CELLULAR AUTOMATA MODELLING OF MULTILANE HIGHWAY TRAFFIC

by

Hunter King

B.S. in Phys., University of Massachusetts (Amherst), 2001

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Physics

Boğaziçi University

2006

ACKNOWLEDGEMENTS

I would like to thank Dr. Levent Kurnaz for his moral support and guidance throughout this work.

I would also like to thank Dr. Ibrahim Semiz and Dr. Ali Nezihi Bilge for their participation in the jury.

I am very much indebted to the extended Turan family for keeping me well fed and providing a comfortable working environment for much of this study.

Also, I would like to thank my wife, Sıla, who helped me through every step of the process and taught me the “printf” function.

ABSTRACT

CELLULAR AUTOMATA MODELLING OF MULTILANE HIGHWAY TRAFFIC

Cellular automata (CA) are often used to model traffic flow with the intent to both better understand the internal dynamics of traffic from a statistical physics point of view, and to create virtual experiments which cannot be normally performed on real roads. Many such models in the recent past have had success in these areas, including the NaSch model we use as a generic example. We have written our own traffic CA model using C++ with the aim of acquiring physically significant and realistic data for a simple virtual highway, and then applying the simulation to more complicated road conditions in order to test the versatility of the model. In analyzing our results, we have studied space-time diagrams showing the position of all vehicles in each time step, and "fundamental diagrams" showing the relationship between flow and density. We have observed two distinct phases in unobstructed flow caused by the formation of spontaneous jams. An unexpected "creeping" in these jams is observed, which is the result of applying our local rule sequentially rather than simultaneously for all vehicles. We observe that allowing lane changing in multilane highways increases total flow by preventing spontaneous jams to some extent. For multilane highways with off ramps, we conclude that placing a sign well ahead of the exit is crucial in preventing large scale congestion in that it allows for exiting cars to change lanes without decelerating and for onward moving cars to move away from the right lane. We also illustrate the ability of our model to produce reasonable results for more complicated highways involving on ramps and auxiliary lanes, which can be used in future research.

ÖZET

ÇOK ŞERİTLİ OTOBAN TRAFİĞİN CELLULAR AUTOMATA MODELLEMESİ

Cellular Automata (CA), istatistik fiziğinin bakış açısıyla trafiğin iç dinamiklerini anlamak, aynı zamanda gerçek yollarda yaratılamayacak sanal deneyler yapmak amacıyla trafik akışını modellemede kullanılan yöntemlerden biridir. Geçmiş yıllarda bizim örnek aldığımız NaSch modelinin de içinde olduğu birçok CA modeli başarıya ulaştı. Bizim trafik CA modelimizi C++ kullanarak yazdık. Programımızı fiziksel olarak anlamlı ve gerçekçi verilere ulaşmak için önce basit bir otobana, daha sonra modelin değişkenliğini test etmek için daha kompleks yol koşullarına uyguladık. Sonuçlarımızı analiz ederken, tüm araçların her bir zaman aralığındaki pozisyonlarını görebilmek için "space-time" diyagramlarını, akış ve yoğunluk arasındaki ilişkiyi görebilmek için "fundamental" diyagramlarını kullandık. Ani tıkanıklıklarının oluşmasından dolayı, herhangi bir engeli olmayan trafik akışında iki farklı faz gözlenmedik. Yerel kuralımızı sırayla (aynı anda değil) tüm araçlara uyguladığımızdan dolayı beklenmedik "çok yavaş ilerleyen" tıkanıklıklar gözlendi. Çok şeritli otobanlarda, şerit değiştirilmesine izin vermemizin, ani tıkanıklıkları engelleyerek toplam akışı yükselttiğini gördük. Çıkışı olan çok şeritli otobanlarda, çıkıştan yeterince önceye tabela yerleştirmenin büyük trafik tıkanıklıklarını, devam eden arabaların sağ şeridi terk etmelerini sağlayarak ve çıkan arabaların yavaşlamadan sollamasına izin vererek, kesinlikle önleyeceği sonucuna vardık. Aynı zamanda modelimizin, çıkışlar ve ek şeritler içeren daha karmaşık otobanlar için mantıklı sonuçlar çıkarma yeteneğini geliştirerek, gelecekteki araştırmalarda kullanılmasına olanak sağladık.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS/ ABBREVIATIONS

d_n	Distance between nth vehicle and its predecessor
f_{pass}	Passing probability term in gas-kinetic model
J	Flow density
P	"Pressure" term in fluid mechanical model
$P(x,v,t)$	Distribution function in gas-kinetic model
P_{des}	"Desired" distribution function in gas-kinetic model
r	Overreaction probability term
s	Deceleration probability term
v_n	Velocity of nth vehicle
v_{max}	Speed limit
x_n	Position of nth vehicle
y_n, z_n	Positions directly left and right of nth vehicle
ρ	Total density
ρ_2	Density of type 2 cars
τ	Relaxation time
ν	"Viscosity" term in fluid mechanical model
CA	Cellular Automata
TCA	Traffic Cellular Automata

1. INTRODUCTION

The mechanics of normal road traffic is an important topic for practical reasons. Besides the inconveniences experienced by commuters and travelers, traffic jams can be the cause of severe inefficiencies for businesses and losses in profit for industries relying on transporting goods. Additionally, as can be seen in more crowded cities, traffic congestion is partly responsible for exhaust related air pollution.

The term traffic jam is casually used to refer to a dense group of cars moving significantly slower than the speed limit of the road it populates. Some external causes of such jams are immediately obvious. Accidents, road work, and road irregularities such as sharp turns and intersections can reduce speeds such that, for higher density flows, jam-like behavior is observed. The ways of eliminating or reducing these causes, when possible, are similarly clear, and are the concern of traffic engineers.

The occurrence of jams in the presence of no apparent external influences indicates that the internal dynamics of traffic flow should also be understood in order to fully address the problem. At first glance, there seem to be too many factors involved to be able to draw any useful relationships. The behavior of any individual vehicle depends on that driver's temperament, memory, and reaction time, as well as the vehicle's own dimensions and other characteristics. In other words, the system is composed of a large ensemble of physically different, interacting bodies controlled by unique psychologies. Developing a fully deterministic and accurate model of the system is clearly hopeless. However, some important questions remain for the optimistic physicist:

What perspectives from physics could be helpful in approaching the problem?
How is the issue of driver psychology to be dealt with without oversimplifying the problem?
How can real life boundary conditions be applied to a prospective model without making calculations impossibly messy?

1.1. PERSPECTIVES AND MODELS

Fluid dynamics seems a hopeful area to begin. At a distance, the flow of cars down a road resemble a two dimensional version of the flow of a compressible, viscous fluid in a pipe. One might at first suppose that the laws of fluid dynamics could be applied to traffic. Unfortunately, some of its basic principles immediately contradict the behavior of cars. For instance, the fluid flow through a narrow portion of pipe is greater than the flow through a wider portion. In the case of traffic, a narrow section of road will produce significantly slower flow. The difference between the systems is profound. Molecules in a fluid move collectively due to pressure gradients, whereas cars move by self-propulsion (are not pushed from behind). Also, the discreteness of molecules in a fluid that makes the small-scale limit for fluid dynamical modeling cannot be avoided in the case of most traffic systems of interest.

Despite these inconsistencies, one of the first models of traffic flow was based on fluid dynamics by Lighthill and Whitham (Lighthill, 1955). The continuity equation, at least, still holds for vehicular traffic, and it makes up the central principle in their model.

$$\frac{\partial \rho}{\partial t} + v_g \frac{\partial \rho}{\partial x} = 0 \tag{1.1}$$

where $\rho(x,t)$ is the density, v_g is $dJ/d\rho$, and $J(x,t)$ is the flow density. The model relies on the total dependence of the flow on the density,

$$J(x,t) = J(\rho(x,t)) \tag{1.2}$$

but this relation must be given explicitly. When a simple form of the relation is inserted (perhaps from empirical data), solutions can be found that show the existence of “kinematic waves”. These waves can be thought of as jam fronts that propagate through the traffic. However, these formations are stable and the theory cannot explain the existence of spontaneous jams.

For a more realistic picture from the dynamics of compressible fluids, a version of the Navier-Stokes equation has been formulated for road traffic

$$\frac{\partial v(x,t)}{\partial t}+v\frac{\partial v(x,t)}{\partial x}=-\frac{1}{\rho}\frac{\partial P(x,t)}{\partial x}+\nu(\rho)\frac{\partial^2 v(x,t)}{\partial x^2}+\frac{v_e(\rho)-v(x,t)}{\tau(\rho)} \tag{1.3}$$

where virtual quantities are used to fill in the fluid-like, but not traffic-like terms. P(x,t) is a "pressure" related to velocity variance, τ is a relaxation time related to delayed response of vehicles, ν is a "viscosity" term, and v_e refers to an equilibrium velocity. As a non-linear, partial differential equation, attaining solutions is difficult even for the most simple conditions, let alone those of practical importance or with irregular boundary conditions. At least this model shows the existence of unstable and metastable traffic states.

The above models are referred to as "macroscopic", in that they don't directly account for the fact that traffic is made up of individual cars. One example of a "microscopic" model that has its background in pure physics is the gas-kinetic model. This approach focuses on a distribution function P(x,v,t) which describes a gas of interacting particles. Time evolution is determined by a Boltzman equation:

$$\frac{\partial P}{\partial t}+v\frac{\partial P}{\partial x}=\left(\frac{\partial P}{\partial t}\right)_{rel}+\left(\frac{\partial P}{\partial t}\right)_{col} \tag{1.4}$$

where $(\delta P/\delta t)_{rel}$ describes the relaxation of the distribution function to a "desired" $P_{des}(v)$ with a characteristic timescale, in the absence of the interaction term $(\delta P/\delta t)_{col}$, which takes the form:

$$\left(\frac{\partial P}{\partial t}\right)_{col}=\int_v^\infty dv'(1-f_{pass})(v'-v)P(x,v',t)P(x,v,t) \\ -\int_0^v dv'(1-f_{pass})(v-v')P(x,v,t)P(x,v',t) \tag{1.5}$$

$(\delta P/\delta t)_{col}$ takes into account both the rate of faster cars meeting slower cars (proportional to $P(x,v,t)P(x,v',t)$) and the probability $(1-f_{pass})$ of being able to overtake them. This can have a positive or negative affect on P and prevent it from reaching P_{des} (Schadschneider, 2002).

There are more examples of analytical microscopic approaches that won't be discussed here. Chowdhury says of these approaches in general "…the actual calculation of even the steady-state properties of traffic from the 'microscopic' models is a highly difficult problem because (apart from the human element involved) (a) the vehicles *interact* with each other and (b) the system is *driven* far from equilibrium, although it may attain a nonequilibrium steady state" (Chowdhury, 1999).

The method of interest to us in this thesis is general enough to model a wide variety of systems, including the self-driven, many particle, out of equilibrium, and even human-dependent systems well beyond the scope of other methods.

1.2. CELLULAR AUTOMATA

It is sometimes said that the concept of the traffic cellular automaton has its origins in statistical mechanics. This may be understood in the sense that its goal is to reproduce accurate macroscopic behavior based on a minimal description of microscopic interactions, but then this is true for all so called microscopic models. The first well known pioneer of cellular automata (CA), John Von Neumann, had originally intended to use the approach in building a human brain-like computer in which processors and data were held on the same footing. He developed the framework of a fully discrete universe of cells which would update their internal states according to a simple recipe. Von Neumann's original plan was never completely finished, but others recognized the potential usefulness of the framework and began applying it to a variety of complex systems (Chopard, 1998). Today, the fundamental idea is being used to better understand or predict, among other things, pedestrian behavior, escape and panic dynamics, spreading of forest fires, population growth and migrations, cloud formations, ant colonies and pheromone trails, and material phenomena such as corrosion, cracks, creases, and peeling (Mikhailov, 2002).

There are four standard ingredients that make up the foundations of any cellular automata model:

The physical environment: This is the space in which the phenomenon of interest takes place. It must be a discrete set of cells, but may involve any topology convenient for the system being modeled. For example, for the CA model for ant pheromone trails, a 1000 by 1000 lattice of Euclidean rectangular cells may be convenient.

The cells' states: Each of the cells in the chosen physical environment are given values corresponding to the allowed physical states of the system. Usually, these values are integers, but are not required to be, as in the case of "coupled map lattices". Often, as in the case of group motion related systems, a cell may have one of two states: vacant or occupied (0 or 1, for instance). An individual cell's value is a "local state" and the set of all the cells' states is referred to as the "global configuration".

The cell's neighborhood: If there is to be any evolution of the local states in the system, the neighborhood within which a cell may be influenced by its environment must be defined. This neighborhood depends on the cell topology chosen, and the range and directionality of interactions in the system. A popular example for Euclidean spaces is the "Von Neumann" neighborhood, composed of the directly adjacent north, east, south, and west cells. The "Moore neighborhood" also includes the diagonally adjacent cells.

The local transition rule: Like the topology, time is also dealt with discretely. With each successive time step, a set of rules is enforced for all cells. These rules tell a cell's state how to change, as a function of its original state and those of the states in its neighborhood. This ingredient carries the information about the internal forces of the system being modeled and is the cause of the evolution of the global configuration. Except in the case of "hybrid" cellular automata, one rule is applied to all cells. Sometimes, there a component of the rule is left to probability, and the model is called "stochastic"; otherwise it is "deterministic".

One of the main advantages of cellular automata models is in the fact that even when these rules are extremely simple representations of the microscopic behavior, their application can yield accurate, complex global behavior. The simplicity of the local rule also allows for fast computation. Spontaneous order on the global scale is often observed as a result of these local rules, a phenomenon known as “emergence” or “self-organization”.

The most serious limitation of any CA model is in the discreteness of both space and time. Choice of cell size and time step length can immediately restrict other possible values of importance to the system being simulated. This discreteness also leads to statistical noise requiring systematic averaging processes, and limited flexibility in adjusting parameters of a rule to accommodate a wider range of physical situations.

The achievements and limitations of the so called “lattice gas automata” are a good example of cellular automata’s advantages and disadvantages. In 1986, it was shown that much of the fluid behavior in a wind tunnel could be duplicated by a fully discrete computer model. It was surprising that despite the inherent discreteness, the model produced behavior consistent with the Navier-Stokes equation for hydrodynamics. As usual, the approach was expected to give results much more efficiently than other numerical methods. However, limitations due to the inflexibility of the cellular automata rule confined the model to simulate high viscosity fluids and could not compute high Reynolds flows. Additionally, the grainy spatial resolution made it impossible to study fully developed turbulence, unless the scale was increased to the point of loosing any advantage in efficiency over other methods on even the fastest computers of the time (Chopard, 1998).

1.3. AN EXAMPLE

In 1970, John Conway made popular the concept of cellular automata (CA) computing with his program, “The Game of life”. The program simulates a simple, hypothetical microorganism’s struggle to survive in an environment with contrasts of loneliness and overcrowding. In order to more clearly illustrate the concept of the cellular automaton, this program is an ideal example. In terms of the previously described ingredients:

The environment is a two dimensional lattice of rectangular cells.
The local state can be either 0 (empty) or 1 (occupied).
The relevant neighborhood is the Moore neighborhood (all 8 adjacent cells).
The rule fills an empty cell if exactly 3 occupied cells are found in its neighborhood (spontaneous life), and empties an occupied cell if more than 3 (death from overcrowding) or less than 2 (death from loneliness) are found.

Though the initial condition is a lattice of cells randomly filled at some given overall density, the CA model is considered deterministic, as the transition rule contains no randomness. From this disorder, an emergence of sorts can be seen within a few time steps, as various ordered patterns of "live" cells are formed. Of the stable patterns that can be seen, some seem to swim or rotate, some are simple and stationary, and some gradually grow through a consistent and elaborate series of changes before settling on a rotating or stationary pattern. Some patterns evolve elaborately only to completely disappear in the end. A sample picture of two global configurations are given in Figure 1.1. Our version of this program can be found in Appendix A.

Figure 1.1. A sample picture of the results of “Game of Life” after 50 time steps with a starting density of %10

2. TRAFFIC CA MODELS

As mentioned before, the goal of CA modeling is to obtain realistic large-scale behavior by applying a simplified model of internal dynamics. Traffic cellular automata (TCA) models, then, fit in amongst the previously mentioned microscopic models of traffic flow. One key difference is that the computation in the TCA's remains entirely on the microscopic level. In the analytical, microscopic models, particle behavior is addressed in order to build other formulas that describe macroscopic behavior. In TCA's, there are no macroscopic formulas like P(x,v,t) to be dealt with. The cells themselves do all the calculations, and the user is left just with the task of analyzing the results. This not only a matter of convenience in computation. The lack of macroscopic formulas make it possible to use irregular boundary conditions for which other methods might not be able to provide solutions. This makes TCA's useful not only for understanding the dynamics of traffic flow, but also as a virtual laboratory for traffic experiments (which are difficult to perform without upsetting many people).

The task is then to translate a vehicle with a thinking driver into the format of discrete cells, integer states, and a generic local rule. When a real vehicle's possible behavior is observed closely, this approach does not seem too inappropriate. In a one lane road, a car may either speed up or slow down. A slower car in front may force it to decelerate, in order to avoid collision and a speed limit may prevent it from accelerating more. At higher speeds, drivers are inclined to leave a gap between them and the car in front in case the car in front suddenly decelerates. As for driver temperament and nerves, probability is often applied to one or more components of the local rule to compensate. In effect, intelligence and emotions of thousands of people can, with reasonable accuracy, amount to a fixed probability of lightly pressing on the brake without external motivation.

Other than risk of collision and speed limit, there are a number of other internal influences on driver behavior, and corresponding TCA models that account for them. A driver's looking far ahead into traffic to anticipate his predecessor's behavior, and use of brake lights and the horn are examples of such previously studied influences. Less psychological influences on internal behavior such as the affect of larger vehicles such as buses and trailers has also been studied. External influences on traffic such as traffic lights, on and off ramps, and merging lanes have been studied to some extent.

The model that spawned most of these studies is the NaSch TCA, proposed by Nagel and Schreckenberg in 1992 (Nagel, 1992).

2.1. THE NASCH MODEL

The NaSch TCA is a minimal model in that all the components of its characteristic local rule are necessary (in the order applied) to reproduce the basic features of realistic traffic. In terms of generic CA ingredients (Section 1.2.), the components of the model are as follows:

The physical environment is a one dimensional array of cells.

A cell may be occupied or unoccupied. Occupied cells may have integer state values between 0 and a given maximum value that represent the forward speed of a vehicle.

The neighborhood of influence on an occupied cell is the space between it and the preceding occupied cell. In this way, drivers may look forward only as far as the preceding car, and not backwards. However, they have no knowledge of the speed of the preceding car.

The local rule, which is applied to occupied cells, consists of four parts, which are applied in the following order:

1. Acceleration. If the nth vehicle (occupied cell) has a velocity less than the given speed limit, its speed is increased by one.

$$v_n \rightarrow \min(v_n + 1, v_{\max}) \qquad (2.1)$$

where v_n is the velocity of the nth car, and v_{max} is the speed limit.

2. Deceleration due to other cars. If the car in front is at a distance within the range of the next movement of the nth car, its speed is decreased accordingly.

$$v_n \rightarrow \min(v_n, d_n - 1) \tag{2.2}$$

where d_n is the gap between the nth car and its predecessor (x_{n+1} - x_n).

3. Randomization. If the nth car's speed is greater than zero, it may decrease its speed by one with some probability p.

$$v_n \rightarrow \max(v_n - 1, 0) \tag{2.3}$$

4. Movement. Once the new velocities are determined by the above rules, the cars are moved simultaneously.

$$x_n \rightarrow x_n + v_n \tag{2.4}$$

Surprisingly, these few rules immediately produce much of the observed behavior of real traffic, including the spontaneous jamming mentioned in Section 1.1. The randomization step is the cause of both random deceleration and overreaction in breaking. When one car unexpectedly decelerates, it is likely to collect a few cars behind it. As the number of closely following cars increases, so does the probability that one of them will also unexpectedly decelerate further, thus causing cars to collect faster. This chain reaction is responsible for the spontaneous jamming first mentioned in Section 1.1. The phenomena can be clearly seen in space-time diagrams such as Figure 2.1.

Another achievement of the NaSch model is in producing a qualitatively accurate "fundamental diagram" of density versus flow (flux). Where most analytical models rely on this density-flow relationship and import it from empirical data, the parameters defined within the NaSch model produce it on their own. Figure 2.2 is an example of such a fundamental diagram, taken from empirical data of single lane highway traffic. In it one can clearly identify two distinct phases. At lower densities, the flow increases linearly with the density to a maximum. This region is called “free flow”. Much like a gas phase, particle-particle interaction plays little role in the behavior. Vehicles move unrestricted by the movement of other vehicles and therefore move at the maximum speed. The slope of the line in this region is just that speed limit. After a particular density, the flow falls sharply and the behavior of the traffic changes significantly. Now the flow is inversely proportional to the density. This region is referred to as the “congested regime”. The fundamental diagram produced by the NaSch TCA is shown in Figure 2.3 for different densities.

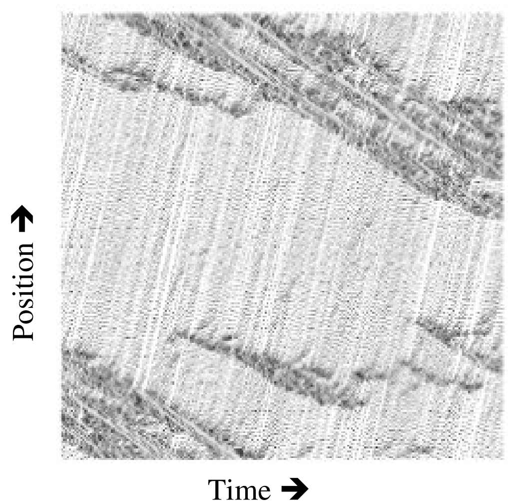

Figure 2.1. A typical space-time diagram for the NaSch TCA. Every vertical slice of the diagram shows the positions of cars on a length of road. As time passes from left to right, the cars (dots) progress along the road. Random deceleration causes spontaneous jams which can be seen to propagate against the flow of traffic

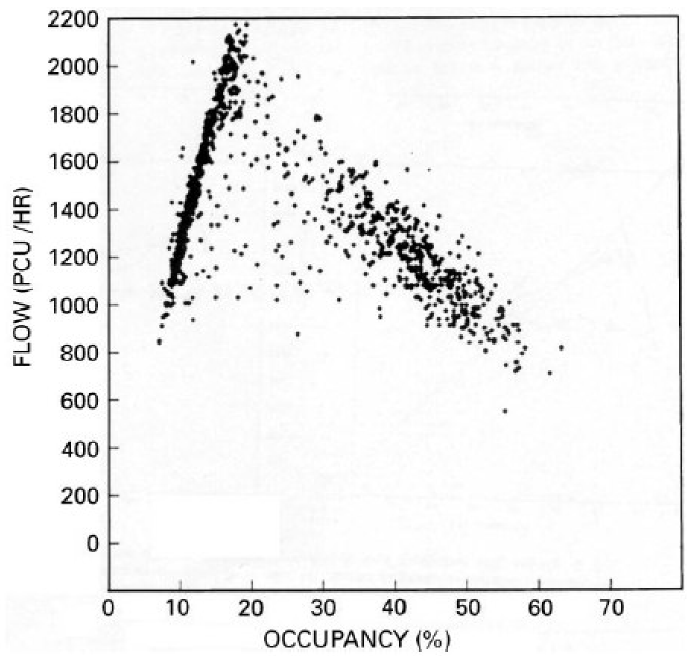

Figure 2.2. An example fundamental diagram taken from empirical data of cars on a Canadian highway. Two states can be distinguished: free flow on the left and congested flow on the right

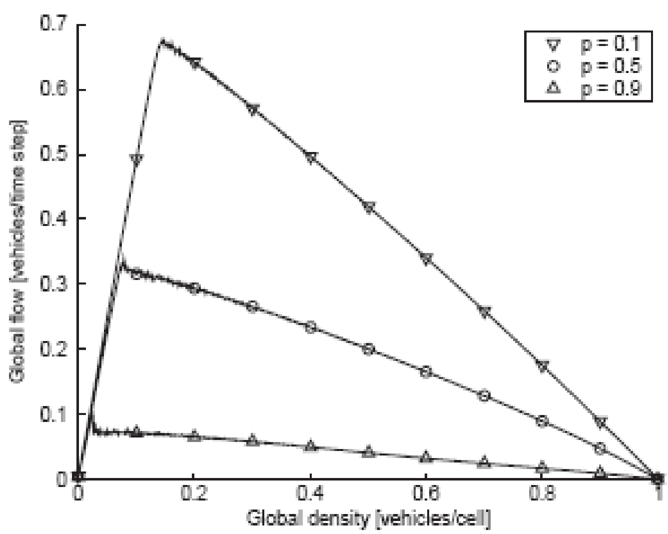

Figure 2.3. A fundamental diagram produced from a stochastic NaSch TCA for single lane traffic. The program was run using different randomization probabilities (p)

2.2. CONVERSION OF QUANTITIES

In order to bridge the gap between simulation and real-world conditions, special attention must be paid to scaling. In TCA modeling, spatial and temporal quantities are discrete, but in the real world they are not. Upon choosing the conversion factor for any discrete quantity, the remaining quantities are restricted, so it must be done carefully. Because the size of cars, on average, is fixed, the scaling of cells could be considered first. In the NaSch model, exactly one car is allowed per one cell, and all cars are considered of equal length. In stopped traffic, one kilometer of a real road can hold approximately 133 vehicles. Therefore, the cell length is chosen as 7.5 meters.

After determining a spatial scaling, choosing the time step length becomes tricky because the velocity and temporal resolution are now coupled. The whole of the NaSch model relies on these quantities to mutually fit with real-life conditions (though this was probably in mind when the local rule was written in the first place). For instance, if one decides that for realistic road traffic, 0.1 second resolution is necessary to describe traffic dynamics, the allowed velocities become unreasonable. In that case, the lowest nonzero velocity would be 270 km/h (see Formula 2.4.). Luckily, without overlooking any important dynamics, reasonable values can be chosen. Normally, the time step is given as 1.2 seconds, producing multiples of 22.5 km/h for allowed velocities. In this case, the typical speed limit for a highway in CA terms is 5 cells/timestep, corresponding to 112.5 km/h (Maerivoet, 2005). Accelerations described in Formulas 2.1 and 2.3 then are fixed as $\pm$ 18.75km/h/s.

3. OUR MODEL

In writing our simulation code, the NaSch model was to be used as a foundation upon which to build a realistic multilane highway including on and off ramps, as well as an auxiliary lane, based vaguely on the Kavacik highway intersection in Istanbul. The result differs in some ways, though the underlying logic is basically the same. Apart from the specialized components in the code that account for the specific road conditions (which may be turned off), some more fundamental differences exist. They will be discussed in more detail in Section 3.4. Now we will describe the form of the basic program with four lanes and no obstructions.

3.1. THE SIMPLE FOUR LANE HIGHWAY

3.1.1. Description

The lanes are defined by an 8-dimensional array of adjustable length. Each pair of columns represents one lane. The cells represented by the left column of each lane may contain a 0, meaning there is no car on the corresponding portion of the road, and a 1, meaning there is a car in that position. The right column is used for storing the integer velocity values of cars that happen to be found at the same position in the left column. The speed limit is 5.

The first step scatters cars randomly across all the lanes and over the length of road with some given total density ρ. Each car is given maximum speed. This is the initial condition. As the program runs, cars may leave the section of road represented by the array, and new cars are placed in the beginning of the road with the same density used in the first step.

The local rule is applied in the following way. An identical "update" array is created to record new positions and velocities. The first rule is applied to the cars sequentially from the end of the road backwards, lane by lane. It finds a car, increases its speed by one (if it is moving less than the speed limit), finds the distance to the updated car in front and moves it to either its distance plus its speed, or to the position just behind the car in front (if within that range) by copying this new state to the updated array. In case the car in front restricted its motion, the new speed is given as the distance it was allowed to travel. At the same time, a probability "r" is given for overreaction such that the speed will be additionally decreased by one. One may notice this "first" rule is composed of several parts:

Acceleration:

$$v_n' = \min(v_n + 1, v_{\max}) \tag{3.1}$$

Movement:

$$x_n' = \min(x_n + v_n', x_{n+1}' - 1) \tag{3.2}$$

Deceleration (if movement influenced car in front):

$$v_n' \to x_{n+1}' - x_n' - 1 \tag{3.3}$$

Overreaction (with probability r, which is proportional to severity of near-collision):

$$v_n' \to x_{n+1}' - x_n' - 2 \tag{3.4}$$

Correction for safe headway (if a car's speed is 5 and there is another car within 2 cells in front, or 4 and there is another car directly in front).

$$v_n' \to 3 \tag{3.5}$$

The second step finds all the cars in the updated array and allows for spontaneous deceleration with probability "s". This is the same as the "randomization" step in the NaSch model.

$$v_n' \rightarrow \max(v_n' - 1, 0) \tag{3.6}$$

After all cars have been updated according to the cars in front, they are allowed to look sideways and decide if they would rather be in another lane for the next time step. This is done by once again scanning the lanes from front to back, this time finding cars which would, in the next application of Formula 3.2., be forced to slow down by the cars in front. For these cars, the distance to the next car in the adjacent lane(s) is calculated. At some probability, the car is allowed to move horizontally into the lane with the larger headway. This adjustment is made in the update array.

$$x_n' \rightarrow y_n' \ or \ z_n' \tag{3.7}$$

The notation used is a bit different than in Section 2.1. because some steps involve copying information to the new array and some involve changing values within the new array. The primed terms are values in the updated array, equal sign refers to writing into the updated array from the original array, and the arrow refers to changing values within one array.

After the rule has been applied, the original array is replaced by the update array and the update array is erased. The next time step applies the same rules again.

3.1.2. Single Lane Condition

Output of our program includes positions of all vehicles for each time step as well as time-averaged flow and density measurements for detectors placed along the extent of the virtual road. The former is used to plot space-time diagrams like the one we showed in

Figure 2.1. The latter is used both to produce fundamental diagrams like the one in Figure 2.2. and to see how the density and flow varies along the course of the road. The

To see how our model compares with previous one lane models of highway traffic, we can turn off the lane changing rule; in affect make four single lane roads. Figure 3.1 is a sample space-time diagram of such a single lane road produced by our model.

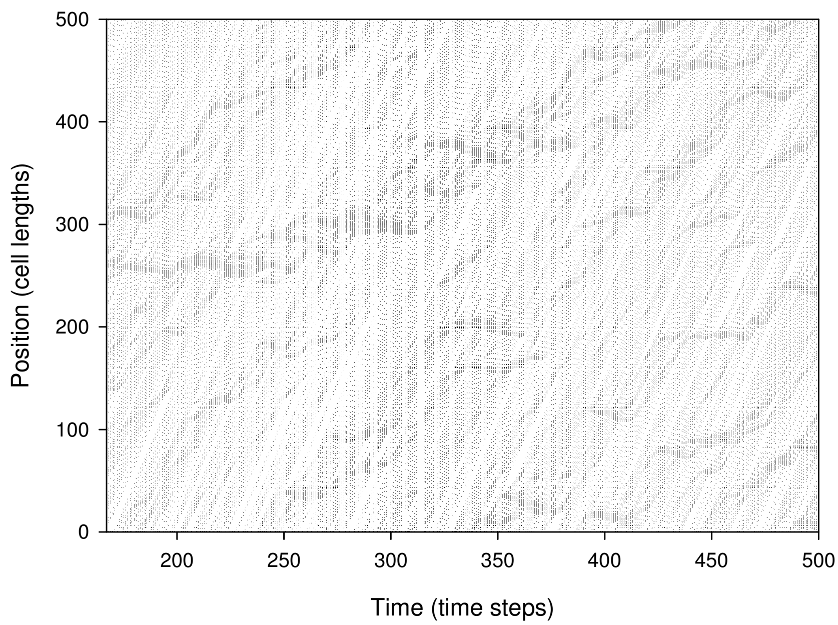

Figure 3.1. Space-time diagram for a single lane in our model with density $\rho = 25$, overreaction constant $r = 80$, slowdown probability $s = 5$ shown for timesteps 167 through 500

We can clearly see the formation of spontaneous jams similar to those produced by the NaSch model. However, our model seems to give more transient, small jams, whereas the NaSch model produces more large, long term jams.

The resulting fundamental diagram for the same conditions is shown in Figure 3.2. Detectors are 10 cells long and time averaging is done over a 10 time step duration. This, in effect, analyzes each 10 by 10 square of the space-time diagram.

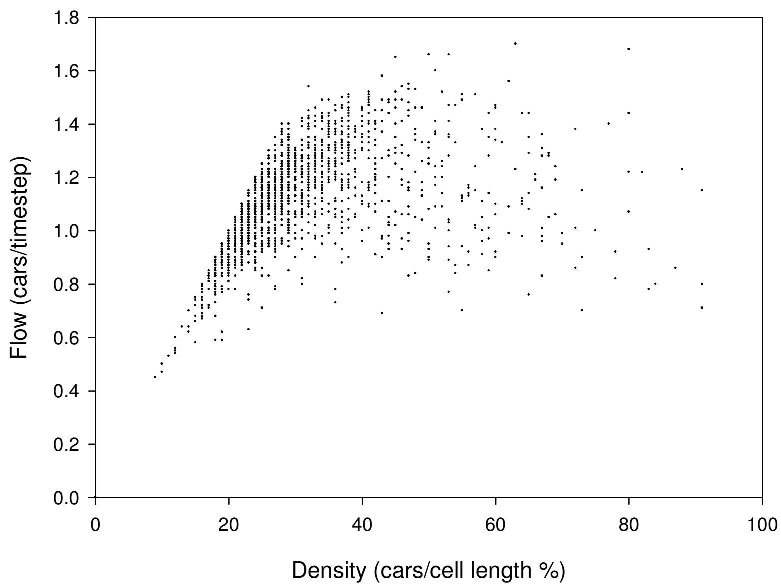

Figure 3.2. Fundamental diagram for one-lane road using our model, with $\rho = 25$, $r = 80$, and $s = 5$

The averaging process creates a fuzziness which somewhat obscures the features of the diagram. For low resolutions (i.e. large detector size and large interval for time averaging), it is difficult to see the distinct phases in the diagram because jams move in and out of the window both spatially and temporally. High window resolutions (small detectors and short term time averaging) produce graininess and much more statistical noise, as individual cars significantly affect the calculations of flow and density.

Nevertheless, one can make out the free flow condition increasing to approximately 1.6 cars/timestep at about 40% density, after which the flow begins to decrease with increasing local density. There seems to be a linear decrease in the flow in this region, which approaches about 0.5 cars/timestep at 100% density.

This minimum of 0.5 cars/timestep is significantly different from the 0 flow minimum from the NaSch model. That is, in our model, even in the densest possible traffic jams, half of the cars are moving at minimum speed. This apparently unrealistic result stems from our sequential application of the movement rule. In the case of completely compact jam, the behavior of the NaSch simulation is as follows: After a car's speed is increased from 0 to 1 in the first step, the car sees there is no distance to the car in front, and, before any motion occurs, the speed is set back to zero. The net effect is that all cars in a jam wait their turn to reach the open space ahead before they become the one car per time step that can leave.

In the same bumper-to-bumper jam, our model operates differently. By the time a car's speed has been increased from 0 to 1, all the cars from the front of the jam backwards have been accelerated and moved. No car actually remains totally immobile for the entire duration of one time step, even though before and after the first rule has been applied it may have a 0 speed state. This can be interpreted as an extreme case of stop-and-go traffic (not to be confused with more obvious stop-and-go traffic which we will see later), where waves of motion propagate backwards and briefly move cars forward. Some have said, however, that this scenario can violate causality, as this information is transmitted through the entire lattice within a fraction of one time step, regardless of the size of the lattice (Maerivoet, 2005). Whether or not this information travels infinitely fast is arguable.

Though this is clearly a practical and logical problem for us, there may be some value to the depiction of this "creeping" traffic in spontaneous jams at maximum density. In real traffic, information about motion at the front of a large jam is related backward in some way. Rarely does one sitting in traffic remain perfectly still until a moment when, all of a sudden, the car in front accelerates consistently away from the jam. The "go" part of stop-and-go definitely shouldn't be at speeds like 22.5 km/h, as is forced to take place because of the discreteness of velocity in our model, and the information shouldn't be transmitted without any response time factored in, but this motion, it seems, could be a part of a realistic model.

3.1.3. Effects of Changing Lanes

Introducing a new degree of freedom for the cars to use to their advantage in trying to move as fast as possible down a road, one would expect, should increase the total flow. When this degree of freedom can also be seen as a source of inconsistencies in an already unstable system, this answer may seem doubtful. A car changing its lane to avoid one jam may be the cause of another. Under the same conditions as the "single lane" road analyzed above, the program can allow for lane changing in all four lanes as described in the third step. The analogous space-time diagram and fundamental diagram for the second lane is shown in Figure 3.3. and Figure 3.4., respectively.

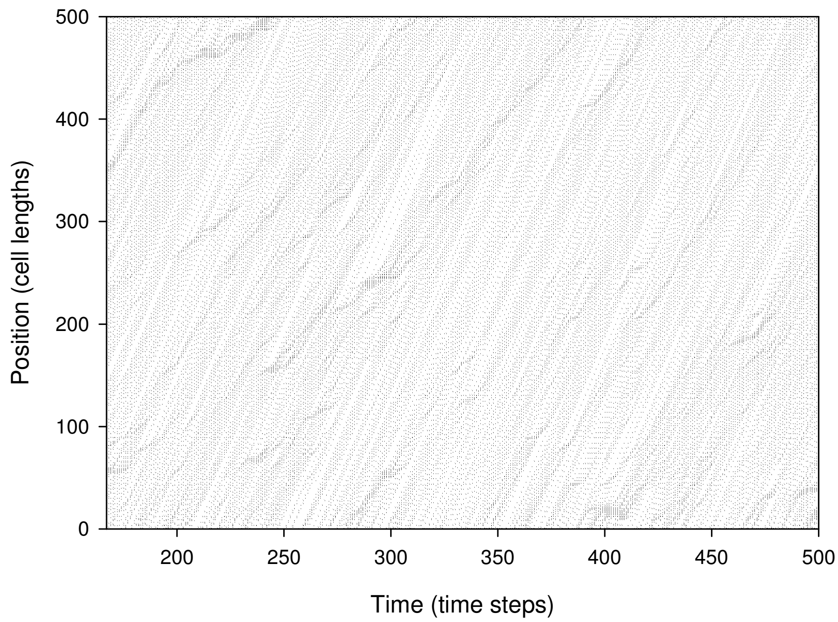

Figure 3.3. Space-time diagram of the second lane of a four lane highway. Allowing cars to change lanes relieves some congestion caused by spontaneous jams. $\rho = 25$, $r = 80$, and $s = 5$

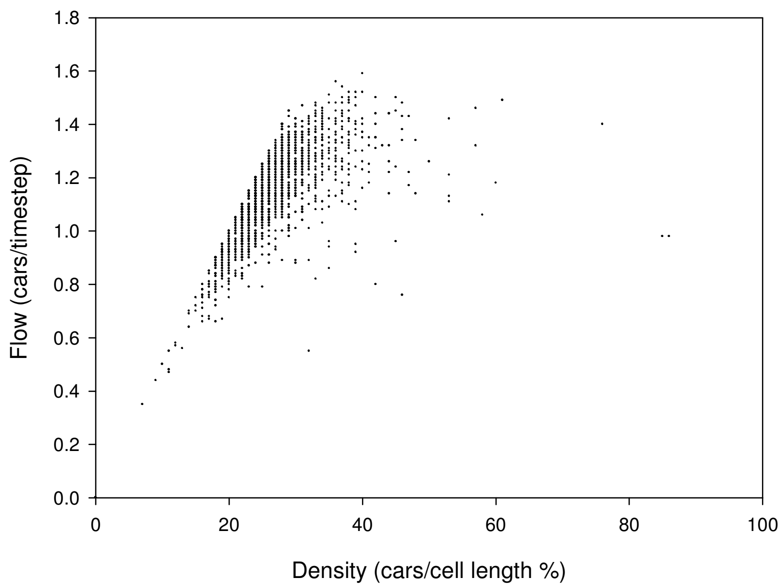

Figure 3.4. Fundamental Diagram of second lane in four lane highway. ρ = 25, r = 80, and s = 5

When comparing Figure 3.1. with Figure 3.3. and Figure 3.2. with Figure 3.4., it is clear that allowing cars to change lanes in our model is advantageous to the total flow. The maximum flow of the free flow regime in the fundamental diagram are approximately the same. The only difference is that the traffic spends little time in the congested regime.

3.2. OFF RAMP ON A FOUR LANE HIGHWAY

To add an off ramp to a one lane road, one may simply determine a probability for cars passing a point on the road to disappear (perhaps decelerate before doing so). That probability will determine the percentage of cars on the road that wanted to leave anyway. Traffic on the main road could be affected if the ramp is short and exiting cars need to decelerate before getting out of the way. Such a disturbance can induce jams.

On a four lane road, the process is tremendously more complicated. Using the same method by applying an exiting probability for cars on the far right lane makes the huge assumption that all the cars who were planning on exiting managed to somehow find themselves in the far right lane long before the simulation started. A typical driver won't move into the right lane kilometers in advance, and many find it difficult to do so until it is almost too late.

3.2.1. Description

Taking into consideration that drivers normally have a travel plan in mind when they set out on the road, we decided to distinguish between onward moving cars and cars planning to leave on the off ramp. When adding cars to the road from the top, now there are two types: those identified by "1", as before, and those identified by "2". A predetermined fraction of the total density is declared type 2 as the cars are placed on the road. For the stretch of road far from the off ramp, the same rules are applied to type 1 and type 2 cars. In the region closer to the off ramp, however, some specialized rules have to be applied to type 2 cars. Apart from the actual extraction of type 2 cars at the off ramp, these rules were built in to the steps described in 3.1.1. Most of the rules are used to force the cars into the right lane on time, but one allows for casual lane change in the right direction.

This more casual rule takes effect at a point that could be thought of as the position of a sign warning drivers that the exit is approaching in x meters. This sign affects the type 2 cars' decisions in lane changing. After seeing the sign, type 2 cars move right whenever the opportunity arises. Even if the option is available by the usual lane change rule, they may no longer move left.

Sometimes, by circumstance or by unsuitable traffic conditions, cars cannot passively move right in time. To take care of this problem, conditional rules are added to the updating steps:

Type two cars are unaffected by the acceleration step if they are not in the right lane within 15 cell lengths from the off ramp. This prevents them from inching forward past the off ramp instead of waiting for an opening in the adjacent right lane in congested traffic.

After the movement, influenced deceleration, overreaction, and safe headway correction rules are applied in the same fashion as always, the randomization step is adjusted for type 2 cars. The type 2 cars in the right lane 10 cell lengths from the off ramp are forced to move no faster than 3 cells/timestep and those 5 cell lengths away 2 cells/timestep. It is assumed this off ramp turns sharply away and requires those exiting to do so slowly. From the perspective of both driver and simulator, the only fail-safe way of ensuring that a car will be able to change into the far right lane is to make it slow down (and stop, if necessary). The farther left the type 2 car, the sooner this rule must be implemented:

Fourth lane within 12 cell lengths from off ramp:

$$v_n' \rightarrow (ramp - x_n' - 6)/2 \tag{3.8}$$

Third lane within 10 cell lengths from off ramp:

$$v_n' \rightarrow (ramp - x_n' - 4)/2 \tag{3.9}$$

Second lane within 8 cell lengths from off ramp:

$$v_n' \rightarrow (ramp - x_n' - 2)/2 \tag{3.10}$$

The lane change rule must also be adjusted to strictly enforce not missing the turn. Starting 40 cell lengths from the off ramp for the fourth lane, 30 for the third lane, and 20 for the second lane, cars are forced into any empty cell they find to the right. Additionally, to account for empty spaces that may pass by a car waiting to move right, the program “cheats” a bit by automatically placing the car in the present position of that empty space, after calculating that it would have passed it by between time steps.

Finally, a new rule is added which removes type two cars from the right lane if their speed plus position falls within the five cell lengths that make up the off ramp. By simply removing vehicles, our model assumes zero congestion after the exit.

3.2.2. All Cars Turn Limit

To illustrate the behavior of type 2 cars clearly, it is helpful to run the program without any type 1 cars. Even a density as low as 7% can induce jams because all cars are being forced into the same lane.

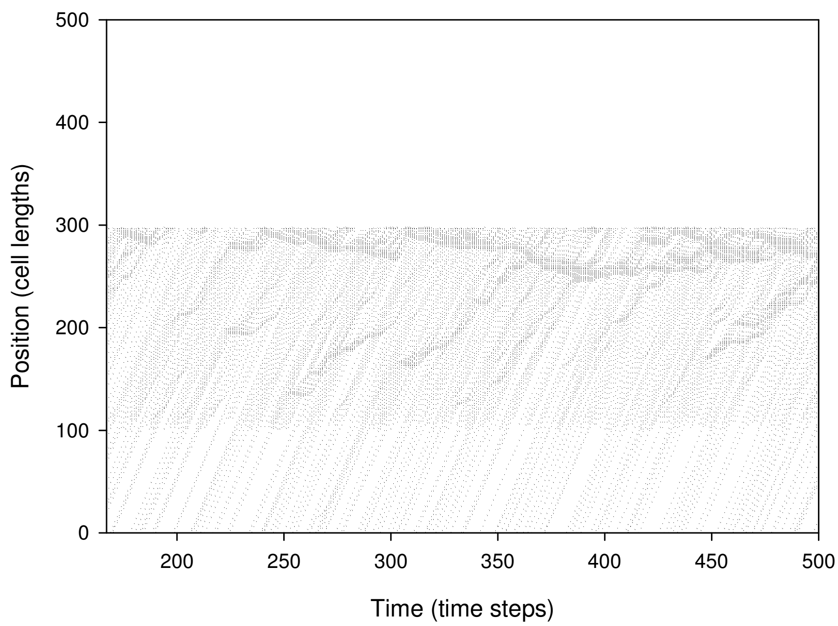

Figure 3.5. Space-time diagram of the first lane of four lane highway with off ramp at 300, exit sign at 100. Density of type 2 cars $\rho_2 = \rho = 7$, $r = 80$, and $s = 5$

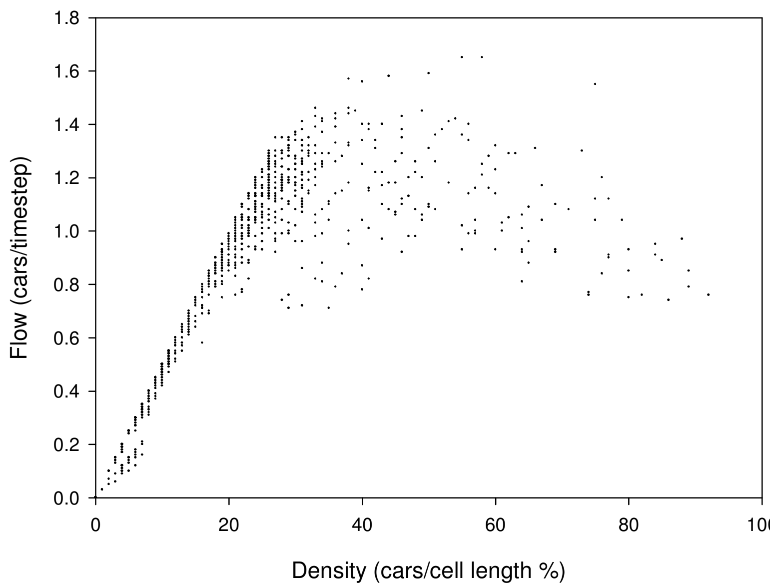

Figure 3.6. Fundamental diagram of the first lane of four lane highway with off ramp at 300, exit sign at 100. Density of type 2 cars $\rho_2 = \rho = 7$, r = 80, and s = 5

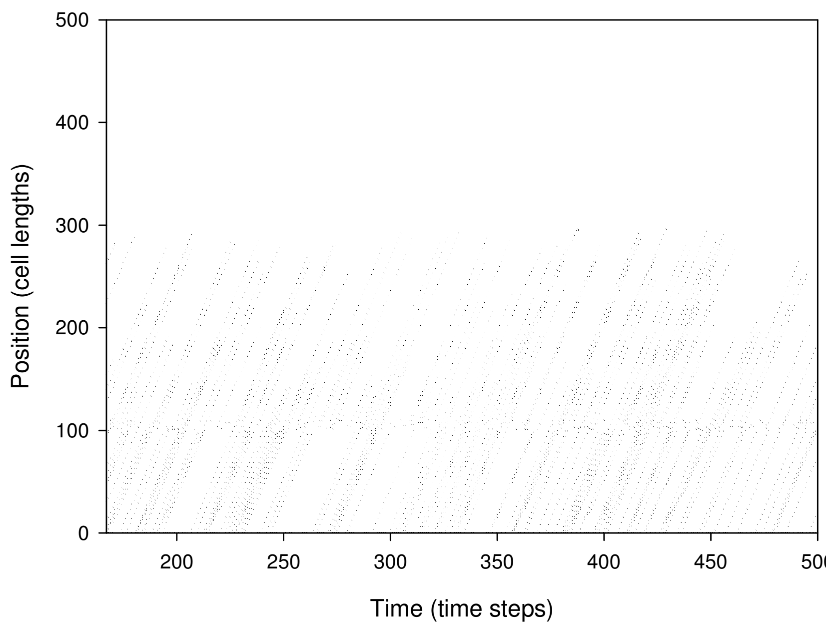

Figure 3.7. Space-time diagram of the second lane of four lane highway with off ramp at 300, exit sign at 100. Density of type 2 cars $\rho_2 = \rho = 7$, r = 80, and s = 5

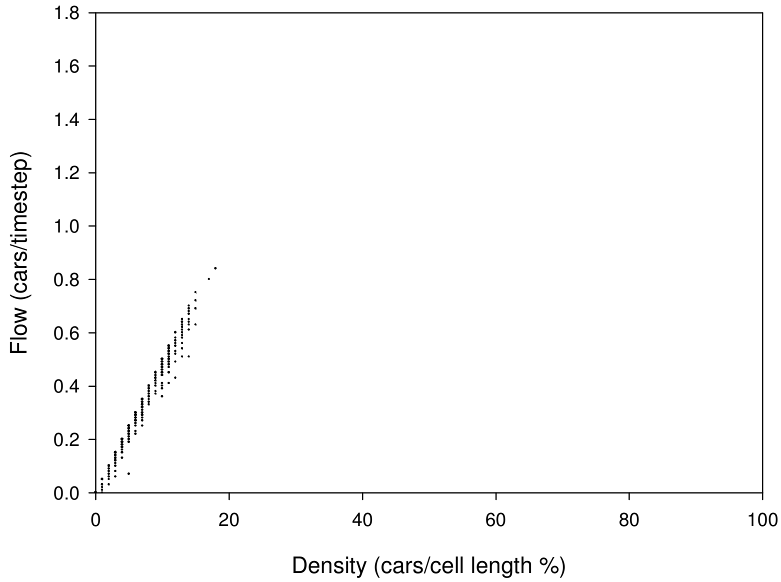

Figure 3.8. Fundamental diagram of the second lane of four lane highway with off ramp at 300, exit sign at 100. Density of type 2 cars $\rho_2 = \rho = 7$, r = 80, and s = 5

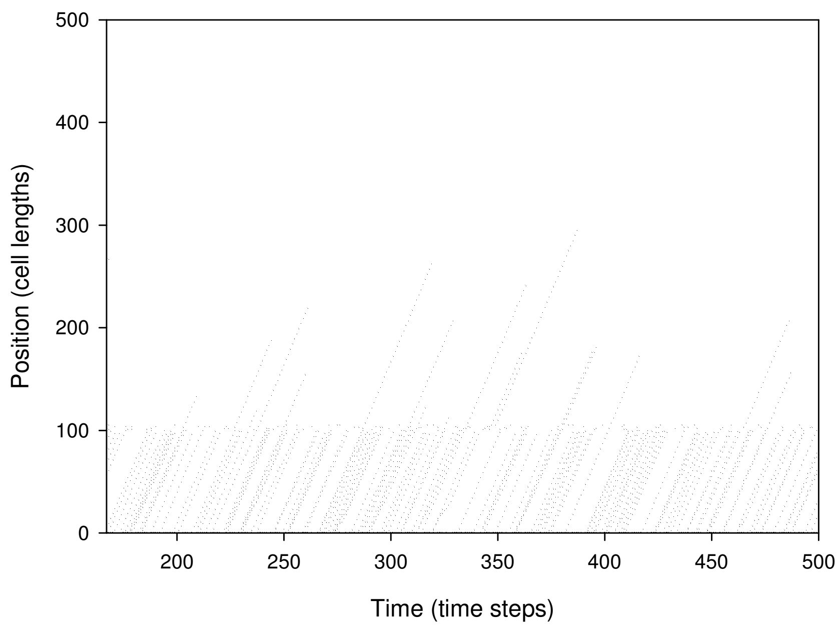

Figure 3.9. Space-time diagram of the third lane of four lane highway with off ramp at 300, exit sign at 100. Density of type 2 cars $\rho_2 = \rho = 7$, r = 80, and s = 5

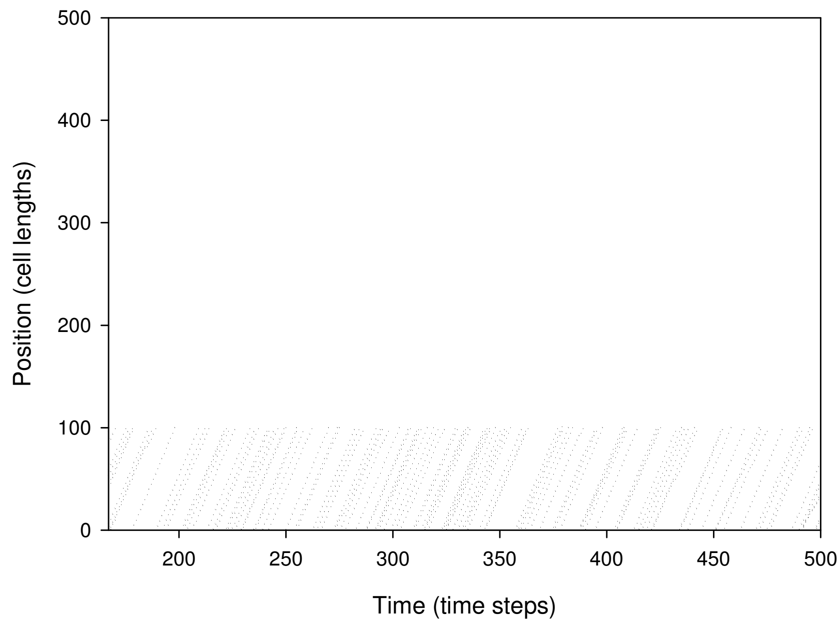

Figure 3.10. Space-time diagram of the fourth lane of four lane highway with off ramp at 300, exit sign at 100. Density of type 2 cars $\rho_2 = \rho = 7$, r = 80, and s = 5

From these graphs one can see that no one missed their exit. Cars were able to evacuate the fourth lane within only a couple time steps of passing the sign, and only a few hesitated in the third. From the second lane, only two cars were forced to slow down slightly in order to merge into the right lane. Due to the cars moving right, the density of the right lane had increased to nearly 28% shortly after the sign position. At the same time, no cars were allowed to move from the right lane, in effect making the lane a single lane road (except that cars are being added from the sides as well) with density great enough to produce spontaneous jams. The fundamental diagram illustrates this fact, as the second regime of inversely proportional flow and density is found, shortly after the maximum flow of about 1.5 cars/timestep. The fundamental diagram of the second lane is also given as an example of free flow far from the maximum point.

3.2.3. Effect of Exit Sign Position

In normal circumstances, only a portion of the cars on a road will plan to exit. The behavior of those cars has been shown. Now it will be illuminating to see how they affect a road composed of onward moving traffic as well. At the same time, the example used here can be used to form a comparison between traffic that has been informed of the exit well in advance and that which has not.

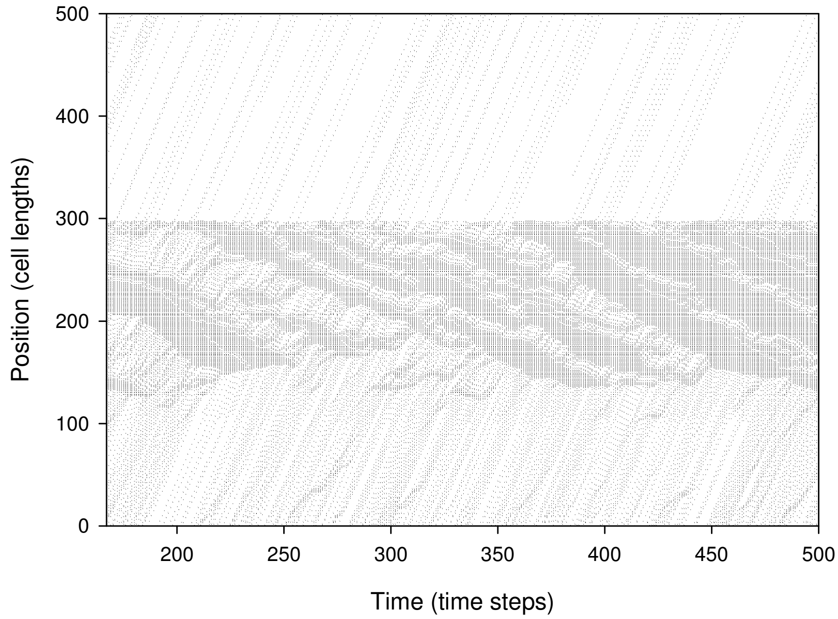

Figure 3.11. Space-time diagram of the first lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

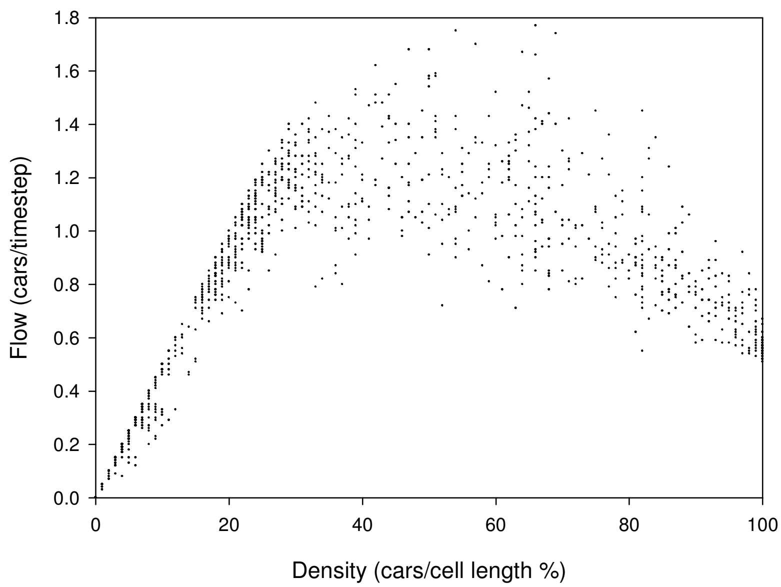

Figure 3.12. Fundamental diagram of the first lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

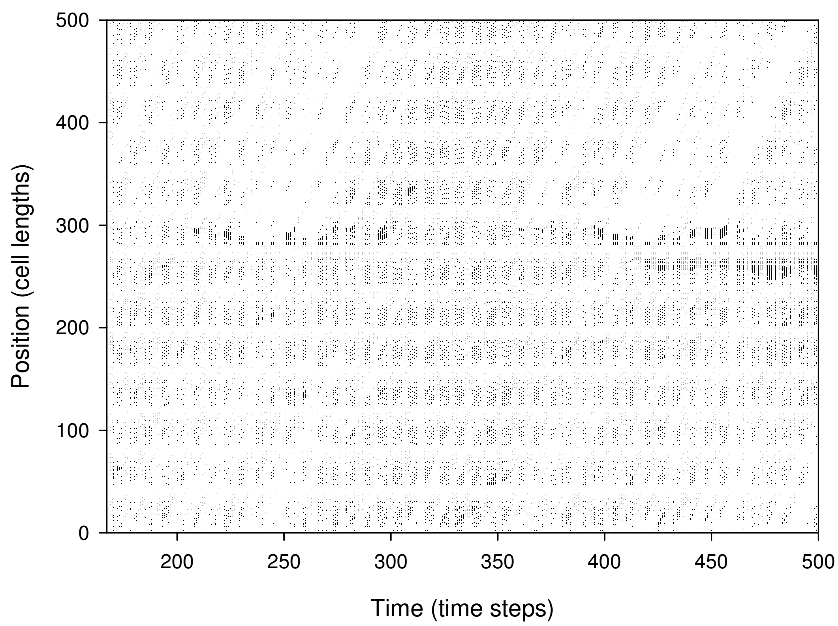

Figure 3.13. Space-time diagram of the second lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

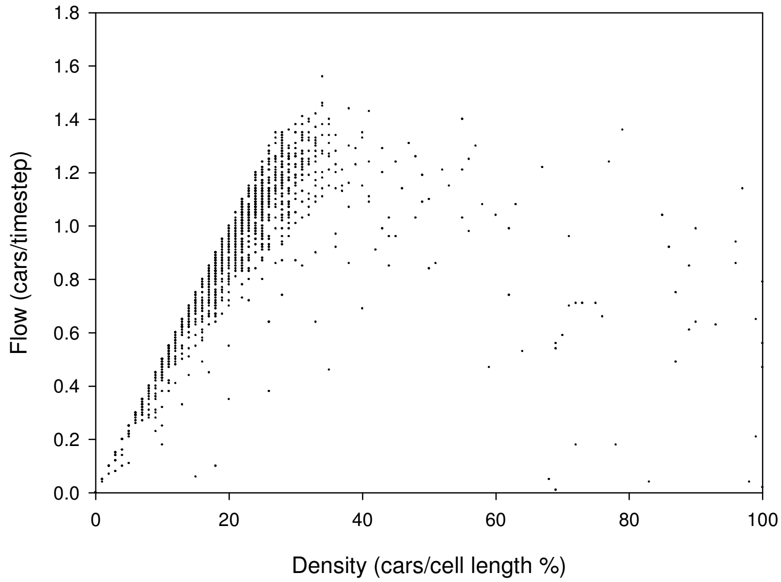

Figure 3.14. Fundamental diagram of the second lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

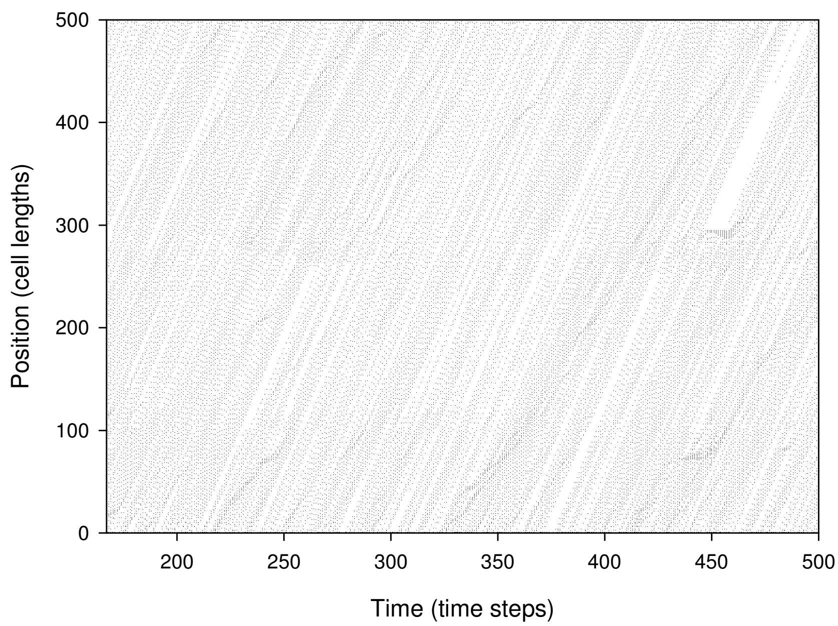

Figure 3.15. Space-time diagram of the third lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

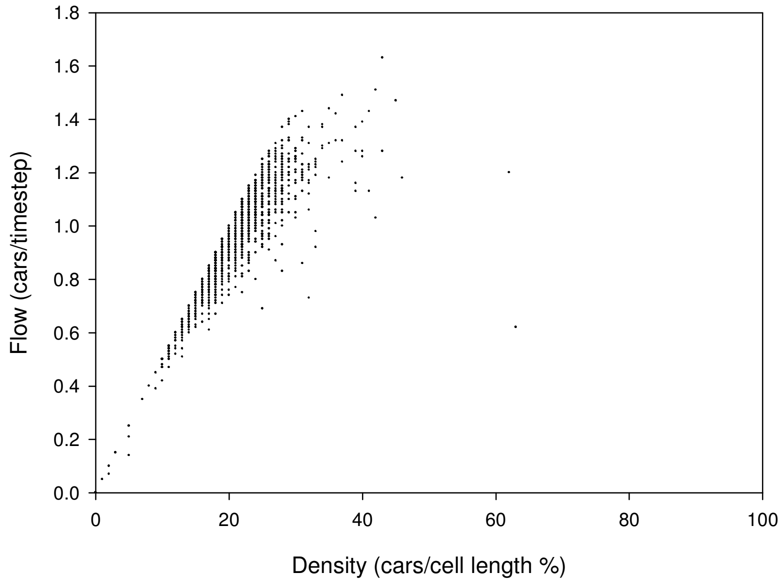

Figure 3.16. Fundamental diagram of the third lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

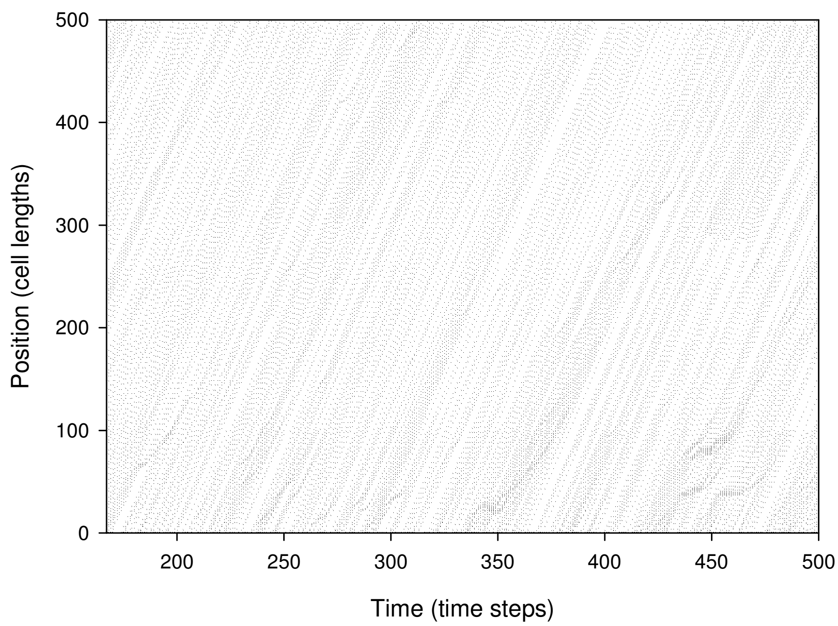

Figure 3.17. Space-time diagram of the fourth lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

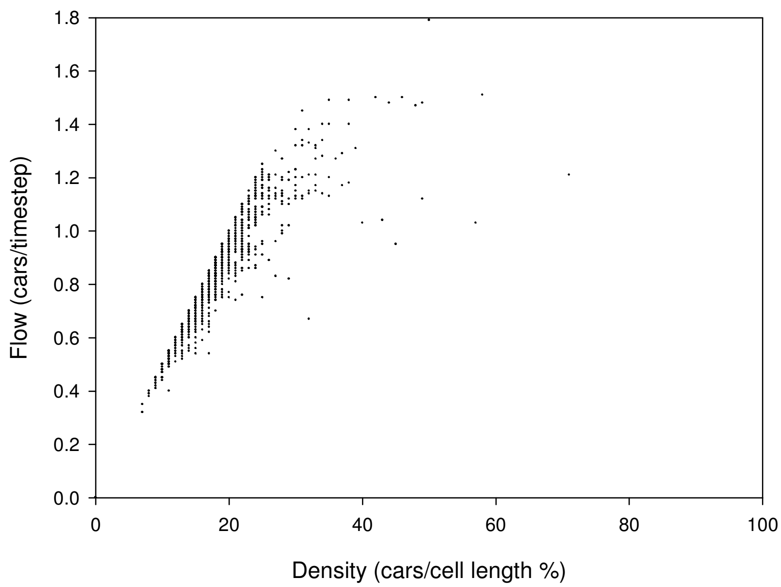

Figure 3.18. Fundamental diagram of the fourth lane of four lane highway with off ramp at 300, exit sign at 100. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

Once again, one can see from the fundamental diagram of the first lane the creeping maximum density flow. In the same lane, however, we can see from the space-time diagram what look like cracks in the large jam preceding the off ramp. These show that when cars are waiting in dense traffic, they are occasionally allowed to move faster before being stopped again. This is the more familiar version of the stop-and-go traffic. From the width and angle of these “cracks”, we can see how many vehicles are involved in one of these short periods of motion and how quickly the motion is translated backwards.

Another interesting feature is that, in the fundamental diagram for the second lane, the flow values are very scattered after the typical maximum flow value. The near-zero values are consistent with the fact that the cars are forced to wait if they are unable to merge right before the off ramp. The high values are more of a mystery.

Another feature of interest is in the first lane’s space-time diagram. The cars continuing in this lane after the off ramp are much more scarce. In other words, type 1 cars are actually avoiding the right lane before the off ramp. This is a common phenomenon in real life, but not one that was programmed explicitly.

To see the effect of what we call the exit sign, the program is run with this position moved from 200 cell lengths before the off ramp to 100 cell lengths before the off ramp.

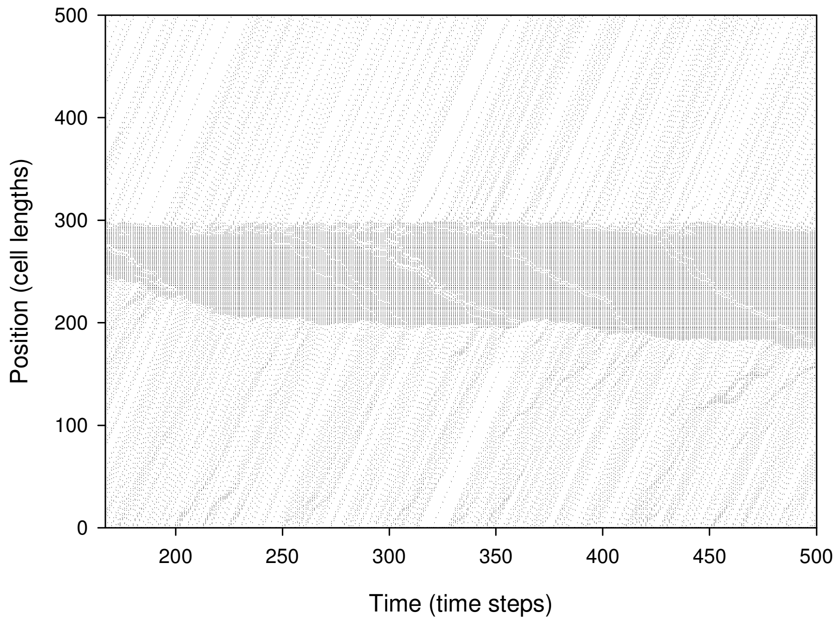

Figure 3.19. Space-time diagram of the first lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

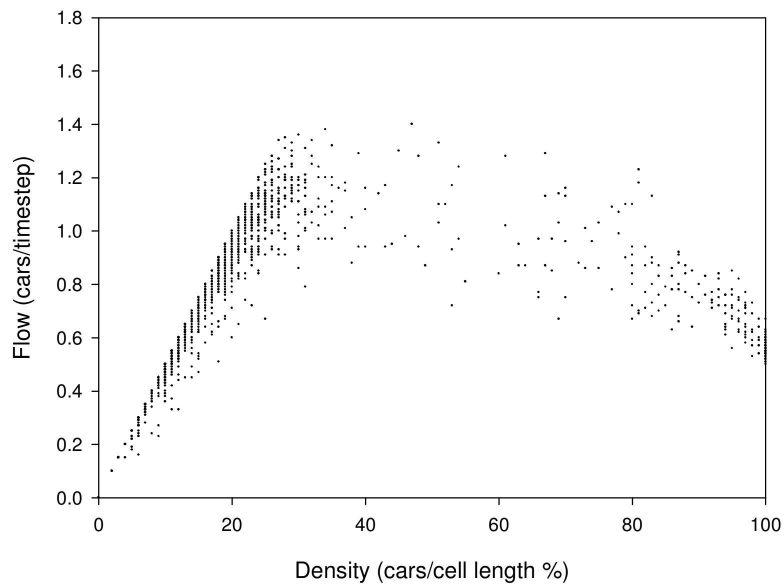

Figure 3.20. Fundamental diagram of the first lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

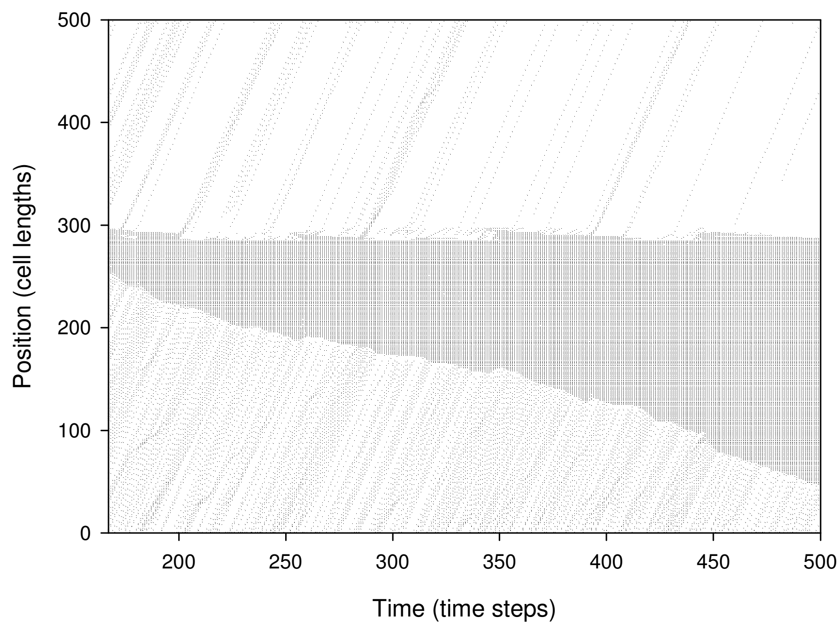

Figure 3.21. Space-time diagram of the second lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

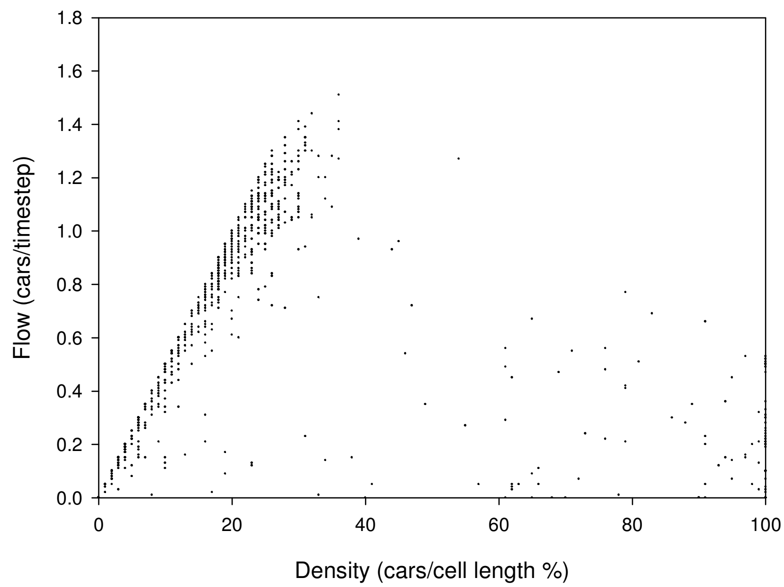

Figure 3.22. Fundamental diagram of the second lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

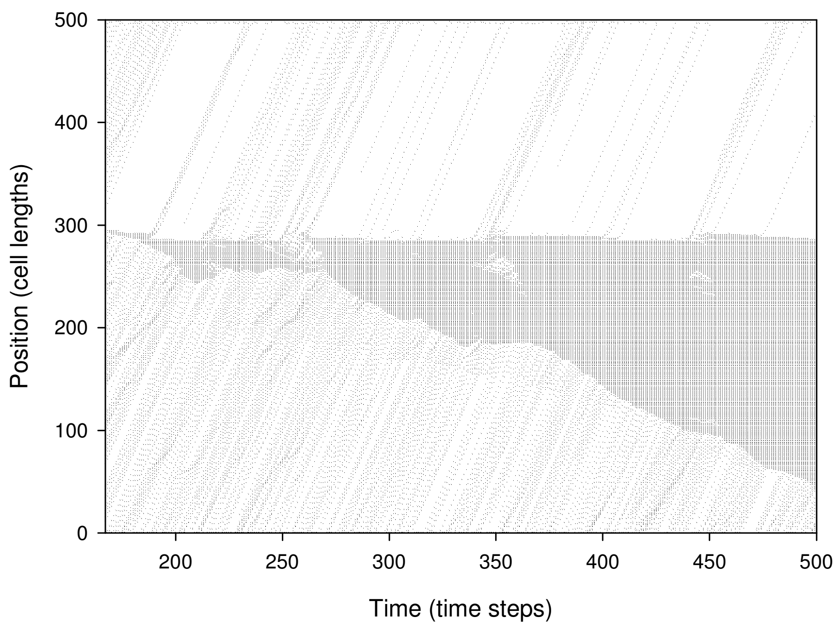

Figure 3.23. Space-time diagram of the third lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

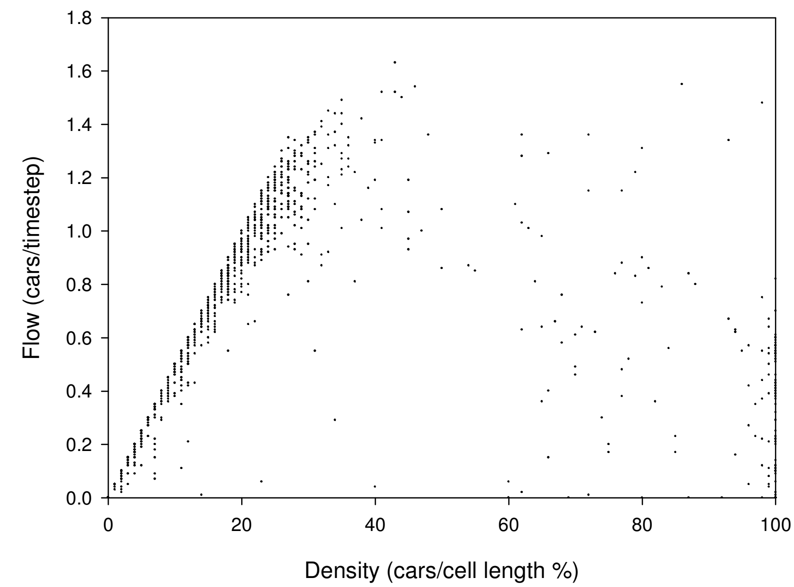

Figure 3.24. Fundamental diagram of the third lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

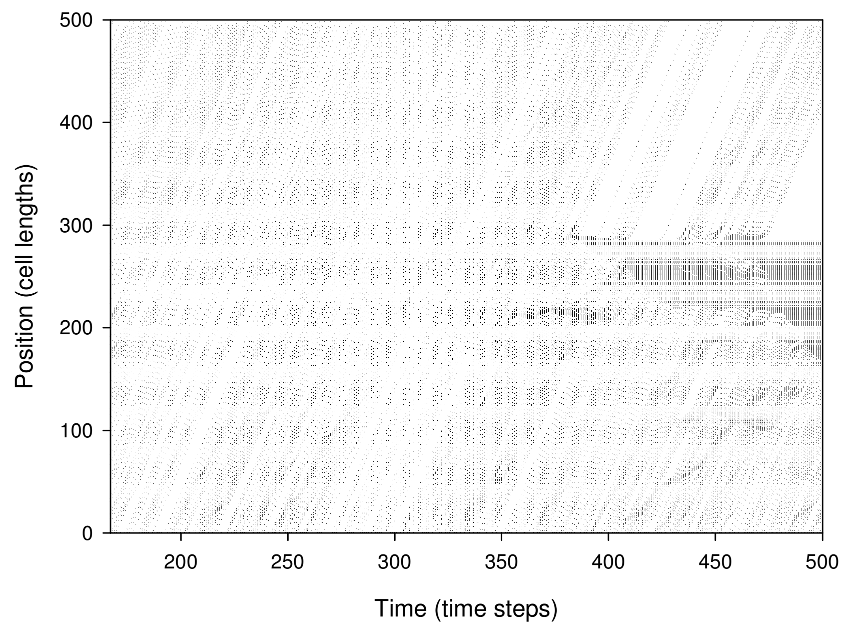

Figure 3.25. Space-time diagram of the fourth lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

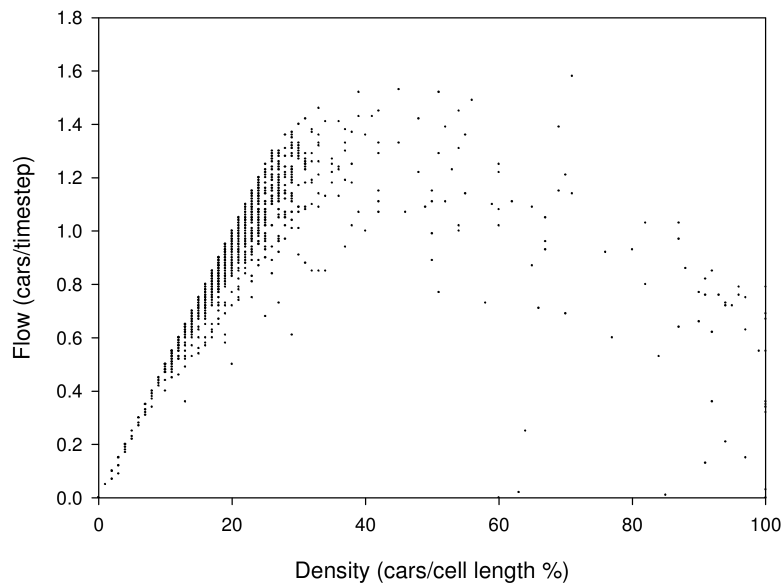

Figure 3.26. Fundamental diagram of the fourth lane of four lane highway with off ramp at 300, exit sign at 200. $\rho_2 = 7$, $\rho = 20$, $r = 80$, and $s = 5$

According to the simulation, encouraging cars to prepare to exit well in advance is very advantageous. This is not a terribly obvious conclusion. Urging some cars into the right lane could have increased the density such that, over a larger distance, spontaneous jams would be formed that could decrease the overall flow. However, as we have seen in Figure 3.9., type 1 cars are apparently escaping fast enough that the density doesn't dramatically increase. In contrast, the relatively greater density of the right lane after the off ramp in Figure 3.19. indicates that in the second case, type 1 cars are unable to move left. Once the right lane becomes saturated, the same situation propagates left, lane by lane, as can be seen in the other space-time diagrams.

4. ADDITIONAL APPLICATIONS

To further demonstrate the versatility of TCA models such as our own, we give an example of a more complicated highway system, based on the Kavacik highway intersection in Istanbul. Relieving the constant traffic problems there would be a great service to many people. The system is a four lane highway with an off ramp, an on ramp, and a separated auxiliary lane.

On ramp:

Our depiction of the on ramp attempts to simulate the appropriate affects on density, but simplifies the dynamics at the location. At the 1 cell on ramp position, a waiting car may enter if it finds an opening, at some probability. It then appears on the main road with a 3 cells/timestep velocity.

Though the on ramp has no dimensions, there is a concept of on ramp density. Otherwise, the rate of cars added to the road would be dependent only on the density of the main road: for low main road densities, there would be more cars added and for low main road density, less cars added. In reality, cars come and wait to enter when the main road is congested, and enter as soon as they arrive when the main road is clear. To account for this, every time step, with a “density” probability, 1 is added to the number of hypothetical waiting cars. Every time a car enters the main road, the count decreases by 1. In the high density limit of the main road, then, the number of waiting cars increases and the entrance probability depends only on the main road density, and in the low limit, the number drops to zero and the entrance probability depends only on the so called density of the on ramp.

Auxiliary lane:

The extra lane added in Kavacik attempts to increase flow by decreasing the average density per lane of the highway. The lane separates from the main highway long before the on and off ramps and reconnects afterwards. In our simulation, the road array is given

2 more dimensions to represent this lane, the overall density of the highway is manually decreased to 4/5 and type 2 cars are restricted to the original four lanes in the case of the auxiliary lane joining after the off ramp. The addition of this lane occurs before the range of the simulation. The lane changing rules are not applied to the cars in the extra lane. At some variable position, all the cars are forced out of the lane and into the fourth. The length of the merge is 13 cell lengths (approximately 100 meters). To make sure the cars merge, they are decelerated as they approach the end, just as is done for the forced lane changing of the type 2 cars before the off ramp.

The following graphs provide a sample of the results that can be produced by our simulation for a section of highway with an on ramp at 315, adding 1 car per time step, an off ramp at 300, and an auxiliary lane which merges with the main road at 250. In this case, type 2 cars are allowed in the auxiliary lane at the same probability. Once they join with the fourth lane, they are forced right by the same rules as are applied to the original four lanes.

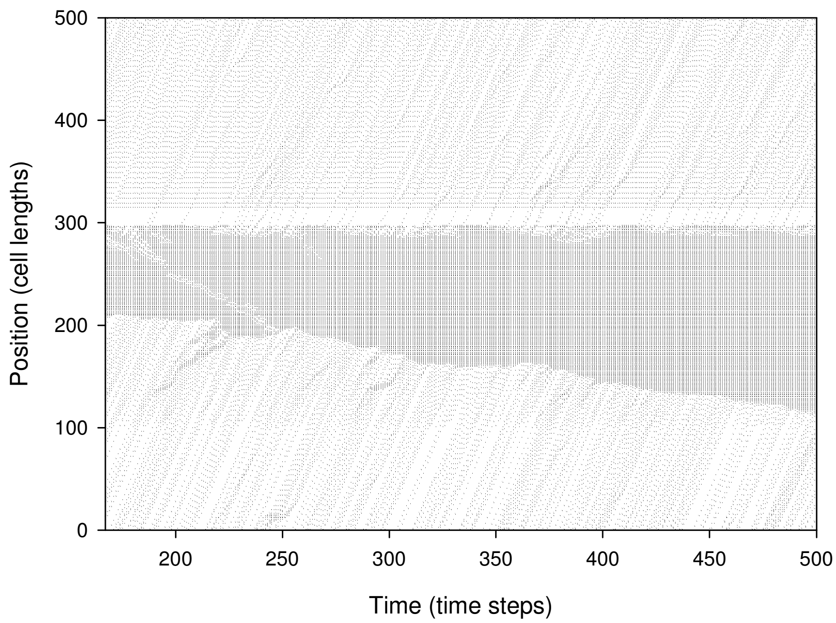

Figure 4.1. Space-time diagram of first lane at "Kavacik". Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

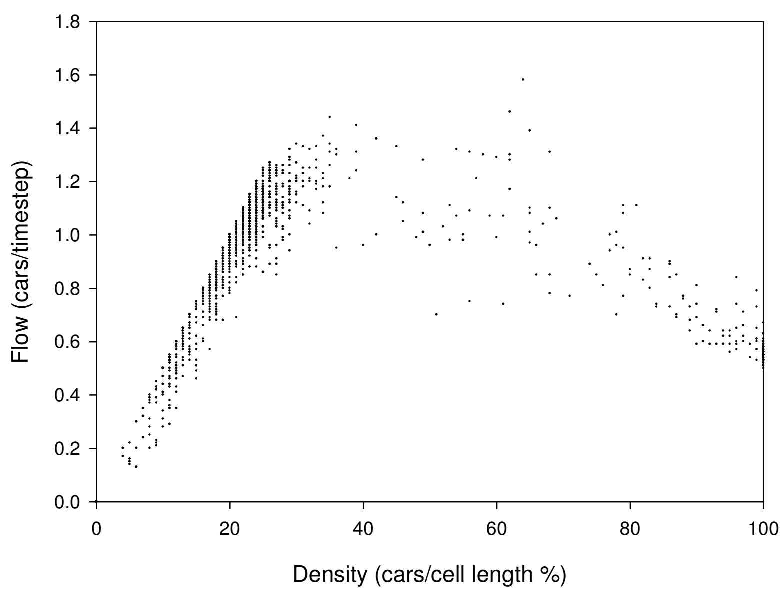

Figure 4.2. Fundamental diagram of first lane at "Kavacik". Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

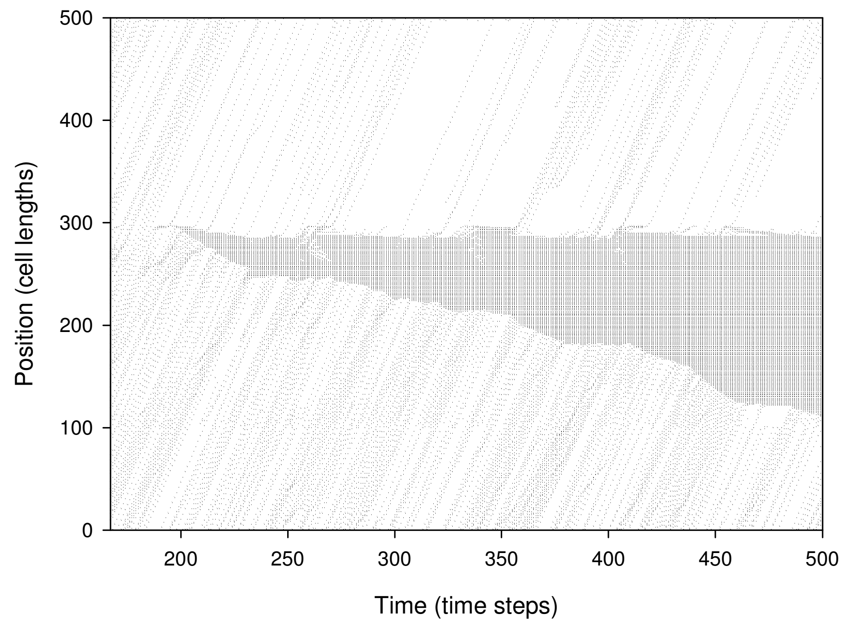

Figure 4.3. Space-time diagram of second lane at "Kavacik". Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

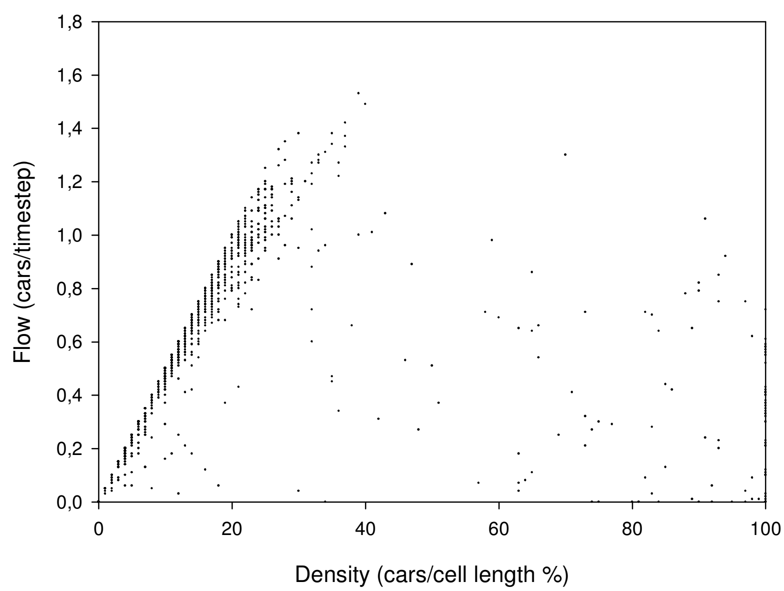

Figure 4.4. Fundamental diagram of second lane at "Kavacik". Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

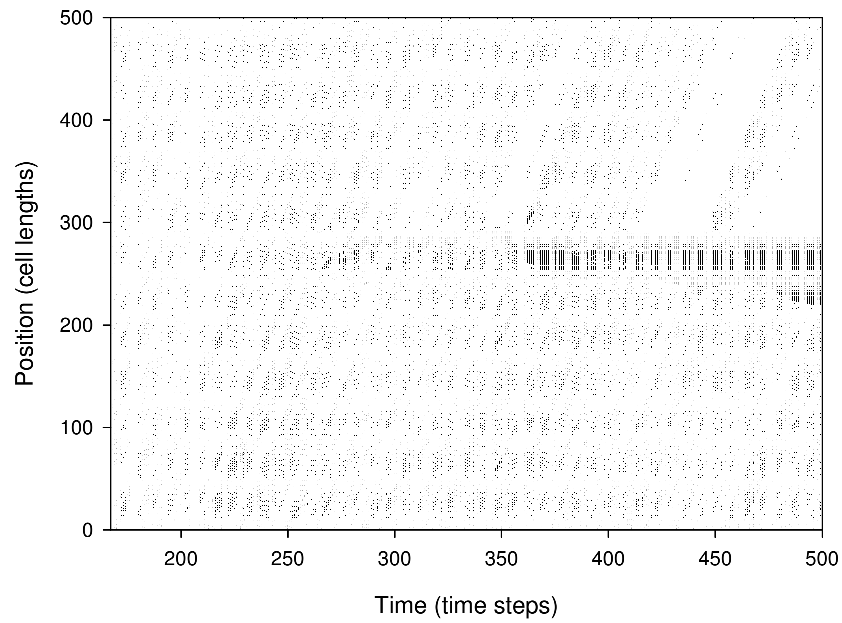

Figure 4.5. Space-time diagram of third lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

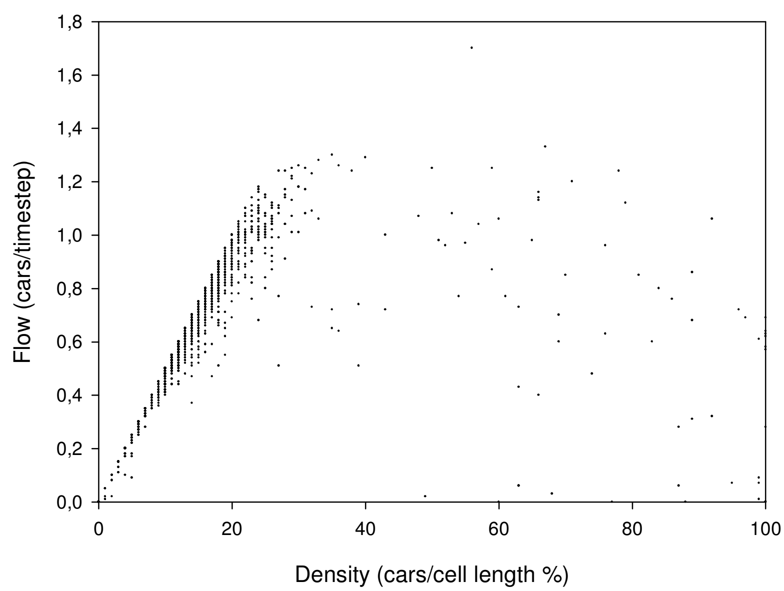

Figure 4.6. Fundamental diagram of third lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

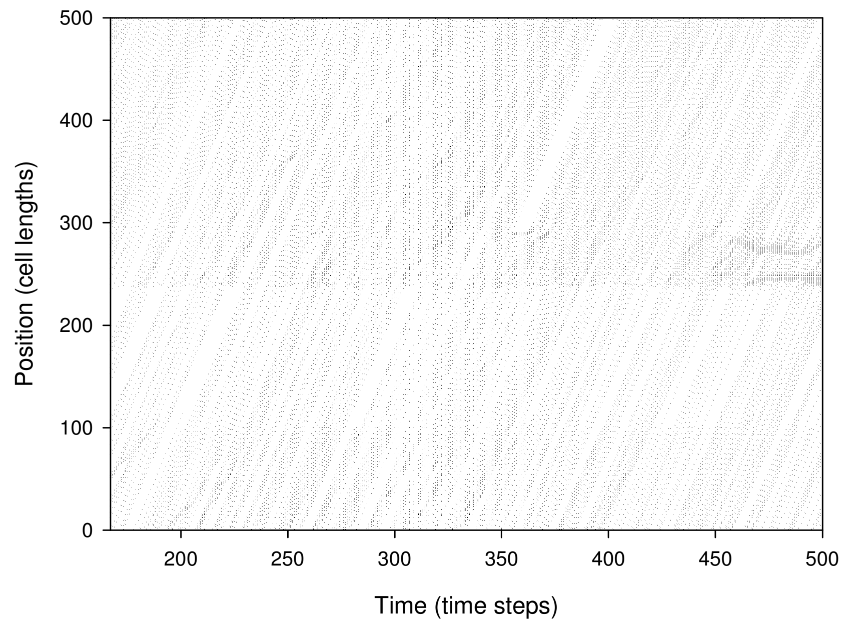

Figure 4.7. Space-time diagram of fourth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

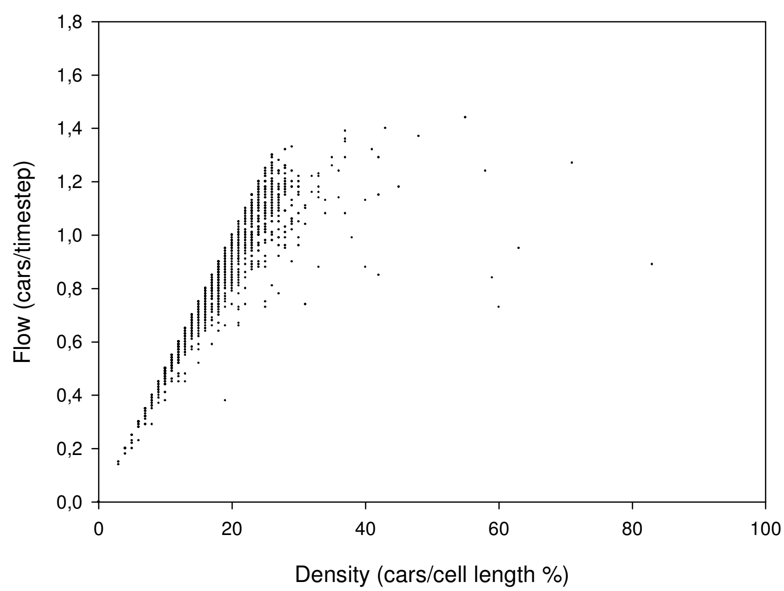

Figure 4.8. Fundamental diagram of fourth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

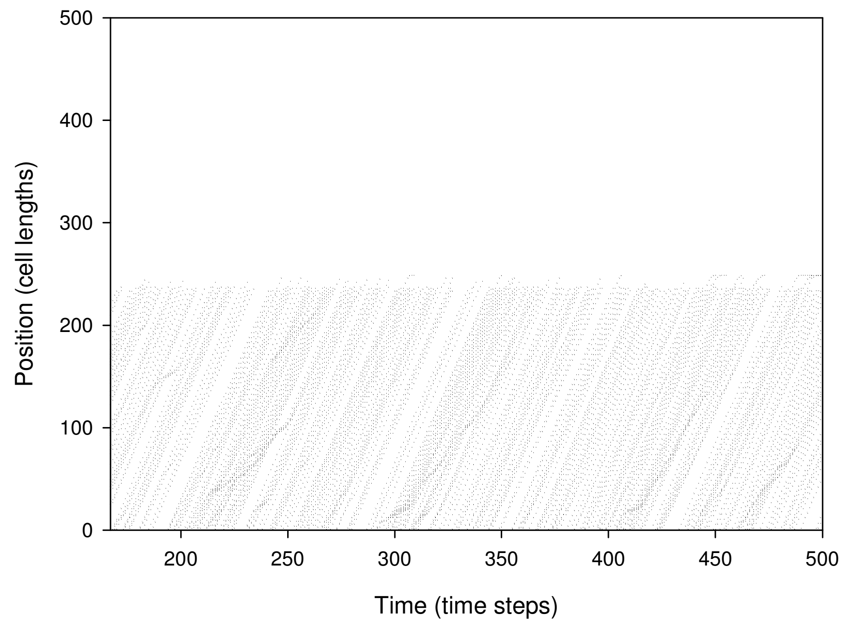

Figure 4.9. Space-time diagram of fifth lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, r = 80, s = 5

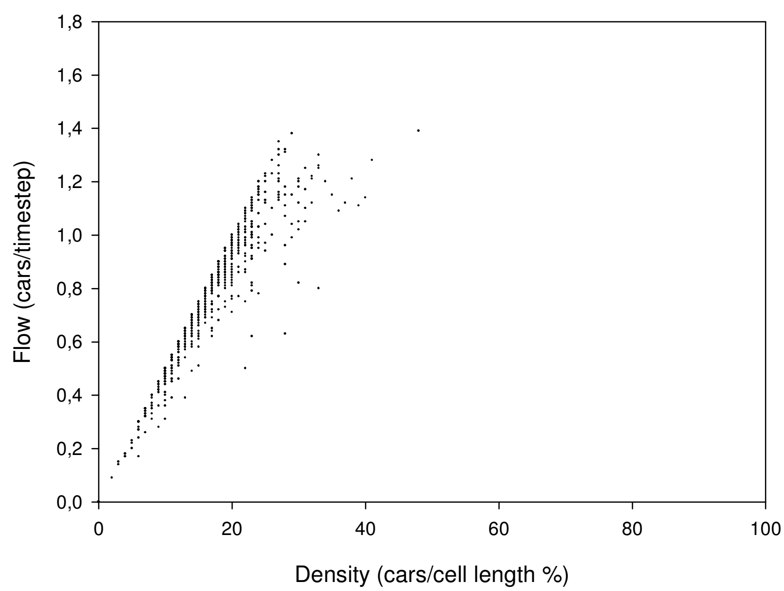

Figure 4.10. Fundamental diagram of fifth lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 250, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, r = 80, s = 5

The following graphs show the diagrams for the same road in which the auxiliary lane merge is moved forward to 300 (the same position as the off ramp, but on the left side of the road. Now type 2 cars are restricted to the four right lanes.

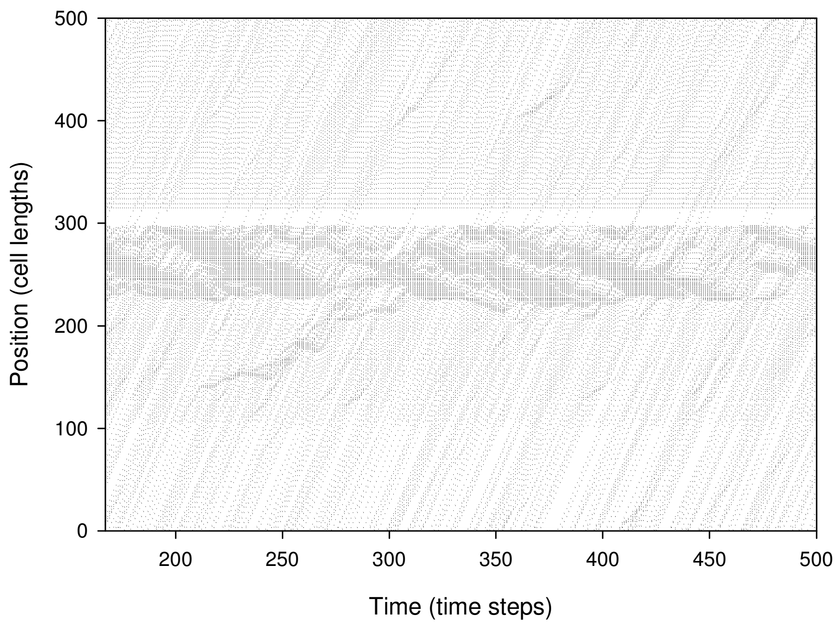

Figure 4.11. Space-time diagram of first lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

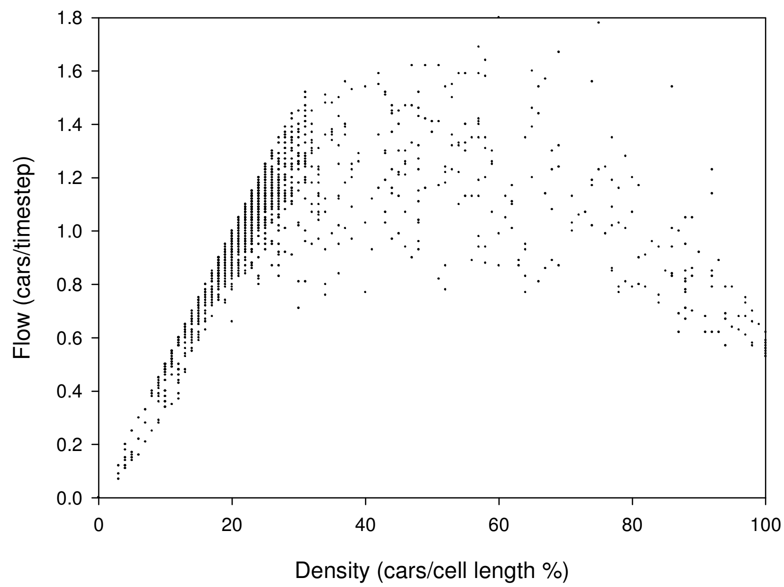

Figure 4.12. Fundamental diagram of first lane at "Kavacik". Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

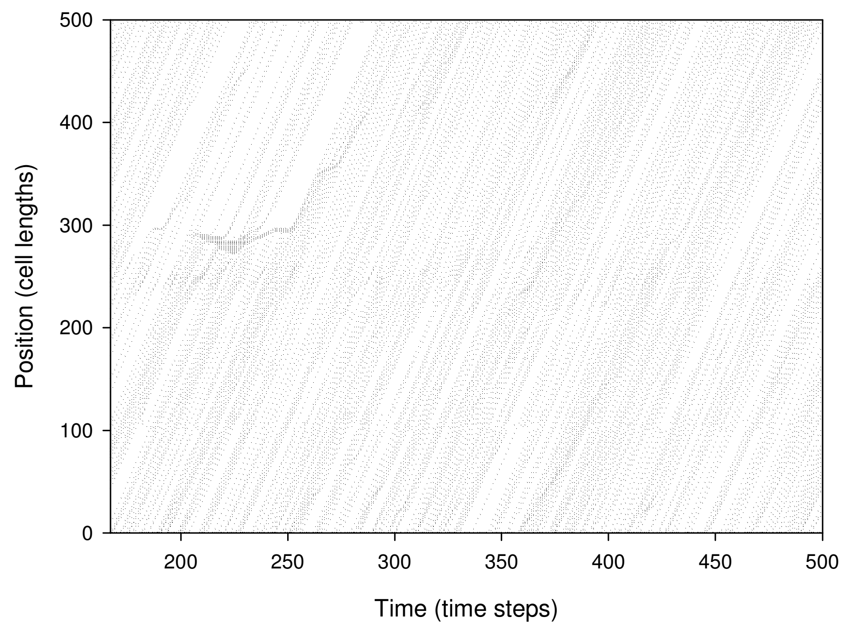

Figure 4.13. Space-time diagram of second lane at "Kavacik". Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

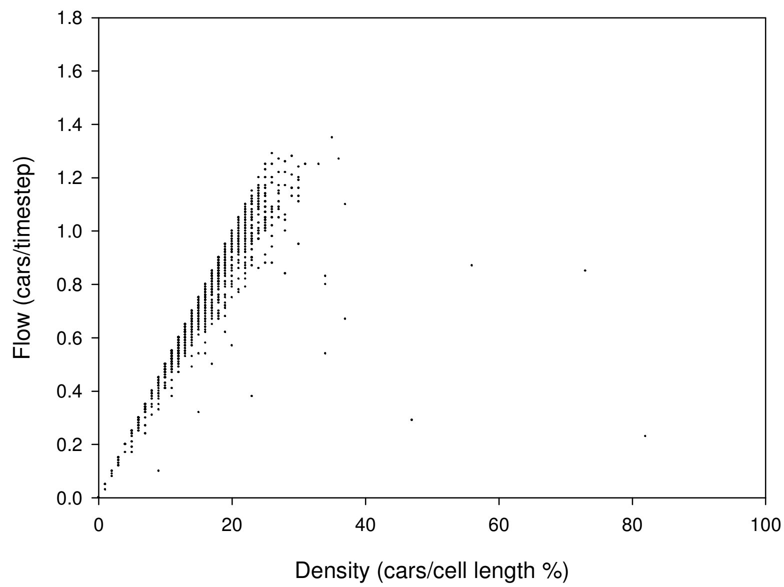

Figure 4.14. Fundamental diagram of second lane at "Kavacik". Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

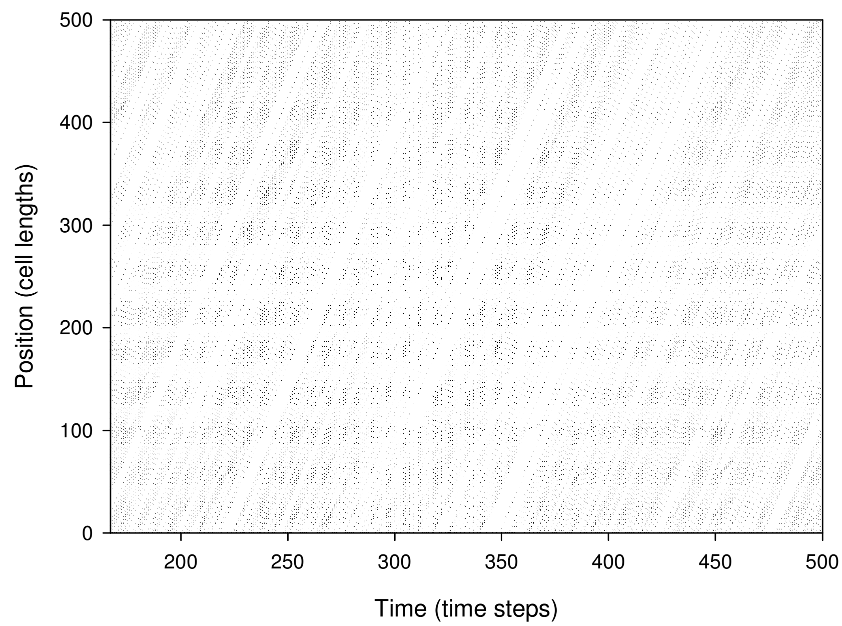

Figure 4.15. Space-time diagram of third lane at "Kavacik". Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

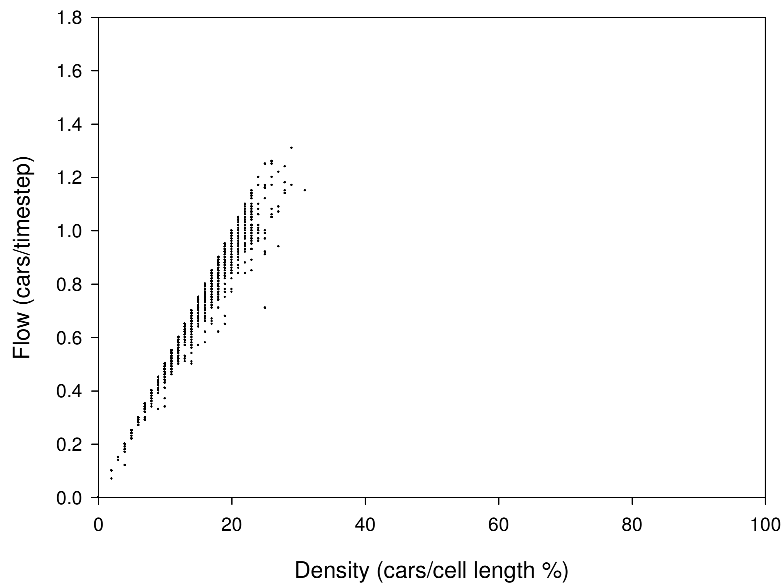

Figure 4.16. Fundamental diagram of third lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

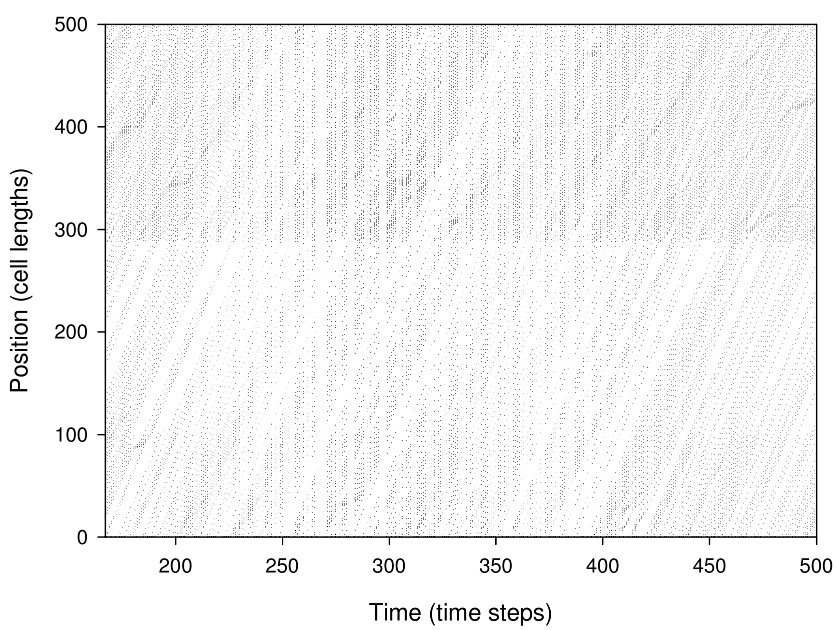

Figure 4.17. Space-time diagram of fourth lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

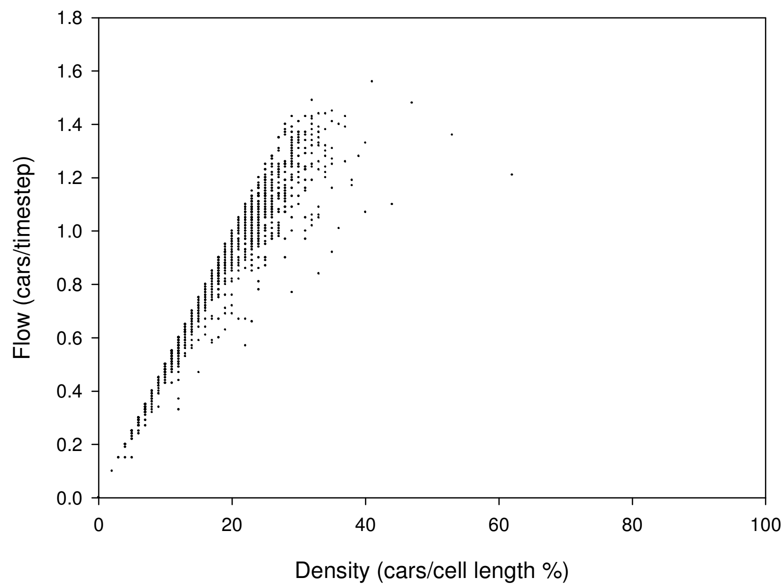

Figure 4.18. Fundamental diagram of fourth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

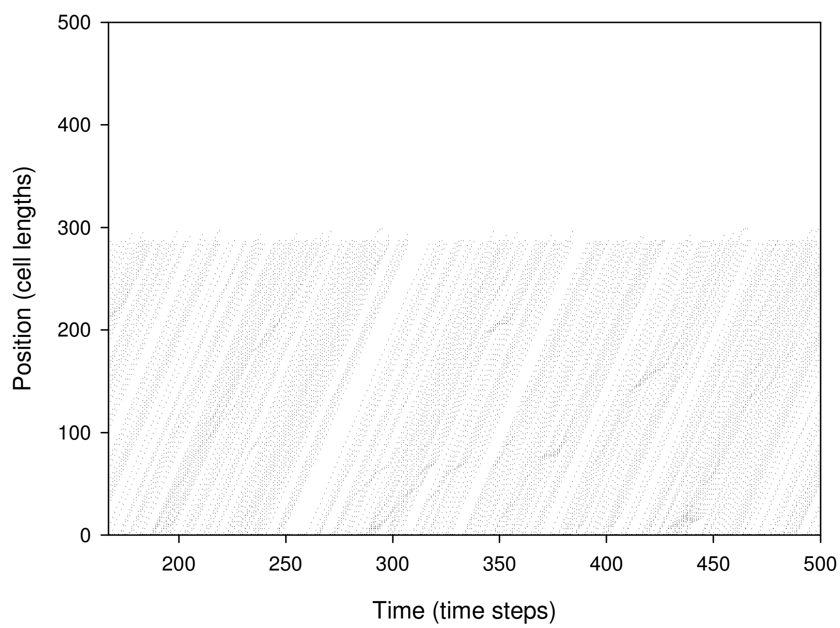

Figure 4.19. Space-time diagram of fifth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

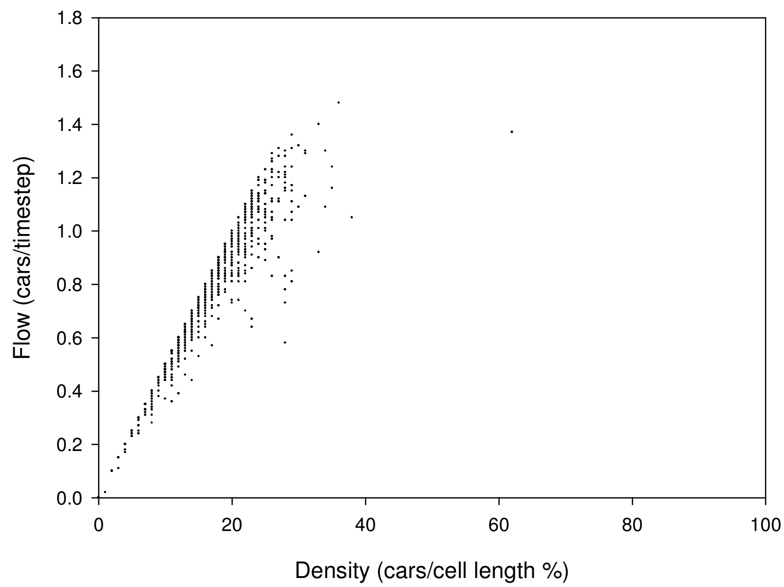

Figure 4.20. Fundamental diagram of fifth lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 300, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

Finally, the following shows results for the road when the merge has been moved to position 350, 375 meters after the off ramp. As we expect, moving the far left lane beyond this point does not significantly affect the dynamics of the traffic.

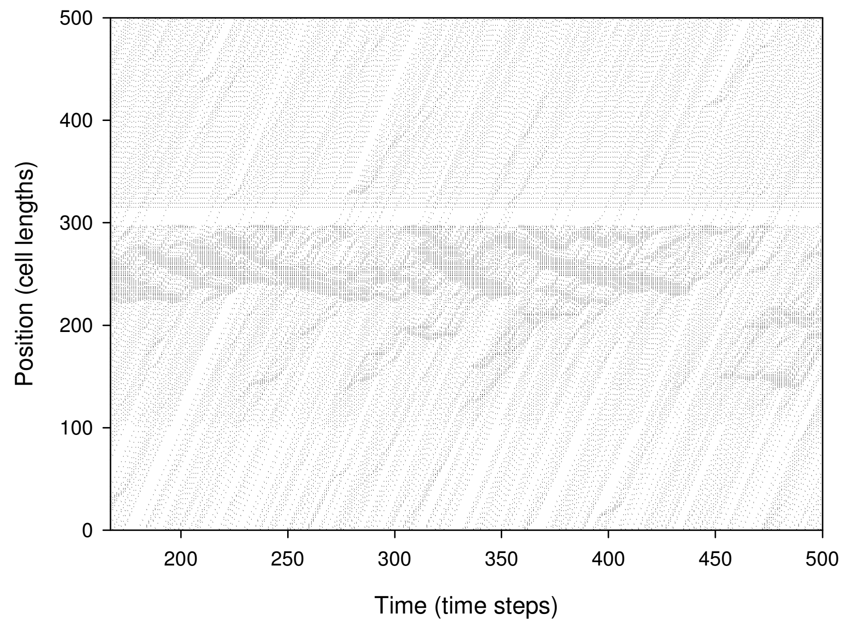

Figure 4.21. Space-time diagram of first lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

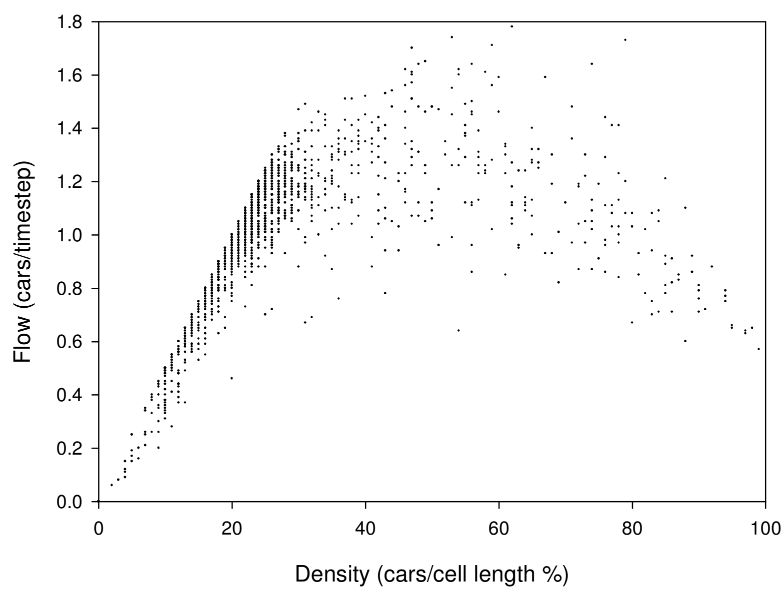

Figure 4.22. Fundamental diagram of first lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

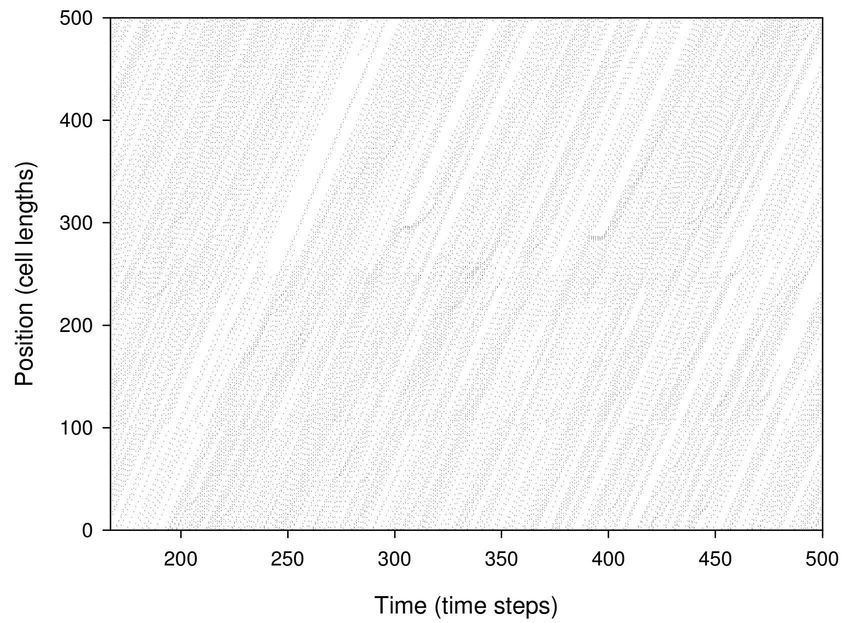

Figure 4.23. Space-time diagram of second lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

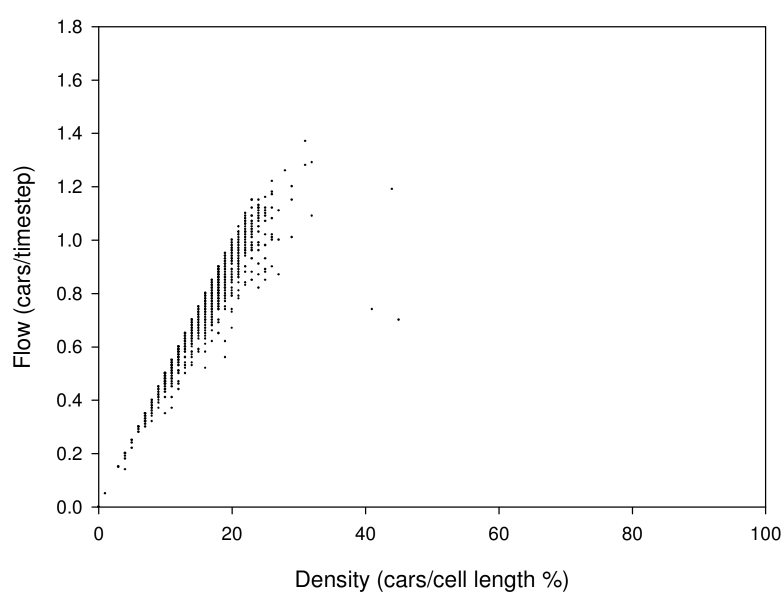

Figure 4.24. Fundamental diagram of second lane at “Kavacik”. Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

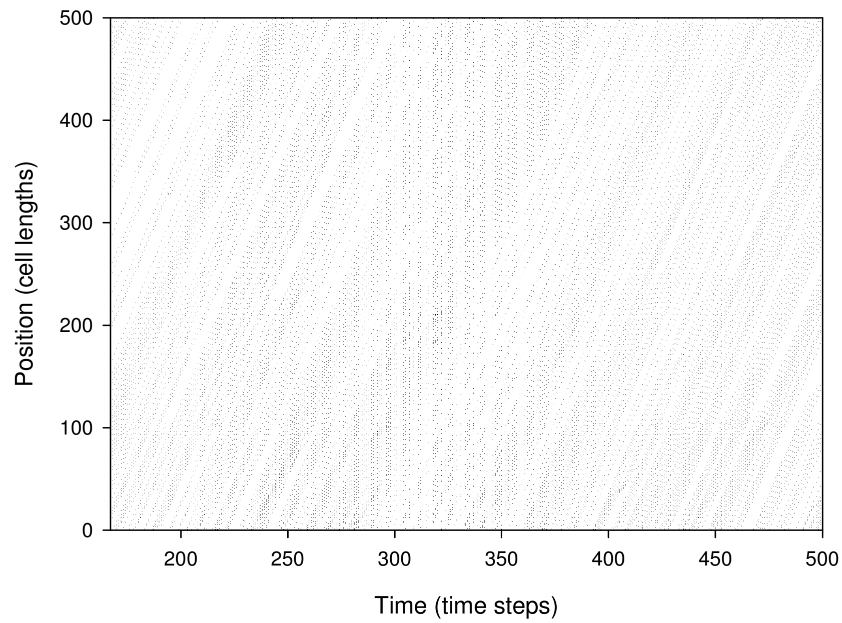

Figure 4.25. Space-time diagram of third lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

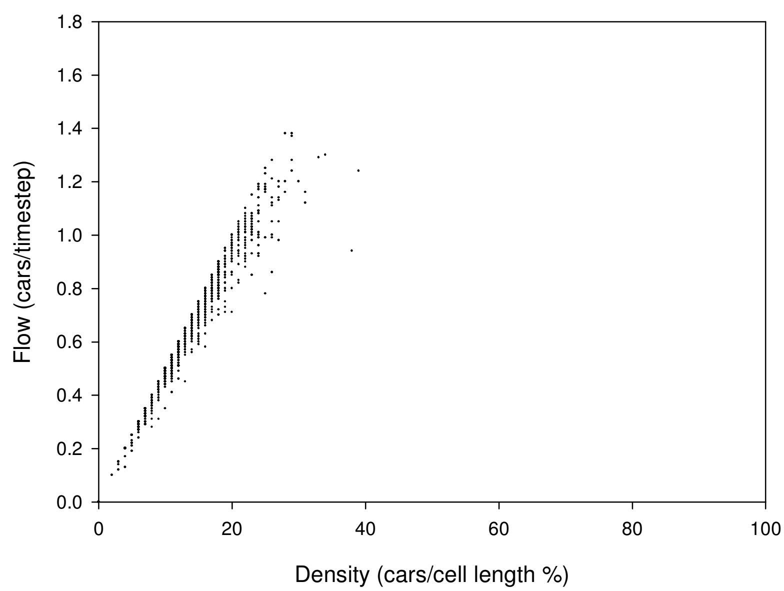

Figure 4.26. Fundamental diagram of third lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density $\rho_{on} = 100$, $\rho_2 = 7$, $\rho = 20$, $r = 80$, $s = 5$

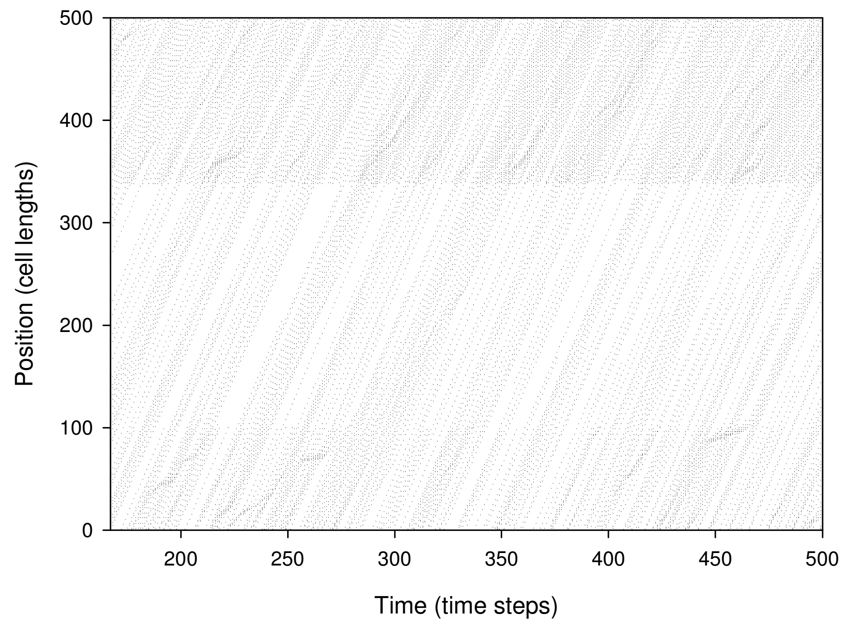

Figure 4.27. Space-time diagram of fourth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

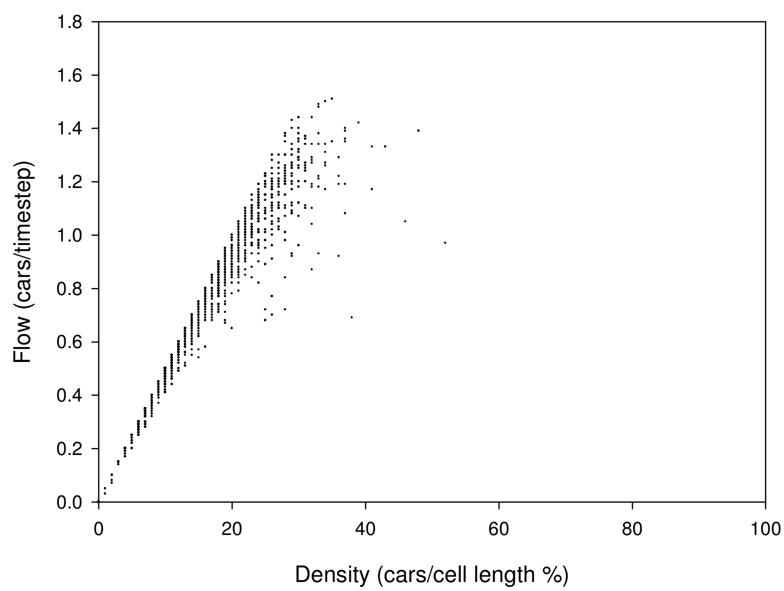

Figure 4.28. Fundamental diagram of fourth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

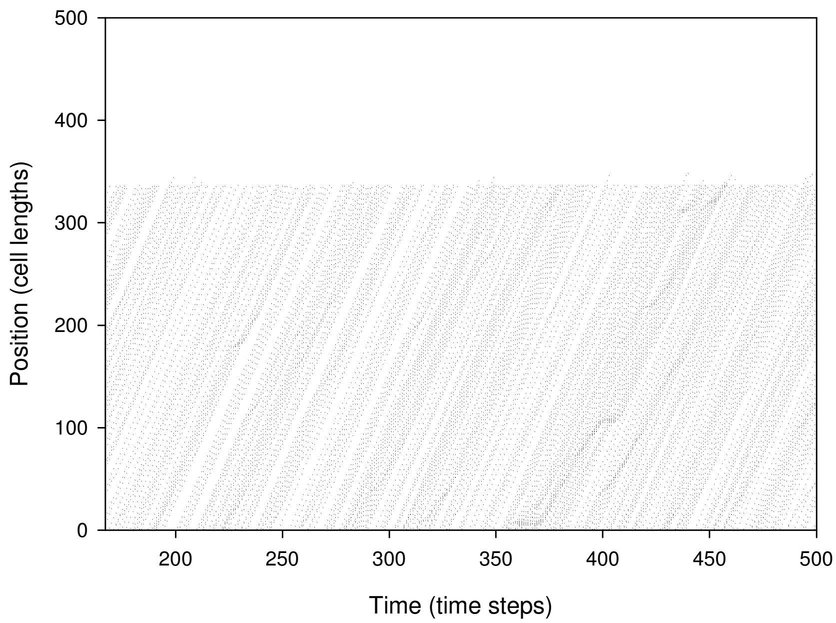

Figure 4.29. Space-time diagram of fifth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

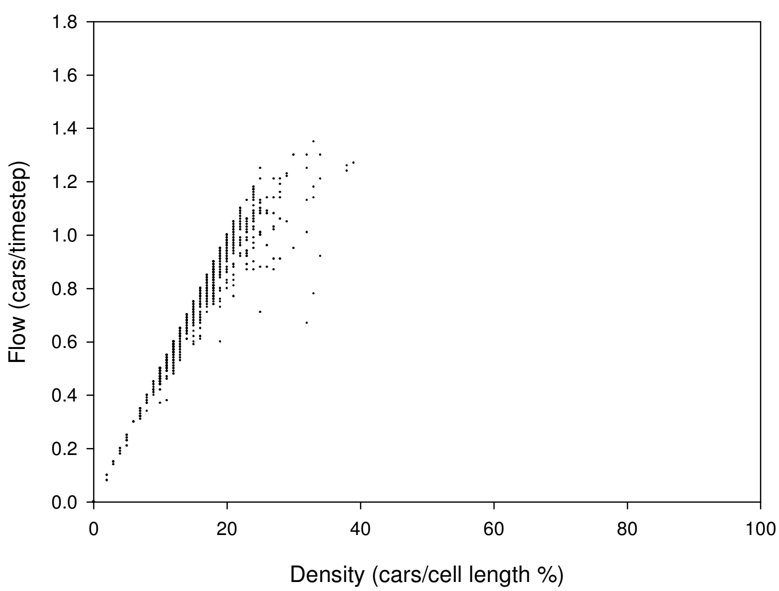

Figure 4.30. Fundamental diagram of fifth lane at "Kavacik". Off ramp at 300, sign at 100, merge at 350, on ramp at 315. On ramp density ρ_{on} = 100, ρ_2 = 7, ρ = 20, r = 80, s = 5

5. CONCLUSION

TCA models are versatile tools both for studying the basic dynamics of road traffic flow and for modeling specialized highway systems. Our unique model has been used for both purposes with varying degrees of success. We have analyzed the results it produces by studying exclusively the corresponding space-time diagrams showing the positions of all cars in the road each time step and "fundamental diagrams" which show the relation between flow and density.

On the side of basic traffic dynamics, our model gives clear evidence of two distinct states of traffic flow: free flow and congested flow. The congested flow seen on unobstructed portions of highway are indicative of "spontaneous" jams which have also been found empirically in real traffic using detectors and with other TCA models such as the famous NaSch model. The features of our fundamental diagrams which illustrate the two phases are qualitatively similar to these other studies, but quantitatively different. Most importantly, our model gives a flow rate at 100% density of approximately 0.5 cars/timestep, at ends with the results of the NaSch model (which predicts 0 cars/timestep) and common sense. The cause of this result is our sequential, rather than simultaneous application of the local rule. This allows information about the movement of the first car to flow backwards through the traffic at an arbitrary rate within one time step. In effect, this increases car movement at higher densities and causes a "creeping" in both spontaneous jams other jams in which the front car is allowed to move (i.e. the right lane before an off ramp).

We have found that allowing cars to change lanes for their own advantage also increases the overall flow on a multilane highway by decreasing the probability of overreaction in deceleration which causes spontaneous jams. This result depends, however, on cars making a full lane change within 1 time step (1.2 seconds). Adjusting this time scale would be an interesting topic of further research and may produce a different conclusion.

We have created a simulation of a multilane highway in which a certain portion of cars try exit from an off ramp. Cars that are bound for the exit may start in any lane, and their motion toward the right lane dramatically affects the overall flow of the highway. We have found that placing a sign well in advance of an oncoming off ramp prevents most of the severe jamming that would occur when cars try to move right closer to the off ramp. The early application of the sign allows for exiting cars to change lanes without resorting to deceleration (an action which undoubtedly causes spontaneous jams). More importantly, placing the sign far in advance allows forward bound cars to abandon the right lane as its density increases as the result of exiting cars gradually moving into it. We have shown that placing this sign too late creates a high probability of trapping forward bound cars in the right lane and creating immobile jams that eventually spread through all the lanes.

We have expanded our program to accommodate more complex realistic simulations, such as that of a problematic highway intersection near Kavacik, Istanbul. Sample space-time and fundamental diagrams for this intersection show that our program gives reasonable results, and is ready for use in further study.

One general comment should be made for all the results shown in this thesis. Because of the stochastic properties of the simulations we used, no results are 100% reproducible. Also, because of the unstable nature of some of the systems, larger features of some of the results do not always occur under the same conditions. For instance, in simulations involving the off ramp (in Section 3.2), we have shown results with permanent jams that, once formed, would continue to grow until the input density is decreased. These jams occur because of instability near the off ramp, but do not necessarily happen every time the program is run. The results given for the Kavacik system in Chapter 4, there is no evidence of such jams but there is a probability for them to occur. We have made an attempt to display results which are most typical for the conditions used and to make comments on the more characteristic results, rather than isolated ones. Further study on the stability and reproducibility would definitely be worthwhile in the future.

APPENDIX A: OUR CODE FOR "THE GAME OF LIFE"

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define   size      57
#define   startprob 10
#define   iter      51

main()
{
int k=0, i, j, add=0, space[size+2][size+2]={0};
int step, a,count, total=0;
int update[size][size]={0};
FILE *fptr, *fptr2;
if ((fptr = fopen("picture.dat", "w")) == NULL)
     {
          printf("File did not open...\n");
          exit(1);
     }
if ((fptr2 = fopen("life.dat", "w")) == NULL)
     {
          printf("File did not open...\n");
          exit(1);
     }
srand((unsigned)time(NULL));
//for(k=0;k<1000;k=k++)
//{
for(i=1;i<size-1;i=i++)
{
     for(j=1;j<size-1;j=j++)
     {
          a = rand()%(100);
          if(a<startprob+add)
               space[i][j]=1;
          if(space[i][j] == 1)
               printf("-");
          else
               printf(" ");
          if (j==size-1)
               printf("\n");
//        else
//             space[i][j]=0;
     }
}
for(step=0;step<iter;step=step++)
{
     for(i=1;i<size;i=i++)
     {
          for(j=1;j<size;j=j++)
          {
               count=space[i][j+1]+space[i][j-1]
                         +space[i+1][j]+space[i-1][j]
                         +space[i+1][j+1]+space[i+1][j-1]
                         +space[i-1][j+1]+space[i-1][j-1];
```

```
	if((space[i][j]==0&&count==3)||(space[i][j]==1&&(count==2||count==3)
))
			{
				update[i][j]=1;
				if (step == 50)
					fprintf(fptr, "%d  %d\n", j, i);
			}
			else
				update[i][j]=0;
		}
	}
	for(i=1;i<size;i=i++)
	{
		for(j=0;j<size;j=j++)
		{
			space[i][j]=update[i][j];
			if (step>-1)
			{
			if(space[i][j]==1)
			{
				printf(".");
//				if (step == iter - 1)
//					total = total++;
			}
			else
				printf(" ");
			if(j==size-1)
				printf("\n");
			}
		}
	}
//if(step>-1){/* THIS IS HELPFUL FOR VIEWING DIRECTLY ON LARGE C PROMPT
			SCREEN */
//printf("\n-------------------------------------------------------\n");
//for (i=0;i<10000000;i=i++)
//	if (i%5==0)
//		j = j + 1;
j=0;
fprintf(fptr2, "%d %d\n", add, total);
total = 0;
if (k%10 == 0)
	add = add++;

}
fclose(fptr);
fclose(fptr2);
return 0;
}
```

APPENDIX B: CODE FOR HIGHWAY SIMULATIONS

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define num                500
#define dens               20
#define iter               500
#define vmax               5
#define reprob             80
#define lanechangegap      1
#define detector_length    10
#define slowprob           5
#define changeprob         100
#define onramp             315
#define enterprob          100
#define onrampdense        100
#define offramp            300
#define twos               7
#define startshift         100
#define det_ave            10
#define merge              250

main()
{
     int i, lane, change = 0, a;
     int car[num][10] = {0}, map[num][5] = {0};
     int cars1, cars2, cars3, cars4, cars5, waitlist, type, j, step, k,
          speed1, l;
     int move[num][10] = {0};
     int lead, speed2;
     float react, ave_speed, totcount=0, totspeed=0;
     int add, cars, slow;
     float count=0, speed=0, dense[num/10][5] = {0}, flow[num/10][5] =
          {0};
     int switchright, switchleft, lanechange, enter, wait;
     FILE *fptr, *fptr2;
     if ((fptr = fopen("map.dat", "w")) == NULL)
     {
          printf("File did not open...\n");
          exit(1);
     }
     if ((fptr2 = fopen("graph.dat", "w")) == NULL)
     {
          printf("File2 did not open...\n");
          exit(1);
     }
     srand((unsigned)time(NULL));
     for (lane=0;lane<9;lane=lane+2)
     {
          for (i=0;i<num;i=i++)
          {
               a = rand()%(100);
               if (a<=dens)
               {
                    car[i][lane] = 1;
```

```
                    car[i][lane+1] = 5;
                }
            }
        }
        cars1 = 0;
        cars2 = 0;
        cars3 = 0;
        cars4 = 0;
        cars5 = 0;
        waitlist = 5;
        for (j=vmax;j<num;j=j++)
        {
//              printf("row %4d: %d - %d  :  %d - %d  :  %d - %d  :  %d -
                    %d\n", j, car[j][0], car[j][1], car[j][2], car[j][3],
                    car[j][4], car[j][5], car[j][6], car[j][7]);
                cars1 = cars1 + car[j][0];
                cars2 = cars2 + car[j][2];
                cars3 = cars3 + car[j][4];
                cars4 = cars4 + car[j][6];
                cars5 = cars5 + car[j][8];
        }
//      printf("total cars is:  %d  %d   %d   %d\n", cars1, cars2, cars3,
            cars4);
        cars1 = 0;
        cars2 = 0;
        cars3 = 0;
        cars4 = 0;
        cars5 = 0;
for (step=0;step<iter;step=step++)    /* Main Loop of Iterations = iter
*/
{
        add = 0;
        if (step > num/3)
        {
            for (lane=0;lane<9;lane=lane+2)
            {
                for (k=0,i=0;k<num;k=k++)
                {
                    if (car[k][lane] != 0)
                    {
//                      if (car[k][lane] != 1)
//                          printf("%d    %d\n", lane, car[k][lane]);
                        map[i][lane/2] = k;
                        i = i++;
                    }
                }
            }
            for (k=0;k<num;k=k++)
            {
                if (map[k][0] != 0 || map[k][1] != 0 || map[k][2] != 0 ||
                    map[k][3] != 0 || map[k][4] != 0)
                fprintf(fptr, "%d  %d   %d   %d   %d   %d\n", step,
                map[k][0], map[k][1], map[k][2], map[k][3], map[k][4]);
            }
            for (lane=0;lane<9;lane=lane+2)
            {
                for (k=0,i=0;k<num;k=k++)
                {
                    map[k][lane/2] = 0;
                }
```

```
		}
	}
	for (lane=0;lane<9;lane=lane+2)
	{
		for (k=num-1;k>=0;k=k--)
		{
			if (car[k][lane] != 0)
			{
				if (k > offramp)
					type = 1;
				else
					type = car[k][lane];
				if ((type == 1 && car[k][lane+1] < vmax) || (type == 2
					&& k < offramp - 15 && car[k][lane+1] < vmax &&
					lane != 0) || (type == 2 && (lane == 0 || lane ==
					8) && car[k][lane+1] < vmax))
					speed1 = car[k][lane+1] + 1;
				else
					speed1 = car[k][lane+1];
				if (k + speed1 < num)
				{
					move[k + speed1][lane] = type;
					move[k + speed1][lane+1] = speed1;
					lead = k;
					break;
				}
			}
		}
		for (k=lead-1;k>=0;k=k--)		/* Main car moving Loop (moves
				from leading car backwards) */
		{
			if (car[k][lane] != 0)
			{
				if (k > offramp)
					type = 1;
				else
					type = car[k][lane];
				if ((type == 1 && car[k][lane+1] < vmax) || (type == 2
					&& k < offramp - 15 && car[k][lane+1] < vmax &&
					lane != 0) || (type == 2 && (lane == 0 || lane ==
					8) && car[k][lane+1] < vmax))
					speed1 = car[k][lane+1] + 1;
				else
					speed1 = car[k][lane+1];
				for (j=k+1;j<num;j=j++)
				{
					if (move[j][lane] != 0)
					{
						speed2 = move[j][lane+1];
						break;
					}
				}
				if (j <= k + speed1)
				{
					react = (k + speed1 - j + 1)*rand()%(100
//					printf("%d\n", react);
					move[j-1][lane] = type;
					if (react > 100 - reprob && j - k - 2 >= 0)
						move[j-1][lane+1] = j - k - 2;
					else
```

```
                    move[j-1][lane+1] = j - k - 1;
               }
               else
               {
                    move[k+speed1][lane] = type;
                    if (speed1 > 3 && ((j == k + speed1 + 1) || (j ==
                         k + speed1 + 2 && speed1 == 5)))
                    {
                         move[k+speed1][lane+1] = 3;
                    }
                    else
                         move[k+speed1][lane+1] = speed1;
               }
          }
     }
     cars  = 0;
     for (k=vmax;k<num;k=k++)
     {
/*        if (lane == 8 && k > 8 && k < 25 && step > 100)
          {
               printf("row %4d: %d - %d  :  %d - %d  :  %d - %d  :
               %d - %d  :  %d - %d\n", k, move[k][0], move[k][1],
               move[k][2], move[k][3], move[k][4], move[k][5],
               move[k][6], move[k][7], move[k][8], move[k][9]);
          }
*/        if (move[k][lane] != 0)
          {
               type = move[k][lane];
               if (lane == 0 && k > offramp - 10 && k < offramp &&
                    move[k][lane+1] > 3 && type == 2)
               {
                    move[k][lane+1] = 3;
               }
               if (lane == 0 && k > offramp - 5 && k < offramp &&
                    move[k][lane+1] > 2 && type == 2)
               {
                    move[k][lane+1] = 2;
               }
               if (type == 2 && lane == 2 && k > offramp - 8 &&
                    move[k][lane+1] > (offramp - k - 2)/2)
               {
                    move[k][lane+1] = (offramp - k - 2)/2;
                    if (move[k][lane+1] < 0)
                         move[k][lane+1] = 0;
               }
               if (type == 2 && lane == 4 && k > offramp - 12 &&
                    move[k][lane+1] > (offramp - k - 4)/2)
               {
                    move[k][lane+1] = (offramp - k - 4)/2;
                    if (move[k][lane+1] < 0)
                         move[k][lane+1] = 0;
               }
               if (type == 2 && lane == 6 && k > offramp - 14 &&
                    move[k][lane+1] > (offramp - k - 8)/2)
               {
                    move[k][lane+1] = (offramp - k - 8)/2;
                    if (move[k][lane+1] < 0)
                         move[k][lane+1] = 0;
               }
               slow = rand()%(100);
```

```
                    if (slow<slowprob && move[k][lane+1] - 1 > 0)
                    {
                         move[k][lane+1] = move[k][lane+1] - 2;
/*                       printf("slow in lane %d\n", lane);
*/                  }
               }
               if (move[k][lane] != 0)
                    cars = cars + 1;
          }
          if (lane == 0)
               cars1 = cars;
          if (lane == 2)
               cars2 = cars;
          if (lane == 4)
               cars3 = cars;
          if (lane == 6)
               cars4 = cars;
          if (lane == 8)
               cars5 = cars;
          for (k=0;k<vmax;k=k++)  /*Loop adds cars to beginning of road*/
          {
               a = rand()%(100);
               if (a<=dens+change && a > 0)
               {
                    if (a<=twos && lane < 7)
                    {
                         move[k][lane] = 2;
//                       printf("another two\n");
                    }
                    else
                    {
                         move[k][lane] = 1;
                    }
                    move[k][lane+1] = 5;
               }
               else
               {
                    move[k][lane] = 0;
                    move[k][lane+1] = 0;
               }
          }
//        if (lane == 8)
//             printf("totals:      %d     %d          %d      %d
                         %d\n", cars1, cars2, cars3, cars4, cars5);
          if (step > 170)
          {
               for (i=10;i<num-detector_length+1;i=i+10)
               {
                    for (k=i;k<i+detector_length;k=k++) /* CALCULATIONS
                         for flow-density graphs */
                    {
                         if (move[k][lane] != 0)
                         {
                              count = count + 1;
                              totcount = totcount + 1;
                              speed = speed + move[k][lane+1];
                              totspeed = totspeed + move[k][lane+1];
                         }
                    }
```

```
                    dense[(i/10)-1][lane/2] = dense[(i/10)-1][lane/2] +
                         count/detector_length;
                    flow[(i/10)-1][lane/2] = flow[(i/10)-1][lane/2] +
                         speed/detector_length;
                    count = 0;
                    speed = 0;
               }
               if (step%det_ave == 0 && lane == 8)
               {
                    for (j=0;j<num/10;j=j++)
                    {
                         for (l=0;l<5;l=l++)
                         {
                              dense[j][l] = 100*dense[j][l]/det_ave;
                              flow[j][l] = flow[j][l]/det_ave;
                         }
                    fprintf(fptr2, "%d %g   %g   %g   %g   %g   %g   %g   %g
                         %g   %g\n", j, dense[j][0], flow[j][0],
                         dense[j][1], flow[j][1], dense[j][2], flow[j][2],
                         dense[j][3], flow[j][3], dense[j][4],
                         flow[j][4]);
//                  printf("%d    %g   %g   %g   %g   %g   %g   %g   %g   %g
                              %g\n", j, dense[j][0], flow[j][0],
                              dense[j][1], flow[j][1], dense[j][2],
                              flow[j][2], dense[j][3], flow[j][3],
                              dense[j][4], flow[j][4]);
                    dense[j][0] = 0;
                    flow[j][0] = 0;
                    dense[j][1] = 0;
                    flow[j][1] = 0;
                    dense[j][2] = 0;
                    flow[j][2] = 0;
                    dense[j][3] = 0;
                    flow[j][3] = 0;
                    dense[j][4] = 0;
                    flow[j][4] = 0;
                    }
               }
          }
     }
//   if (step > 60 && lane == 6)
//   for (k=offramp-10;k<offramp+5;k=k++)
//        printf("row %d:  %d - %d  :  %d - %d  :  %d - %d  :  %d -
          %d\n", k, move[k][0], move[k][1], move[k][2], move[k][3],
          move[k][4], move[k][5], move[k][6], move[k][7]);

     for (lane=0;lane<7;lane=lane+2)      /*  This loop allows cars to
                    change lanes under appropriate conditions  */
     {
          for (i=num-1;i>vmax;i=i--)
          {
               if (move[i][lane] != 0)
               {
                    type = move[i][lane];
                    speed1 = move[i][lane+1];
                    for (j=i+1;j<i+speed1;j=j++)
                         if (move[j][lane] != 0)
                              {
                                   speed2 = move[j][lane+1];
                                   break;
```

```
                    }
                if ((type == 1 && i+speed1>j+speed2) || type == 2)
                {
                    if (lane < 6 && ((type == 1) || (type == 2 && i <
                        startshift && i+speed1>j+speed2)))
                    {
                        for (k=i-1;k<num;k=k++)
                            if (move[k][lane+2] != 0)
                                break;
                        if (k - i > lanechangegap)
                            switchright = k;
                        else
                            switchright = 0;
                    }
                    else
                        switchright = 0;
//                  printf("lane %d, row %d's swright = %d\n", lane,
                            i, switchright);
                    if (lane > 0 && ((type == 1) || (type == 2 && i >
                        startshift)))
                    {
                        for (k=i-1;k<num;k=k++)
                            if (move[k][lane-2] != 0)
                                break;
                        if (move[i][lane-2] != 0 && ((lane == 2 && i
            > offramp - 15) || (lane == 4 && i > offramp - 30) || (lane ==
            6 && i > offramp - 45)) && type == 2)
                        {
                            speed1 = move[i][lane+1];
                            for (l=i-1;l>i-vmax;l=l--)
                            {
                                if (move[l][lane-2] != 0 && move[l-
                                        1][lane-2] == 0)
                                {
                                    speed2 = move[l][lane-1];
                                    if (l + speed2 > i + speed1)
                                    {
                                        move[l-1][lane-2] = 2;
                                        move[l-1][lane-1] =
                                            speed1;
                                        k = 0;
                                        switchright = 0;
                                        move[i][lane] = 0;
                                        move[i][lane+1] = 0;
                                    }
                                }
                            }
                        }
                        if (k - i > lanechangegap)
                            switchleft = k;
                        else
                            switchleft = 0;
                        if (type == 2 && move[i][lane-2] == 0)
                        {
                            if ((lane == 2 && i > offramp - 50) ||
                        (lane == 4 && i > offramp - 60) || (lane ==
                        6 && i > offramp - 70))
                            switchleft = num;
                        }
                    }
```

```
                        else
                             switchleft = 0;
//                      printf("lane %d, row %d's swleft = %d\n", lane,
                                  i, switchleft);
                        if (switchright != 0 || switchleft != 0)
                        {
                             lanechange = rand()%(100);
                             if (lanechange < changeprob || (type == 2 &&
            (i > startshift || (lane == 2 && i > offramp - 15) || (lane ==
            4 && i > offramp - 30) || (lane == 6 && i > offramp - 45))))
                             {
                                  if (switchright>=switchleft)
                                  {
                                       move[i][lane+2] = type;
                                       move[i][lane+3] = speed1;
                                       move[i][lane] = 0;
                                       move[i][lane+1] = 0;
                                  }
                                  else
                                  {
                                       move[i][lane-2] = type;
                                       move[i][lane-1] = speed1;
                                       move[i][lane] = 0;
                                       move[i][lane+1] = 0;
                                  }
                             }
                        }
                   }
              }
         }
    }
    for (k=offramp;k>=offramp-vmax;k=k--)  /*  Takes type 2 cars off the
         road at offramp  */
    {
         if (move[k][0] == 2)
         {
              speed1 = move[k][1];
              if ((k+speed1 >= offramp) || (k <= offramp + 3 && k >
         offramp - 2))
              {
                   move[k][0] = 0;
                   move[k][1] = 0;
//                 printf("eksitted at %d\n", k);
              }
         }
    }
    for (k=onramp;k<onramp+2;k=k++)   /*  Puts cars on the road at
         onramp  */
    {
         if (move[k][0] == 0 && waitlist > 0)
         {
              enter = rand()%(100);
              if (enter < enterprob)
              {
                   move[k][0] = 1;
                   move[k][1] = 3;
                   waitlist = waitlist - 1;
//                 printf("entered at %d\n", k);
              }
         }
```

```
     }
     wait = rand()%(100);
     if (wait<onrampdense)             /*  Manages the cars waiting to
          enter on onramp  */
     {
          waitlist = waitlist + 1;
//        printf("waitlist is %d\n", waitlist);
     }
     for (k=merge;k>merge-13;k=k--)
     {
          if (move[k][8] != 0)
          {
               speed1 = move[k][9];
               if (move[k][6] == 0)
               {
                    move[k][6] = 1;
                    move[k][7] = speed1;
                    move[k][8] = 0;
                    move[k][9] = 0;
               }
               else
               {
                    if (k > merge - 9 && speed1 > (merge - k)/2 - 1)
                         move[k][9] = (merge - k)/2 - 1;
               }
          }
     }
     for (lane=0;lane<9;lane=lane+2)
     {
          for (k=0;k<num;k=k++)
          {
               car[k][lane] = move[k][lane];
               car[k][lane+1] = move[k][lane+1];
               move[k][lane] = 0;
               move[k][lane+1] = 0;
          }
     }
/*   if (step%100==0)                    For changing density of added
     cars over time
          change = change + 1;
*/
}
     ave_speed = totspeed/totcount;
     printf("count = %8g     total speed = %8g  average speed = %g\n",
          totcount, totspeed, ave_speed);
     fclose(fptr);
     fclose(fptr2);
     return 0;
     }
```

REFERENCES

Chopard, B and M. Droz, 1998, *Cellular Automata Modeling of Physical Systems*, Cambridge University Press, Cambridge.

Chowdhury, D., L. Santen, and A. Schadschneider, 1999, “Statistical Physics of Vehicular Traffic and Some Related Systems”, *Physics Reports*, Vol. 329, pp. 199-329.

Gazis, D., 2002, *Traffic Theory*, Kluwer Academic Publishers, Boston.

Jia, B., R. Jiang, and Q. Wu, 2005, “The Effects of Accelerating Lane in the On-ramp System”, *Physica A*, Vol. 345, pp. 218-226.

Larraga, M. E., J. A. del Rio, L. Alvarez-lcaza, 2005, “Cellular Automata for One-Lane Traffic Flow Modeling”, *Transportation Research Part C*, Vol. 13, pp. 63-74.

Maerivoet, S. and B. De Moor, 2005, “Cellular Automata Models of Road Traffic”, *Physics Reports*, Vol. 419, pp. 1-64.

Mikhailov, A. S. and V. Calenbuhr, 2002, *From Cells to Societies: Models of Complex Coherent Action*, Springer, Berlin.

Schadschneider, A., 2000, “Statistical Physics of Traffic Flow”, *Physica A*, Vol. 285, pp. 101-120.

Schadschneider, A, 2002, “Traffic Flow: A Statistical Physics Point of View”, *Physica A*, Vol. 313, pp. 153-187.