

# Advancing the Frontier of Simulation as a Capacity and Quality of Service Analysis Tool

**GEORGE LIST**

*Department of Civil Engineering  
Rensselaer Polytechnic Institute, Troy, NY, USA*

**ROD TROUTBECK**

*School of Civil Engineering  
Queensland University of Technology  
Brisbane, Queensland, Australia*

## ABSTRACT

We address issues related to the use of simulation models to assess highway capacity and quality of service. This topic is of great interest currently given the growing trend toward the use of simulation for such purposes. Descriptive paradigms, continuity, clarity, reliability, repeatability, and consistency with theory are among the issues examined. Four paradigms for describing simulation models are presented along with their relative advantages. We conclude by anticipating directions in which this field of investigation will proceed in the future.

## 1. INTRODUCTION

Simulation, from microscopic through macroscopic, is increasingly becoming a popular paradigm for analyzing traffic operations and highway capacity. Many of the main traffic analysis tools in use today, like CORSIM (Kaman Sciences Corporation 1996), INTEGRATION (Van Aerde 1993), WATSIM (Lieberman 1991), EVIPAS (Bullen et al. 1987), and TEXAS (Lee and Fong-Ping 1981) are simulation models. CORSIM, INTEGRATION, and WATSIM are models for entire networks of freeways and surface arterials. EVIPAS and TEXAS focuses on single intersections. Athanailos (1994), Kuhne and Rodiger (1991), Kyte et al. (1996), List and DiCesare (1998), Troutbeck (1993), Ran et al. (1998), and Stewart et al. (1996) are but a few examples. More can be found in the references. (We have not focused on pedestrian, bicycle, and transit capacity issues; those are important domains as well. Our comments are equally applicable, but we have not provided examples.)

Trends in the *Highway Capacity Manual* (Transportation Research Board 1997) provide evidence of simulation's impacts. In many cases, the methodologies presented in the 1994 and 1997 updates, while still analytical procedures, have used simulation-based analyses, augmenting field observations, to provide the justification for the methodology and provide the rationale for the default parameter values suggested.

The problem with most if not all simulation models is that they are not well documented. Typically, no readily accessible text describes their logic in a clear and unambiguous

manner. Calibration and validation are also major challenges, as is the identification of locations where high-quality validation tests can be conducted.

While everyone appreciates the difficulties of calibration and validation, the lack of documentation is also understandable. Simulation models are constantly being improved. Any documentation that might be developed would be constantly be out-of-date and in need of revision. When up-to-date documentation is provided, progress is slower and far more expensive. Only when strict software development guidelines are employed (e.g., ISO-9000), is it likely that the documentation faithfully represents the logic actually used by the software.

The lack of documentation keeps universal acceptance from occurring. It makes the models look like “black boxes.” No one knows for sure how they work. It is difficult to accept them as the “benchmark” for any kind of analysis, especially when the outputs have legal standing or implications. Users must trust that the simulation model faithfully replicates the environment it represents. There is no way to “open the hood” and see what actually takes place.

This healthy skepticism has made the Highway Capacity and Quality of Service Committee of the Transportation Research Board hesitant to adopt any simulation based model for the prediction of highway capacity or quality of service. While the committee recognizes the value of doing so, and understands that such a paradigm needs to be found, the committee has not ascertained what that paradigm should be. While the HCM-2000 will contain a new chapter devoted to simulation and other models, that presents examples of simulation models, it does not in any way discuss the type of descriptive paradigms that could be employed.

This paper presents four ideas about how such documentation could be provided. Four are based on classical paradigms: program code, flowcharts, spreadsheets, and pseudo code. The fourth is a newer idea, based on Petri Nets (PN), which has promise because of its ability to be both clear and concise.

Whatever paradigm(s) are eventually employed, one challenging test must be passed. Completely independent software developers must be able to create commercial packages which implement the simulation-based methodology in an unequivocal fashion. It must be possible for those programs to obtain exactly the same results for the same input data. This means developers must be able to agree that the procedures they have implemented are identical at the input-to-output level. This test sets a very high bar for the clarity and preciseness of the documentation paradigm.

In the case of simulation models what makes the test even more challenging is that this identical match requirement must be met in spite of the fact that stochasticity is involved. Simulation, in its most common implementation, involves Monte Carlo simulation, and that implies the use of random numbers, random number series and random variables. The input data must also specify the random number series if the input is to be identical.

As time progresses, it is not a question of whether, but how simulation will be used. It is clear that such models will be part of the next generation of capacity analysis tools.

Unsignalized intersections, roundabouts, signalized intersections, and two-lane highways are very likely instances where such models will be employed. The real question is how they will be described, so that this lack-of-ambiguity test can be met and universal understanding of the procedure can be assured.

## 2. PROCESS PARADIGMS

Early simulation packages such as GPSS (Banks et al. 1989), SIMSCRIPT (CACI Products Company 1994), and SIMAN (Pegden et al. 1990) all had, and continue to have “text-and-graphics” formats for creating models. ARENA (Systems Modeling Corporation 1994), a more recent tool (built on SIMAN), uses a very graphical user interface (GUI) to assist in model creation. The problem with these paradigms is that they involve a lot of “behind-the-scenes” technical detail and the schema are nominally proprietary.

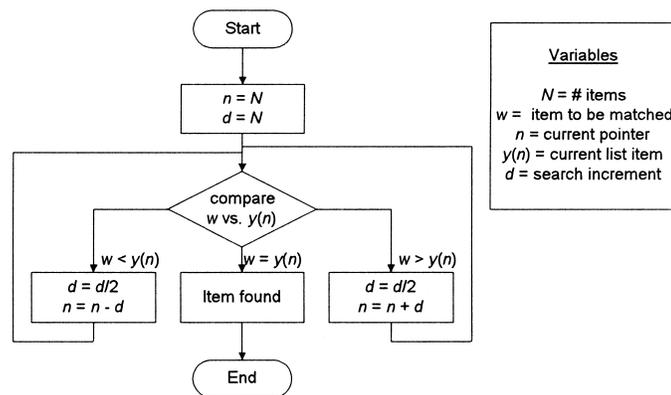
Other paradigms have been developed through formal research. Petri (1962) developed a graphical paradigm, now called a Petri Net, to describe the event-to-event logic pertaining to state machines, functionally very similar to discrete event simulation. Schafer and LaRue (1994) present a “block oriented network simulator” for describing discrete event simulations. It closely parallels the Petri Net paradigm.

Textbooks about modeling and simulation use paradigms to describe the models they discuss. Bossel (1994) uses a graphical paradigm that is carefully styled to effectively and efficiently describe coupled differential equations. Sedgewick (1988) uses pseudo code to describe the algorithms presented.

Our goal in this section is to present five paradigms by which simulation models could be presented. Four are derived from classical roots: flowcharts, spreadsheets, pseudo code, and computer code. The fifth is based on the Petri Net paradigm, a popular choice in other modeling environments.

### 2.1 Flowchart

A flowchart is the easiest idea to present and it is probably the oldest. Programmers of 30 years ago learned to create flowcharts before developing code. Figure 1 presents the flowchart for a binary search routine.  $N$  is the number of items in the sorted list (e.g., vehicle types),  $w$  is the item being sought (e.g., 2-axle, 6-tire).  $n$  is the pointer to the current item being compared and

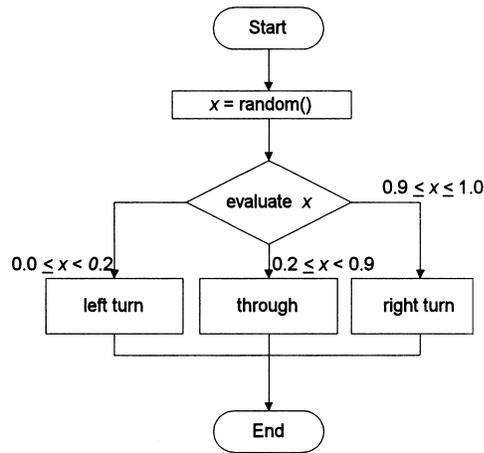


**FIGURE 1** A simple flowchart.

$y(n)$  is the value of that item (e.g.,  $n = N$  and  $y(n)$  = triple-bottom tractor trailer).  $d$  is the increment by which  $n$  is to be changed based on the outcome of the comparison test. If  $w = y(n)$ , then a match has been found and the search terminates. If  $w < y(n)$ , then the item occurs earlier in the list, and  $n$  must be decreased. If  $w > y(n)$ , then  $n$  must be increased.

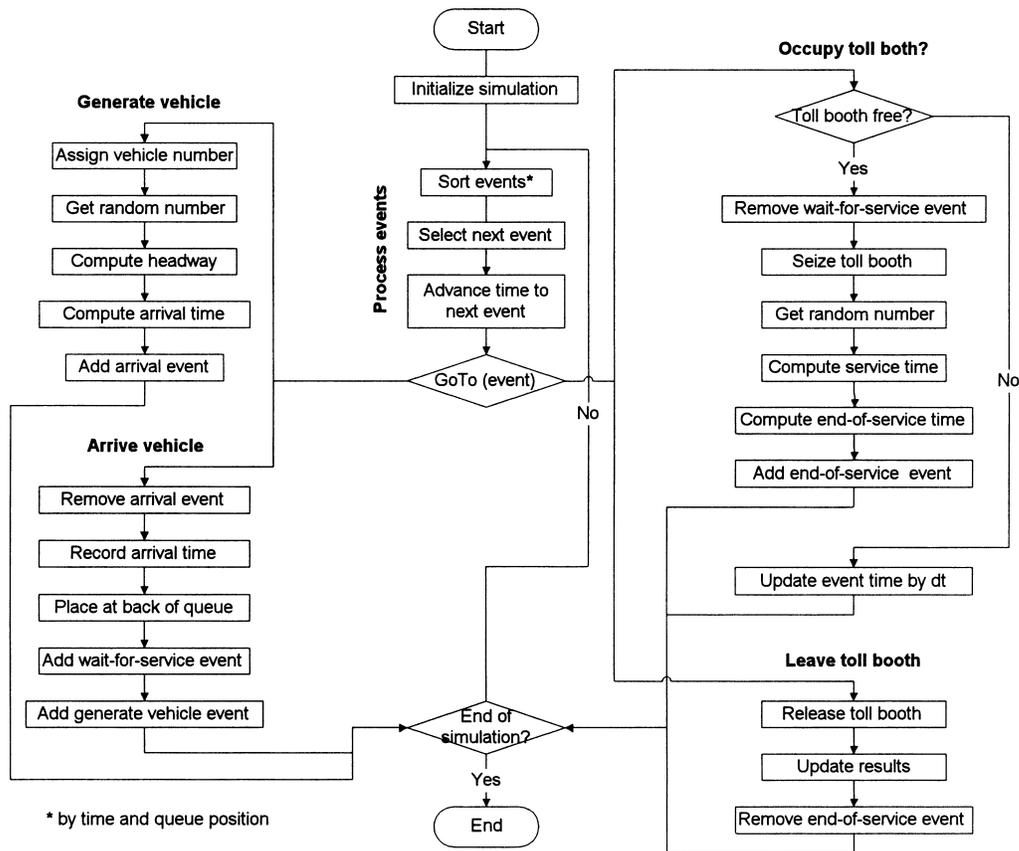
The clever part of this procedure is that very few tests are performed.  $n$  is always updated to a value exactly in the middle of the range that still needs to be searched.

Figure 2 presents a flowchart for turning movement selection as it might appear in a traffic simulation model. A random number  $x$  is generated from a uniform distribution ranging from 0 to 1. That number is then evaluated and one of three actions is taken. If  $x$  is between 0.0 and 0.2, the vehicle makes a left turn. Between 0.2 and 0.9, it goes straight ahead, and between 0.9 and 1.0 it turns right. Thus, 20% of the approaching vehicles will be left turns, 70% will go straight ahead, and 10% will make a right turn.



**FIGURE 2 Turn selection flowchart.**

These ideas can be extended to represent entire traffic operation situations. Figure 3 shows the flowchart for a toll booth model. At the top middle is the “start” block where simulation begins. The first step that takes place is initialization, parameter values are set and a generate vehicle event is placed in the event queue.



**FIGURE 3 Toll booth simulation.**

We can continue the discussion by considering the generate vehicle event for any arbitrarily selected vehicle.

Let's assume a "generate vehicle" event is first in the event queue. At the GoTo block control then passes to the "generate vehicle" step sequence. A new vehicle number is assigned, a random number is generated, a headway is computed as well as an arrival time (current time plus the headway), and an "arrival event" is added to the event list.

Control passes back to the "process events" step sequence, a check is made to see if the "end of simulation" has arrived (maximum vehicles, maximum time, etc.) and if not, the process repeats.

At a future juncture, let's assume the "vehicle arrival" event emerges at the top of the event list. Time advances to that event and control passes to the "arrive vehicle" step sequence. The arrival event is removed, the time of arrival is recorded, the vehicle is placed at the back of the current queue, a "wait-for-service" event is added to the event list, and a "generate vehicle" event is added (to continue the vehicle generation process).

At some later point in the simulation, the "wait-for-service" event reaches lead position in the event queue. Control passes to the "occupy toll booth?" step sequence. A test is made to see if the toll booth is free. If it is not, the event time (when to consider the event next) is incremented by a minimal amount ( $dt$ ) and the event is placed back in the event queue (which is resorted by event time and queue position). If the toll booth is free, control passes to the sequence of steps below the test. The wait-for-service event is removed, the toll booth is seized, a random number is obtained, a service time is computed, the end-of-service time is obtained, and an end-of-service event is placed in the event queue.

Finally, when the "end-of-service" event emerges at first position in the event queue, the toll booth is released, the end-of-service event is removed, and the simulation results are updated, indicating that this vehicle has passed through the system. Its time of exit is recorded and its delay statistics are updated.

This model is not a new research result or necessarily "best" or "optimal" in some way. Rather, it simply serves to illustrate how a traffic simulation model might be described in a flow chart context. We see that it has a master block of logic which controls the simulation, and several subordinate blocks that are used to take specific actions. Control passes from the master block to these subordinate blocks in response to the events that emerge as first-in-queue in the event list. Even if the simulation is time based rather than event based, the same logic pertains. At each time step, there is a list of actions which must be taken before time can progress to the next time step.

It should be apparent that flowcharts are a pliant paradigm for describing many types of logical processes. We have described a search process, a random assignment process, and the operation of a toll booth.

What may also be apparent is that the paradigm is fairly verbose. Considerable space and text are required to present the processing logic. It may also be apparent that graphical elements are used only to a limited degree. Shapes indicate the types of action, arcs show

how one processing step relates to all others, but text is used to describe the actions themselves.

There is also no straightforward way to tell if the logic is correct. The text and network topology must both be carefully examined. And one can imagine that the text, which is free form, could in certain instances be interpreted several different ways. A formal language structure and syntax would be needed to ensure that ambiguity could not arise.

## 2.2 Pseudo Code

Pseudo code is a descriptive paradigm far newer than flowcharts. It is in part a byproduct of the standardization in syntax that has occurred within programming languages over the past 30–50 years. Pseudo code is different from true programming code in that (1) it is language independent, (2) it describes processing “thoughts” not steps, (3) it is not intended for implementation, and (4) it does not describe the data structure(s) needed. For example, “generate a vehicle” would be an allowable pseudo code statement, but no programming language (yet) would allow such a “loose” statement in executable code.

In Figure 4 we present pseudo code for the toll booth simulation logic just described above as a flow chart.

The “main” routine controls the overall processing. First the simulation is initialized, and then the events list is sorted and processed until the end of simulation is reached ( $T = T_{Max}$ ). Time advances to the event time for each event, and the appropriate submodule is selected based on the event taking place: generate a vehicle, arrive a vehicle, start toll booth service, or end toll booth service. A review of each of the routines will show that the logic they contain is the same as that shown in the flowchart.

Clearly, this is much closer to the code that will be produced than was the flowchart. It is possible, in fact, in pseudo code, to capture the data structure as well as the processing logic, although we have not shown that here.

```

Main
Initialize simulation
Do While ( $T < T_{Max}$ )
    Sort the event list by time and queue
    position
    Move time to current event time
    Select GenVehicle, ArriveVehicle,
    StartService, or EndService as
    appropriate for the event indicated
Loop
End Main

GenVehicle
Increment NVeh
Create vehicle
Assign headway and arrival time
Add Arrival Event
End GenVehicle

ArriveVehicle (VNum)
Delete arrival event
Increment queue length
Add vehicle to back of queue
Add wait-for-service event
End ArriveVehicle

StartService (VNum, NEv)
If (TollBooth = Idle) then
    Seize toll booth
    Delete wait-for-service event
    Update queue
    Assign service time and release time
    Add end-of-service event
Else
    Increment event time by dT
End StartService

EndService (VNum)
Delete end-of-service event
Release toll booth
Update results for exiting vehicle
End EndService

```

**FIGURE 4** Pseudo code.

The main drawback to pseudo code is that it is not an intuitive way to present algorithms from a layman standpoint. Users of the HCM have become accustomed to worksheets

because they are like tax forms. (Spreadsheets are effectively computerized worksheets.) Implicitly, pseudo code assumes some degree of familiarity with common computer languages such as Pascal, Basic, or C++. Among HCM users, this is not likely to be a reasonable assumption for least another generation.

Pseudo code also has the drawback that it is not technically precise. Ambiguity can be present. The textural descriptions of actions to take can leave room for interpretation unless a formal set of language definitions is employed.

In addition, it is not easy to determine when closure has been reached. Since the pseudo code is only loosely tied to the computer code it describes, and it cannot be implemented, its veracity cannot be ascertained. Completeness and comprehensiveness cannot be ensured, nor can efficiency.

The popularity of pseudo code is not to be underestimated, however. Since the ultimate goal of such a text is to teach programming, it is not surprising that a program-like paradigm is employed.

### **2.3 Computer Code**

One presentation option is computer code itself. For instance a C++ code listing would uniquely describe the process. If other users were to code the same process they could simply use the code or do an exact translation of the code for another compiler. While this can be done, there are three major problems with using the code directly. First, it is difficult to fully interpret the detail of another person's code especially when the verbal description of steps is often missing or only briefly described. Second, code is not a condensed format, so the overview of the process is obscured. It offers no reasonable benefits to other users. Finally, the code is often copyright protected and could be difficult to get. One institution is not likely to be willing to share its code with another, without seeking some protection. In summary, actual computer code is not likely to be a satisfactory means of specifying the details of a process.

### **2.4 Spreadsheets**

Spreadsheets start with input parameters and use a step-by-step process, typically single pass, to obtain an end result. Worksheets are the antecedents to spreadsheets.

There are at least three ways to use a spreadsheet for purposes of simulation. In the first, illustrated in Figure 5, rows are used for events and the individual cells perform calculations based on prescribed formulas (embodied in the cells). We can use vehicle #5 in Figure 5 to illustrate. A first random number (0.0138) is drawn so that a headway (4.2) can be computed. This produces an arrival time (35.9). A second random (0.0781) yields a service time (3.3). Since the arrival time (35.9) is sooner than the point in time when vehicle #4 finishes service (37.3), vehicle #5 must wait, incurring a delay (1.39). At 40.6 it completes its servicing and leaves the system.

If the formulas in the cells are explicitly stated, this paradigm does provide a clear and unambiguous way to describe the simulation taking place. Any two stand-alone programs, derived from the spreadsheet will produce the same results for the same input data (as long

| Veh# | RV-Hdwy | Hdwy | Arrive | RV-Svc | SvcTim | StartSvc | Leave | Delay |
|------|---------|------|--------|--------|--------|----------|-------|-------|
| 1    | 0.3880  | 8.7  | 8.7    | 0.7475 | 6.0    | 8.7      | 14.6  | 0.00  |
| 2    | 0.6293  | 11.6 | 20.2   | 0.5347 | 5.1    | 20.2     | 25.3  | 0.00  |
| 3    | 0.1356  | 5.6  | 25.8   | 0.5360 | 5.1    | 25.8     | 31.0  | 0.00  |
| 4    | 0.1593  | 5.9  | 31.7   | 0.6396 | 5.6    | 31.7     | 37.3  | 0.00  |
| 5    | 0.0138  | 4.2  | 35.9   | 0.0781 | 3.3    | 37.3     | 40.6  | 1.39  |
| 6    | 0.0221  | 4.3  | 40.2   | 0.0965 | 3.4    | 40.6     | 44.0  | 0.44  |
| 7    | 0.3062  | 7.7  | 47.9   | 0.4172 | 4.7    | 47.9     | 52.5  | 0.00  |
| 8    | 0.2559  | 7.1  | 54.9   | 0.1014 | 3.4    | 54.9     | 58.3  | 0.00  |
| 9    | 0.5979  | 11.2 | 66.1   | 0.7773 | 6.1    | 66.1     | 72.2  | 0.00  |

**FIGURE 5 Spreadsheet simulation.**

as the same the random number generators and seeds are used). The major drawback is that only certain types of traffic operations fit this simple tabular format. The formulas in the spreadsheets constitute the code, so there is no saving in the process. This does not provide a a great deal of flexibility.

A second way to employ a spreadsheet, akin to the first, involves using macros or user-created simulation programs running in the background. The spreadsheet is mainly for data entry and results presentation. Many programs are developed in this way. But this type of implementation technique does not provide a paradigm for describing the logic employed.

A third way to use a spreadsheet is to explicitly present the logic, and use the cells to record the last numerical value obtained from each operation. An illustration of this can be found in Figure 6. Steps 0.1–0.9 are the initialization block for the simulation. We see that the ending time has been set to 3600 seconds, the time increment to 0.1 seconds, etc. Control then passes to steps 1.1–1.5. As with the flowchart, the event list is sorted into time-queue position order, and the event in lead position is executed. Control passes to steps 2.1, 3.1, 4.1, or 5.1 as appropriate. Steps 2.1–2.6 are for vehicle generation; 3.1–3.7 are for vehicle arrival; 4.1–4.10, for wait-for-servicing; and 5.1–5.4, for end-of-service. The fact that steps 1.3 and 3.4 are in bold-face type indicates that the simulation is currently at step 3.4, assigning a queue position

| Step        | Initialization Action            | Result        | Step        | Generation Action          | Result        | Step        | Wait-for-Service Action       | Result |
|-------------|----------------------------------|---------------|-------------|----------------------------|---------------|-------------|-------------------------------|--------|
| 0.1         | Vehicle number                   | 0             | 2.1         | Assign new vehicle number  | 32            | 4.1         | If toll booth is idle, then   | No     |
| 0.2         | Time                             | 0             | 2.2         | Get random number          | 0.171         | 4.2         | Toll booth = busy             | 12.2   |
| 0.3         | Ending time (sec)                | 3600          | 2.3         | Compute headway            | 2.3           | 4.3         | Remove wait-for-service event | 12.2   |
| 0.4         | Time increment (dt, sec)         | 0.1           | 2.4         | Compute arrival time       | 16.1          | 4.4         | Get random number             | 0.635  |
| 0.5         | Minimum headway (sec)            | 1.5           | 2.5         | Add arrival event          | 13.8          | 4.5         | Compute service time          | 5.0    |
| 0.6         | Average arrival headway (sec)    | 6.0           | 2.6         | Return to step 1.3         |               | 4.6         | Compute end of service time   | 17.2   |
| 0.7         | Minimum service time (sec)       | 2.0           |             |                            |               | 4.7         | Add end-of-service event      |        |
| 0.8         | Average service time (sec)       | 5.0           |             |                            |               | 4.8         | Else                          |        |
| 0.9         | Go to step 1.1                   |               | <b>Step</b> | <b>Arrival Action</b>      | <b>Result</b> | 4.9         | Increase event time by dt     | 16.1   |
|             |                                  |               | 3.1         | Remove arrive event        | 16.1          | 4.10        | Return to step 1.3            |        |
|             |                                  |               | 3.2         | Record arrival time        | 16.1          |             |                               |        |
|             |                                  |               | 3.3         | Assign to back of queue    | 16.1          | <b>Step</b> | <b>End-of-Service Action</b>  |        |
| <b>Step</b> | <b>Event Processing Action</b>   | <b>Result</b> | 3.4         | Assign queue position      | 16.1          | 5.1         | Remove end-of-service event   | 12.2   |
| 1.1         | Sort event list*                 |               | 3.5         | Add wait-for-service event | 13.8          | 5.2         | Toll booth = idle             | 12.2   |
| 1.2         | Advance time to the next event   | 16.1          | 3.6         | Add generate vehicle event | 13.8          | 5.3         | Update results                | 12.2   |
| 1.3         | Execute the event                | 16.1          | 3.7         | Return to step 1.3         |               | 5.4         | Return to step 1.3            |        |
| 1.4         | If time < ending time, go to 1.1 | 15.9          |             |                            |               |             |                               |        |
| 1.5         | Stop                             |               |             |                            |               |             |                               |        |

| Initial Event List |      |       |       |        |        |  | Current Event List |      |       |       |          |        |
|--------------------|------|-------|-------|--------|--------|--|--------------------|------|-------|-------|----------|--------|
| Event #            | Time | Q-Pos | Veh # | Status | Action |  | Event #            | Time | Q-Pos | Veh # | Status   | Action |
| 1                  | 0    | -     | -     | -      | 2.1    |  | 171                | 16.1 | -     | 32    | Arrive   | 3.1    |
|                    |      |       |       |        |        |  | 172                | 16.2 | 1     | 30    | In Queue | 4.1    |
|                    |      |       |       |        |        |  | 173                | 16.2 | 2     | 31    | In Queue | 4.1    |
|                    |      |       |       |        |        |  | 174                | 17.2 | -     | 29    | End Toll | 5.1    |

**FIGURE 6 Simulation logic in a spreadsheet.**

for the current vehicle, control having been passed from step 1.3 to step 3.1 and thence forward. This step sequence was visited last at  $t = 13.8$  when the last vehicle arrival occurred. In block 4 we see that the last time the toll booth was made busy was at  $t = 14.6$  with a computed service time of 2.6 seconds, and an end-of-service time of 17.2. This will be when the toll booth is released, as can be seen from the current event list.

This “code in a spreadsheet” is a workable paradigm. Its structure is similar to both the flowchart and the pseudo code, with one major advantage. The logic in the “result” cells makes the processing unambiguous. Two software developers, given access to the spreadsheet could create realizations of the simulation model which achieve identical results for the same inputs.

The paradigm does have its drawbacks, however. One is that the presentation format is strictly textual, and quite verbose. Second, the technical detail is not contained in the event descriptions but rather in the formulas within the result cells. Hence, to a layman, the processing logic is not truly accessible.

## 2.5 Petri Nets

A Petri Net (PN) is a graphical way to present simulation logic in a clear and unambiguous fashion. It is extremely well suited to discrete event simulations and can be adapted to use in time-based simulations. Created by Petri (1962), it is an outgrowth of state machines and their use in creating simulation models of complex systems. Wang et al. (1993), DiCesare et al. (1994), and List and DiCesare (1998) have demonstrated their applicability to traffic operations settings.

Before turning attention to the toll booth example used before, let us start with a simple PN, like the one shown in Figure 7, which represents a two-phase actuated signal. It contains all the fundamental elements: places (the large circles, A and B), transitions (the large rectangles, 1 and 2), arcs (the arrows that connect the places to the transitions), and a token (the dot within place A). Transition 1 is “enabled” since a token is present in every upstream place. It will fire, based on service time derived from the distribution shown next to it. And when that service time is complete, a (the) token in A will be removed (one from each upstream place) and a token will be deposited in B (every downstream place). This will produce the situation shown in Figure 8. Transition 2 will be enabled, it will fire, take a ser-

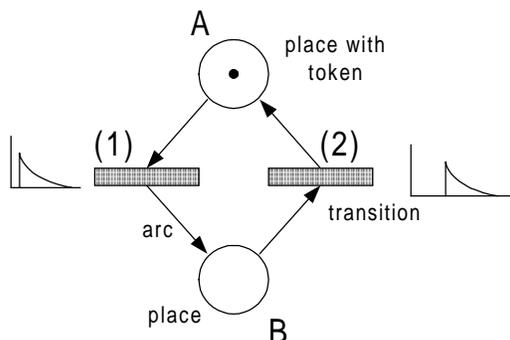


FIGURE 7 Simple PN, first state.

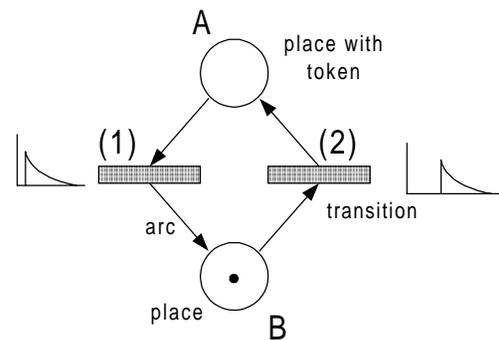


FIGURE 8 Simple PN, second state.

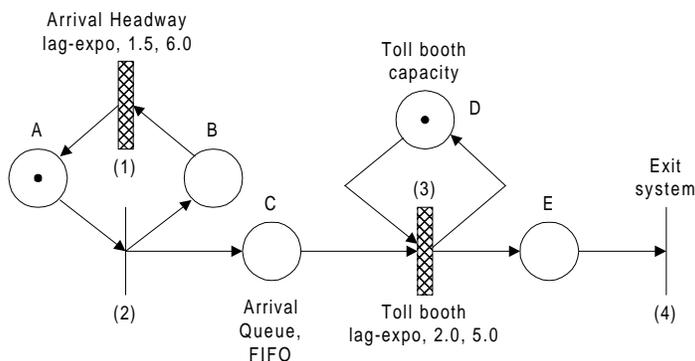
vice time based on its associated distribution, and at the end of its service time, remove a (the) token from B and deposit one in A. The process will then repeat. As the reader can see, this graphical format provides a clear and unambiguous way to describe the intended simulation logic. Any two developers, charged with creating a program that is a realization of this simulation model will be able to create code which produces the same output from the same input data. The elements not presented in Figure 7, but which must be specified, are the initial marking of the system (which in this case could be that shown in Figure 7, the random number generator and initial seed, and the rule to be imposed when breaking ties among competing simultaneous events.

A PN that portrays the toll booth example presented earlier is shown in Figure 9. The left-hand subnet generates vehicles, the right-hand one represents the toll booth.

The token in A is present initially, so that simulation can start. The token in A enables transition (2). It fires immediately (implied by the thin vertical line), and deposits tokens in B and C (i.e., *all* downstream places). The one in C represents a vehicle. The one in B enables transition (1) to time down (following a lagged exponential distribution with a minimum of 1.5 seconds and an average of 6.0) and deposit another token in A (representing the arrival of another vehicle). The token deposited in C, in conjunction with the token in D enables transition (3). It fires, removes tokens from C and D (i.e., *all* upstream places), times down based on a lagged exponential having a minimum of 2.0 seconds and an average of 5.0, and when done timing, deposits tokens in E and D (i.e., *all* downstream places). The token in D replaces the one removed when transition (3) fired. The one in E represents the vehicle exiting the toll booth. The token in E enables transition (4). It fires immediately upon being enabled and the token in E is removed, indicating that the vehicle has exited the system. Over time, vehicles are generated and pass through the toll booth, creating output results like those depicted in Figure 5 which show the delays encountered and, implicitly, the number of vehicles processed per unit time.

Once the reader is familiar with the PN paradigm, it becomes obvious that this This model description is complete and unambiguous. Although almost entirely graphical, it completely specifies the simulation model and the logic involved.

Moreover, this is an accessible paradigm. Clearly any two developers, schooled in PNs would be able to create simulation code based on the model presented and be able to obtain



**FIGURE 9** PN toll booth model.

identical results from identical input data. Moreover, with a small amount of familiarization, it is also reasonable to assert that lay individuals can learn how to read and understand such nets. They would find the graphical presentation of the logic both engaging and clairvoyant.

It is useful to observe, also, that the underlying theory of PNs is very technically precise. In fact, a net like that shown in Figure 9 has provable properties. Its freedom from deadlock can be assessed. States reachable from a given initial condition can be assessed. Moreover, there are software packages that can take the graphical model representation shown in Figure 9 and create C++ simulation model code directly with no human intervention. To stress this further, if the logic is changed, or new model features are added, a brand new simulation model can be created automatically. This makes the use of such a paradigm quite compelling.

Petri Nets are a suitable way of uniquely describing event and time based simulation processes. The authors recommend that it be used to document simulation processes that are frequently programmed by others.

Petri Nets define both the process as well as some of the assumptions used in the processes through the definition of the stochastic distributions. If further details are given about default values, many if not all assumptions will be identified and defined.

### **3. MORE CHALLENGES**

Finding a suitable paradigm to define the simulation process uniquely is only one of the challenges to be overcome in working toward greater use of simulation. Other challenges are problem definition, continuity, reliability, repeatability, and consistency with theory.

#### **3.1 Problem Definition**

For the processes to be uniquely defined, it must be straightforward to define the problem setting, especially the assumptions involved. It must be possible to assess whether they are suitable to the proposed evaluation task. As the complexity of the traffic environment increases, the processes that are used either need to become more complex or otherwise use assumptions that make the models (simulation or otherwise) more tractable. The assumptions also define the level of understanding or detail. An assumption could be that 30 percent of vehicles have poorly tuned engines. This could be programmed to give increased fuel consumption and emissions or poorer acceleration performance. This assumption defines the level of detail to be included in the model. The users of the simulation models have a responsibility to verify that the assumptions of the model match the traffic task being analyzed. The assumptions as well as the processes must be defined in the process paradigms.

#### **3.2 Continuity**

Continuity relates to smoothness among the outputs as affected by changes in input values. It also relates to outputs obtained as limits are reached (e.g., actuated signal operation trending toward pre-timed or any system reaching capacity). Analytical models

tend to be continuous and differentiable, either explicitly or implicitly. So the effects of changes in the inputs can be predicted and trends toward limits assessed, either by examining the equations employed or systematically varying the inputs. With simulation this is more difficult. No predictive equations ( $y = f(x)$ ) are involved. Trends toward limits can only be discerned through parametric analysis. Outcomes are a result of the inter-related effects of large sets of rules and small-scale cause-and-effect relationships. In addition, the rules tend to use breakpoints in deciding what to do (e.g., accept or reject a critical gap rather than a continuous function). Not the least of these is the order in which ties are broken when events have simultaneous times (e.g., permissive left turns). These can produce discontinuities in behavior. Moreover, the randomness in simulation, which is its strength, tends to obscure fundamental trends. Hence, clarity and continuity are difficult to ensure and perceive.

If a discontinuity is observed or expected then this should be clearly stated in the process. Sometimes there are reasons for discontinuities in behavior. The most common is the effect of lane discipline. Increasing the width of a road affects lane usage. At some point the road will cease to operate as a single wide lane and will start to operate as a narrow two lane road.

### **3.3 Reliability and Repeatability**

Reliability and repeatability are similar challenges. Simulation models must produce consistent (reliable) results whenever the same circumstances are examined (repeatability). But probability density functions that have long tails (e.g., exponential, normal) make these goals difficult to achieve. Long simulations and multiple replications are needed so that reasonable sample sizes (e.g., 30–1000) are obtained for the least frequent phenomena of interest (e.g., delays to right-turns-on-red). The random number generators need to be uncorrelated and ties between simultaneous events need to be broken in random order. Only then will tight confidence intervals be obtained for measures of interest. Importance sampling (see, for example, Dantzig and Infanger 1992) is also critical. Reliability and repeatability can only be ensured if these actions are taken.

Any tests on reliability or repeatability must involve sufficient examples or cases so that statistical tests can be applied with confidence. This might involve many runs or runs with sufficiently long periods.

### **3.4 Consistency with Theory**

Consistency with theory is probably the toughest challenge. In the case of analytical models, consistency is assured. Or the absence thereof is obvious. Either the assumptions are stated explicitly or they can be deduced implicitly from the equations employed. Their veracity can be assessed and challenged. With simulation models, such questions can only be addressed by considering the totality of rules employed, or by conducting parametric analyses and fitting functions to the results. Even so, those functions that result have no basis in theory, implicit or explicit. Consistency can only be deduced, like a hypothesis not rejected, by studying the outputs generated by the model. Not unlike science experiments in which the theory predicts outcomes which are then proved or disproved by

experimental work, the difference here is that the experimental world is synthetic, not real. The logic can become circular, and potentially detached from the real world, which is nominally being modeled. Ensuring consistency with theory is an area that requires considerable investigation.

#### **4. TECHNICAL DETAILS**

Three additional technical details are critical if consistency is to be assured among implementations of a simulation model. The first concerns random number sequences; the second relates to the way ties are broken among simultaneous events; and the third deals with computational procedures. Each of these can be the focus of a stand-alone paper, but we only intend to present a brief discussion.

##### **4.1 Random Number Sequences**

Clearly, one of the main reasons to use simulation is the randomness it introduces. In a stochastic simulation model, vehicles can arrive randomly, they can have varying characteristics (e.g., vehicle type), they can change lanes randomly, and so forth.

But randomness also means different outcomes. The sequence of random numbers employed directly affects what happens. One run can produce high delays while another produces lower values. One run may have higher than desired volumes while another may fail to be high enough. One run is typically not sufficient to get definitive performance results.

Multiple runs are required, with different seeds and number sequences, to ensure that a true assessment of performance has been obtained and that correlation, serial and otherwise, is avoided.

Picking the random numbers and/or specifying the process by which they are generated is very important. Hellekalek (1998) provides an excellent discussion of this issue. On the one hand, to verify or validate one simulation model against another, the exact same (set of) sequence(s) of random numbers must be used, in exactly the same way. Otherwise the results may be different. On the other hand, to produce quickly converging results, and tight confidence bounds, e.g., for the means of distributions (e.g., average delay) etc., one must be certain that the best possible uncorrelated (multiple) sequences of random numbers are employed. Infanger (1998) provides discussion on this point.

##### **4.2 Breaking Ties**

At first, it may not seem important, but the way in which ties are broken among simultaneous events is critically important to a simulation model's predictions. If one considers an unsignalized intersection, or a two-lane rural highway, the importance of this fact becomes clear. If a model of the former is coded so that the northbound approach is always considered first, or for the latter, that the eastbound direction on the two-lane highway always gets to consider passing opportunities first, a bias will be introduced. This will always be true as long as the same decision sequence is followed.

So the order in which ties are broken must be well defined. On the one hand, if clear sequences exist in real life, they must be followed. On the other, if no sequence exists, randomness must prevail. Moreover, multiple runs are needed, with the results being averaged, to cover all possible sequence options and ensure that defensible results are obtained. Weiland (1998) provides an excellent discussion of this point.

In sum, given the randomness in traffic phenomena, where randomness in decision making sequences is a central concern, this aspect of the simulation model must be carefully addressed.

### **4.3 Computational Procedures**

While it may seem relatively unimportant, being careful about computational procedures is important; different computers and compilers tend to use different standards for floating point arithmetic.

Since increments of time with randomly varying values are so critical to traffic operations (e.g., gap times, actuated signal timing, vehicle headways, etc.), the floating point arithmetic must be carefully specified.

Many simulation models actually use integer arithmetic for the simulation since it is faster. Time is defined based on a minimum time step (e.g., 0.1 second). All time increments become integer multiples of this value. For example, 10.3 seconds becomes 103 time units, 1.35 seconds becomes 14 time units. Rounding and truncation must be carefully considered.

Simulation models that use floating point arithmetic have other issues. One is the error tolerance on “if” statement evaluations. Another is the number of decimal places retained, and/or the number of significant figures kept. If ANSI standards are followed, some, but not all of these issues are resolved. But tolerances on “if” statements and similar issues must be addressed explicitly in a clear and precise manner.

For the simulation models created by two different developers to achieve the same results, these floating point arithmetic issues must be resolved in exactly the same way. Or put a different way, standards must be specified by which floating point arithmetic operations will be done. This is a “little” detail that can have major importance.

## **5. PROGRESS TOWARD THE FUTURE**

Much to do remains ahead. This paper has simply indicated that there are significant challenges to be met before simulation models can become the mainstay of capacity and operational analysis. The question is not whether, but how and when.

Among the challenges we have highlighted are: adoption of an acceptable paradigm for describing such models and finding ways to address issues of continuity, reliability, repeatability, and consistency. A simple statement like ISO-9000 compliance is insufficient. Data definitions, objects, and structures are also important.

Finding a suitable paradigm is probably the most significant issue. Without a “language” or focused verbiage by which models can be stated and presented, the other discussions are moot. Ambiguity must be avoided, and the model logic must be clear to practitioners.

Even once a paradigm is found, the issues of problem definition, continuity, reliability, repeatability, and consistency must be successfully addressed. A QA/QC (quality assurance, quality control) framework is needed along with a process for verification and validation so we can be sure that every model meets acceptable performance standards.

These authors would recommend starting from a basis like the PN models, because of their clarity and preciseness, and work toward creation of a set of operands (graphical symbols) whose simulation interpretation is precise and exact, and which can be combined in a hierarchical fashion (DiCesare et al. 1994; List and DiCesare 1998) to build higher-level models.

In closing, these are exciting times. Computer technology has reached performance levels where the use of simulation is reasonable. The formalism of simulation has reached a juncture where creating an unambiguous syntax for describing traffic operations models should be possible. Our challenge is to create that syntax and define with precision the attributes of the building blocks upon which traffic simulations can be based.

## 6. ACKNOWLEDGMENTS

The authors are deeply indebted to their respective institutions for the opportunity to carry out the research leading to this paper. They are also thankful for the assistance of Brian Menyuk, Frank DiCesare, and Christopher Carothers, who provided important and useful counsel in the modeling paradigms which might be employed.

## REFERENCES

- Athanailos, E.G. (1994). Integration of the highway capacity manual and the TRAF-NETSIM simulation model. *ITE Journal*, 64(5), pp. 33–38.
- Athanailos, E., and Berman, L. (1993). The integration of the highway capacity manual and the TRAF-NETSIM simulation model, *Proc., 4th International Conference on Micro-computers in Transportation*, pp. 403–414.
- Banks, J., Carson, J.S., and Sy, J.N. (1989). *Getting started with GPSS/H*. Wolverine Software Corporation, Annandale, Va.
- Bonneson, J.A., and Fitts, J.W. (1996). Effect of upstream signal on non-priority movement capacity and delay. *Transportation Research Record 1572*, pp. 174–182.
- Bossel, H. (1994). *Modeling and Simulation*. A.K. Peters, Wellesley, Ma.
- Botha, J.L., Zeng, X., and Sullivan, E.C. (1993). Comparison of performance of TWOPAS and TRARR models when simulating traffic on two-lane highways with low design speeds. *Transportation Research Record 1398*, pp. 7–16.

Bullen, A.G.R., Hummon, N., Bryer, T., and Nekmat, R. (1987). EVIPAS: A Computer Model for Optimal Design of a Vehicle-Actuated Traffic Signal. *Transportation Research Record 1114*, pp. 103–110.

CACI Products Company. (1994). *SIMSCRIPT II.5 Programming Language*. Los Jolla, Ca.

Courage, K.G., Wallace, C.E., and Alqasem, R. (1988). Modeling the effect of traffic signal progression on delay. *Transportation Research Record 1194*, pp. 139–146.

Dantzig, G.B., and Infanger, G. (1992). Large-scale stochastic linear programs — importance sampling and Benders decomposition. *Computational and Applied Mathematics I*, pp. 111–119.

Dicesare, F., Kulp, P., Gile, M., and List, G.F. (1994). The application of Petri nets to the modeling, analysis and control of intelligent urban traffic networks. In: Valette (ed.), *Application and Theory of Petri Nets 1994, Proc., 15th International Conference*, pp. 2–15.

Gerlough, D.L. (1964). Simulation of Traffic Flow. *Highway Research Board Special Report 79*.

Halati, A., and Torres, J.F. (1990). *FRESIM Calibration/Validation*. Federal Highway Administration, Washington, D.C.

Hellekalek, P. (1998). Don't trust parallel Monte Carlo!. *Proc., 12th Workshop on Parallel and Distributed Simulation*, pp. 82–89.

Kaman Sciences Corporation. (1996). *TSIS User's Guide, Version 4.0, CORSIM User's Guide Version 1.0*, Federal Highway Administration, Washington, D.C.

Kosonen, I. (1990). Simulation tool for traffic signal control planning. *IEE Conference Publication 320*, pp. 72–76.

Kuhne, R.D., and Rodiger, M.B. (1991). Macroscopic simulation model for freeway traffic with jams and stop-start waves. *Winter Simulation Conference Proceedings*, pp. 762–770.

Kyte, M., Kittelson, W., Zhong, T.Z., Robinson, B., and Vandehey, M. (1996). Analysis of traffic operations at all-way stop-controlled intersections by simulation. *Transportation Research Record 1555*, pp. 65–73.

Lee, C.E., and Fong-Ping, L. (1981). Simulation of Traffic Performance, Vehicle Emissions, and Fuel Consumption at the Intersections: The TEXAS-II Model. *Transportation Research Record 1984*, pp. 133–140.

Lieberman, E. (1991). Integrating GIS, simulation and animation, *Winter Simulation Conference Proceedings*, pp. 771–775.

List, G.F., and Dicesare, F. (1998). *A Prototype Real-Time Control System for Signalized Networks*. New York State Energy Research and Development Authority, Albany, N.Y.

Massoumi, R., Wyznyckyj, L.C., and Menaker, P. (1997), WATSIM micro-simulation: I-780/I-680/I-80 corridor, *Proc., Conference on Traffic Congestion and Traffic Safety in the 21st Century*, pp. 90–96.

Messer, C.J., and Bonneson, J.A. (1997). *Capacity Analysis of Interchange Ramp Terminals*, NCHRP Report 3–47.

Ostrom, B., Leiman, L., and May, A.D. (1993). Suggested procedures for analyzing freeway weaving sections. *Transportation Research Record 1398*, pp. 42–48.

Papacostas, C.S., and Willey, M. (1993). Use of TRAF-NETSIM to estimate the traffic impacts of an urban-resort area development, *Proc., 4th International Conference on Microcomputers in Transportation*, pp. 368–379.

Pegden, C.D., Shannon, R.E., and Sadowski, R.P. (1990). *Introduction to simulation using SIMAN*. McGraw-Hill, New York, N.Y.

Petri, C. (1962). *Kommunikation mit Automaten*. Ph.D. dissertation, University of Bonn, Bonn, Germany.

Rakha, Hesham A., and Van Aerde, M.W. (1996). Comparison of simulation modules of TRANSYT and integration models. *Transportation Research Record 1566*, pp. 1–7.

Ran, B., Leight, S., and Chang, B. (1998). Microscopic simulation for AHS merging analysis. *Proc., International Conference on Applications of Advanced Technologies in Transportation Engineering*, pp. 77–82.

Roess, R.P., McShane, W.R., and Prassas, E.S. (1998). *Traffic Engineering*, 2nd Ed., Prentice-Hall, Upper Saddle River, N.J.

Rouphil, N.M., and Akcelik, R. (1992). Preliminary Model of Queue Interaction at Signalized Paired Intersections. *Proceedings of Australian Road Research Board*, 16(5), pp. 325–345.

Sabnayagam, S., Schuster, J.J., and Konyk, S. (1990). Traffic simulation at unsignalized intersections using GPSS. *Microcomputer Applications in Transportation III*, pp. 425–433.

Schaffer, S.J., and LaRue, W.W. (1994). BONeS Designer: A graphical environment for discrete-event modeling and simulation. *Proc., IEEE International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, IEEE, pp. 371–374.

Sedgewick (1988). *Algorithms*. Addison-Wesley Publishing, New York, N.Y.

Stewart, J., Baker, M., and Van Aerde, M. (1996). Evaluating weaving section designs using INTEGRATION. *Transportation Research Record 1555*, pp. 33–41.

Systems Modeling Corporation. (1994). *ARENA getting started guide*, Pittsburgh, Pa.

Taori, S., and Rathi, A.K. (1996). Comparison of NETSIM, NETFLO I, and NETFLO II traffic simulation models for fixed-time signal control. *Transportation Research Record 1566*, pp. 20–30.

Torres, J.F., Halati, A., Gafarian, A., and Guevrekian, S. (1983). *Statistical Guidelines for Simulation Experiments*, Volumes 1, 2, and 3. Federal Highway Administration, Washington, D.C.

Transportation Research Board. (1997). *Special Report 207: Highway Capacity Manual*. Washington, D.C.

Troutbeck, R.J. (1993). Effect of heavy vehicles at Australian traffic circles and unsignalized intersections. *Transportation Research Record 1398*, pp. 54–60.

U.S. Department of Transportation. (1994). *TRAF User Reference Guide, Version 4.2*, Federal Highway Administration, Washington, D.C.

Van Aerde, M. (1993). *INTEGRATION: A Model for Simulating Integrated Traffic Networks*. M. Van Aerde & Associates, Toronto, ON.

Wang, Y., and Prevedouros, P.D. (1998). Comparison of INTEGRATION, TSIS/CORSIM, and WATSim in replicating volumes and speeds on three small networks. *Transportation Research Record 1644*, pp. 80–92.

Wang, H., List, G.F., and DiCesare, F. (1993). Modeling and evaluation of traffic signal control using timed Petri nets. *Proc., 1993 IEEE International Conference on Systems, Man and Cybernetics 2*, pp. 180–185.

Wieland, F. (1999). The threshold of event simultaneity. *Transactions of the Society for Computer Simulation International*, 16(1), pp. 23–31.

Wong, S.-Y. (199x). Comparing capacities and delays estimated by highway capacity software and TRAF-NETSIM to field results. *Compendium of Technical Papers, Annual Meeting—Institute of Transportation Engineers*, pp. 224–227.

Wong, S.-Y. (1991). Capacity and level of service by simulation. A case study of TRAF-NETSIM. *Proc., International Symposium on Highway Capacity and Level of Service*, pp. 467–483.

Zarean, M., and Nemeth, Z.A. (1988). WEAVSIM. A microscopic simulation model of freeway weaving sections. *Transportation Research Record 1194*, pp. 48–54.