A Population-Based Traffic Search

Brooks P. Saunders, Scott T. Giorlando
Email: (brooks_saunders, scott_giorlando)@student.uml.edu

***Abstract*—Most searches that are used for traffic systems are distance based searches, where they try to find the shortest or most optimal route to their goal. The only issue in real life is that while a route may be shorter, it could actually end up taking longer to reach a desired goal because of other factors that can appear in your travels. We came up with a different approach to this, a search based off population density and not distance. With population density we are trying to find a route that avoids higher populated zones as those are more likely to cause and non-optimal routing solution. To do this we implemented an A* search with a modified heuristic to accommodate population density versus distance. Based off our results we found that population density is a good measure against traffic and we also found that there are multiple different applications outside of just traffic that a search on population density could be used for.**

I. Introduction

A. Problem Description and Motivation

In today's society we rely on technology to aid in every facet of our life and transportation is no exception. A major aspect of a gps based routing system is the ability to provide pathing for the user based on consideration given to traffic and other variables that impact the time of the journey. Simply obtaining the shortest path from point A to point B would not be sufficient to getting the most optimal path with time considerations. The problem is, thus, to devise a search agent which takes traffic into consideration throughout the searching process of finding the path from locations A to B.

The characterizations of our problem is that it is fully observable since it always knows the attributes of its neighboring states and its previous states to calculate the correct path. Our problem is also a single agent because it only simulates one car at a time instead of multiple at once. The problem is also characterized as deterministic because it has constant and determined speed limits for each zone. Our problem is sequential because it requires a memory of past actions versus being a one shot action. Our problem is also considered to be static because the world state never changes to the agent acting on the world state. Our problem is characterized as discrete because the agent stops and deliberates at every intersection it runs into.

B. Background and Approach

Often in dense rural areas we find that traffic tends to get congested in specific spots, this causes choke points and choke points cause commutes to be minutes or hours longer than they should regularly be. This can cause many issues, one of which being an inconvenience for everyone in the area. One type of inconvenience is leaving work early, this causes you to lose some of the money you could be making normally. For example, if you are commuting to work and want to minimize the time it takes to get there, then it would be better to find something that can you give you an alternate route to avoid these densely populated areas. Once you avoid these areas you can now work at your regular hours instead of having to "beat traffic" and leave your house at earlier times to not risk being late.

Of the algorithms researched and seen implemented, A* best fit our circumstances. Most traffic searching algorithms using A* are focused on reaching a goal using the least amount of distance possible. Most of these types of searches apply a heuristic to find the most satisfactory option or route to be taken. Our approach was to not reinvent the wheel but to attempt a different and more interesting look on this problem. We decided to search by population density versus distance. In order to do so we had to modify our heuristic with a scale in order to receive complement the results seen in studies on population density.

II. Related Work/Literature Review

Two of the searching methodologies found through the literature of traffic routing search agents consists of a genetic algorithm implementation and an A* search algorithm implementation.

A. Review of Genetic Algorithm paper by Oh, Byonghwa

Genetic Algorithms are one of the few ways to have a dynamic traffic routing system that chooses the most optimal/shortest roads to be taken. One way to find the most optimal route is to have each vehicle on the road upload their traffic data to some sort of database and we have a Genetic Algorithm find and choose the most optimal route that way [4].

One issue that lies with this is that choosing the most optimal or shortest route may actually be longer in travel time due to other limitations such as lights or stop signs. Sometimes shorter distance does not mean shorter travel times. Another issue that lies with Genetic Algorithms is that using gene crossovers for the most optimal routes may not actually work since some roads on one path may not be connected to roads on the other path.

In this paper the authors suggest their idea of working around the normal Genetic Algorithm idea of only the strongest survive, where they instead run all the routes through a regular search algorithm like A* and find the shortest route that way. Once they find that route they use the Genetic Algorithm to dynamically modify the route by travel time versus distance. The routes are only modified after a certain

point, so it cannot dynamically modify the whole route since that would render the A* search as useless and it would just waste resources. The rerouting point that is found is based off how long the normal path usually is, and is usually a small area so it is manageable.

Another thing to note about Genetic Algorithms is that there are a few different ways to come to a similar conclusion. One could use the Dijkstra's Algorithm, which would guarantee us the shortest route but it would be very costly in terms of running time.

B. Review and Analysis of paper on A* by Huang, Ke

In the paper Real-time Transportation Route Selection With Traffic Considerations, the authors decided to use an A* searching algorithm [3]. The heuristic chosen for this paper is shown below, where the authors decided to base it on a time cost with consideration to velocity. This heuristic gets real time velocity information from a simulation of traffic packed roads.

The literature on this subject showed that commonly traffic information was gathered in a real-time manner using sophisticated simulations taking repeated measurements. In order to obtain a simpler model, this project will use a metric which is constant through time, but can still properly give a reading of traffic. The fixed metric to determine traffic was chosen to be based on the population of the area under consideration. This metric is justified by the considerable correlation that exists between population density and traffic. The Federal Highway Administration had found "that traffic volumes at typical densities tend to rise at least 80 percent of the rate of population density increase" [1].

III. Methodology

A. Simulated Worldspace

For the implementation of the described problem, A worldspace is created consisting of a gridded region, R, of equivalently distanced lines that are connected to vertices, v_i. Every v_i then conjoins four lines at an angle of 90 degrees from each other. Every line represents a traversable road, r_i, and therefore the set of all roads $S_r \equiv \{r_1, \ldots, r_i, \ldots, r_{k_r}\}$ represent all 'drivable' areas of R, where k_r denotes the total number of roads. A given vertex is defined as the vector containing an x location, y location and the four roads connecting to it, $v_i = [x_{v_i}, y_{v_i}, r^N_{v_i}, r^E_{v_i}, r^S_{v_i}, r^W_{v_i}]$.

1) Worldspace and Zoning Implementation: The worldspace is implemented such that a state is defined as a vertex v_i, and each road r_i represent an action with some cost, $r_i = [c_{r_i}]$. Thus, any given state, v_i, has four possible actions: $[r^N_{v_i}, r^E_{v_i}, r^S_{v_i}, r^W_{v_i}]$ with costs corresponding to each action.

The region R is then populated with an arbitrary amount of zones encompassed by the set $S_Z \equiv \{Z_1, \ldots, Z_i, \ldots, Z_{k_Z}\}$, where each zone, Z_i, is a vector defined by an x and y location, a width, and a population density (in people/sq.meter), $Z_i = [x_{z_i}, y_{z_i}, w_{z_i}, d_{z_i}]$.The x location, y location and width of any given zone, $x_{z_i}, y_{z_i}, w_{z_i}$ respectively, are randomized with the only condition being that no zones can overlap. The population density of a given zone, d_{z_i} is a random value between $1000-4000$ people/square meter.

2) Cost Function: The cost function for each road is a function of the population density of the zone the road is within, $c_{r_i} = f(d_{z_i})$, where d_{z_i} is the zone the road is inside. $f(d_{z_i})$ was implemented as the hours/sq.mile it takes to travel the given road, derived from Table I.

A line of best fit through the data of Table I was calculated to be:

$$f(d_{z_i}) = \frac{d_{z_i}}{4.2{*}10^{-10}(-d_{z_i}{+}3000)^3{+}2.8}$$

The graph of this function is seen in Fig. 1.

TABLE I
Population Density and Related Traffic Levels of [2]

Population Density	Vehicle Hours/Sq.mile
$>$ 4000	1,675
3,000 - 3,999	1,237
2,000 - 2,999	911
$<$ 2000	609

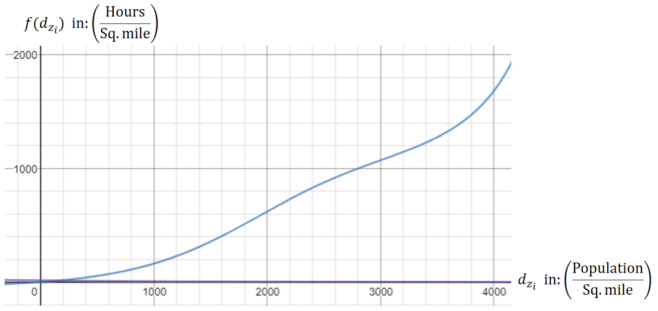

Fig. 1. Cost function graph

B. Search Agent

1) Search Strategy: The search strategy we decided to follow is one that forms a tree with nodes equal to states, v_i, and connections between nodes signifying actions, r_i. The root of the tree represents the initial state and all downward connections from there on represent the possible paths one could take. There are three distinct ways to generate and traverse such a tree: a depth-first traversal, a breath-first traversal, and an A* traversal. Each of these searching methods are implemented using a frontier, which is a queue or stack which holds nodes as they are expanded. The significance of each traversal method is in how each specifically handles choosing a new path. For our implementation, we shall go with the A* traversal method which follows the same idea as the depth first traversal except that it chooses the next node to expand based on which in the frontier has the least cost.

Our hypothesis is therefore defined as the agent successfully choosing the most cost efficient path, avoiding the high density zones.

2) Internal Structure Implementation: Our search strategy was implemented via an A* search with a class we called $driverState$ which includes a state that has two coordinates x and y and its cost from the start. Inside this class we have a function named $generateValidSuccessors$ that finds all possible states in every direction and the cost of these states that can come from the current position that the program is in. Then we have a function called lookup that opens our input file of the zones and paths that are generated and splits each zone on to a new line to check for the location, this does this for every direction possible and returns a cost. Then we have a search class that has a start location and an end location. The search class has a function $isGoal$ that checks to see if the current state is in the goal state. Then we have the function $aStarSearch$ which uses a priority queue and stores the path of each state we are going to be using in our path based off the heuristic given in $generateValidSuccessors$.

Algorithm 1 Class $driverState$

1: $x \leftarrow$ x position of *driverState*
2: $y \leftarrow$ y position of *driverState*
3: *CostFromStart* $\leftarrow$ Cost from start to *driverState*
4: **procedure** $init(xPos, yPos, CostFromStart)$
5: **procedure** $generateValidSucessors$
6: **procedure** $lookup(xPos, yPos)$

Algorithm 2 Class $search$

1: *StartDriverState* $\leftarrow$ *driverState* defining start of path
2: *EndDriverState* $\leftarrow$ *driverState* defining end of path
3: **procedure** $init(StartDriverState, EndDriverState)$
4: **procedure** $isGoal(driverState)$
5: **procedure** $aStarSearch$

C. Results and Analysis

In Fig. 2 and Fig. 3, it can be seen that the agent successfully traverses the region from the start point of (40, 3) to reach the end point (37, 45). It can be seen that the path chosen avoids several zones but also travels through various zones of the minimum cost.

Table II represents the different densities our our zones illustrated by different colors. The implementation of our densities and their according zone color can be seen in Fig. 2, Fig. 3, Fig. 4 and Fig. 5.

IV. Discussion

From the results illustrated in figures 3, and 5, it can be seen that the search agent had properly traversed the region from a start point to an end point. The hypothesis made earlier in the paper is that the agent should choose the most cost effective path. This was expected to be the case as the A* heuristic should give it sufficient information to expand the most optimal nodes. The aforementioned figures indicate that this is the case as the algorithm successfully chooses the most optimal path and avoids the most costly zones.

TABLE II
Zoning Color Representation

Zone color	Density of color
Black	< 300
Magenta	300 - 600
Green	600 - 900
Blue	900 - 1200
Red	> 1200

The fact that the heuristic was based on data acquired from legitimate studies in the correlation of population density to traffic rates, the results indicate a realistic simulation of traffic routing. Figures 2, 3, 4 and 5 represent a randomized simulation of zones of arbitrary distance units. This means that the search agent may be applied to population density data of any sized location (ex. North America, Massachusetts, Lowell).

V. Conclusion

After finishing our implementation and testing the search algorithm we used, we found that this problem of using a search based on population density and distance instead of searching only based off distance can have a multiple amount of uses outside of just traffic. Since this focuses on population density in zones we found that there a few different applications that are correlated with density in populated districts. For example, crime is a big factor in highly dense and rural areas. This algorithm we designed could be used to distinguish zones that are more likely to have crime related incidents and this could help people to avoid those areas. Another example is fire hazards as that is also correlated to population density, this could help decide which areas are more likely better to avoid in terms of safety.

References

[1] Agafonov, Anton, and Vladislav Myasnikov, *Efficiency comparison of the routing algorithms used in centralized traffic management systems*, Procedia engineering 201 (2017): 265-270

[2] Cox. Wendell, *How Higher Density Makes Traffic Worse*, May 2003, www.publicpurpose.com/pp57-density.html.

[3] Huang, Ke, *Real-Time Transportation Route Selection With Trafc Considerations*, pdfs.semanticscholar.org/

[4] Oh, Byonghwa, *Genetic Algorithm-Based Dynamic Vehicle Route Search Using Car-to-Car Communication*, Pdfs.semanticscholar.org/, 4 Nov. 2010,pdfs.semanticscholar.org/47d4/61f3ff0afeab855da2c0d31d5ec2275e8dd6.pdf.

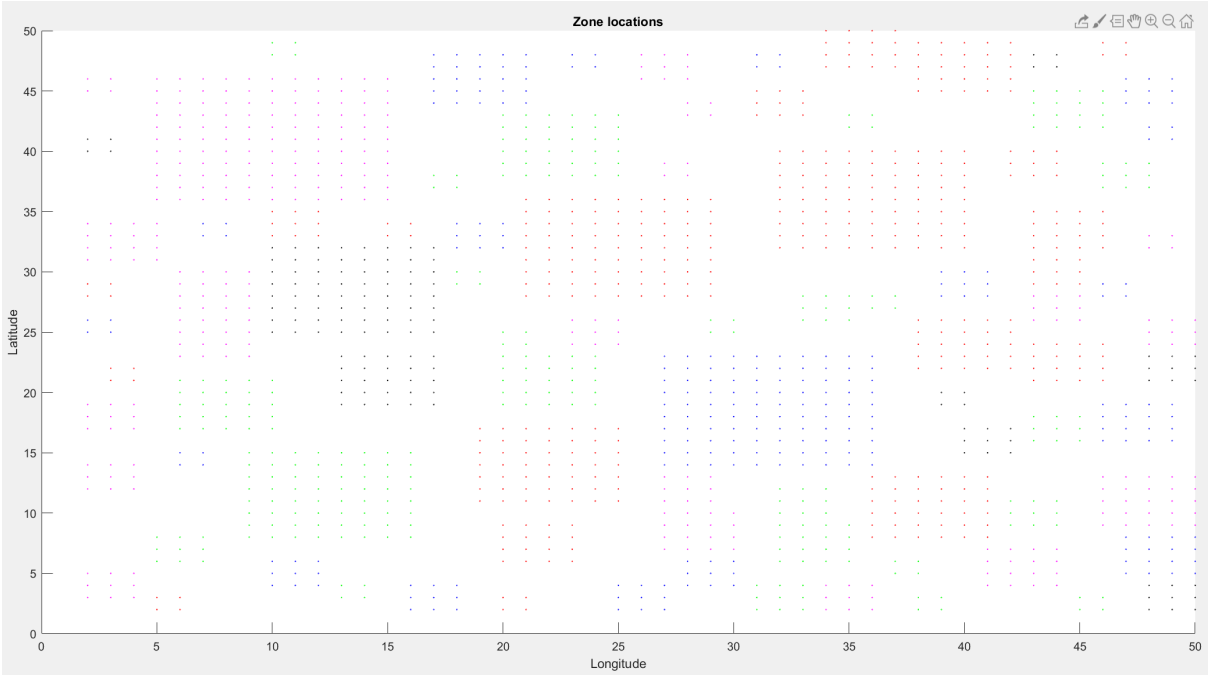

Fig. 2. Sample #1 of Zone without path

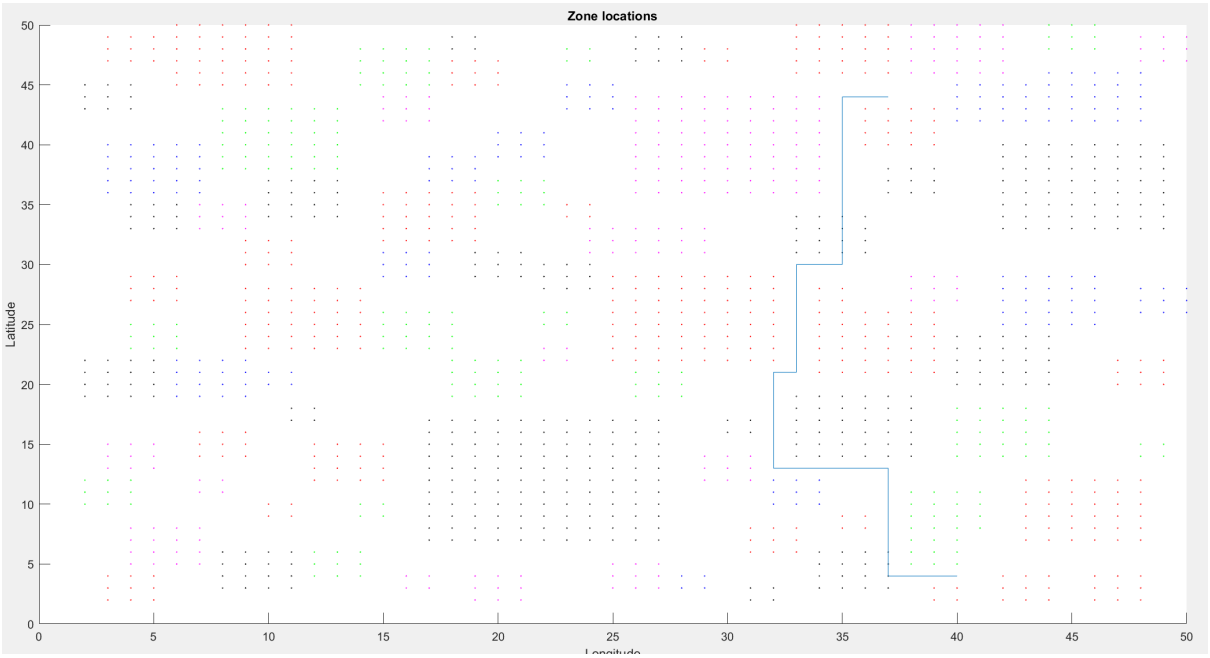

Fig. 3. Sample #1 of Zone with path

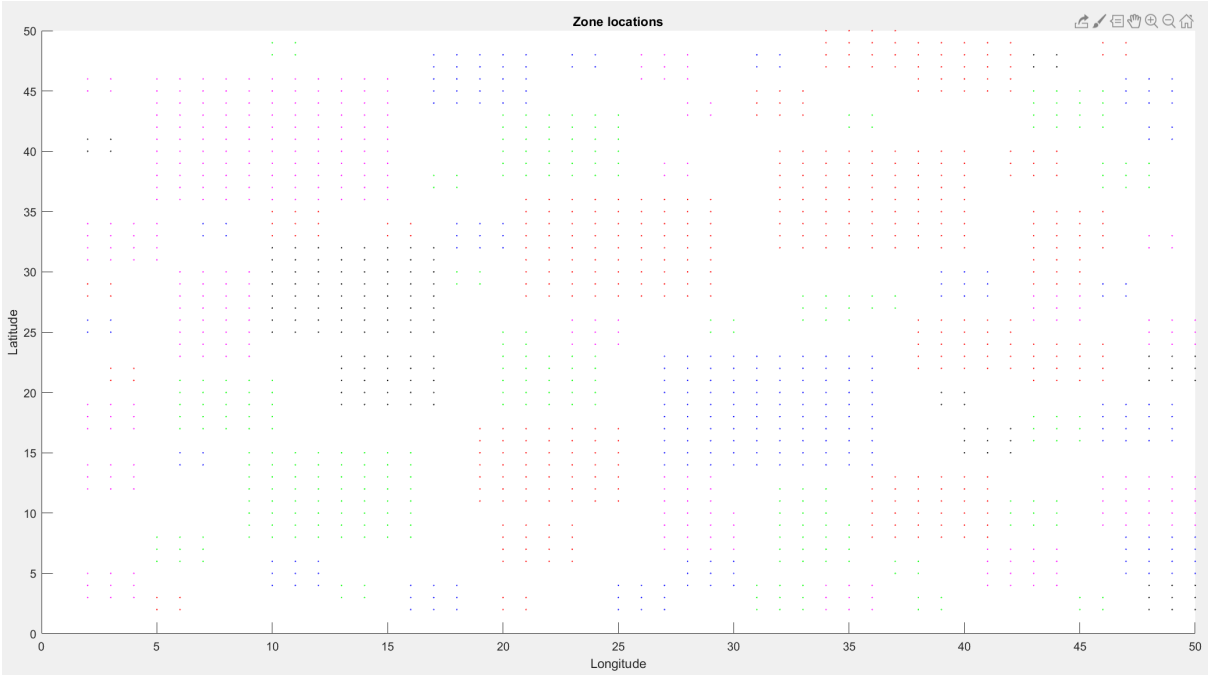

Fig. 4. Sample #2 of Zone without path

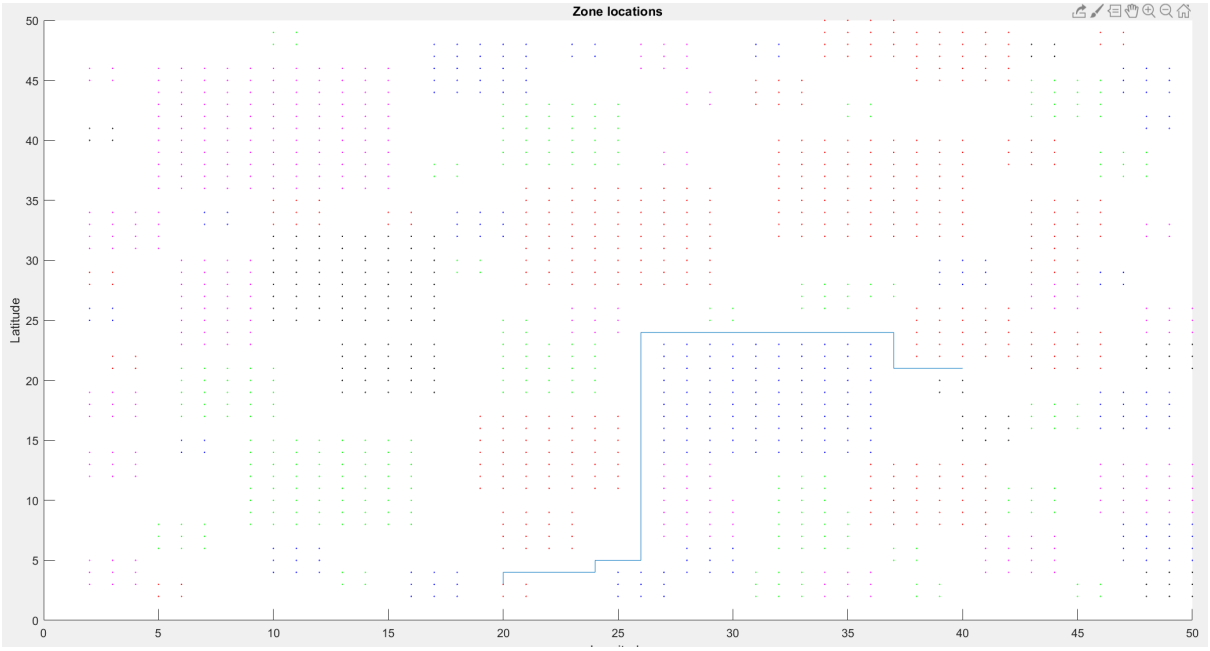

Fig. 5. Sample #2 of Zone with path