Traffic Simulation

Team 65

New Mexico

Supercomputing Challenge

Final Report

April 2, 2014

Team Members

Nate Delgado

Phillip Heikoop

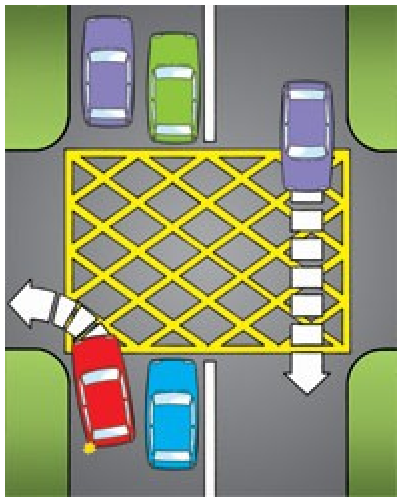
Cade Mallett

Teacher

Lee Goodwin

Executive Summary

This project in its current form is a functioning traffic model with agent cars and traffic control measures. The purpose of the project is to determine the best forms of traffic control at intersections. This goal was accomplished by using a predetermined map based on real intersections in "commuter-friendly" cities in America. The intersection stops were rotated to create all the possible combinations of intersections that can be sorted in the map. Then the maps were compared to see which one most effectively and quickly moved cars from place to place. Total speed and efficiency were the key determinants, i.e. the map that most quickly disperses people and the one takes the least time to cover the most distance. The combination of traffic stops that is found to be most effective can be analysed to show both the circumstances that allow an intersection to be at its most effective. This is accomplished by looking at the randomized variables surrounding each intersection and finding the ones that every instance of each control measure has in common. Obviously, the project can also be used to create the most efficient traffic system for a hard-coded map as well.


http://bright-indonesia.blogspot.com/2012/12/apa-maksud-kotak-kuning-di-perempatan.html

Problem

We set out to make a computer model that would improve roads by demonstrating slightly niche but still functional instances of traffic control measures. Some examples of the rarer intersections are the box and Michigan left. While the former is prevalent in Europe and the latter occasionally found in the Midwest, neither has experienced mainstream implementation in the bulk of America. However, both very clearly offer benefits not found in other types of intersections, and are the best option in some use cases. So, we set out to find those cases.

The problem we want to solve is any inefficiency found in roadways because of a lack of adoption of new intersection designs. Whether or not implementation of such new techniques and others improves traffic flow and management could be considered the question we answered.

Methodology

To solve the aforementioned problem, we created an agent-based model of a road system in the Java programming language. A hard-coded map helped us to maintain the scientific method as the model created every possible combination of traffic control features at each intersection. Randomly generated car agents were also a control variable. Each was given a random start and finish, and the car would travel along the road network. The cars were then given a new A.I. (artificial intelligence) every time they entered an intersection. The A.I. was specific to each intersection and gave cars instructions on factors like when they could enter, how fast they could go, and when to yield. Every car recorded its travel time and distance. These values were totalled to find the efficiency of the map being tested. Every possible combination of intersections was tested, and the least time came from the best model.

http://indiadrivesafe.com/articles/yellow-box-junctions.html

Validation

To validate the model, we first had to show that it supports the travel of cars. This was our first step after the debugging of the model. We tested that the cars can travel along the road and reach their destination within a normal number of iterations. With these we found that the road supports cars moving from place to place and that they could keep a constant start and end point. The model was also validated to function using the scientific method. We ensured that the number of cars, their destinations and the map's features remained constant through each series of trials. This was accomplished simply by testing that through a series of repeated trials the expected conditions remained the same. Each of the above variables was printed to the screen. Thus, we showed that the model can maintain one or more of the variables through the trials or change them in a regulated fashion. The final proof of functionality came through implementing a simple A.I. to treat each intersection as a four-way stop. The cars were again placed on the road and their behavior was monitored. This time the results of the trial was slightly offset from the previous, stop-free tests. Random cars were chosen that would never meet at an intersection and kept the same for each test. The number of turns each car took to get to its destination was increased by an expected amount that could be calculated by the number of stops it now had to make. We saw similar and often greater increases when we allowed some of the cars to meet at the junctions. New delays were caused by the waiting of cars when other vehicles were already in the intersection. The expected behaviors and results were found from each of these tests and many similar ones we used to further verify the model.

Code Examples

The base map was stored as a two dimensional array of integers, roads marked by ones, intersections by negative ones, and empty space by zeros. For each trial this map was parsed into a two dimensional array of road objects which the actual model used.

```java
	public static Road[][] parseIntoRoads(int[][] map){
		Road[][] newmap = new Road[mapsize][mapsize];
		for (int i = 0; i < mapsize; i++){
			for (int k = 0; k < mapsize; k++){
				if (map[i][k] == 0){
					newmap[i][k] = noRoad;
				}
				else if (map[i][k] == 1){
					newmap[i][k] = normalRoad;
				}
				else if (map[i][k] == 2){
					newmap = makeRoundABout(i, k, newmap);
				}
				else if (map[i][k] == 3){
					newmap[i][k] = boxJunction;
				}

			}
		}
		return newmap;
	}
```

To ensure that the start and destination for each car were actually reachable locations, we made a simple function to read through the base map and add coordinate pairs to a master list of valid locations. Coordinate pairs were represented as simple length two arrays in form:

[x , y]

When spawning a car, its start and destination were simply randomly selected from the list of coordinate pairs.

```java
	public static ArrayList<int[]> validCoordinatePairs = generateNewPairs();

	public static ArrayList<int[]> generateNewPairs(){
```

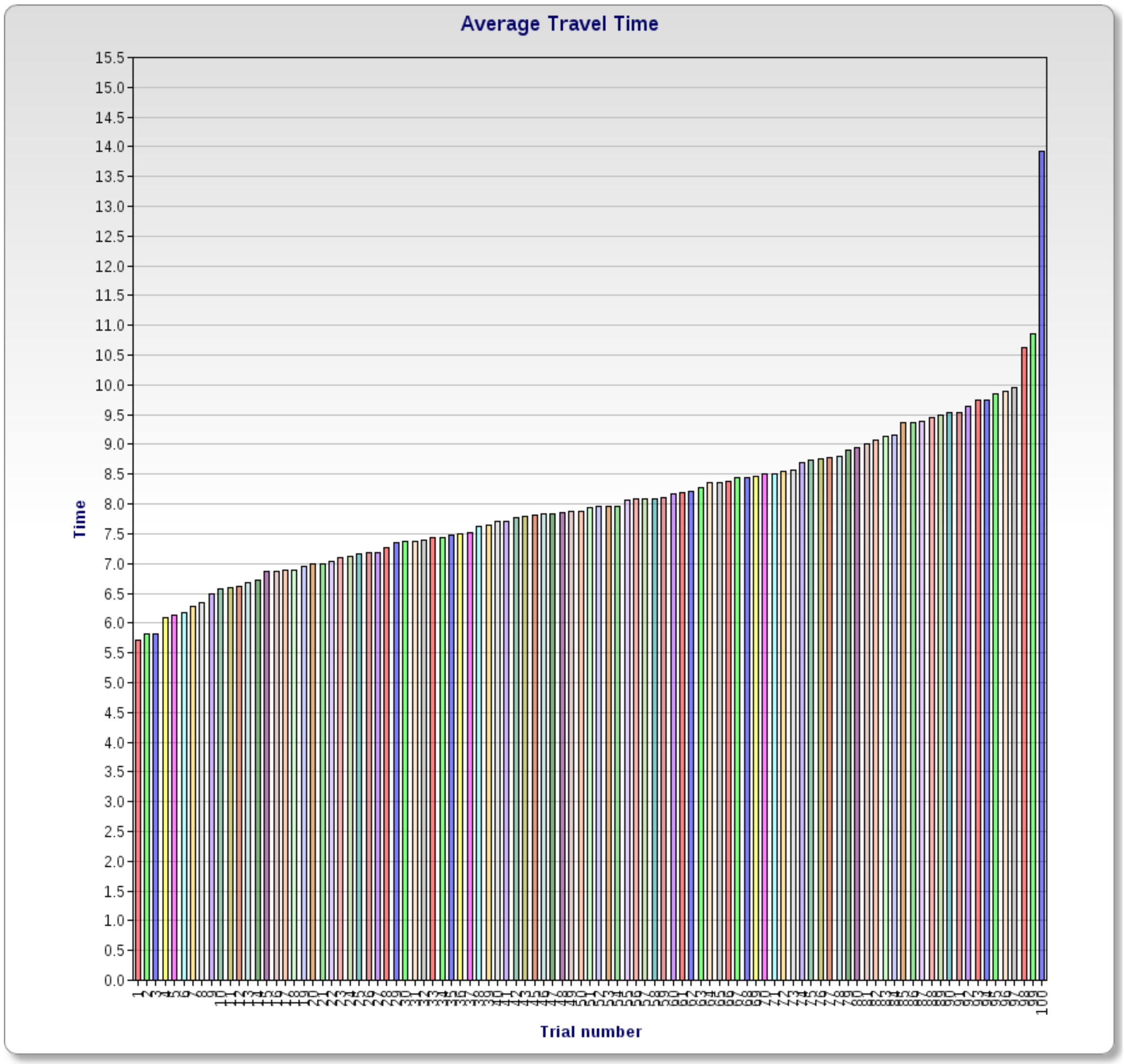
```
                ArrayList<int[]> pairs = new ArrayList<int[]>();
                for (int i = 0; i < Main.mapsize; i++){
                        for (int k = 0; k < Main.mapsize ; k++){
                                if (normalmap[i][k] == 1){
                                        pairs.add(new int[] {i, k});
                                }
                        }
                }
                return pairs;
        }
```

Significant Achievement

The most significant achievement in this project is the ability to easily add new traffic features and have them mesh into the optimization process. This provides great opportunities for continuing the project because the model offers an established base where we can create and simulate new traffic rules and features without having to worry about making the main code support them. That is to say, simply drafting a new A.I. and assigning it to cars at certain points on the map, we can expound upon the model and allow it to reflect different conditions. We have created the base for any project that reflects our goals. However, this starting point is superior because it makes expansion, modification, and implementation of additional features more accessible. All of the above benefits are possible without any alteration to the project’s existing code.

Results

The results of this project are not very conclusive because the artificial intelligence for the cars is flawed, mostly in the sense that some cars are simply unable to reach their destination. Improving and refining the AI would be the goal of a possible continuation of the project. Despite this, we were able to perform optimization on the road network. Our primary test was running 80 cars over 100 random map iterations. For each iteration we recorded the average of each car's travel time, obviously the most efficient road network is the one that takes the least time for people to reach their destination on average. As visible in the graph, there was a significant decrease in travel time for certain roadway configurations.



Conclusion

In the course of this project, we attempted to construct a model that could be used to optimize a traffic system by reducing average travel time by substituting different traffic features in place of normal four-way intersections, namely box junctions and roundabouts. The construction of the car AI was found to be much more difficult and time consuming than expected, which provided significant hindrance to the project, and the refinement of said AI is the main focus of a possible project continuation.

References

"How to Do Box Junctions." *Box Junctions in the UK*. N.p., n.d. Web. Nov.-Dec. 2013.

"Intersection (road)." *Wikipedia*. Wikimedia Foundation, n.d. Web. Nov.-Dec. 2013.

"The Michigan Left." *Michigan Highways: In Depth:*. N.p., n.d. Web. Nov.-Dec. 2013.

Using the Road (159 to 203). N.p., n.d. Web. Nov.-Dec. 2013.