A Traveler's Problem

Thu Tran (tran0623)
Sandhya Ratnam (ratna031)

May 11, 2020

1 Abstract

The Traveling Salesperson Problem (TSP) is one of the most discussed problems in artificial intelligence where a salesperson visits a set of cities once before returning home. We are exploring a subset of this problem by applying it to planning a trip for a user. The breadth-first search and greedy best-first search algorithms can be applied to this situation to find routes that can be generated for the user based on the user's input. The greedy best-first search stops searching after it has found a route to visit each of the countries. The breadth-first search algorithm, on the other hand, finds the optimal route by iterating through all of the possible options and calculating the time it takes to travel to each of the countries. However, both of the algorithms provide valid routes for the user.

2 Introduction

The novel, "Around the World in Eighty Days", by Jules Verne, has inspired us to use a conjunction of different artificial intelligence concepts, such as search algorithms and the traveling salesperson problem, to provide a route for a user to use to travel between countries. This story is about two travelers who attempt to circumnavigate the world in eighty days in order to win a bet. They have many interesting experiences and encounters along the way as they travel to each country before returning home.

There are certain constraints involved with planning an international trip, such as time to travel, the countries to visit, and where you begin your trip.

For those who love traveling and want to visit as many countries as possible in one trip, modern technology can be utilized to find a route that visits all of the countries that they want to go to and could potentially maximize their trip potential. Technology can help reduce the manual labor involved with planning the route. Many times, people will seek assistance in a travel agency to aid with the planning of their trip, who can use our solutions to find the best route, given the user's constraints. We have devised two algorithms that take into account a time constraint, starting location constraints, and countries to visit constraints to find a route for the user.

Users are able to input the specific countries that they want to visit and have the option to enter the duration of the trip in days. They can visit any number of countries that they wish, whether it is just one or all of the 241 countries that are accounted for. If the user wants to visit n countries, then there are n! possible routes. The time constraint also plays a major role because it may not be possible to visit all of the wanted countries in the specified time frame. Rather than trying to figure out all of the possible routes and if it is possible to travel to those countries in each route in the specified period, the algorithms provide solutions for the user and take into account the time. Based on the user's input, different routes will be generated and analyzed based on the distance between each city. Two routes will be generated. The first route is generated using the greedy best-first search algorithm. The "greedy choice" is defined as the closest neighbor country to the current country. The second route is calculated using the breadth-first search algorithm. This algorithm is used because it will iterate through every possible route option and find the optimal route.

Many algorithms can be used to find a route. Some may provide the optimal solution, while others may not. Comparing the solutions of different algorithms provides insight into the scenarios that the algorithm is more useful for.

3 Literature Review

3.1 Introduction

To find a route, there are many factors involved, including the countries that want to be visited, the distance between each country, and the length of the trip. Research has been done on similar subjects related to

route planning. This literature review will outline the traveling salesperson problem (TSP) in addition to the breadth-first search and greedy best-first search algorithms, which were used to find solutions. In "Artificial Intelligence: A Modern Approach", written by Norvig and Russell, the authors provide information regarding how the TSP and different search algorithms work [8]. In the TSP, each node must be visited only once and the shortest route should be found. This is useful when deciding paths for those to travel because travelers will visit each place only once on their trip. This reference also provides details about different search algorithms that are utilized in this experiment, such as breadth-first search and greedy best-first search.

3.2 Traveling Salesperson Problem

There have been many studies conducted regarding the algorithms to carry out the TSP. The "Comparison of Algorithms for Solving Traveling Salesman Problem" article in the 2015 International Journal of Engineering and Advanced Technology compares various heuristics algorithms to solve TSP [3]. The article examines 3 heuristics algorithms, the nearest neighbor, genetic, and greedy heuristic algorithms based on their run time, number of iterations to get to a solution, and the quality of the solutions on a small set of cities (n=20) and a large set of cities (n=100 and n=1000). Their research concludes that the greedy heuristics algorithm yields the closest result to the optimal solution, while genetic algorithms might not converge at all. This is useful when determining the best algorithm to solve our version of TSP with a static initial city and 241 countries in the set.

In another study, part of "Traveling Salesman Problem Theory and Application", however, the authors discuss how impossible it is to find the optimal solution of a TSP, especially of a TSP with larger size, and examines the efficacy of approximate approaches [5]. Nevertheless, different heuristics can be used to solve the TSP. These include the closest neighbor, greedy, insertion, and Christofide heuristics. The goal is to determine a heuristic that works well with a greater size TSP. The "Traveling salesman problem heuristics: Leading methods, implementations and latest advances" article analyzes the computational performances of leading heuristics for the TSP problem, specifically the Lin-Kernighan (LK) and stem-and-cycle (SC) methods [7]. The paper concludes that SC approaches provide better solutions but have longer run time than the basic LK approach. The SC approach will be important to consider to achieve heuristics that get closer to the optimal

solution.

3.3 Greedy Search Algorithms

The goal of a greedy search algorithm is to make the optimal choice at each stage in order to reach the goal as fast as possible [8]. The "A Comparison of Greedy Search Algorithms" proceeding compares algorithms within best-first, hill-climbing, and bean search families to solve the shortest path problems [9]. The paper aims to recommend suitable algorithms for a variety of situations based on cross-family comparisons. The proceeding includes a cross-family comparison to solve the TSP, which includes A*, LSS-LRTA*, Window A*, and Beam searches. The paper rules out the use of beam search in to solve the TSP, since for a map with 100 cities, their beam search implementation takes 10 seconds to return the result while weighted A* takes 2 seconds. The different greedy search algorithms will be taken into account to create a route for the user.

3.4 Breadth-First Search Algorithm

According to Norvig and Russell, in a breadth-first search (BFS) algorithm "all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded" [8]. Though the optimal solution will be found, efficiency will come at a cost because every node will be looked at. The research paper, "Planning as heuristic search" discusses different heuristic search planners [4]. BFS is considered and performs well in comparison to the hill-climbing search algorithm when considering "the number of problems solved, the time to get those solutions, and the length of those solutions measured by the number of actions." Compared to the standard best-first search, the BFS can be more efficient in terms of memory because it requires less memory to prevent the opening of closed nodes [10]. It also outperforms other systematic search algorithms as graph-search problems become more complex. The BFS can also detect duplicate nodes. However, it can result in the worst case when there is a tie.

In addition, Tamás Kádek and János Pánovic's research in "Extended Breadth-First Search Algorithm" discusses a type of BFS known as the extended breadth-first search (EBFS) in more detail [6]. This algorithm is able to handle larger state spaces. It is also able to run more than one breadth-first search starting from multiple known states. With EBFS, more information

is stored. For example, the relationship between each initial state and node is known. It is more efficient than the standard breadth-first search when the state space is large. This algorithm is useful if the user selects a large set of countries to visit.

3.5 Conclusion

There have been many studies done on various search algorithms to see which provide the best solutions in different scenarios. When working with the TSP, greedy search heuristics and SC methods have been shown to perform the best. By utilizing different search algorithms, the difference in solutions can be analyzed and provide insight into the algorithms.

4 Approach

The steps to solve this TSP-modified problem include gathering data on distances between countries, creating a representation of the problem, applying breadth-first search and greedy best-first search to solve the problem, and calculating statistics to compare the solutions.

The "country-capitals.csv" file contains countries, their capitals, and the latitudes and longitudes of the capitals. For the purpose of the problem, it is assumed that the traveling person would visit the capital of the country, thus the latitude and longitude of the capital will be representing the latitude and longitude of the country. The Haversine formula is then utilized to calculate the distance between countries based on their latitudes and longitudes. The time to travel between countries is then estimated as the time it takes to cover the distance at 900 km/h, the typical cruising speed of a commercial airline. Therefore, the assumptions are that the traveler is traveling by plane and that the traveler travels consecutively without resting in any country.

The problem domain is represented as a nested dictionary structure in Python that stores all of the countries that the traveler wants to visit and the distance in kilometers from each country to all others in the domain. The time, in hours, to travel is calculated as distance divided by 900 km/h.

In a traditional TSP problem, the goal state is a situation where all cities have been visited and the traveler returns to their starting city. However, since there is a limit on the number of hours the traveler can travel in this experiment, the constraints will have to be incorporated into the goal states.

The goal state is defined as a situation where the traveler has returned home, either visited all countries in the domain or has reached the maximum time limit to travel. The results returned by the search algorithms shall contain the route and the cost to cover the route. A valid route shall end with the same country the route starts with. The solution cost is defined as the time it takes to cover the route in order. When the time limit is infinite, the solution must contain all countries the traveler wants to visit and the last country in the route is the starting one. When the time limit is finite, the solution cost must be smaller or equal to the time limit.

To compare the efficiency and results of the BFS and greedy best-first search algorithms, statistics were generated. Ten different problems were randomly generated for routes of size 3, 5, 8, and 10. For each problem size, two statistics were calculated, the average time it took to run the set of problems, and the average cost of the solution in hours. This allows for comparison between the two algorithms based on how fast each of the algorithms run in each scenario and the solution's cost in terms of how many hours it takes to travel the full route. The results of these trials are recorded in Tables 1 and 2.

5 Design

Solving the problems include building the knowledge base, representing the problem domain, implementing BFS and best-first greedy search, and calculating the statistics for analysis. To build the knowledge base of the location of each country, a function called read_csv() was created to read the "country-capitals.csv" file [1]. The function returns a dictionary with the country name in string format as the keys. The value associated with each key is a tuple containing the key's capital city as a string, the key's latitude as a float, and the key's longitude as a float.

The representation of the problem domain was inspired by the file "tsp.py" in the aima-python repository, the code base associated with "Artificial Intelligence: A modern approach" [2]. In 'tsp.py', the problem domain is represented as a nested dictionary, where the keys are the cities. The value of each key is another dictionary with all other cities in the domain as the key and the distance between those to the key city in the first layer as value. Similarly, the function construct_distance() constructs a nested dictionary where the keys are the countries in the domain. The value of each key is another

dictionary with all other countries in the domain as key and the distance between those to the key country in the first layer as value. The function's parameters are the knowledge-based dictionary returned from read_csv() and a list of countries' names in a string that the user wishes to be included in the route, assuming that the selected countries are in the knowledge base. Since the knowledge base only contains the latitudes and longitudes, function construct_distance() also needs to convert the longitudes and latitudes between countries to distances in kilometers using the Haversine formula.

The best-first greedy search function and the breadth-first search function take in the starting country as a string, the problem domain as a nested dictionary, and the time limit in hours. If the time limit is not provided, then it is assumed to be infinite. The time to travel, in hours, between two countries is calculated by dividing the distance between them by 900 km/h, the estimated speed of a commercial airliner. In the best-first greedy search implementation, the greedy choice is defined as the nearest neighbor country to the current country. In breadth-first search implementation, the most optimal solution is defined as the route with the most number of cities with the least amount of time to visit. That is, if a solution allows the user to visit more cities than another, that solution is chosen. If two solutions allow the user to visit the same number of cities, the solution with lower cost, or lower time to travel, is chosen.

The quality of the solution is determined by measuring the average cost of a solution and the time to generate the solution from 10 randomly generated problems with a predetermined size. Specifically, 10 randomly generated problems are generated for maps containing 3, 5, 8, and 10 cities. Two functions were created to collect statistics on the quality of best-first greedy search and breadth-first search. Function solution_quality_stat() computes the average costs of the solutions generated by each algorithm. Function performance_stat() computes the average time in seconds for each algorithm to solve the problem. The program was run on a Microsoft Windows 10 Education operating system on an x64-based PC with Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz, 2195 Mhz, 2 Core(s), 4 Logical Processor(s) processor.

The statistics were recorded and analyzed in Microsoft Excel. The best-fit function is used to estimate the time the algorithms will take to solve a problem with a larger number of countries. This is useful because although it is possible for the best-first greedy search to generate a solution for a route with all 241 countries, it is not feasible for breadth-first search to do so on the

aforementioned machine configuration. The best-fit function helps estimate the time breadth-first search will take to generate a solution for a route with all 241 countries.

6 Results

Size (countries)	BFS (hours)	Best First Greedy Search (hours)
3	27.1557	27.1557
5	36.1239	37.0341
8	48.2647	51.7457
10	53.1500	57.5657
241		189.8522

Table 1. Average solution costs of BFS and Best First Greedy Search across varying problem size.

Size (countries)	BFS (seconds)	Best First Greedy Search (seconds)
3	3.9200e-05	3.9200e-05
5	3.7167e-04	4.8000e-05
8	7.1585e-02	1.1833e-04
10	214.81	1.4598e-04
241		6.9798e-03

Table 2. Average time to generate a solution by BFS and Best First Greedy Search across varying problem size.

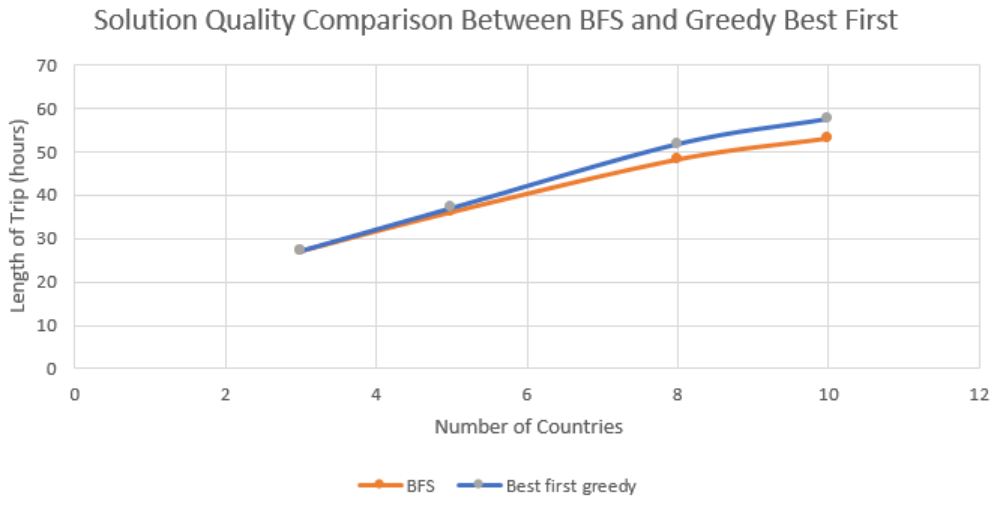

Figure 1. A graph of average solution quality of BFS and Best First Greedy Search across varying problem size.

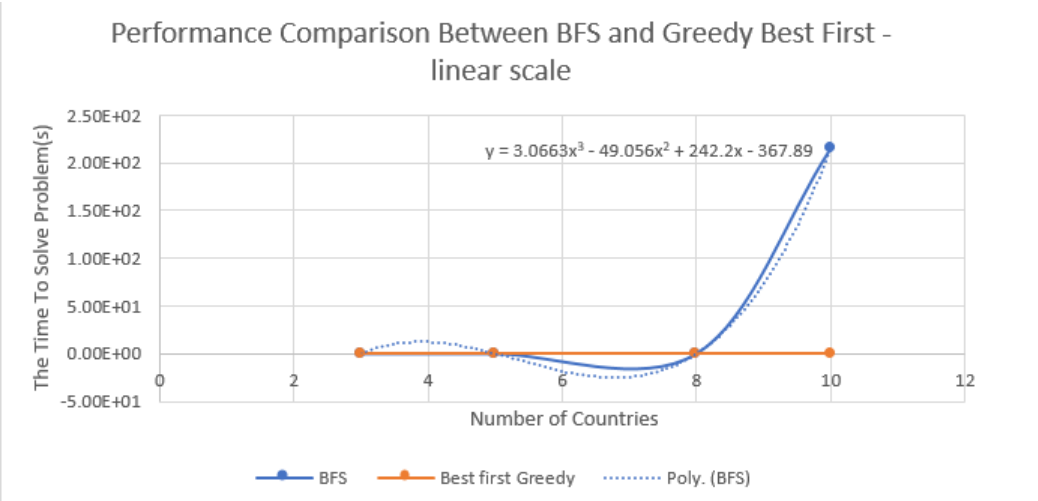

Figure 2. A graph of average time to generate a solution by BFS and Best First Greedy Search across varying problem size.

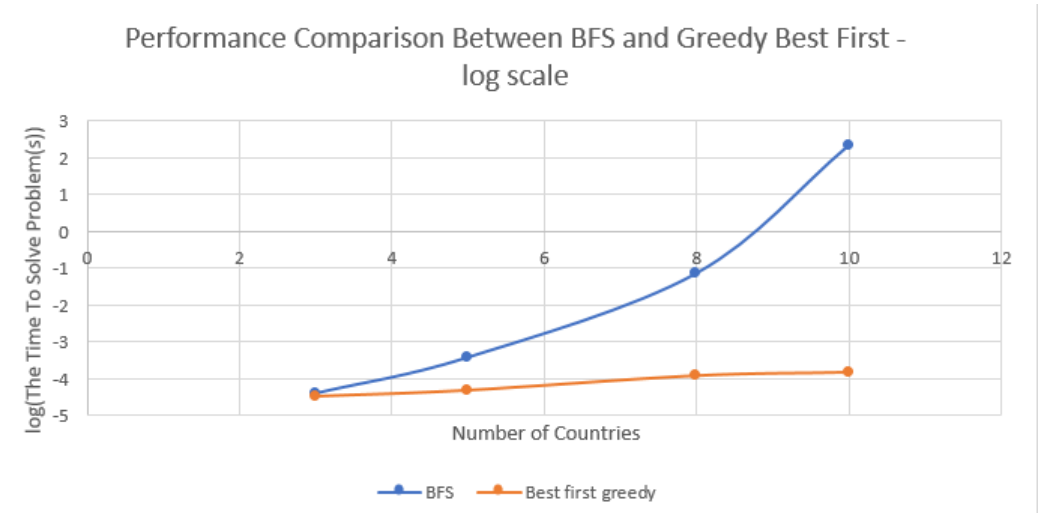

Figure 3. A log-scaled graph of average time to generate a solution by BFS and Best First Greedy Search across varying problem size.

7 Analysis

By testing out different sized routes, the difference in solutions was more noticeable. Looking at the solution quality results, with inputs ranging from 3 to 10 countries, BFS and greedy best-first search reach the same solution when the input size is 3. However, as the number of countries increases, the BFS results in finding shorter paths.

The results show that the greedy best-first search came up with a solution before the BFS in every case, according to Table 2. It takes less time to provide a solution because the algorithm ends after it finds the first possible solution. While the greedy best-first search was able to provide a solution to a problem containing all 241 countries, the BFS was unable to do this in reasonable time due to the large size of the data set. BFS provides the optimal solution, and the best-fit function of its performance time is a third-order polynomial. In contrast, the best-first greedy search performance time grows linearly as the number of countries in the problem increases.

When it came to the quality of the solution for each trial, the BFS provided better results, as shown in Figure 1. The length of the trip for BFS is generally lower for each set of the number of countries. This is because the optimal solution was found in each case rather than the first solution, like in greedy best-first search. However, using BFS for trials of more than ten countries was not realistic because of the length of time it took to expand each node.

In Figure 2, the time it took to come up with each solution is plotted for both of the searches. In Figure 3, the log of the times from Figure 2 are plotted. The trend line is also shown for BFS in both figures. The complexity of the BFS algorithm is O(n!) compared to O(n) for greedy best-first search. For BFS, the trend is exponentially increasing at a rate faster than that of the greedy best-first search. This greedy search performs better overall when looking at the time it takes to find a solution.

Originally, a route consisting of all of the countries was to be included in the results. However, because BFS iterates through every possible option, getting the experimental time that it takes to produce a solution is unrealistic. Instead, the time can be estimated using the trendline in Figure 2. This line, y = 3.0663x3 - 49.056x2 + 242.2x - 367.89, provides a polynomial relationship between number of countries and time. This equation can be used to estimate how long it would take to find a solution using the BFS. For 241 countries, it is estimated to take 40129379.4163 seconds. This is equivalent to over 464

days. For the greedy best-first search, the 241 countries took 0.006979751587 seconds to generate a solution. The large difference in these times is due to the different purposes of each algorithm. In addition, the route would take 189.8522496 hours to complete.

When considering a realistic user, it is reasonable to assume that in most cases less than ten countries will be visited. In this case, using the BFS algorithm would be beneficial because the optimal solution would be provided. However, there are many users who may want to travel all over the world and visit more than ten countries. When this is the case, using the greedy best-first search is more beneficial. This is because it provides a solution in a reasonable amount of time.

8 Conclusion

To summarize, finding a route for a user based on the countries that they want to visit interleaves several different artificial intelligence components, such as Traveling Salesman Problem, breadth-first search, and greedy choice. Because planning a trip for a user is a subset of this problem, we are able to use the same techniques to find a solution. The breadth-first search and greedy best-first search were used in this case to find possible routes that results in varying solution costs and performance time.

Overall, the two algorithms were able to provide a route for the user based on the input countries and time constraints. BFS found an optimal solution when the input was less than ten countries, while the greedy search found a solution that met the constraints. When it comes to deciding whether to use BFS or greedy best-first search to plan the trip, the number of wanted countries need to be taken into consideration. If the number of countries is lower than 10, BFS can be used to generate a more effective route. Otherwise, greedy best-first search is the only choice to come up with a solution in reasonable amount of time. It is safe to conclude that finding the optimal solution for large number of countries based on the findings of this paper and other papers on TSP. However, it is worthwhile to research different heuristics, such as stem-and-cycle or Lin-Kernighan, and algorithms, such as genetics algorithms, to generate solutions as close to the optimal solution as possible.

9 Division of Work

Each section of the report contains contributions by both members. Sandhya mainly worked on the abstract, introduction, analysis, literature review, conclusion, results, and code. For the code, she focused on the BFS and solution quality. Thu focused on the code, literature review, design, conclusion, approach, and results. She wrote much of the code to calculate the distance between countries, greedy best first search, and performance statistic.

References

[1] List of countries and capitals, http://techslides.com/list-of-countries-and-capitals.

[2] Python implementation of algorithms from russell and norvig's "artificial intelligence - a modern approach".

[3] A. Abdulkarim and I. Alshammari. Comparison of algorithms for solving traveling salesman problem. 2015.

[4] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1):5 – 33, 2001.

[5] Donald Davendra, editor. *Traveling Salesman Problem, Theory and Application.* 2010.

[6] Tamás Kádek and János Pánovics. Extended breadth-first search algorithm. 2014.

[7] D. Gamboa F. Glover Rego, C and C. Osterman. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. 2011.

[8] S Russell and P. Norvig. *Artifical Intelligence: A modern approach.* Pearson, third edition, 2009.

[9] Jordan Thayer Wilt, Christopher and Wheeler Ruml. *Proceedings of the Third Annual Symposium on Combinatorial Search (SOCS-10).*

[10] Rong Zhou and Eric A. Hansen. Breadth-first heuristic search. *Artificial Intelligence*, 170(4):385 – 408, 2006.