Combining Artificial Intelligence Methods to Optimize Bus Routes in a Variable Environment

Soham Patil, Cody Waldecker

Abstract - When there is a task with the freedom of many potential solutions, humans can have a hard time finding an optimal solution in a sea of options. Computers, however, are faster at finding solutions because they can analyze all of the available options more rapidly than humans. This paper will be demonstrating the use of artificial intelligence in the development of bus routes in an area. For this, we need to consider the bus stops, the distance from one stop to the other, and the traffic in the area. Using all of that information, we can draw the best bus route for every bus in a district. In the event of a changing environment such as a missing bus driver or changes in traffic, a genetic algorithm can adapt quicker to the changes than a human. In this paper, we will use a combination of a branch and bound search and a genetic algorithm to determine the optimal route a bus should take to pick up students in a particular area. In addition, in the event that a machine does not provide the best route due to a lack of time for testing alternate routes, we demonstrate that it still finds a better route than a human given the same amount of time. Our use case demonstrates that a machine learning algorithm is capable of performing a task with more efficiency and better results than humans alone. These results provide evidence that machine learning using a genetic algorithm may be used throughout the commercial industry to improve productivity in many fields.

1. Introduction

This paper uses a combination of two artificial intelligence methods to determine the fastest route for a bus to pick up students on the way to school. The simulation setup involves a predefined grid of streets, the locations of the school and the stops, and traffic conditions between each intersection on the grid. Once the environment is constructed, an initially random route is selected as the baseline and the solution is further refined. The solution is refined through the use of a genetic algorithm that determines the optimal ordering of each stop along the route. This genetic algorithm then makes use of a branch and bound search algorithm which determines the fastest route from one stop to another, for example from the school to stop A.

A genetic algorithm is an operation in computer science that is inspired by natural selection. Genetic algorithms are commonly used to find solutions for problems where there are many different possible outcomes. Genetic Algorithms work by using selection to refine generations of solutions to get better results over the course of many generations. The process is initialized with a single generation with a defined number of population members. Then, each of those population members is identified by what is called a genome, it is usually a set of binary numbers that can be used to determine an artificial “score” for each member of the population. Each member of a generation is evaluated on its score and, like natural selection, the members with the best score are used to determine the members of the next generation. This selection process continues for a predetermined number of generations and ultimately refines the solution space over time to converge on an optimal solution. The genome construction and scoring will be further investigated in a following chapter.

In order to determine the score for each population member in a generation, a branch and bound search is utilized to find the fastest route between two stops and return that time to the genetic algorithm. The branch and bound search used here is a breadth first search algorithm. The search is best visualized as a tree structure where each branch of the tree represents a potential solution to get from point A to point B. The branches are differentiated by what route was taken since there are many solutions available depending on what turn is made at each intersection. The breadth first search starts with an initially seeded path neglecting the traffic conditions. Then additional branches are broken off of the initial route at each intersection and the process continues using the same train of thought. The search down each branch continues until it reaches the desired point B, where it is evaluated to see if it is the fastest current solution, or until the time taken exceeds the current best time. This method reduces the need to search a wide solution space for every option available by terminating the search when a branch no longer offers an optimal solution. The branch and bound algorithm utilized here will be discussed further in a following chapter.

2. Analysis

2.1 Simulation Setup

The initial task for this research is to set up a simulated grid of bus stops and traffic patterns. Instead of a large map with many intersections, dead ends, and changing traffic conditions, an n-by-n grid with stops and traffic conditions is generated based on the user input. This creates the environment in which our genetic algorithm will find the best route between 2 or more stops. There were two main variables factored in to create the grid, the number of stops and the number of grid lines. Using these two variables, a grid of any shape with as many stops as desired can be generated. In addition, the generation of the bus stops is randomized so that the same seed is never tested twice. To indicate the traffic conditions, the programmed time to transfer between grid points is assigned a value of 1-3 and the traffic pattern is output to a plot indicating a green line if there was less traffic, a yellow line if there was moderate traffic, and a red line if there was heavy traffic, see Figure 1 below. These lines are placed on the grid lines so the genetic algorithm can also consider traffic.

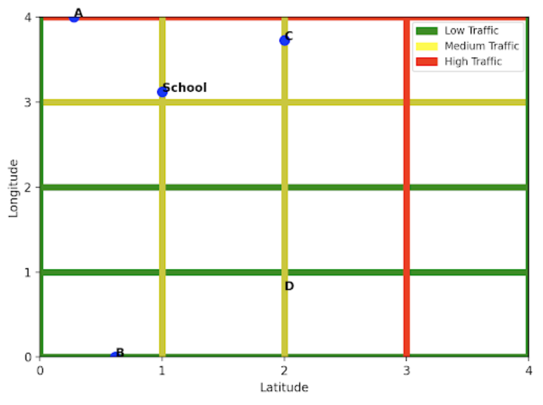

Fig. 1 Simulation Environment Example

2.2 Branch and Bound Algorithm

With the simulated traffic and stops, a genetic algorithm may be used to find the best route between stops, however, this can be sped up by implementing a branch and bound search to collect transfer information. A branch and bound algorithm is used to search along different solution path branches, for example going from one bus stop to another, and optimizing the route taken to reduce the time spent traveling in this scenario.

In order to seed the problem, a dictionary is created to hold a simulation of the grid. This dictionary will simulate all of the intersections and every node that the branch and bound algorithm goes through. Each intersection then has a number assigned to it, for example, the top left corner is indicated “1”. With the knowledge of which intersections there are on the grid, the branch and bound algorithm is initiated. The algorithm is then initialized with a shortest route by first locating the closest intersection to the stop, taking into account the direction of the desired stop. After the closest intersection for the first point and the second point (using the same logic) are located, a breadth-first-search algorithm to find the best route between the two routes is initiated. The initial seed is generated by creating a queue of the first point (the nearest intersection for the starting point) and finding the shortest route between the two (fewest number of turns). The route is saved as a list of the intersections it traveled to, for example, if there are two points and the route was going to be “1” to “5”, the path traveled would be “1”, “2”,“3”, “4”, “5”. Even though the path was created, there has to be a way to use directions(URDL) to make it easier for the user to understand the simulation. In order to do that, each one of the intersections is encoded with a direction. This makes the route much more readable to the person finding the best route.

The optimal path depends also on the traffic conditions and the size of the grid, where if the size of the grid is bigger, there might be a better alternative route. For that, the branch and bound algorithm is used. With the initial guess, further options are evaluated and weighted using the traffic conditions along each route. By branching at each intersection and terminating the depth search when the current best time is exceeded or it reaches the target destination, the solution space is reduced and the true best route is found. A visual of the branch search may be found in Figure 2 below where each branch represents a potential solution for the transit from A to B.

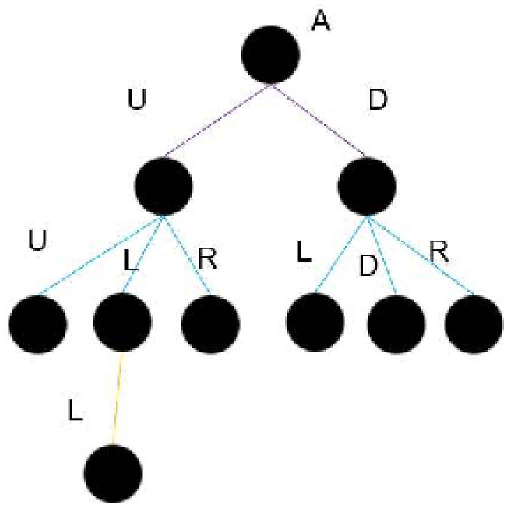

Fig 2. Branch and Bound Visualization

Above, the orange line indicates the first solution found, which follows the path (Up-Left-Left), then, all of the other branches are continued until they either reach the target destination or their time exceeds the current best solution.

2.3 Genetic Algorithm

A genetic algorithm is an operation in computer science that is inspired by natural selection. Genetic algorithms are commonly used to find solutions for problems where there are many different possible outcomes. Genetic algorithms use selection to refine the current best solution over time. Genetic algorithms guarantee the best outcome while humans doing the same thing takes much longer and may not guarantee the best outcome.

Now, the optimal path between two stops has already been found. For real world scenarios, it is needed to find the best route between more than two stops at a time. This is where the genetic algorithm comes into the picture. The genetic algorithm allows the user to find the best combination of stops to visit to have the least amount of time taken over the course of the route. If there are 4 stops that the bus needs to go through, then an example final route would look something like B, C, A, D. This provides the order of stops for the breadth first algorithm to go through, not necessarily the time or route it takes.

The way that the genetic algorithm works is simple to understand. First, defining the number of generation and generation size can increase the accuracy of the result. If there are few stops in a small area, it is better to use a smaller number of generations and generation size to save time in the program. The algorithm first defines the first generation. The first generation is a random variation of the stops. For example, (B, C, A, D) may be a member of the first generation. The algorithm then finds the score for that route. The score is the total time traveled considering the current traffic conditions. For example, if the route was (A, B, C, D), then the program would calculate the distance from A to B then to B to C then from C to D. If the previous best score requires more time than the score for this route, this route is saved as the best option. The best genome of the first generation is called a parent.

In the second generation, the algorithm re-assigns and re-shuffles each combination of stops and those are called the children. It does the same process for the same generation, replacing each generation if the score is less than before and assigns it as a parent. It does it for the remaining number of predetermined generations.

In order to accurately model the natural selection process, a mutation function is applied for each new generation. Before the random children are generated, the previous best solution is mutated by swapping two of the genomes. Those mutated members are then included in the next generation along with the other randomly generated children.

2.4 Results

After the algorithm is complete, it outputs the best route and best time along with a plot of the best route on the graph between two points. Every time the bus moves to the next stop, it plots the route it took on the grid, telling the user where the bus went and making it easier to understand where it is going next. An example of one optimal route may be found in the figure below, which was converged on after less than 2 seconds of computer run-time. This trivial amount of time to converge on an optimal solution is not only faster than a human, but also likely presents a better solution than a human may be able to find in any reasonable amount of time.

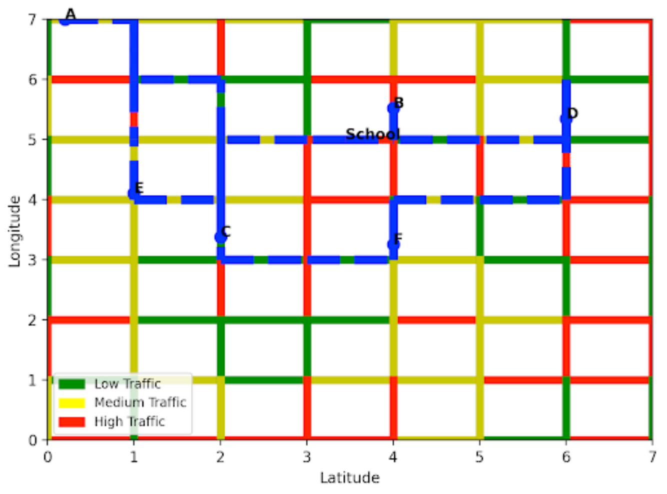

Fig 3. Example Optimal Route

3. Conclusion

In conclusion, a combination of machine learning algorithms are demonstrated for the use case of developing an optimal school bus route. The branch and bound algorithm creates the graph that the breadth first algorithm traverses through, finds the closest intersection from the point and finds the best route between two points. Also, it plots the best route on a graph so that it is easier for the user to see. Then, a genetic algorithm was demonstrated to find the best route between all of the stops and can be put together with the breadth first algorithm to find the best total route. With all of this put together, an algorithm has been developed that can traverse through points with any size grid with any amount of stops and can find the best route and plot it for user feedback.

Github Link to Project:
https://github.com/Soham-West/Bus-route-genetic-algorithm