Controlling the supermarket service

José A. Montero Valverde, Luis E. Sucar Succar

Reporte Técnico No. CCC-10-006
30 de septiembre de 2010

Luis Enrique Erro 1
Sta. Ma. Tonantzintla,
72840, Puebla, México.

Controlling the supermarket service

José Antonio Montero L. Enrique Sucar
Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
E-mail: {jamontero1@prodigy.net.mx, esucar@inaoep.mx }

Abstract

Cashiers are the service windows of supermarkets, which not only are the images of the supermarkets but also service quality and business efficiency. The long queue of customers waiting to be served by cashier is not an ideal. Most people prefer to give up or go away instead of waiting in queue. With similar quality and price, service quality is the key for prefer one or another supermarket. So the design of systems that analyses the queues in supermarkets and suggest opening or closing check−out stands service optimization−based approach, are of great support for managements. This work presents an approach that combines queueing theory and decision theory for design and build an automatic making decision human and assist support. The setting is a simulated supermarket service controlled by policies generated by Factored MDPs to minimize the long-term cost function. Preliminaries tests demostrates that the performance of the simulated model is according with the mathematical counterpart, and the actions selected by the controller are according the optimal policy.

1 Introduction

Currently, supermarkets understand that the overall service is the key for the success, and the service quality is the key for winning the competition in supermarkets with similar quality and price. For supermarkets, more cashiers mean more investments, however, few cashiers may lead to serious waiting, affecting service quality and causing loses of customers. Some times, managers make decisions that not always result the best option for the business efficiency, so the design of automatic systems that supports management decisions are of great help. In this work, we present the design of a simulator for the queue supermarket joined with a controller based on Factored Markov decision process (FMDPs) [1], for optimizing the supermarket service. Specifically, the overall designed system suggests the optimal number of cashiers in a planning period, this decision is based on minimum−cost approach. For getting the proposed system we integrate aspects of queueing theory and decision theory. In particular, queueing theory addresses problems related to the optimal design and control of queues and allows one to determine, for example, how to assign servers to diferent types of jobs or how many servers to employ so as to optimize some measure of system performance [2], [3]. Problems dealing with queue design are static in nature: the goal is to find optimal values for parameters which, once determined, become fixed characteristics of the queueing system, such as the maximum number of allowed customers [4], maximum waiting time for customers in queue [4] or the total number of available servers [5], [6], [7]. Problems dealing with queue (service) control are dynamic: the goal is to find the optimal operating policy, that is, rules for turning the server in occupied and idle, that result in the lowest long-run cost, see Cooper (1990). Then this models determine an optimal action to take when the queue is in a particular state [3], [8], [9], [10], [11], [12]. Some works propose solutions to problems dealing with optimal queue design and queue control, [13] [14].

The problem we address in this paper deals with both queue design and service control, since we would like to determine, based on minimum total cost (cashier cost + waiting cost), the optimal number of cashiers that should be attending in the supermarket (a fixed characteristic of the system) and how this cashiers should be selected during a planning horizon (what action should be taken in a particular state of the queue). For reaching this, we structured our work in two parts (See Figure 1).

Part **I**. Based on queueing theory we designed a simulation model for the supermarket queue system with C servers. Then we simulated the behaviour of the events (arrival rate and service rate) that occur when customers are waiting for service in the supermarket queue. Our simulator is based in the M/M/C (under the Kendall notation [15]) mathematical model, this model represents a Markovian process, where the first M means that arrival process (λ) is represented by a Poisson distribution, the second M indicates that the service time (μ) follows a exponential distribution, C means multiple servers. The parameter values obtained with the simulated model were compared for matching with the analytical mathematical model.

Part **II**. Based on the Factored Markov Decision Process (Factored MDPs) framework and using the parameters values of the supermarket queue model previously simulated, we obtain the policy for the controller. The optimal policy is constructed using Spudd [16] and the reward is based on a measurement of minimum cost reached during the planning period. At the moment, the actions selected by the policy, and that controller suggests are: (i) *Continues* (no changes), (ii) *Add one cashier*, and (iii) *Eliminate one cashier*. Preliminary tests indicate that selected actions by the controller are according to the observed states in the supermarket queue model.

The rest of the document is organized as follows. Section 2 presents an overview of the queuing theory. In Section 3 we present the design of the simulator for the supermarket queue model. In Section 4, we present the Markov Decision Process and the design of the contrroller based on Factored MDPs. In Seccion 5 we discuss related work. Finally, conclusions and extensions to the present work are described in Section 6.

2 Queuing theory

Queueing theory deals with problems which involve queuing (or waiting) [2]. A typical example might be supermarkets with customers waiting for service. As we know queues are a common every-day experience. Queues form because resources are limited. In fact it makes economic sense to have queues [3]. For example how many supermarket tills you would need to avoid queuing?, in designing queueing systems we need to aim for a balance between service to customers (short queues implying many servers) and economic considerations (not too many servers). In essence all queuing systems can be broken down into individual sub-systems consisting of entities queuing for some activity. Typically we can talk of this individual sub-system as dealing with customers queuing for service. To analyse this sub-system we need information related mainly to:

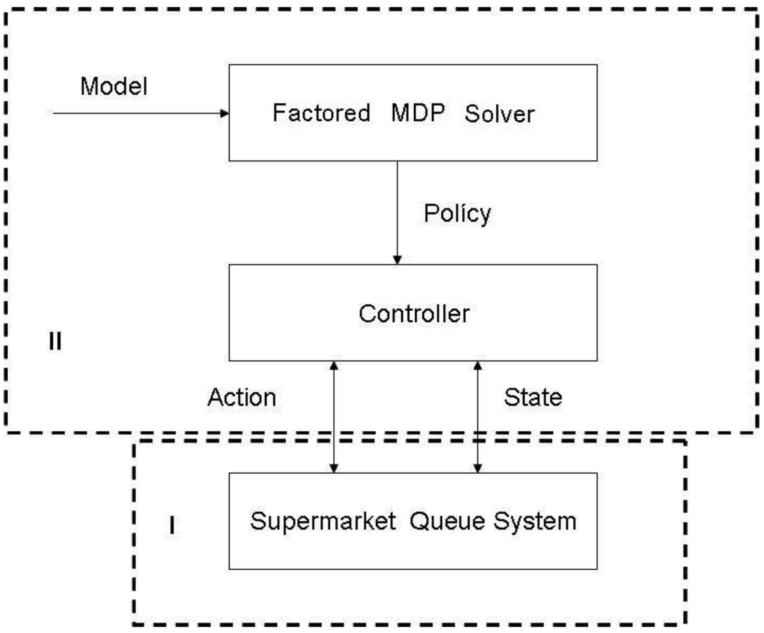


Figure 1. Components of the supermarket service controller system

- Arrival process (how customers arrive, how the arrivals are distributed in time, whether there is a finite population of customers or (effectively) an infinite number). The simplest arrival process is one where we have completely regular arrivals (i.e. the same constant time interval between successive arrivals). A Poisson stream of arrivals corresponds to arrivals at random. In a Poisson stream successive customers arrive after intervals which independently are exponentially distributed. The Poisson stream is important as it is a convenient mathematical model of many real life queuing systems and is described by a single parameter: the average arrival rate [17]. Other important arrival processes are scheduled arrivals; batch arrivals; and time dependent arrival rates (i.e. the arrival rate varies according to the time of day).
- Service mechanism (the service time distribution, the number of servers available, whether the servers are in series (each server has a separate queue) or in parallel (one queue for all servers)). Assuming that the service times for customers are independent and do not depend upon the arrival process is common. Another common assumption about service times is that they are exponentially distributed.

Queueing Theory tries to answer questions like the mean waiting time in the queue, the mean system response time (waiting time in the queue plus service times), mean utilization of the service facility, number of servers to be employed, average number of customers waiting in the queue, distribution of the number of customers in the queue, distribution of the number of customers in the system and so forth [5], [6], [7]. These questions are mainly investigated in a stochastic scenario, where the interarrival times of the customers or the service times are assumed to be random as said earlier.

In order to get answers to the above questions there are two basic approaches:

- Analytic methods or queuing theory (formula based); and

- Simulation (computer based).

The reason for there being two approaches (instead of just one) is that analytic methods are only available for relatively simple queuing systems. Complex queuing systems are almost always analysed using simulation (more technically known as discrete-event simulation). The simple queueing systems that can be tackled via queueing theory essentially: (i) consist of just a single queue; linked systems where customers pass from one queue to another cannot be tackled via queueing theory and (ii) have distributions for the arrival and service processes that are well defined (e.g. standard statistical distributions such as Poisson or Normal); systems where these distributions are derived from observed data, or are time dependent, are difficult to analyse via queueing theory.

2.1 Basic notation

A basic model of a service center (in this case a supermarket) is shown in Figure 2. The customers arrive to the service center in a random Poisson stream fashion. The service facility can have one or several servers, each server capable of serving one customer at a time, the service times needed for every customers are also modeled as random variables exponentially distributed. We make the following assumptions:

The customer population maybe finite or infinite size, in the supermarket setting the assumption is infinite population, the interarrival times are independent random stream and defined by $\frac{1}{\lambda}$.

The service times for each customer are also random variables. The average expected service time is obtained with $\frac{1}{\mu}$.

Queueing systems may not only differ in their distributions of the interarrival$-$ and service times, but also in the number of servers, the size of the waiting line (infinite or finite), the service discipline and so forth. Some common service disciplines are:

- **FIFO:** (First in, First out): a customer that finds the service center busy goes to the end of the queue.
- **LIFO**: (Last in, First out): a customer that finds the service center busy proceeds immediately to the head of the queue. She will be served next, given that no further customers arrive.
- **Random Service**: the customers in the queue are served in random order.
- **Round Robin**: every customer gets a time slice. If her service is not completed, she will re$-$enter the queue.
- **Priority Disciplines**: every customer has a (static or dynamic) priority, the server selects always the customers with the highest priority. This scheme can use preemption or not.

The Kendall Notation is used for a shorthand characterization of range of these queueing models [15]. It is a three$-$part code $a/b/c$, where a specifies the interarrival time distribution, b the service time distribution, and c denote the number of servers. This model can be extended to include other parameters as the maximum size of the waiting line and the queue discipline. The notation for the supermarket model used in this work is $(M/M/C) : (FCFS/\infty/\infty)$, this queuing model assumes that all interarrival times are independently and identically distributed according to an exponential distribution, that all service times

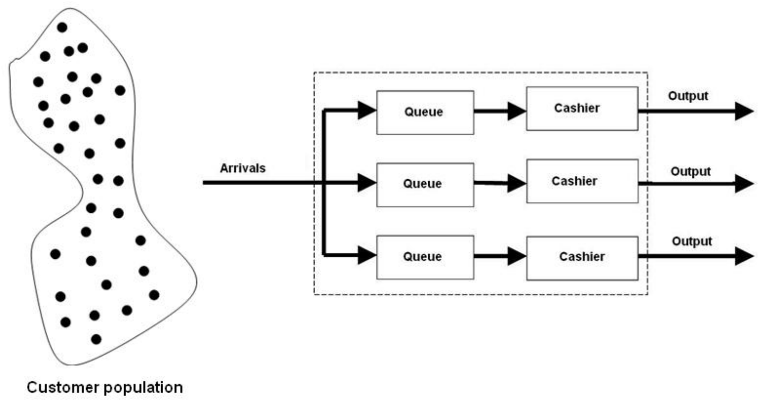


Figure 2. Supermarket service model

are independent and identically distributed according to another exponential distribution, that the number of servers is C, that all customers are first come first served, that there is no limit of customers on the entire system, and that the size of the calling source is infinite. In the succesive we only refer this model as $(M/M/C)$.

Some performance measures. Relevant performance measures in the analysis of queueing models are:

Server utilization: is the percent of time server is busy

$$\rho = \frac{\lambda}{c\mu} < 1 \tag{1}$$

Where
λ : means the average arrival ratio of customers in the queue,
μ : the average service ratio. and
c : the number of cashiers.

The probability that the system (queue and server) is empty:

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{1}{n!} \left(\frac{\lambda}{\mu}\right)^n + \frac{1}{c!} \left(\frac{\lambda}{\mu}\right)^c \frac{cu}{c\mu - \lambda} \right]^{-1} \tag{2}$$

The average lenght of queue is represented as

$$L_q = \left[\frac{(c\rho)^c}{c!\,(1-\rho)^2} \right] \rho P_0 \tag{3}$$

The average waiting time for customers in queue is represented as

$$W_q = \frac{L_q}{\lambda} \tag{4}$$

In particular, in this work we are interested in how obtain the total cost derived by the time that customers spent in the queue, and the cost derived by service time of cashiers. The waiting time cost is not easy to obtain, but there are works related with its study [8, 13] that give some ideas.

3 Simulation of supermarket service

Computer simulation is a method to obtain information about the behaviour of a system in order to provide information for decision. It is an effective way of solve complicated practical problems. The queuing system is the most typical problem in discrete event systems. Computer simulation is a quite effective way of solve queuing problems, analyzing the performance of the queuing system. In this work we realize the design and control of a supermarket service facility in order to decide when to open or close a server under a minimum$-$cost approach. Using information about the estimation of waiting cost, the decision maker can to determine the optimal number of servers for each planning period by minimizing the total cost (service cost $+$ waiting cost). In this section we are approaching only the process of design of a supermarket service model. In the proposed model the simulation of multiple servers is realized varying the value of λ according to the number of servers attending, and maintaining constant the service ratio (μ), this is a valid assumption when there are not many interchanges of customers between the rows in the queue. So we are using multiple M/M/1 model.

3.1 Theoretical Model for Supermarket Service

In supermarket service, the basic model includes the input process, the service time and the queuing rule. The supermarket service system is an M/M/C model. In the simulation of the queuing system, there are some conceptions usually being used as below.

Customers Arrival Mode. The customer size is limitless, and the arrival is individual, arrival mode is often described with arrival interval. The random arrival mode applied in this model is represented by the Poisson distribution [17]. Poisson distribution arrival is defined as:

$$P\left\{N(t) = k\right\} = \frac{\lambda e^{-\lambda}}{K!} \tag{5}$$

Where: $N(t)$: is the number of entity arrival in $(0, t)$, $t \geq 0$, $k = 1, 2, ...$

If the entity arrival satisfies steady Poisson distribution the arrival interval will obey Index distribution and density function is:

$$f(t) = \lambda e^{-\lambda t} = \frac{1}{\beta} e^{\frac{t}{\beta}} \tag{6}$$

Where $t \geq 0$, $\beta = \frac{1}{\lambda}$ is the average value of arrival interval.

Service Mode. The random service time is described with a exponential distribution defined as:

$$f(t) = \mu e^{-\mu T} \tag{7}$$

Performance measurements. The performance measurements used in this work correspond to the M/M/1 model and are the following:

Server utilization

$$\rho = \frac{\lambda}{\mu} < 1 \tag{8}$$

Average waitig time that customers spent in the queue

$$W_q = \frac{\lambda}{\mu(\mu - \lambda} \tag{9}$$

Average lenght of the queue is defined as

$$W_q = \frac{\lambda^2}{\mu(\mu - \lambda)} = \lambda W_q \tag{10}$$

3.2 Details of the simulator

For some aspects of the simulator, we performed a modification and extension to the simulation computer program for MM1 model by Law and Kelton [18]. We replaced the random number generator by another appropriate to our end[1], also we added an interface to allow the interaction with users after we included multiple threads to simulate multiple MM1 queues.

Discrete-event simulation: The discrete−event simulation of a system is a representation where the state variables change instantaneously at separated points in time. More precisely, state can change at only a countable number of points in time, these points in time are when an event occurs. An event is defined as instantaneous occurrence that may change the state of the system. In the supermarket M/M/1 model we define the following:

State variables.

- Status of server (idle, busy), needed to decide what to do with an arrival.
- Current length of the queue, to know where to store an arrival that must wait in line.
- Time of arrival of each customer to the queue, needed to compute time in queue when service starts.

[1]We are using the Mersenne Twister [19] that generates periods long enough for many practical applications.

Events

- Arrival of a new customer.
- Service completion (and departure) of a customer.

Time−advanced mechanisms. Simulation clock: Variable that keeps the current value of (simulated) time in the model. Time advanced is next−event based, using the following:

- Initialize simulation, clock set to 0.
- Determine times of occurrence of future events − event list.
- Clock advances to next (most imminent) event, which is executed, this execution may involve updating the event list.
- Continue until stopping rule is satisfied (must be explicitly stated).
- Clock jumps from one event time to the next, and does not exist for times between successive events −periods of inactivity are ignored.

Components and Organization of a Discrete-Event Simulation Model. Each simulation model must be customized to target system but there are several common components (see Figures 3 and 4).

- System state, variables to describe state.
- Simulation clock, current value of simulated time.
- Event list, times of future events (as needed).
- Statistical counters to accumulate quantities for output.
- Initialization routine to initialize model at time 0.
- Timing routine to determine next event time, type; advance clock.
- Library routines to generate random variates, etc.
- Report generator to summarize, report results at the end.
- Main program that ties routines together, executes them in the right order.

Problem Statement. The following statements are considered to the M/M/1 queue model:

- Assume interarrival times are independent and identically distributed random variables.
- Assume service times are identically distributed random variables, and independent of interarrival times.
- Queue discipline is FIFO.

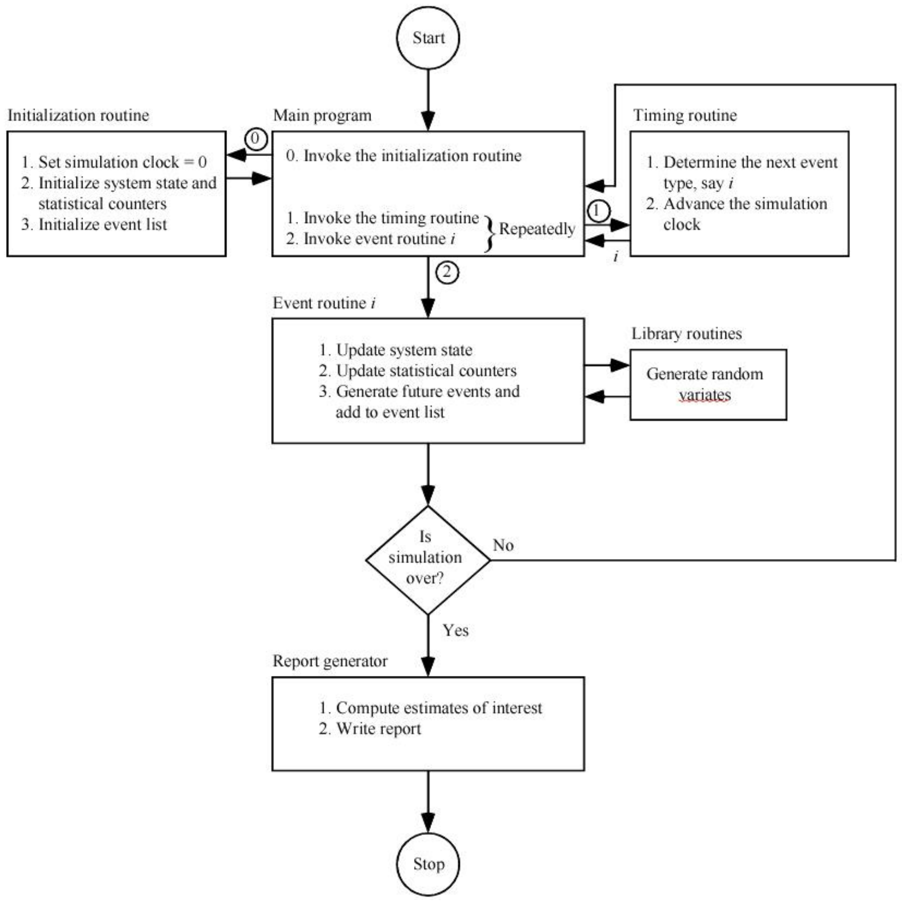

Start
Initialization routine
1. Set simulation clock = 0
2. Initialize system state and statistical counters
3. Initialize event list
Main program
0. Invoke the initialization routine
1. Invoke the timing routine
2. Invoke event routine i
Repeatedly
Timing routine
1. Determine the next event type, say i
2. Advance the simulation clock
0
1
2
i
Event routine i
1. Update system state
2. Update statistical counters
3. Generate future events and add to event list
Library routines
Generate random variates
Is simulation over?
No
Yes
Report generator
1. Compute estimates of interest
2. Write report
Stop

Figure 3. Components and Organization of a Discrete-Event Simulation Model. The simulator is structured by the following main blocks: (a) Initialization routine, (b) Main program, (c) Timing routine, (c) Event routine, (d) Library routines and (e) The report generator [18].

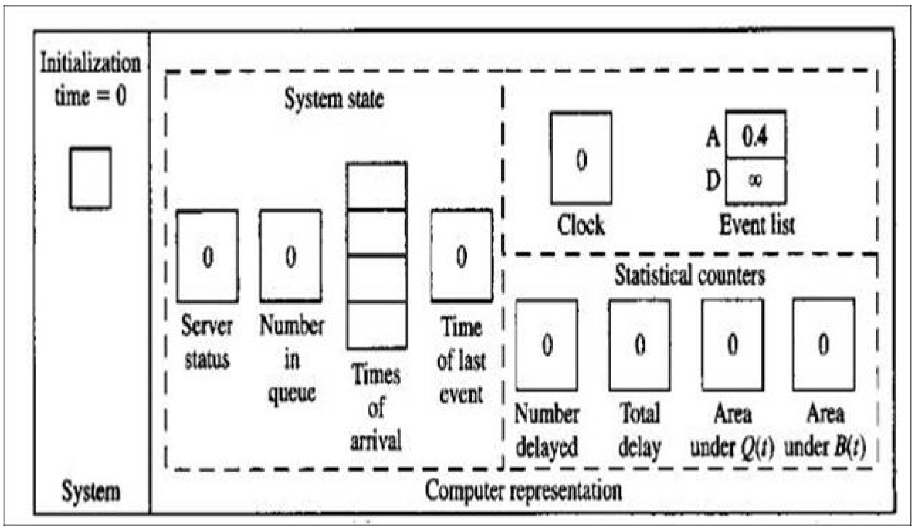


Figure 4. Internal components of the Discrete-Event Simulation Model [18].

- Start empty and idle at time 0.
- First customer arrives after an interarrival time, not at time 0.
- Stopping rule: When the last customer has completed delay in queue (i.e., enters service) we are assigning a value of 1000.

Quantities to be estimated and showed in our interface (see Figure 5).

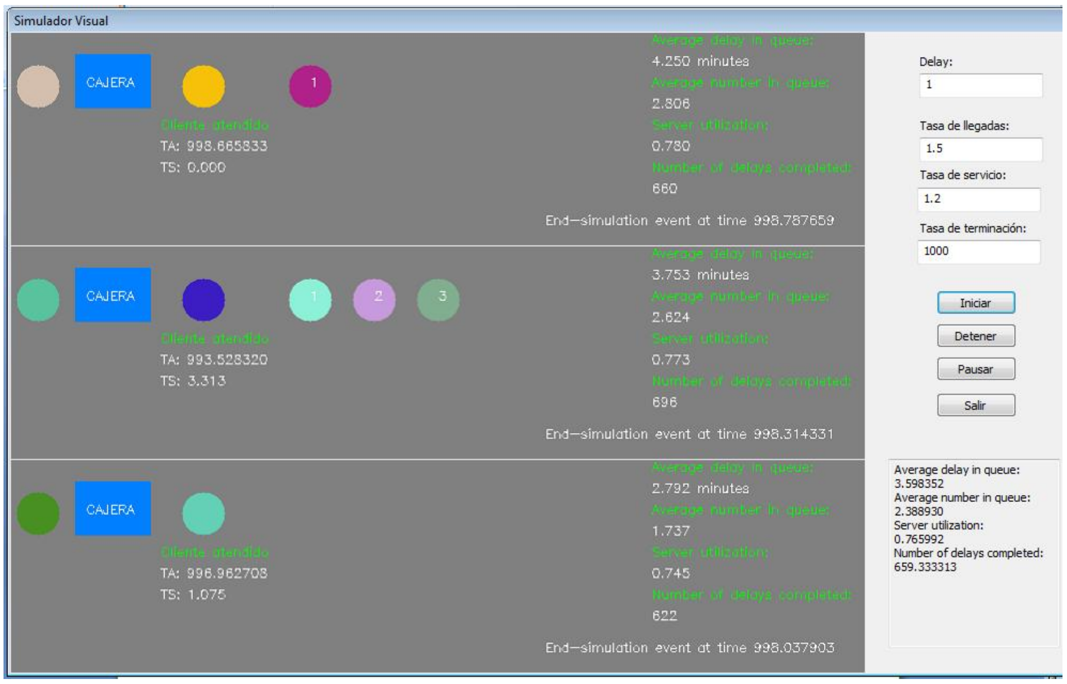
- Expected average delay in queue (excluding service time).
- Expected average number of customers in the queue.
- Server utilization.
- Number of delays completed.

Generation of the random number. In order to generate an observation from an exponential distribution, we assume a perfect random$-$number generator that generates independent and identically distributed variates from a continuos uniform distribution on the interval $[0, 1]$ denoted the $U(0, 1)$ distribution, obeying the next two steps: (1) generate a random number U, and (2) return $X = -\beta lnU$, this is similar to the approach of Lian et al to obtain the random number generator [20].

3.3 Running the simulator

We tested the simulator varying the rate of arrivals (λ) and rate of service (μ), then we analysed the behaviour of the supermarket model, using the obtained information from the relevant variables (L_q, W_q, ρ, c).

Experiment 1. Behaviour of the system when the rate of arrivals and the rate of service are varying. In Figures 6 and 7, we show the results of the simulation model when the rate of arrivals (λ) and the rate of



```
Cliente atendido
TA: 998.665833
TS: 0.000

Average delay in queue:
4.250 minutes
Average number in queue:
2.806
Server utilization:
0.780
Number of delays completed:
660
End—simulation event at time 998.787659

Cliente atendido
TA: 993.528320
TS: 3.313

Average delay in queue:
3.753 minutes
Average number in queue:
2.624
Server utilization:
0.773
Number of delays completed:
696
End—simulation event at time 998.314331

Cliente atendido
TA: 996.962708
TS: 1.075

Average delay in queue:
2.792 minutes
Average number in queue:
1.737
Server utilization:
0.745
Number of delays completed:
622
End—simulation event at time 998.037903
```

```
Delay:
1
Tasa de llegadas:
1.5
Tasa de servicio:
1.2
Tasa de terminación:
1000
```

```
Average delay in queue:
3.598352
Average number in queue:
2.388930
Server utilization:
0.765992
Number of delays completed:
659.333313
```

Figure 5. One view of the interface realized for the simulation of the queue supermarket model, here showing three cashiers attending customers (indicated by color balls) and the values of relevant variables. Input data -top right side- are: interval time between customers arrivals and service time. When simulation is finished the output data - left and bottom side-, are: the performance of the system (ρ), average waiting time (W_q) and expected number of customers in queue (L_q)

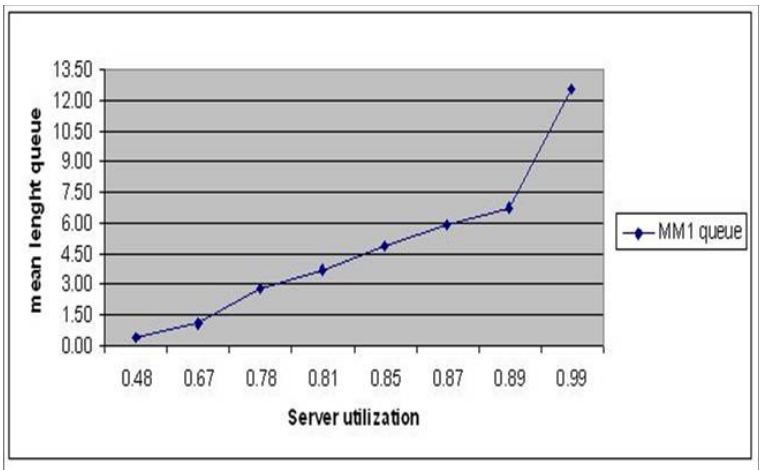


Figure 6. Behaviour of the L_q and ρ parameters during the variation of λ and μ.

service (μ) are varying. Following we explain the results observed. For values of interarrival times greather than values of service times, the behaviour of the system is stable (few queue lenght and few waiting time for customers), it is observed in the Figures 6 and 7, in the values for ρ ranking from 0.48 to 0.81, where the values corresponding for the mean queue lenght (L_q) ranking from 0.45 customers to 6.7 customers, while the average waiting time during the same interval goes ranking from 1.33 minutes to 6.7 minutes. However, when the values of inter$-$arrival times and service times are closing, the expected queue lenght and waiting time are rising, this incremental for values are observed for server utilization (ρ) values ranking from 0.85 to 0.99, during this interval we observed that the values for the average queue lenght and the average waiting time are increasing, the growing for the queue lenght reach values from 4.87 customers to 12.6 customers, at time, the average waiting time is reaching values from 9.8 minutes to 19.7, which is a excesive waiting time. Some researchers are strongly recomended values for optimal server utilization factor, and fixed appropriate values in the interval 0.60 to 0.80, being 0.75 the optimal value [3, 2], this means that the cashiers are busy the 75% of the full service time. Then, taking in mind this recomendation, our model simulator suggest that, under this operation conditions 3 customers is an appropriate number for the lenght queue, and 3 minutes an appropriate waiting time for customers, these values were obtained for inter$-$arrival times of 1 minute and service time of 0.8 minutes.

Experiment 2. Behaviour observed when the inter arrival time is maintained constant and the service rate is varying. Figure 8 indicates the behaviour observed in the simulated queue model when the inter arrival rate is constant and the service rate is varying. This condition could be observed when the customer service is realized by different cashiers or could be reveal the animical state of the same cashier. The inter arrival time rate in this test is fixed in $\lambda_1 = 1$, this means that in this supermarket model a customer arrives each minute. The values assigned for the service rate for this test are $\mu = \{2, 1.75, 1.5, 1.33, 1.16, 1.08, 1.03\}$, these values indicates a service rate of: 120 customers served per hour, 105 customers served per hour, 90 customers served per hour, 80 customers served per hour, 70 customers attended per hour, and 65 customers attended per hour. With the analysis of the results given by the simulator, the conclusions are: (i) when the service rate is twice ($\mu = 2$) the arrival rate, the server utilization is 50% ($\rho = 0.50$) , means that in one hour the cashier is idle the half of the time, (ii) when the service rate aproximates the arrival rate, the server utilization is increasing ($\rho = (0.57, 0.64, 0.75, 0.85, 0.92)$, then the cashier is occupied more and more time,

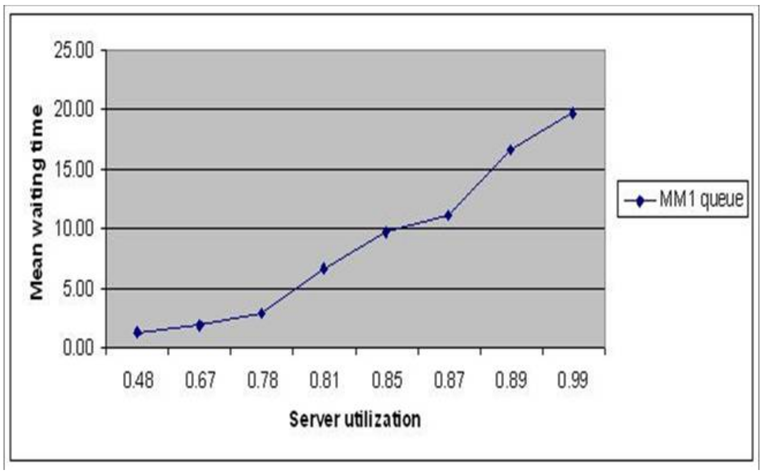


Figure 7. Behaviour of the W_q and ρ parameters during the variation of λ and μ.

if the service rate continues with this behaviour, the system collapse. The mean queue lenght is bigger than twelve customers when the server utilization is 92% $(\rho = 0.92)$ and the waiting$-$time in this case is bigger than 12 minutes per customer. This indicates that must exists a balance between the service rate and inter arrival times in order to permitt the system operates in steady condition, when this goal is reached, the result is an appropriate environment between customers and service institutions.

Experiment 3. Behaviour observed when the rate of arrivals superates the service rate. Figures 9 and 10 illustrating the behaviour observed in the simulated queue model when the arrival rate superates the service rate. This is a common situation when the managers not taking prevission for rush hours in some sales places. To simulate this condition, we divide the arrival rate stream with the number of active cashiers (λ/c), this is a valid assumption when not exists many interchanges between customers in the queues [3]. Then, the operation of the simulator is very similar with $M/M/1$ mathematical queue model. For this test we consider the following: (i) the arrival rate stream is 60 customers per hour, and the service rate of 30 customers per hour, this means, that arrival rate duplicates the service rate. Figure 9 shows a value[2] bigger than 18 for the waiting time and one cashier attending, so the waiting time value is excessive, with the same condition the mean lenght queue is bigger than 40 customers $--$excessive too$--$. When the cashiers attending increments in one -now are two cashiers- the average waiting time is reduced to 13.7 minutes (big still), but, if the number of cashiers increments in one again -now are three-, the waiting time diminishes drastically its value to 1.73 minutes $--$a very reasonable waiting time$--$. In Figure 10, we observe the variation between the server utilization and the cashiers attending (service rate), when there is only one cashier attending, the system crashes, if one cashier is adding the server utilization is 95 percent $--$ the queue may soon collapse under this condition $--$, if a third cashier is added, the server utilization is 64 percent, and the system is under stability conditions. As we said before, must exists a compromise between the cost generated by the service and the cost caused by the waiting time for an appropriate purshasing environment for people in a supermarket facility.

Experiment 4. Comparation between the simulated model and the analytical model. Figure 11 shows

[2]In fact, the value in this case is bigger than 40, but we assign this value for clarify and visualization purposes.

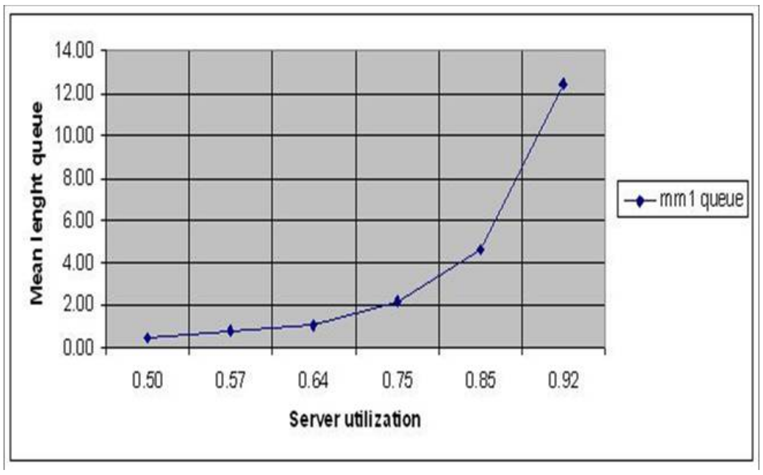


Figure 8. Behaviour of the L_q and ρ parameters when the service rate μ is varying and the arrival rate λ is maintained constant.

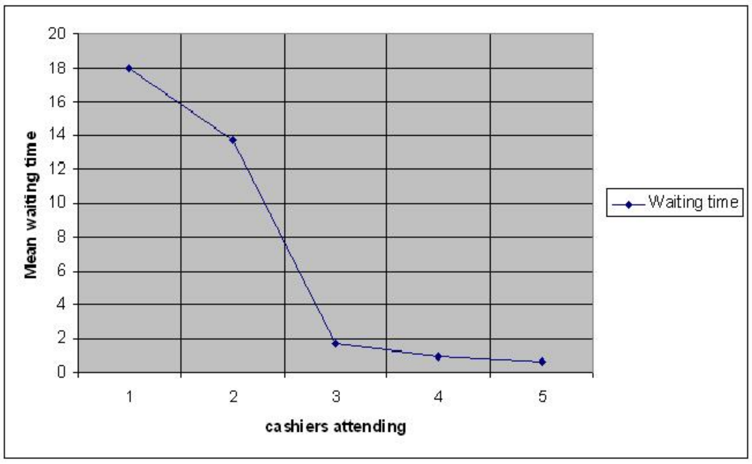


Figure 9. Behaviour of the W_q and C when arrival stream is bigger than service rate.

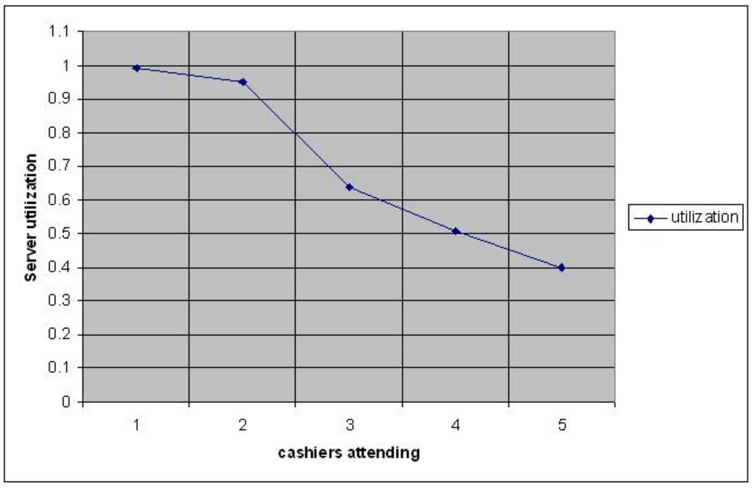


Figure 10. Behaviour of the server utilization and the number of active cashiers C when arrival stream is bigger than service rate.

the results obtained between the analytical mathematical $M/M/1$ model and our simulated $M/M/1$ model. The parameters used in this test are the following: (a) the value for the arrival rate is 60 customers per minute and is maintained fixed, (b) the values for the service rate are varying with the following values $\mu = \{120, 105, 90, 80, 70, 65\}$ customers per hour. The values obtained by our simulated model are very close to computed by the analytical model. Results obtained for both models exhibited a variation ranked from 4% to 14%, as we can appreciate in the Figure 11.

We conclude according with experimental results, that our simulation model designed for the supermarket service queue is reliable.

3.4 Optimal number of cashiers

Determining the optimal number of cashiers is one of the problems that addresses queueing theory. Basically, there are two approaches to solve this problem: (i) Decision cost-based model and (ii) Decision accept-based model [3]. Following we explain each one.

Decision cost-based model. The goal is to balance the total cost, the optimal service level is reached when the sum of both costs is minimal. Using the estimation of waiting cost allows decision maker to have the capability of determining the optimal number of servers for each planning period by minimizing the total cost including the service cost and the waiting cost. However, this estimation is not easy and almost always present difficulties [8], when this occurs, the optimizing can be obtained using another approach, like as the acceptation level model, this consists in balancing conflictives measures, like the average waiting time and idle servers time.

Decision accept-based model. In this case, the goal is to balance conflictives measures, like the average waiting time (W_q) and idle server time (X). Generally the values for this variables are adjusted using a

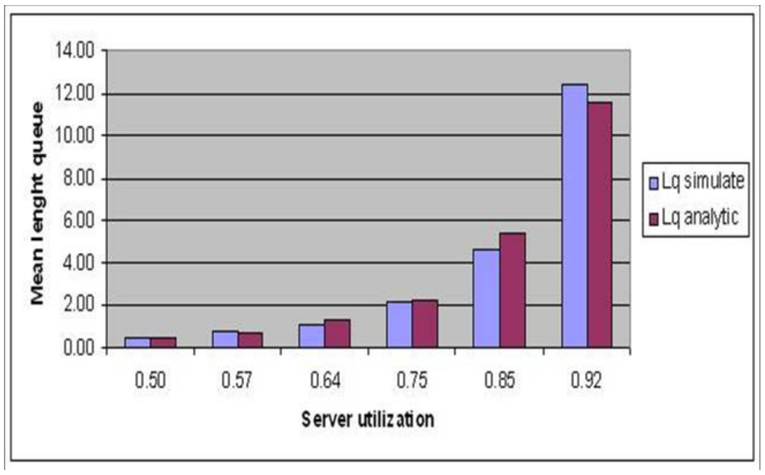


Figure 11. Comparation between analytical results and simulated results for the queue service model.

subjetive approach for each particular application [3]. In this work, the value for (W_q) is determined by the simulator queue model under steady operation conditions $(\lambda < \mu)$. Figure 7 illustrates the behaviour of the supermarket queue model under steady operation conditions, this values were gotten varying the arrival rate and service rate, then the value for (W_q) variable must be less than 7 minutes (value obtained when server utilization value is 81%). To determine the idle time value, some researchers [3, 18] suggest that optimal values for idle time server are from 20% to 30%. In this work, we consider interval values computed by the simulator of the supermarket queue model under steady operation (see Figure 7), then selected values for the X variable are ranking from 19% to 33% (very similar with the suggested values). However, not always is possible matching this values, then the management must to decide the best action. For example, Figure 12 illustrates values for W_q and X variables, when the value of the arrival rate (λ) and service rate (μ) are varying. Taking into account the values observed in this Figure, the optimal number of cashiers are indicated by the shadowed areas. This means, that in the first row the average customer waiting time is 5.43 minutes (less than 7 minutes) and the idle server time is 18.6% (very close to 19%), then the optimal number for cashier is 1, in the second row the values of waiting time and idle server time suggest that 2 cashiers are the optimal number under this operation conditions.

In this work we applied the last approach, in which, the goal, as we said before, is to balance con?icting measurements, like the average waiting time Wq and idle server time X.

Next, we are interested in determining the optimal number of cashiers during a planning horizon (what action should be taken in a particular state of the queue), for getting it, we construct an optimal policy using Factored MDP‘s.

4 Markov Decision Process

A Markov decision process (MDP) is a mathematical framework for sequential decision making problems in stochastic domains. MDPs thus provide underlying semantics for the task of planning under uncertainty. While the notion of an MDP may appear quite simple, it encompasses a wide range of applications and has

	Cashiers attending					
	1	2	3	4	5	parameter values
Wq	5.43	0.88	0.27	0.26	0.19	μ=65, λ=60
X	18.6	54	73	77	83	
Wq	41.17	1.12	0.57	0.38	0.21	μ=65, λ=70
X	2	48	67	73	81	
Wq	43.5	1.9	0.78	0.56	0.38	μ=50, λ=60
X	1	35	63	71	78	
Wq	52.7	3.74	1.36	0.73	0.47	μ=40, λ=60
X	0.73	28	49	76	77	
Wq	59.7	18	1.85	0.97	0.63	μ=60, =120
X	0.17	6	34	47	63	

Figure 12. Illustration of the optimal cashiers attending, the parameter values are determinated by the simulator under steady operation conditions, recommended values in each case are appearing by shadowed boxes.

generated a rich mathematical theory. A concise overview of the MDP framework is the following.

A Markov decision process (MDP) is defined as a 5-tuple (S,A,R, T, γ) where: S is a finite set of $|S| = N$ states; A is a finite set of actions; R is a reward function $R : S \times A \to \Re$, such that $R(s, a)$ represents the reward obtained by the agent in state s after taking action a, T is a Markovian transition model defined as $T : S \times A \to Prob(S)$, where the probability of going from state s to state s' after taking action a is represented as $P(s'|s, a)$; and γ, the planning horizon. The process has a planning horizon, i.e. the time points at which the system has to be controlled. The objective of MDP is to determine a policy, a decision rule for each decision time point and each history (including the present state) of the process, that optimizes the performance of the system. The performance is measured by a utility function. This function assigns to each policy, given the starting state of the process, a value according to optimality criteria, which generally is a metrics of the total expected reward over an horizon (finite or infinite).

While classic solving methods, such as value iteration [21], policy iteration [22], or linear programming [23], scale up well in terms of the total number of states and actions, these techniques present a problem, in purely discrete settings, the running time of these algorithms grow up exponentially when the number of domain features increasing. Then, classical dynamic programming, which requires explicit enumeration of the state space, is typically infeasible for feature-based planning problems. So, factorized representations that allows us to exploiting the problem structure to represent exponentially$-$large MDPs very compactly, like Factored MDPs [1], are an efficient solution. In this Section, we designed and built the second part (II), of the controller for the supermarket system (See Figure 1). The design of such controller is based on a Factored MDP, specifically in a dynamic abstraction method for solving MDPs using algebraic decision diagrams (ADDs) [24] to represent value functions and policies [16].

4.1 Factored Markov Decision Process

Factored Markov decision processes (MDPs) allow us to represent complex uncertain dynamic systems very compactly by exploiting problem−specific structure. Specifically, the state of the system is described by a set of variables that evolve stochastically over time using a compact representation called a dynamic Bayesian network (DBN) [25]. A DBN exploits the structure of the problem assuming that the short−term evolution of a particular variable only depends on a few other variables, e.g., the state of waiting−time for customers in supermarket queue is only directly affected by a few other states −in this case the number of cashiers attending−. Factored MDPs often allow for an exponential reduction in representation complexity. However, the complexity of exact solution algorithms for such MDPs grows exponentially in the number of variables. So, the research of a framework that allows to build and approximate planning algorithms exploiting the structure in Factored MDPs to solve problems with large states and actions efficiently, is the field of current research [26], [16], [27].

4.2 Algebraic Decision Diagrams (ADDs)

Algebraic decision diagrams (ADDs) [24] are a compact and efficiently manipulable data structure for representing boolean functions. These data structures have been used extensively in the VLSI CAD field and have enabled the solution of much larger problems than previously possible. ADDs generalize binary decision diagrams to represent real-valued functions $B^n \rightarrow \Re$; thus, in an ADD, we have multiple terminal nodes labeled with numeric values. More formally, an ADD denotes a function as follows:

1. The function of a terminal node is the constant function $f() = c$ where c is the number labelling the terminal node.

2. The function of a nonterminal node labeled with boolean variable S_1 is given by $f(s_1...s_n) = s_1.f_{then}(s_2...s_n) + \overline{s_1}.f_{else}(s_2...s_n)$ where s_i are viewed as boolean values, and f_{then} and f_{else} are the functions of the ADDs rooted at the *then* and *else* children of the node.

ADDs has several useful properties. First, for a given variable ordering, each distinct function has a unique reduced representation. In addition, many common functions can be represented compactly. Furthermore, efficient algorithms (e.g., depth-first search with a hash table to reuse previously computed results) exist for most common operations, such as addition, multiplication, and maximization. Finally, because ADDs has been used extensively in other domains, very efficient implementations are readily available. As we will see, these properties make ADDs an ideal candidate to represent structured value functions in MDP solution algorithms.

4.3 Representation of MDPs using ADDs

In a Factored MDP, the set of states is described via a set of *random (state) variables* $S = \{S_1, ..., S_n\}$, where each S_i takes on values in some finite domain *Dom*(S_i). A state s defines a value $s_i \subset$ *Dom*(S_i) for each variable S_i. In general, we use upper case letters (e.g., S) to denote random variables, and lower case (e.g., s) to denote their values. We assume each S_i is a multi−valued variables. Actions are often most naturally described as having an effect on speci?c variables under certain conditions, implicitly inducing state transitions. DBN action representations [25], [26] exploit this fact, specifying a local distribution over each variable describing the (probabilistic) impact an action has on that variable. A DBN for action requires two sets of variables, one set $S = \{S_1, ..., S_n\}$ referring to the state of the system before action has been executed, and $S' = \{S'_1, ..., S'_n\}$ denoting the state after a has been executed. Directed arcs from variables in S to variables in S' indicate direct causal influence. The conditional probability table (CPT) for each post$-$action variable S'_i defines a conditional distribution over $P^a_{s'_i}$ over S'_i i.e., actions effect on S'_i for each instantiation of its parents. This can be viewed as a function $P^a_{s'_i}(S_1...S_n)$ but where the function value (distribution) depends only on those S_j that are parents of S'_i . No quanti?cation is provided for pre-action variables S_i since the process is fully observable, we need only use the DBN to predict state transitions. We require one DBN for each action $a \in A$.

5 Decision-theoretic controller

The policy for the supermarket service queue is defined using an MDP framework. Based on the simulated queue model results, the controller selects the most appropriate decision to assist management in decision making. The parameters of the MDP model corresponding with the supermarket queue model are the following:

States. The state space is characterized by three variables with three values each one that models the supermarket queue service, and are the following: **npc** (number of customers in queue), **nca** (number of attending cashiers) and **tep** (average waiting time in the queue). Each state maintains three values (l*ow, normal, high)* indicating the queue state for each variable during different periods of the day. Interval under steady operation conditions for each state previously mentioned are obtained from the simulation model and they are illustrated in Figure 12 and Figure 13 . Next, the interpretation of each one is presented:

nca (number of attending cashiers). The valid values[3] for this state variable under steady operation conditions are: *low* $[0.84 - 0.91]$, *normal* $[0.75 - 0.83]$ and, *high* $[0.68 - 0.74]$. The mentioned values for this state variable are normalized ranked from 0 to 1. This values were selected considering the operation of the simulator under stable condition and the goals of the designed system: optimizing the supermarket service, this means: (i) optimizing the cashiers number and (ii) optimizing the waiting time (see Figure 12). The optimal values for *nca* were previoulsy obtained, the l*ow* interval were obtained from the ρ variable, then for the interval $[0.84 - 0.91]$, means that the server utilization is ranking from 84% to 91%, this indicates the rate of busy of the cashiers by unit time. When the controller is planning during an horizon and observes this values, it suggests that the action must be $"continue"$. The policy must suggest another action when this values are different, by example, in the *low* interval if the value is bigger than 0.91, this indicates that the arrival rate is bigger than the service rate (See Figure 10), then the selected action is "*Add a cashier*".

tep (average waiting time). Like the state variable previously described, the valid values[4] for this state variable under steady operation conditions are: *low* $[1.8 - 3.8]$, *normal* $[3.9 - 5.9]$ and, *high* $[6.0 - 8.0]$. When the controller is planning during an horizon and observes those values -computed by the simulator-, the best decision must be $"continue"$. The policy indicates another action when this values are different, by example, in the *high* interval if the value observed is bigger than 8.0, this indicates that the arrival rate is bigger than the service rate (See Figure 9), then the selected action is "*Add a cashier*".

[3]The unit considered for this variable is a real number indicating the number of cashiers attending.

[4]The units for this variable are minutes.

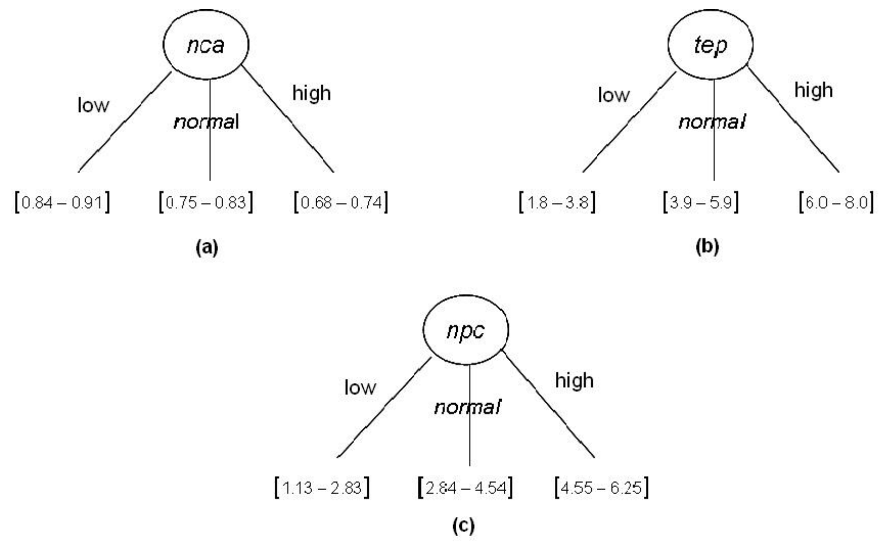


Figure 13. Set of states and interval of values computed by the simulator when the operation conditions are steady. In this case $\lambda = 40$ customers arriving per hour and $\mu = 50$ customers served per hour.

npc (number of customers in the queue). Like both previously described state variables , the valid values[5] for this state variable and under steady operation conditions are: *low* $[1.13 - 2.83]$, *normal* $[2.84 - 4.54]$ and, *high* $[4.55 - 6.25]$. When the controller observes those values, according to the policy the best action selected must be "$continue$". The policy selects another action when this values are different, by example, in the *high* interval if the value observed is bigger than 6.25, this indicates that the arrival rate is bigger than the service rate (See Figure 8), then the selected action is "*Add a cashier*".

Actions. In the same way that Spudd and other works [16], [25], [26], actions are often most naturally described as having an effect on specific variables under certain conditions, implicitly inducing state transitions. DBN action representations exploit this fact, specifying a local distribution over each variable describing the (probabilistic) impact an action has on that variable. A DBN for action requires two sets of variables, one set $S = \{S_1, ..., S_n\}$ referring to the state of the system before action has been executed, and $S' = \{S'_1, ..., S'_n\}$ denoting the state after a has been executed (See Figure 14). Directed arcs from variables S_i in to variables in S'_i indicate direct causal influence and have the usual semantics, conditional probability table (CPT) for each post$-$action variable defines a conditional distribution over S'_i. At the moment, the actions defined in the MDP model are:

Continue action. We select this action, when the controller is observing that data computed by the simulator indicates an operation of the queue supermarket system is in stable conditions, this means, value *normal* for the state variables. Figure 15 is illustrating the influence of this action in each state variable. The effect in this case is represented by probabilities values in terminal nodes according to the ADD semantics. For

[5]In this case the unit is a real number indicating the number of customers in the queue.

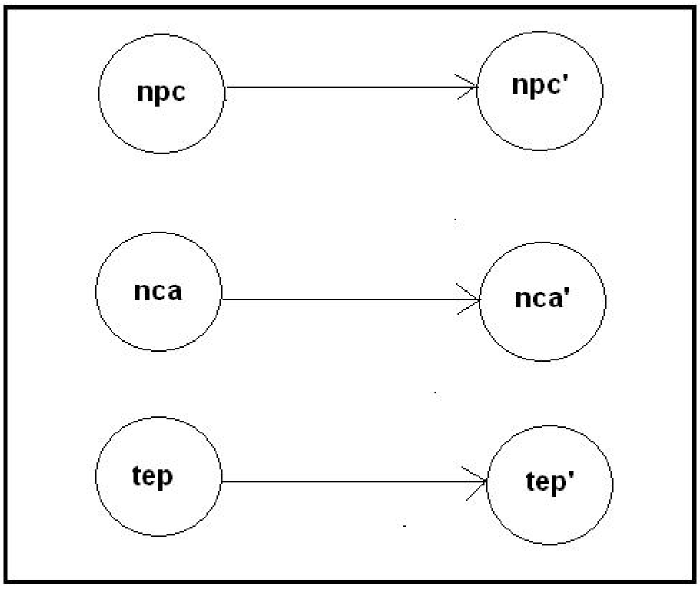


Figure 14. Set of states represented as a DBN for the action $Continue$.

example, when the system is operating under stable condition (*normal* values) the value of probability value $--$from the simulator$--$ for the transition of the state variable tep (average waiting time) from time t to $t+1$ is 0.6 (see Figure 15). This action is like the *Null action* in other works [25], [28],.

Add a cashier action. When the controller is observing $-$during the planning period$-$, that the operation model is under unestable conditions, this means: (i) the average waiting time in queue is *high* $-$is over the *normal* expected value$-$, (ii) the expected number of customers in queue override the *normal* operation conditions, and (iii) the rate of performance is high -over the 90%-, then the controller must to suggest that appropriate action is *Hire a cashier*. Figure 5 illustrates the DBN that represents the action *"Add a cashier"*. We can observe that, when controller selects this action, the state variable *nca* (number of cashiers attending customers), affects the state variables *npc* (expected customers in queue) and *tep* (expected waiting time), and *nca* too, in the next step time. Representation of conditional probabilities tables (CPTs) for this action using ADDs is illustrated in Figure 17, where (a) is the ADD representation for the *npc* state variable, (b) represents the *nca* state variable using the ADD semantic, and finally (c) illustrates the ADD representation for the *tep* state variable. Generally this action is sugested when during the planning horizon the arrival rate is bigger than the service rate, and maintained under that condition during several planning period.

Eliminate a cashier action. Currently, the controller only recognizes three actions and *eliminate a cashier* is the third of them. This action is suggested when the operation condition reported by the queue supermarket simulator is below the *normal* operation or steady condition. For example, if the *low* value for *npc* variable is lower than 1.13, or the computed value for the *tep* variable is lower than 1.8, or the value for the *nca* variable is lower than 0.68, then the controller policy indicates that the system is operating with many cashiers, so, the appropriate decision under this operation condition can be *eliminate a cashier*. The goal of

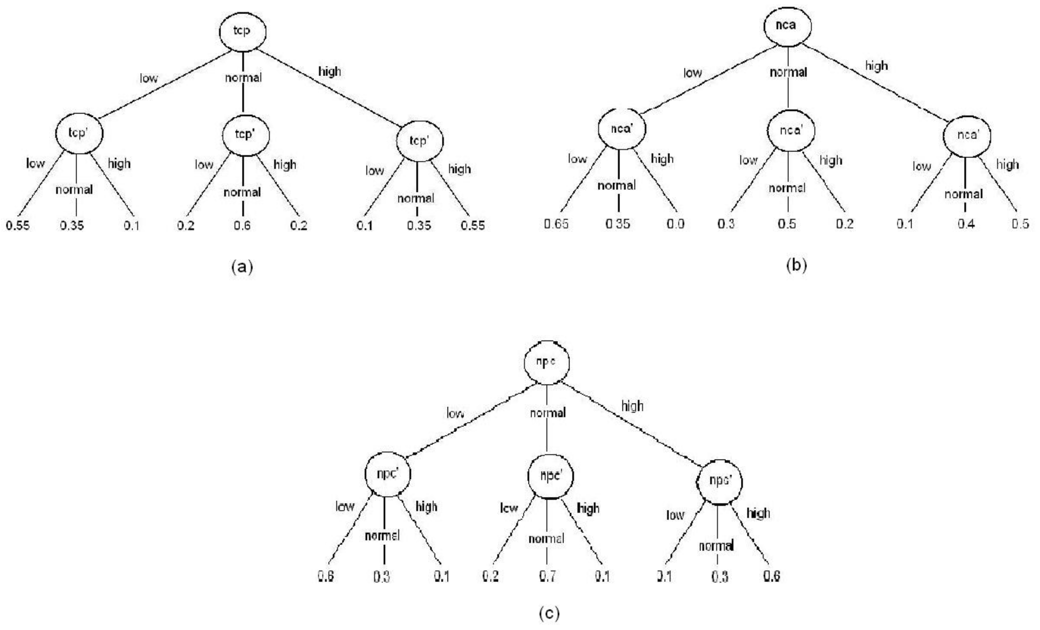


Figure 15. ADD representation of the transition function for the action $Continue$**, the probability values indicated in the leaves of the tree were obtained from the simulator when the system was operating under stable condition (**$normal$ **value). For example, image (a) illustrates the** $average\ waiting\ time$ **with the values before (tep) and after (tep') action** $Continue$ **is selected. In this case, the value of probability for the transition of this state variable from time** t **to** $t+1$ **is** 0.6**, the probability values for the transition of the** low **and** $high$ **values** $--$**of the same state variable**$--$ **are** 0.55 **respectively, (b) this figure illustrates the probability values** $--$**obtained from the simulator**$--$ **fot the state that represents the number of** $active\ cashiers$ **(nca) attending customers during the same condition (**$normal$ **value), and (c) is showing the values of probabilities for the** $expected\ customers\ in\ queue$ **under the influence of this action. Basically this action indicates: do nothing.**

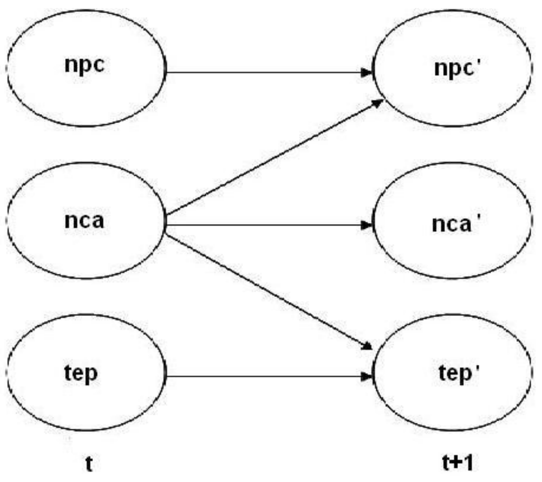


Figure 16. DBN representation for the action *Add a cashier*.

.

the controller is that the supermarket queue system, maintains its operation under normal condition, then, must exists a balance between the service rate −indicated by the number of cashiers attending− and the waiting time −indicated by the expected customers in the queue−.

Rewards. Each state and action has a reward associated with it. Since there are several choices of actions at each state, the goal is to determine which action yields the highest expected reward. Some state-action pairs may yield an immediate reward of zero (or may yield negative rewards), but may transition the system into a state where a large positive reward is possible. We therefore want the system to be able to look an arbitrary number of steps into the future when determining how to maximize its reward, but we wish the system to value immediate rewards (rewards that are fewer steps away) more than long-term rewards. To bias the system towards closer rewards, we discount future rewards by a factor of γ for each step it takes to reach them (recall that $\gamma < 1$), in this work, we use the deffault value of $\gamma = 0.9$ used by Spudd ([16]). In order to determine which actions produce the highest expected discounted reward, we compute the value function of the MDP, using ADD aproach described below. This problem formulation is known as a discounted infinite-horizon optimality criteria. For a through survey of optimality criteria, we refer the reader to Littman ([29]). To obtain optimal policy, our system must make a balance between an adequate selection of actions in each state observed after a planning horizon. In this work, the reward function represents a commitment between the goal of the supermarket management (optimal cashier number with minimum cost) and the goal of the customers (minimum waiting time). This goals are reached based on: (i) optimal number of cashiers using an accept-cost model approach (see Figure 12) and, (ii) optimal values for the queue parameters (*nca, npc, tep*) based on the steady operation conditions of the simulated supermarmarket queue model (see Figure 10). Figure 18, tries to explain the way of approach this aspects in this work. *Normal* condition of operation is the joint of the two goals. Figure 19, illustrates the ADD representation of the reward function utilized in this work. The preferred operation conditions for the supermarket queue simulator model are for the *normal* value and *low* value in each state variable.

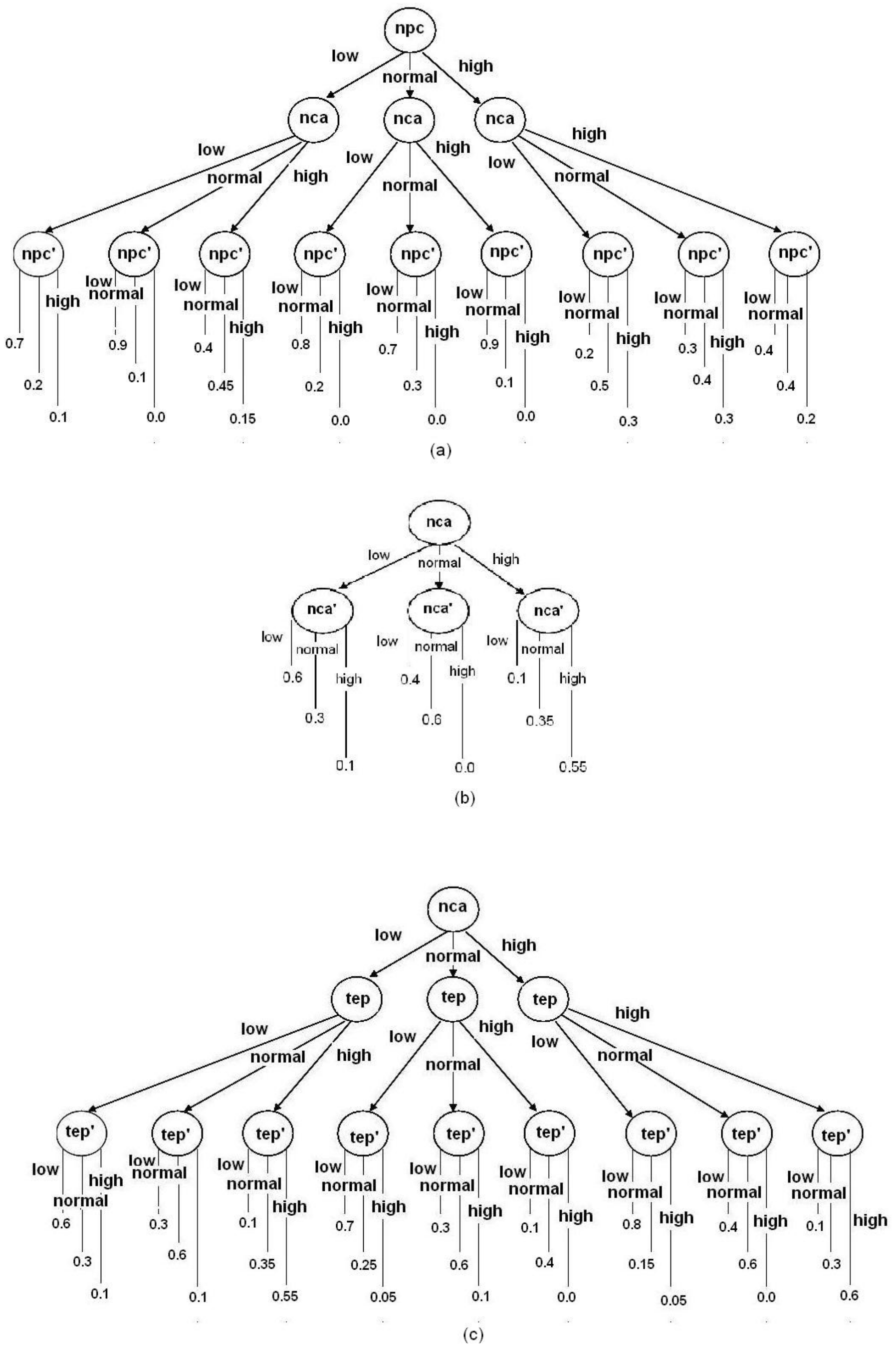
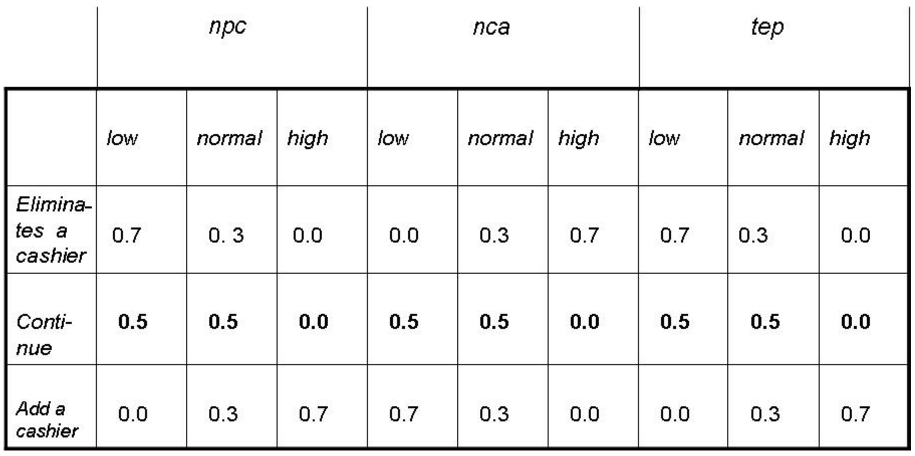


Figure 17. ADD representation of the transition function for the action *Add a cashier*. **This action is selected when the controller is observing −during the planning period−, that the operation model is under unestable conditions, this means: (i) the average waiting time in queue is** *high* **−is over the** *normal* **expected value−, (ii) the expected number of customers in queue override the** *normal* **operation conditions, and (iii) the rate of performance is high (bigger than** 90%**). Image (a) illustrates the probabilities values −−obtained from the simulator−− for the state variable** *npc* **(number of customer in the queue), when the action** *Add a cashier* **is selected, (b) this figure illustrates the same condition for the variable** *nca* **(number of cashiers attending) and (c) is showing the values of probabilities for the variable** *tep* **(average waiting time) during the influence of this action.**



	npc			*nca*			*tep*		
	low	*normal*	*high*	*low*	*normal*	*high*	*low*	*normal*	*high*
Eliminates a cashier	0.7	0. 3	0.0	0.0	0.3	0.7	0.7	0.3	0.0
Continue	**0.5**	**0.5**	**0.0**	**0.5**	**0.5**	**0.0**	**0.5**	**0.5**	**0.0**
Add a cashier	0.0	0.3	0.7	0.7	0.3	0.0	0.0	0.3	0.7

Figure 18. Adequate balance between optimal number of cashiers and minimum waiting time. Customers preferences are minimum waiting time: then, many cashiers attending. Supermarket preferences are: minimum waiting time, but, low cashiers attending. Then, to equilibrate this preferences, and to maintain the operation system under steady operation conditions, prefered action is $Continue$ and prefered value for the sate variables is $Normal$. Prefered actions are represented by a value indicating an associate cost value in the interval $0 - 1$, where 0 value indicates minimum preference.

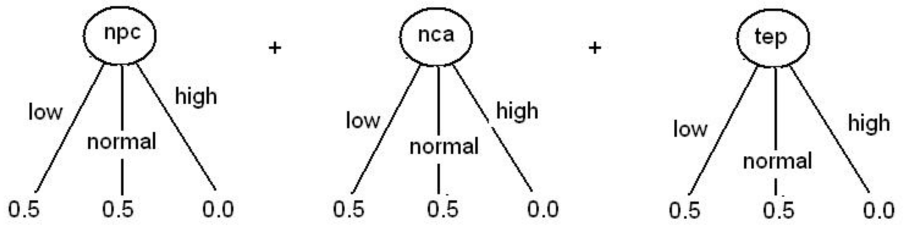


Figure 19. ADD representation of the reward function for the supermarket queue model controller.

5.1 Solving the MDP with SPUDD

To solve our MDP and find optimal policy construction that avoids the explicit enumeration of the state space, we have used the SPUDD (stochastic planning using decision diagrams) system . SPUDD implements classical value iteration, but uses ADDs to represent value functions and CPTs. It exploits the regularities in the action and reward networks, made explicit by the ADD representation described in the previous section, to discover regularities in the value functions it constructs. This often yields substantial savings in both space and computational time. Here we showed a concise description of this algorithm, for a deep knowledge refered to work realized by Hoey et al [16]. We assume that the domain of interest can be modeled as a fully-observable MDP with a finite set of states S and actions A . A stationary policy $\pi : S \to A$ describes a particular course of action to be adopted by an agent, with $\pi(s)$ denoting the action to be taken in state . Initially, we assume that the agent acts indefinitely (an infinite horizon), after we consider planning with finite horizon. We compare different policies by adopting an expected total discounted reward as our optimality criterion wherein future rewards are discounted at a rate $0 \leq \beta \leq 1$, and the value of a policy is given by the expected total discounted reward. The expected value $V_\pi(s)$ of a policy at a given state s satisfies [23]:

$$V_\pi(s) = R(s) + \beta \sum_{t \in S} Pr(S, \pi(s), t).V_\pi(t) \tag{11}$$

A Policy π is optimal if $V_\pi \geq V'_\pi$ for all $s \in S$ and policies π'. The optimal value function V^* is the value of any optimal policy. Value iteration is a simple iterative approximation algorithm for constructing optimal policies. It proceeds by constructing a series of N-stage-to-go value functions V^n . Setting $V^0 = R$, we define

$$V^{n+1}(s) = R(s) + max_{a \in A} \left\{ \beta \sum_{t \in S} Pr(S, \pi(s), t).V^n(t) \right\} \tag{12}$$

The sequence of value functions V^n produced by value iteration converges linearly to the optimal value function V^*. For some finite n, the actions that maximize Equation 12 form an optimal policy, and V^n approximates its value. A commonly used stopping criterion specifies termination of the iteration procedure when

$$\left\| V^{n+1} - V^n \right\| < \frac{\epsilon(1-\beta)}{2\beta}$$

This ensures that the resulting value function V^{n+1} is within $\epsilon/2$ of the optimal function V^*. at any state, and that the resulting policy is $\epsilon-$optimal [23]. Following the previous approach indicated by Hoey et al [16] to constructing optimal policies, we solved our MDP model previously described. Figure 20, illustrates the policy obtained with Spudd and that our controller is obeying, to suggest the best decision for supermarket management. Evaluation of this policy is illustrated in Figure 21. As we can see in both tests, the operation of the controller is correct, and it suggest the appropriate action according the policy computed by Spudd.

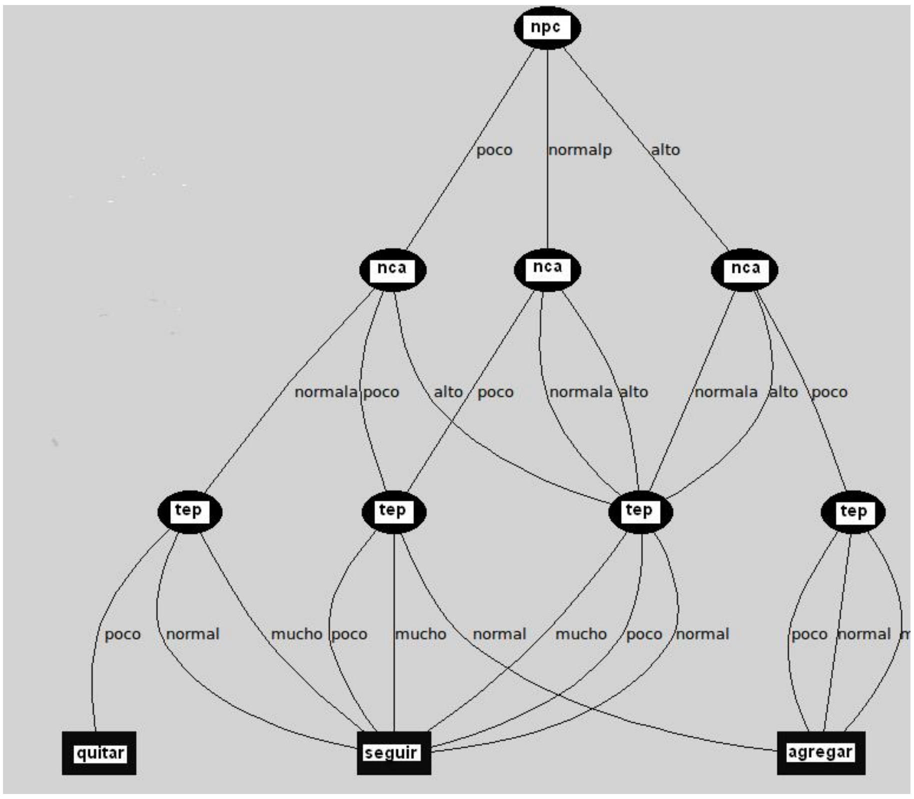
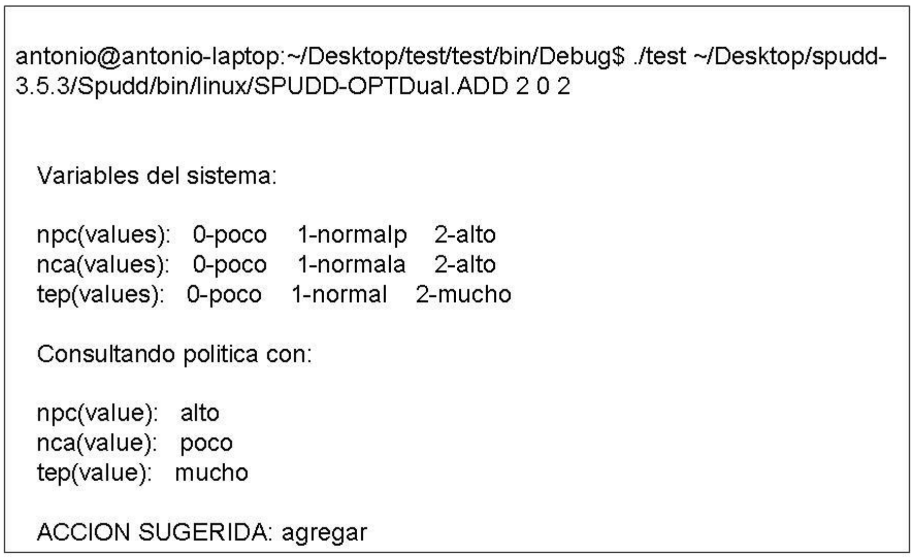
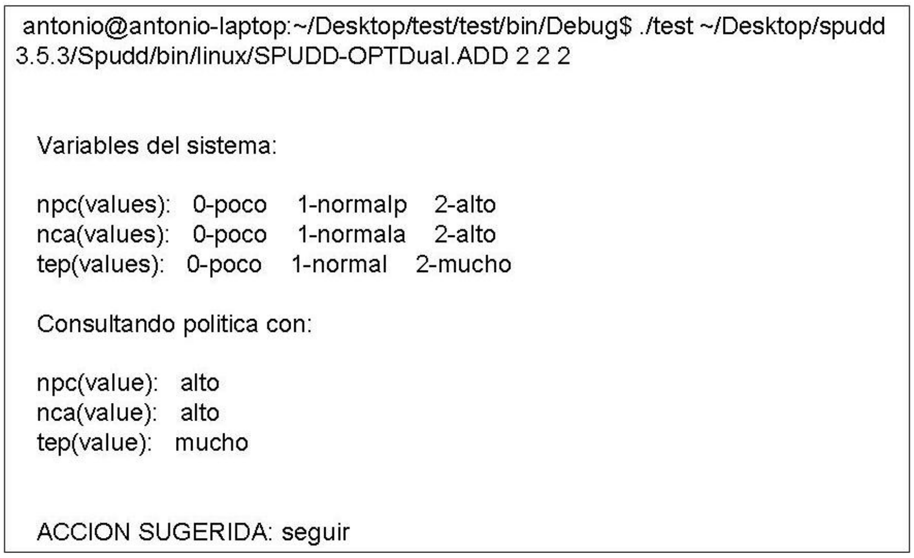


Figure 20. *Optimal* **policy computed by the Spudd and based in classical value iteration algorithm. State variables are appearing in the order required by Spudd and are the following:** *npc* **represents the number of person waiting in queue, as we can observe this variable has three values:** *low* **(poco),** *normal* **(normalp) and** *high* **(alto). After, the variable** *nca* **represents the number of cashiers attending customers, and in the same way that previous variable has three values:** *low* **(poco),** *normal* **(normala) and** *high* **(alto). Next, the image depicts tha variable** *tep* **that represents the waiting time in the queue with three values too. Finally and closed by rectangles are the actions that the system selects, these are the following: eliminate a cashier (***quitar***), continue (***seguir***) and add a cashier (***agregar***). So, this Figure depicts the set of rules that the controller selects, during the operation of the queue system and during a planning horizon. For example, if the controller observes that the value of the** *npc* **variable is** *normal* **(normalp) and the value of the** *nca* **variable is** *normal* **(normala) and the waiting time in queue is** *low* **(poco), then the selected action is continue (***seguir***).**



```
antonio@antonio-laptop:~/Desktop/test/test/bin/Debug$ ./test ~/Desktop/spudd-
3.5.3/Spudd/bin/linux/SPUDD-OPTDual.ADD 2 0 2


  Variables del sistema:

  npc(values):   0-poco    1-normalp    2-alto
  nca(values):   0-poco    1-normala    2-alto
  tep(values):   0-poco    1-normal    2-mucho

  Consultando politica con:

  npc(value):   alto
  nca(value):   poco
  tep(value):   mucho

  ACCION SUGERIDA: agregar
```

(a)



```
 antonio@antonio-laptop:~/Desktop/test/test/bin/Debug$ ./test ~/Desktop/spudd
3.5.3/Spudd/bin/linux/SPUDD-OPTDual.ADD 2 2 2


  Variables del sistema:

  npc(values):   0-poco    1-normalp    2-alto
  nca(values):   0-poco    1-normala    2-alto
  tep(values):   0-poco    1-normal    2-mucho

  Consultando politica con:

  npc(value):   alto
  nca(value):   alto
  tep(value):   mucho


  ACCION SUGERIDA: seguir
```

(b)

Figure 21. Test of the controller: (a) illustrates when the action *add a cashier* **is suggested, this is selected with the combination of** *npc = high*, *nca = low* **and** *tep = high*. **In (b) the accion** *continue* **is selected with the combination** *npc = high*, *nca = high* **and** *tep = high*.

6 Related work

The optimal design of queues as a static problem has been studied formally as an optimization problem by some researchers. The earliest queueing design model may be due to Brigham [5]. He was concerned with the optimum number of clerks to place behind tool crib counters in plants belonging to the Boeing airline company. Using an M/M/c model [15], and using a linear total cost function, he presented curves showing the optimal c as a function of the system performance (ρ) and the ratio of customers waiting cost to clerk idle cost. Morse [4] considered several economic models: an M/M/1 model for ships arriving at a harbor with a single dock, an M/M/1/K with lost customers, an M/M/c/c model with impatient customers, and some machine-repair problems. In all models he seeks the optimal service rate. He solves the first two models using simple calculus and presents for the third model a graph where, for a given λ, μ, and ratio of the service cost to profit per customer, the optimal c can be found. Hillier [6], considers three general classes of models, the first dealing with optimizing c, the second dealing with optimizing both λ and c, and the third dealing with optimizing μ and c. Linear cost functions are built for each model. Many of Morse models turn out to be special cases of one of these classes. Winston [30] considers an exponential queueing system with several removable servers. The operator of the system wishes to choose the number of servers in operation so as to minimize the expected discounted cost incurred over a finite or infinite horizon. Using both discrete-time and continuous-time Markov decision processes, the author derives the conditions under which the number of servers in operation is a non-decreasing function of the number of customers in the system. Grassman et al. [7] address the problem of choosing the optimal rate for a one-server queue with state-dependent arrivals in an M/G/1 system, given that the service rate μ can be adjusted continuously. Probabilities after departure are linked to random time probabilities by use of Bayes' theorem.

As we said before, the optimal control of a queueing system is to determine when and how to change arrival or service rates to optimize some objective function. Most of the times, this corresponds to determine queue levels at which service should start or stop. The optimal control of queues has been the subject of numerous research papers. We will not be concerned with the theory of controllable queueing systems which involve the control of admission, servicing, routing and scheduling jobs in queues and networks. Rather, we will restrict ourselves to research whose objective is to find the optimal operating policy, that is rules for turning the server on and off that result in the lowest long-run cost. A stationary policy is defined as one that always prescribes the same action whenever the system is in a given state. In the beginning, a series of models concerned with varying the service rate of the system according to the number of customers in the queue were treated in the literature. The following papers are examples of such models.

The earliest model of this type may be that of Romani [9]. He considers a policy where, if the queue builds up to a certain critical value, additional servers are added as new arrivals come, thus preventing the queue from ever exceeding the critical value. Servers are then removed when they finish service and no one is waiting in the queue. Crabill [10] considers an M/M/1 queue with k (finite) possible service rates. The cost structure considered comprises costs associated with each service rate and a cost associated with the state of the system. Crabill proves that the optimal policy that minimizes the long-run average cost rate is characterized completely by $k-1$ numbers and that the optimal service rate is nondecreasing in the state of the system. Of particular interest among stationary policies are the threshold (or control limit) policies, which turn the server on only when the queueing process is equal to or larger than a given value, and turns the server off when the queue is empty. Optimality of threshold policies has been shown by many researchers like as Kella [11] and Piunovskiy [12].

Daria et al, integrates constraint programming and queueing theory to solve the problem of switching workers between two rooms depending on demand, They assume stochastic customer arrival and service times, and using a smallest-cost combination of cross-trained and specialized workers, obtain a policy for switching the cross-trained workers between the rooms, which satisfies constraints on the expected customer waiting time and expected number of workers in the back room. Anther work related with the switching of workers was realized by Berman et al [14], who assume that only cross-trained workers can be hired by the facility. The objective of Berman et al. is to determine when to switch workers between two rooms, so that expected customer waiting time is minimized, but the requirement on the minimum number of back room workers is met (this case is known as the problem P1). In another situation (problem P2), the goal is to find the minimum number of cross-trained workers such that a switching policy exists to meet constraints on the expected customer waiting time and on the expected number of back room workers. Berman et al [14] propose two heuristics, also called $P1$ and $P2$, for solving these problems. Liao [8] implements a creative and effective approach to formulate waiting cost including balking loss and reneging loss. Using the estimation of waiting cost allows decision maker to have the capability of determining the optimal number of servers for each planning period by minimizing the total cost including the service cost and the waiting cost.

In this work we approached two problems: the queue design and service control, generally and as we observed in related works, these problems are approached by separated, and only a few works are approached together. Our designed system offered an structured and overall solution based on minimum total cost (cashier cost + waiting cost), we determinated the optimal number of cashiers that should be attending in one supermarket setting and how this cashiers should be selected during a planning horizon (what action should be taken in a particular state of the queue). For reach this, we based our work in two powerful techniques: queueing theory and decision theory. The optimal policy construction is based on Factored MDP's.

7 Conclusions and future work

At the moment we have obtained some interesting findings from our work, the most important are listed below:

- We presented the design of a simulator for the queue supermarket joined with a controller based in Factored Markov decision process (FMDPs), for optimizing the supermarket service. The designed system selects the optimal number of cashiers in a planning period, this decision is based on minimum$-$cost approach. For getting the proposed system we integrated two powerful techniques: queueing theory and decision theory. In particular, queueing theory addressed problems related to the optimal design and control of queues and joined with decision theory, allowed us to determine how to assign cashiers in the supermarket queue model, while some parameters are varying and some measure of system performance is optimum.
- We provided experimental evidence illustrating that the performance of the simulated model was according with the mathematical counterpart, and the actions selected by the controller were according the optimal policy constructed by Spudd.

Future directions.

We now outline some directions for future.

- Partially observable Markov decision process (POMDP). We have assumed that the underlying supermarket queue planning problem is fully observable, but in more general formulations, the controller may be able to make noisy observations about the world, for example, using vision sensors. Then, such planning problem can be formulated as a partially observable Markov decision process (POMDP).
- Reinforcement learning (RL). RL is a model-free approach, that is, it attempts to obtain successful policies without explicitly building a model of the environment. Model-free algorithms do not need to make strong assumptions about the underlying structure of the world. Unfortunately, as no model of the world is maintained, it is often dif?cult to bound the quality of the current solution, or design effective exploration strategies. Model-based approaches, on the other hand, build a parametric model of the world and use this model to explore the environment effectively . Furthermore, if the model parameterization is a good approximation of the underlying world, then model-based methods can be very effective. An interesting future direction is to apply this learning approach assuming that the underlying system can be modelled by a POMDP.

References

[1] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, pages 1–11, 1999.

[2] L. Tadj and G. Choudhury. Optimal design and control of queues. *Sociedad de Estadistica e Investigacion Operativa*, 13:359–412, 2005.

[3] D. Gross and C. Harris. Fundamentals of queueing theory. *John Wiley Sons, Inc.*, 1998.

[4] P. M. Morse. Queues, inventories and maintenance. *John Wiley Sons, Inc.*, 1958.

[5] Brigham. On a congestion problem in an aircraft factory. *Operations Research*, 3:412–428, 1955.

[6] F. S. Hillier. Economic models for industrial waiting line problems. *Management Science*, 10:119–130, 1963.

[7] W. K. Grassman, X. Chen, and Kashyap B.R.K. Optimal service rates for the state-dependent m/g/1 queues in steady state. *Operations Research Letters*, 29:57–63, 2001.

[8] Pen-Yuan Liao. Optimal staffing policy for queuing systems with cyclic demands: waiting cost approach. *POMS 18th Annual Conference Dallas, Texas, U.S.A.*, pages 1–14, 2007.

[9] J. Romani. Un modelo de la teorfa de colas con numero variable de canales. *Trabajos de Estadistica*, 8:175–189, 1957.

[10] T. B. Crabill. Optimal control of a service facility with variable exponential service times and constant arrival rate. management science. *Management Science*, 18:560–566, 1972.

[11] O. Kella. Optimal control of the vacation scheme in an m/g/1 queue. *Operations Research*, 38:724–728, 1990.

[12] A. B. Piunovsky. Bicriteria optimization of a queue with a controlled input stream. *Queueing Systems*, 48:159–184, 2004.

[13] Daria Terekhov, J. Christopher Beck, and Kenneth N. Brown. A constraint programming approach for solving a queueing design and control problem. *Technical Report, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, Canada.*, pages 1–29, 2008.

[14] O. Berman, J. wang, and K. P. Sapna. Optimal management of cross-trained workers in services with negligible switching costs. *European Joint of Operational Researchs.*, 167:349–369, 2005.

[15] L. Kleinrock. Queueing systems. *Vol. 7: Theory, Wiley, New York*, 1975.

[16] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. Spudd: Stochastic planning using decision diagrams. *Proceedings of UAI 99, Stockholm Sweden*, pages 1–10, 1999.

[17] R. W. Wolff. Poisson arrivals see time averages. *Operations Research*, 30:223–231, 1982.

[18] M. Law and D. Kelton. Simulation modeling and analysis. *McGraw-Hill, New York*, 1991.

[19] Agner Fog. Instructions for the random number generator libraries. *GNU General Public License*, pages 1–30, December 2008.

[20] Hong Lian and Zhenkai Wan. The computer simulation for queuing system. *World Academy of Science, Engeneering and Technology*, 34(1):176–179, 2007.

[21] R. Bellman. Dynamic programming. *Princeton University Press*, 1957.

[22] Ronald A. Howard and James E. Matheson. Influence diagrams. *In Ronald A. Howard and James E. Matheson, editors, The Principles and Applications of Decision Analysis*, pages 719–762, 1984.

[23] M. L. Puterman. Markov decision process discrete stochastic dynamic programming. *New York, New York: John Wiley Sons, Inc.*, 1994.

[24] R. Iris Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E.Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *IEEE Intl. Conf. Computer-Aided Design*, pages 188–191, 1993.

[25] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, pages 142–150, 1989.

[26] Carlos Ernesto Guestrin. Planning under uncertainty in complex structured environments. *PhD. Thesis, Stanford University*, pages 1–303, August 2003.

[27] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. *Proceedings IJCAI-95, Montreal*, pages 1104–1111, August 1995.

[28] J. Hoey, P. Poupart, C. Boutilier, and Alex Mihailidis. Semi-supervised learning of a pomdp model of patient-caregiver interactions. *In Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 1–9, 2005.

[29] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, 1995.

[30] W. Winston. Optimality of monotonic policies for multiple server exponential queueing systems with state-dependent arrival rates. *Operations Research*, 26:1089–1094, 1978.