A Straightforward Solution to a Roundabout Problem

Stephen Demjanenko, Jesse Livezey, Joshua Edgerton

February 28, 2009

Abstract

The first generation of traffic circles was prone to traffic jams and were poorly optimized. Over time, traffic circles became more effective at controlling traffic. Continuing in this tradition of improvement, we analyze what types of traffic control systems are capable of keeping traffic moving at its most efficient rate. Our model is scalable from small roundabouts to large urban traffic circles. Since all traffic circles are unique, it is important to have a general and customizable model so that we can treat each case separately.

Traffic circles are conservative; cars are neither created nor destroyed. As a result, we chose to model the traffic circle by solving the traffic flow partial differential equation using the Lax-Friedrich finite difference method. This gives a macroscopic picture of traffic flow in and around the circle. We chose to simulate stop signs, yield signs and several variants of traffic lights with which a traffic engineer can control the flow of cars into and around the traffic circle.

In order to optimize the traffic control system we define our metric to be the number of cars that pass through the circle per unit time. We then ran several basic cases in order to compare the systems of traffic control. Using this information, several methods of traffic control were formulated based on combining the best systems of control, as determined from our initial tests. These methods then ran on several more complex geometries in order to determine which method was the best for each geometry. Generalizing the results of these test cases, we formulate basic guidelines which a civil engineer can use to determine an appropriate method of traffic control depending on the properties of the traffic circle. We find that in general, roads which carry a low volume of cars should yield while those carrying a high volume of cars should be flow limited.
1 Introduction

“Columbus Circle is like a black hole, cars go in, cars go out, but you never know what’s going on inside.”

-Ms. Ethel Scheffer, chairwoman: Columbus Circle task force; 1987

Traffic circles in the United States date back to as early as 1905 when Columbus Circle was first built in New York City [4]. It has undergone countless renovations and improvements over the century in an effort to reduce the ‘chaos’ within. The initial Columbus Traffic Circle design proved inefficient with its use of yield signs on traffic in the circle which created high congestion and caused many accidents. As with everything in this world, inefficiencies such as these are never left alone for too long. Modifications were made.

Most existing traffic models simulate highway systems. Unlike highways where traffic moves long distances between decisions, traffic circles can be highly chaotic with different cars needing to take different paths through the circle in order to reach their destinations.

Our goal is to find a means of optimizing the traffic controls in a traffic circle. More specifically, we want to develop methods to regulate incoming, circling, and outgoing traffic to maximize efficiency for a variety of different traffic circles. Our work will culminate in a technical summary detailing our the best methods of control for generalized traffic circles.

When approaching the problem, we needed to look at all of the ways that inefficiencies could arise in a traffic circle. There is a wealth of mathematical models describing traffic flow in response to three key problem areas [1]:

- Alleviating congestion
- Maximizing the flow of traffic
- Eliminating accidents

We decided to model the first two bullets. We found that the most common traffic controls employed are stop signs, yield signs, and traffic lights. As a result, we explicitly model these systems in an effort to find the most efficient method for controlling cars in a traffic circle.

1.1 Our Understanding of the Problem

We were tasked with the challenge of choosing the optimal traffic control strategies for a variety of traffic circle layouts. There are several parameters that we want to explore which most certainly affect traffic flow. They fall into three categories:

- **Circle geometry:** The number of lanes in the circle and the number of roads, as well as their sizes, that lead into and out of the circle. We also simulate a separation between the paired entrances and exits on the circle for each particular road.
Traffic patterns: We want to see how adaptive our system is to different amounts of traffic density which might be caused by the time of day, the current density in the circle, or congestion caused by many cars attempting to leave at the same exit.

Traffic control systems: There are many, but we focus on yield signs, stop signs, and traffic lights.

Generally, models are categorized into one of two categories, microscopic or macroscopic traffic models. The former attempts to model traffic by individual cars and how each behaves as a member of the whole, typically using ordinary differential equations, while the latter evaluates traffic flow as a whole without consideration of the characteristics and features of individual vehicles in the traffic stream. We chose to treat the problem in the macroscopic sense since we are primarily concerned with the average traffic conditions in and around the circle. This has a few ramifications which we will list in the limitations section.

2 The Model

As many of us have experienced on the road, cars (under normal operating conditions) cannot be created or destroyed. This means that the density time rate of change is related to the change in flow over a distance. This results in a continuity equation:

\[
\frac{\partial \rho}{\partial t} + \frac{\partial f(\rho)}{\partial x} = 0
\]

where \(\rho\) is the density of traffic at a certain point in space and time and \(f(\rho)\) is the flow of traffic at a certain point in space and time. This is a very general equation which alone cannot tell us anything about the movement of traffic in a traffic circle. Thus additional assumptions are needed to turn this into a useful equation.

The empirically measured function \(f(\rho)\), which is flux, has been well studied, particularly for traffic flow. There is some density of traffic called the jam density at which the speed of traffic goes to zero. As the density increases from zero so does the flow. This makes intuitive sense because if the cars are freely flowing, then the flow should be proportional to the density. At some density the flow reaches its maximum value, then congestion sets in and jams start to form.

One model for the flow is the Greenshield's Flow model [1]:

\[
f(\rho) = u_{\text{max}} \times \rho \times (1 - \frac{\rho}{\rho_{\text{max}}})
\]

This equation has all of the features that we want in an equation for flow:

- It starts at zero and has a maximum flow rate of \(\frac{\rho_{\text{max}} u_{\text{max}}}{4}\).
- It also goes to zero as \(\rho\) goes to \(\rho_{\text{max}}\).

2.1 Definitions

Greedy vs. Non-Greedy: A stop or yield sign is greedy if the traffic entering the circle has the right of way. Non-greedy is when the traffic in the circle has the right of way.
• **Traffic Flow** $f(\rho)$: Average number of cars passing per unit time.

• **Density of Traffic** $(\rho)$: Number of cars per unit distance.

• **Macroscopic Traffic Model**: Model consisting of equations for a few aggregate quantities like the density of traffic $\rho$, the average velocity $u_{avg}$ and traffic flow $f(\rho)$. Our model falls into this category.

• **Velocity** ($u$): The distance per unit time that a car travels.

### 2.2 Assumptions

There are a number of assumptions that we used in creating our model. There are also assumptions that need to be taken into account when interpreting the results from our model.

- The number of cars are conserved. $cars_{in} = cars_{out}$. Since we use a continuous model we check to make sure that the flux in equals the flux out plus the amount of traffic still in the circle.

- Our model could have incoming densities that depend on time or density in the circle.

- The interactions of cars at the microscopic level are not important in the larger, long term traffic patterns. The validity of this is questionable since there have been multiple studies of “butterfly effects” on highways where one driver’s decisions can affect several hundred miles of roadway. [5]

- The exit roads do not clog.

- Traffic through the circle is a continuous stream. There are no accidents, obstructions, stops, pedestrians, etc.

- All the cars travel in the same direction.

- All the cars at a particular location in the circle are traveling at the same speed.

- We do not model lanes.

- There is always one entrance paired with one exit although there can be a finite distance between them.

- We normalize $u_{max}$ and $\rho_{max}$ to 1. This allows easy comparisons but is not useful for real problems.

### 2.3 Foundation and Development

Looking at measures of traffic density, flow, and velocity, we treat the system as a continuously operating system fluctuating based on the time of day or traffic signals. The foundation of our model is an algorithm to solve the continuity equation. We use a variant on the Lax-Friedrich’s method to propagate the density of traffic around the ring. Our model simulates some time-dependent (and possibly circle density dependent) distribution of cars down roads and into the circle. These cars then go
around the circle and get off at their exit if possible. Our model allows for everyone
getting off at their exit or some fraction getting off and the rest taking another loop
around. When analyzing what happens at the points where traffic can enter or exit
the loop, we use routines based on which type of traffic control is present at the
intersection to distribute the cars.

2.4 Lax-Friedrich Implementation

We numerically solve the traffic flow PDE, the continuity equation, using the Lax-
Friedrich finite difference method. Given some distribution of cars along the circle,
the traffic flow PDE steps the distribution of cars forward. However, keeping track
of which exit the cars are going to is crucial for a working model of the traffic circle.
In order to get around this problem, we create separate circles (called exit circled)
for each exit. This allows us to keep track of which exit the cars from different
entrances are going to. We then sum up all of these exit circles to create the net
circle distribution.

Our Lax-Friedrich program runs on this net circle distribution to get the new
distribution at the next timestep. We then extract the velocities at each location on
the net circle distribution from before the run and after the run and average them
to get \( u_{avg} \). These velocities are then used to move the corresponding densities in
each exit circle forward \( \Delta x = u_{avg} \times \Delta t \) where \( \Delta t \) is the time-step. We then sum the
exit circles together and check to make sure that \( \rho_{max} \) is not exceeded anywhere. If
it is then we simply spread the extra density backwards, filling up to \( \rho_{max} \). While
doing this we keep track of the proportions of each exit circle and properly move their
distributions since all cars are treated the same in the net circle.

2.4.1 Stability

In order to guarantee stability of the results one needs to ensure that \( \lambda = \frac{\Delta t}{\Delta x} \leq \frac{1}{|u_{max}|} \).
Moreover, it might be necessary to increase the spatial and temporal resolutions since
we move cars in the exit circles forward by \( \Delta x = u_{avg} \times \Delta t \) as mentioned above. As
a result, cars could end up passing by their exit without being removed. Decreasing
\( \Delta t \) will fix this problem but result in a longer runtime.

2.5 The Metric

Many ideas come to mind about what one desires when traveling on roadways. Es-
sentially, one wants to get from point A to point B as quickly as possible. This last
statement captures what we desire to optimize: the average time a person spends
traveling. In this problem we only look at cars moving through the traffic circle. Fur-
thermore, modeling the average time in a traffic circle does not indicate how many
cars go through the traffic circle. Thus we suggest that the measure to be optimized
is the number of cars per unit time that will make it through the circle. This is flux.

2.6 Optimization

Due to the sheer number of variables and potential traffic situations it is not reason-
able to find a global optimum. Instead, we treated this optimization problem much
more qualitatively trying a few elements at a time to find an generalization of what
the best setting for each property was. We then use these generalized properties to present reasonably well optimized traffic control methods for both of the small and large traffic circles.

2.7 Circle Geometry

We consider two types of traffic circles: small (3-5 roads) and large (5-8 roads). Each of these circles can support some maximum density ($\rho_{\text{max}}$) of cars going through them per unit time. Instead of explicitly solving for $\rho_{\text{max}}$, we define $\rho_{\text{max}} = 1$ and then scale the roads to input a flux based their size. The model allows for roads to be placed anywhere around the circle.

2.8 Traffic Patterns

We expect traffic to be periodic depending on the time of day and the day of the week. Our model takes the input roads mentioned above and sends cars down them. Just before they reach the circle, they hit some sort of traffic signal (or maybe none). This could potentially cause backups on the input roads.

Once the cars enter the circle, they will try to get out at their exit at the first opportunity that they get. In the model we consider resistance between two groups of cars as they go by the exit: (1) all the cars and (2) the subset trying to get out at that particular exit. As the density of (1) approaches $\rho_{\text{max}}$, we would expect the resistance to increase. Similarly, as $\rho_{(2)}$ increases, we expect the cars that want to get out will compete with each other more and some will be blocked from exiting. We also consider thresholds for $\rho_{(1)}$ and $\rho_{(2)}$ for which cars will always be able to exit. The proportion that does not get out is forced to do a full circle around the loop before trying again. This can cause further traffic problems for other exits.

2.9 Traffic Control Systems

There are many potential methods of traffic control ranging from stop signs to coordinated traffic lights. Since we are considering traffic flow control around a traffic circle, our options are quite limited. We modeled yield signs, stop signs, and traffic lights which obey varying rules for determining their color.

We model an incoming distribution of cars coming down the roads with the circle end having the value calculated by the corresponding traffic control system at that intersection. The Lax-Friedrich method is then used to model how cars will fill up the road while waiting to enter the circle.

2.9.1 Yield Signs

A Yield/Give Way sign is used to indicate to vehicles that they must slow down and be ready to stop if required, however if traffic conditions allow, a stop is sometimes not required. There are the greedy and non-greedy versions of the yield sign.

2.9.2 Stop Signs

A stop sign requires a full stop; under no conditions may someone go through without stopping. This effectively bottlenecks an intersection by limiting traffic to some
maximum flow rate. In order to model this we simply have some flow limit per unit time.

\[
flow rate = \begin{cases} 
    f_{in} & f_{in} < f_{limit} \\
    f_{limit} & f_{in} \geq f_{limit}
\end{cases}
\]

We then feed this flow rate into the yield program to create greedy and non-greedy stop signs.

### 2.9.3 Traffic Lights
Traffic lights act like stop signs when they are red and in the case of a traffic circle, they act like a yield sign when green (or yellow or whichever color is appropriate in the case of a traffic circle). That is, in our model traffic lights do not stop traffic from moving in the circle, it only stops some roads from sending cars into the circle for short periods of time.

We modeled several traffic lights. The most basic was a strict traffic light which changed colors in regular time intervals. We also had 'smart' lights which changed colors based on the density of cars on each path to optimize the filling of the circle.

### 2.9.4 Smart Traffic Lights
These traffic lights are special in that they can look at the traffic density locally. One of the simplest models is that each traffic circle has an optimum speed. In the case of our metric the measure is flux through the circle per unit time. Since \( f(\rho) = \rho \times u(\rho) \) there may be some optimum density which maximizes this flux. Thus, the traffic lights should attempt to attain this optimum density throughout the circle. This can be attempted locally by each individual traffic light. In the real world these smart lights need some sort of sensing system which could be sensors in the road bed or cameras with vehicle recognition technology.

## 3 Results and Analysis
Using the model detailed above we analyze the setup of the circle, the traffic control methods and incoming traffic levels. We began with basic setups where all of the inputs are the same and try to quantify the differences between the different control methods. It should be noted that the results we obtain may not be entirely numerically accurate since we compromised on the resolution of both the circle-steps and time-steps due to the limited processing power available over a period of 96 hours. With our preferred settings a run would have taken around an hour on a 3.8 Ghz Core 2 Duo (stable overclocked Intel ES500).

We did not model time-dependent input cases due to the amount of time that this task would have required. Instead we kept the input values fixed for each input road. However, we feel comfortable that our recommendations will hold since we stress tested them by simulating very high incoming density, \( \rho = .95 \), on the inputs. This may be excessive for the average traffic during non-peak hours but for the peak hours it should be able to handle the traffic reasonably well.
3.1 Data

Using our model and comparing specific methods of control we feel comfortable that we have found reasonable methods of control for the traffic circle. We start with a few basic setups to gauge the efficiency of our different traffic control elements. We then create several methods of control based on methods currently used in traffic circles as well as elements which our basic setup tests indicated work well.

3.1.1 Sanity Checks on Basic Setups

There are a few cases where intuition can give a reasonable estimate of what the best solution should be for a specific geometry. For instance, if there is a very small ring with only three or four entrances and exits and a very small amount of traffic then you would naively expect that the control system would have little effect on the efficiency of the circle. If there are low amounts of traffic you should almost never have to stop and traffic should flow smoothly.

The baseline example is six entrances and exits with uniform time independent inputs from each entrance. Each intersection has a traffic light set to locally achieve the optimal traffic velocity in the circle.

![Figure 1: Baseline Example.](image)

Zero density is at the radius of 1. Since we have a $\rho_{\text{max}}$ of 1 the density reads between 1 and 2. Traffic moves counter clockwise. The six pairs of peaks and troughs are the results of traffic getting jammed up and cleared at the entrances and exits.
The steady-state traffic/sec is .239.

We first tried changing the geometry of the circle without changing any of the control features. An example is six entrances and exits all on one side of the circle.

![Figure 2: Closely Spaced Roads.](image)

The average density has decreased from .395 to .363 and the average velocity has increased from .605 to .637. This is reasonable because the traffic on the left side of the ring can move faster but is carrying less density. The traffic/sec for this case is .231 which is slightly worse than the equally spaced circle.

The final change in geometry we tried was changing the filling levels of each input.
We can see that different inputs are filling the circle to different levels near the entrances but then evening out as the traffic mixes in the circle. The traffic/sec for this case is $0.236$.

Starting with an empty circle you can watch the circle fill and approach the optimal traffic/sec value of $\frac{1}{4}$. Here is a graph of traffic/sec plotted against timestep (whose units have no relevance). For steady state conditions our model obtains a very stable value.
We also tried choosing a small and large circle geometry and changing the traffic control methods. The small circle has three incoming and outgoing roads and the larger circle has six. We ran our model under the different traffic control systems and two different rates of traffic input. Our inputs are not time dependent so for all cases except the strict traffic lights our model settled down into a steady state solution. See Tables 1 and 2 in Appendix A for the results from the basic setup.

As mentioned above, in order to optimize the circle we want to maximize the flux per time for the circle. For the $f(p)$ equation from Greenshield’s Flow model this occurs when $\rho = \frac{\rho_{max}}{2} = \frac{1}{2}$ for our specific model. Thus a particular flow control method is optimized if the steady-state traffic/sec is closest to $\frac{1}{2}$.

For both circles the traffic light which attempts to fill the circle up to $\rho = \frac{\rho_{max}}{2}$ is optimal. It seems to work better for heavier traffic. It is also interesting that the strict traffic light is horrible for low flux roads but works well for high flux roads. It can also be seen than the non-greedy yield/stop is better than the greedy yield/stop. This confirms the observation that the circle traffic should not have to yield for the incoming traffic.

**Small Circle Analysis** Looking at the small circle data we see that for low flux inputs, the sign does not matter. For high flux inputs we see that the non-greedy stop works best. For both densities the $u_{optimized}$ traffic light is best.

**Large Circle Analysis** For low flux inputs, the non-greedy yield and stop are equal and again the $u_{optimized}$ traffic light is best. For the high flux inputs, the non-greedy yield is best while the $u_{optimized}$ traffic light is again the best.

**Results from Simple Runs** We noticed that for small input densities the average steady-state density in the circle was proportional to the number of the roads. We would expect this to continue until the circle start to saturate at $\frac{\rho_{max}}{2}$.

In circles of all sizes if the average traffic density is much less than the jam density then the control system makes little difference in the efficiency of the circle. It would make sense in this case to use yield signs on all roads. This will be the least expensive and easiest solution.
Effect of Separation of Entry and Exit  We ran several cases where we separated the entry and exit points for an individual road. We saw no significant difference between the cases and so we concluded that this parameter is not very important.

Time Dependent Entrance Density  Our model can incorporate a time dependent input density. If you were modeling a real traffic circle you would need to study the traffic patterns in the area. This parameter space is so large it did not seem worthwhile to explore it due to our time constraint.

3.1.2 Traffic Control Methods

We formulated five methods of control based on our observations from the basic tests above as well as some of the popular systems in use.

Method 1: Have non-greedy yields at every intersection.

Method 2: Have non-greedy stops at every intersection.

Method 3: Have the strict stoplight on the major roads and yields on all the minor roads.

Method 4: Have the density stoplight on the major roads and yields on all the minor roads.

Method 5: Have the $u_{opt}$ stoplight on the major roads and yields on all the minor roads.

3.1.3 Results of Models

We looked at a few runs of different basic setups with equal probabilities what cars coming in a particular entrance would go to each of the other exits. We only ran the methods for extremely high traffic densities ($\rho = 0.50$) and we looked at large circle (6 roads) size only due to the runtime constraints of the program. The models ran with several different road weights simulating one, two and three major roads and the rest minor roads. See Appendix A Tables 3, 4 and 5 for the data from these runs.

We found that for one major road entering the circle, Method 3 works best. For two or three major roads entering the circle, Method 5 works best. It should be noted that placing the $u_{opt}$ stoplights on all roads would be the most optimal, but this is unnecessary for small roads entering the traffic circle as our basic tests have shown.

3.2 Analysis & Solutions

From the quantitative analysis of our test traffic circles along with a great deal of intuition we reached the conclusion that Method 5 is best for circles that have two or more major roads ($\rho \geq 0.6$) while Method 3 is best where there is only one major road. It should be noted that the timing of the traffic light was not one of our considerations. In our tests it was a constant duration traffic light that was not affected by the current traffic densities.
For small circles, we can infer from our runs that it would be best to have non-greedy yields for circles with minor roads and stop signs if there is a major road. If there are two major roads then intuitively, there probably should not be a traffic circle, rather just have the two major roads connect with the minor roads intersecting normally.

3.3 Strengths and Limitations of Our Model

The continuous model provided simplicity that proved to be invaluable given the time constraints for solving this problem. It allowed us to focus on general attributes of a traffic circle and obtain results that could be used in a thorough analysis of a real world traffic circle. Furthermore, our model allowed us to easily extract data from our simulations.

The biggest limitation of our model is that it does not simulate multiple lanes. This might prevent our model from accurately simulating large traffic circles. Since we treated the problem continuously, we are not able to account for individual car’s actions and behaviors.

4 Technical Summary

Here we present a straightforward approach to determine traffic controls in a traffic circle. Let a small circle be defined as having three to five roads connecting to it and large as having six to eight roads. Traffic density through the circle is a variable that should be carefully measured before this analysis is done. It is possible for a single traffic circle to fall under multiple categories depending on the time of day. If traffic lights are used, their programming can be changed based on the time of day.

Following the guidelines, install the traffic controls for the traffic circle fitting best to one of the categories listed:

Small traffic circle with no major roads  Placing yield signs on the incoming roads leads to a relatively well optimized traffic circle. These are cost effective and will still prevent congestion if typical capacity is exceeded.

Small traffic circle with several major roads  Use stop signs on the major roads and yield signs on the minor roads to optimize the flow of traffic. It should be noted that a stop light is relatively inefficient since cars can clear the traffic circle quickly.

Medium traffic circle with one major road  Under most conditions, the strict traffic light installed at each intersection of the traffic circle and the major road along with yield signs at all the minor roads is sufficient for maximizing flow through the circle. The traffic light should be timed so as to give cars entering the circle time to fully clear it. This way the major road traffic does not create too much congestion in the circle.

Medium traffic circle with more than one major road  It is best if a smart traffic light installed at each intersection of the traffic circle and major road. If one
desires to use fewer traffic signals, putting stop signs at every incoming road gives a reasonably close result.

**Large traffic circle** We suggest that one place smart traffic lights at all major intersections and then yield signs for all minor roads.

It is not necessary to stick to these guidelines strictly. If the traffic circle does not fit into one of these categories it can be explicitly programmed into our model and tested. This allows a very detailed analysis of many circle geometries with many different control systems.

5 Conclusion

We obtained a collection of general results with our model that allow us to give recommendations for basic categories of traffic circles. We present examples of more specific geometries and traffic conditions. Our model gives results which seem intuitively reasonable. Most importantly we have created a modular code base that can be adapted to many other geometries and traffic conditions.

We can conclude from our data that in general, incoming streets with large traffic flux are best controlled with traffic lights designed to optimize the speed in the circle. Incoming streets with small traffic flux can be adequately controlled with a yield sign.

5.1 Future Work

Ideally, for a problem of such complexity we would have liked to have worked on this problem for a longer period of time. Firstly, modeling a time dependent input flux would be useful in order to simulate traffic flow varying on the time of day. We would have liked to model the exit roads and their intersections with other roads close to the circle so as to watch under what conditions those exit roads fill up, causing the traffic circle to get congested. It would also be useful to examine existing traffic circles in order to calibrate our model to give more accurate results instead of normalizing speeds and densities to one.

A more accurate model with actual cars instead of density points where driver decisions, lane changes, accidents, pedestrians, varying types of vehicles (trucks, buses, motorcycles, bicycles, etc.) would all be taken into account would be ideal. However, that would require a significant amount of time as well as a relatively large budget. Such a model would allow for the simulation of traffic at the microscopic level in the roundabout. This would allow us tackle the issue of modeling multiple traffic lanes in the traffic circle, entry roads and exit roads. The simulation of traffic on such a level is difficult since the aforementioned “Butterfly Effect” would certainly create irreproducible effects due to the chaotic nature of individual cars in traffic.
References


### Appendix A: Tabulated Results

<table>
<thead>
<tr>
<th>Roads</th>
<th>Control System</th>
<th>In Flow Rate</th>
<th>Steady-state Traffic/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Greedy Yield</td>
<td>0.2</td>
<td>0.039</td>
</tr>
<tr>
<td>3</td>
<td>Greedy Yield</td>
<td>0.8</td>
<td>0.202</td>
</tr>
<tr>
<td>3</td>
<td>Non-greedy Yield</td>
<td>0.2</td>
<td>0.039</td>
</tr>
<tr>
<td>3</td>
<td>Non-greedy Yield</td>
<td>0.8</td>
<td>0.206</td>
</tr>
<tr>
<td>3</td>
<td>Greedy Stop</td>
<td>0.2</td>
<td>0.039</td>
</tr>
<tr>
<td>3</td>
<td>Greedy Stop</td>
<td>0.8</td>
<td>0.202</td>
</tr>
<tr>
<td>3</td>
<td>Non-greedy Stop</td>
<td>0.2</td>
<td>0.039</td>
</tr>
<tr>
<td>3</td>
<td>Non-greedy Stop</td>
<td>0.8</td>
<td>0.218</td>
</tr>
<tr>
<td>3</td>
<td>Light: Strict</td>
<td>0.2</td>
<td>0.032</td>
</tr>
<tr>
<td>3</td>
<td>Light: Strict</td>
<td>0.8</td>
<td>0.221</td>
</tr>
<tr>
<td>3</td>
<td>Light: Density Optimized</td>
<td>0.2</td>
<td>0.039</td>
</tr>
<tr>
<td>3</td>
<td>Light: Density Optimized</td>
<td>0.8</td>
<td>0.214</td>
</tr>
<tr>
<td>3</td>
<td>Light: u Optimized</td>
<td>0.2</td>
<td>0.039</td>
</tr>
<tr>
<td>3</td>
<td>Light: u Optimized</td>
<td>0.8</td>
<td>0.230</td>
</tr>
</tbody>
</table>

**Table 1: Traffic Control - Basic Setup - Small Circle**

<table>
<thead>
<tr>
<th>Roads</th>
<th>Control System</th>
<th>In Flow Rate</th>
<th>Steady-state Traffic/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Greedy Yield</td>
<td>0.2</td>
<td>0.077</td>
</tr>
<tr>
<td>6</td>
<td>Greedy Yield</td>
<td>0.8</td>
<td>0.228</td>
</tr>
<tr>
<td>6</td>
<td>Non-greedy Yield</td>
<td>0.2</td>
<td>0.078</td>
</tr>
<tr>
<td>6</td>
<td>Non-greedy Yield</td>
<td>0.8</td>
<td>0.226</td>
</tr>
<tr>
<td>6</td>
<td>Greedy Stop</td>
<td>0.2</td>
<td>0.077</td>
</tr>
<tr>
<td>6</td>
<td>Greedy Stop</td>
<td>0.8</td>
<td>0.224</td>
</tr>
<tr>
<td>6</td>
<td>Non-greedy Stop</td>
<td>0.2</td>
<td>0.078</td>
</tr>
<tr>
<td>6</td>
<td>Non-greedy Stop</td>
<td>0.8</td>
<td>0.218</td>
</tr>
<tr>
<td>6</td>
<td>Light: Strict</td>
<td>0.2</td>
<td>0.060</td>
</tr>
<tr>
<td>6</td>
<td>Light: Strict</td>
<td>0.8</td>
<td>0.236</td>
</tr>
<tr>
<td>6</td>
<td>Light: Density Optimized</td>
<td>0.2</td>
<td>0.078</td>
</tr>
<tr>
<td>6</td>
<td>Light: Density Optimized</td>
<td>0.8</td>
<td>0.218</td>
</tr>
<tr>
<td>6</td>
<td>Light: u Optimized</td>
<td>0.2</td>
<td>0.078</td>
</tr>
<tr>
<td>6</td>
<td>Light: u Optimized</td>
<td>0.8</td>
<td>0.238</td>
</tr>
</tbody>
</table>

**Table 2: Traffic Control - Basic Setup - Large Circle**

It should be noted that Method 4 did not stabilize for tests 1 and 2.
<table>
<thead>
<tr>
<th>Method</th>
<th>Steady-state Traffic/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.237</td>
</tr>
<tr>
<td>2</td>
<td>.237</td>
</tr>
<tr>
<td>3</td>
<td>.241</td>
</tr>
<tr>
<td>4</td>
<td>NA</td>
</tr>
<tr>
<td>5</td>
<td>.229</td>
</tr>
</tbody>
</table>

Table 3: Methods 1-5: Test 1

<table>
<thead>
<tr>
<th>Method</th>
<th>Steady-state Traffic/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.230</td>
</tr>
<tr>
<td>2</td>
<td>.232</td>
</tr>
<tr>
<td>3</td>
<td>.226</td>
</tr>
<tr>
<td>4</td>
<td>NA</td>
</tr>
<tr>
<td>5</td>
<td>.234</td>
</tr>
</tbody>
</table>

Table 4: Methods 1-5: Test 2

<table>
<thead>
<tr>
<th>Method</th>
<th>Steady-state Traffic/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.230</td>
</tr>
<tr>
<td>2</td>
<td>.244</td>
</tr>
<tr>
<td>3</td>
<td>.230</td>
</tr>
<tr>
<td>4</td>
<td>.222</td>
</tr>
<tr>
<td>5</td>
<td>.236</td>
</tr>
</tbody>
</table>

Table 5: Methods 1-5: Test 3
Appendix B: Matlab Source Code

Main Code  

function out = model5();

% this model is based on the previous ones. An issue that the previous ones
% had was a lag during which cars entered the circle and then would leave.
% Previous ones instantly removed cars elsewhere, in some cases going to
% negative density.
% this model will have separate "circles" for each exit point. Each entry
% point will contribute a certain density to that "circle". These
% "circles" also get summed to the total circle, whose density may not
% exceed pmax.
% the traffic flow PDE then is used to determine the flow speed. This is
% then relayed to each of the "circles" and the densities are moved
% appropriately.
% In each of these "circles" density is partially or fully removed
% (depending on the density of the circle at the intersection, someone
% might need to circle multiple times to reach the exit, or maybe the exit
% is backed up).
% This program will only keep track of the last step in order to save
% space. Then it will use a length(x)*n matrix where n is the number of
% intersections to model the "circles". There will then be another matrix,
% which only needs to keep track of the last step, but will probably be
% made to have the full time copy. This matrix will simply be the sum of
% the length(x)*n matrix.
% This sum-matrix will be acted on only by the traffic PDE. There will be
% no addition/removal algorithms that directly affect the sum-matrix.
% The algorithms just listed will affect the exit "circle" matrixes.
format long;
% initial constants
% pmax = 1;
% a = inline('(p-(p^2)/1)', 'p');% make sure to divide p^2 by max
% aprime = inline('(1-2*p)');% you need to be really careful with these numbers to ensure that you dont
% screw up lambda: lambda should be less than 1/umax.
% also if you find that the density is getting really "spikey" you need to
% increase the N value (k~1/N which is the time step) - need to refine the
% timestep
I = 100;
N = 400;
pthresh = .1*pmax;
psthresh = .05*pmax;
% umax = 1;
umax = 30;
diameter=3;
I_road = 30+1;
T=64/umax;
h=(pi*diameter)/I;
k=T/(8*N);
\begin{verbatim}
lambda = k/h;

\%set x and t vectors
x = 0:h:(pi*diameter);
t = 0:k:T;

\%pfun locations [1,N+1]
pfunloc = [5 22 38 54 73 89];

\%stop light variables
period = .0628;
duration = .0314*5;
phase = .0314/4;

\%pfun contains the probability functions for the entry points
A = pfun(x)

\%define psi probabilities - percentage of cars going from one node to another
\%can be a function of time
psi1 = [0 1/5 1/5 1/5 1/5 1/5];
psi2 = [1/5 0 1/5 1/5 1/5 1/5];
psi3 = [1/5 1/5 0 1/5 1/5 1/5];
psi4 = [1/5 1/5 1/5 0 1/5 1/5];
psi5 = [1/5 1/5 1/5 1/5 0 1/5];
psi6 = [1/5 1/5 1/5 1/5 1/5 0];
psi = [psi1' psi2' psi3' psi4' psi5' psi6'];

\%road weights (.7 for large, .4 for medium and .2 for small)?
road_weight = [.7 .4 .2 .7 .7 .2];

\%which event occurs at which entry
event_entry = [9 4 4 9 9 4];

\%weight events - based on number of different events - currently 10
event_weight(1) = 1;
event_weight(2) = 1;
event_weight(3) = 1;
event_weight(4) = 1;
event_weight(5) = 1;
event_weight(6) = 1;
event_weight(7) = 1;
event_weight(8) = 1;
event_weight(9) = 1;
event_weight(10) = 1;

\%separation in number of cells between the entry and exit for each %intersection
exitseparation = 3;

\%sign constants
pstopsign = .3;

\%U = ones(length(x),length(t));
U = zeros(length(x),length(t));
\end{verbatim}
Unew = zeros(length(x), 1);
tempcircles = zeros(length(x), length(pfunloc));
%
tempcircles = ones(length(x), length(pfunloc));
val = zeros(length(t), 1);
val = zeros(length(t), 1);
U_road = zeros(I_road, length(pfunloc));
%
modify U here if you want an initial density function
% can also set initial tempcircle density distribution
%U=.25*U;
%tempcircles=(.25/4)*tempcircles;
%initial metric
[pavginit uavginit psiginit usiginit fluxinit] = metric(U(:, 1), x, umax);
fluxadd = 0;
fluxremove = 0;
etfluxadd = -fluxinit;
etfluxremove = -fluxinit;
stopflux = 0;
%
Lax-Friedrichs Method
for i = 1:length(t)
changing = zeros(length(x), 1);
Unew = lax_friedrichs(a, lambdata, U(:, i), x, umax);
% find the evolutionary change in the U matrix
% the tempcircles will mirror this change
% not quite sure what factor the removal will have
for j = 1:length(x)
    changing(j) = 1/2*umax*((1-Unew(j)) + (1-U(j,i)));
end
Unew = zeros(length(x), 1);
for j = 1:length(pfunloc)
    hold = pfunloc(j);
    A = pfun(U(hold, i), (i-1)*k);
% can put in hold1/2 event vectors for each J so that a different
% event is occurring at each intersection.
    event(1) = 1; % no event is slowing traffic down
    event(2) = 0; % road is closed
    [event(3) event(4)] = yieldsign(road_weight(j)*U_road(I_road, j), U(hold, i), pmax);
    [event(5) event(6)] = stopsign(road_weight(j)*U_road(I_road, j), U(hold, i), pstopsign,
        pmax);
    event(7) = stoplightstrict(duration, period, j*phase, (i-1)*k, road_weight(j)*U_road(I_road, j),
        U(hold, i), pmax);
    event(8) = stoplightdensity(road_weight(j)*U_road(I_road, j), U(hold, i), pmax);
    event(9) = stoplightuopt(.5*pmax, U(hold, i), road_weight(j)*U_road(I_road, j));
% needs duration period phase
% needs fluxamount and tinit
% fluxamount = .5;
% tinit = 0;
%[event(10) stopflux] = stoplightflux(fluxamount, stopflux, tinit, k, U_road(I_road, j),
    U(hold, i));
event = hold1;
Ba = psi*A';
% its the exit
% allow for separation between entry and exit (hold is entry
% hold - exit separation is exit
%
uxremove = uxremove + tempcircles(hold,j)*(1-remove_percentage(U(hold,i),tempcircles(hold,j), pmax, pthresh, pthresh));
 tempcircles(hold,j) = tempcircles(hold,j)*remove_percentage(U(hold,i),tempcircles(hold,j), pmax, pthresh, pthresh);

uxremove = uxremove + tempcircles(hold,j);
 tempcircles(hold,j) = 0;
uxremove = uxremove + tempcircles(hold - exitseparation,j);
 tempcircles(hold - exitseparation,j) = 0;
uxremove = uxremove + tempcircles(hold - exitseparation,j)*(1-rem_percentage(U(hold - exitseparation,j),tempcircles(hold - exitseparation,j), pmax, pthresh, pthresh));
 tempcircles(hold - exitseparation,j) = tempcircles(hold - exitseparation,j)*rem_percentage(U(hold - exitseparation,j),tempcircles(hold - exitseparation,j), pmax, pthresh, pthresh);

% its an entry
for z=1:length(pfunloc)
 if z==j
 continue;
 end
 B=entry_roads(U_road(:,j), diameter, A(j), k, event(z), a);
 for m=1:I_road
 U_road(m,j)=B(m);
 end

 weights = road_weight(z)*event(event_entry(z))*event_weight(event_entry(z));
 tempcircles(hold,z) = tempcircles(hold,z) + weights*psi(j,z)*U_road(I_road,j);

 fluxadd = fluxadd + weights*psi(j,z)*U_road(I_road,j);

 tempcircles(hold,j) = tempcircles(hold,j) + hold2*dot(psi(j,:),A);
 fluxadd = fluxadd + hold2*dot(psi(j,:),A);

 if B(j)>U(hold,i)
 B(j)=U(hold,i);
 end
 if A(j)>1-U(hold,i)
 A(j)=1-U(hold,i));
 end
 flux = flux + A(j); % amount entering
 fluxb = fluxb + B(j); % amount leaving

 Unew(hold) = Unew(hold) + (hold1*A(j)-B(j));
temp-circle = zeros(length(x), 1);
for m=1:length(x)
delta-x = change(m)*k; %distance the density moves
delta = delta_x/h;
delta-floor = floor(delta);
m1 = m + delta-floor;
difference = delta - delta_floor;
if m1>length(x)
m1 = m1 - length(x);
elseif m1 == 0
m1 = length(x);
elseif m1<0
m1 = m1 + length(x);
end
m2 = m1+1;
if difference==0;
m2 = m1;
elseif m2>length(x)
m2 = m2-length(x);
elseif m2 == 0
m2 = length(x);
elseif m2<0
m2 = m2 + length(x);
end
% m
% change(m)
% m1
% m2
% j
% temp-circle(m)
% delta
% delta-floor
% split across adjacent bins based on the diff value
% need to check to make sure you do not overfill the bins
temp-circle(m1) = temp-circle(m1) + temp-circle(m,j) * (1-difference);
temp-circle(m2) = temp-circle(m2) + temp-circle(m,j) * difference;
end
for m=1:length(x)
temp-circle(m,j)=temp-circle(m);
Unew(m)=Unew(m)+temp-circle(m,j);
end
end
Unewhold = Unew;
count = 1;
% if a cell is above pmax you need to split it over previous
% cells.
% runs into issues with
while count
count = 0;
for m = 1:length(x)
m_down = m - 1;
if m == 1, m_down = length(x); end  % these complete the circle
if Unew(m) > pmax
    Unew(m_down) = Unew(m_down) + (Unew(m) - pmax);
    Unew(m) = pmax;
    count = count + 1;
end
end
end
% figure out the change in these moves, ideally you could just
% multiply the temp circles by this, but there could be zeros
changer = zeros(length(x), 1);
% adder
adder = zeros(length(x), 1);
% Unew = Uhold - Unew;
for m = 1:length(x)
    if Unewhold(m) == 0
        changer(m) = (Unew(m) / Unewhold(m));
        adder(m) = 0;
    else
        changer(m) = 1;
        adder(m) = Unewhold(m) - Unew(m);
    end
end
Unew = zeros(length(x), 1);
for j = 1:length(pfunloc)
    for m = 1:length(x)
        if Unewhold(m) == 0
            tempcircles(m, j) = changer(m) * tempcircles(m, j) + adder(m) * (tempcircles(m, j) / Unewhold(m));
        end
        Unew(m) = Unew(m) + tempcircles(m, j);
    end
end
% for j = 1:length(x)
%    Unew(x) = sum(tempcircles(:, j));
% end
% fluxsum = fluxsum + sum(B);
% current metrics
sump = 0;
for j = 1:length(x)
    sump = sump + Unew(j);
end
pavg = sump / length(x);
uavg = umax * (1 - pavg);
flux1 = pavg * uavg;
netuxadd = fluxadd + flux1;
netuxremove = fluxremove + flux1;
pval(i) = pavg/pmax;
ual(i) = uavg/umax;

%average U at time i+1 values (3 adjacent cells)
%Uavg = density_averager(Unew,x);
Uavg = Unew;
for j=1:length(x)
U(j,i+1) = Uavg(j);
end
plotter(U(:,i)', x, netuxadd, netuxremove, pval(i), uval(i), diameter, 100*i*k/T);
%plotter(tempcircles(:,3)', x, netuxadd, netuxremove, pval, uval(i), diameter);
%polar((x+20)*2*pi/40, (1+U(:,i)'), 'r');
%axis([-2 2 -2 2])
%plot(U_road(:,1));
%pause();
pause(.0001);
end
for z=1:length(uval)
ual(z) = uval(z)*pval(z);
end
h = figure;
figure(h);
plot(uval);
%calculate the metric
[pavg uavg psig usig flux1] = metric(U(:, length(t)), x, umax);
%flux = flux + flux1;
%netflux = flux-fluxinit;
%
% % %deltas
% dpavg = pavg-pavginit
% duavg = uavg-uavginit
% dpsig = psig-psiginit
% dusig = usig-usiginit
% netflux
end

Entry_roads function Unew = entry_roads(U, r, abegin, k, aend, a);
% This program models the entry roads where there is some time-dependent
% density at its start which propagates down the road to some event, such
% as a stoplight, stop sign or yield sign before entering the traffic
% circle.
% initial constants
I = size(U);
% set speed and density for these side roads
umax = 10;
% $p_{\text{max}} = 0.6$;
$h = r / ((I(1)-1));$
$\lambda = k / h$;
% set x and t vectors
$x = 0: h: r;$
% $U = \text{zeros(length(x),1)}$;
$Unew = \text{zeros(length(x), 1)}$;
% modify $U$ here if you want an initial density function
% can also set initial tempcircle density distribution
% Lax-Friedrichs Method
$U(1) = \text{begin};$
$U(\text{length(x)}) = U(\text{length(x)}) - \text{aend};$
$Unew = \text{lax_friedrichs}(a, \lambda, U, x, \text{umax});$
$\% Unew = Unew * p_{\text{max}};$
\text{end}

\textbf{Lax-Friedrichs} function \text{out} = \text{lax_friedrichs}(a, \lambda, U, x, \text{umax});
\text{format long;}
\text{Unew} = \text{zeros(length(x), 1)};
\text{% Lax-Friedrichs Method}
\text{for } j = 1: \text{length(x)}
\text{\quad j$_\text{up}$} = j + 1;
\text{\quad j$_\text{down}$} = j - 1;
\text{\quad if } j == 1, \text{\quad j$_\text{down}$} = \text{length(x)}; \text{\quad end} \% these complete the circle
\text{\quad if } j == \text{length(x)}, \text{\quad j$_\text{up}$} = 1; \text{\quad end}
\text{\quad Unew}(j) = (1/2) * (U(j$_\text{up}$) + U(j$_\text{down}$)) - (\lambda / 2) * (\text{umax} * a(U(j$_\text{up}$)) - \text{umax} * a(U(j$_\text{down}$)));
\text{\quad end}
\text{\quad out} = \text{Unew};\text{\quad end}

\textbf{Pfun} function $A = \text{pfun}(p, t);$ format long;
\text{if } t < 0.5
\text{\quad A(1) = (1 - p) * sin(t);}\text{\quad end}
\text{\quad A(2) = (1 - p) * 3 * sin(t);}\text{\quad end}
\text{\quad A(3) = (1 - p) * cos(t);}\text{\quad end}
\text{\quad A(4) = (1 - p) * 3 * sin(t);}\text{\quad end}
% check to make sure the entry density isn't negative (if so make it 0)
%for i=1:length(A)
% if A(i)<0
% A(i)=0;
% end
%end
%else
% A(1)=0;
% A(2)=0;
% A(3)=0;
% A(4)=0;
%end
% A(1) = (1 - p)*.1;
% A(2) = (1 - p)*.3;
% A(3) = (1 - p)*.4;
% A(4) = (1 - p)*.3;
A(1) = .8;
A(2) = .8;
A(3) = .8;
A(4) = .8;
end

Plotter  function plotter(U, x, netuxa, netuxb, pval, uval, diameter);
% Plots the density as you advance in time.
format long;
%plot(x,lil_u);
polar(x*2/diameter,1+U,'r');
title(sprintf('net ux add: %f \t net ux remove: %f \t avg densit y: %f pval \t
avg speed: %f',netuxa, netuxb, pval,uv al))
%axis([-20 20 0 1])
axis([-2 2 -2 2])
end

Remove density function out = remove_percentage(pnet, pspecific, pmax, pthresh, pthresh);
% This program calculates a probability that a car will not be able to
% leave the circle when it wants to, making it have to loop around again.
% if pnet is high then cars will be moving slowly and will have trouble
% getting out.
% if pspecific is high, or large in proportion to pnet, then those cars
% will block each other, forcing some to go around again.
% there will be some threshold beyond which congestion will drop
% dramatically. This model just says all resistance drops to 0 if both
% thresholds are satisfied.
format long;
% prevent divide by 0 error and check thresholds
if pnet == 0 || (pnet<pthresh && pspecific < pthresh)
out = 0;
elseif pnet < pthresh && ~((pspecific < psthresh)
out = (1-pspecific/pnet);
elseif ~(pnet < pthresh) && pspecific < psthresh
out = (pmax-pnet);
else
out = (pmax-pnet)*(1-pspecific/pnet);
end
end

Stop Light Density  function out = stoplightdensity(pin, pcircle, pmax);
  % This program implements the basic concept of a stoplight
  % This program looks solely at the values of pin and pcircle and lets
  % whichever one go which is greater.
  % The path with a higher density will create a higher pressure.
  format long;
  if pin > pcircle
    % light is green
    A=yieldsign(pin, pcircle, pmax);
    out = A(1);
  else
    % light is red
    out = 0;
  end
end

Stop Light Strict  function out = stoplightstrict(duration, period, phase, t, pin, pcircle, pmax);
  % This program implements the basic concept of a stoplight
  % The stoplight is red for the duration, beginning at some time tinit and
  % ending at time tinit + duration
  % No density gets through a red light. When it is green however, it acts
  % as a greedy yield.
  format long;
  if mod(t-phase,period) > duration
    % light is green
    A=yieldsign(pin, pcircle, pmax);
    out = A(1);
  else
    % light is red
    out = 0;
  end
end

Stop Light uopt  function out = stoplightuopt(popt, pcircle, pin);
  % This program implements the basic concept of a stoplight
  % This program tries to keep the circle moving at what has been determined
  % to be the optimal velocity.
format long; 
if pcircle < popt 
  \% is less than the optimal, so we can let cars in 
  A = yieldsign(pin, pcircle, popt); 
  out = A(1); 
else 
  \% it is above optimal and letting more cars in would slow things down. 
  out = 0; 
end 
end

**Yield Sign** function [A B] = yieldsign(pin, pcircle, pmax); 
  \% This program implements the basic concept of a yield sign 
  \% The A value is the "greedy yield" where the intersection will fill up the 
  \% circle top either its pmax or until all of the density is transferred to 
  \% the circle 
  \% The value B is "non-greedy yield" where some percentage of the entry density will 
  \% get in. The lower the density on the circle, the greater chance that a 
  \% car has of getting into the circle, and thus more density is then 
  \% transferred. 
format long; 
popen = (pmax-pcircle); 
\%entry right of way 
if pin < popen 
  A = pin; 
else 
  A = popen; 
end 
\% circle has right of way 
\% percentage fill 
B = popen * pin; 
end

**Stop Sign** function [A B] = stopsign(pin, pcircle, pstopmax, pmax); 
  \% This program implements the basic concept of a stop sign 
  \% The A value is the "greedy stop" and value B is the "non-greedy stop" 
  \% A stop sign just puts a limit on the number of cars which can pass 
  \% through an intersection, that is it limits some maximum density. 
  \% It can also be argued that it introduces a slight lag in the system, but 
  \% this is negligible (or so it is assumed) compared to the overall traffic 
  \% patterns. 
  \% This program just checks to see if the pin exceeds the pstopmax. If it 
  \% does than pin is changed to pstopmax. Then after this check the results 
  \% are fed into the yieldsign program. 
format long; 
popen = (pmax-pcircle); 
if pin > pstopmax
hold = pin;
pin = pstopmax;
end
[A B] = yieldsign(pin, pcircle, pmax);
end