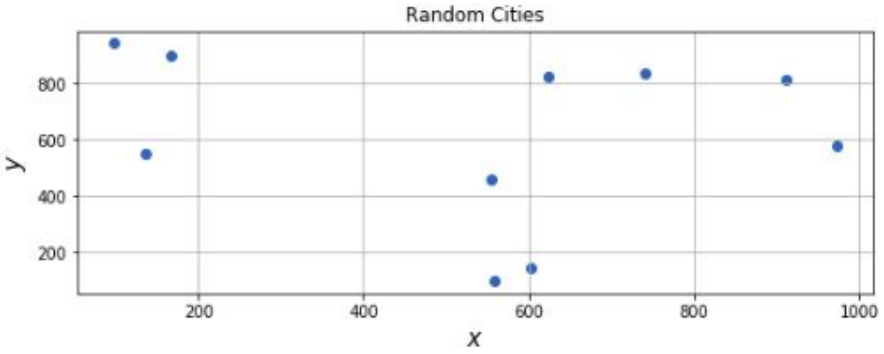


Figure 1. *Group of cities.*

2.2.1.2 Random Shuffle

Shuffle randomly to get the best possible answer. This method could converge faster than brute force but is important to know that this solution does not ensure global optimum.

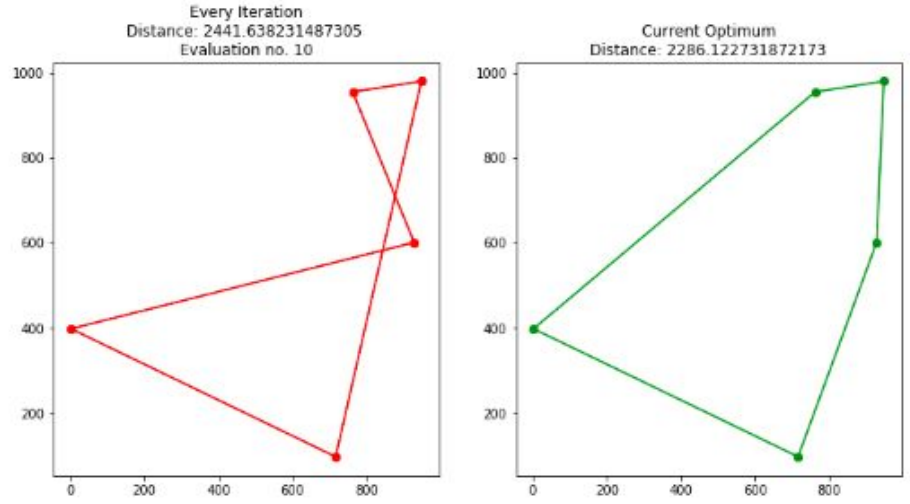


Figure 2. *Random Shuffle - Uninformed Solution.*

2.2.1.3 Brute Force

For this solution, the code generates and get all the possible permutations of the cities and test all possible combinations.

Where all possible combinations is: $factorial(nCities)$

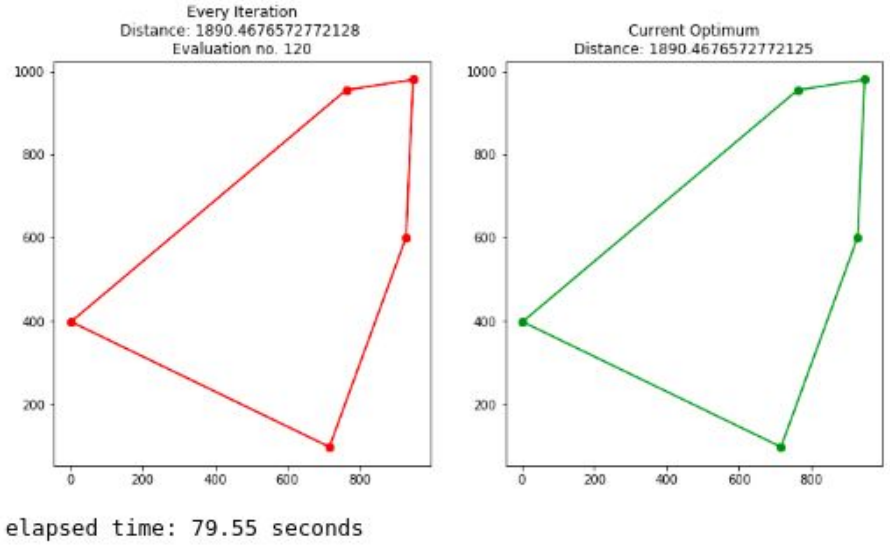


Figure 3. *Brute Force - Uninformed Solution.*

2.2.1.4 Greedy

Greedy Search is an intuitive algorithm, which performs a local search and therefore makes the best decision in each state, it is important to note that this algorithm does not guarantee the best solution.

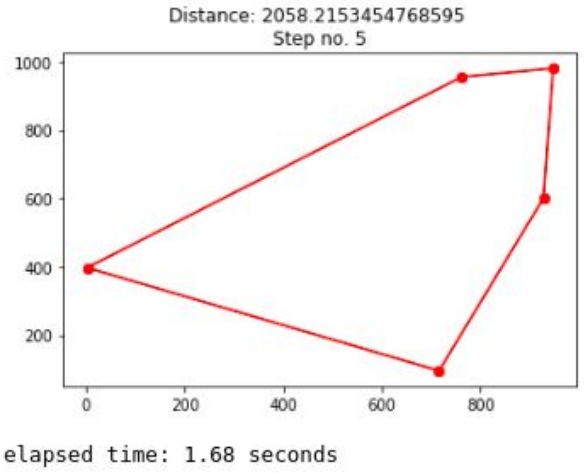


Figure 4. Greedy Search - *Informed Solution.*

2.3 Improvements

In this section, a list of continuous improvements for the Travelling Salesman Problem is shown:

- Better visualization tools.
- Implement the A*algorithm for the Uninformed Search
- Fast Execution.
- Interaction with other types of data.
- Improve processing speed in order to implement more cities.

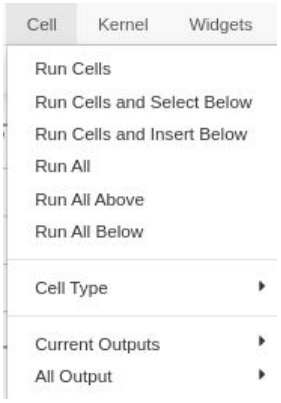
3 User Manual

3.1 Installation Dependencies

- **Install Python 3**
  - Install python 3 on your computer
  - https://www.python.org/downloads/release/python-380/
- **Install jupyter notebooks**
  - https://jupyter.readthedocs.io/en/latest/install.html
  - **the jupyter docs highly recommend installing anaconda**
    - **install Anaconda**
      - https://docs.anaconda.com/anaconda/install/

3.2 Execute the program

- Clone the following repository: https://github.com/monterrubio-miguel/pyTSP.git
- Open a terminal in the corresponding directory
- Start jupyter Notebook server:
    - run in terminal: *jupyter notebook*
    - See the results of the command in the terminal and copy and paste URL from terminal to web browser; example: http://localhost:8888/?token=YOUR_TOKEN
    - open .ipynb file
    - you can modify, run an individual cell or run all cells, example:



4 Conclusions

Emilio. This activity was useful for a more efficient understanding of the main differences and similarities between the informed and uninformed search algorithms. It also helped me improve my reports in the LNCS format.

Miguel. This activity helped us understand the differences between informed and uninformed algorithms, as well as efficiency, accuracy, and usefulness of each one in resolving the Travelling Salesman Problem.

Eugenio. With this activity we learned several algorithms to tackle the Travelling Salesman problem and for that matter, general optimization problems in which we

used an Uninformed and Informed approach to create an optimized real-world solution.

References

1. “Greedy Algorithms.” *Brilliant Math & Science Wiki*, brilliant.org/wiki/greedy-algorithm/.

2. “Informed Search.” *Informed Search*, people.sju.edu/~jhodgson/ugai/infsrch.html.

3. “Uninformed Search.” *Usearch*, cs.lmu.edu/~ray/notes/usearch/.