For office use only	Team Control Number	For office use only
T1 _____	**71536**	F1 _____
T2 _____		F2 _____
T3 _____	Problem Chosen	F3 _____
T4 _____	**B**	F4 _____

2017
MCM/ICM
Summary Sheet.

The design of merging infrastructure for vehicles exiting a tollbooth and re-converging onto a highway must balance multiple, sometimes paradoxical, factors. For example, while most highway-users might think design considerations should expedite toll service, a toll-booth system that processes vehicles too quickly risks creating a gridlock of vehicles surging onto the merging area. Our goal is to identify the "balance point" of all considerations with mathematical precision. To achieve this, we frame the selection of an optimal design for a post-tollbooth merge (PTM) as a discrete optimization problem of some queue-theoretical quantities, providing decision-makers with the means to choose the design optimal to their use case. More significantly, our model delineates the boundaries of the following considerations: the boundary at which increased flow from toll-booths forces infrastructure to process vehicles at a faster rate, even if at a greater cost; and the boundary at which too many autonomous cars delay, rather than expedite, transit through toll-booths.

Contents

1 Introduction

Merges in traffic are one of the major contributors of road accidents [1]. They take place most often at places where *multiple* lanes converge, commonly in the merging areas immediately after toll-booths. Vehicles from a highway split into smaller lanes for toll booths, which are usually more than the number of lanes. After paying their toll, vehicles move back into the original number of lanes that the highway comprised of, often resulting in accidents.

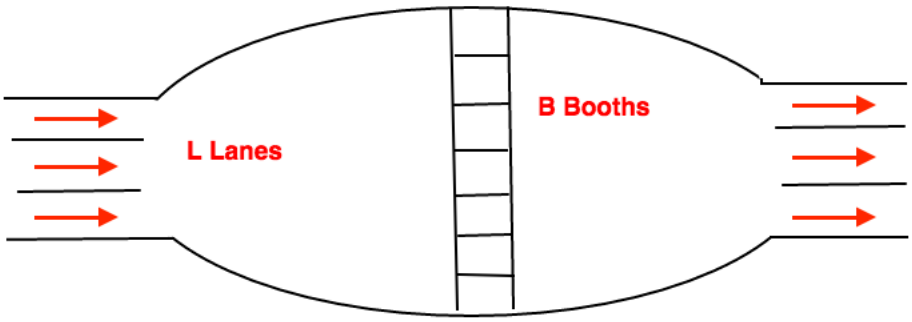
Apart from accidents, the process of traffic merging together is cumbersome; merging reduces the average speed of vehicles and increases travel time [2]. Bearing in mind the safety of vehicular users and the consequences of overcrowding upon exit from a toll-booth, this paper attempts to minimize the financial cost of the "fanning-in" or merging process.

In this paper, we use some results from Queuing Theory to frame a discrete optimization problem and algorithm for it. Our model finds the optimal way of building a certain highway consisting of L lanes and B toll-booths using two infrastructural pieces, traffic lights and barriers.

2 Model

The design of the post-tollbooth merge (PTM) must carefully balance several competing factors. For example, with more infrastructure in the PTM (such as traffic lights or lane barriers), we can expect to incur greater construction and maintenance costs. We can also expect that past a point, more infrastructure decreases the exit rate of vehicles, potentially creating a road blockage in the PTM. On the other hand, a PTM with little to no infrastructure compels vehicular users to negotiate exit from a toll-booth amongst themselves, creating a chaotic situation that might compromise the safety of vehicular users. Negotiating a balance between these competing factors naturally invites an optimization model. In what follows, we explain the workings of a discrete optimization model that chooses the most cost effective assortment of infrastructural features from all possible assortments which ensure vehicular user safety and smooth transit.

Figure 1: Shape of a one-sided highway



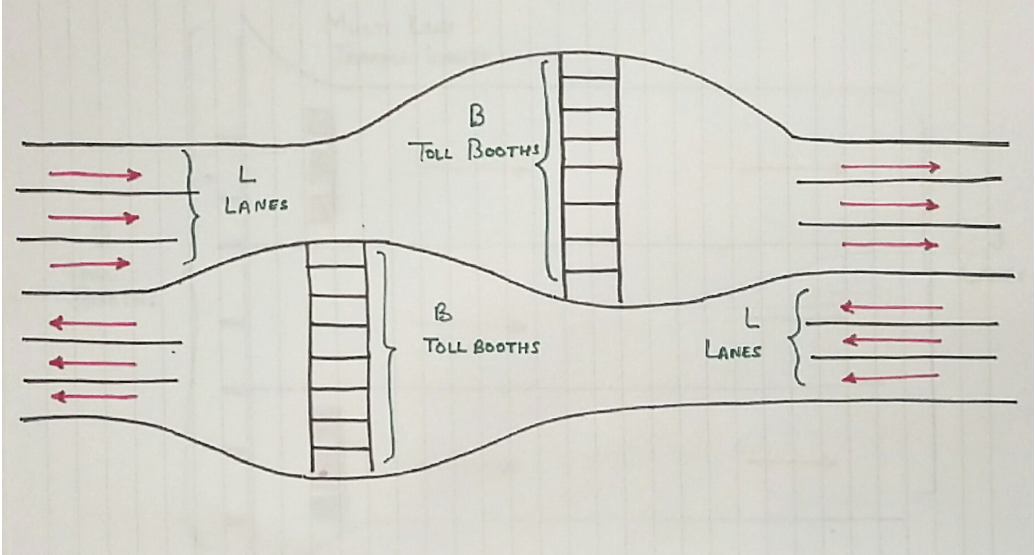
2.1 Assumptions

1. The infrastructure takes one of two forms: a road barrier or a multi-lane traffic light.

   Road barriers restrict the vehicles which are leaving the toll-booths to certain, designated lanes. In our case, the placement of road barriers partitions the B toll booths into L summands or parts. Figure 3 shows a 3 lane highway with 8 toll booths being partitioned into 3 summands allowing 3, 2, and 3 vehicles respectively in each part. L lanes require L-1 road barriers to be built for dividing into summands. Our model also tries to optimize the number of summands for both accident prevention and infrastructure cost.

   Let a_i be the number of toll booths that a summand receives vehicles from. Whenever a summand $i \in \{1 \dots L\}$ contains vehicles that arrive from at least 3 booths ($|a_i| \geq 3$), we require that a multi-lane traffic light to be constructed. The multi-lane traffic light

Figure 2: Shape of a two-sided highway

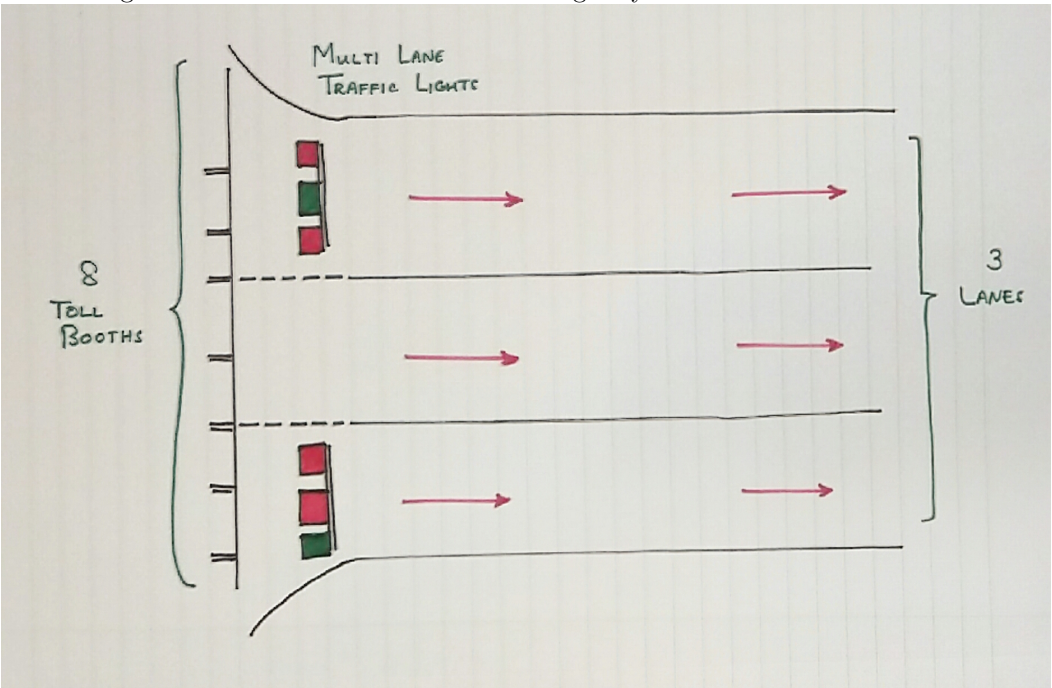


prohibits all but one vehicle from exiting the summand (one vehicle is shown a green light to pass through while the others are shown red lights). In Figure 3, the first summand with 3 vehicles will have 3 traffic lights, one for each vehicle.

2. The process of exiting a toll-booth can be modeled using queuing theory. In particular, for each toll-booth j, the random event of a car leaving a toll-booth follows a rate δ_j. More specifically, immediately after a car leaves a toll-booth, the point in time at which the next car leaves the toll-booth follows an unknown, general probability distribution G with rate δ_j. Since δ_j represents the rate at which a car exits a particular toll-booth j, and many toll-booths can lead to one channel for merging, the "true" rate at which vehicles come onto one channel is given by $\lambda_i = \sum_j \delta_j$ where j is over unspecified but appropriate indices. The true rate is given by a δ function in our model that performs this summation. The random event of a car leaving an infrastructural piece i (either a roadblock or a traffic light) follows a poisson distribution with rate μ_i. That is, once vehicles simultaneously reach the end of the barrier, they must either take the decisions amongst each other (when there are less than 3 vehicles), or a traffic light must make the decision as to who leaves the barrier first. When a vehicle reaches the end of a barrier before making a decision, it will leave the barrier with rate μi, following a poisson distribution. Technically, these queues are G/M/1 queues, but for our purposes, we need only ensure that $\frac{\lambda_j}{\mu_i} \leq 1$ (flow out must be at least as much as flow in). We specify these queues as G/M/1 ones, because the designation will be consequential for future work (see section 6).

These assumptions give rise to the following model. Admittedly, w is pre-determined, but we include it as a parameter to minimize, anyway:

Figure 3: Infrastructure of a one-sided highway with 8 booths and 3 lanes



2.2 Model

$$\underset{(w,t)}{\text{minimize}} \quad \text{Cost}((w,t)) = Xw + Yt$$

$$\begin{aligned}
\text{subject to} \quad & \exists (a_1 \dots a_n) && \implies \sum_{i=1}^{n} (a_i) = B \\
& t = |\{i : a_i \geq 3\}| \\
& n = L \\
& w = n - 1 \\
& \frac{\lambda_i}{\mu_i} \leq 1 && \forall i \\
& \mu_i \leq M && \forall i \\
& \lambda_i = \delta(a_i) && \forall i \\
& w, t \in \mathbb{N} \cup \{0\} \\
& a_i \in \mathbb{Z}^+ && \forall i
\end{aligned}$$

2.3 Algorithm

We devise a brute-force algorithm to solve this optimization problem. The exact algorithm (in Python) is given in the appendix of this paper, but we motivate the key features of the algorithm here:

1. Given a B set, partition it to form a set of all possible summands – that is, the set $\{a_i \dots a_n \mid n = L,\ \sum_{i=1}^{n} a_i = B\}$ – it helps to know how to first enumerate the set. To enumerate the set use a standard combinatorial trick: given a B-set of "1"s and barriers between each 1, $\underbrace{1|1|1|1|1|1}_{B\text{ 1s}}$, systematically select all possible combinations of barriers between the "1"s. Imaginatively, given a selected combination of barriers, collapse the barriers and, consequently, combine the formerly separated "1"s into a single block. For example, if we collapse the first and fourth barrier of the above example, then we get $\{2, 1, 2, 1\}$. There are, therefore, 2^{B-1} possible barriers that can be broken.

2. Over the 2^{B-1} resulting partitions, consider only the ones that satisfy the constraints above – namely, that the partition length be L and that the flow rates not lead to overcrowding. Choose from among these partitions the one with least cost.

3 Results

We assess the optimal post-tollbooth merge (PTM) constructed by our model using the questions outlined in the prompt:

1. Determine the performance of your solution in light and heavy traffic.

2. How does your solution change as more autonomous (self-driving) vehicles are added to the traffic mix?

3. How is your solution affected by the proportions of conventional (human-staffed) toll-booths, exact-change (automated) toll-booths, and electronic toll collection booths (such as electronic toll collection via a transponder in the vehicle)?

We answer these questions in the scenario that vehicles from $B = 8$ toll-booths converge onto $L = 3$ lanes, where a multi-lane traffic light costs \$8000 and a 40 ft barrier costs \$2200 to install and use [7]. Our model and algorithm allows for variability in these aforementioned variables, but we treat them as constant, since from the experience of this paper's authors, most toll-booth services follow $L = 3$ and $B \in [2L, 3L]$. In what follows, we consider the variations in optimal PTM design as a result of variations in our δ function. Recall that the δ function assigns to each member of an L partition of a B-set ($\{a_1 \dots a_L | \sum_{i=1}^{L} a_i = B\}$) the rate (vehicles per second) of vehicles entering the queue. Intuitively, the δ function adds together the rates of vehicles leaving toll-booths, for each cluster of toll-booths that eventually force vehicles to merge onto the same lane. Our assumption is that the scenarios in the questions above (proportions of toll-booths, increased traffic etc.) can be understood solely by considering variations in the rate at which vehicles leave toll-booths.

1. We take "light" traffic and "heavy" traffic to refer to the number of vehicles on a highway. When few vehicles occupy the road way, transit is smooth, whereas when

many vehicles occupy the road way, movement for drivers is inhibited. Therefore, as traffic increases, the rate at which vehicles exit a toll-booth also increases, since the time span between the departure of vehicles reduces. Assuming that all 8 toll-booths are of the same kind, we observe what happens as λ (the rate of vehicles leaving a toll-booth every second) increases from 0.04 to 0.0675 with the increase in traffic. The results are formulated in the table below. They lead to a clear interpretation that, when toll-booth exit rates are not excessively large, minimum costs are obtained by using as few traffic lights as possible (in the case of [1,2,5], there is one traffic light for the 5 toll-booth stalls that converge onto one lane). As traffic rates increase, however, the convergence of vehicles from 5 toll-booth lanes onto 1 lane creates backwash and a longer queue, so toll-booths must be partitioned more equitably and, if need be, more traffic lights must be installed (hence the progression onto [2,2,4] and then [2,3,3], which requires two traffic lights).

λ vehicles/second	Optimal Layout
0.04	[1,2,5]
0.0425	[2,2,4]
0.045	[2,2,4]
0.0475	[2,2,4]
0.05	[2,2,4]
0.0525	[2,3,3]
0.055	[2,3,3]
0.0575	[2,3,3]
0.06	[2,3,3]
0.0625	[2,3,3]
0.065	[2,3,3]
0.0675	no solution

2. Since autonomous vehicles are not dependent on human interaction, they will necessarily visit only the electronic toll-booths (for example, in the United States, EZ-Pass lanes). Passage through electronic toll-booths is a quick process, since computer systems quickly record a piece of identification for a vehicle (for example, a registration plate) and later identify the vehicle's user using the identification. Thus, the rate of passage through electronic toll-booths is much quicker than the rates of passage through other types of toll-booths, and as more autonomous vehicles occupy highways, we can expect rates of passage through these toll-booths to increase. Correspondingly, with the increase in autonomous vehicles, the usage of the other two types of toll-booths decreases. As a result, we expect rates of passage of vehicles through other types of toll-booths to decrease. Running a simulation in which the first two booths are electronic ones and the remainder human-staffed ones, we find that when the distribution of rates for toll-booths is $[0.04, 0.04 \dots]$, the optimal layout is $[1, 2, 5]$, and this continues until $[0.044, 0.044, 0.039 \dots]$. Later on, at an extreme, we have the following pair: $[0.078, 0.078, 0.0305 \dots] \implies [1, 1, 6]$. As the rate of the two electronic toll-booths increases, we find that, increasingly, they must be given their exclusive merging lanes to, presumably, prevent accidents from occurring. This implies that, counter-intuitively, increased efficiency brought about my autonomous cars leads to safety compromises caused by the overcrowding of toll-booth services!

3. Reasoning that, in order, full-service, exact-change, and electronic booths service vehicles with increasing speed, we vary the rates (vehicles/second) of each of the eight toll-booths in three different ways most reflective of the current setup of toll-booths in the United States based on our experiences (all cash, 1/3 each to each component, 1/2 cash and 1/2 electronic). The resulting, optimal designs are below:

$$\{0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04\} \implies [1, 2, 5]$$
$$\{0.05, 0.05, 0.05, 0.06, 0.06, 0.04, 0.04, 0.04\} \implies [2, 2, 4]$$
$$\{0.04, 0.04, 0.06, 0.06, 0.06, 0.06, 0.04, 0.04\} \implies [2, 2, 4]$$

The upshot is that, with even faster methods of service, infrastructure must put up against the greater likelihoods of backwash by choosing to use more expensive infrastructure i.e. more traffic lights and more equitable distributions of vehicles converging onto lanes.

4 Strengths and Weaknesses

4.1 Strengths

1. By considering all possible partitions of B toll-booths, no possible solution escapes this algorithm. Our outline in section 2 should be sufficient to formalize the proof that our algorithm considers all partitions to find the optimal one.

4.2 Weaknesses

1. Our optimization problem is intractable to solve using our algorithm for large enough inputs of B. It considers all 2^{B-1} partitions of the toll-service having B booths. In other words, our algorithm follows an exponential running time, making it difficult to compute for extremely large numbers. However, B is usually relatively small, considering that the number of lanes on a highway and the corresponding number of toll-booths are few in number.

2. Additional weaknesses are considered in Future Work. Indeed, many of the points in Future Work were thought of bearing in mind this model's weaknesses.

5 Future Work

1. Our optimization problem does not consider toll-booth user satisfaction as a consideration. Reckoning that users base their satisfaction on the basis of how long it takes them to exit the toll-booth as well as their perception of how crowded a toll-booth is (a fully engaged toll-booth is not necessarily always a slow one), a future model could add additional constraints into the optimization problem (for example, waiting time to exit the toll-booth might be bounded) or revise the cost function to reflect some component measuring satisfaction as a function of queue length. These revisions can be mathematically formalized. Recall that we specified earlier that the process of exiting a toll-booth can also be modeled as a queue. In our case, each queue is a G-M-1 queue, and closed forms are known for the average amount of waiting time for such queues [6].

2. Our analysis of question 3 solely considered three variations in the proportions of toll-booth types. Of course, many more variations could have been studied: Formally, given n representing the number of toll-booths, the set of all variations is precisely the sum of multinomial coefficients, namely $\sum_{a_1+a_2+a_3} \binom{n}{a_1!a_2!a_3!}$ where $a_1 + a_2 + a_3 = n$. Oftentimes, the logic underlying the enumeration of a finite set informs the algorithm giving all members of the set. With more time, we could have devised this algorithm to thoroughly assess the effects of different proportions of toll-booths.

3. Our analysis of question 2 takes into consideration that as more and more autonomous cars occupy highways (and, hence, electronic toll-booths), fewer and fewer vehicles visit other types of toll-booths. Presumably, however, the total number of vehicles should remain the same. This is not considered in our model, since our model indirectly reflects an increase in autonomous cars by increasing transit rates through electronic toll-booths. Verification is needed of this indirect reflection: that, in the process of increasing toll-booth rates, the sheer number of vehicles is not increased.

6 Letter to the Turnpike Authority

January 23, 2017

To the New Jersey Turnpike Authority:

We are team 71536, and we are excited to introduce our highway-toll model that increases the safety and cost-effectiveness of the post-tollbooth merging process (PTM). In particular, we offer an algorithm to determine, given features of your tollbooths and the condition of road traffic, the ideal shape, size and merging pattern of the (PTM). The ideal geometry of the PTM is not complicated and relies only on the use of wall barriers and multi-lane traffic lights, which are readily removable and placeable.

Researchers agree that most accidents in a PTM occur as a result of drivers trying to navigate the design of the PTM, so we pause to explain some of the safety designs: lanes coming from the toll booths are coalesced into subgroups that are separated from each other using walls; all vehicles in a subgroup can only exit from one lane (that is, they cannot tread onto other ones). Finally, any sub-group catering to more than two toll booths has a multi-lane traffic light to help reconvene the subgroup onto a main highway lane. We reason that more than 2 drivers cannot decide between themselves who should first exit a subgroup and proceed onto a main highway lane.

While giving utmost priority to safety, the algorithm finds the optimal design to reduce financial costs of a design; the infrastructural implementations are also relatively simple, making this an easy to implement solution. Finally, we offer the unsurprising remark that different conditions of traffic warrant different PTM layouts. Our algorithm will help you determine the optimal layout for each set of traffic conditions so that you might alter the layout of your PTM to best optimize transit. We hope that this will improve upon your existing design. Please contact us if you need any clarification or any further details.

Sincere regards,
Team 71536

7 Appendix

Our Python code is given below. All methods work as they are entitled; constructTuple, we take care to note, gives all possible partitions of an n set, when passed with a list containing the power set of an $n-1$ set and the length n. This code is also available on Github:

https://github.com/yogabbagabb/PythonProjects under the directory MCM2017.

7.1 Python Code for Algorithm

```
'''
Created on Jan 22, 2017

@author: ahanagrawal
'''


import numpy as np

import decimal

def drange(x, y, jump):
    while x < y:
        yield float(x)
        x += float(decimal.Decimal(jump))

def powerset(s):
    listOfLists = list()
    x = len(s)
    for i in range(1 << x):
        innerList = list()
        for j in range (x):
            if (i & (1 << j)):
                innerList.append(s[j])
        listOfLists.append(innerList)
    return listOfLists

def constructTuple(power, length):
    finishedTuples = list()
    for aList in power:
        ones = np.ones((1,length), int)
        finTuple = list()
        for num in aList:
            ones[0,num-1] += 1
            ones[0, num] = 1

        total = 0
```

```
        i = 0
        while i < (length):

            if (i == length 1):
                finTuple.append(1)
                i += 1
            elif ones[0,i] <= ones[0, i+1]:
                finTuple.append(1)
                i += 1
            else:
                j = i+1
                total = ones[0,i]
                while (ones[0,j] != 0):
                    total += ones[0,j]
                    j += 1
                finTuple.append(total)
                i = j+1

                total = 0

        finishedTuples.append(finTuple)


    return finishedTuples

def delta(a, b, Delta):
    return sum(Delta[a:a+b])

def minimize(finishedLists, delta, Delta, M, L, length, X = 8000, Y = 2200):
    #assuming 40 ft, 275 * 8 = 2200
    #8000 for 1 portable traffic light
    costDict = dict()
    theMin = 0 # garbage right now
    firstTime = True

    for eachList in finishedLists:

        validEntry = True
        numberGreaterThanThree = 0
        totalGone = 0
        for entry in eachList:
            theLambda = delta(totalGone, entry, Delta)
            if (entry >= 3):
                numberGreaterThanThree += 1
            totalGone += entry
            if (theLambda/M > 1 or len(eachList) != L):
                validEntry = False
                break
```

```
        if (validEntry and len(eachList) == L):
            trafficLights = numberGreaterThanThree
            walls = length    1
            costDict[trafficLights*X + walls*Y] = eachList

            if firstTime:
                theMin = trafficLights*X + walls*Y
                firstTime = False

    for key in costDict:
        if key < theMin:
            theMin = key

    return costDict[theMin]


def distribution():
    thePower = powerset([1,2,3,4,5,6,7])
    for x in thePower:
        print (x)
    print("\n")
    theTuples = constructTuple(thePower, 8)
    for x in theTuples:
        print (x)

    Delta = [0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04]
    DeltaTwo = [0.05, 0.05, 0.06, 0.06, 0.05, 0.04, 0.04, 0.04]
    DeltaThree = [0.04, 0.06, 0.06, 0.06, 0.06, 0.04, 0.04, 0.04]

    print(Delta, minimize(theTuples, delta, Delta, 0.20, 3, 8))
    print(DeltaTwo, minimize(theTuples, delta, DeltaTwo, 0.20, 3, 8))
    print(DeltaThree, minimize(theTuples, delta, DeltaThree, 0.20, 3, 8))


def trafficIncrease():
    thePower = powerset([1,2,3,4,5,6,7])
    for x in thePower:
        print (x)
    print("\n")
    theTuples = constructTuple(thePower, 8)
    for x in theTuples:
        print (x)

    Delta = [0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04]


    print("\n")
```

```
    for i in drange(0, 0.10, '0.0025'):
        Doddy = np.array(Delta)
        Doddy[0:8] += i
        print(Doddy, minimize(theTuples, delta, Doddy, 0.20, 3, 8))

def autonomous():
    thePower = powerset([1,2,3,4,5,6,7])
    for x in thePower:
        print (x)
    print("\n")
    theTuples = constructTuple(thePower, 8)
    for x in theTuples:
        print (x)

    Delta = [0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04]


    print("\n")

    for i in drange(0, 0.10, '0.0025'):
        Doddy = np.array(Delta)
        Doddy[0:2] += i
        Doddy[0:8] = 0.2*i
        print(Doddy, minimize(theTuples, delta, Doddy, 0.20, 3, 8))


if __name__ == '__main__':
    distribution()
```

References

[1] National Highway Traffic Safety Administration, Lane Change/Merge Crashes: Problem Size Assessment and Statistical Description

[2] New Jersey Institute of Technology - National Center for Transportation and Industrial Productivity / International Intermodal Transportation Center, Ten Year Plan to Remove the Toll Barriers on the Garden State Parkway

[3] Design and Evaluation of Toll Plaza Systems by Xiuli Chao, Ph.D. Department of Industrial and Manufacturing Engineering New Jersey Institute of Technology

[4] A Short Introduction to Queueing Theory Andreas Willig Technical University Berlin, Telecommunication Networks Group Sekr. FT 5-2, Einsteinufer 25, 10587 Berlin

[5] The toll plaza optimization problem: Design, operations, and strategies Seongmoon Kim * School of Business, Yonsei University, Shinchon-dong 134, Seodaemun-gu, Seoul, Republic of Korea

[6] The G/M/1 Queue revisited Ivo Adan1 , Onno Boxma1 , David Perry2 1 EURANDOM and Department of Mathematics and Computer Science, Eindhoven University of Technology 2 Department of Statistics, University of Haifa

[7] USBarricades.com, Estimates on traffic light and barrier costs.