Accurate Rapid Simulation of Urban Traffic using Discrete Modelling

Gordon Russell
Department of Computer Studies
Napier University, Craiglockhart, 219 Colinton Road, Edinburgh, EH14 1DJ, Scotland, UK

Paul Shaw
Department of Computer Science
University Of Strathclyde, 26 Richmond Street, Glasgow, G1 1XH, Scotland, UK

Neil Ferguson
Department of Civil Engineering
University Of Strathclyde, 26 Richmond Street, Glasgow, G1 1XH, Scotland, UK

Abstract

Increasing model complexity has traditionally been viewed as a key way of improving microscopic model accuracy. However, with complexity comes an increase in execution time. In some applications, such as UTC systems, low execution times and a high degree of accuracy are important design objectives. Discrete modelling can allow fast execution times to be attained, but this approach has always been viewed as an inaccurate approach to traffic simulation.

In this paper, we investigate the accuracy of the JUDGE model. This model was developed to be as simple and accurate as possible. Model complexities were only included where the improved accuracy could be justified. This has produced a simple, discrete modelling technique which can rival many traditionally derived microscopic models in accuracy terms, with a simulation speed for entire cities measurably faster than real-time. We give a high-level introduction to the proposed target application of the JUDGE model, its underlying structure, and a summary of some other research into high-speed modelling techniques. We then present out graphical interface to the JUDGE model, and compare results generated by JUDGE to results calculated from theory. We believe that these results show that the JUDGE modelling scheme is sufficiently accurate to be used as a slot-in replacement for many systems currently based on traditional (and significantly slower) microscopic modelling techniques.

1   Introduction

Urban Traffic Control (UTC) systems have an important role in meeting future transport objectives, especially with today's ever increasing congestion on our road networks. Adaptive control of roadspace over time is the principal advantage of using UTC systems. Such adaptive control can be achieved in a number of ways, including fixed pre-computed (based on a number of traffic scenarios) and traffic responsive (where the control policy can be changed dynamically within set constraints) schemes. Initially, this paper examines the adaptability of UTC systems in response to congestion, focusing on non-recurring congestion caused by incidents.

The remainder of this paper concentrates on the JUDGE system, developed by the authors. This is a simplified microscopic modelling scheme targeted towards a custom hardware implementation. The idea here is to use a model which is accurate enough to give acceptable results for large simulation studies (*e.g.* a simulation of a city over the next hour), with the results for the whole simulation accessible within a few seconds. Although targeted towards hardware, the current simulator runs completely in software, allowing the authors to make modifications to the design quickly and easily. Even in software, the JUDGE system can run a 1000 junction problem ten times faster than real-time. It is believed that with a hardware implementation of the simulation model, simulation times can be improved on by several orders of magnitude, and that this capability can be used to improve the performance of a UTC system in response to traffic congestion.

1.1 Existing UTC Systems

Fixed-time approaches to adaptive control use pre-calculated timing plans based on historical traffic data. Dynamic changes within a network are categorized, and used to select which of the plans to use at that time. As a last resort, an operator can specify signal timings manually. In general, there is no historical data for incidents, as these can occur anywhere in the network, and not necessarily at exactly the same time (or place) as any previous incident. Congestion can spread rapidly, and the longer the operator response time to an incident the harder it gets to circumvent the incident. The operator must be sure that his or her decision is correct, as it is possible to make a change to the network configuration which can make the situation worse.

With traffic responsive systems, signal settings can be altered automatically on a cycle by cycle basis [1] in response to current network loading. An example of a responsive system is SCOOT [5], which is utilized by many cities throughout the world. SCOOT also possesses features broadly similar to those of other responsive systems. In the presence of congestion, signal settings are altered based on simple queue management and gating rules. Where changes occur rapidly, SCOOT may not be able to adapt quickly enough to circumvent the congestion build up. There may also be a requirement to implement a novel pattern of signal settings over a large number of signals; this too is beyond SCOOT's capabilities [2]. Instead, the favoured approach is to use a pre-prepared SCOOT procedure [14]. However, such plans are not suitable for every situation, since the effects of the incidents and the alleviating strategies must be known in advance. This is not always possible given the uncertainties associated with incident-induced congestion.

1.2 Proposed Approach

In general, on detection of congestion, the operators of UTC systems will have the following three options. The choice between these options is dictated by the level of congestion, the level of spare capacity in the system, and the particular UTC and operations manual in use. The options are;

- Take no manual action, allowing the UTC system to handle the situation automatically.

- Implement a pre-prepared plan.

- Specify signal timings and other definable network stimuli (such as message boards) manually.

The results of operator decisions cannot be easily predicted in the majority of cases, and instead the operators rely on their experience and judgement. A superior control strategy may exist which the operators simply had insufficient time to devise or implement. It is suggested that, given the ability to examine the future state of a network quickly, given a number of possible decisions, and then to select the most appropriate decisions in the real-world network, would be of significant benefit to a UTC system.

In our proposed approach, the layout of the entire network of interest would be previously entered into a simulator. The simulator would have knowledge of the current state of the network, presumably through traffic monitoring devices such as detectors or video cameras. It would also have knowledge of the routes being followed by the vehicles in the network.

On the occurrence of an incident, an operator could then make a number of decisions, and have the simulator simulate the network from the current time to some future point (*e.g.* 15 minutes into the future). The simulator, which runs many times faster than real-time, simulates the network for all the different specified parameters and returns statistics on the future state of the network given those decisions. The operator needs only then select the most appropriate decision, either solely on the results of the simulations or using the results as a weighting factor in combination with other factors (*e.g.* experience). The operator could also amend the simulations, fine-tuning the decisions based on the initial results. With this knowledge it is suggested that an improved response model to traffic incidents would be obtained.

The simulation model would clearly have to both be reasonably accurate and be able to return useful information within a few seconds concerning the future state of the network. Such a simulation model is discussed in this paper, concentrating on the accuracy with the aid of example networks. This model, named JUDGE (JUst Designed to be Good Enough), is still the subject of ongoing research, and so the results presented here should be considered as preliminary.

Note that in the event of congestion some drivers will divert from their original routes. To provide an accurate description of the expected traffic flows requires an origin-destination matrix for the periods of interest, and an assignment procedure (such as that proposed in [13] which estimates route choices and the effect of unexpected congestion). The

---

[1] It is not normal to change signal settings during a cycle, as this can confuse drivers and may lead to accidents.
[2] SCOOT's reactions to congestion have been dampened to minimize undesirable signal oscillations.

current stage of our research, the JUDGE system does not possess a routing capability. This feature will be addressed in the next stage of the research.

1.3 Existing High-Speed Simulation Systems

The proposal for the JUDGE implementation involves using FPGA (Field-Programmable Gate Arrays) to further reduce the simulation over the current software-only implementation [2, 11]. This is intended to provide a cheap (in the range of $10000) hardware platform on which to simulate traffic. There are two other research teams which have also considered the problem of performing high-speed simulations of large networks; one in the University of Maryland [6], and the other in Edinburgh University [8]. Both these approaches are based on a Connection machine [4]. The cost of such hardware is in the multi-million dollars range, which is significantly more expensive than a JUDGE-based approach. It is also argued that the JUDGE system possesses a similar modelling accuracy to both of these two approaches.

The JUDGE system is based on discrete movement, as this is much easier to implement in hardware. Other models have also used a discrete movement approach, with the first in 1955 [7].

A newer example of a discrete movement model can be found in [3]. Here, an attempt is made to use one bit to identify the presence of vehicles, and masks and shifting operations to implement car movement. Although fast to execute, our experimentation has shown that most time is spend processing junctions (with their gap acceptance rules and route-following requirements). The work demonstrated in [3] does not really address the issue of junctions in any great depth. In addition, the freeflow speed control for roads is more complex than the JUDGE system for no real advantage, and its road capacity control is practically non-existant. It does however demonstrate the advantages of one bit discrete modelling over more traditional counterparts.

An example of discrete modelling for freeway traffic is described in [9]. This uses a significantly more complex modelling scheme than JUDGE, implementing a range of speeds per vehicle. From our experiments, the effect of maintaining speeds on a per vehicle basis can also be achieved by evaluating speed continuously based on headway and average freeflow speed for the road in question (which is the method used in the JUDGE model), and this approach is significantly computationally cheaper. Again, junctions are ignored by those authors. They do however conclude that the discrete modelling technique is not only fast to compute, but that is implements important aspects of fluid-dynamics which is also present in traffic flow in a natural way. They also note that this leads to better modelling scheme as a side-effect of this approach is to include some of the effects of perceived (statistical) driver behaviour.

2 The JUDGE Model

The basic car movement model used in the JUDGE approach is that of discrete movement. Here, continuous roadways are broken down into small cells, each of which can either hold a single pcu or be empty. Time is also quantized into ticks, and on the occurrence of a tick a vehicle moves instantly from its current cell to the one immediately ahead, provided that cell is currently empty. This type of movement can be seen in figure 1.
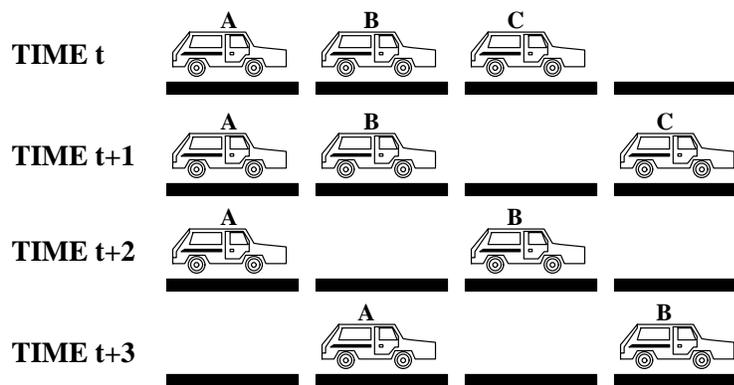

Figure 1: The basic discrete movement model used in JUDGE

With quantized time, synchronization of time-based events can only in general be approximated to the nearest tick. In the case of traffic sequencing in JUDGE, an accuracy to the nearest second was considered essential. Due to the way

that traffic control has been implemented in the current model, sequencing is rounded to the nearest even numbered tick, and thus a tick must be equivalent to 0.5 seconds to maintain the required accuracy. Another factor which contributed to this decision was that of maximum flow; with a tick of 1 second, the maximum flow through a roadway was only 1800 pcu/hr. This was deemed insufficient, as urban roads have been classified up to 2500 pcu/hr, and in some literature freeways with 2500 pcu/hr maximum flows are sometimes considered in simulations. With a tick of 0.5 seconds, the maximum saturation flow which can be implemented is 3600 pcu/hr, although the interface to the simulator currently limits the selectable range to a maximum of 2500 pcu/hr.

Each cell is defined to be equivalent to 6.0 metres of roadway. This size is selected to match expected pcu space utilization when within a stationary queue. These two constants define the free flow speed of vehicles in figure 1 as 43.2 km/h. While this is acceptable as a minimum possible selectable free flow speed for a road, higher speeds should also be possible.

To obtain faster speeds than 43.2 km/h, we introduce the notion of *fast cells*. These cells are effectively teleporters, such that if a vehicle wished to enter a fast cell in any one tick, and that both the fast cell and the next cell is empty, then the vehicle will effectively teleport over the fast cell. If teleportation is not possible, then the fast cell acts as a normal cell instead. Multiple fast cells can be strung together, permitting the possibility of teleporting a vehicle across multiple cells in a single tick. An example of fast and normal cells in operation can be found in figure 2.
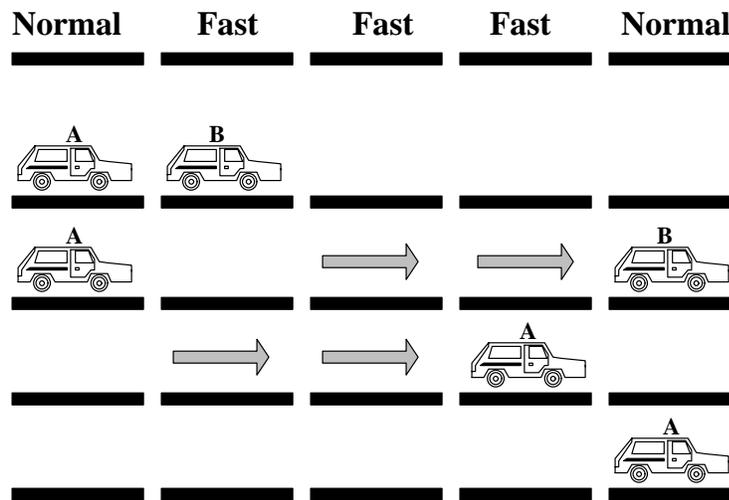

Figure 2: How speeds above the minimum free flow speed of the model are implemented

2.1 Flow Control

Speeds are not the only aspect of roadways which must be implemented within the JUDGE model; flow control must also be present. There are two areas of flow control which are of interest; road capacity, and the slow decline of average speed as the flow increases between zero and the point where vehicles begin to enter stop-start queueing. We term this area of gradual average speed reduction as the *first regime*, taken from the name of the same area found in descriptions of the three-regime speed/flow model. These two areas of flow control are tackled by separate components.

For flow control in the first regime, the model used to implement roadways effectively increases travel times of vehicles by up to 1 second as the actual loading on the roadway increases. This is achieved by dynamically configuring a number of cells (known as the pseudo area) in the roadway as either slow cells or fast cells in response to traffic loading. At high loadings, the selected number of cells are configured as slow cells, while light loadings configure the cells as slow cells. Intermediate loadings use a combination of fast cells and slow cells. To preserve average roadspeed, the speed is configured with all cells in the pseudo area set to fast cells. If all the cells in the pseudo area are set to be slow cells, then the travel time for a free flow vehicle over the entire roadway would be reduced by 1 second.

To configure the capacity of a roadway, the flows of vehicles both entering and leaving a roadway are carefully controlled. This control mechanism monitors vehicle headways of entering and leaving flows. If the headways exceed those expected for the selected capacities of that roadway, delays are introduced. Since there is no possibility of incidents

within roadways (as far as the model is concerned), no capacity control is needed except at the start (to prevent roads from being over-saturated) and at the end (to control queue dissipation).

2.2   Junctions

A junction can connect multiple roadways together, and can contain multiple signals. All junctions are constructed from a mosaic of 6 metre squares, which are equivalent to normal cells of the roadways except that cars can enter and leave a cell from any of the four directions. Each input roadway can have a number of routes to follow across the surface of the junction, each of which must connect to an output roadway. A probability distribution must also be supplied, indicating the chance of a car from a particular input road following any of the specified routes. Other factors can also be specified, including the priorities of routes where a route crosses other routes, and if there are any squares in the junction which cars should not be permitted to stand on (equivalent to hatched junction areas). All vehicles look along interfering vehicle streams for a specified gap acceptance before crossing such a stream, taking into account any additional time required so as to not stop on a hatched square.

2.3   Source and Sink Components

Finally, the input and maximum output flows of the network can be configured using random distributions of vehicles. A different distribution can be specified for each five minute period of a simulation, independently for each source and sink component. Care should be taken so that queues of vehicles do not tail back into source components, otherwise the specified input flows may not be achieved.

3   Simulator Interface

Our simulation model is accessed through a graphical interface, constructed with Tcl/Tk [10]. The development environment is Unix/X11, but the use of Tk should also allow the simulator to run under Windows. C++ was used to support the Tk interface where speed or complex data handling techniques were required, and C was used in the core of the simulation model for performance reasons. The basis view of the interface can be found in figure 3.
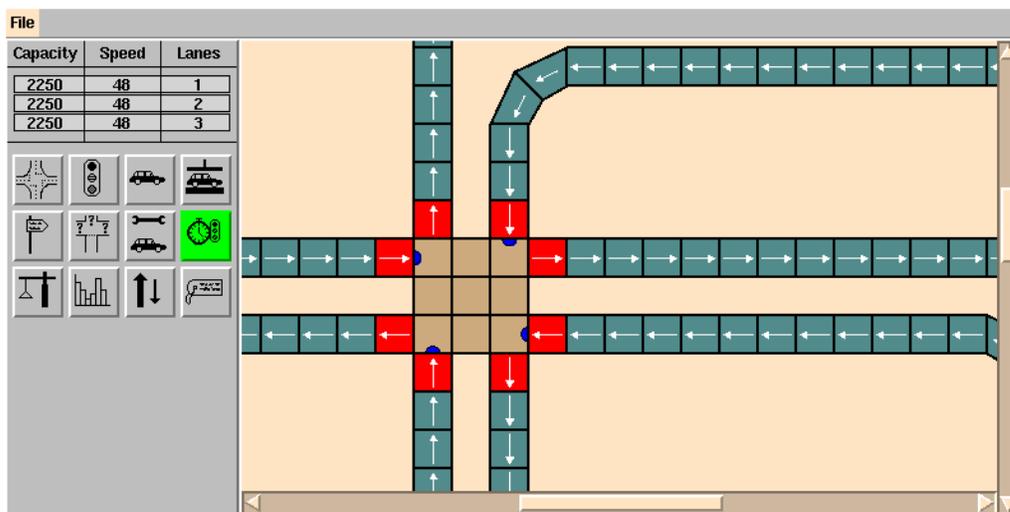

Figure 3: A screen dump of the JUDGE graphical interface

Using the interface, traffic networks are built up with a model-railway type of approach. The basic building block is a road cell, and multiple road cells can be placed end-to-end to construct roads. The roadways can each have their own maximum speeds and capacities. A road can join another road by simple abutment, or, where more complex joins are required, at a junction. Junctions are constructed by laying down junction squares, which form a mosaic covering the whole junction surface. A roadway can join to any edge of a junction again by simple abutment. Traffic signals can be placed on any edge of a junction square, and affect all traffic crossing that edge independently of the direction of travel.

All roadways must have a connection at each end to another component; a junction square, a source component, or a sink component. Source components generate vehicles with a flow specified by the user, and sink components remove vehicles at a user-definable rate. Both source and sink rates can be configured per five minutes of the simulation.

Other characteristics of the network must be specified in addition to its geometric layout:

- Each route which a car can follow across a junction must be specified. This is done by clicking on the input and output roadway lane, and then by clicking on the intervening squares which a car would travel over.

- Each junction square can be marked as *normal*, or as a *non-stopping* square. In a non-stopping square, no car can wait. Thus cars will use knowledge of surrounding traffic flows and junction routes to avoid having to stop in a non-stopping square.

- Each input roadway must be given a list of probabilities as to the distribution of vehicles which will follow each route from that roadway.

- Any traffic signal can be linked to any other signal in that junction to form signal groups, and thus share sequencing information.

- For each traffic signal group, green and red signalling can be controlled on a per-second basis.

Although the simulator permits multiple lanes to be entered for each roadway, its implementation is still under investigation. The examples in this paper therefore concentrate only on roadways consisting of a single lane.

3.1 Multiple Configurations

To allow different characteristics for a network to be readily compared, multiple characteristics can be defined for a single component. For example, in the sequencing configuration window depicted in figure 4, there are 139 different configurations noted in the bottom-left of the window. Each can be stepped through via the next and prev buttons, with the current entry being deleted with the delete button. The current entry can be duplicated with the add button, allowing one of the duplicates to be modified to provide a different set of characteristics.
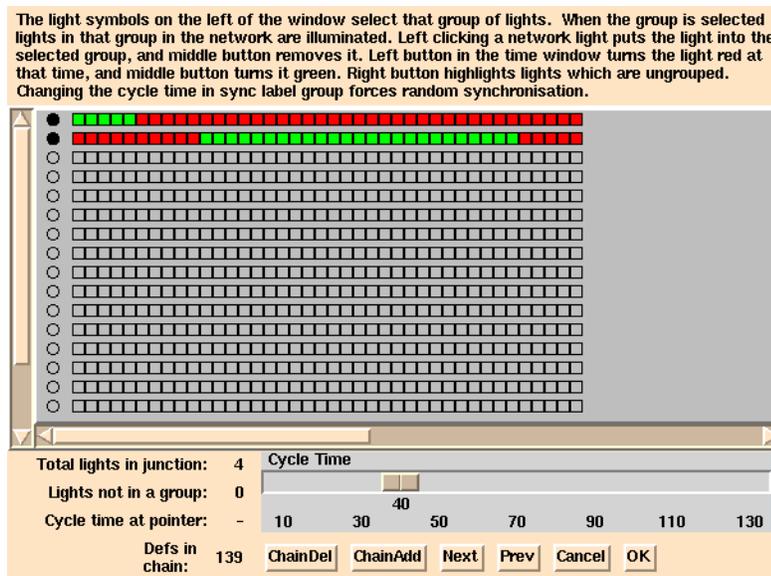

Figure 4: Screen dump of the signal sequencing configuration window

The sequencing of signals is only one of four components which can be configured for multiple characteristics. The source and sink components can each contain a number of different flows. Each roadway can also be configured for a variety of speeds and capacities.

During simulations of the network, each combination of the lists of characteristics are used to generate a network, which is simulated independently from all other combinations. The results of each simulation is collated, and can be examined independently or collectively once all simulations have been completed.

3.2 Measurements

In order to evaluate particular traffic networks, some mechanism for measuring traffic characteristics during simulations is essential. These measurements can then collated for post-simulation analysis. A number of characteristics can be obtained with the current simulator; average density, average speed, average flow, average queue length, and average delay per vehicle. The averages are calculated over user-definable periods throughout the simulation runs. Any number of roadways can be selected graphically for analysis.

Figure 5 shows the program interface during the configuration of the roads to analyse during subsequent simulation runs. Basically, clicking on a roadway adds it to a list of links, which can be stepped through using the link manipulation buttons. The characteristics to be measured can be toggled for each roadway selected.
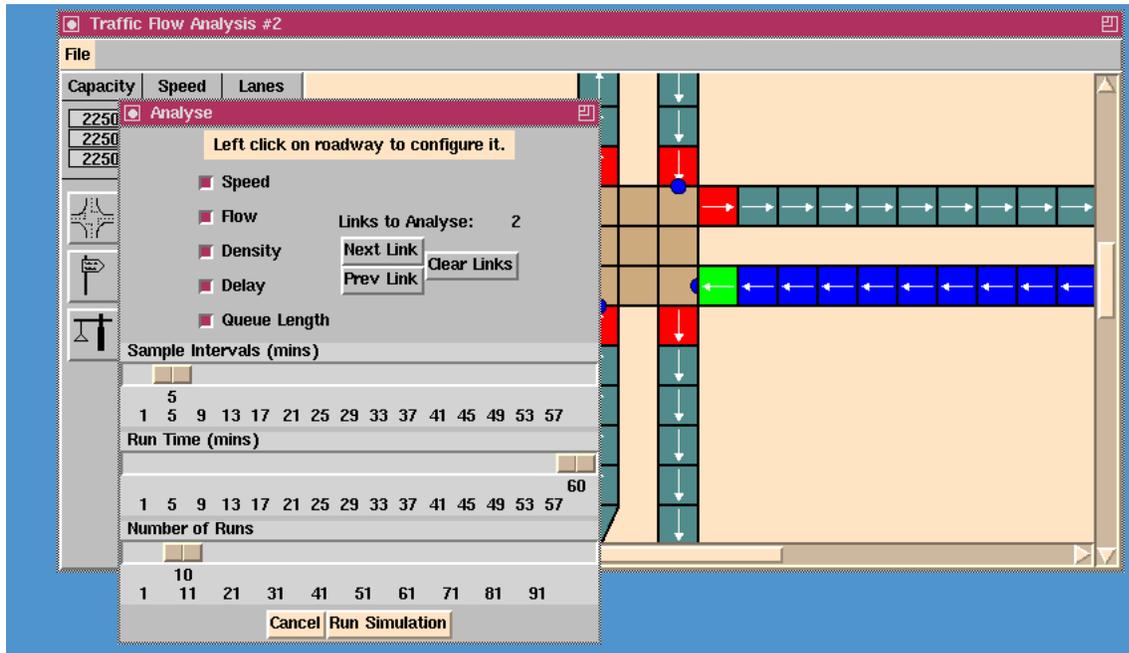

Figure 5: Screendump of the Simulator Showing the Analysis Window.

Each of the characteristics are measured in a way convenient for the simulator;

Flow: This is measured at the exit point of the roadway.

Density: The density is measured over the entire length of the roadway.

Speed: This is calculated from flow over density.

Delay: The average travel time for each vehicle in the roadway is measured, and the travel time at freeflow subtracted away. This is averaged over the sample period.

Queue Length: A queue is considered to exist when a car has stopped at the end of a roadway for five seconds. The queue length is then taken to be the length of the queue when the car at the end of the roadway is just about to move off. Again, this maximum is averaged over the sample period.

3.3 Results

Once all runs have been completed, the results can either be viewed from within the program or saved to disk for later investigation. The interface provides graphing functions to view any of the runs individually, graphing the data on 2D line graphs. The x- and y-axis of each graph can be designated by the user from a selection of any of the characteristics selected for analysis. In addition, all runs made in a set of simulations can be viewed in summary charts. All the result graphs shown in the examples have been created from external applications using data constructed from results saved to disk.

4 Road Characteristics

The basic roadway characteristics of the JUDGE model are examined in this section. A simple roadway bottleneck has been set up (as in figure 6), where a long roadway, which is split into two lengths **A** and **B** consisting of a 2400 pcu/hr capacity connects directly to an 1600 pcu/hr capacity roadway **C**, which eventually leads on to another 2400 pcu/hr capacity roadway **D**. By manipulating the traffic loading on the network, we should be able to observe freeflow traffic and stop-start queueing from the point where the bottleneck roadway **C** is entered back through **B** and **A**. The results of an hour of simulation of this network can also be found in figure 6.
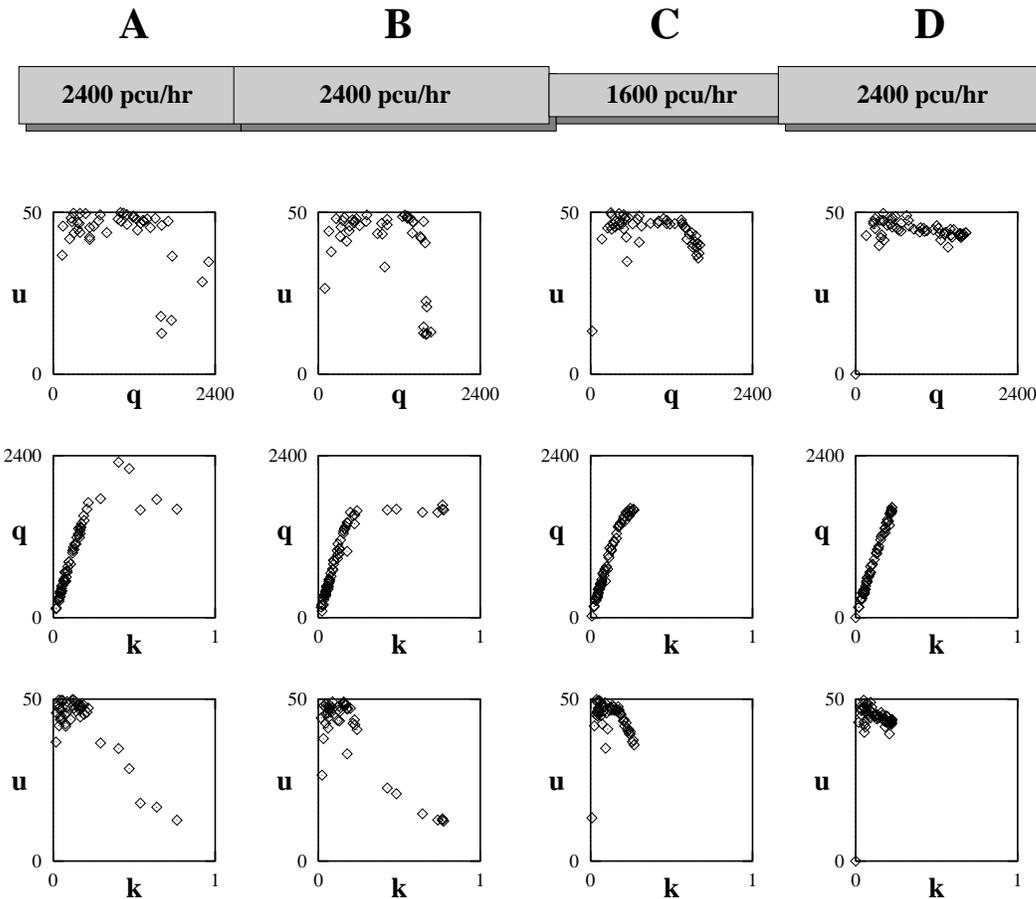

Figure 6: The bottleneck network used to investigate roadway characteristics in JUDGE

The input flow was carefully controlled so that it could be demonstrated that **A** really could handle a flow of 2400 pcu/hr. Clearly, this is well beyond the capacity of the bottleneck section, and a large queue was formed. Before the queue could dissipate, the end of the queue reached back into **A**. After a few minutes, the queue length was reduced and was completely held within **B**. The evidence of this can be seen in the speed against flow graph of **A**. It can be seen that the flow did reach 2400 pcu/hr, with a speed of about 35 km/hr. The measurements were taken when the queue were present can be found where the speed was about 15 km/hr; at this point the flow was only 1600 pcu/hr, which was the capacity of the bottleneck (the governing constant for the queue dissipation rate).

In **B**, where stop/start queueing occurred, flows are strictly limited to below 1600 pcu/hr, due to the effects of the bottleneck. Short-duration queues form regularly with the change in headway between roads **B** and **C**. This causes a variation in the average speed of cars in **B** of anything up to 10 km/hr.

Vehicles within **C** place a heavy loading on that particular roadway. Here one can see the knee in the speed against flow graph where the flow has reached a sufficiently high value to cause the speed to curve down. The roadway is never over-saturated, as the vehicles which would have caused the capacity to be exceeded are queued back in **B**.

Finally, when cars reach **D**, their maximum flow is only 1600 pcu/hr, which was the capacity of roadway **C**. This

flow is not sufficiently near the capacity of **D** to cause the knee in the speed/flow graph which was present in **C**.

The results gathered in this experiment appear (*e.g.* by comparing them visually to the results indicated throughout [12]) to match those produced by both theory and real-world experiments. This would suggest that the road characteristics supported in the JUDGE model is sufficient for meaningful modelling of roadways. In the next section we examine our accuracy for junction modelling.

5   Comparison to Theory

The accuracy of simulations using the JUDGE model is hard to estimate, since even comparing simulations of a real junction against its real-world situation can only at best give accuracy estimates for that single junction design and traffic flow. A comparison between JUDGE and 100% accurate theoretical equations would be ideal, but as yet no such equations have been developed.

Instead, the best that can be achieved is to compare JUDGE against a reasonably well accepted theoretical model, and to bear in mind that the theory itself may not be particularly accurate. We have chosen to compare our simulator against Akcelik [1]. This gives the average delay per vehicle for one approach road connected to a signalized junction as;

$$d = \frac{c(1 - \lambda)^2}{2(1 - \lambda X)} + \frac{N_o X}{q} \tag{1}$$

where

$$N_o = \begin{cases} \frac{Qt}{4} \left(z + \sqrt{z^2 + \frac{12(X - X_o)}{Qt}} \right) & \text{for } X > X_o \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$
\begin{aligned}
z &= X - 1 \\
c &= \text{cycle time} \\
\lambda &= \text{proportion of cycle effectively green} = g/c \\
t &= \text{simulation time} \\
q &= \text{flow in vehicles/sec} \\
g &= \text{effective green time for this arm} \\
Q &= \text{capacity} = \frac{sg}{c} \\
s &= \text{saturation flow in vehicles/sec} \\
X &= \text{degree of saturation} = \frac{q}{\lambda s} \\
X_o &= 0.67 + \frac{sg}{600}
\end{aligned}
$$

In our experiments, we have implemented a four-arm two-phase signalized junction with no right-turning traffic and balanced flows on each arm. Each phase is of identical length, and the lost time is 10 seconds. We plot measured delay as percentage error of the delay given by the simulator against d from equation 2. Note that the error is considered zero if the actual difference between simulation-generated delay and Akcelik delay is less than 4 seconds. This avoids spuriously high percentage errors where the delays in question are very small (*e.g.* for a simulated delay was 2.0 s, Akcelik 0.012, percentage error would be almost 17000%). In the graphs, $0.2 \leq X \leq 2.5$. During the experiments, we vary

c	$\{40, 60, 80, 100\}$ seconds
Q	$\{1200, 1800\}$ pcu/hr
t	$\{5, 15, 25\}$ minutes
q	$\{300, 500, 700, 900\}$ pcu/hr
S (max roadspeed)	$\{48, 64, 96.6\}$ km/hr

Roadspeed does not come into the Akcelik equation, suggesting that the speed of vehicles has no effect on the simulation results, and thus varying this parameter should have no effect on our results either (and therefore should be a good test of how well we have implemented vehicle speed in JUDGE).

Equation 2 is known to have a number of inaccuracies, which mostly manifest themselves as evaluating significantly too much delay in the approximate region of $0.7 \leq X \leq 0.95$. Thus in our analysis of the results, we would expect to see a significantly negative delay error in that region. Note also that headways in Akcelik are considered uniform, whereas the simulator uses randomly distributed headways. This too may have an effect on the accuracy of the results, although we attempted to minimise the effect of this by running each set of parameters 10 times in the simulator and

9

taking the average delay. More runs could have smoothed some of the plots, but it would have taken proportionally longer to produce the results.

Note that Akcelik considers that the flow of vehicles has reached the junction at the start of the simulation period. Our simulator does not preload the roadway with a distribution of cars equal to that expected from the input flow. Thus we start with an empty approach road, and delay the start of the measurements until enough time has elapsed for the first car to reach the junction (the run-in period). This time is calculated from the length of the approach arm and its maximum speed. For our simulations, we needed an approach arm sufficient to hold the maximum possible queue of vehicles, and thus opted for an arm of approximately 2 km in length. Our simulator also considers all delays which effect a vehicle on the approach arm, such as that caused by non-queueing vehicles interacting with each other during normal movement (generally amounting to a few seconds per vehicle), and since Akcelik does not take this into account an attempt is made to remove this effect from the results. The simulation results amount to 48 graphs, each consisting of 6 individual datasets. This is too much data to include in this publication, so only selected results are shown here.

Where the runtime t was 25 mins, the error graphs all showed excellent correlation to Akcelik, except in the expected region of Akcelik inaccuracy, where the error percentage does bulge significantly negative as predicted. Before $X = 0.7$, the simulator shows excellent accuracy, while after about $X = 1.0$ the error percentage was generally within 5% of Akcelik. An example graph can be found in figure 7.
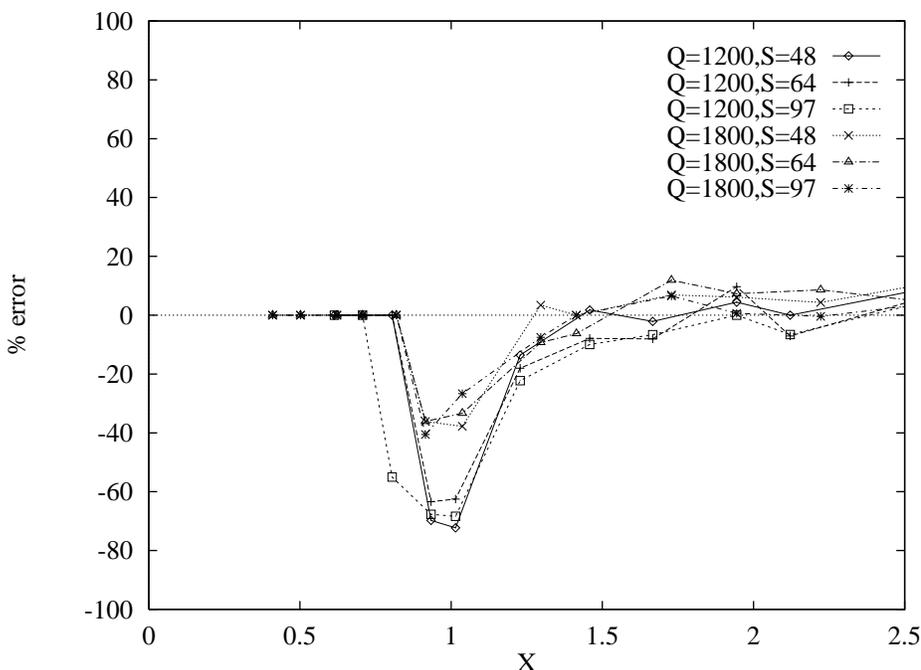

Figure 7: An error plot with q=700,c=40 and a runtime of 25 minutes

We would predict that as the runtime is reduced to 15 and 5 minutes, then the results would become progressively less accurate. This is due to two main factors; less runtime gives less time to average out the delays, and the effect of not starting the simulation measurements at the beginning of a light cycle (due to the run-in period) and terminating without running a simulation for a whole number of cycle periods.

For a runtime of 15 minutes, an example of the errors can be found in figure 8. This has a similar trend to the graphs produced with runtimes of 25 minutes, but in the oversaturated parts of the graph ($X > 1.0$) the errors vary more between 0 and 10% than the -5% to 5% of the previous graph. However, the results are on the whole still reasonably accurate.

Finally, figure 9 shows an example of the results with the runtime of only 5 minutes. The results are significantly worse than for either the 15 or 25 minute runtime experiments, especially once the junction is saturated. Note that there is 80% less run time in comparison to the 25 minute experiments to obtain delay averages. The standard deviations of the averages used in this graph are significantly higher than in either the 15 or 25 minute examples. Additionally, we measure the flow at only one point on the approach arm, and the algorithm becomes confused with small delays. This effects both direct delay measurements and the delay caused by vehicles interacting with each other within the non-
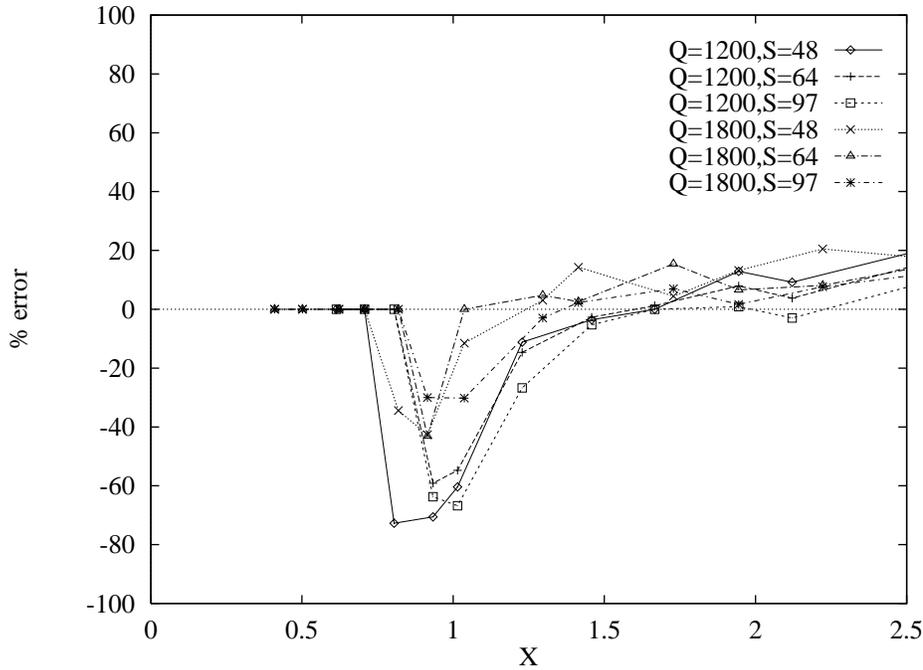
10

Figure 8: An error plot with q=700,c=40 and a runtime of 15 minutes

queueing areas of the approach arm (which as indicated earlier is subtracted away from the direct delay measurements to allow sensible comparison to Akcelik). Even so, the use of such small run times could lend itself to the first stages of a signal optimization algorithm, for example where only approximate delays are required for the first few iterations of a search method based on Genetic Algorithms.

In the cases of runtimes of 25 and 15 minutes, it was seen that the effect of modifying either capacity or maximum roadspeed has largely negligible. All the results suggest that the JUDGE modelling technique is an excellent way of simulating traffic rapidly, without necessarially compromising accuracy.

6 Conclusions

The JUDGE modelling approach to traffic simulation permits fast predictions to be obtained on the state of a network. Such information can be used to modify network parameters (*e.g.* traffic signal timing) in real-time, helping to minimize traffic delays, especially in response to non-recurring incidents.

An interface has been constructed around JUDGE. This allows traffic networks to be entered graphically into the simulator, and then modelled over a user-definable time period. Network characteristics are fully controllable by the user, and traffic statistics can be measured during simulations for subsequent storage and analysis.

A number of experimental networks were entered into JUDGE, in an attempt to measure the accuracy of this modelling technique. Accuracy was measured by comparing a known theoretical model of traffic against the results obtained during simulations. This theoretical model, Akcelik, has known deficiencies, but by taking these into account it was hoped that a meaningful comparison could be made. This comparison suggests that the JUDGE modelling scheme simulates traffic in a way which closely correlates with known theory, and thus that the JUDGE approach to traffic simulation provides a reasonably accurate and fast method of urban traffic modelling.

In the future we will continue work on the improving the accuracy of the JUDGE modelling scheme. We are also looking at dynamic routing of vehicles across the network, and at Genetic Algorithms to modify traffic signal characteristics. Finally, we are continuing to work on mapping our simulation model onto reprogrammable hardware devices (FPGAs), in an attempt to perform simulations orders-of-magnitude faster than that possible with software-based approaches, while still offering a low-cost solution to traffic simulation.
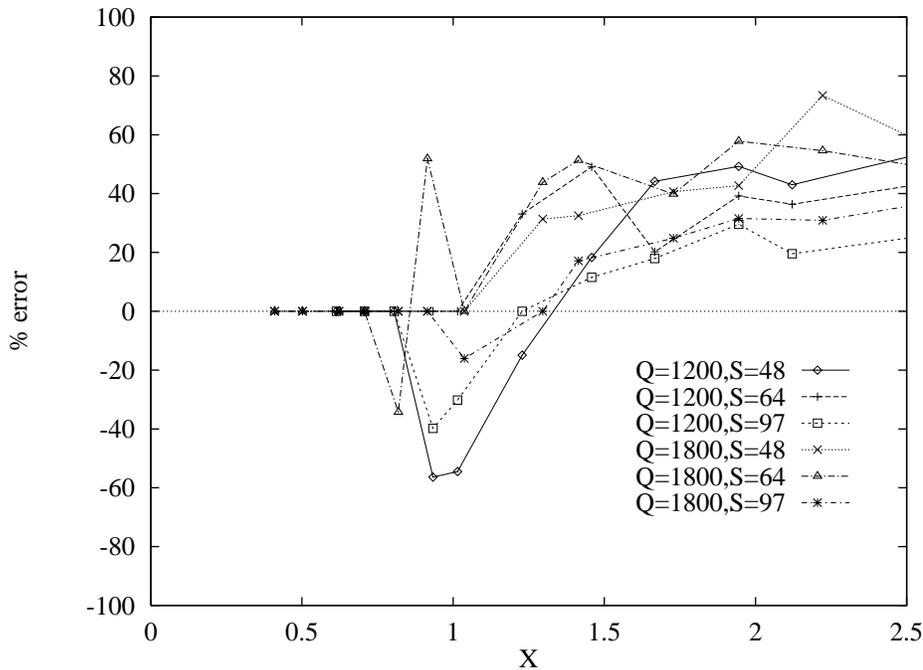
Figure 9: An error plot with q=700,c=40 and a runtime of 5 minutes

References

[1] R. Akcelik. Time-dependent expressions for delay; stop rate and queue length at traffic signals. Technical Report AIR 367-1, ARRB, 1980.

[2] Martin Bate, Alex Cowie, George Milne, and Gordon Russell. Process algebras and the rapid simulation of highly concurrent systems. In *Proceedings of the 18th Australasian Computer Science Conference*, February 1995.

[3] M. Cremer and J. Ludwig. A fast simulation model for traffic flow on the basis of boolean operations. *Mathematics and Computers in Simulation*, (28):297–303, 1986.

[4] W. Daniel Hillis. *The Connection Machine*. MIT Press, 1985.

[5] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton. SCOOT - a traffic responsive method of co-ordinating signals. *TRRL Laboratory Report*, (1041), 1981.

[6] Thanavat Junchaya and Gang-Len Chang. Exploring real-time traffic simulation with massively concurrent parallel computing architecture. *Transportation Research - C*, 1(1):57–76, 1993.

[7] J. H. Mathewson, D. L. Trautman, and G. L. Gerlough. Study of traffic flow by simulation. In *Highway Research Boards Proceedings*, 1995.

[8] Dr David McArthur, G. D. B. Cameron, M. D. White, and B. J. N. Wylie. Paramics: Parallel microscopic traffic simulator. Available from Dave McArthur, EPCC, Room 2412, JCMB, King's Buildings, Edinburgh, January 1994.

[9] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journale de Physique*, 1(2):2221–2229, December 1992.

[10] John Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1993. ISBN 0-201-63337-X.

[11] Gordon Russell, Paul Shaw, John McInnes, Neil Ferguson, and George Milne. The rapid simulation of urban traffic using field-programmable gate arrays. In *Proceedings of the International Conference on the Application of New Technologies to Transport Systems*. Australasian Road Research Board Ltd, May 1995.

[12] R. J. Salter. *Highway Traffic Analysis and Design*. MacMillan Education, second edition, 1990.

[13] T. Van Vuren and D. Leonard. Urban congestion caused by incidents. *Traffic Engineering and Control*, 35(7/8):422–429, 1994.

[14] K. Wood. Urban traffic control, systems review. *Transport Reserch Laboratory Project Report*, (41), 1993.