An $M/G/C/C$ State Dependent Network Simulation Model

F. R. B. Cruz[*], J. MacGregor Smith
Department of Mechanical & Industrial Engineering,
University of Massachusetts, Amherst, MA 01003, USA.

R. O. Medeiros
Departamento de Estatística,
Universidade Federal de Minas Gerais,
31270-901 - Belo Horizonte - MG, Brazil.
{fcruz,jmsmith}@ecs.umass.edu, rey@est.ufmg.br

August 15, 2003

Abstract — **A discrete-event digital simulation model is developed to study traffic flows in $M/G/C/C$ state-dependent queueing networks. Several performance measures are evaluated, namely (i) the blocking probability, (ii) throughput, (iii) the expected number of the customers in the system, and (iv) expected travel (service) time. Series, merge, and split topologies are examined with special application to pedestrian planning evacuation problems in buildings. Extensive computational experiments are presented showing that the simulation model is an effective and insightful tool to validate analytical expressions and also to analyze general accessibility in network evacuation problems especially in high-rise buildings.**

Keywords — discrete-event simulation, queueing systems, finite capacity, state dependent.

[*]On sabbatical leave from the *Departamento de Estatística, Universidade Federal de Minas Gerais*, 31270-901–Belo Horizonte–MG, Brazil.

1 Introduction

Congestion and finite capacity queueing systems are probably one of the most prevalent facts of modern life. Congestion usually leads to a decrease in the systems service rates and finite capacity impedes overall system throughput. The main focus of this paper are finite capacity and state dependent service rate networks, which are appropriate tools for modeling congestion in pedestrian [1, 2] and vehicular [3] traffic networks, accumulation conveyor systems [4], and many other systems with finite capacity and decaying service rates as a function of increased density of customers.

In particular, this paper is concerned with $M/G/C/C$ state dependent networks, which following Kendall's notation stand for Markovian arrival processes, general state dependent service rates, c parallel servers, and a total capacity c (including the servers). Additionally, the service rate is a decreasing function of the number of customers in the system.

1.1 Motivation

State dependent queueing network models occur throughout many parts of the world. These networks must be designed and controlled so that the flow of customers through them achieve the best possible performance. The dynamic nature of the $M/G/C/C$ state dependent models requires powerful tools for analysis because of the complex blocking that occurs and the simultaneous dynamic updating required to adjust the services rates of the customers.

We have developed many stationary approaches for the long-run equilibrium (steady-state) design and analysis of $M/G/C/C$ state dependent models, yet the non-stationary or transient nature of these models has remained beyond our capabilities. Thus, we need tools and a corresponding methodology to model state-dependent $M/G/C/C$ models for non-stationary environments. Examples include traffic control for signalized intersections, vehicular and pedestrian network evacuation, and many other transient situations in which networks do not reach steady-state. In fact, the non-stationary situation is more prevalent than the stationary one. Certainly it would be most useful if we could develop an analytical non-stationary model for $M/G/C/C$ networks, yet having a discrete-event simulation model is a practical first-step.

Not only it will be argued that non-stationary models are important for themselves, but it will be shown in the paper that non-stationary models complement, enhance, and help us evaluate the stationary models already developed.

1.2 Outline

Section 2 details the analytical queueing and congestion models that form the foundation for the simulation models used in this paper while Section 3 presents the details of the discrete event digital simulation model. Section 4 demonstrates validating results for the simulation system along associated applications. In Section 5, a network evacuation model of a ten-story building is demonstrated in order to illustrate the efficacy of the simulation model in this important application area. Finally, Section 6 summarizes and concludes the paper.

2 Mathematical Models

2.1 Introduction

There are many real traffic systems in which customers flowing through the system adjust their speed because of the density of customers in the system. In this section of the paper, empirical model results are described along with consequent analytical representations of the empirical models of this service system decay rate. As will be described, linear and exponential decay functions are quite appropriate in characterizing the service rate decay function. While the focus will be on the use of $M/G/C/C$ models for pedestrian traffic, it should be remarked that these models are interchangeable for vehicular traffic and whenever state dependent service rates are appropriate models for the application.

2.2 Notation

As an aid to the reader, a brief section of notation which is used throughout the paper is provided:

c	:= corridor capacity in number of occupants;
l	:= corridor length in meters;
w	:= corridor width in meters;
V_n	:= average walking speed for n occupants in a corridor in m/s;
V_1	:= average lone occupant walking speed in m/s;
V_a	:= average walking speed when pedestrian density is 2 ped/m^2 in m/s;
V_b	:= average walking speed when pedestrian density is 4 ped/m^2 in m/s;
γ, β	:= shape and scale parameters for the exponential model;
λ	:= pedestrian arrival rate in ped/s;
N	:= random variable that denotes the number of occupants in a corridor;
n	:= actual number of pedestrians in a corridor;
$p(n)$	:= Pr{N = n}, probability of n occupants in the system, for $n = 1, 2, \ldots, c$;
$p(0)$	:= Pr{N = 0}, empty system probability;
$p(c)$	:= Pr$\{N = c\}$, blocking probability;
θ	:= throughput in ped/s;
L	:= $E(N)$, mean (expected) number of occupants in the system (also known as work-in-process);
W	:= $E[T]$, mean waiting time (expected service time) in seconds;
$E[T_1]$	:= mean waiting time for lone occupant in seconds.

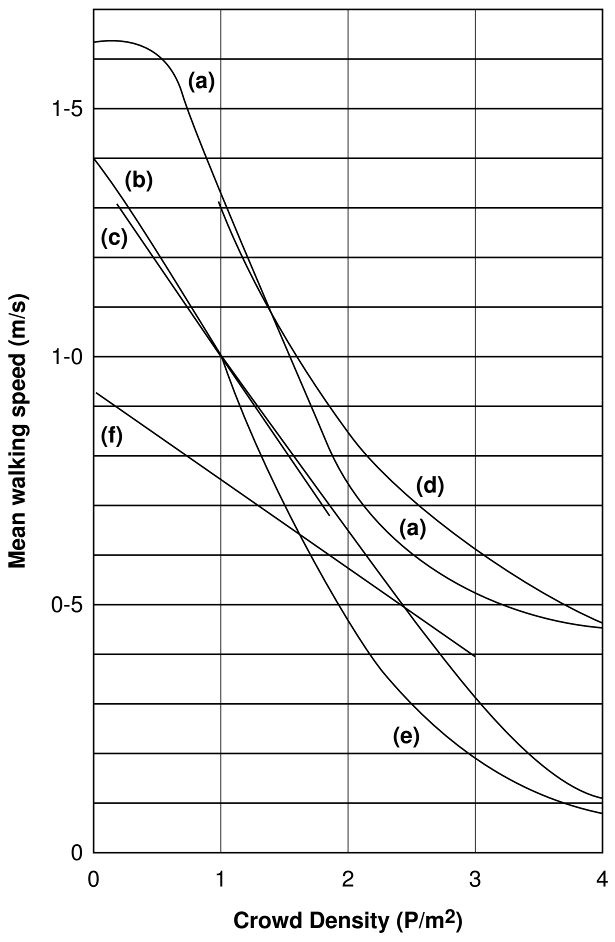


Figure 1: Average walking speed.

2.3 Pedestrian Congestion Modeling

Congestion is easily observed in pedestrian circulation areas, such as corridors, stairwells, shopping malls, ramps, elevators *etc.*, and it essentially occurs when the number of people within the circulation system increases. The increase in traffic density tends to reduce the average speed of all users. Of course, several factors can affect the average walking speed, for instance the shape of the surrounding walls, colors, lighting level, and floor materials in the pedestrian walkway system. While it is known that males tend to walk faster than females and teenagers, faster than adults, however, several experimental studies have shown that all these factors are minor compared to the total number of pedestrians within the corridor. And, in fact, this total number changes dynamically over time. As the number of people increases, it will be more likely that slower pedestrians block faster pedestrians. Thus, higher pedestrian densities, defined as the number of people per unit area, reduce the individual pedestrian traffic velocity. Thus, pedestrian density can be viewed as the main factor in determining mean walking speed.

One of the most interesting problems that could be treated by state dependent queueing systems is related to pedestrian traffic flows in the circulation systems of buildings [1, 5, 6, 7]. Figure 1 presents experimental curves (a through f) that relate the walking speed of pedestrians to crowd density, based on various empirical studies that illustrate that at a mean density of three pedestrians per square meter (3 ped/m^2) walking is reduced to a shuffle, and at 5 ped/m^2, forward movement essentially comes to a halt (see Tregenza [5]). Thus, the corridor capacity equals the largest integer not superior to five times its area in m^2, that is:

$$c = \lfloor 5 \times l \times w \rfloor, \tag{1}$$

in which c is the corridor capacity, l is its length in meters, w, its width in meters, and $\lfloor x \rfloor$ is the largest integer not superior to x.

Tregenza [5] also affirms that the lone occupant walking speed is around 1.5 m/s. Note that the flow tends to stop as $n \to c$ but some movement may exist when $n = c$. On the other hand, all movement should essentially come to a halt beyond this point, that is, $V_n = 0$, for all $n \geq c + 1$. Therefore, a straightforward linear model that complies with these requirements would be:

$$V_n = V_1 \frac{c + 1 - n}{c}, \tag{2}$$

in which $V_1 = 1.5$ m/s is the lone occupant walking speed.

However, in light of Figure 1, it seems that an exponential model might be another reasonable approximation. Thus, Yuhaski and MacGregor Smith [1] proposed the following model for uni-directional pedestrian flows:

$$V_n = V_1 \exp\left[-\left(\frac{n-1}{\beta} \right)^{\gamma} \right], \tag{3}$$

in which

$$\gamma = \frac{\ln\left[\frac{\ln(V_a/V_1)}{\ln(V_b/V_1)}\right]}{\ln\left(\frac{a-1}{b-1}\right)}, \ \beta = \frac{a-1}{\left[\ln\left(V_1/V_a\right)\right]^{1/\gamma}} = \frac{b-1}{\left[\ln\left(V_1/V_b\right)\right]^{1/\gamma}},$$

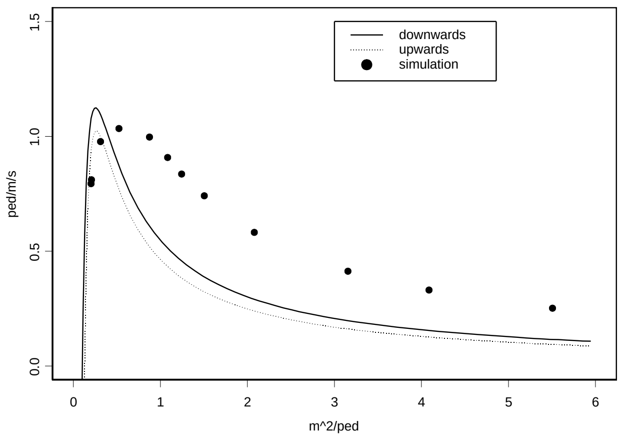


Figure 2: Fruin's curves [8] for stairwells.

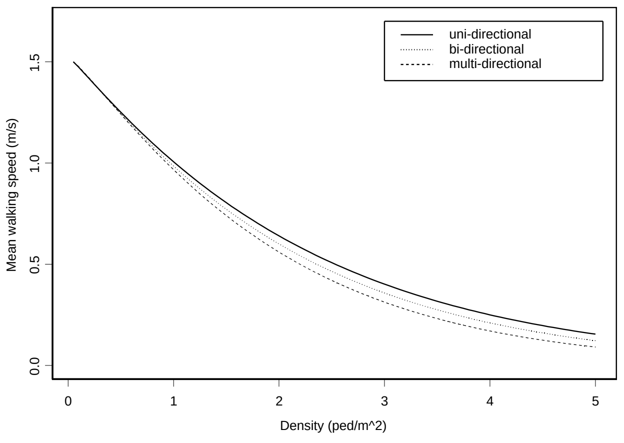


Figure 3: Exponential models.

$V_1 = 1.5$ m/s, $V_a = 0.64$ m/s, $a = 2lw$, $V_b = 0.25$, and $b = 4lw$.

Note the reader that the model was adjusted based on three representative points of the six curves in Figure 1, that is, based on the coordinates $V_1 = 1.5$ m/s, for density 1 ped/m^2, $V_a = 0.64$ m/s, for density 2 ped/m^2, and $V_b = 0.25$ m/s, for density 4 ped/m^2. Yuhaski and MacGregor Smith [1] pointed out that there are other possibilities, *e.g.* non-linear regression or piece-wise linear approximations, but the results would not differ significantly. The model seems to be well calibrated since it reproduces with some accuracy the pattern of Fruin's curves [8] for stairwells, as it is seen in Figure 2.

Cheah and MacGregor Smith [9] extended the exponential model to represent bi-directional and multi-directional pedestrian flows by using $V_a = 0.60$ m/s and $V_b = 0.21$ m/s in Eq. (3), for bi-directional flows, and $V_a = 0.56$ m/s and $V_b = 0.17$ m/s, for multi-directional flows. As it is seen in Figure 3, the results do not differ significantly.

While the previous stationary models are effective tools, we have been unable to compare these stationary models with a simulation model or other non-stationary analytical models. As argued earlier, non-stationary environments are more typical than stationary environments. Current discrete-event simulation languages are not effectively capable of dynamically updating the service rates of the customers flowing through the system without significant computational overhead. Thus, the discrete event simulation model described in the next section of the paper begins to fulfill this need.

3 Discrete Event Digital Simulation Model

For convenience, the discrete-event digital simulation model proposed here (more on discrete-event simulation models, in the book of Fishman [10]) is implemented in C++, a flexible, powerful, and fast programming environment. However, there exist nowadays many general purpose simulation languages that could be used as well, including GPSS, SIMSCRIPT, SLAM, and SIMAN, among others (more details can be found in the paper of Nance [11]). Unfortunately, these packaged simulation models do not have sophisticated data structures to deal with the dynamic updating of the customer or vehicular service rates as a function of the density of customers in the system, so they tend to run extremely slow in comparison with a dedicated simulation model as developed in this paper.

3.1 Data Structures

The simulation model is within a broader package that follows the structure presented in Figure 4. The module `congestion models` is composed by the virtual class `GenCM`, that provides general purpose methods such as the service rate V_n, the capacity c, the expected service time for lone occupant $E[T_1]$, and lone occupant speed V_1. The derivative classes `LinCM` and `ExpCM` implement the linear and exponential services as described earlier and the module `user congestion models` adjust linear and exponential models particularly for the application. The module `analytical model`, which implements the Generalized Expansion Method (GEM) [12, 13], and the module `optimization system` will not be discussed in this paper since the focus here is on the performance evaluation rather than optimization. However, additional details on the GEM will be given in Section 4.

Of main interest here is the module `simulation model`, which essentially implements the object `MgccSim`, seen in Figure 5, which depicts all data structures within the object and how they interrelate. Basically, this object has the number of nodes in the $M/G/C/C$ state dependent network, `nOfNodes`, the total simulation time, `totalTime`, the origin-destination matrix, `arcs`, a vector of objects, `MgccResource`, and an event queue, `MgccEventQueue`. The objects `MgccResource` keep track of all statistics that might be of interest for each $M/G/C/C$ queue, which are the sum of blocking, arrivals, and departures, the sum of overall time within the system, and the current number of users. Also part of each `MgccResource` is the congestion model `GenCM`, with methods to access its mean speed (service rate) V_n, capacity c, and expected service time for lone occupant $E[T_1]$.

The critical part of the object `MgccSimul` is the object `MgccEventQueue` which implements the event queue. The event queue is implemented as a linked list dynamically created at running time that keeps all discrete events. Unexpectedly, after much experimentation, it was seen that the best way to ensure low CPU times was to keep the events in the event queue unsorted, with the burden of inspecting the entire queue to recover the earliest event (more details on this matter will be given shortly).

Each object `MgccEvent` of the event queue contains the indication of which $M/G/C/C$ queue it relates, `whichQueue`, the expected occurrence time, `occurTime`, the event `type` (`arrival`, `departure`, or `end_simulation` events), and the entity to which it is related, `MgccEntity`. The object `MgccEntity` represents each user (pedestrian) of the $M/G/C/C$ state dependent network and has a unique user identification number, `id`, the time it has arrived at the system, `sisArrival`, the time it has arrived at the current queue, `queueArrival`, the time there was the last state change (an user enters of leaves the queue), `lastChange`, and the user's position at the last state change, `lastPosition`. As one may has already noticed, the architecture used is freely inspired by the simulation system Arena [14].

3.2 Algorithm

The simulation algorithm in pseudo-code is illustrated in Figure 6. All this algorithm does is to initialize the event queue `MgccEventQueue` by programming the last event (`end_simulation`) and the first, time-zero, events (`arrival`). Then, the algorithm iteratively searches the earliest event (in the order of occurrence) and processes it, until the final event (`end_simulation`) is reached.

The arrivals must be treated by the algorithm shown in Figure 7. At this point, it is possible to schedule the next arrival at that queue and to include this newly created event in the event queue. Since the arrivals are assumed to be coming from a Markovian process, the inter-arrival times follow the exponential distribution $\text{Exp}(\lambda_i)$. The current arrival generates a `departure` event, that must be also inserted into the event queue. The occurrence of that departure is after a delay of l/V_n, in which l is the length of the respective corridor and V_n, the average speed for n occupants. The most complex operation in the algorithm `ProcessArrival` is to delay the departure of every user in the queue, that is, to take into consideration the state dependency of the system.

For all entities at that queue, the algorithm `DelayDeparture` (not shown) must update the position since the last change, `lastPosition`, the time since the

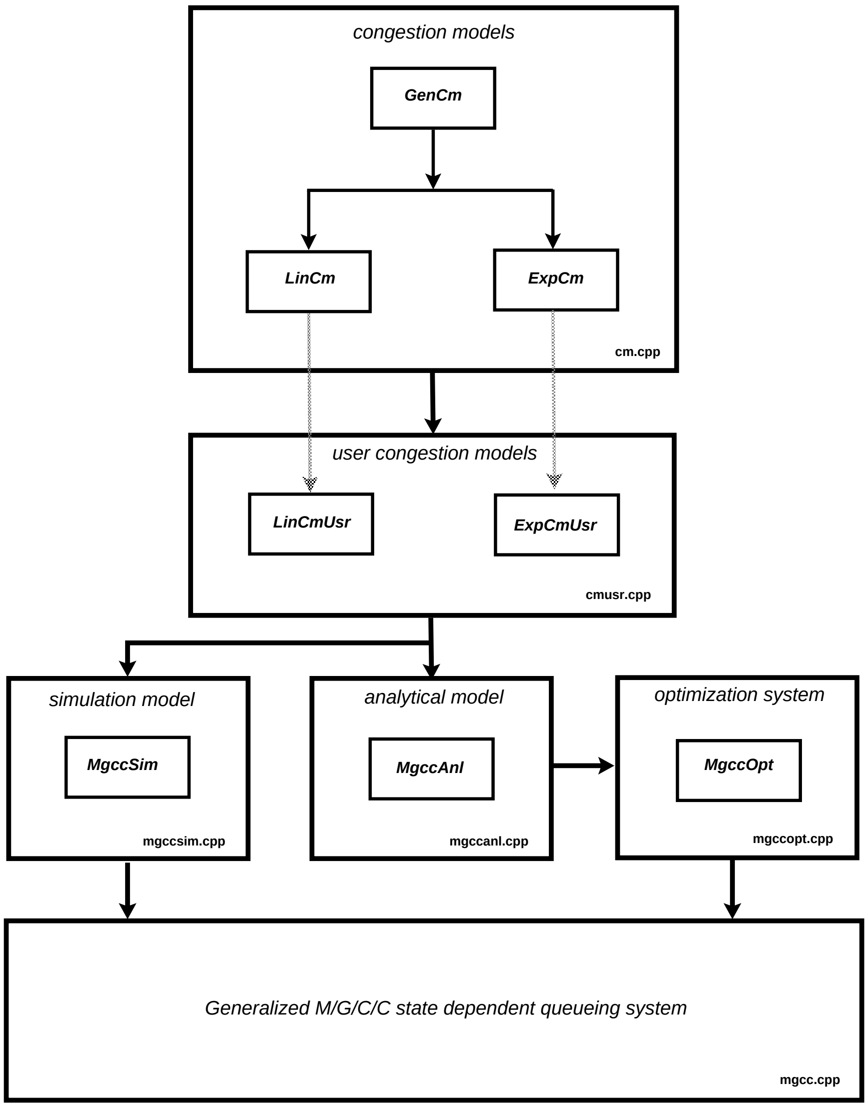


Figure 4: Generalized $M/G/C/C$ state dependent queueing system.

last change, `lastChange`, and the event's new occurrence time, `occurTime` (Figure 5), as follows:

$$\begin{aligned}\texttt{lastPosition} &\leftarrow \texttt{lastPosition} + V_{n-1} \times (t - \texttt{lastChange}),\\ \texttt{lastChange} &\leftarrow t,\\ \texttt{occurTime} &\leftarrow t + (l - \texttt{lastPosition})/V_n,\end{aligned}$$

in which t is the current time. Note that since it is possible that the user is blocked after service, it is necessary to make sure that the `lastPosition` does not go beyond the corridor's length l after updating.

The departure event treatment is a bit trickier. It is shown in Figure 8. The departure event involves carefully determining the new destination of the customer. For example, if the corresponding queue is a split node, then the blocking across the split junction must be carefully analyzed. If the destination queue is at capacity, then the departure from the origin queue must be delayed (blocking after service) until there is room in the destination queue. Otherwise, the new departure time of this entity is reprogrammed by adding a delay of l/V_n, advancing the departures of all entities at the origin queue, and delaying all departures at the destination queue.

The algorithm `AdvanceDeparture` (not shown) follows the same philosophy as the algorithm `DelayDeparture`. The position since the last change, the time since the last change, and the event's new occurrence time are updated as follows:

$$\begin{aligned}\texttt{lastPosition} &\leftarrow \texttt{lastPosition} + V_{n+1} \times (t - \texttt{lastChange}),\\ \texttt{lastChange} &\leftarrow t,\\ \texttt{occurTime} &\leftarrow t + (l - \texttt{lastPosition})/V_n.\end{aligned}$$

Note that, by updating (advancing or delaying) the occurrence times of all entities at a particular queue, a previously sorted (by time of occurrence) event queue would end up unsorted. This is the explanation for a better overall performance of a simulation model based on unsorted event queues. It is a bit costly to recover the earliest event in an unsorted queue, since all the elements of the queue must be inspected, but it would be even more costly to keep that queue sorted.

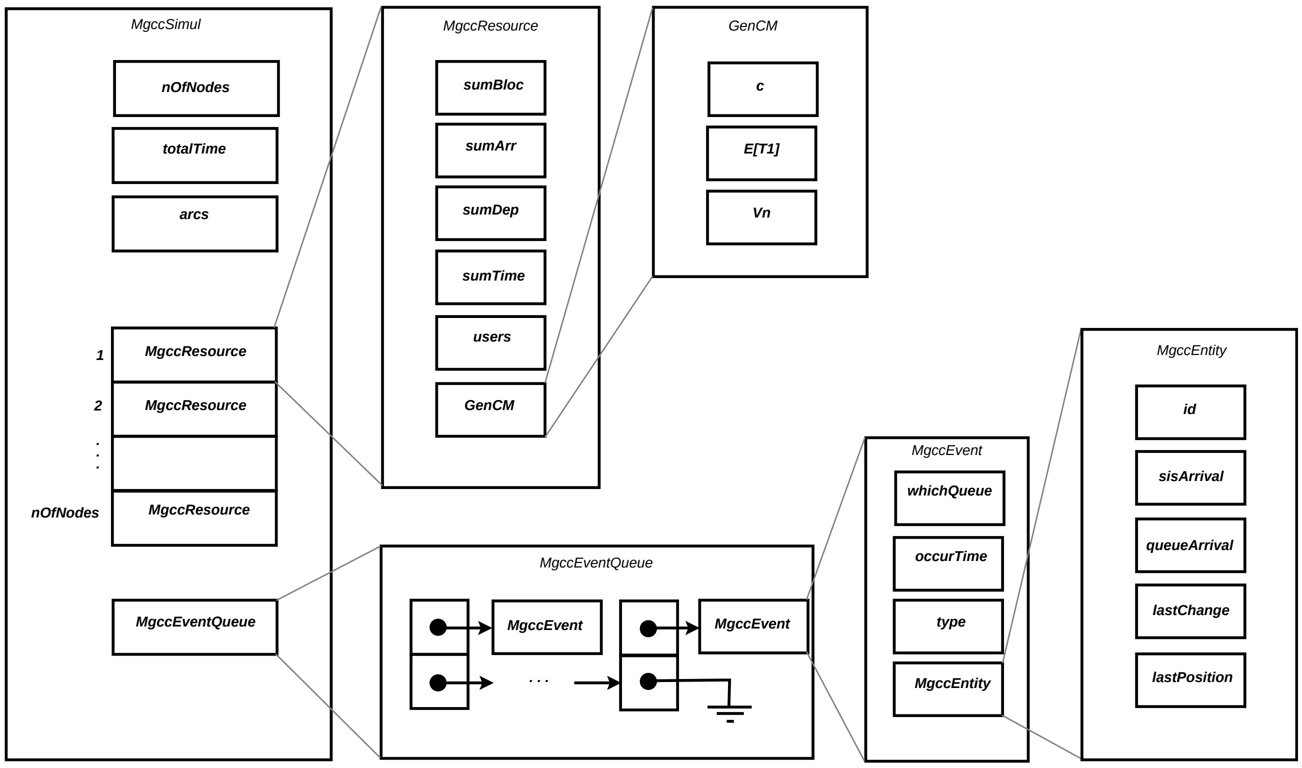


Figure 5: Object `MgccSim`.

4 Computational Experiments

All computational experiments were conducted in a PC, CPU Pentium II 400 MHz, 64 MB RAM, under Windows NT 4.00 operating system, and were divided into two parts. The first experiments validate the simulation model for a single $M/G/C/C$ queue, since it is well known that $M/G/C/C$ systems are stochastically equivalent to $M/M/C/C$ systems [9], since that the stationary probabilities are exact. The second part validates the approximate expressions of the GEM [12, 13].

4.1 Simulator Validation

The limiting probabilities for the number of pedestrians in a $M/M/C/C$ queueing model have been developed before. It has been shown that $M/M/C/C$ and $M/G/C/C$ state dependent queues are stochastically equivalent [9], as well as that the limiting probabilities for the number of pedestrians in an $M/G/C/C$ state dependent queueing model are as follows [1]:

$$p(n) = \left(\frac{\Big[\lambda E[T_1]\Big]^n}{n! f(n) \cdots f(2) f(1)} \right) p(0),\ \forall n = 1, 2, \ldots, c, \tag{4}$$

in which

$$p(0)^{-1} = 1 + \sum_{i=1}^{c} \left(\frac{\Big[\lambda E[T_1]\Big]^i}{i! f(i) \cdots f(2) f(1)} \right)$$

is the empty system probability, λ is the arrival rate, $E[T_1] = l/V_1$ is the expected service time of a lone occupant in a corridor of length l, and $f(n) = V_n/V_1$ is the service rate, considered to be the ratio of the average walking speed of n people in the corridor to that of a lone occupant V_1.

From Eq. (4), one can derive all performance measures of interest:

$$\left\{ \begin{array}{lcl} \theta & = & \lambda\Big(1 - p(c)\Big), \\ L & = & E(N) = \sum_{n=1}^{c} \Big(np(n)\Big), \\ W & = & L/\theta, \end{array} \right. \tag{5}$$

in which $p(c)$ is the blocking probability, θ is the throughput, L is the expected number of users in the system (known as work-in-process, WIP), and W, derived from Little's formula, is the mean waiting (service) time in seconds.

Concerning the simulation, the computation is nothing more difficult (see Figure 5):

$$\left\{ \begin{array}{lcl} p(c) & \leftarrow & \texttt{sumBlock/sumArr}, \\ \theta & \leftarrow & \texttt{sumDep/totalTime}, \\ L & \leftarrow & \texttt{sumTime/totalTime}, \\ W & \leftarrow & \texttt{sumTime/sumDep}. \end{array} \right. \tag{6}$$

All simulation results were carried out for 20,000 seconds and 30 replications, for a corridor 8 meters long × 2.5 meters wide. The results are presented in Tables 1, 2, and 3, and in Figures 9, 10, and 11, which show the

```
algorithm Simulate
        /* initialize event queue */
    Inicitialize(MgccEventQueue);
        /* create and insert 'last' event */
    MgccEvent ← new();
    MgccEvent.occurTime ← totalTime;
    MgccEvent.type ← end_simulation;
    Insert(MgccEventQueue,MgccEvent);
        /* create and insert 'first' events */
    for $\forall n| \lambda_n \neq 0$ do
        MgccEvent ← new();
        MgccEvent.whichQueue ← $n$;
        MgccEvent.occurTime ← 0.0;
        MgccEvent.type ← arrival;
        Insert(MgccEventQueue,MgccEvent);
    end for
        /* simulate */
    MgccEvent ← GetNext(MgccEventQueue);
    while MgccEvent.type ≠ end_simulation do
        if MgccEvent.type = arrival then
            ProcessArrival(MgccEventQueue,MgccEvent);
        else if MgccEvent.type = departure then
            ProcessDeparture(MgccEventQueue,MgccEvent);
        else
            error, unknown event
        end if
        MgccEvent ← GetNext(MgccEventQueue);
    end while
    print results
end algorithm
```

Figure 6: Simulation algorithm.

```
algorithm ProcessArrival(EventQueue,Event)
        /* program next arrival */
    newEvent ← new();
    newEvent.whichQueue ← Event.whichQueue;
    newEvent.occurTime ~ Exp($\lambda$)+ Event.occurTime;
    newEvent.type ← arrival;
    Insert(EventQueue,newEvent);
        /* process this arrival */
    if queue is blocked then
        reject arrival and count it
    else
        newEvent ← new();
        newEvent.whichQueue ← Event.whichQueue;
        newEvent.occurTime ← $l/V_n$+ Event.occurTime;
        newEvent.type ← departure;
        Insert(EventQueue,newEvent);
        count arrival
            /* delay departure of every user in this queue
        DelayDeparture(EventQueue,Event.whichQueue);
    end if
end algorithm
```

Figure 7: Algorithm `ProcessArrival`.

```
algorithm ProcessDeparture(EventQueue,Event)
    generate next destination queue $d$
    if final destination then
        count departure
            /* advance departure of users */
        AdvanceDeparture(EventQueue,Event.whichQueue);
    else if queue is blocked then
        postpone departure and count it
    else
            /* create and insert departure */
        newEvent ← new();
        newEvent.whichQueue ← $d$;
        newEvent.occurTime ← $l/V_n$+ Event.occurTime;
        newEvent.type ← departure;
        Insert(EventQueue,newEvent);
        count departure
        count arrival
            /* advance departure of users */
        AdvanceDeparture(EventQueue,Event.whichQueue);
            /* delay departure of users */
        DelayDeparture(EventQueue,$d$);
    end if
end algorithm
```

Figure 8: Algorithm `ProcessDeparture`.

behavior of state dependent service rate systems under different arrival rates, corridor widths, and lengths. The CPU columns show the times for the total set of 30 replications. The confidence intervals are too narrow to be noticeable in the figures and the results seem to indicate a close agreement of the simulation model.

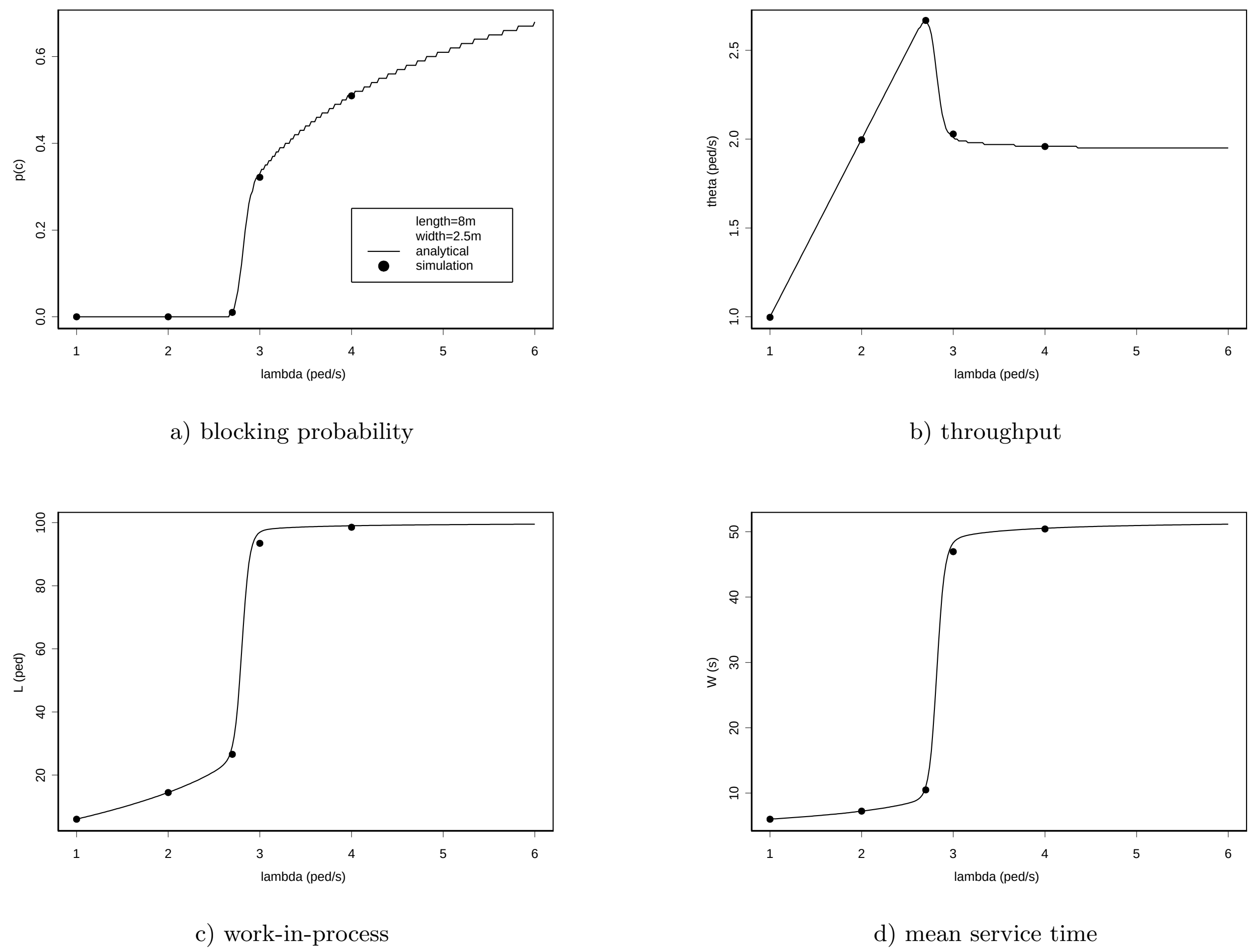
Table 1 and Figure 9 show the performance measures under variable arrival rates. Note the dynamic, non-linear performance of $M/G/C/C$ nodes with the $\theta \times \lambda$ curve being the most dramatic, Figure 9-b. It is seen that the blocking is low and the throughput is linear up to 2.7 ped/s, Figure 9-a. That seems to be the limit of this corridor. From that arrival rate on, the blocking increases, the throughput drops, the number of users also increases up to the system capacity, and the service time worsens around 8 times. Arrival rates above 3 ped/s cannot improve the system throughput that reaches its limit around $\theta = 2$ ped/s. Surprisingly, the system would be able to give a higher throughput (around 2.7 ped/s), under a slightly lower arrival rate of 2.7 ped/s. Note some divergence between analytical and simulated results around the arrival rates 2.7 and 3.0.

Under a scenario in which the width is changing, Table 2 and Figure 10, it is seen that the simulation and analytical results are in close agreement, except perhaps slight differences around the critical width values, 2.0 m, for $\lambda = 2.5$, and 4.0–4.5 m, for $\lambda = 5.0$. The corridor's width appears to be perhaps the most important control variable, as its increase reduces the blocking probability, increases the throughput, reduces the number of users in the system, and reduces the service time. However, it is also seen that no sensible effect at all is felt after the critical widths. In other words, there was no further improvements in the performance measures considered beyond 2.0 m, for an arrival rate of 2.5 ped/s, and 4.5 m, for an arrival rate of 5.0 ped/s. As a final remark, note how heavily dependent on the arrival rate the critical widths are.

Finally, changing the corridor length one can confirm once more the accuracy of the simulation model, as shown in Table 3 and Figure 11. It is remarkable that the length has a (linear) influence on the average number of users and in the service time. However, the length

Table 1: Single node performance measures *versus* arrival rate.

λ	Model	$p(c)$	θ	L	W	CPU (s)
1.0	Analytical	0.00	1.00	6.02	6.02	
	Simulation	0.00	1.00	6.00	6.02	33
	95% CI	[0.00;0.00]	[1.00;1.00]	[5.99;6.02]	[6.02;6.02]	
2.0	Analytical	0.00	2.00	14.49	7.24	
	Simulation	0.00	2.00	14.46	7.24	140
	95% CI	[0.00;0.00]	[1.99;2.00]	[14.42;14.49]	[7.23;7.25]	
2.7	Analytical	0.01	2.66	29.20	10.98	
	Simulation	0.01	2.67	27.91	10.49	350
	95% CI	[0.00;0.02]	[2.65;2.68]	[26.28;29.55]	[9.81;11.18]	
3.0	Analytical	0.33	2.01	96.96	48.31	
	Simulation	0.32	2.03	95.21	46.95	890
	95% CI	[0.32;0.33]	[2.02;2.04]	[94.52;95.90]	[46.37;47.53]	
4.0	Analytical	0.51	1.96	99.01	50.53	
	Simulation	0.51	1.96	98.77	50.43	910
	95% CI	[0.51;0.51]	[1.96;1.96]	[98.75;98.79]	[50.41;50.45]	



a) blocking probability

b) throughput

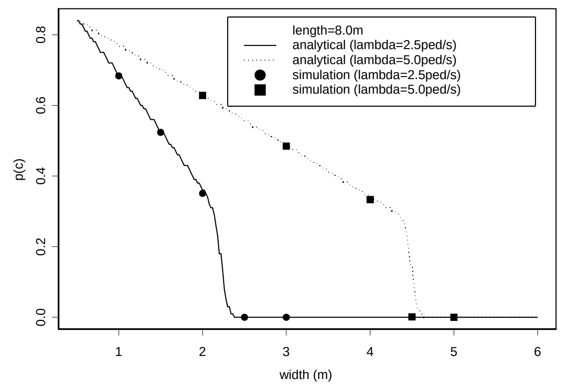
c) work-in-process

d) mean service time

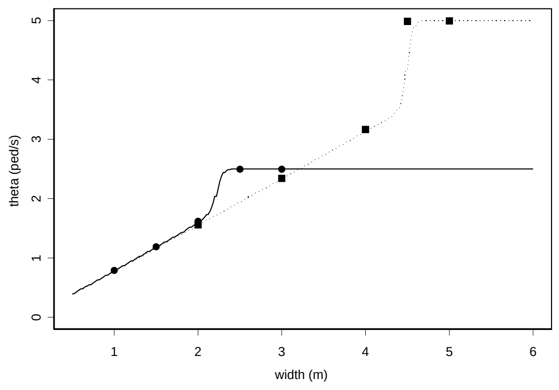
Figure 9: Single node performance measures *versus* arrival rate.

Table 2: Single node performance measures *versus* width.

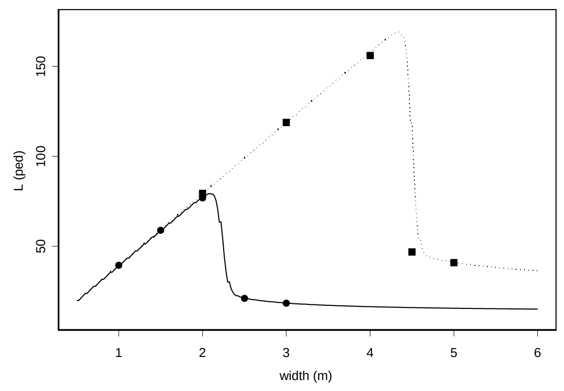
λ	Width	Model	$p(c)$	θ	L	W	CPU (s)
2.5	1.0	Analytical	0.68	0.79	39.53	50.02	
		Simulation	0.68	0.79	39.46	49.98	150
		95% CI	[0.68;0.68]	[0.79;0.79]	[39.45;39.46]	[49.96;49.99]	
	1.5	Analytical	0.52	1.19	59.05	49.69	
		Simulation	0.52	1.19	58.91	49.59	330
		95% CI	[0.52;0.52]	[1.19;1.19]	[58.90;58.93]	[49.57;49.61]	
	2.0	Analytical	0.36	1.61	77.71	48.32	
		Simulation	0.35	1.62	76.87	47.50	580
		95% CI	[0.35;0.35]	[1.61;1.62]	[76.63;77.11]	[47.23;47.76]	
	2.5	Analytical	0.00	2.50	21.07	8.43	
		Simulation	0.00	2.50	21.00	8.41	250
		95% CI	[0.00;0.00]	[2.49;2.50]	[20.94;21.06]	[8.40;8.43]	
	3.0	Analytical	0.00	2.50	18.39	7.36	
		Simulation	0.00	2.50	18.35	7.35	220
		95% CI	[0.00;0.00]	[2.49;2.50]	[18.31;18.39]	[7.34;7.36]	
5.0	2.0	Analytical	0.69	1.56	79.54	51.00	
		Simulation	0.69	1.56	79.40	50.95	600
		95% CI	[0.69;0.69]	[1.56;1.56]	[79.39;79.40]	[50.95;50.96]	
	3.0	Analytical	0.53	2.34	119.10	50.86	
		Simulation	0.53	2.34	118.83	50.76	1300
		95% CI	[0.53;0.53]	[2.34;2.34]	[118.81;118.84]	[50.75;50.77]	
	4.0	Analytical	0.37	3.14	158.24	50.46	
		Simulation	0.37	3.17	156.04	49.29	2400
		95% CI	[0.36;0.37]	[3.16;3.17]	[155.55;156.53]	[49.01;49.57]	
	4.5	Analytical	0.11	4.45	95.66	21.49	
		Simulation	0.00	4.99	46.80	9.39	1100
		95% CI	[0.00;0.00]	[4.97;5.00]	[45.54;48.05]	[9.10;9.68]	
	5.0	Analytical	0.00	5.00	40.94	8.19	
		Simulation	0.00	5.00	40.88	8.18	960
		95% CI	[0.00;0.00]	[4.99;5.00]	[40.80;40.96]	[8.18;8.19]	



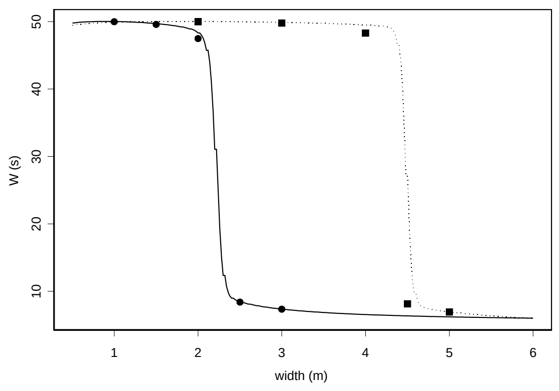
a) blocking probability



b) throughput



c) work-in-process

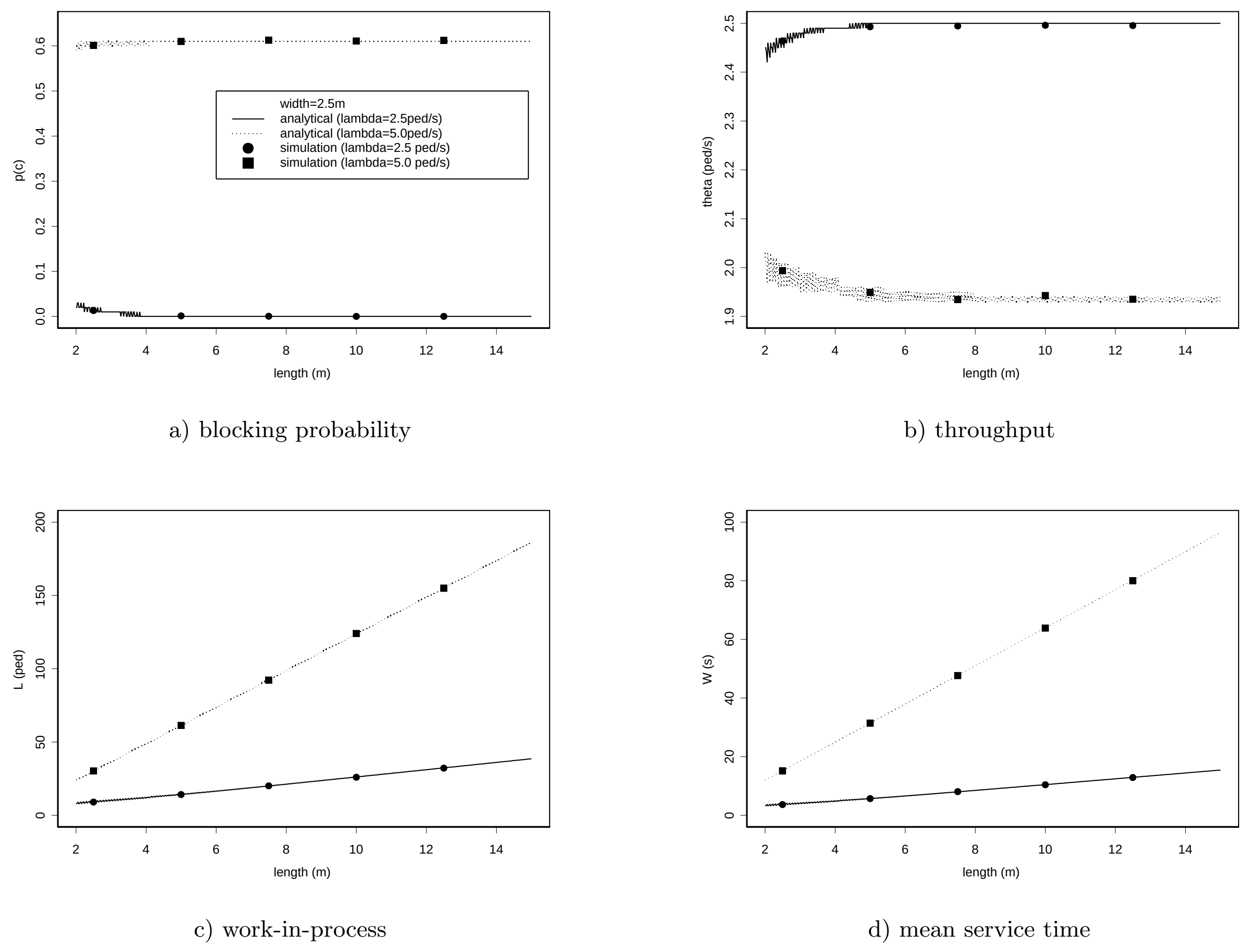


d) mean service time

Figure 10: Single node performance measures *versus* width.

Table 3: Single node performance measures *versus* length.

λ	Width	Model	$p(c)$	θ	L	W	CPU (s)
2.5	2.5	Analytical	0.01	2.47	9.08	3.68	
		Simulation	0.01	2.46	9.03	3.66	110
		95% CI	[0.01;0.01]	[2.46;2.47]	[8.94;9.12]	[3.63;3.70]	
	5.0	Analytical	0.00	2.50	14.18	5.68	
		Simulation	0.00	2.49	14.15	5.67	170
		95% CI	[0.00;0.00]	[2.49;2.50]	[14.01;14.28]	[5.62;5.73]	
	7.5	Analytical	0.00	2.50	20.05	8.02	
		Simulation	0.00	2.49	20.11	8.06	240
		95% CI	[0.00;0.00]	[2.49;2.50]	[19.83;20.39]	[7.95;8.18]	
	10.0	Analytical	0.00	2.50	26.01	10.41	
		Simulation	0.00	2.50	25.94	10.39	310
		95% CI	[0.00;0.00]	[2.49;2.50]	[25.86;26.01]	[10.38;10.41]	
	12.5	Analytical	0.00	2.50	32.27	12.91	
		Simulation	0.00	2.50	32.18	12.89	380
		95% CI	[0.00;0.00]	[2.49;2.50]	[32.09;32.27]	[12.88;12.91]	
5.0	2.5	Analytical	0.60	1.99	30.30	15.19	
		Simulation	0.60	1.99	30.28	15.19	290
		95% CI	[0.60;0.60]	[1.99;1.99]	[30.28;30.29]	[15.19;15.20]	
	5.0	Analytical	0.61	1.95	61.34	31.46	
		Simulation	0.61	1.95	61.27	31.44	590
		95% CI	[0.61;0.61]	[1.95;1.95]	[61.26;61.28]	[31.43;31.44]	
	7.5	Analytical	0.61	1.94	92.36	47.71	
		Simulation	0.61	1.93	92.19	47.66	850
		95% CI	[0.61;0.61]	[1.93;1.93]	[92.19;92.20]	[47.65;47.66]	
	10.0	Analytical	0.61	1.94	124.36	63.94	
		Simulation	0.61	1.94	124.05	63.85	1200
		95% CI	[0.61;0.61]	[1.94;1.94]	[124.04;124.06]	[63.84;63.86]	
	12.5	Analytical	0.61	1.94	155.36	80.18	
		Simulation	0.61	1.94	154.89	80.03	1500
		95% CI	[0.61;0.61]	[1.94;1.94]	[154.87;154.90]	[80.02;80.04]	



a) blocking probability

b) throughput

c) work-in-process

d) mean service time

Figure 11: Single node performance measures *versus* length.

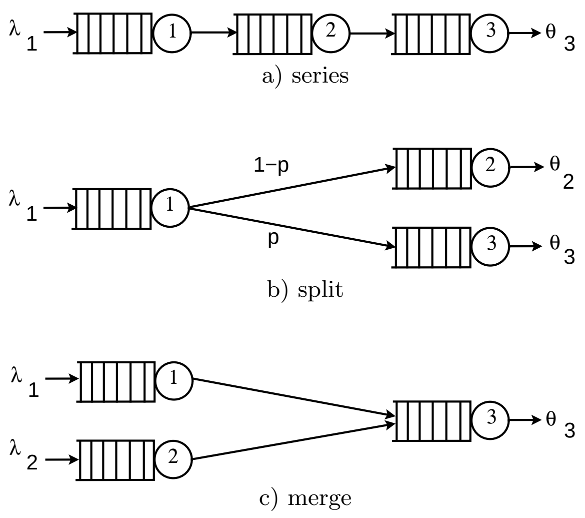


Figure 12: Basic topologies.

is not an effective control variable because the blocking probability and the throughput, perhaps the most regarded performance measures, are almost independent. Fortunately, in most of the practical cases, the length is not under control and it is merely a distance that must be covered.

4.2 Generalized Expansion Method (GEM) Validation

The GEM, proposed by Kerbache and MacGregor Smith [12, 13], is an analytical model for the analysis and design of finite queueing networks for an arbitrary topology. The method is a combination of repeated trials and node-by-node decomposition approximation methods. Its key characteristic is to add an artificial holding node preceding each finite queue in the network in order to register blocked customers that attempt to join the finite node when it is at its capacity. Doing so, the queueing networks are 'expanded' into an equivalent Jackson network in which each node can be decomposed and analyzed separately. Since the GEM was developed, its has been used successfully in many finite capacity systems [1, 2, 6, 7, 15]. Details on how the GEM can be adapted to $M/G/C/C$ state dependent queueing networks will not be given here but are easily found in the literature [2, 9].

Few simulation results are available on how well the GEM approximates the actual performance measures [3, 9], largely because of the inherent complexity of dynamically updating the service rate in the simulation model. This section provides comparisons for three basic topologies, series, split and merge (see Figure 12), as it is seen in Tables 4, 5, and 6. All simulations were run for 20,000 seconds and 30 replications were performed to build 95% confidence intervals.

Table 4 shows the results for a three node series topology, composed by three corridors 8.0 meters long × 2.5 meters wide and an arrival $\lambda_1 = 3.0$ ped/s. On can see the surprisingly accuracy of the GEM in such a case. Almost all analytical (approximate) performance measure were covered by the 95% confidence intervals.

The results for the split topology are shown in Table 5. The simulation considered three corridors, also 8 × 2.5 meters, a split probability $p = 0.6$, and an arrival rate $\lambda_1 = 3.0$ ped/s. Again, the results confirm the accuracy of the GEM. Almost all 95% confidence intervals covered the analytical results.

Finally, Table 6 presents the results for three corridors, 8.0 × 2.5 meters, in a merge topology, for an arrival rate $\lambda_1 = \lambda_2 = 3.0$ ped/s. The relative complexity of this topology may be better understood by the CPU time spent which is considerably higher than those for the previous topologies. Here, a remarkable agreement between analytical and simulation results may be seen for all queues, for which the confidence intervals are pretty close to the analytical results. Usually, in optimization applications for example, the low processing time is very important since multiple evaluation of alternative configurations usually must to be done by the search algorithm. For these situations, the GEM should be an effective tool.

5 Network Evacuation Applications

One of the main intended uses of the simulation model is for the network evacuation problem. This problem occurs in the evacuation of pedestrians from buildings, naval vessels, chemical plant explosions, as well as for urban and rural evacuation of vehicular traffic in case of hurricanes, fires, floods or other natural and man-made disasters.

In this regard, the queueing network of corridors, stairwells, ramps, streets, and passageways become crucial links for the safe evacuation of the occupants of the affected regions or buildings. Of main concern in this modeling problem is to identify the key bottlenecks constricting the flow of occupants out of the affected area. Once the bottlenecks can be identified, then additional capacity can be directed towards alleviating the cause of the constriction, and thus a safer, more efficient evacuation process could result.

In the demonstration below, a network evacuation from a ten-story high rise building is examined, and the efficacy of the simulation model is shown for this type of application.

5.1 Example Application

An interesting and relevant application for the simulation system just proposed may be illustrated by the performance analysis of a network evacuation of pedestrian occupants from a ten-story high-rise building, see Figure 13. An 8.5 meter in length × 1.2 meter wide stairwell is assumed to interconnect each floor and each stairwell is modeled as an $M/G/C/C$ state dependent queue. Thus, there will be a total of 10 $M/G/C/C$ state dependent queues in a series-merge network topology. Also assumed is an arrival rate λ equally assigned for each of the ten

Table 4: Results for 3-node series topology ($\lambda_1 = 3.0$).

	node 1			node 2			node 3		
	Analytical	Simulation*		Analytical	Simulation*		Analytical	Simulation*	
Measure			95% CI			95% CI			95% CI
$p(c)$	0.33	0.33	[0.32 ; 0.33]	0.00	0.01	[0.00 ; 0.01]	0.00	0.01	[0.00 ; 0.03]
θ	2.01	2.02	[2.01 ; 2.03]	2.01	2.02	[2.01 ; 2.02]	2.01	2.02	[2.01 ; 2.02]
L	96.96	95.87	[95.35;96.40]	14.56	16.44	[15.00;17.88]	14.56	16.51	[15.23;17.79]
W	48.31	47.53	[47.10;47.96]	7.26	8.15	[7.44 ; 8.86]	7.26	8.19	[7.56 ; 8.81]

*CPU time = 4 minutes 55 seconds

Table 5: Results for 3-node split topology ($\lambda_1 = 3.0$).

	node 1			node 2			node 3		
	Analytical	Simulation*		Analytical	Simulation*		Analytical	Simulation*	
Measure			95% CI			95% CI			95% CI
$p(c)$	0.33	0.32	[0.32 ; 0.33]	0.00	0.00	[0.00;0.00]	0.00	0.00	[0.00;0.00]
θ	2.01	2.03	[2.02 ; 2.04]	1.20	1.22	[1.21;1.23]	0.80	0.81	[0.81;0.82]
L	96.96	95.04	[94.23;95.84]	7.48	7.61	[7.55;7.67]	4.70	4.76	[4.73;4.80]
W	48.31	46.82	[46.16;47.49]	6.21	6.24	[6.23;6.26]	5.86	5.87	[5.86;5.88]

*CPU time = 2 minutes 58 seconds

floors (queues). The floor arrival rates complicate the analysis of the model since depending on the flow rates of the various floors, significant blocking can occur. For the sake of the argument, the uniform arrival rates for each floor are used in the experimental results of the paper, however, non-uniform rates could be used as well in the simulation model.

The simulation was carried out for 20,000 seconds and 30 replications were made. Longer and shorter simulation times were tested (not shown) but 20,000 seconds represents the best compromise between low variability among replications and low CPU time consumption.

5.2 Experimental Results

Table 7 shows the performance measures for each floor, obtained for four different arrival rates, as well as the respective 95% confidence intervals. The CPU times were heavily dependent on the arrival rate λ, as it is seen in the bottom of Table 7. The confidence intervals were narrow, excepting perhaps for the lower floors under light traffic, which presented a considerable higher variability, understandable because they reflect the accumulated blocking and congestion of the upper levels.

The importance of the simulation model is that it can predict the bottleneck of the dynamic network evacuation system and how it moves as a function of the incoming traffic from each floor level. This type of prediction of the bottleneck is crucial in any network evacuation model planning, whether it is for pedestrians or for vehicles.

While the issue of the optimal width will not be addressed in this paper, it will be the subject of future papers. The simulation model becomes an important tool in the performance evaluation of alternative building designs for the optimal egress problem because it can help planners identify the bottlenecks and correlate the nonlinear effects of traffic loads on the egress system.

These data can be better appreciated plotted in graphs, as shown in Figure 14. Here one can see the effects of the arrival traffic from each floor on the network performance measures. The blocking probability $p(c)$ rises very quickly on these equivalent-width stairwells. The only unknown variable is where the bottleneck will occur. The bottleneck increases in the upper levels, under heavy traffic. The bottleneck increases in lower levels, under light traffic. Under high traffic, the throughput rapidly saturates to the system capacity (around 1.0 ped/s), and the number of users trapped between floors rises quickly to the stairwell capacity of 51 pedestrians, and the travel time becomes excessive, around 55 seconds, almost 8 times higher than the 6 seconds spent by a lone occupant.

6 Summary and Conclusion

In this paper, the scope and importance of $M/G/C/C$ state dependent queueing networks were examined, and an effective discrete-event digital simulation model was presented to model these dynamic traffic flows. Some of the most universal and significant applications where finite state dependent $M/G/C/C$ networks occur include pedestrian and vehicular traffic flows where customers compete for the limited movement space. The simulation model was validated and characteristics of a single $M/G/C/C$ state dependent queue were discussed. The simulation model confirmed the accuracy of the Generalized Expansion Method, an approximation technique largely used in the analysis and syntheses of finite queueing network systems. The simulation model was also used to analyze the problem of evacuation of a ten-story building and the results indicate that the simulation model is an extremely valuable tool when planning emergency egress for buildings.

Acknowledgments

The work of Frederico R. B. Cruz has been partially funded by the CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*) of the Ministry for Sci-

Table 6: Results for 3-node merge topology ($\lambda_1 = \lambda_2 = 3.0$).

	node 1			node 2			node 3		
	Analytical	Simulation*		Analytical	Simulation*		Analytical	Simulation*	
Measure			95% CI			95% CI			95% CI
$p(c)$	0.67	0.68	[0.68 ; 0.68]	0.67	0.68	[0.68 ; 0.68]	0.51	0.50	[0.50 ; 0.50]
θ	0.98	0.97	[0.97 ; 0.97]	0.98	0.97	[0.97 ; 0.97]	1.96	1.93	[1.93 ; 1.93]
L	99.51	98.63	[98.60;98.67]	99.51	98.61	[98.56;98.67]	99.02	99.76	[99.75;99.76]
W	101.6	102.0	[101.8;102.2]	101.6	101.9	[101.8;102.1]	50.54	51.70	[51.70;51.71]

*CPU time = 3 hours 0 minutes 28 seconds

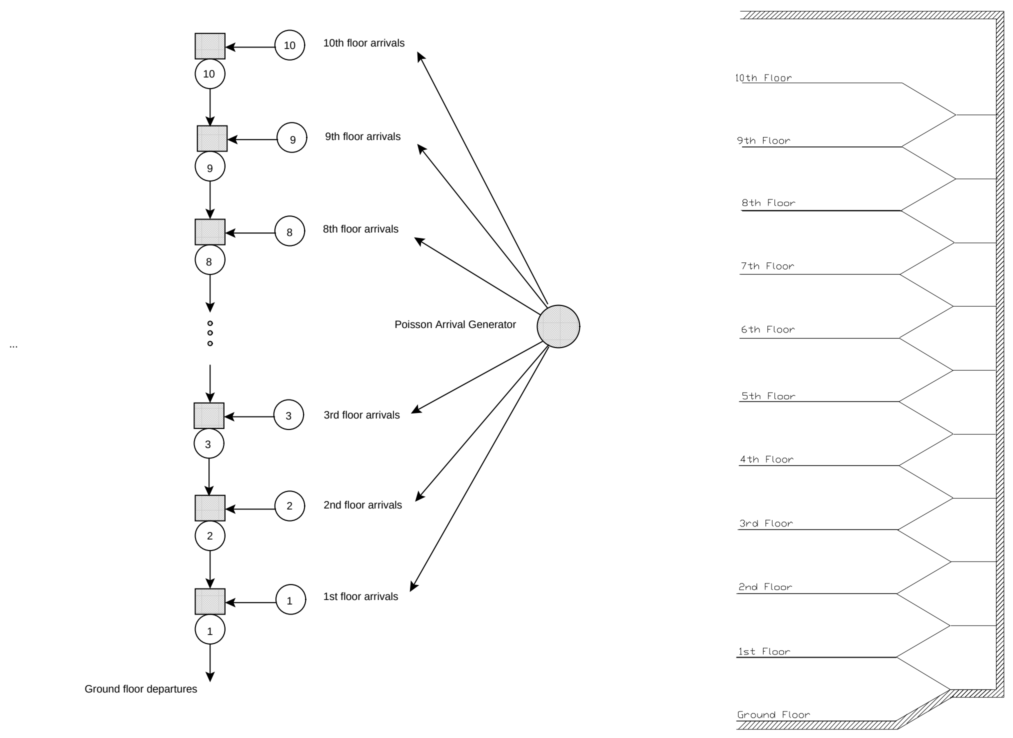


Figure 13: A ten-story building.

ence and Technology of Brazil, grants 301809/96-8 and 201046/94-6, the FAPEMIG, grants CEX-289/98 and CEX-855/98, and PRPq-UFMG, grant 4081-UFMG/ RTR/ FUNDO/ PRPq/99. Reinaldo O. Medeiros is funded by the CNEN (National Commission of Nuclear Energy - *Comissão Nacional de Energia Nuclear*) and the CDTN (Nuclear Technology Development Center - *Centro de Desenvolvimento da Tecnologia Nuclear*) of the Ministry for Science and Technology of Brazil. The authors thank the support of Dr. Clédola Cássia Oliveira de Tello, from the CDTN, and Dr. Enrico Antônio Colosimo, from the UFMG.

References

[1] Yuhaski SJ, MacGregor Smith J. Modeling circulation systems in buildings using state dependent models. *Queueing Systems* 1989; 4:319–338.

[2] Mitchell DH, MacGregor Smith J. Topological network design of pedestrian networks. *Transportation Research Part B* 2001; 35:107–135.

[3] Jain R, MacGregor Smith J. Modeling vehicular traffic flow using $M/G/C/C$ state dependent queueing models. *Transportation Science* 1997; 31(4):324–336.

[4] Thumsi V, MacGregor Smith J. $M/G/C/C$ state-dependent service rates in material handling systems. Manuscript 1998.

[5] Tregenza PR. *The Design of Interior Circulation.* New York: Van Nostrand Reinhold Company, 1976.

[6] MacGregor Smith J. Application of state-dependent queues to pedestrian/vehicular network design. *Operations Research* 1994; 42:414–427.

[7] Bakuli DL, MacGregor Smith J. Resource allocation in state-dependent emergency evacuation networks.

Table 7: Building simulation results.

		$\lambda = 1.000$[a]		$\lambda = 0.500$[b]		$\lambda = 0.250$[c]		$\lambda = 0.125$[d]	
Floor			95% CI		95% CI		95% CI		95% CI
10	$p(c)$	0.10	[0.10 ; 0.10]	0.00	[0.00 ; 0.00]	0.00	[0.00 ; 0.00]	0.00	[0.00 ; 0.00]
	θ	0.90	[0.90 ; 0.90]	0.50	[0.50 ; 0.50]	0.25	[0.25 ; 0.25]	0.12	[0.12 ; 0.13]
	L	43.37	[43.05;43.69]	3.46	[3.43 ; 3.49]	1.51	[1.51 ; 1.52]	0.73	[0.72 ; 0.73]
	W	48.33	[47.95;48.70]	6.92	[6.88 ; 6.97]	6.04	[6.03 ; 6.04]	5.84	[5.84 ; 5.84]
9	$p(c)$	0.67	[0.67 ; 0.67]	0.18	[0.17 ; 0.18]	0.00	[0.00 ; 0.00]	0.00	[0.00 ; 0.00]
	θ	0.92	[0.91 ; 0.92]	0.90	[0.90 ; 0.90]	0.50	[0.50 ; 0.50]	0.25	[0.25 ; 0.25]
	L	50.83	[50.83;50.84]	44.55	[44.23;44.87]	3.26	[3.25 ; 3.27]	1.51	[1.50 ; 1.51]
	W	55.53	[55.48;55.57]	49.48	[49.12;49.85]	6.51	[6.51 ; 6.52]	6.03	[6.03 ; 6.04]
8	$p(c)$	0.66	[0.66 ; 0.66]	0.59	[0.59 ; 0.60]	0.00	[0.00 ; 0.00]	0.00	[0.00 ; 0.00]
	θ	0.92	[0.92 ; 0.92]	0.92	[0.92 ; 0.92]	0.75	[0.75 ; 0.75]	0.37	[0.37 ; 0.38]
	L	50.86	[50.85;50.86]	50.72	[50.70;50.73]	6.48	[6.42 ; 6.55]	2.34	[2.33 ; 2.35]
	W	55.22	[55.19;55.24]	55.39	[55.35;55.42]	8.65	[8.57 ; 8.72]	6.26	[6.25 ; 6.26]
7	$p(c)$	0.65	[0.65 ; 0.66]	0.59	[0.58 ; 0.59]	0.29	[0.28 ; 0.29]	0.00	[0.00 ; 0.00]
	θ	0.92	[0.92 ; 0.92]	0.92	[0.92 ; 0.92]	0.91	[0.90 ; 0.91]	0.50	[0.50 ; 0.50]
	L	50.85	[50.85;50.86]	50.79	[50.78;50.81]	46.02	[45.83;46.21]	3.25	[3.24 ; 3.26]
	W	55.06	[55.04;55.08]	55.15	[55.13;55.17]	50.85	[50.64;51.07]	6.51	[6.50 ; 6.51]
6	$p(c)$	0.64	[0.64 ; 0.64]	0.57	[0.57 ; 0.58]	0.54	[0.54 ; 0.54]	0.00	[0.00 ; 0.00]
	θ	0.93	[0.92 ; 0.93]	0.92	[0.92 ; 0.92]	0.92	[0.92 ; 0.92]	0.62	[0.62 ; 0.63]
	L	50.86	[50.85;50.87]	50.78	[50.76;50.79]	50.41	[50.38;50.44]	4.25	[4.23 ; 4.26]
	W	54.98	[54.96;55.00]	54.97	[54.94;54.99]	54.95	[54.91;54.99]	6.80	[6.80 ; 6.81]
5	$p(c)$	0.63	[0.62 ; 0.63]	0.56	[0.55 ; 0.56]	0.52	[0.52 ; 0.53]	0.00	[0.00 ; 0.00]
	θ	0.93	[0.93 ; 0.93]	0.93	[0.92 ; 0.93]	0.92	[0.92 ; 0.92]	0.75	[0.75 ; 0.75]
	L	50.86	[50.86;50.87]	50.79	[50.78;50.81]	50.63	[50.62;50.65]	5.36	[5.33 ; 5.38]
	W	54.93	[54.91;54.95]	54.90	[54.87;54.92]	54.91	[54.88;54.94]	7.16	[7.15 ; 7.17]
4	$p(c)$	0.61	[0.61 ; 0.61]	0.53	[0.53 ; 0.54]	0.51	[0.50 ; 0.51]	0.00	[0.00 ; 0.00]
	θ	0.93	[0.93 ; 0.93]	0.93	[0.93 ; 0.93]	0.92	[0.92 ; 0.92]	0.87	[0.87 ; 0.87]
	L	50.86	[50.86;50.87]	50.78	[50.76;50.79]	50.65	[50.62;50.68]	11.22	[10.70;11.74]
	W	54.89	[54.88;54.90]	54.81	[54.78;54.83]	54.77	[54.73;54.81]	12.87	[12.29;13.44]
3	$p(c)$	0.59	[0.59 ; 0.60]	0.51	[0.50 ; 0.51]	0.48	[0.47 ; 0.48]	0.34	[0.32 ; 0.36]
	θ	0.93	[0.93 ; 0.93]	0.93	[0.93 ; 0.93]	0.93	[0.93 ; 0.93]	0.93	[0.93 ; 0.94]
	L	50.86	[50.85;50.87]	50.79	[50.78;50.80]	50.62	[50.59;50.65]	40.26	[37.75;42.77]
	W	54.84	[54.83;54.85]	54.77	[54.76;54.79]	54.62	[54.57;54.67]	43.27	[40.45;46.08]
2	$p(c)$	0.58	[0.58 ; 0.58]	0.48	[0.47 ; 0.48]	0.43	[0.43 ; 0.44]	0.40	[0.37 ; 0.43]
	θ	0.93	[0.93 ; 0.93]	0.93	[0.93 ; 0.93]	0.93	[0.93 ; 0.93]	0.96	[0.95 ; 0.97]
	L	50.86	[50.86;50.87]	50.79	[50.77;50.80]	50.55	[50.44;50.67]	43.16	[40.58;45.74]
	W	54.82	[54.80;54.83]	54.72	[54.70;54.74]	54.36	[54.22;54.50]	45.13	[42.13;48.13]
1	$p(c)$	0.56	[0.56 ; 0.56]	0.44	[0.43 ; 0.44]	0.24	[0.17 ; 0.31]	0.27	[0.23 ; 0.31]
	θ	0.93	[0.93 ; 0.93]	0.94	[0.93 ; 0.94]	1.03	[0.99 ; 1.08]	1.02	[0.99 ; 1.04]
	L	50.85	[50.83;50.86]	50.28	[49.83;50.73]	34.44	[27.25;41.62]	35.73	[31.61;39.85]
	W	54.64	[54.62;54.66]	53.64	[52.84;54.43]	35.98	[27.87;44.09]	35.80	[31.23;40.37]
CPU:		[a]1 h 40 min 50 sec		[b]1 h 20 min 40 sec		[c]48 min 00 sec		[d]9 min 42 sec	

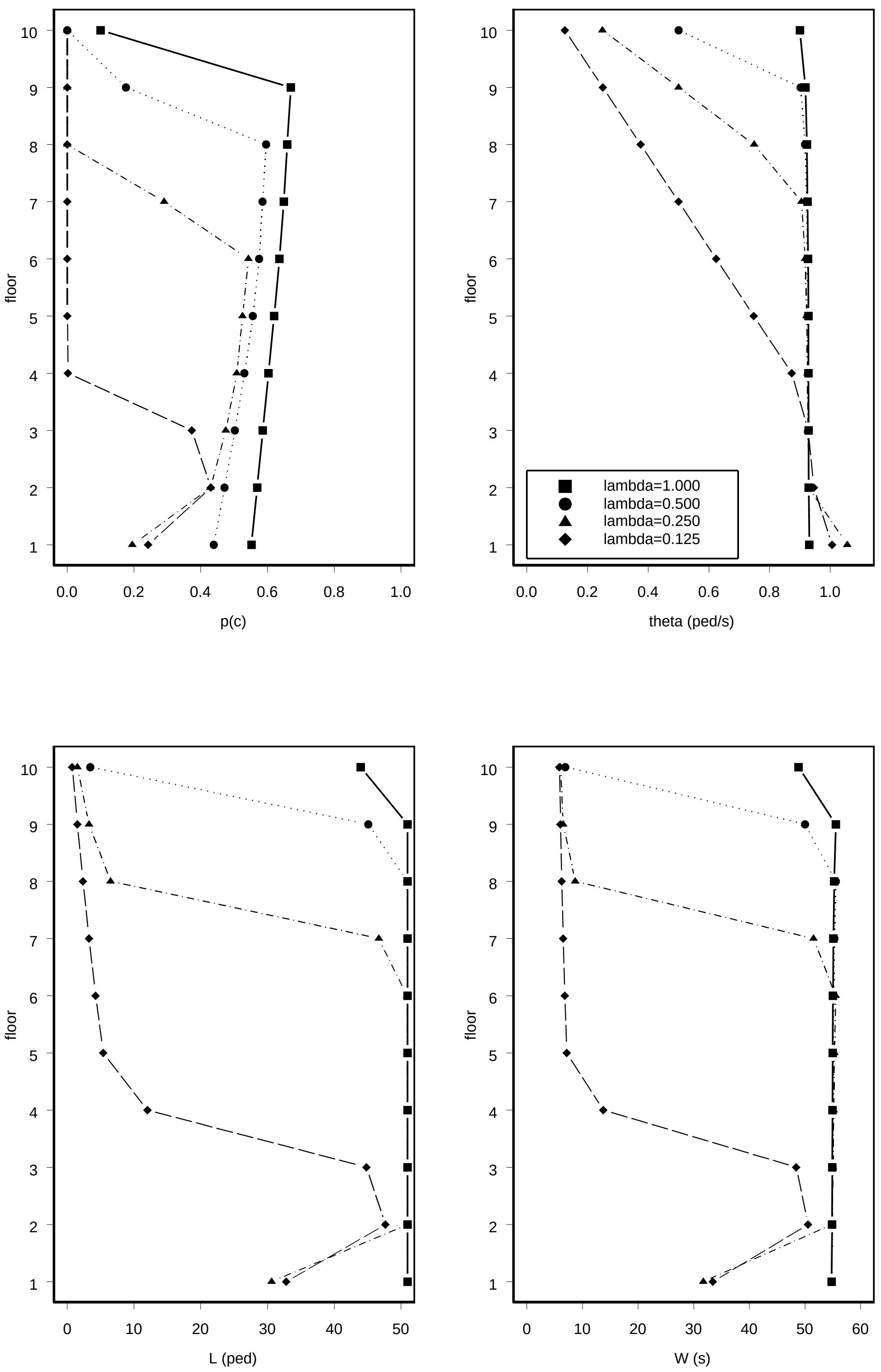


Figure 14: Building simulation results.

European Journal of Operational Research 1996; 89: 543–555.

[8] Fruin JJ. *Pedestrian Planning and Design.* New York: Metropolitan Association of Urban Designers and Environmental Planners, Inc., 1971.

[9] Cheah J, MacGregor Smith J. Generalized $M/G/C/C$ state dependent queueing models and pedestrian traffic flows. *Queueing Systems* 1994; 15: 365–386.

[10] Fishman GS. *Discrete-Event Simulation: Modeling, Programming, and Analysis.* New York: Springer, 2001.

[11] Nance RE. A history of discrete event simulation programming languages. In Bergin TJ, Gibson RJ, (Eds.). *History of Programming Languages.* ACM Press and Addison-Wesley Publishing Company, 1996. p. 369–427.

[12] Kerbache L, MacGregor Smith J. The generalized expansion method for open finite queueing networks. *European Journal of Operational Research* 1987; 32:448–461.

[13] Kerbache L, MacGregor Smith J. Asymptotic behavior of the expansion method for open finite queueing networks. *Computers & Operations Research* 1988; 15(2):157–169.

[14] Kelton D, Sadowski RP, Sadowski DA. *Simulation with Arena.* New York: MacGraw Hill College Div., 2001.

[15] MacGregor Smith J. Topological network design of state-dependent queueing networks. *Networks* 1996; 28:55–68.