APLICACIÓN Y VALIDACIÓN DE DOS ALGORITMOS, PARA EL DISEÑO ÓPTIMO DE SISTEMAS DE COLAS MARKOVIANOS

GUILLERMO AUGUSTO VESGA ACEVEDO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2008

APLICACIÓN Y VALIDACIÓN DE DOS ALGORITMOS, PARA EL DISEÑO ÓPTIMO DE SISTEMAS DE COLAS MARKOVIANOS

GUILLERMO AUGUSTO VESGA ACEVEDO

Proyecto de grado para optar por el titulo de Ingeniero Industrial

Directores:

Ing. Pablo Maya
Universidad Industrial de Santander

Dr. Hillel Kumin
The University of Oklahoma

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2008

AGRADECIMIENTOS

El autor expresa sus agradecimientos a:

Ing. Pablo Maya, por su colaboración y atención para la presentación de este trabajo de grado.
Dr. Hillel Kumin, quien me enseño todo lo que se sobre el Diseño Óptimo.

A la Universidad Industrial de Santander y a todos sus docentes, por todas sus enseñanzas durante más de 5 años de estudio.

A la Universidad de Oklahoma y a todos sus docentes, por el apoyo durante el tiempo de mi estadía allí.

A mi familia, por su ayuda incondicional y por sacrificarse por mí.

TABLA DE CONTENIDO

Pág.

INTRODUCCIÓN 14
1. GENERALIDADES DEL TRABAJO DE GRADO 16
1.1 DESCRIPCIÓN DEL PROBLEMA DE INVESTIGACIÓN 16
1.2 JUSTIFICACIÓN DEL PROBLEMA DE INVESTIGACIÓN 17
1.3 OBJETIVOS DEL TRABAJO DE GRADO 18
1.3.1 Objetivo General 18
1.3.2 Objetivos Específicos 19
1.4 ALCANCE DEL TRABAJO DE GRADO 19
2. FUNDAMENTACIÓN TEÓRICA 21
2.1 TEORÍA DE LOS PROCESOS ESTOCÁSTICOS 21
2.2 TEORÍA DE COLAS 23
2.2.1 Definición de un Sistema de Colas 23
2.2.2 Elementos de un Modelo de Colas 24
2.2.3 Notación de Modelos de Colas 25
2.3 ANÁLISIS DE MODELOS DE COLAS USANDO CADENAS DE MARKOV 26
2.4 DISEÑO Y CONTROL DE SISTEMAS DE COLAS 28
2.4.1 Comparación entre Diseño y Control 29
2.4.2 Explicación del Diseño Óptimo 30
2.4.3 Diferentes Enfoques sobre el Diseño Óptimo 32
2.5 PROBLEMAS DE DISEÑO DE MAYOR RELEVANCIA PARA ESTE PROYECTO 42
2.5.1 Problema de Diseño M/M/s por Kumin 43
2.5.2 Problema de Diseño M/M/1 por Evans 49
2.5.3 Problema de Diseño M(k)/M/k por Wuyi YUE 57

2.6 HERRAMIENTAS INFORMÁTICAS DE MATLAB UTILIZADAS 59
2.6.1 El programa MATLAB 59
2.6.2 M – files. 60
2.6.3 Herramientas del *Optimization Toolbox* de MATLAB 60
3. DISEÑO ÓPTIMO DE SISTEMAS DE COLAS USANDO CADENAS DE MARKOV 62
3.1 DESCRIPCIÓN DEL PROBLEMA 62
3.2 CONSIDERACIONES ESPECIALES DEL PRO 64
3.3 EXPLICACIÓN DEL ALGORITMO 1 65
3.3.1 Diagrama de flujo del Algoritmo 1 69
3.3.2 Ejemplo general Algoritmo 1 70
3.3.3 Convergencia del Algoritmo 1 71
3.4 EXPLICACIÓN DEL ALGORITMO 2 72
3.5 EXPERIMENTACIÓN PREELIMINAR 75
3.5.1 Algoritmo 1 75
3.5.2 Algoritmo 2 81
3.5.3 Conclusiones Preeliminares 84
4. IMPLEMENTACIÓN Y EXPERIMENTACIÓN 86
4.1 ALGORITMO 1 86
4.1.1 Escenario A1 91
4.1.2 Escenario A2 94
4.1.3 Escenario B 95
4.1.4 Escenario C 97
4.1.5 Observaciones generales Matriz Tridiagonal 99
4.1.6 Escenario D 100
4.1.7 Escenario E 102
4.1.8 Escenario F 103
4.1.9 Observaciones generales Matriz Hessenberg Superior 103
4.2 ALGORITMO 2 105

4.2.1 Escenario A1 106
4.2.2 Escenario A2 108
4.2.3 Escenario B 109
4.2.4 Escenario C 110
4.2.5 Observaciones Matriz Tridiagonal 111
4.2.6 Escenario D 111
4.2.7 Escenario E 113
4.2.8 Escenario F 114
4.2.9 Observaciones Matriz Hessenberg Superior 114
5. APROXIMACIÓN INICIAL A LA SOLUCIÓN DEL SISTEMA M/M/S 115
5.1 PLANTEAMIENTO DEL PROBLEMA 115
5.2 NÚMERO ESPERADO DE UNIDADES EN EL SISTEMA M/M/S 116
5.3 EXPLICACIÓN DE LA SOLUCIÓN 121
6. CONSIDERACIONES FINALES 125
6.1 CONCLUSIONES 125
6.2 CONTRIBUCIONES DEL TRABAJO DE GRADO 127
6.3 RECOMENDACIONES 127
BIBLIOGRAFÍA 129
ANEXOS 131

LISTA DE TABLAS

Pág.

Tabla 1. Ordenamiento de los estados 27
Tabla 2. Resultados numéricos - M/M/s 46
Tabla 3. Resultados Algoritmo 1 80
Tabla 4. Resultados Algoritmo 2- Iteración 1 82
Tabla 5. Resultados Algoritmo 2- Iteración 2 83
Tabla 6. Resultados Algoritmo 2- Iteración 3 83
Tabla 7. Resultados Algoritmo 2- Iteración 4 83
Tabla 8. Resultados Algoritmo 2- Iteración 5 84
Tabla 9. Experimentos n = 10 para A1 91
Tabla 10. Valores óptimos μ^* encontrados mediante el algoritmo 1 en A1 92
Tabla 11. Valores $g(\mu^*)$ algoritmo 1 en A1 92
Tabla 12. Desviaciones $|\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}|$ en A1 93
Tabla 13. Valores óptimos μ^* encontrados mediante el algoritmo 1 en A2 94
Tabla 14. Valores $g(\mu^*)$ algoritmo 1 en A2 94
Tabla 15. Desviaciones $|\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}|$ en A2 95
Tabla 16. Valores óptimos μ^* encontrados mediante el algoritmo 1 en B 96
Tabla 17. Valores $g(\mu^*)$ algoritmo 1 en B 96
Tabla 18. Desviaciones $|\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}|$ en B 97
Tabla 19. Valores óptimos μ^* encontrados mediante el algoritmo 1 en C 98
Tabla 20. Valores $g(\mu^*)$ algoritmo 1 en C 98
Tabla 21. Desviaciones $|\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}|$ en C 99
Tabla 22. Experimentos n = 5 para D 101
Tabla 23. Valores óptimos μ^* encontrados mediante el algoritmo 1 en D 101
Tabla 24. Valores $g(\mu^*)$ algoritmo 1 en D 101

Tabla 25. Valores óptimos μ* encontrados mediante el algoritmo 1 en E 102
Tabla 26. Valores g(μ*) algoritmo 1 en E 102
Tabla 27. Valores óptimos μ* encontrados mediante el algoritmo 1 en F 103
Tabla 28. Valores g(μ*) algoritmo 1 en F 103
Tabla 29. Experimentos n = 10, k = 6 107
Tabla 30. Valores óptimos μ* y g (μ*) encontrados mediante el algoritmo 2 107
Tabla 31. Desviaciones | $\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}$ | en A 108
Tabla 32. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 108
Tabla 33. Desviaciones | $\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}$ | en A2 109
Tabla 34. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 109
Tabla 35. Desviaciones | $\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}$ | en B 110
Tabla 36. Valores óptimos μ* y g(μ*)encontrados mediante el algoritmo 2 110
Tabla 37. Desviaciones | $\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}$ | en C 111
Tabla 38. Experimentos n = 3, k = 6 112
Tabla 39. Experimentos n = 10, k =6 112
Tabla 40. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 en D 113
Tabla 41. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 113
Tabla 42. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 114
Tabla 43. Resultados numéricos de los Algoritmos para el sistema M/M/s 123

LISTA DE FIGURAS

Pág.

Figura 1. Matriz de Transición P 22
Figura 2. Distribución de estado estable 23
Figura 3. Estructura básica de los modelos de colas 25
Figura 4. Sistema de Colas Simple 26
Figura 5. Posibles transiciones del modelo 27
Figura 6. Matriz de transición del modelo 28
Figura 7. Función Convexa M/M/s 47
Figura 8. Cambio de c1 – M/M/s 48
Figura 9. Cambio de c2 – M/M/s 48
Figura 10. Cambio de c3 – M/M/s 49
Figura 11. Matriz de Transición modelo Evans 52
Figura 12. Sistema de Colas M(k)/M/k 57
Figura 13. Función *fmincon* de MATLAB 61
Figura 14. Sistema de colas M/M/1 63
Figura 15. Matriz Tridiagonal 64
Figura 16. Matriz Hessenberg Superior 65
Figura 17. Diagrama de flujo del Algoritmo 1 69
Figura 18. Diagrama de flujo del Algoritmo 2 74
Figura 19. Matriz- Q para el sistema M/M/s 118
Figura 20. Función Convexa M/M/s. Enfoque: Cadenas de Markov 123
Figura 21. Función Convexa M/M/s. Enfoque: Expresiones Cerradas 124

LISTA DE ANEXOS

Pág.

ANEXO A. Códigos de programación 93

ANEXO B. Experimentos realizados 105

RESUMEN

TITULO: APLICACIÓN Y VALIDACIÓN DE DOS ALGORITMOS, PARA EL DISEÑO ÓPTIMO DE SISTEMAS DE COLAS MARKOVIANOS[*]

AUTOR: VESGA ACEVEDO, Guillermo[**]

PALABRAS CLAVES: Diseño Óptimo, Sistemas de Colas, Algoritmos, Cadenas de Markov, Modelo M/M/1, Optimización.

DESCRIPCIÓN:

En este trabajo de grado se presenta la validación y aplicación de dos algoritmos que sirven para el Diseño Óptimo de Sistemas de Colas Markovianos. Estos algoritmos utilizan las propiedades de matrices de transición asociadas con ciertas cadenas de Markov para hallar los valores óptimos del problema de diseño. Para hacerlo se llevo a cabo una investigación exhaustiva de los diferentes enfoques dados al problema de diseño de un sistema de colas, y se presento una descripción completa sobre los algoritmos implementados en este proyecto.

Posteriormente se implementaron y validaron los algoritmos propuestos, mediante la aplicación de diversos programas desarrollados por medio del software de optimización de MATLAB, y la solución de diferentes problemas de diseño asociados a un modelo de tiempo discreto de un sistema de colas con un solo servidor M/M/1. Se obtienen resultados idénticos mediante la aplicación de los algoritmos y el uso de una expresión cerrada para el diseño de este sistema.

Finalmente se realizó una aproximación inicial a la solución de un problema de diseño de un sistema de colas M/M/s utilizando las propiedades de una cadena de markov de tiempo discreto. Esta matriz se determino mediante la transformación de una cadena de markov de tiempo continuo en su cadena de Markov encajada. Mas adelante se compara el resultado obtenido con los métodos convencionales de diseño optimo basados en expresiones cerradas.

---

[*] Proyecto de grado en la modalidad de Investigación.

[**] Facultad de Ingenierías Físico Mecánicas, Escuela de Estudios Industriales y Empresariales, Ing. Pablo Maya Duque.

ABSTRACT

TITLE: APPLICATION AND VALIDATION OF TWO ALGORITHMS FOR THE OPTIMAL DESIGN OF MARKOVIAN QUEUING SYSTEMS.*

AUTOR:

VESGA ACEVEDO, Guillermo**

KEY WORDS :

Optimal Design, Queuing Systems, Algorithms, Markov Chains, M/M/1 model, Optimization.

.

DESCRIPTION:

This Project presents the validation and application of two algorithms for the Optimal Design of Markovian Queuing Systems. These algorithms use the properties of the transition matrices associated with certain markov chains to obtain the optimal values of the Makov chain design problem. A comprehensive research was prepared for this purpose to study the different approaches given to the design of queuing systems. A complete explanation of the algorithms studied and implemented in this project is prepared.

Subsequently, the proposed algorithms were implemented and validated by solving a variety of problems associated with the discrete model of a single server queue M/M1 using several programs designed in MATLAB's optimization software. The results obtained with the application of the algorithms and those found with the use of a closed form expression to design this queuing system are identical.

Finally an initial approximation to the solution of a design problem associated with a M/M/s queuing system is performed using the algorithms and the properties of a discrete time markov chain. This matrix is obtained through the transformation of a continuous time markov chain to its embedded markov chain. Afterwards, the results obtained by applying the algorithms are compared with those using traditional methods of optimal design based on closed form expressions.

---

* Undergraduate Project, Research

** Physical-Mechanical Engineering Faculty, School of Industrial and Managerial Studies, Eng. Pablo Maya Duque.

INTRODUCCIÓN

En el diseño de nuevas instalaciones o plantas, la elección de la tasa de servicio óptima a proveer en las diferentes estaciones de servicio normalmente se realiza a través de la experiencia o por medio de simulación. La simulación permite reunir información sobre el comportamiento del sistema y los datos recopilados se usan después para diseñar el sistema. Sin embargo, la simulación no es una técnica de optimización. (Taha, 1997). El enfoque de esta investigación es diseñar nuevos sistemas mediante la aplicación de técnicas de optimización en modelos de diseño óptimo.

Los modelos de diseño óptimo son generalmente colas estándares, a las cuales se les imponen funciones de costo o utilidad para ser optimizadas con respecto a parámetros como: la tasa de servicio, la tasa de llegadas o el número disponible de servidores. (Tadj y Choudhury, 2005).

Es importante clasificar los modelos de colas en dos tipos: descriptivos y prescriptivos. Los modelos descriptivos describen los sistemas del mundo real actual, mientras los modelos prescriptivos prescriben como los sistemas del mundo real deben ser; lo más óptimos posibles. (Gross, 1998). El desarrollo de la teoría de colas ha estado dominado principalmente por modelos descriptivos. Muy poca atención ha sido brindada a modelos prescriptivos, modelos sobre el óptimo diseño y control de colas. (Tadj y Choudhury, 2005).

En esta investigación se estudian las diferentes aproximaciones dadas al problema del diseño óptimo de sistemas de colas markovianos (Modelos prescriptivos); específicamente se aborda el problema de hallar la tasa de servicio

óptima a proveer en el diseño de un sistema de colas M/M/1 solucionando problemas de optimización.

Este proyecto de investigación esta basado en los resultados obtenidos por el doctor Hillel Kumin en el desarrollo de su investigación doctoral, los cuales se presentan en el paper, “The Design of Markovian Congestion Systems”, H. Kumin (1968), y nace como respuesta a la necesidad de continuar con la línea de trabajo iniciada en un intercambio con la Universidad de Oklahoma, a finales de 2006; período durante el cual se avanzó en la comprensión de la investigación desarrollada por Kumin y que permitió identificar el problema de investigación se pretende abordar.

Este trabajo se dividió en varias etapas. La primera consistió en la revisión de la literatura disponible y en la compresión, por medio de esta, del problema de diseño de sistema de colas markovianos y de la metodología de diseño propuesta por Kumin.

Luego se retomaron los resultados obtenidos por Kumin en su trabajo doctoral, principalmente los algoritmos para el Diseño Óptimo de Sistemas de Colas Markovianos y se estudió un modelo de tiempo discreto asociado a un sistema de colas M/M/1, mediante el cual se implementaron y validaron los algoritmos.

Finalmente se aprovecharon los sistemas actuales para extender el trabajo numérico realizado anteriormente, solucionando problemas de mayor complejidad y tamaño. Se realizó una aproximación inicial a la solución de un problema de diseño asociado a un sistema de colas M/M/s.

1. GENERALIDADES DEL TRABAJO DE GRADO

1.1 DESCRIPCIÓN DEL PROBLEMA DE INVESTIGACIÓN

En esta investigación se estudia un modelo de tiempo discreto, de un sistema de colas M/M/1 (single server queue), con una sola cola y un único servidor; donde las tasas de llegada λ y de servicio μ, tienen distribución de probabilidad exponencial.

El problema consiste en hallar el valor del parámetro μ, tasa de servicio, que minimice una función de costos que depende: del costo asociado a la tasa de servicio c1 y del costo asociado al número esperado de clientes en el sistema c2. Este número esperado de clientes E(μ, λ), a su vez depende de la tasa de servicio y de la tasa de llegadas.

Se dice que al hallar el valor de μ que minimice la función de costos, este valor se establece como parámetro, y por lo tanto, se diseña un sistema. La función objetivo a minimizar (optimizar) es la siguiente:

$$F.O: \quad \text{Min}_{\mu}\, g(\mu) = c_1 \mu + c_2 E(\mu, \lambda)$$

Para este trabajo de grado se propuso la validación de los algoritmos propuestos por Kumin(1968) y a partir de los avances de los sistemas actuales, lograr adelantos en la optimización de sistemas o modelos de colas más complejos

Se propuso también dar solución entre otras a las siguientes preguntas de investigación:

¿Convergen los algoritmos a la solución óptima del problema de diseño planteado y de ser así, que factores, asociados a la definición del algoritmo y del problema, tienen incidencia en dicha convergencia?

¿Qué ventajas y/o desventajas presentan los algoritmos para el diseño evaluados, con respecto al uso de expresiones cerradas para la optimización de los sistemas?

¿Tiene validez el estudio de la aplicación de estos algoritmos o algoritmos similares soportados en los mismos principios, para la solución de problemas más complejos, particularmente el caso del modelo con múltiples servidores?

1.2 JUSTIFICACIÓN DEL PROBLEMA DE INVESTIGACIÓN

La teoría sobre cadenas de markov y los sistemas de colas es bien conocida, sin embargo, se ha investigado poco sobre el diseño óptimo de sistemas de colas markovianos en los últimos años. La mayoría de los sistemas de colas descritos en los libros de introducción a la Investigación de Operaciones son markovianos; pero los resultados presentados son generalmente descriptivos (Stidam, 2006).

Estos están enfocados principalmente en el análisis de las medidas de rendimiento del sistema como: el número promedio de clientes en el sistema, el tiempo promedio de espera en el sistema o el tamaño esperado de la cola. Estas medidas son necesarias para la evaluación de alternativas para controlar o mejorar sistemas que se encuentran en funcionamiento; sin embargo, se desarrolla actualmente poco trabajo en el diseño de sistemas de colas. Es decir,

solucionando problemas de optimización basados en ellos, para hallar los valores de sus parámetros y diseñar sistemas óptimos. Este es el enfoque que se propone en esta investigación.

Los problemas de diseño son de interés desde varias perspectivas. Muchos problemas reales como: el número de estaciones de servicio y las tasas de servicio a proveer, son importantes en el diseño de nuevas instalaciones o plantas. Existen numerosos problemas de colas donde se pueden aplicar estos problemas como supermercados, teatros, aeropuertos, plantas de mantenimiento, cualquier lugar donde haya que esperar. (Evans, 1981)

Kumin (1968) formuló dos algoritmos iterativos que convergen a la solución óptima de los problemas de diseño propuestos. Los algoritmos presentados tienen la particularidad de usar las propiedades de las matrices de transición asociadas con las cadenas de Markov para diseñar sistemas markovianos.

Estos algoritmos no han sido validados pues cuando se propusieron el hardware y software disponibles fueron insuficientes para resolver los problemas propuestos y realizar complejas operaciones numéricas, algo más fácil de realizar con los sistemas actuales. Hasta el momento nadie ha aplicado estos algoritmos por lo que en esta investigación serán retomados, bajo la dirección de Kumin, su creador.

1.3 OBJETIVOS DEL TRABAJO DE GRADO

1.3.1 Objetivo General: Aplicar y validar dos algoritmos para el diseño óptimo de sistemas de colas markovianos, solucionando los problemas asociados a un modelo de colas de tiempo discreto, e identificando a partir de la experimentación los aspectos más relevantes asociados a la convergencia del algoritmo y su aplicabilidad en sistemas más complejos.

1.3.2 Objetivos Específicos

- Realizar un análisis exhaustivo de las distintas aproximaciones dadas al problema de diseño de sistemas de colas markovianos.
- Estudiar los algoritmos propuestos por Kumin para el diseño óptimo de sistemas de colas, como una metodología alterna al método convencional de diseño óptimo.
- Diseñar e implementar un programa que permita aplicar los algoritmos que son objeto de estudio de este proyecto.
- Solucionar los diferentes problemas de diseño propuestos asociados con un modelo de tiempo discreto de un sistema de colas con un solo servidor, variando los valores de sus parámetros; y validar los algoritmos.
- Estudiar el desempeño de los algoritmos respecto a aspectos como su convergencia.
- Realizar una aproximación inicial a la solución de un problema de diseño de sistemas de colas markovianos con más de un servidor.

1.4 ALCANCE DEL TRABAJO DE GRADO

Se crea una descripción completa de las distintas aproximaciones dadas al problema de diseños markovianos, la cual servirá como base para proyectos y como fuente de consulta para los proyectos futuros que podrían derivarse.

Se diseñan e implementan varios programas por medio del software de Optimización de Matlab de The MathWorks, Inc que permiten aplicar los algoritmos que son objeto de estudio de este proyecto, de forma automática a cualquier instancia que cumpla con los supuestos y condiciones básicas delimitadas por el tipo de problemas que se pretende abordar. Mediante la aplicación de dicho programa se validan los algoritmos presentados y se evalúa su desempeño principalmente en términos de su convergencia.

Los algoritmos propuestos son aplicados para hacer una aproximación inicial a la solución de un problema de diseño de sistemas de colas markovianos con más de un servidor. Además esta aproximación se compara con el resultado obtenido de solucionar un problema similar usando otra metodología (expresiones cerradas).

2. FUNDAMENTACIÓN TEÓRICA

2.1 TEORÍA DE LOS PROCESOS ESTOCÁSTICOS

Uno de los problemas más estudiados el campo de la Investigación de Operaciones son los Procesos Estocásticos. En este tipo de procesos parte de la información más importante no se conoce con certeza como es el caso de los determinísticos, sino que más bien se comporta de una manera probabilística. *Las Cadenas de Markov son un tipo especial de proceso estocástico. (Taha, 1997)*

Un proceso estocástico se define sencillamente como una colección indexada de variables aleatorias $\{ X_t \}$, donde el subíndice t toma valores de un conjunto T, llamado *rango de tiempo*. T es un subconjunto de $(-\infty,+\infty)$ y X_t significa una observación en el tiempo t. Si T es discreto, entonces T = { 0,1,2,3....}, y se dice que es un *proceso estocástico de tiempo discreto*. Si T es continuo, entonces $T = \{ t : 0 \leq t \geq +\infty \}$, se dice que es un *proceso estocástico de tiempo continuo* (Takacs, 1962). En esta investigación se consideran procesos estocásticos de tiempo discreto, y específicamente, *Cadenas de Markov de Tiempo Discreto.*

Teoría de las Cadenas de Markov. Una cadena de Markov es una serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. En efecto, las cadenas de este tipo solo recuerdan el último evento y esto condiciona las posibilidades de los eventos futuros, esta es la propiedad fundamental de un sistema de Markov. Esta dependencia del evento anterior distingue a las cadenas de Markov de las series de eventos independientes, como tirar una moneda al aire o un dado. (Takacs, 1962)

En general, si se considera un proceso aleatorio en el que se produce un cambio de estado en ciertos instantes de tiempo discretos t = 0, 1, 2, . . .,n. Supongamos que hay un número finito de eventos (estados) exhaustivos y mutuamente excluyentes posibles S_1, S_2, . . . , Sn. Surge así una sucesión de situaciones X_0,X_1, . . . ,X_t, . . ., en la que cada X_t es igual a uno de los n estados. Este proceso se llama ***Cadena de Markov*** si las probabilidades condicionadas que expresan este cambio de situaciones satisfacen ***la propiedad de Markov***:

$$\mathbf{P(X_{t+1} = S_j \,|X_t = S_i, X_{t-1} = S_{i\,t-1}, \ldots, X_0 = S_{i\,0}) =}$$
$$\mathbf{P(X_{t+1} = S_j \,|X_t = S_i)}$$

para todos los instantes t, todos los estados S_i, S_j , y todas las posibles sucesiones S_{i0} , . . . , $S_{i\,t-1}$ de estados previos. Esta propiedad traduce la condición de que las cadenas de Markov no guardan su historia pasada en la memoria; en cada momento, la evolución al estado siguiente depende del resultado de la situación inmediatamente anterior, pero es independiente de la sucesión de todas las situaciones anteriores.

El valor $p_{ij}(t) := P(X_t = S_j \,|X_{t-1} = S_i)$ es la probabilidad de que la cadena esté en el estado Sj en el instante t dado que haya pasado por el estado S_i en el instante t−1; por esto, a $p_{ij}(t)$ se le llama la probabilidad de transición de moverse del estado S_i al S_j en el instante t. La matriz de las probabilidades de transición P(t) : = ($p_{ij}(t)$) es no negativa para toda t, y sus filas suman 1. Así pues P(t) es una matriz estocástica cuando las probabilidades de transición no dependen del tiempo.

Figura 1. Matriz de Transición P

$$P = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \ldots\ldots \\ P_{10} & P_{11} & P_{12} & P_{13} \ldots\ldots \\ P_{20} & P_{21} & P_{22} & P_{23} \ldots\ldots \\ P_{30} & P_{31} & P_{23} & P_{33} \ldots\ldots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Recíprocamente, toda matriz estocástica P = (pij) de tamaño n x n define una cadena de Markov de n estados porque los elementos pij definen un conjunto de probabilidades de transición, que puede ser interpretado como una cadena de Markov estacionaria con n estados.

Distribución de estado estable. Sea P la matriz de transición de una cadena de M estados. Existe entonces un vector $\pi = [\pi_1, \pi_2, .., \pi_m]$ tal que:

Figura 2. Distribución de estado estable

$$\lim_{n \to \infty} P^n = \begin{bmatrix} \pi_1 & \pi_2 & \dots & \pi_M \\ \pi_1 & \pi_2 & \dots & \pi_M \\ . & . & & . \\ . & . & & . \\ \pi_1 & \pi_2 & \dots & \pi_M \end{bmatrix}$$

Se establece que para cualquier estado inicial i, $\lim_{n \to \infty} P_n(n) = \pi_j$. El vector $\pi = [\pi_1, \pi_2, .., \pi_m]$ a menudo se llama *distribución de estado estable*, o también distribución de equilibrio para la cadena de Markov. (Taha, 1997)

2.2 TEORÍA DE COLAS

En la mayoría de las organizaciones existen ejemplos de procesos que generan colas de espera. Estas colas suelen aparecer cuando un usuario, un empleado, una máquina o una unidad tiene que esperar a ser servidas debido a que la unidad de servicio o servidor, operando a plena capacidad, no puede atender temporalmente a este servicio. (Serra, 2002)

2.2.1 Definición de un Sistema de Colas. Un sistema de colas es una estructura en la que se producen llegadas en el tiempo de ciertas unidades denominadas clientes para recibir algún tipo de operación o servicio por parte de otras unidades

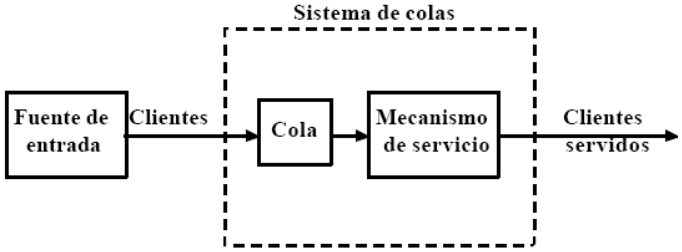
que se denominan servidores. Cuando los tiempos de llegada y de servicio tienen distribución de probabilidad exponencial, el sistema de colas es un *"Sistema de Colas Markoviano".*

Los modelos de colas intentan simular el sistema en donde puede existir congestión (y por lo tanto, colas) y generan una serie de parámetros que permiten evaluar el sistema de colas actual y valorar la realización de modificaciones en el servicio en cuestión.

2.2.2 Elementos de un Modelo de Colas. El mecanismo de servicio de un sistema de colas es el siguiente: en el instante en el que se produce la llegada de un cliente, éste comienza a ser atendido si existe algún servidor desocupado. Si por el contrario, todos los servidores se encuentran ocupados atendiendo a otros clientes, el cliente que llega debe aguardar para recibir su servicio formando una línea de espera o bien, abandonar el sistema, según estén definidas las características del mismo.

La fuente de entrada define la forma como los clientes llegan y se unen al sistema. El número de clientes puede ser infinito o finito, pueden llegar en grupos o individualmente. Un factor importante a tener en cuenta es el patrón estadístico mediante el cual se generan los clientes a través del tiempo. La disciplina de la cola, que representa el orden en el que los clientes se seleccionan de una cola, es un factor importante en los modelos de colas. La disciplina más común es al que llega primero se le atiende primero (FCFS). Otras disciplinas incluyen al que llega de ultimo se le atiende primero (LCFS) y servicio en orden aleatorio (SIRO).

Figura 3. Estructura básica de los modelos de colas



(Taha, 1997)

2.2.3 Notación de Modelos de Colas. Según Ausin (2004), se puede lograr una descripción adecuada del funcionamiento del sistema con cinco características básicas que se pueden sintetizar de forma compacta siguiendo la notación de Kendall (1953) como:

A / S / c / K / R

La interpretación es la siguiente: A y S reflejan la conducta de las llegadas y los servicios, respectivamente, c es el número de servidores, K representa la capacidad del sistema y, por último, R es la disciplina establecida para acceder al servicio. Si alguna de las variables A o S se distribuye exponencialmente se denota con M, que hace referencia a que el proceso de llegadas o de servicio sigue un proceso de Markov. (Ausin, 2004)

En esta investigación se estudiaran principalmente dos modelos: M/M/1, modelo donde tanto los tiempos entre llegada, como los tiempos de servicio son exponenciales y se tiene solo un servidor y M/M/s, modelo donde tanto los tiempos entre llegada, como los tiempos de servicio son exponenciales y se tienen s servidores.

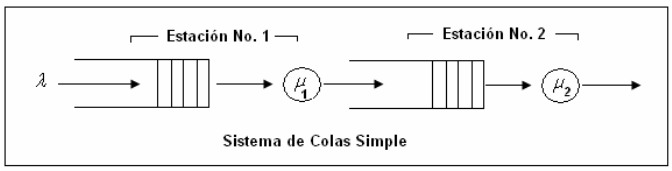
2.3 ANÁLISIS DE MODELOS DE COLAS USANDO CADENAS DE MARKOV

Normalmente ocurre que las cadenas de markov que se analizan provienen de modelos de colas; por tal razón es apropiado examinar estos modelos y las características de sus cadenas de markov resultantes. Los modelos de colas imponen en la matriz de transición, también denominada *generador infinitesimal*, una estructura que juega un papel importante en el desarrollo de métodos numéricos exitosos para solucionar cadenas de markov. (Stewart, 1994)

Concluyendo esta sección, se considerara un modelo de colas elemental, con el propósito de exponer los pasos necesarios para generar su cadena de Markov asociada.

Ejemplo: Dos estaciones de servicio consecutivas

Figura 4. Sistema de Colas Simple



(Stewart, 1994)

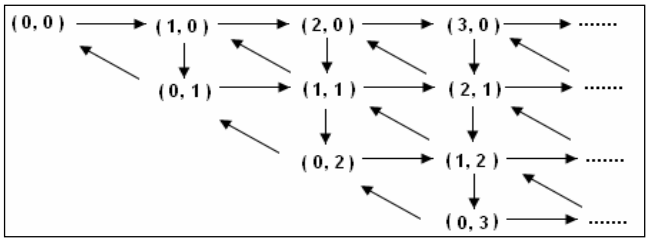
Descripción del modelo: El proceso de llegadas es poisson, con una tasa λ y La distribución del servicio es exponencial, con un tasa μ. n_1 y n_2 representan el número de clientes en las estaciones 1 y 2 respectivamente

Pasos:

1. Definir la representación de los estados: el estado del sistema será descrito por (n_1 , n_2), donde n_1 representa el número de clientes en la estación No. 1 y n_2 representa el número de clientes en la estación No. 2.

2. Establecer todas las transiciones que pueden ocurrir entre estos estados: para generar la Cadena de Markov se necesita considerar las posibles *transiciones de un solo paso* que pueden ocurrir entre dos estados cualesquiera. Ver figura

Figura 5. Posibles transiciones del modelo



(Stewart, 1994)

En resumen, las transiciones posibles, y las tasas correspondientes son: $(n_1 , n_2) \rightarrow (n_1 + 1, n_2)$: a una tasa λ, $(n_1 , n_2) \rightarrow (n_1 - 1 , n_2 + 1)$: a una tasa μ_1, $(n_1 , n_2) \rightarrow (n_1 , n_2 - 1)$: a una tasa μ_2.

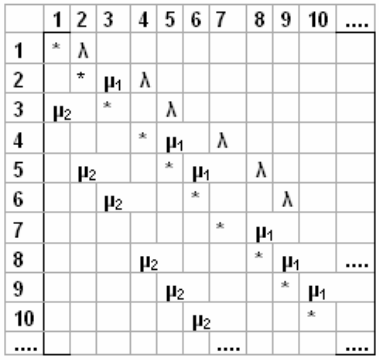
3. Crear la matriz de transición o generador infinitesimal: la estructura de la matriz de transición para esta cadena de markov depende del orden impuesto para los estados.

Tabla 1. Ordenamiento de los estados

Estado No.	Descripción: (n_1 , n_2)
1	(0, 0)
2	(1, 0)
3	(0, 1)
4	(2, 0)
5	(1, 1)
6	(0, 2)
7	(3, 0)

8	(2, 1)
9	(1, 2)
10	(0, 3)

Figura 6. Matriz de transición del modelo



(Stewart, 1994)

De acuerdo al ordenamiento establecido se obtiene la matriz de transición; este tipo de estructuras provienen frecuentemente de cadenas de markov asociadas con modelos de colas, y de esta forma pueden ser analizadas por métodos numéricos para estudiar las probabilidades de estado estable. (Stewart, 1994). La anterior explicación permitirá entender de donde provienen las matrices de transición que son objeto de estudio de esta investigación.

2.4 DISEÑO Y CONTROL DE SISTEMAS DE COLAS

Desde los inicios de la Investigación Operacional existió cierto interés en el desarrollo de modelos prescriptivos (que buscan optimizar la solución) en oposición a modelos descriptivos, los cuales hacen parte de la mayoría de la literatura sobre colas. Este interés en los modelos prescriptivos ha estado dirigido principalmente al estudio de los Procesos de Decisión Markovianos, procesos exclusivamente de control. (Stidham, 2006)

Con respecto a las aplicaciones de los modelos prescriptivos, es muy frecuente encontrar numerosos textos investigativos que mencionan el diseño y control de

colas. Los modelos prescriptivos han sido aplicados satisfactoriamente a través de los años en numerosos problemas en diferentes campos. Tadj y Choudhury (2005) mencionan los diferentes autores de estos modelos y señalan los siguientes campos de aplicación: sistemas telefónicos, computadores y sistemas de comunicación, (ISDN) redes digitales de servicios integrales, telecomunicaciones, redes de almacenamiento, sistemas de recuperación, sistemas de manufactura de empresas de aviación, sistemas de producción, inventarios, programación de tareas, control de calidad, sistemas de transporte , etc.

2.4.1 Comparación entre Diseño y Control. A continuación se hace una breve descripción de los Procesos de Decisión Markovianos, procesos exclusivamente de control. Este proceso inicialmente fue descrito por Howard (1960):

> *Proceso en el cual el sistema realiza transiciones markovianas de un estado a otro, ente un conjunto finito de estados, acumulando un resultado en cada transición. Un decisión es tomada en cada paso entre un número finito de alternativas; esta decisión afecta las probabilidades de transición y los resultados obtenidos al abandonar el estado presente. El problema consiste en especificar la política óptima o "regla" de decisiones a realizar en cada estado, que maximice la ganancia total esperada al final del experimento.*

Recientes investigaciones señalan la aplicabilidad de los procesos de decisión markovianos, tal es el caso de Stern (2005); en su paper "Optimal control of a facility with periodic interrupted demand" se estudia un sistema que consiste en el flujo periódico e interrumpido de materiales a una planta, una operación de procesado con tasas de servicio deterministicas y controlables, y un almacén. Se asocia al sistema una estructura de costos compuesta por los costos de procesamiento y almacenamiento. Una regla de control sobre las tasas de servicio de procesado a determinar, que minimice el costo esperado en el largo plazo es determinada.

De la misma manera Baris (2006), en su paper “Dynamic Control of an M/M/1 Service System with Adjustable Arrival and Service Rates”, estudia una planta de servicio donde el jefe del sistema controla las tasas de llegada y servicio para maximizar la ganancia esperada en el largo plazo.

Si en vez de realizar un proceso de control como el descrito anteriormente, tenemos que realizar una sola elección de un sistema, dadas ciertas condiciones iniciales, se estaría implementando un proceso de diseño en lugar de uno de control. Evans (1981) realiza una distinción entre los problemas de diseño y los de control de la siguiente manera:

En el problema de diseño, se fija la estructura del servicio una vez en el tiempo 0. Esta estructura se usa para todos los estados, en todos los tiempos. Por ejemplo, se seleccionan unas condiciones iniciales para un solo sistema, y se halla el valor de un parámetro como la tasa de servicio para minimizar una función de costos asociada a ese sistema. Este único valor es la solución del problema, y es el que minimiza los costos para este sistema, por tal razón se establece como un parámetro de diseño.

En un problema de control, se puede escoger una tasa de servicio diferente cuando se desee, ya sea con la llegada o partida de algún cliente para controlar el sistema. Por ejemplo, un problema de control es el de hallar la política óptima de tasas de servicio a seleccionar, en cada estado, para minimizar el costo esperado en el largo plazo. Se dice que la principal diferencia entre el diseño y el control, es que en el segundo, el problema puede ser estudiado estado por estado, mientras en el diseño no.

2.4.2 Explicación del Diseño Óptimo. Los problemas de diseño óptimo, fueron vistos como problemas de optimización estáticos, es decir, problemas en los

cuales varias configuraciones de sistemas son estudiados, y el análisis resultante del comportamiento de las probabilidades del estado estable, permite determinar *"el mejor sistema"* para optimizar algún criterio en el largo plazo como el costo o la utilidad.

El término "estático" se refiere a que una vez la configuración del sistema se establece, las características del sistema no cambian con el tiempo. Se llaman modelos dinámicos o de control, si las características operativas del sistema pueden cambiar en el tiempo. La mayor parte de los estudios han estado concentrados en modelos dinámicos. (Stidham, 2006)

Esta investigación estará concentrada especialmente en este tipo de problemas; en los modelos estáticos, parámetros como la tasa de llegadas λ, la tasa de servicio μ o el número de servidores *c*, se permiten variar para optimizar una función objetivo asociada a un modelo de colas. Es importante enfatizar que estos parámetros no varían con el tiempo. Por eso, este tipo de problemas son comúnmente llamados problemas de diseño óptimo, pues establecer parámetros constituye un "diseño de un sistema".

Algunos ejemplos de las funciones típicas de costo y utilidad de estos modelos son:

$$\min_c Z = C_0 + C_w \lambda W + C_c c \text{ or } \min_\mu Z = C_0 + C_w \lambda W + C_\mu \mu$$

Donde:

Co es el costo fijo de operación del sistema
Cw es el costo por hora de espera del cliente
Cc es el salario por hora de un servidor
Cu es el salario de un servidor que trabaja a una tasa u
W es el tiempo promedio de espera en el sistema
C es el número de servidores

λ es la tasa de llegadas

Primero que todo es necesario hallar el valor de W (tiempo promedio de espera en el sistema para este ejemplo) para el sistema de colas y luego incluirlo en la función de costos para ser minimizada.

Muchas investigaciones en esta categoría comparan la misma función objetivo para un conjunto finito de diseños alternativos y el problema consiste en hallar el valor mínimo de la función. Sin embargo, el concepto básico de este tipo de análisis es el de hallar las probabilidades de estado estable o los valores esperados de congestión para un determinado sistema de colas en una forma estándar (ecuaciones, funciones, etc.) y luego usar esas medidas y construir una función objetivo para ser optimizada. (Thomas, 1977)

2.4.3 Diferentes Enfoques sobre el Diseño Óptimo. El diseño óptimo de un sistema de colas consiste en determinar los parámetros óptimos de un sistema, como la tasa de servicio óptima o el número óptimo de servidores a ubicar en una estación de servicio. El problema estático o de diseño ha sido estudiado formalmente como un problema de optimización por pocos investigadores.

A continuación se realiza una descripción de los diferentes enfoques brindados por diversos autores desde el inicio del diseño óptimo hasta el día de hoy. Esta recopilación bibliográfica esta basada en información encontrada en 4 papers: Kumin(1968), Evans(1981), Tadj y Choudhury (2005), Stidham, S. Jr. y Bagajewicz, M. (2006).

El primer sistema de colas diseñado corresponde a Brigham (1955). El estuvo interesado en determinar el número óptimo de servidores a ubicar en las estaciones de servicio de las plantas de la compañía de aviones Boeing. Usando un Modelo M/M/c y la siguiente función lineal para el costo

$$C = r_c * W_c + r_a * W_a$$

Donde W_a es el tiempo total de espera de un cliente, W_c es el tiempo total de espera por servidor, r_a es el costo de un cliente por unidad de tiempo y r_c es el costo de un servidor por unidad de tiempo. Brigham presenta curvas mostrando el valor óptimo de c, como una función de λ / μ. La función de costo es llamada lineal porque la contribución de los diferentes componentes al costo total se asumen lineales usando sus valores promedio.

Morse (1958) consideró varios modelos económicos: un modelo M/M/1 para estudiar un problema de llegadas de barcos a un puerto con una sola entrada, un modelo M/M/1/k con perdidas de clientes, un modelo M/M/c/c con clientes impacientes, y varios problemas de reparación de maquinas. En todos los modelos se buscaba hallar la tasa óptima de servicio. Morse asumió que el costo del servicio era directamente proporcional a la velocidad del servicio y a la utilidad bruta por servicio completado.

Los dos primeros modelos fueron resueltos usando cálculo simple y para el tercer modelo presento una grafica donde para valores de λ, μ, y la razón entre el costo del servicio y la utilidad por cliente, el valor óptimo de c era determinado. Para los modelos de reparación de maquinas se desarrollaron funciones de mantenimiento preventivo.

Hllier (1963) consideró tres modelos, los cuales asumen lo siguiente:

- El costo total de esperar es proporcional al tiempo total que todas las llegadas perduran en el sistema.
- El costo del servicio en cada planta es una función lineal del número de servidores en cada planta.
- Todas las llegadas que se unen a la cola esperan hasta ser servidos y no hay limitación en el tamaño de la cola.
- La cola esta en condición de estado estable

El *primero modelo* es el caso en el cual las tasas de llegada y de servicio eran fijadas y el problema consistía en determinar el número de servidores. El método usual para su solución era de prueba y error. Para el caso de un sistema M/M/s, Mangelsdorf (1955) encontró soluciones al establecer la primera diferencia de las funciones objetivo con respecto a s igual a cero y resolvió gráficamente.

El *segundo modelo* es el caso en el cual la tasa de llegadas y el número de servidores eran variables de decisión. Como la tasa de llegadas para la población entera se asume conocida, la naturaleza de la estructura de costos permite mostrar que (1) un servidor es lo óptimo.

El *tercer modelo* es el caso en el cual la tasa de servicio μ y el número de servidores s son variables de decisión y que el conjunto de valores que μ puede tomar es finito. El problema se resuelve por enumeración directa. Si A es el conjunto de valores posibles para μ, luego para cada μ, el valor de s es hallado de tal forma que genera el menor costo. Luego de estos valores del costo, el menor valor es seleccionado.

Swersey (1967) consideró una red cerrada de colas formada por un conjunto finito de N clientes, un conjunto finito de M servidores y un conjunto de arcos (i, j) que representan el movimiento de la estación i a la j. La ubicación óptima de servidores entre un número finito es considerada y se muestra como en una cola cíclica, el mejor resultado se obtiene al ubicar igual número de servidores en cada centro de servicio.

Los anteriores estudios fueron diseños simples de un solo parámetro. Según Kumin, si miramos diseños de dos parámetros, un enfoque analítico se vuelve difícil e impráctico, inclusive si consideramos sistemas de colas elementales.

Kumin (1968) estudió el caso de un problema de diseño de dos parámetros asociado a un sistema M/M/s. El problema considera un sistema de colas estándar con canales paralelos en el cual los clientes llegan al sistema con una tasa λ de distribución poisson. En cada canal de servicio se tiene la misma tasa de servicio con parámetro μ. Cuando el cliente llega es servido si hay un canal de servicio disponible, de lo contrario, se une a la cola y espera. La disciplina de la cola es FCFS. Los parámetros de diseño son el número de servidores s (variable discreta) y la tasa de servicio μ (variable continua). El problema de diseño consiste en hallar el valor óptimo de s y μ, tal que minimicen el costo total.

En el pasado se contó con mucha dificultad para resolver este problema mediante las técnicas de diseño óptimo tradicional debido a la complejidad de la función objetivo, además la dificultad existente para hallar expresiones cerradas para funciones definidas con elementos del conjunto de las probabilidades de estado estable, estableció la necesidad de una teoría de diseño que utilice las propiedades de las cadenas de Markov para encontrar el valor óptimo de estos parámetros, y no expresiones cerradas para los las medidas de desempeño del sistema. Más adelante se realizará una ampliación sobre las diferentes aproximaciones realizadas en esta investigación para solucionar este problema.

Kumin (1968) en su tesis doctoral desarrolló una teoría alterna de diseño óptimo, donde propone dos algoritmos para el diseño de sistemas de colas markovianos usando cadenas de markov. Consideró un modelo de tiempo discreto asociado a un sistema M/M/1. Los problemas propuestos en su investigación serán objeto de estudio en este proyecto, y se aprovechan los sistemas actuales para extender el trabajo realizado a problemas mucho más complejos.

Stidam (1970) considera el modelo M/M/c de Hiller donde las variables de decisión son el número de servidores c y la tasa de servicio μ. Morse (1958) había demostrado que el valor óptimo de c para este modelo es 1. Asumiendo una estructura de costos no lineales, Stidam generaliza este resultado para los modelos G/M/c, G/Ek/c, y G/D/c

Brosh (1970) considera un sistema de colas de capacidad finita L, con clientes demandando servicio cada cierto tiempo constante. La probabilidad de tener una llegada de un cliente por periodo es λ, y la probabilidad para terminar el servicio por periodo es μ_k ,para k = 1,2. Definiendo una regla R, la cual determina el tipo de servicio K que debe ser usado en el estado i = 0, 1, ... L, la secuencia infinita de estados observados es una cadena de markov finita. Brosh escribe las ecuaciones balanceadas que permiten la computación de su distribución estacionaria $\Pi_j(R)$, luego expresa la función esperada del costo así:

$$\Psi_R = c_1 \sum_{i \in M_1} \Pi_i(R) + c_2 \sum_{i \in M_2} \Pi_i(R) + \lambda c_3 \pi_L(R) - (1-\lambda) c_0(R) \Pi_0(R)$$

Donde c_k es el costo por unidad de tiempo incurrido por usar el servicio tipo k = 1,2 y c_3 es el costo debido a la perdida de un cliente, y M_{ij} (i = 1, 2), son los conjuntos de estados a los cuales la regla R les impone el servicio tipo i. Para determinar el mínimo de la función del costo esperado, Brosh compara cada función de costo con las demás. El reduce el número de políticas consideradas de 2^{L-1} a L – 2 y propone un algoritmo simple para hallar la política óptima.

Evans (1971) considera el problema de optimizar las probabilidades de estado estable de una cadena de markov con respecto a un parámetro desconocido μ. El asume una función de costo general y una matriz de transición de la cadena de markov considerada. Denotando el vector de las probabilidades de estado estable como una función del parámetro desconocido μ por p(μ) y asumiendo que la

matriz de probabilidades tiene la forma Q + μ*W, se formula el problema de la siguiente manera:

$$\text{Optimizar} \qquad c(\mu) + [\, f * p(\mu) \,]$$
$$\text{Sujeto a} \qquad p(\mu)\,[Q + \mu W\,] = (p(\mu),$$

Donde c (μ) es el costo asociado con el parámetro μ, f es el vector de los costos o utilidades y f · p(μ) es el producto punto de los vectores f y p(μ). Evans se concentra en obtener expresiones para las derivadas de la función objetivo con respecto de μ. Estas expresiones conllevan a planes iterativos que utilizan algoritmos de gradiente para hallar cadenas óptimas localmente. Este modelo puede ser interpretado de la siguiente manera: en la ausencia de una acción μ = 0 la matriz de transición es Q . Esto puede ser modificado en proporciones fijas sumando μW a Q, a un costo de c(μ). El problema define la modificación de y μ* que minimiza el costo de control c(μ) y el costo dependiente de las probabilidades de estado estable f · p(μ)

Bell (1980) utilizó una estructura de costos lineal para investigar el problema de ajustar el número de servidores trabajando en un sistema de colas M/M/2. Se demostró que una política o regla óptima se caracteriza por tener 4 parámetros: R1, R2, S0 y S1, denotando el número de clientes en el sistema cuando el número de servidores debe ser ajustado a 1, 2, 0 o 1 respectivamente.

Evans (1981) consideró el problema de determinar el valor del parámetro de diseño μ para maximizar un función de utilidad, la suma infinita de las recompensas esperadas. Se propuso un algoritmo iterativo que usaba valores aproximados para la función objetivo y gradientes aproximados en cada etapa. El algoritmo utiliza las cadenas de markov para la solución del problema. Trabajo numérico fue realizado en el mismo modelo de colas adoptado por Kumin, pero la

función objetivo y las restricciones eran otras. Más adelante se realizará una explicación mas detallada de este problema.

Kella (1990) considera un modelo M /G /1 donde los servidores toman descansos de acuerdo al siguiente plan: al volver del descanso (i – 1), si el sistema se encuentra vacío, el servidor decide tomar otro descanso con una probabilidad pi o permanece en el sistema con una probabilidad 1 – pi. Si al regreso de su descanso, un servidor encuentra clientes esperando, enseguida comienza a prestar el servicio. Usando el criterio del costo promedio esperado en el largo plazo con: un costo lineal asociado al servicio, costos fijos de alistamiento, y una función objetivo lineal cóncava por estar en descanso; Kella encuentra una secuencia $\{ pi \}_{i=1}$ ($0 \leq pi \leq 1$) que minimiza el costo promedio de funcionamiento del sistema en el largo plazo.

Martin y Artalejo (1997) consideran una cola M/G/1 con dos tipos de clientes: absolutamente impacientes y no impacientes. Si un cliente absolutamente impaciente encuentra un servidor ocupado enseguida abandona el sistema. Si un cliente no impaciente encuentra un servidor ocupado, se une a la cola y espera a ser servido después. El cliente en la cabeza de la cola espera ser atendido después de cierto tiempo con distribución exponencial, y parámetro $\mu > 0$. Martin y Artalejo desarrollan un análisis exhaustivo de este sistema y luego discuten la optimización del parámetro μ.

Selim (1997) considera un sistema de colas con alto volumen de servicios $M/M^N/1/N$ con tamaños de lote iguales a la capacidad del sistema. Se obtienen distribuciones de probabilidad dependientes del tiempo para el número de clientes en el sistema de manera cerrada y luego se discute la optimización de la tasa de servicio $\mu(t)$ en el intervalo $[0,T]$. Se considera la medida de rendimiento del sistema:

$$J(T,\mu) = \alpha \int_0^T p_N(t)dt + \beta \int_0^T \sum_{n=0}^{N} np_n(t)dt + \gamma \int_0^T \mu(t)dt$$

Donde α, B, **γ** son constantes positivas, y p_n (t) es la probabilidad de que hayan n clientes en el sistema en el tiempo t. El primer termino de la medida de rendimiento penaliza el sistema cada vez que el sistema esta lleno, los segundo y tercer términos son el número de unidades promedio en el sistema y el costo del servicio respectivamente. Para resolver este problema, Selim volvió discreto el horizonte del tiempo [0, T] en k intervalos de tamaño Δt = T/K y luego determino numéricamente la mejor tasa de servicio μ_k en cada intervalo ((k-1) Δt, k Δt)).

Ke y Wang(1999) consideran el problema de reparación de maquinas, donde las maquinas dañadas pueden abandonar la cola. Los servidores R también pueden fallar. La siguiente función lineal de costo es usada:

$$F(R) = c_1 L_q + c_2(L - L_q) + c_3 E[I] + c_4 E[B] + c_5 E[D] + c_L L.R$$

Donde el costo por unidad de tiempo $c_{i,}$ (i = 1,...5) y c_L son incurridos cuando una maquina dañada esta esperando por el servicio, cuando una maquina dañada se une al sistema y entra a servicio, cuando un servidor esta desocupado, cuando un servidor esta ocupado, cuando un servidor falla y cuando una maquina dañada abandona la cola, respectivamente. Las medidas desempeño del sistema usadas son el número promedio de maquinas dañadas en la cola Lq, el número promedio de maquinas dañadas en el sistema L, el número promedio de servidores desocupados en el sistema E[I], el número promedio de servidores ocupados en el sistema E[B], el número promedio de servidores que fallan E[D], y el promedio de maquinas dañadas que abandonaron la cola L.R. El número óptimo de servidores se determina numéricamente.

Grassman (2001) considera el problema de escoger la tasa óptima de servicio para un sistema de una cola con un solo servidor con llegadas dependientes de estados en un sistema M/G/1, dado que la tasa de servicio μ puede ser ajustada continuamente. Las probabilidades después de la partida esta ligadas a las probabilidades aleatorias de tiempo usando el teorema de Bayes. El *throughput* U(μ) y el número de unidades en el sistema L(μ) son derivados, y la siguiente función de utilidad es considerada:

$$f(\mu) = rU(\mu) - wL(\mu) - c(\mu)$$

Donde c(μ) es el costo por el servidor, w es el costo de esperar, y r es la utilidad por cada cliente servido. La tasa óptima de servicio es determinada numéricamente.

Qi-Ming y Neuts (2002) estudian un sistema formado por dos colas M/M/1 con transferencia de clientes. En este sistema cuando la diferencia entre el tamaño de las colas alcanzaba el valor L, un grupo de K clientes era transferido de la cola más larga a la más corta. Una matriz de transición describe el proceso markoviano de las dos colas. Usando resultados teóricos, se explora numéricamente el *diseño óptimo de estos sistemas de colas* en términos de tasas de transferencia de clientes, tamaño de grupo a transferir, throughput, tasas de servicio y tasas de llegadas. En particular se observa que un sistema de colas balanceado tiene una tasa de transferencia que es en lo posible la más pequeña.

Artalejo y Hernández-Lerma (2003) consideran la cola de tiempo discreto Geo/Geo/c donde las tasas de llegadas son p1 o p2 donde (p1>p2) y las tasas de servicio son q1 o q2 donde (q1<q2). Se consideran 2 problemas de diseño. Usando el criterio del costo descontado de horizonte infinito, se determinan las políticas estacionarias óptimas y los valores de las funciones. También se

comparan los dos problemas de diseño asumiendo parámetros iniciales p1 y q1, y se determina si es mejor cambiar la tasa de llegadas de p2 a p1 o modificar la tasa de servicio de q2 a q1.

Wuyi YUE (2004), considera el problema de diseño óptimo de un sistema de colas del tipo M(k)/M/k en el cual la tasa de llegadas de los usuarios depende del número de servidores. Se busca determinar el valor de k, el número de servidores para minimizar el costo total del sistema en estado estable C(k), que depende de H (k), el costo de espera en cola, S (k), el costo del servicio por usar k servidores y Lq(k) el número aproximado de clientes en el sistema. Donde el número aproximado de clientes en el sistema Lq(k) es una expresión cerrada.

Ausín Olivera (2004) , en su tesis doctoral “Análisis bayesiano de sistemas de colas”, proponen procedimientos bayesianos para la inferencia, predicción y diseño de diferentes sistemas de colas que incluyen modelos con distribuciones generales para el tiempo entre las llegadas y/o el tiempo de servicio, con uno o varios servidores y con capacidad finita e infinita. Se abordan también *problemas de diseño* en los que el objetivo consiste en decidir el número óptimo de servidores que minimizan una función de coste que depende de las distribuciones estacionarias estimadas.

Brill y Hlynka (2004) consideraron un problema estocástico de programación de tareas, los trabajos eran procesados como una cola M/M/c y se estableció un trabajo especial Cp teniendo como fecha de vencimiento a t*. Si Cp debe esperar, luego Cp eventualmente comienza servicio en algún momento con una probabilidad p, o Cp tendrá que esperar a que un servidor se desocupe. El valor óptimo de p se determina resolviendo el problema de optimización:

$$\begin{array}{ll} \text{minimize} & p \\ \text{subject to} & 1 - F_p(t^*) \leq \epsilon \\ & 0 \leq p \leq 1 \end{array}$$

Donde Fp es la función de distribución acumulativa del tiempo de espera Wp de un cliente especial Cp.

Kochel (2004) considera un sistema M/M/K/S y estudia las propiedades de primer orden (monotonía) y de segundo orden (convexidad / concavidad) de las diferentes medidas de rendimiento. Usando estas propiedades, un algoritmo simple para buscar los valores K y L = K – S que minimice la utilidad esperada por unidad de tiempo es propuesto:

$$G(K, L) = gH(K, L) - cK - wQ(K, L) - aR(K, L),$$

Donde g es la ganancia por unidad de tiempo para un servidor ocupado, c es el costo por unidad de tiempo y unidad de servicio, w es el costo por unidad de tiempo por trabajo en espera y a es el costo por trabajo rechazado. También H(K,L) es el número promedio de trabajos rechazados.

2.5 PROBLEMAS DE DISEÑO DE MAYOR RELEVANCIA PARA ESTE PROYECTO

En las siguientes secciones se realizará una descripción más profunda de las investigaciones sobre el diseño de sistemas de colas que tienen mayor relevancia con el problema abordado en este proyecto. Estos trabajos permitieron avanzar en la compresión de los problemas propuestos.

2.5.1 Problema de Diseño M/M/s por Kumin

Descripción del problema. Kumin(1968) formuló el siguiente problema de diseño de dos (2) parámetros asociado *un sistema de colas* M/M/s con las siguientes características:

- Una (1) cola, s número de servidores
- Las tasas de llegada λ y de servicio μ, tienen distribución de probabilidad exponencial.
- La disciplina de la cola es FCFS.

El problema de diseño consiste en hallar el valor óptimo de s y μ, tal que minimicen la función objetivo del costo total. Se considera el caso de un sistema de colas estándar con canales paralelos en el cual los clientes llegan al sistema con una tasa λ de distribución Poisson. En cada canal de servicio se tiene la misma tasa de servicio con parámetro μ. Cuando el cliente llega es servido si hay un canal de servicio disponible, de lo contrario, se une a la cola y espera.

La función objetivo a minimizar es

$$F.O: \quad Min\ f(s, \mu) = c_1 * s + c_2 * \mu + c_3 * En$$

$$Sujeta\ a: \quad \lambda / (s * \mu) < 1$$

$$\lambda > 0\ ,\ \mu > 0$$

Donde,

- c1 es el costo asociado al número de servidores en el sistema.
- c2, es el costo asociado a la tasa de servicio μ.
- c3, es el costo asociado al número esperado de clientes en el sistema.
- c1, c2 y c3 son constantes arbitrarias.

- μ, es la tasa de servicio y es un parámetro de diseño (variable de decisión continua)
- s, es el número de servidores en el sistema y es un parámetro de diseño (variable de decisión discreta).
- λ, es la tasa de llegadas, y se le dará un valor predeterminado.
- En, es el número esperado de clientes en el sistema de colas y es una expresión cerrada que depende de la tasa de servicio λ, μ y s.

Se conoce que el estado estable existe para este sistema si ρ = (λ / s * μ) < 1. Asumiendo que el número esperado de unidades o clientes en el sistema según Prabhu (1965) es:

$$E_n = \frac{P_0 * \left(\frac{\lambda}{\mu}\right)^j * \rho}{s! * (1-\rho)^2} \quad \text{Donde:} \quad P_0 = \left[\sum_{j=0}^{s-1} \frac{\left(\frac{\lambda}{\mu}\right)^j}{j!} + \frac{\left(\frac{\lambda}{\mu}\right)^s}{s!} * \frac{1}{1-\left(\frac{\lambda}{s\mu}\right)} \right]^{-1}$$

y $\rho = \frac{\lambda}{s\mu}$

Como se puede observar, inicialmente es necesario conocer el valor esperado para el número de clientes en el sistema E_n, y posteriormente incluirlo en la función objetivo asociándolo a un costo. Esta es una expresión cerrada de la teoría de colas tradicional. Es importante recordar la dificultad existente en hallar este tipo de expresiones.

Ejemplo de aplicación. Se consideró un ejemplo en el cual se definieron las siguientes condiciones iniciales:

- c1 = c2 = c3 = 1
- λ = 200

- μ y s son las variables de decisión
- μ > 0
- La minimización esta sujeta a la siguiente restricción: ρ = (λ / s * μ) < 1. Es decir, (200 / s) * μ < 1.

La función objetivo a minimizar es:

$$\textbf{F.O:} \quad \textbf{Min } f(s, \mu) = s + \mu + \frac{P_0 * \left(\frac{200}{\mu}\right)^j * \rho}{s! * (1-\rho)^2}$$

$$\textbf{Sujeta a:} \quad 200 / (s * \mu) < 1$$
$$\mu > 0$$

Descripción de la solución. Se realizó una investigación completa sobre el software disponible para solucionar este problema debido a la complejidad de la *función mixta entera no lineal* a minimizar. Esta función involucra una variable discreta y una continua, además posee sumatorias desde cero hasta s- 1 y varios cálculos con factorial de s. Esta búsqueda fue bastante difícil ya que la mayoría del software disponible no es capaz de manejar este tipo de optimización. Varios programas fueron encontrados como Matlab, Gams, Mosel Xpress, Excel Solver y Matcad. La mayoría pueden minimizar problemas no lineales, sin embargo presentan mucha dificultad para manejar problemas mixtos enteros que involucren sumatorias y factoriales.

Por lo anterior, se desarrolló un programa en Matlab, este software no es capaz de solucionar este problema directamente, pero con la función ***"fmincon"*** del Optimization Toolbox, se escribió un código que minimizaba la función hallando el valor μ para cada s desde s = 0 hasta s = 19, la combinación de s y μ que logra el menor valor de la función es considerado como el valor óptimo (ver *Anexo A.1.1*

Archivo Project1.m). Adicionalmente se crearon dos archivos de apoyo ".m" para poder incluir la restricción no lineal en la minimización (ver *Anexo A.1.3 Archivo nonlin.m*) y la expresión cerrada del número esperado de clientes en el sistema (ver *Anexo A.1.2 Archivo fun.m*)

Resultados numéricos experimentales. A continuación se presenta el valor de μ (tasa de servicio) obtenido para cada s (número de servidores), y el valor que toma la función objetivo para cada combinación de s y μ.

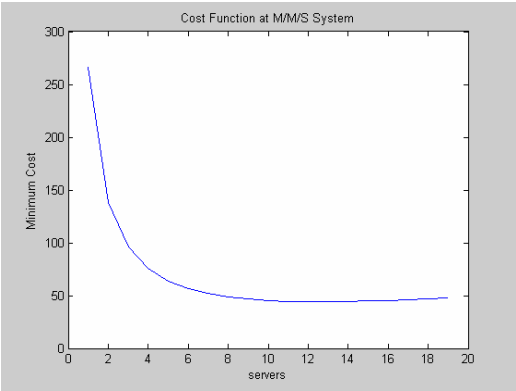
Tabla 2. Resultados numéricos - M/M/s

s	μ	f(μ)
1	243,14	266,46
2	123,50	138,31
3	83,17	96,07
4	62,87	75,60
5	50,63	63,91
6	42,45	56,65
7	36,61	51,93
8	32,22	48,81
9	28,82	46,75
10	26,11	45,44
11	23,91	44,67
12	22,09	44,31
13	**20,57**	**44,25**
14	19,29	44,45
15	18,21	44,83
16	17,30	45,36
17	16,53	46,01
18	15,90	46,76
19	15,38	47,59

Como se puede observar en la tabla 2 el valor mínimo de la función objetivo, f = 44.25, se obtiene cuando s = 13 y μ =20.57.

La función es convexa; cuando el número de servidores toma valores muy pequeños digamos 1 o 2, el costo total es muy alto, en la medida que el número de servidores va aumentando el costo se disminuye hasta el valor óptimo de 13 servidores, de ahí en adelante el costo empieza a aumentar poco a poco.

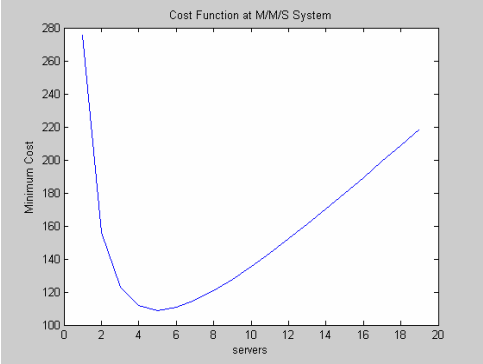
Figura 7. Función Convexa M/M/s



Modificando los valores de los costos: Partiendo del ejemplo anterior, los valores de c1, c2 y c3 se modificaron para investigar el comportamiento de la función y comparar estas soluciones con el resultado inicial.

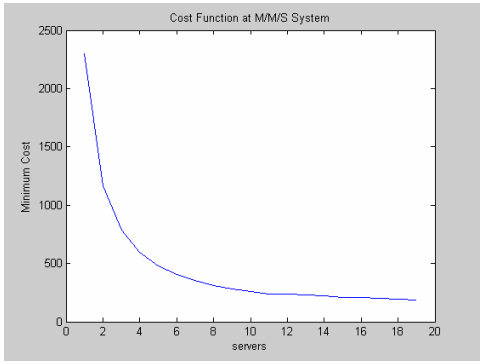
Primero el valor de c1 se cambió a 10 y los otros costos se mantuvieron igual a 1. Se espera que el valor óptimo se obtenga a un menor valor de s debido al castigo impuesto al número de servidores. El valor mínimo de la función objetivo, f = 108.911, se obtiene cuando s = 5 y μ =50.6321.

Figura 8. Cambio de c1 – M/M/s



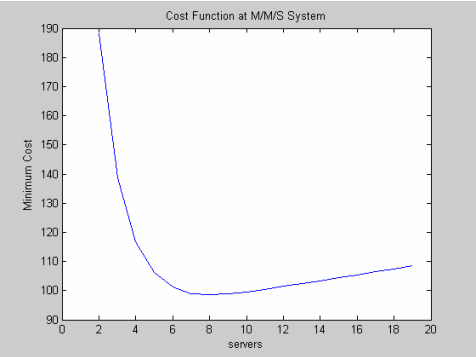
Luego el valor de c2 se cambió a 10 y los otros costos se mantuvieron igual a 1. Se espera que el valor óptimo se obtendrá a un menor valor de μ debido al castigo impuesto a la tasa de servicio. El valor mínimo de la función objetivo, f = 187.3533, se obtiene cuando s = 19 y μ = 15.5295.

Figura 9. Cambio de c2 – M/M/s



Luego el valor de c3 se cambió a 10 y los otros costos se mantuvieron igual a 1. Se obtiene una disminución en el número de servidores pero un incremento en la tasa de servicio. El valor mínimo de la función objetivo, f = 98.511, se obtiene cuando s = 8 y μ = 47.9423

Figura 10. Cambio de c3 – M/M/s



2.5.2 Problema de Diseño M/M/1 por Evans. Una búsqueda completa acerca de la literatura referente al diseño de sistemas de colas, indica que Kumin (1968) y Evans (1980) son los únicos que han utilizado las propiedades de las cadenas de Markov para optimizar sistemas de colas; usando un enfoque diferente al diseño óptimo tradicional.

Evans considera problemas de diseño de sistemas de colas usando cadenas de markov. Se asume que las cadenas del problema tienen matrices de transición, y recompensas esperadas por periodo, las cuales son funciones diferenciables de un parámetro vector. Se presenta un algoritmo de gradiente[1] para maximizar la suma de las recompensas esperadas descontadas o el límite de la recompensa esperada por periodo. El algoritmo usa valores aproximados para las funciones objetivo y gradientes[2] aproximados en cada etapa. Se realizó trabajo numérico en un modelo de colas simple unidimensional para resolver el problema de diseño correctamente.

[1] Los algoritmos de gradiente utilizan el gradiente de la función objetivo para calcular la ley recursiva de actualización del vector de paramentos estimados. Estos algoritmos iterativos parten de una solución inicial del vector parámetros, digamos θ_0, y van generando una secuencia finita de estimaciones θ_1, θ_2, ... θ_n, hasta obtener una aproximación suficientemente valida. (Ayesa, 2007)

[2] En muchas aplicaciones es deseable conocer en que dirección crece (o decrece) la función objetivo mas rápidamente; esta dirección se llama la dirección de máxima (o mínima) pendiente, y viene dada por el gradiente. (Larson, 1999)

El problema consiste en hallar el valor de los parámetros markovianos que maximicen las funciones objetivo consideradas. El algoritmo iterativo es necesario ya que estos parámetros generalmente no pueden ser resueltos explícitamente. El algoritmo utilizado converge a la solución óptima de estos problemas usando un ejemplo sencillo, pero importante: un modelo de tiempo discreto de una sistema de colas M/M/1. (Kumin, 1968; Morse, 1958)

Descripción de problemas. El primer problema considerado usa como función objetivo la suma de las recompensas esperadas descontadas:

$$\max_{\mu \in R} \quad \sum_{\mu \in R} \sum_{n=0}^{\infty} \beta^n P_0 T^n(\mu) V(\mu)$$

Donde:

μ es el parámetro de diseño y pertenece al conjunto R.

Po es el vector de m dimensiones de las probabilidades iniciales.

$T^n(\mu)$ es la matriz de transición m x m, función de μ.

V (μ) es el vector de m dimensiones, de las recompensas esperadas de un periodo.

B es el factor de descuento y 0 ≤ B< 1

Esta función objetivo será llamada la *función objetivo descontada* y el problema se llamara el *problema descontado.*

Usando los mismos símbolos, el segundo problema tiene como función objetivo el límite de la recompensa esperada por periodo:

$$\max_{\mu \in R} \ \{\lim_{n \to \infty} P_0 T^n(\mu) V(\mu)\}$$

Esta función objetivo será llamada la *función objetivo típica por periodo* y el problema se llamara el *problema típico por periodo.*

Es más sencillo trabajar con estas funciones objetivo de una manera alternativa usando un vector m dimensional llamado S(μ). Es importante recalcar que S(μ) es diferente para cada problema.

Para la función objetivo descontada, $S(\mu) = \sum_{n=0}^{\infty} \beta^n P_0 T^n(\mu)$, es el número de visitas a los estados del sistema, y $S(\mu)(I - \beta T(\mu)) = P_0.$

Para la función objetivo típica por periodo, $S(\mu) = \lim_{n\to\infty} P_0 T^n(\mu).$, son las probabilidades del estado limitante. Asumiendo que estos límites no dependen de Po. Para este caso S(μ) se define por

$$S(\mu)(I - T(\mu)) = 0$$
$$|S(\mu)| = 1$$

Donde I es un elemento neutro, 0 es el origen o vector de 0's y el vector normal $|\ |$ es la suma de valores absolutos de las coordenadas. Usando estos vectores, los problemas son:

$$\max_{\mu}(S(\mu),\ V(\mu))$$

Sujeto a:

$$S(\mu)(I - \beta T(\mu)) = P_0, \qquad \mu \in R$$

y

$$\max_{\mu}(S(\mu),\ V(\mu))$$

Sujeto a :

$$S(\mu)(I - T(\mu)) = 0, \qquad |S(\mu)| = 1, \qquad \mu \in R$$

De esta forma μ es un vector k dimensional, ambos problemas son problemas de programación no lineal con restricciones de igualdad no lineales.

Consideraciones especiales del problema. El modelo considerado utiliza la siguiente matriz de transición:

Figura 11. Matriz de Transición modelo Evans

$$T(\mu) = \begin{pmatrix} 1-\lambda & \lambda & 0 & \cdots & 0 & 0 \\ \mu & 1-\lambda-\mu & \lambda & \cdots & 0 & 0 \\ 0 & \mu & 1-\lambda-\mu & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1-\lambda-\mu & \lambda \\ 0 & 0 & 0 & \cdots & \mu & 1-\mu \end{pmatrix}$$

(Evans, 1980)

El parámetro λ puede ser considerado como la tasa de llegadas o como el número esperado de llegadas en un periodo. Similarmente, μ es la tasa de servicio o el número esperado de servicios completados en un periodo. Estas tasas son por unidad de tiempo en tiempo discreto.

Muchas variaciones de este modelo simple son posibles. Si otra escala de tiempo es usada, puede ser más útil asumir λ y μ como tasas, y λ Δt y μ Δt como probabilidades de transición donde Δt es un pequeño intervalo de tiempo. Este es el enfoque usual cuando el proceso de tiempo discreto se usa para derivar ecuaciones diferenciales de un modelo de tiempo continuo. Esta misma matriz de transición ocurre cuando el intervalo de tiempo discreto es el tiempo de espera

para la ocurrencia del siguiente evento, el cual puede ser una llegada, un servicio, o un evento de no cambio.

A pesar de que la cadena usada en esta investigación tiene diversas interpretaciones en colas, algunos lectores pueden preferir otras cadenas como modelos de tiempo discreto de procesos de congestión simples. En conjunto con la anterior matriz de transición el ejemplo incluye las siguientes suposiciones:

- La tasa de servicio μ es el único parámetro de diseño y el conjunto de los posibles valores R para μ es el intervalo [0 , 1 – λ]
- La recompensa es una ganancia de *g* por cada cliente servido (la ganancia esperada por periodo en el estado i es μ g δ(i) donde δ(i)= 0 si i = 0 y δ(i)= 1 si i > 0.
- El costo de congestión depende del estado inicial i del periodo y tiene la forma de $h(i) = a_1 i + a_2 i^2 + a_3 i^3$.
- El costo determinístico de usar la tasa de servicio μ para el periodo es

$$C(\mu) = \sum_{i=1}^{6} C_i \mu^i.$$

Combinando lo anterior, la recompensa esperada por empezar el periodo en el estado i es:

$$V(\mu, i) = \mu g \delta(i) - h(i) - C(\mu).$$

El vector de las recompensas esperadas por periodo *V (μ) es V (μ,0), V (μ,1),... V (μ, m-1)* un proceso de m estados.

Inclusive para este ejemplo sencillo, el método clásico de hallar formulas explicitas para *S(μ),* y luego hallar la derivada de la función objetivo e igualar a cero para hallar el valor óptimo, es muy complicado de realizar analíticamente. Es interesante notar como la escasez de formulas para parámetros óptimos de modelos de colas dificulta la solución de este tipo de problemas por medio del

análisis tradicional, el cual de ser aplicado a este ejemplo dependería de expresiones precisas de *T(μ) y V(μ).* Evans (1980)

El algoritmo presentado combina aproximaciones numéricas para *S(μ)* y sus derivadas con métodos de optimización.

Explicación del algoritmo utilizado. El enfoque usado por el algoritmo es combinar la búsqueda de gradiente de la programación no lineal y un procedimiento para hallar el valor de *S(μ).* La necesidad de calcular el gradiente en la programación no lineal, conlleva a extender el procedimiento para hallar la solución iterativa, incluyendo cálculos de derivadas parciales aproximadas de *S(μ).*

Más específicamente el algoritmo computa una secuencia de valores parámetros de diseño μ^{j} empezando por un valor arbitrario μ^{1}. El valor μ^{j+1} se obtiene moviéndose una distancia en la dirección del gradiente aproximado desde el valor μ^{j}. La aproximación para el vector $S(\mu^{j})$ es A_0^{j+1}. Asumiendo que μ tiene dimensión k, la aproximación del vector de las derivadas parciales de $S(\mu^{j})$ con respecto a la coordinada i de μ es el vector A_i^{j+1} para i = 1,...,k. Para hacer la notación más concisa, A^{j+1}, es el conjunto de vectores { $A_0, A_1, A_{2,...,} A_k$ }. El conjunto de vectores A^{j+1} es computado por una iteración empezando por A^{j}.

Un paso sencillo en el cálculo iterativo es definido por las funciones I^1 y J^1 para el problema descontado y el problema típico por periodo respectivamente. Para un valor dado de μ estas funciones generan un conjunto de vectores aproximantes A, dentro un nuevo conjunto de vectores aproximantes $I^1(\mu, A)$ y $J^1(\mu, A)$, respectivamente.

Usando I^1 para el problema descontado, para μ y el conjunto de vectores A = { $A_0, A_1, A_{2,...,} A_k$ }, el valor de I^1 es el conjunto de vectores:

$$I^1(\mu, A) = \{P_0 + \beta A_0 T(\mu), \beta(A_1 T(\mu) + A_0 T_1(\mu)), \cdots, \beta(A_k T(\mu) + A_0 T_k(\mu))\}$$

$T_r(\mu)$ es la matriz de las derivadas parciales de las entradas en T (μ) con respecto a la coordenada r de μ. El proceso iterativo para este caso se define inductivamente por:

$$I^j(\mu, A) = I^1(\mu, I^{j-1}(\mu, A)) = I^{j-1}(\mu, I^1(\mu, A))$$

Usando J^1 para el problema típico por periodo. Su valor es:

$$J^1(\mu, A) = \{A_0 T(\mu), (A_1 T(\mu) + A_0 T(\mu)), \cdots, (A_k T(\mu) + A_0 T_k(\mu))\}$$

Del conjunto de vectores de A, la función objetivo aproximada se define como:

$$f(\mu, A) = (A_0, V(\mu))$$

y el gradiente aproximado es:

$$\nabla f(\mu, A) = (A_1, V(\mu)) + (A_0, V_1(\mu)), \cdots, (A_k, V(\mu)) + (A_0, V_k(\mu))$$

Donde $V_{r\,1}(\mu)$ es el vector de las derivadas parciales de las coordinadas de V(μ) con respecto a la coordinada r de μ.

La búsqueda iterativa de la programación no lineal consiste en moverse una distancia γ en la dirección del gradiente aproximado. Esta distancia γ se reduce a αγ ($0 < \alpha < 1$) si el estado j no produce un incremento en la función objetivo aproximada.

La exactitud de la aproximación de S(μ) y sus derivadas parciales contenidas en el conjunto de vectores $A^{j+1} = \{ A_0^{j+1}, A_1^{j+1}, \ldots, A_k^{j+1} \}$ es controlado por una iteración interna N_j veces. Específicamente el calculo produce secuencias de

valores parámetros { μ^j } y secuencias { A^j } de conjuntos de vectores k + 1 de acuerdo a:

$$\mu^{j+1} = \mu^j + \{\gamma/|\nabla f(\mu^j, A^j)|\}\nabla f(\mu^j, A^j),$$

$$A^{j+1} = I^{N_j}(\mu^{j+1}, A^j) \quad \text{or} \quad J^{N_j}(\mu^{j+1}, A^j)$$

Este algoritmo de aproximación procede como sigue:

(i) Si $f(\mu^{j+1}, A^{j+1}) \geq f(\mu^j, A^j)$ parar o proceder a los cálculos de la fase j + 1.
(ii) Si $f(\mu^{j+1}, A^{j+1}) < f(\mu^j, A^j)$ parar o reemplazar γ por (αγ) y proceder a los cálculos de la fase j + 1.

Para ejecutar este algoritmo se necesita de: un regla para detenerse, valores para N_j , valores iniciales para γ y α, y condiciones iniciales para μ^1 y A^1 .Para el problema de descuento, la elección de A^1 no es critica, pero $A_0{}^1 \geq 0$, donde 0 es el origen y $| A_0 | \leq 1/ (1 – B)$. Para el problema típico por periodo ≥ 0, y $| A_0 | = 1$ y $(A_r{}^1 , E) = 0$, donde E es el vector de dimensiones m donde todos sus elementos son 1.

Resultados Experimentales. Varios cálculos numéricos fueron realizados para este ejemplo. La mayoría de estos consideraron principalmente la función objetivo descontada. Para este caso el algoritmo usaba $A_0{}^1 = P_0$, $A_1{}^1 = 0$, y $N_j = 1$. Los valores iniciales usados para γ y α, fueron 0.1 y 0.5 respectivamente. Para detener el algoritmo se requiere del cumplimiento de las siguientes desigualdades:

$$\nabla f(\mu^j, A^j) < 0.001$$

$$|A_0^{j+1} - A_0{}^j| < 0.001$$

$$|A_1^{j+1} - A_1{}^j| < 0.001$$

En la mayoría de los casos, los experimentos convergieron en entre 40 y 60 iteraciones.

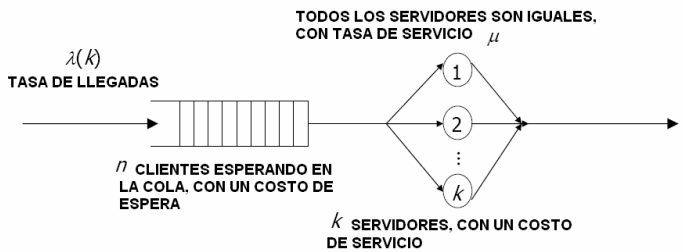
2.5.3 Problema de Diseño M(k)/M/k por Wuyi YUE. La teoría de colas es una herramienta fundamental en el diseño y dimensionado de redes de comunicación. Esta permite el análisis de los fenómenos que ocurren cuando muchos usuarios (ordenadores) están compitiendo por el mismo servicio (Internet). El diseño óptimo de estos sistemas permite la optimización de los recursos disponibles y permite entender como se genera o evita la congestión de los sistemas de comunicación. (Mandjes, 2002)

Wuyi YUE (2004) aborda este problema real considerando el diseño de un sistema de colas del tipo M(k)/M/k, en el cual la tasa de llegadas de los usuarios depende del número de servidores. Este estudio fue realizado con el propósito de analizar el desempeño de redes de comunicación inalámbricas, cuyos sistemas exhiben las propiedades del sistema de colas analizado.

Descripción del Problema. Se estudia un sistema de colas del tipo M(k)/M/k, con las siguientes características:

- Una (1) cola
- k servidores
- Las tasas de llegada λ y de servicio μ, tienen distribución de probabilidad exponencial. λ, depende del número de servidores k.

Figura 12. Sistema de Colas M(k)/M/k



(Wuyi YUE, 2004)

El problema consiste en hallar el valor del parámetro k, numero de servidores, que minimice una función de costos que depende de: el costo de espera en cola H (k) y del costo del servicio por usar k servidores S (k). H (k) a su vez depende del número aproximado de clientes en el sistema Lq(k).

La función objetivo a minimizar es:

F.O: Min { C (k) / k = 1, 2, 3….}

Min { C(k) = S(k) + H(k) = S(k) + h* Lq(k)}

Donde,

k es el número de servidores (variable de decisión)

λ(k) es la tasa de llegadas, función de k.

μ es la tasa de servicio para cada servidor.

ρ(k)=λ(k)/(kμ)<1 es la intensidad del trafico.

h(n,k) es el costo por esperar cuando hay clientes en la cola.

C (k) es el costo total esperado para el estado estable

H (k) es el costo de espera en cola

S (k) es el costo del servicio por usar k servidores

Lq(k) es el número aproximado de clientes en el sistema

Donde el número aproximado de clientes en el sistema Lq(k) es una expresión cerrada:

$$Lq(k) = \sum_{n=0}^{\infty} \pi_n(k) \cdot (n-k)^+ = \frac{k^k \rho^{k+1}(k)}{k![1-\rho(k)]^2} \cdot \pi_0(k)$$

,

que depende de las probabilidades de estado estable para n clientes en el sistema:

$$\pi_n(k) = \begin{cases} \frac{1}{n!} k^n \rho^n(k) \pi_0(k), & 1 \le n \le k \\ \frac{1}{k!} k^k \rho^n(k) \pi_0(k), & n \ge k \end{cases}$$

Donde:

$$\pi_0(k) = \left[\sum_{i=0}^{k-1} \frac{k^i \rho^i(k)}{i!} + \frac{k^k \rho^k(k)}{k!} \cdot \frac{1}{1-\rho(k)} \right]^{-1}$$

El costo de espera en cola H (k) también depende de las probabilidades de estado estable y se halla de la siguiente manera:

$$H(k) = \sum_{n=0}^{\infty} \pi_n(k) h(n,k) = \sum_{n=0}^{\infty} \pi_n(k) \cdot h \cdot (n-k)^+ = h \cdot Lq(k)$$

La información consultada es bastante limitada con respecto a los métodos utilizados para hallar la solución del problema. Sin embargo es un ejemplo práctico que permite ilustrar el diseño óptimo de un sistema de colas usando métodos tradicionales. También se puede observar la necesidad limitante de conocer y emplear expresiones cerradas para las medidas de desempeño del sistema en estado estable e incorporarlas al análisis del modelo.

2.6 HERRAMIENTAS INFORMÁTICAS DE MATLAB UTILIZADAS

En esta sección se describen brevemente las herramientas informáticas de apoyo empleadas para el desarrollo del proyecto.

2.6.1 El programa MATLAB. MATLAB es el nombre abreviado de “MATrix LABoratory”. MATLAB es un programa para el desarrollo de algoritmos, visualización de datos, análisis de datos y computación numérica, que utiliza un lenguaje técnico de alto nivel capaz de resolver problemas más rápidamente que otros como C, C++ y Fortran, entre otros. Es una magnífica herramienta para desarrollar aplicaciones técnicas, fácil de utilizar, y posee un lenguaje de programación propio.

El lenguaje utilizado por MATLAB esta soportado por operaciones de tipo matricial y vectorial, fundamentales para resolver problemas de ingeniería, que eliminan en muchos casos la necesidad de estructuras de bucle y que permiten la programación y ejecución rápida de algoritmos sin necesidad de llevar a cabo actividades administrativas de bajo nivel como declaración de variables y especificación de tipos de datos.

MATLAB posee un entorno especial para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares −tanto reales como complejos−, con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones.

2.6.2 M – files. MATLAB provee un lenguaje de programación que permite escribir una serie de instrucciones en un archivo y luego ejecutar estas instrucciones con un solo comando. Se puede escribir el programa en un archivo de texto ordinario, dándole el nombre de archivo.m. El término usado para este archivo (.m) se convierte en el nuevo comando que MATLAB asocia con ese programa. La extensión del archivo ".m" hace que este sea un M – file. Los M – files pueden ser escritos que simplemente ejecutan un serie de instrucciones de MATLAB, o también pueden ser funciones que también aceptan argumentos de entrada y producir un output.

2.6.3 Herramientas del *Optimization Toolbox* de MATLAB. El Toolbox - *paquete de herramientas* - de Optimización es una colección de funciones que extiende la capacidad del entorno numérico computacional de MATLAB. Todas las funciones del paquete son archivos ".m" de Matlab , que implementan algoritmos de optimización especializados. Este paquete incluye una serie de rutinas para varios tipos de optimización incluyendo: Minimización no-lineal con

restricciones, Minimización no-lineal sin restricciones, Programación cuadrática y lineal, y solución de sistemas de ecuaciones no lineales

Función *fmincon.* Es una función del paquete de optimización de MATLAB, esta se encarga de hallar el mínimo de una función no lineal multivariable con restricciones.

Figura 13. Función *fmincon* de MATLAB

$$\min_x f(x) \quad \text{suj}$$
$$c(x) \le 0$$
$$ceq(x) = 0$$
$$A \cdot x \le b$$
$$Aeq \cdot x = beq$$
$$lb \le x \le ub$$

Donde x, b, beq, lb, y ub son vectores, A y Aeq son matrices, c(x) y ceq(x) son funciones que devuelven vectores y f(x) es una función que devuelve un escalar. f(x), c(x), y ceq(x) pueden ser funciones no lineales.

La función *fmicon* se ejecuta de acuerdo a la siguiente sintaxis:

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlin)

MATLAB 6.5, Mathworks Inc.

Se encarga de minimizar la función *fun* contenida en el archivo fun.m; y la minimización esta sujeta a las inigualdades c(x) o igualdades ceq(x), definidos en el archivo nonlin.m. *fmincon* optimiza de tal forma que c(x) <= 0 y ceq(x) = 0 y determina lb=[] y/o ub=[] si no existen límites inferior y superior para la variable x.

3. DISEÑO ÓPTIMO DE SISTEMAS DE COLAS USANDO CADENAS DE MARKOV

Se ha podido observar que a lo largo de este trabajo la mayor parte de las investigaciones realizadas utilizan un enfoque de diseño óptimo tradicional, en el cual se parte de la necesidad de hallar expresiones cerradas para los valores esperados de congestión del sistema. Posteriormente a partir de estas expresiones se crea una función objetivo para ser optimizada. Sin embargo, es bien conocida la dificultad existente para hallar expresiones cerradas para funciones definidas con elementos del conjunto de las probabilidades de estado estable y generalmente estas expresiones por su complejidad, generan bastante dificultad para optimizar la función objetivo resultante.

A diferencia de usar expresiones cerradas para los valores esperados de congestión en el sistema, Kumin (1968) propone una teoría que utiliza las propiedades de las cadenas de markov para hallar estos valores. En esta investigación "el numero esperado de unidades en el sistema" será hallado a partir de las matrices de transición del modelo. Una serie de ejemplos numéricos son propuestos y se solucionan codificándolos con el software Matlab.

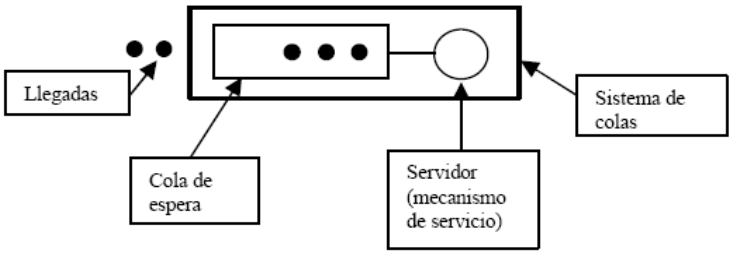
3.1 DESCRIPCIÓN DEL PROBLEMA

Se estudiará un *modelo de tiempo discreto de un sistema de colas M/M/1 (single server queue)* con las siguientes características:

- Una (1) cola
- Un (1) servidor

- Las tasas de llegada λ y de servicio μ, tienen distribución de probabilidad exponencial.
- La disciplina de la cola es FCFS.

Figura 14. Sistema de colas M/M/1



(Serra, 2002)

El problema consiste en hallar el valor del parámetro μ, tasa de servicio, que minimice una función de costos que depende: del costo asociado a la tasa de servicio *c1* y del costo asociado al número esperado de clientes en el sistema *c2*. Este número esperado de clientes a su vez depende de la tasa de servicio. Se dice que al hallar el valor de μ que minimice la función de costos, este valor se establece como parámetro, y por lo tanto, se diseña un sistema.

La función objetivo a minimizar (optimizar) es:

$$F.O: \quad Min_{\mu}\, g(\mu) = c_1 \mu + c_2 E(\mu, \lambda)$$

$$Sujeta\ a: \quad \lambda / \mu \leq 1$$

$$0 < \lambda < 1, \quad 0 < \mu < 1, \quad 0 \leq a_{ij} \leq 1$$

Donde,

- c1, es el costo asociado a la tasa de servicio μ.
- c2, es el costo asociado al número esperado de clientes en el sistema.
- μ, es la tasa de servicio y la variable de decisión.
- λ, es la tasa de llegadas, y se le dará un valor predeterminado.

- E(μ, λ), es el número esperado de clientes en el sistema de colas y es una expresión que depende de la tasa de servicio μ y de λ.

E(μ), será hallado usando las propiedades de las matrices de transición asociadas con ciertas cadenas de markov (matriz tridiagonal y hessenberg superior), *sin necesidad de una expresión cerrada*. Estas cadenas de markov provienen de modelos de colas elementales de tiempo discreto.

3.2 CONSIDERACIONES ESPECIALES DEL PROBLEMA

Matrices de transición utilizadas (generadores infinitesimales):

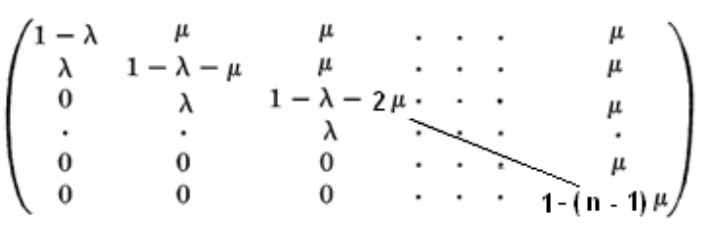
Según Kumin (1968), si consideremos un sistema de colas M/M/1 de cola finita, sabemos que el número esperado de clientes en el sistema en el tiempo t para este sistema es un proceso markoviano de tiempo continuo. Ahora, de estos procesos de tiempo continuo, pueden ser derivados varias cadenas de markov de tiempo discreto. Por ejemplo, si el proceso anterior es regido por su matriz de transición P(t) para t>0, y observamos este proceso en ciertos intervalos de tiempo, obtenemos una cadena de markov de tiempo discreto. También, si solo consideramos esos puntos en el tiempo donde las transiciones ocurren y definimos un nuevo evento llamado “no cambio”, podemos derivar cadenas de markov regidas por matrices de transición como las siguientes:

Figura 15. Matriz Tridiagonal

$$\begin{pmatrix} 1-\lambda & \mu & 0 & \cdots & 0 & 0 \\ \lambda & 1-\lambda-\mu & \mu & \cdots & 0 & 0 \\ 0 & \lambda & 1-\lambda-\mu & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1-\lambda-\mu & \mu \\ 0 & 0 & 0 & \cdots & \lambda & 1-\mu \end{pmatrix}$$

(Kumin, 1968)

Figura 16. Matriz Hessenberg Superior



(Kumin, 1968)

Evans(1963) demostró que el comportamiento límite de la matriz tridiagonal como modelo de tiempo discreto del M/M/1 es igual al comportamiento límite del proceso continuo “número esperado de clientes en el sistema” para el sistema de colas M/M/1 de cola finita.

Kumin (1968) desarrollo dos algoritmos para resolver problemas de diseño, cuyas funciones objetivo son función de matrices hessenberg superior y tridiagonal.

3.3 EXPLICACIÓN DEL ALGORITMO 1

Considerar el siguiente problema de diseño:

$$F.O: \quad Min_{\mu}\, g(\mu) = c_1 \mu + c_2 E(\mu, \lambda)$$

El número esperado de clientes en el sistema se desglosa de la siguiente manera:

$$E(\mu) = (F, X)$$

E(μ) = (F, X) donde, F es un vector creciente, F = (0,1,..,n-1), donde n representa los estados del sistema y X, satisface AX = X, y son las probabilidades absolutas.

A es alguna de las matrices de transición nxn, modelos de tiempo discreto (tridiagonal o hessenberg superior) asociadas con cadenas de markov.

Según Taha, si la cadena es ergódica, las probabilidades absolutas

$$\mathbf{X_n = X_0 A^z}$$

siempre convergen de forma única a una distribución limitante conforme a $z \rightarrow \infty$, donde la distribución limitante es independiente de las probabilidades iniciales.

Por eso, la función objetivo se reescribe así

$$g(\mu) = c_1 \mu + c_2 (F, X)$$
$$= c_1 \mu + c_2 (F, AX)$$
$$= c_1 \mu + c_2 \lim_{z \rightarrow \infty} (FA^z, P_0)$$ *Independiente de cualquier distribución inicial P0* .

Se puede observar como la función objetivo esta compuesta por dos partes: la primera parte, ($c_1 \mu$) es una función de los parámetros del sistema y la segunda $c_2 \lim_{z \rightarrow \infty} (FA^z, P_0)$ es una función de las probabilidades de estado estable del sistema (sin necesidad de una expresión cerrada).

Los parámetros n, número de estados, λ, la tasa de llegadas y z el exponente en FA^z serán modificados para probar los algoritmos en varios experimentos.

El algoritmo es el siguiente:

Permita que **F = (0,1,..,n-1)**. Seleccione un valor arbitrario para z. En la primera iteración seleccione un sistema inicial, digamos P^1

$P^1 =$

$$\begin{bmatrix} 1 \\ 0 \\ . \\ . \\ 0 \end{bmatrix}$$

Hallar el valor de μ que minimice:

$$\min_{\mu} g_1(\mu) = [c_1 \mu + c_2 (FA^z, P^1)]$$

*Note que después de realizar la multiplicación "**$((FA^z) P^1)$**" donde **F** es un vector de 1 x n, **A** es una matriz de tamaño n x n y es función de μ y **P^1** con tamaño n x1, se obtiene como resultado un vector de 1x1. El cual es una expresión que depende solo de μ y es el número esperado de clientes en el sistema.*

El valor obtenido es → μ_1^z

Llame a este valor μ_1^z. Como z fue escogido desde un principio y permanece constante, dejamos el exponente (z) y escribimos $\mu_1^z = \mu_1$

Se debe notar que la matriz A es función de μ, es decir A = A(μ). Ahora asociaremos con μ_1, una matriz A_1; y llamaremos $A_1 = A(\mu_1)$. Actualizamos P por $P^2 = A_1 P^1$. La función g queda como:

$$\min_{\mu} g_2(\mu) = [c_1 \mu + c_2 (FA^z, P^2)]$$
$$= [c_1 \mu + c_2 (FA^z, A_1 P^1)]$$

El valor obtenido es → μ_2

Llame a este valor μ_2.

Asociar μ_2 con la matriz A, como $A_2 = A(\mu_2)$.

Actualizar **P por $P^3 = A_2 A_1 P^1$** . La función g queda como:

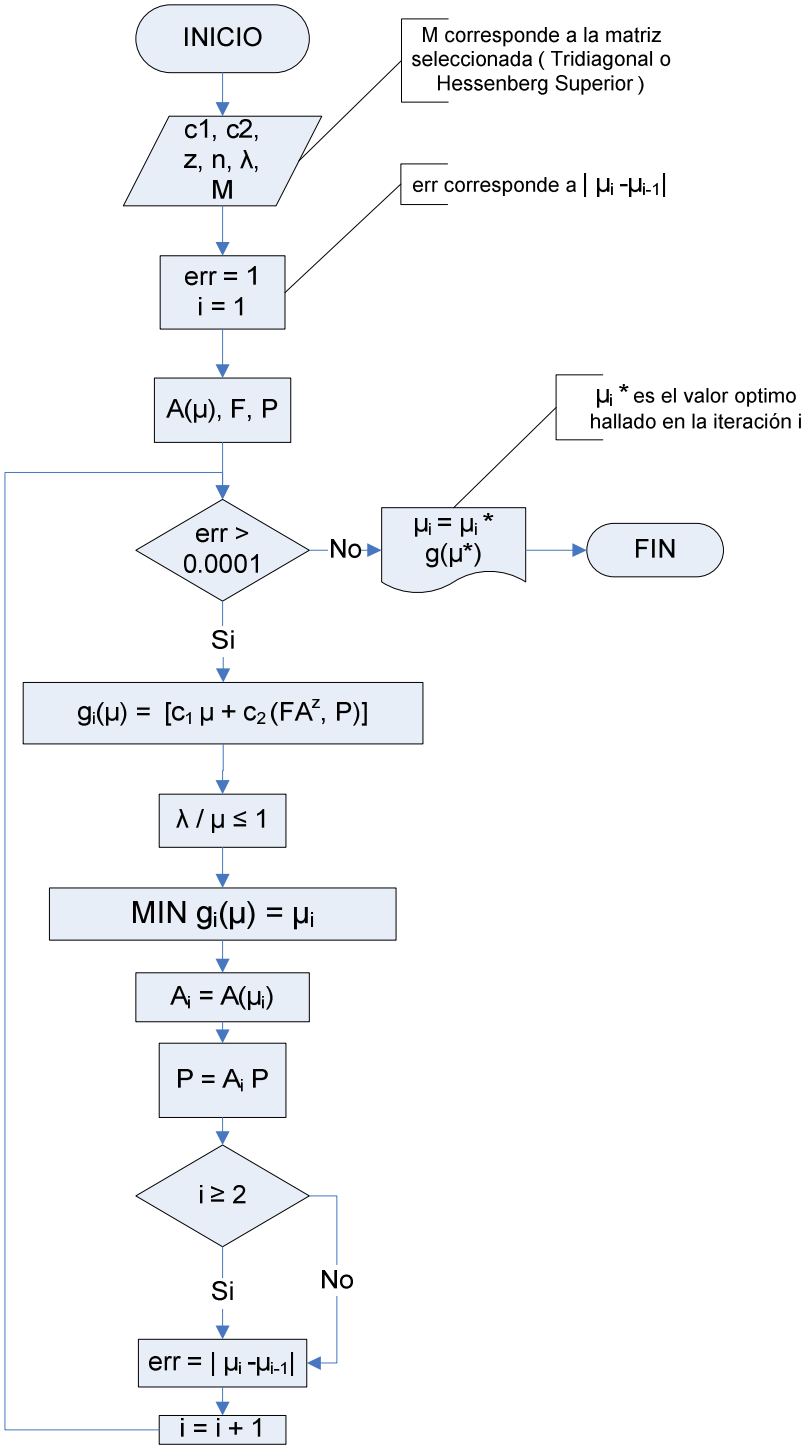
$$\min_{\mu} g_3(\mu) = [c_1 \mu + c_2 (FA^z, P^3)]$$
$$= [c_1 \mu + c_2 (FA^z, A_2 A_1 P^1)]$$

El valor obtenido es→ μ_3

El algoritmo continúa generando las siguientes secuencias: $\{ \mu_i \}_{i=1}^{\infty}$, $\{ A_i \}_{i=1}^{\infty}$, $\{ P^i \}_{i=1}^{\infty}$. *La secuencia* $\{ \mu_i \}_{i=1}^{\infty}$ *converge a* μ^*_i *y se obtiene uno* $g(\mu_i^*)$ *asociado. El criterio de convergencia seleccionado para los ensayos con el Algoritmo 1 es* $|\mu^*_i - \mu_{i-1}| < 0.0001$, *donde* μ^*_i *es el valor obtenido en la iteración i.*

3.3.1 Diagrama de flujo del Algoritmo 1

Figura 17. Diagrama de flujo del Algoritmo 1



3.3.2 Ejemplo general Algoritmo 1

Considerar las siguientes condiciones: z = 2, c1 = c2 = 1

Considerar una matriz de transición Tridiagonal de tres (3) estados

$A(\mu) =$ $P^1 =$

	1	2	3
1	1 – λ	μ	0
2	λ	1 – λ – μ	μ
3	0	λ	1 - μ

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$F =$

$$\begin{bmatrix} 0 & 1 & 2 \end{bmatrix}$$

Donde: $0< \lambda < 1$, $0< \mu < 1$, y $0 \leq a_{ij} \leq 1$

Iteración 1: $g_1(\mu) = [\mu + (FA^2, P^1)]$

$$= \mu + [2\lambda - \lambda\mu \quad 2\lambda\mu + (\mu+\lambda-1)^2 + 4\lambda(1-\lambda-\mu) + 2\lambda^2 \quad \lambda\mu - 2\mu + 2] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \mu + 2\lambda - \lambda\mu = 2\lambda + (1- \lambda)\mu$$

Claramente $\text{Min}_{\mu}\, g_1(\mu) = \text{Min}\ 2\lambda + (1- \lambda)\mu = 2\lambda$

Entonces $\mu_1 = 0$ y

$A_1 =$

$$\begin{bmatrix} 1-\lambda & 0 & 0 \\ \lambda & 1-\lambda & 0 \\ 0 & \lambda & 1 \end{bmatrix}$$

$P^2 =$

$$\begin{bmatrix} 1-\lambda & 0 & 0 \\ \lambda & 1-\lambda & 0 \\ 0 & \lambda & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$=$

$$\begin{bmatrix} 1-\lambda \\ \lambda \\ 0 \end{bmatrix}$$

Iteración 2: : $g_2(\mu) = [\mu + (FA^2, P^2)]$

$$= \mu + \begin{bmatrix} 1 - \lambda \\ 0 \\ 0 \end{bmatrix}$$

$$[2\lambda - \lambda\mu \quad 2\lambda\mu + (\mu+\lambda-1)^2 + 4\lambda(1-\lambda-\mu) + 2\lambda^2 \quad \lambda\mu - 2\mu + 2]$$

$$= \mu + 2\lambda(1-\lambda) - \lambda(1-\lambda)\mu + 2\lambda^2\mu + \lambda(\mu+\lambda-1)^2 + 4\lambda^2 (1-\lambda-\mu) + 2\lambda^3$$

$$\frac{dg_2(\mu)}{d(\mu)} = 1 - \lambda(1-\lambda) + 2\lambda^2 + 2\lambda(\mu+\lambda-1) - 4\lambda^2 = 0$$

$$\mu_2 = 3\lambda - \lambda^2 - 1 / 2\lambda$$

El algoritmo continúa generando las siguientes secuencias de la misma manera: $\{ \mu_i \}^{\infty}_{i=1}, \{ A_i \}^{\infty}_{i=1}, \{ P^i \}^{\infty}_{i=1}$.

3.3.3 Convergencia del Algoritmo 1. Según Kumin(1968), permitir que $\Psi = \{ \mu_i \mid \mu_i$ sea generada en la iteración i del Algoritmo 1, i = { 1,2,...} y permitir que $\theta = A_i \mid A_i = A(\mu_i)$, i = { 1,2,...}. Claramente se observa que a cada elemento x_i de Ψ, le corresponde un elemento $A(\mu_i)$ en θ.

Definición 1

Permitir a μ_i ser un elemento del conjunto Ψ. Luego la secuencia $\{ \mu_i \}^{\infty}_{i=1}$ converge a μ* si para todo ε > 0 existe un entero positivo K_ε tal que: $| \mu_r - \mu |$ para los $r \geq K_\varepsilon$.

Definición 2

Permitir $A_k = (a_{ij})_k$ ser un elemento del conjunto θ. Luego la secuencia de matrices $\{ A_i \}^{\infty}_{k=1}$ converge a la matriz A si por algún ε > 0 existe un entero positivo K_ε tal que:

$| (a_{ij})_r - (a_{ij}) | < \varepsilon$ para todos los $r \geq K_\varepsilon$ y todos los i, j = 1, 2,...n

Definición 3

Permitir $B_m = (b_{ij})_m = \prod_{s=0}^{m-1} A_{m-s}$ donde A_i es un elemento del conjunto θ. Luego Lim $B_m = (b_{ij})$ si por algún $\varepsilon > 0$ existe un entero positivo K_ε tal que:

$| (b_{ij})_r - (b_{ij}) | < \varepsilon$ para todos los $r \geq K_\varepsilon$

Para una explicación más detallada de estas definiciones, consultar "The design of markovian congestion systems"(Kumin, 1968).

3.4 EXPLICACIÓN DEL ALGORITMO 2

El Algoritmo 2 consiste en usar el Algoritmo 1 repetidamente para encontrar un valor más exacto de μ*.

Considerar: $\mathbf{g_z(\mu) = [c_1 \mu + c_2 (FA^z, P)]}$ y

$\mathbf{g(\mu) = [c_1 \mu + c_2 (F,X)]}$ donde F, A, P y X han sido previamente definidos.

Permitir que $\mathbf{\mu^*_z}$ y $\mathbf{\mu^*}$ cumplan:

$$\mathbf{g(\mu^*_z) = \min_\mu g_z(\mu)}$$

$$\mathbf{g(\mu^*) = \min_\mu g(\mu)}$$

Previamente se estableció la convergencia usando una z fija para el Algoritmo 1 es μ^*_z . Sin embargo es posible que para $\epsilon > 0$ y una z fija es posible que:

$$\mathbf{| \mu^*_z - \mu^* |} \not< \epsilon$$

Por tal razón el Algoritmo 1 no garantiza que se llega tan cerca como se quiere de **μ*,** entonces se propone usar el siguiente algoritmo que consiste en usar el

Algoritmo 1 repetidamente.

1. Seleccionar un valor inicial para z, llamarlo z_o
2. Usar el Algoritmo 1 para hallar $\mu^*_{z\,0}$
3. $z_i = z_i + k$ para i = 1, 2,..... y k puede ser cualquier entero positivo
4. Usar el Algoritmo 1 para hallar $\mu^*_{z\,i}$

Al usar el Algoritmo 1 repetidamente se genera la secuencia $\{ \mu^*_{z\,i} \}_{i=0}^{\infty}$

Cada iteración del Algoritmo 1 es considerada una subiteración del Algoritmo 2. μ^*_j *es el valor óptimo obtenido por el Algoritmo 1 en la iteración j del algoritmo No 2. El criterio de convergencia para los ensayos con el Algoritmo 2 es:*

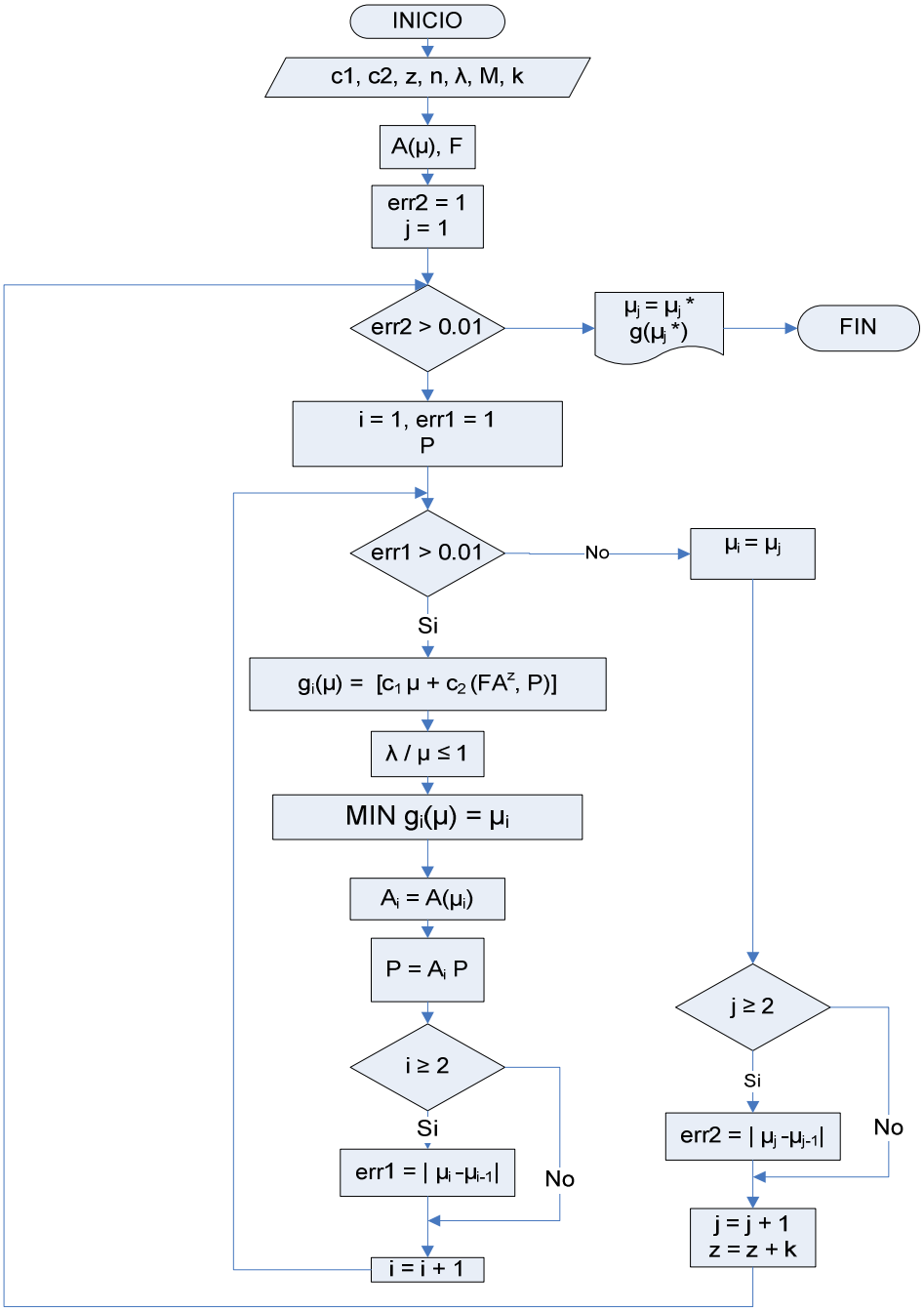
$$\left| \mu^*_i - \mu_{i-1} \right| < 0.01$$

y

$$\left| \mu^*_j - \mu_{j-1} \right| < 0.01$$

Diagrama de flujo del Algoritmo 2

Figura 18. Diagrama de flujo del Algoritmo 2



3.5 EXPERIMENTACIÓN PREELIMINAR

Después de lograr la compresión de los algoritmos de Kumin, se realizaron experimentos con las diferentes herramientas de MATLAB. Inicialmente no se contaba con un programa que ejecutara el algoritmo automáticamente, los ensayos se realizaban en la ventana de comandos del software manualmente, con el apoyo de dos archivos: *fun.m*, archivo donde se encontraba la función objetivo y *nonlin.m*, archivo donde se encontraba definida la restricción no lineal.

La función *fmicon* era ejecutada desde la ventana de comandos, donde se realizaba la minimización; en la medida que se realizaban las iteraciones, se actualizaba el archivo fun.m, agregándole las matrices resultantes de cada iteración. Era un proceso largo y dispendioso ya que para algunos ejemplos había que crear un gran número matrices manualmente y simultáneamente modificar la función objetivo. Sin embargo, se pudo observar como el algoritmo convergía para un reducido número de experimentos realizados, y por primera vez se realizaron ensayos con problemas de gran tamaño. Teniendo en cuenta estos resultados, se estableció la necesidad de diseñar e implementar un programa que permitiera aplicar los algoritmos que son objeto de estudio de este proyecto de forma automática.

A continuación se presentan dos problemas de menor dificultad para facilitar la comprensión de los algoritmos objeto de estudio de esta investigación.

3.5.1 Algoritmo 1. Para ilustrar el primer algoritmo se propone el siguiente ejemplo:

$$\textbf{Min}_{\mu}\ g_1(\mu) = [\mu + (FA^5, P^1)]$$

$$\textbf{Sujeta a: } 0.1 / \mu \leq 1$$

$$0 < \mu < 1,\ 0 \leq a_{ij} \leq 1$$

Donde:

- A es una matriz Tridiagonal
- c1 = c2 = 1
- λ = 0.1
- n = 3 estados y z = 5

$$A = \begin{array}{c|ccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{1} & \mathbf{1-0.1} & \boldsymbol{\mu} & \mathbf{0} \\ \mathbf{2} & \boldsymbol{\Lambda} & \mathbf{1-0.1-\mu} & \boldsymbol{\mu} \\ \mathbf{3} & \mathbf{0} & \mathbf{0.1} & \mathbf{1-\mu} \end{array} \qquad P^1 = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}$$

Iteración 1:

$$\mathbf{Min}_{\mu}\ \mathbf{g_1(\mu) = [c_1\,\mu + c_2\,(FA^5, P^1)]}$$

$$\mathbf{Min}_{\mu}\ \mathbf{g_1(\mu) = \mu +}$$

$$\begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{2} \end{bmatrix} \begin{bmatrix} \mathbf{0.9} & \boldsymbol{\mu} & \mathbf{0} \\ \mathbf{0.1} & \mathbf{0.9-\mu} & \boldsymbol{\mu} \\ \mathbf{0} & \mathbf{0.1} & \mathbf{1-\mu} \end{bmatrix}^5 \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

Es importante analizar el lado derecho de la función objetivo, nótese que después de realizar la multiplicación "**((FA^5) P^1)"** donde **F** es un vector de 1 x 3, **A^5** es una matriz de tamaño 3 x 3 y es función de μ y **P^1** con tamaño 3 x1, se obtiene como resultado un vector de 1x1. Este resultado es una expresión donde μ es la variable incógnita.

El valor a obtener es → μ_1

De esta minimización se obtiene que el valor mínimo de μ es 0.1. Es decir, $\mu_1 = 0.1$. A partir de este resultado se crea la matriz $A_1 = A(\mu_1)$.

$A_1 =$

$$\begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix}$$

Actualizar **P** por $\mathbf{P^2 = A_1 P^1}$

$P^2 =$

$$\begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Iteración 2

$\mathbf{\min_{\mu} g_2(\mu) = \mu + (FA^5, P^2)}$

$\mathbf{\min_{\mu} g_2(\mu) = \mu + (FA^5, A_1P^1)}$

$\mathbf{\min_{\mu} g_2(\mu) = \mu +}$

$$\begin{bmatrix} 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0.9 & \mu & 0 \\ 0.1 & 0.9 - \mu & \mu \\ 0 & 0.1 & 1 - \mu \end{bmatrix}^5 \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

El valor a obtener es → μ_2

De esta minimización se obtiene que el valor mínimo de μ es 0.103. Es decir,

$\mu_2 = 0.103$. A partir de este resultado se crea la matriz $A_2 = A(\mu_2)$.

$A_2 =$

$$\begin{bmatrix} 0.9 & 0.103 & 0 \\ 0.1 & 0.797 & 0.103 \\ 0 & 0.1 & 0.897 \end{bmatrix}$$

Actualizar **P por $P^3 = A_2 A_1 P^1$** .

$P^3 =$

$$\begin{bmatrix} 0.9 & 0.103 & 0 \\ 0.1 & 0.797 & 0.103 \\ 0 & 0.1 & 0.897 \end{bmatrix} \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Iteración 3

$\min_{\mu} g_3(\mu) = [c_1 \mu + c_2 (FA^5, P^3)]$

$\min_{\mu} g_3(\mu) = [c_1 \mu + c_2 (FA^5, A_2A_1P^1)]$

$\min_{\mu} g_3(\mu) = \mu +$

$$\begin{bmatrix} 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0.9 & \mu & 0 \\ 0.1 & 0.9 - \mu & \mu \\ 0 & 0.1 & 1 - \mu \end{bmatrix}^5 \begin{bmatrix} 0.9 & 0.103 & 0 \\ 0.1 & 0.797 & 0.103 \\ 0 & 0.1 & 0.897 \end{bmatrix} \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

El valor a obtener es → **μ_3**

De esta minimización se obtiene que el valor mínimo de μ es 0.1769. Es decir, $\mu_3 = 0.1769$. A partir de este resultado se crea la matriz $A_3 = A(\mu_3)$.

A_3 =

$$\begin{bmatrix} 0.9 & 0.1769 & 0 \\ 0.1 & 0.7231 & 0.1769 \\ 0 & 0.1 & 0.8231 \end{bmatrix}$$

Actualizar **P por $P^4 = A_3A_2A_1P^1$** .

P^4 =

$$\begin{bmatrix} 0.9 & 0.1769 & 0 \\ 0.1 & 0.7231 & 0.1769 \\ 0 & 0.1 & 0.8231 \end{bmatrix} \begin{bmatrix} 0.9 & 0.103 & 0 \\ 0.1 & 0.797 & 0.103 \\ 0 & 0.1 & 0.897 \end{bmatrix} \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Iteración 4

$\min_{\mu} g_4(\mu) = [c_1\mu + c_2(FA^5, P^4)]$

$\min_{\mu} g_4(\mu) = [c_1\mu + c_2(FA^5, A_3A_2A_1P^1)]$

$\min_{\mu} g_3(\mu) = \mu +$

$$\begin{bmatrix} 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0.9 & \mu & 0 \\ 0.1 & 0.9-\mu & \mu \\ 0 & 0.1 & 1-\mu \end{bmatrix}^5 \begin{bmatrix} 0.9 & 0.1769 & 0 \\ 0.1 & 0.7231 & 0.1769 \\ 0 & 0.1 & 0.8231 \end{bmatrix} \begin{bmatrix} 0.9 & 0.103 & 0 \\ 0.1 & 0.797 & 0.103 \\ 0 & 0.1 & 0.897 \end{bmatrix} \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

El valor a obtener es → **μ_4**

De esta minimización se obtiene que el valor mínimo de μ es 0.2183. Es decir, $\mu_4 = 0.2183$. A partir de este resultado se crea la matriz $A_4 = A(\mu_4)$.

Como se puede observar en la medida que se generan las secuencias $\{ \mu_i \}^{\infty}_{i=1}$ y $\{ A_i \}^{\infty}_{i=1}$ s, la función objetivo se va modificando de la siguiente manera:

$$\min_{\mu} g_4(\mu) = [c_1 \mu + c_2 (FA^5, A_j ..... A_5 A_4 A_3 A_2 A_1 P^1)]$$

El algoritmo continúa generando las siguientes secuencias: $\{ \mu_i \}^{\infty}_{i=1} = \{ \mu_1, \mu_2, \mu_3, ... \}$ y $\{ A_i \}^{\infty}_{i=1} = \{ A_1, A_2, A_3 ..... \}$, hasta que la secuencia $\{ \mu_i \}^{\infty}_{i=1}$ converge a μ^* y se obtiene uno $g(\mu^*)$ asociado.

Resultados numéricos. Se obtiene la secuencia de valores μ y $g(\mu)$, y el algoritmo converge a $\mu^*_{18} = 0.2888$ en la iteración 18.

Tabla 3. Resultados Algoritmo 1

ITERACIÓN	μ	g(μ)
1	0.1000	0.5095
2	0.1030	0.5685
3	0.1769	0.6130
4	0.2183	0.6400
5	0.2433	0.6570
6	0.2592	0.6679
7	0.2694	0.6750
8	0.2762	0.6797
9	0.2806	0.6827
10	0.2835	0.6847
11	0.2854	0.6860
12	0.2866	0.6868
13	0.2875	0.6874
14	0.2880	0.6878
15	0.2883	0.6880
16	0.2886	0.6881
17	0.2887	0.6882
18	0.2888	0.6883
	μ* = 0.2888	**g(μ*) = 0.6883**

En la tabla 3 se observan los valores correspondientes a μ y g(μ) obtenidos mediante la combinación de n = 3 estados y z = 5. Para realizar este experimento, el programa tuvo que realizar 18 iteraciones, es decir crear 18 matrices nuevas y actualizar la función objetivo simultáneamente como se explico en la sección anterior. El resultado final de este experimento es μ* = 0.2888 y g(μ*) = 0.6883. En el capitulo 4 se exponen los resultados finales de cada combinación de n y z, considerando distintos escenarios donde se varían el tipo de matriz, la estructura de costos y el valor de la tasa de llegadas.

3.5.2 Algoritmo 2. Para ilustrar el segundo algoritmo se propone el siguiente ejemplo:

$$\textbf{Min}_{\mu}\ g_1(\mu) = [\mu + (FA^z, P^1)]$$

$$\textbf{Sujeta a:} \qquad 0.1 / \mu \leq 1$$

$$0 < \mu < 1,\ 0 \leq a_{ij} \leq 1$$

Donde:

- A es una matriz Tridiagonal
- c1 = c2 = 1
- λ = 0.1
- $z_0 = 2$
- n = 3 estados y k = 6

La variable de decisión es μ.

El valor de z_i se modifica de acuerdo a k donde $z_i = z_i + k$. Tomando un valor inicial de $z_0 = 2$, entonces, $z_1 = z_0 + 6 = 8$; $z_2 = z_1 + 6 = 14$; y así sucesivamente.

Resultados numéricos

Se empezara con un $z_0 = 2$.

Se obtiene la secuencia de valores μ y g(μ).

Tabla 4. Resultados Algoritmo 2- Iteración 1

ITERACIÓN	μ	g(μ)
1	0.1000	0.2900
2	0.1000	0.3710
3	0.1000	0.4439
4	0.1000	0.5095
5	0.1000	0.5686
6	0.1000	0.6217
7	0.1000	0.6695
8	0.1000	0.7126
9	0.1000	0.7513
10	0.1000	0.7862
11	0.1000	0.8176
12	0.1000	0.8458
13	0.1134	0.8712
14	0.1420	0.8879
15	0.1431	0.8908
	μ^*_0= 0.1431	**g(μ^*_0) = 0.8908**

En la tabla 4. se observan los 15 resultados de cada iteración del algoritmo 1 o subiteración del algoritmo 2, y el resultado final de la primera iteración del algoritmo 2 es μ^*_0= 0.1431 y g(μ^*_0) = 0.8908. La siguiente iteración del algoritmo 2 consiste en experimentar con z = 8, considerando las mismas condiciones fijadas para el tipo de matriz, numero de estados, costos, λ y k.

Hacemos $z_1 = 2 + 6 = 8$

Se obtiene:

Tabla 5. Resultados Algoritmo 2- Iteración 2

ITERACIÓN	μ	g(μ)
1	0.2354	0.6337
2	0.2674	0.6484
3	0.2866	0.6580
4	0.2990	0.6646
5	0.3072	0.6690
	μ^*_1= 0.3072	$g(\mu^*_1)$ = 0.6690

Hacemos $z_2 = 8 + 6 = 14$

Tabla 6. Resultados Algoritmo 2- Iteración 3

ITERACIÓN	μ	g(μ)
1	0.3278	0.67317
2	0.33205	0.67459
	μ^*_2 = 0.33205	$g(\mu^*_2)$ = 0.67459

Hacemos $z_3 = 14 + 6 = 20$

Tabla 7. Resultados Algoritmo 2- Iteración 4

ITERACIÓN	μ	g(μ)
1	0.3415	0.6775
2	0.3423	0.6777
	μ^*_3 = 0.3423	$g(\mu^*_3)$ = 0.6777

Hacemos $z_4 = 20 + 6 = 26$

Tabla 8. Resultados Algoritmo 2- Iteración 5

ITERACIÓN	μ	g(μ)
1	0.3441	0.6782
2	0.3443	0.6782
	$\mu^*_4 = 0.3443$	$g(\mu^*_4) = 0.6782$

Como se cumple el criterio de convergencia:

$\left| \mu^*_4 = 0.3443 - \mu^*_3 = 0.3423 \right| < 0.01$, se detienen las iteraciones.

Para la obtención de la solución final, el algoritmo 2 tuvo que realizar 5 iteraciones sucesivas en la cuales se experimenta con z = 2, 8, 14, 20, 26. Después de comparar los resultados de cada iteración, se establece que los valores óptimos determinados por el algoritmo 2 cuando n = 3 y k = 6, son $\mu^*_4 = 0.3443$ y $g(\mu^*_4) = 0.6782$ en la quinta iteración. En el capitulo 4 se exponen los resultados finales de cada combinación de n y k, considerando distintos escenarios donde se varían el tipo de matriz, la estructura de costos y el valor de la tasa de llegadas.

3.5.3 Conclusiones Preeliminares. A partir de estos ensayos iniciales se plantearon las siguientes conclusiones preeliminares:

- Los algoritmos convergen a la solución óptima del problema de diseño planteado, sin necesidad de hallar una expresión cerrada para los valores esperados de congestión del sistema.
- La convergencia de los algoritmos depende de sus parámetros, pero principalmente del valor asignado al exponente (z) de la matriz A y del tamaño del sistema estudiado.
- Los algoritmos convergen a la solución óptima con un menor número de iteraciones asignando valores altos al exponente (z) de la matriz A.

- La aplicación de estos algoritmos o algoritmos similares soportados en los mismos principios permitiría aproximarnos a la solución de problemas más complejos.

4. IMPLEMENTACIÓN Y EXPERIMENTACIÓN

En este capitulo se presenta la solución de diferentes problemas de diseño, asociados al sistema M/M/1. Se diseñaron varios programas por medio del software de Optimización de Matlab de The MathWorks, Inc que permiten variar los diferentes parámetros de los algoritmos de Kumin. En este proyecto se aprovechan los sistemas actuales para realizar experimentos de gran complejidad que no se hubieran podido realizar en el momento en que se crearon los algoritmos.

4.1 ALGORITMO 1

Desarrollo de escenarios. Para validar este algoritmo se diseñaron varios escenarios en los cuales se quiere probar la convergencia del algoritmo considerando los siguientes aspectos:

- Dos (2) Diferentes matrices: Tridiagonal o Hessenberg Superior.
- Dos (2) Diferentes estructuras de costos del sistema: Costos iguales o Costos diferentes.
- Dos (2) Diferentes valores para la tasa de llegadas λ: 0.1 o 0.2.
- Variación de n, el número de estados del sistema.
- Variación de z, el exponente de A.
- El criterio de convergencia seleccionado para los ensayos con el algoritmo1 es $\left| \mu^*_i - \mu_{i-1} \right| < 0.0001$, donde μ^*_i es el valor obtenido en la iteración i.

Una variación en el tipo de matriz, el tamaño del sistema (número de estados n), la estructura de costos o la tasa de llegadas, hacen de cada problema un sistema modelado diferente bajo el cual se quiere probar el algoritmo, es decir, la combinación de estos factores permiten la creación de diferentes escenarios, en los cuales se estudia la convergencia del algoritmo.

Descripción del programa. Se desarrollo un programa, *ver Anexo A.2 DISEÑO SISTEMA M/M/1 – ALGORITMO 1* que permite aplicar el algoritmo 1; este programa recibe como datos de entrada:

- La matriz que se quiere usar (Tridiagonal o Hessenberg Superior)
- El valor de los costos c1 y c2
- El número de estados n
- El valor de λ
- El valor de z

Automáticamente el programa genera la función objetivo, y desarrolla todas las iteraciones necesarias para llegar a la solución del problema; este arroja como resultado:

- El valor óptimo de μ
- El valor mínimo de la función objetivo
- El tiempo en obtener la solución del problema

El criterio de convergencia seleccionado para los ensayos con el algoritmo 1 es $\left| \mu^*_i - \mu_{i-1} \right| < 0.0001$, donde μ^*_i es el valor obtenido en la iteración i.

Diseño Óptimo Tradicional para el sistema M/M/1 como parámetro de comparación. Según Tadj y Choudhury (2005), si consideramos el caso de un modelo M/M/1 con una tasa de llegadas λ conocida, la tasa de servicio es de μ; asumiendo que el costo de cada servidor por unidad de tiempo es Cs y el costo

asociado a cada cliente esperando es h, el problema de diseño consiste en hallar la tasa de servicio óptima que minimice el costo promedio por unidad de tiempo:

$$\mathbf{TC(\mu) = Cs * \mu + h * L}$$

Donde $L = \lambda / \mu - \lambda$ es el número esperado de clientes en el sistema en un M/M/1. Por eso:

$$\mathbf{TC(\mu) = Cs * \mu + h * (\lambda / \mu - \lambda)}$$

Entonces el valor para el cual $TC(\mu)$ alcanza su valor mínimo es μ^*; si se satisfacen las siguientes condiciones:

$$dTC(\mu) / d\mu \mid_{\mu = \mu^*} = 0 \quad y \quad d^2TC(\mu) / d\mu^2 \mid_{\mu = \mu^*} > 0$$

Al derivar $TC(\mu)$ e igualar a cero el valor óptimo de μ se encuentra cuando:

$$\mathbf{\mu^* = (\sqrt{(h * \lambda)/Cs}) + \lambda}$$

El problema anterior es el mismo problema de diseño de esta investigación, el diseño óptimo de un sistema M/M/1. Sin embargo se propone dar solución a este problema de manera tradicional. La expresión anterior es determinada para hallar la tasa óptima de servicio en un sistema M/M/1 usando expresiones cerradas.

Este método tradicional es bastante eficaz para encontrar la solución del problema de diseño de un sistema M/M/1 debido a la poca complejidad de la expresión para el número esperado de unidades en el sistema, y de la función objetivo resultante. Esta visto que para sistemas más complejos o de mayor tamaño, encontrar la expresión del número esperado es bastante complicado.

Los algoritmos de Kumin proponen un teoría alterna para el diseño de sistemas de colas usando cadenas de markov, sin necesidad de expresiones cerradas; por lo tanto en los capítulos siguientes se compararan lo resultados obtenidos mediante expresiones cerradas con los resultados obtenidos mediante los algoritmos de Kumin para el diseño de un sistema M/M/1.

De esta forma se validan los algoritmos de Kumin, y no solo se demuestra que estos convergen a un valor específico, como es planteado en la formulación de los algoritmos, si no que también convergen a la solución óptima del problema diseño. Una aplicación certera de este método alterno en el diseño de un sistema M/M/1 permitiría inferir su posible aplicación en sistemas más complejos.

Recordemos que Evans(1963) demostró que el comportamiento límite de la matriz tridiagonal como modelo de tiempo discreto del M/M/1 es igual al comportamiento límite del proceso continuo “número esperado de clientes en el sistema” para el sistema de colas M/M/1 de cola finita. Por tal razón se comparan los resultados obtenidos mediante el método tradicional (*Exp. Cerradas*) de diseño con aquellos conseguidos mediante la matriz tridiagonal y los algoritmos de Kumin. Sin embargo no se ha comprobado tal estrecha relación entre la matriz Hessenberg superior y el sistema M/M/1, por eso no se considera valido comparar los resultados al utilizar esta matriz con aquellos usando expresiones cerradas.

Experimentos Matriz Tridiagonal. Inicialmente se realizó una gran cantidad de experimentos en los cuales se implementa el algoritmo 1 (ver *Anexo B.1 EXPERIMENTOS ALGORITMO 1* para observar tablas de resultados); considerando las condiciones establecidas para cada escenario se varían los valores de n (el número de estados) y z (exponente de la matriz A) para estudiar la convergencia del algoritmo.

Para los escenarios de la matriz tridiagonal, se realizó un análisis más extenso mediante una comparación con el método tradicional de diseño óptimo (expresiones cerradas) y se analizan los dos factores principales del algoritmo. En las siguientes secciones los resultados numéricos de los experimentos se agrupan en 3 tablas:

Valores óptimos $\mu^*_{(algoritmo)}$ *encontrados*: esta tabla agrupa las tasas de servicio óptimas encontradas para cada combinación de z y n. Hay que considerar que cada casilla representa el resultado de un experimento independiente para el cual se hicieron un número considerable de iteraciones.

Valores $g(\mu^*_{(algoritmo)})$ *de la función objetivo*: esta tabla agrupa el costo total, es decir, el valor de la función objetivo obtenido para cada combinación de z y n. Estos valores están ligados estrechamente con la tabla de valores óptimos.

Desviaciones $|\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}|$*:* esta tabla agrupa el resultado de la desviación entre los valores óptimos encontrados mediante los algoritmos y aquellos obtenidos mediante expresiones cerradas (parámetros de comparación).

Experimentos Matriz Hessenberg Superior. Se llevo a cabo la misma metodología seguida con la matriz Tridiagonal para el desarrollo de los experimentos. Se definieron nuevos escenarios en los cuales se varían los valores de n (el número de estados) y z (exponente de la matriz A) para estudiar la convergencia del algoritmo. A pesar de que no existe un parámetro para comparar los resultados obtenidos, se pueden verificar las propiedades convergentes del algoritmo 1.

Los resultados numéricos de la experimentación se agrupan de manera similar en las 2 tablas: *Valores óptimos* $\mu^*_{(algoritmo)}$ *encontrados y Valores* $g(\mu^*_{(algoritmo)})$ *de la función objetivo.*

4.1.1 Escenario A1

Características

λ = 0,1

Matriz Tridiagonal

Costos iguales, C1 = C2 = 1

Experimentos realizados. Para cada escenario, se realizaron una serie de experimentos (ver *Anexo B.1 EXPERIMENTOS ALGORITMO 1*) en los cuales se estudia cada combinación de n y z.

A continuación se muestra una tabla de resultados para la combinación de n = 10 estados con los diferentes valores de z (ver *Anexo B.1.1 Experimentos Escenario A1*). Cada fila de la siguiente tabla es un experimento independiente. Se puede observar como el tiempo de ejecución es pequeño cuando el exponente z es alto, lo cual se debe a la realización de un menor número de iteraciones para obtener la solución del problema. Esta característica se mantiene para todos los escenarios de la matriz Tridiagonal.

Tabla 9. Experimentos n = 10 para A1

10 ESTADOS				
λ	**Z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	4,7	0,188	1,249
0,1	5	2,5	0,330	0,762
0,1	20	2,3	0,412	0,732
0,1	35	1,2	0,415	0,732
0,1	100	0,6	0,416	0,732

La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario:

Tabla 10. Valores óptimos μ* encontrados mediante el algoritmo 1 en A1

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,147	0,181	0,188	0,189	0,189
	5	0,289	0,327	0,33	0,33	0,33
	20	0,344	0,405	0,412	0,412	0,412
	35	0,345	0,407	0,415	0,415	0,415
	100	0,345	0,408	**0,416**	**0,416**	**0,416**

Tabla 11. Valores g(μ*) algoritmo 1 en A1

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,897	1,136	1,249	1,252	1,252
	5	0,688	0,753	0,762	0,762	0,762
	20	0,678	0,728	0,732	0,732	0,732
	35	0,678	0,728	0,732	0,732	0,732
	100	0,678	0,728	0,732	0,732	0,732

Se puede observar un comportamiento convergente de izquierda a derecha y de arriba abajo en las tablas 10 y 11, indicando una convergencia total en la esquina inferior derecha. Los valores subrayados corresponden a los valores óptimos de diseño para el sistema M/M1.

En la tabla 11 se observan valores de g(μ*) menores que los supuestos óptimos mínimos (los subrayados); es importante reconocer que estos valores son simplemente la solución de un experimento para unos determinados n y z, pero el criterio para hallar la solución al problema de diseño M/M/1 es ubicar las casillas de la tabla de donde se obtiene la convergencia total.

Las anteriores dos observaciones también se cumplen para los demás escenarios de la matriz Tridiagonal.

Desviaciones con respecto al parámetro de comparación. Usando la misma notación llevada en esta investigación, la expresión cerrada para obtener el valor óptimo de la tasa de servicio para un sistema M/M/1 es:

$$\mu^* = (\sqrt{(c2^* \lambda)/c1}) + \lambda$$

Recordemos que:
Si λ = 0.1, y c1=c2=1, podemos aplicar $\mu^* = (\sqrt{(c2^* \lambda)/c1}) + \lambda$ y hallar el valor óptimo de μ de manera tradicional, es decir:
$\mu^* = \sqrt{(1^* 0.1/1)} + 0.1 = \sqrt{0.1} + 0.1$
$\mu^* = 0.416$ *(Parámetro de comparación)*

Remplazando $\mu^* = 0.416$ en
$C = c1 * \mu + c2 * (\lambda / \mu - \lambda)$
$C = 0.732$

Si determinamos $| \mu^*_{(algoritmo)} - \mu^*_{(tradicional)} |$ para cada combinación de factores n y z obtenemos la siguiente tabla de desviaciones:

Tabla 12. Desviaciones $| \mu^*_{(algoritmo)} - \mu^*_{(tradicional)} |$ en A1

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,269	0,235	0,228	0,227	0,227
	5	0,127	0,089	0,086	0,086	0,086
	20	0,072	0,011	0,004	0,004	0,004
	35	0,071	0,009	0,001	0,001	0,001
	100	0,071	0,008	0	0	0

Se obtiene una desviación nula en el costado inferior derecho de la tabla. La desviación más alta obtenida es 0,269 cuando z = 2 y n = 3. Se puede observar que para obtener la solución óptima del problema de diseño hay que experimentar con los valores más altos de los factores del algoritmo.

4.1.2 Escenario A2

Características

λ = 0,2

Matriz Tridiagonal

Costos iguales, C1 = C2 = 1

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver *Anexo B.1.2 Experimentos Escenario A2*):

Tabla 13. Valores óptimos μ* encontrados mediante el algoritmo 1 en A2

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,267	0,345	0,365	0,365	0,365
	5	0,439	0,545	0,545	0,558	0,558
	20	0,473	0,617	0,644	0,644	0,644
	35	0,473	0,618	0,646	0,646	0,646
	100	0,473	0,618	**0,647**	**0,647**	**0,647**

Tabla 14. Valores g(μ*) algoritmo 1 en A2

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	1,077	1,373	1,539	1,547	1,547
	5	0,963	1,091	1,091	1,115	1,115
	20	0,96	1,079	1,094	1,094	1,094
	35	0,96	1,079	1,094	1,094	1,094
	100	0,96	1,079	1,094	1,094	1,094

Desviaciones con respecto al parámetro de comparación

Si λ = 0.2 y c1=c2=1,

$\mu^* = \sqrt{(1^* 0.2/1)} + 0.2 = \sqrt{0.2} + 0.2$

$\mu^* = 0.647$

Remplazando $\mu^* = 0.647$ en

$C = c1 * \mu + c2 * (\lambda / \mu - \lambda)$

C = 1,094

Si determinamos $| \mu^*_{(algoritmo)} - \mu^*_{(tradicional)} |$ para cada combinación de factores n y z obtenemos la siguiente tabla de desviaciones:

Tabla 15. Desviaciones $| \mu^*_{(algoritmo)} - \mu^*_{(tradicional)} |$ en A2

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,38	0,302	0,282	0,282	0,282
	5	0,208	0,102	0,102	0,089	0,089
	20	0,174	0,03	0,003	0,003	0,003
	35	0,174	0,029	0,001	0,001	0,001
	100	0,174	0,029	0	0	0

Se obtiene una desviación nula en el costado inferior derecho de la tabla. La desviación más alta obtenida es 0,38 cuando z = 2 y n = 3. Se puede observar que para obtener la solución óptima del problema de diseño hay que experimentar con los valores más altos de los factores del algoritmo.

4.1.3 Escenario B

Características

λ = 0,1

Matriz Tridiagonal

Costos iguales, C1 =1,4 C2 = 1

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario(ver *Anexo B.1.3 Experimentos Escenario B*):

Tabla 16. Valores óptimos μ* encontrados mediante el algoritmo 1 en B

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,219	0,12	0,134	0,134	0,135
	5	0,219	0,259	0,264	0,264	0,264
	20	0,287	0,351	0,359	0,359	0,359
	100	0,288	0,356	**0,367**	**0,367**	**0,367**

Tabla 17. Valores g(μ*) algoritmo 1 en B

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,831	1,802	2,322	2,361	2,365
	5	0,831	0,946	0,971	0,971	0,971
	20	0,804	0,880	0,888	0,888	0,888
	100	0,804	0,880	0,888	0,888	0,888

Desviaciones con respecto al parámetro de comparación

Si λ = 0.1, y c1=1.4, c2=1, podemos aplicar **μ* = (√(c2* λ)/c1) + λ** y hallar el valor óptimo de μ de manera tradicional, es decir:

μ* = (√(1* 0.1)/1.4) + 0.1

μ* = 0.367

Remplazando μ* = 0.367 en

C = c1 *μ + c2 *(λ / μ - λ)

C = 0,888

Si determinamos $| \mu^*_{(algoritmo)} - \mu^*_{(tradicional)} |$ para cada combinación de factores n y z obtenemos la siguiente tabla de desviaciones:

Tabla 18. Desviaciones $| \mu^*_{(algoritmo)} - \mu^*_{(tradicional)} |$ en B

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,148	0,247	0,233	0,233	0,232
	5	0,148	0,108	0,103	0,103	0,103
	20	0,08	0,016	0,008	0,008	0,008
	100	0,079	0,011	0	0	0

Se obtiene una desviación nula en el costado inferior derecho de la tabla. La desviación más alta obtenida es 0,148 cuando z = 2 y n = 3. Se puede observar que para obtener la solución óptima del problema de diseño hay que experimentar con los valores más altos de los factores del algoritmo.

4.1.4 Escenario C

Características

λ = 0,1

Matriz Tridiagonal

Costos iguales, C1 =1 C2 = 1,4

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver *Anexo B.1.4 Experimentos Escenario C*):

Tabla 19. Valores óptimos μ* encontrados mediante el algoritmo 1 en C

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,223	0,253	0,257	0,257	0,257
	5	0,367	0,404	0,406	0,406	0,406
	20	0,409	0,467	0,472	0,472	0,472
	100	0,41	0,468	**0,474**	**0,474**	**0,474**

Tabla 20. Valores g(μ*) algoritmo 1 en C

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,945	1,096	1,134	1,134	1,134
	5	0,804	0,858	0,862	0,862	0,862
	20	0,8	0,845	0,848	0,848	0,848
	100	0,8	0,845	0,848	0,848	0,848

Desviaciones con respecto al parámetro de comparación

Si λ = 0.1, y c1=1, c2=1.4, podemos aplicar **$\mu^* = (\sqrt{(c2^* \lambda)/c1) + \lambda}$** y hallar el valor óptimo de μ de manera tradicional, es decir:

$\mu^* = (\sqrt{(1.4^* 0.1/1)} + 0.1$

$\mu^* = 0.474$

Remplazando $\mu^* = 0.474$ en

$C = c1 * \mu + c2 * (\lambda / \mu - \lambda)$

$C = 0,848$

Si determinamos $| \mu^*_{(algoritmo)} - \mu^*_{(tradicional)} |$ para cada combinación de factores n y z obtenemos la siguiente tabla de desviaciones:

Tabla 21. Desviaciones | $\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}$ | en C

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,251	0,221	0,217	0,217	0,217
	5	0,107	0,07	0,068	0,068	0,068
	20	0,065	0,007	0,002	0,002	0,002
	100	0,064	0,006	0	0	0

Se obtiene una desviación nula en el costado inferior derecho de la tabla. La desviación más alta obtenida es 0,251 cuando z = 2 y n = 3. Se puede observar que para obtener la solución óptima del problema de diseño hay que experimentar con los valores más altos de los factores del algoritmo.

4.1.5 Observaciones generales Matriz Tridiagonal. Mediante la realización de estos experimentos se pudo observar que existe consistencia en las características de los resultados para los escenarios A1, A2, B y C, es decir, para aquellos que están basados en la matriz Tridiagonal. A continuación se explican estas observaciones:

Para cada combinación de n y z, cuando el valor de z es pequeño, hay que realizar un elevado número de iteraciones para obtener la convergencia del algoritmo. Cuando el valor de z es grande, hay que realizar un reducido número de iteraciones para obtener la convergencia del algoritmo.

Una misma combinación de parámetros genera respuestas idénticas, es decir, la variación entre las muestras tomadas para un mismo tratamiento (combinación específica de un n y un z) es nula. Si se realiza una combinación de un determinado n y un determinado z, siempre va dar el mismo valor para la tasa óptima de servicio.

Para un determinado n, en la medida que aumenta el valor del exponente z, el algoritmo converge a un valor específico. Para un determinado z, en la medida que aumenta el valor de n, el algoritmo converge a un valor específico. Entonces se puede observar un comportamiento convergente de izquierda a derecha y de arriba abajo en las tablas indicando una convergencia total en la esquina inferior derecha.

Es muy importante reconocer que en todos los escenarios considerados estudiados el algoritmo 1 se aproximo con cero desviación a la solución óptima tradicional del problema de diseño M/M/1 mediante la experimentación con un número de estados n mayor de 10 y un exponente de z igual a 100. Experimentos de este magnitud nunca se habían realizado antes y le dan validez a la aplicación de este algoritmo en el diseño del sistema M/M/1.

4.1.6 Escenario D

Características

λ = 0,1

Matriz Hessenberg Superior

Costos iguales, C1 = C2 = 1

Experimentos Realizados. A continuación se muestra una tabla de resultados para n = 5 estados, cada fila de la siguiente tabla es un experimento independiente (ver *Anexo B.1.5 Experimentos Escenario D*). Se puede observar como el tiempo de ejecución es menor cuando z toma un valor alto debido a la reducción en el número de iteraciones que hay que realizar para obtener la solución del problema. Esta característica se mantiene para todos los escenarios de la matriz Hessenberg Superior.

Tabla 22. Experimentos n = 5 para D

5 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,7	0,174	0,635
0,1	5	1,0	0,255	0,589
0,1	20	0,4	0,274	0,587
0,1	35	1,3	0,274	0,587
0,1	50	0,4	0,274	0,587
0,1	100	0,4	0,274	0,587

La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario:

Tabla 23. Valores óptimos μ* encontrados mediante el algoritmo 1 en D

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,16	0,174	0,175	0,175	0,175
	5	0,246	0,255	0,255	0,255	0,255
	20	0,266	0,274	0,274	0,274	0,274
	100	0,266	0,274	*	*	*

Tabla 24. Valores g(μ*) algoritmo 1 en D

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,637	0,635	0,635	0,635	0,635
	5	0,584	0,589	0,589	0,589	0,589
	20	0,582	0,587	0,587	0,587	0,587
	100	0,582	0,587	*	*	*

Las casillas marcadas con un asterisco representan los experimentos en los cuales la capacidad del programa fue insuficiente para hallar la solución del problema. Sin embargo, debido al comportamiento de los resultados en las filas

donde se solucionaron todos los experimentos, se puede inferir que la convergencia total se alcanza a obtener cuando n = 5 estados (es visible que para un determinado z, los valores óptimos son idénticos para los n > 5.)

4.1.7 Escenario E

Características

λ = 0,1

Matriz Hessenberg Superior

Costos iguales, C1,4 = C2 = 1

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver *Anexo B.1.6 Experimentos Escenario E*):

Tabla 25. Valores óptimos μ* encontrados mediante el algoritmo 1 en E

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,11	0,128	0,128	0,128	0,128
	5	0,191	0,202	0,202	0,202	0,202
	20	0,217	0,226	*	*	*
	100	0,217	0,226	*	*	*

Tabla 26. Valores g(μ*) algoritmo 1 en E

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,779	0,771	0,771	0,771	0,771
	5	0,682	0,69	0,69	0,69	0,69
	20	0,678	0,687	*	*	*
	100	0,678	0,687	*	*	*

4.1.8 Escenario F

Características

λ = 0,1

Matriz Hessenberg Superior

Costos iguales, C1 = 1 C2 = 1,4

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver *Anexo B.1.7 Experimentos Escenario F*):

Tabla 27. Valores óptimos μ* encontrados mediante el algoritmo 1 en F

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,223	0,235	0,235	0,235	0,235
	5	0,31	0,318	0,318	0,318	0,318
	20	0,324	*	*	*	*
	35	0,324	*	*	*	*

Tabla 28. Valores g(μ*) algoritmo 1 en F

		n, número de estados				
		3	**5**	**10**	**15**	**50**
z, exponente de la matriz A	**2**	0,736	0,736	0,736	0,736	0,736
	5	0,699	0,703	0,703	0,703	0,703
	20	0,698	*	*	*	*
	100	0,698	*	*	*	*

4.1.9 Observaciones generales Matriz Hessenberg Superior. Se puede observar que existe consistencia en el comportamiento de los resultados para los escenarios D, E y F.

Se contó con bastante dificultad para solucionar problemas de mayor tamaño, especialmente para aquellos problemas que involucran costos diferentes en su función objetivo(ver casillas marcadas con un asterisco *). En aquellos casos el software diseñado no tenía la capacidad computacional para solucionar estos problemas y después de varios minutos iterando se quedaba sin memoria para correr el algoritmo.

De manera similar a los experimentos con la matriz tridiagonal:

> Para cada combinación de n y z, cuando el valor de z es pequeño, hay que realizar un elevado número de iteraciones para obtener la convergencia del algoritmo. Cuando el valor de z es grande, hay que realizar un reducido número de iteraciones para obtener la convergencia del algoritmo.
>
> Una misma combinación de parámetros genera respuestas idénticas, es decir, la variación entre las muestras tomadas para un mismo tratamiento (combinación específica de un n y un z) es nula. Si se realiza una combinación de un determinado n y un determinado z, siempre va dar el mismo valor para la tasa óptima de servicio.

Para un determinado n, en la medida que aumenta el valor del exponente z, el algoritmo converge a un valor específico. Para un determinado z, en la medida que aumenta el valor de n, el algoritmo converge a un valor específico, pero mucho más rápido que con la matriz tridiagonal; se puede verificar como los resultados para un determinado z convergen a un valor después de n = 5, lo cual permite inferir que no es necesario realizar experimentos de gran tamaño ya que con los de menor tamaño se consigue llegar al punto óptimo con esta matriz. Se puede observar un comportamiento convergente de izquierda a derecha y de arriba abajo en las tablas.

Desafortunadamente no se cuenta con un modelo o un parámetro para comparar los resultados obtenidos con la matriz Hessenberg Superior, pero si se demuestra que el algoritmo consigue solucionar problemas de diseño cuyas funciones objetivo son función de cadenas de Markov, lo cual le da validez a la aplicación del algoritmo en problemas similares**.**

4.2 ALGORITMO 2

Para la validación de este algoritmo se utilizan los mismos escenarios usados para validar el algoritmo 1. Es importante recordar que el algoritmo 2 consiste en usar el algoritmo 1 repetidamente hasta encontrar un valor más óptimo para μ*, por lo tanto los dos algoritmos están estrechamente relacionados.

Cada iteración del algoritmo 1 es considerada una subiteración del algoritmo 2. μ^*_j es el valor óptimo obtenido por el algoritmo 1 en la iteración j del algoritmo 2. El criterio de convergencia para los ensayos con el algoritmo 2 es:

$$\left| \mu^*_i - \mu_{i-1} \right| < 0.01$$

y

$$\left| \mu^*_j - \mu_{j-1} \right| < 0.01$$

Se podrá observar como el algoritmo 2 depende en gran medida del algoritmo 1; como es sabido el algoritmo 2 inicia su operación fijando un $z_{0,}$ y usando el algoritmo 1 halla los valores óptimos para ese z, posteriormente incrementa el valor de z en k unidades($z_0 + k = z_1$) y para ese nuevo z_1 determina los valores óptimos correspondientes. El algoritmo 2 continua generando los diferentes z y sus respectivos valores óptimos, y luego se detiene cuando la diferencia entre los valores óptimos de dos iteraciones es menor a 0.01 es decir $\left| \mu^*_j - \mu_{j-1} \right| < 0.01$.

Para estos experimentos se agrega la variable k, que modifica el exponente z, y se utiliza en cada iteración del algoritmo 2. Este se probara con tres valores, uno pequeño (k =6), uno intermedio (k =15), uno grande (k=30), para ver si tiene alguna incidencia en la convergencia del modelo. Adicionalmente se limitan el número de niveles a probar para el número de estados del sistema. Se probara con un valor pequeño (n =3), uno intermedio (n=10) y una grande (n=50). z_0 el exponente inicial de A se fijara a $z_0 = 2$.

Descripción del programa. Se desarrollo un programa, *ver Anexo B.2 EXPERIMENTOS ALGORITMO 2* que permite aplicar el algoritmo 2; este programa recibe como datos de entrada:

- La matriz que se quiere usar (Tridiagonal o Hessenberg Superior)
- El valor de los costos c1 y c2
- El número de estados n
- El valor de λ
- El valor de k

Automáticamente el programa genera la función objetivo y desarrolla todas las iteraciones y subiteraciones. El programa se para cuando el algoritmo 2 converge a la solución final. El resultado final es una tabla resumen en la cual se muestran los resultados de cada iteración, es decir, los valores de μ* y g(μ*) para cada z.

4.2.1 Escenario A1

Características

λ = 0,1

Matriz Tridiagonal

Costos iguales, C1 = C2 = 1

Experimentos realizados. A continuación se muestra una tabla de experimentos para n = 10 y k= 6, se puede observar que z se incrementa en k = 6 unidades en cada iteración. El valor óptimo obtenido por el algoritmo 2 se encuentra cuando z = 26. (ver Anexo *B.2.1 Experimentos Escenario A1* para observar tablas de experimentos).

Tabla 29. Experimentos n = 10, k = 6

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,102	0,961
	8	0,348	0,714
	14	0,379	0,719
	20	0,402	0,728
	26	**0,410**	**0,731**

La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario:

Tabla 30. Valores óptimos μ* y g (μ*) encontrados mediante el algoritmo 2 en A1

		N		
		3	**10**	**50**
K	**6**	μ* = 0,344 g(μ*) = 0,678	μ* = 0,41 g(μ*) = 0,731	μ* = 0,41 g(μ*) = 0,731
	15	μ* = 0,345 g(μ*) = 0,678	μ* = 0,416 g(μ*) = 0,732	μ* = 0,416 g(μ*) = 0,732
	30	μ* = 0,344 g(μ*) = 0,678	μ* = 0,416 g(μ*) = 0,732	μ* = 0,416 g(μ*) = 0,732

Tabla 31. Desviaciones $|\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}|$ en A1

$\mu^*_{(tradicional)} = 0,416$		n		
		3	**10**	**50**
K	**6**	0,072	0,006	0,006
	15	0,071	0	0
	30	0,072	0	0

4.2.2 Escenario A2

Características

λ = 0,2

Matriz Tridiagonal

Costos iguales, C1 = C2 = 1

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver Anexo *B.2.2 Experimentos Escenario A2* para observar tablas de experimentos):

Tabla 32. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 en A2

		N		
		3	**10**	**50**
K	**6**	μ* = 0,473 g(μ*) = 0,960	μ* = 0,643 g(μ*) = 1,094	μ* = 0,643 g(μ*) = 1,094
	15	μ* = 0,473 g(μ*) = 0,960	μ* = 0,647 g(μ*) = 1,094	μ* = 0,647 g(μ*) = 1,094
	30	μ* = 0,473 g(μ*) = 0,960	μ* = 0,647 g(μ*) = 1,094	μ* = 0,647 g(μ*) = 1,094

Tabla 33. Desviaciones $|\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}|$ en A2

$\mu^*_{(tradicional)} = 0,647$		n		
		3	**10**	**50**
K	**6**	0,174	0,004	0,004
	15	0,174	0	0
	30	0,174	0	0

4.2.3 Escenario B

Características

λ = 0,1

Matriz Tridiagonal

Costos iguales, C1 =1,4 C2 = 1

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver Anexo *B.2.3 Experimentos Escenario B* para observar tablas de experimentos):

Tabla 34. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 en B

		n		
		3	**10**	**50**
K	**6**	μ* = 0,287 g(μ*) = 0,804	μ* = 0,361 g(μ*) = 0,886	μ* = 0,361 g(μ*) = 0,886
	15	μ* = 0,288 g(μ*) = 0,804	μ* = 0,366 g(μ*) = 0,888	μ* = 0,366 g(μ*) = 0,888
	30	μ* = 0,288 g(μ*) = 0,804	μ* = 0,367 g(μ*) = 0,888	μ* = 0,367 g(μ*) = 0,888

Tabla 35. Desviaciones | $\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}$ | en B

		n		
$\mu^*_{(tradicional)}$ = 0,367		3	10	50
K	6	0,08	0,006	0,006
	15	0,079	0,001	0,001
	30	0,079	0	0

4.2.4 Escenario C

Características

λ = 0,1

Matriz Tridiagonal

Costos iguales, C1 =1 C2 = 1,4

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver Anexo *B.2.4 Experimentos Escenario C* para observar tablas de experimentos):

Tabla 36. Valores óptimos μ* y g(μ*)encontrados mediante el algoritmo 2 en C

		n		
		3	10	50
K	6	μ* = 0,409 g(μ*) = 0,800	μ* = 0,472 g(μ*) = 0,848	μ* = 0,472 g(μ*) = 0,848
	15	μ* = 0,41 g(μ*) = 0,800	μ* = 0,474 g(μ*) = 0,848	μ* = 0,474 g(μ*) = 0,848
	30	μ* = 0,41 g(μ*) = 0,800	μ* = 0,474 g(μ*) = 0,848	μ* = 0,474 g(μ*) = 0,848

Tabla 37. Desviaciones | $\mu^*_{(algoritmo)} - \mu^*_{(tradicional)}$ | en C

$\mu^*_{(tradicional)} = 0,474$		n		
		3	**10**	**50**
K	**6**	0,065	0,002	0,002
	15	0,064	0	0
	30	0,064	0	0

4.2.5 Observaciones Matriz Tridiagonal. Se verifica la gran dependencia que existe entre el algoritmo 1 y el algoritmo 2, en la mayoría de los experimentos, cuando se usa algún valor alto para k, digamos k = 15 o 30, el exponente z crece con mayor rapidez y por lo tanto el algoritmo 2 se aproxima al parámetro de comparación con mayor rapidez y exactitud, pero este resultado se debe a las propiedades del algoritmo 1.

Se mantiene la tendencia observada en el comportamiento de los resultados con el algoritmo 1. Existe un comportamiento convergente de izquierda a derecha y de arriba abajo en las tablas indicando una convergencia total en la esquina inferior derecha.

Sin embargo en ninguno de los experimentos se observa un beneficio considerable en el uso del algoritmo 2, ya que se llega al mismo resultado que se obtuvo con el algoritmo 1. Por tal razón el algoritmo 2 se considera como una reutilización sistemática de algoritmo 1 que permite predefinir los experimentos que se quieren realizar.

4.2.6 Escenario D

Características

λ = 0,1

Matriz Hessenberg Superior

Costos iguales, C1 = C2 = 1

Experimentos Realizados. A continuación se muestran dos tablas de experimentos, una para la combinación n = 3 y K= 6, y otra para n = 10 y k =6 (ver Anexo *B.2.5 Experimentos Escenario D* para observar tablas de experimentos). En la primera tabla se puede observar que z se incrementa en k = 6 unidades en cada iteración; el algoritmo 2 obtiene el valor óptimo μ* = 0,266 cuando z = 20. En la segunda tabla se puede observar que z se incrementa en k = 6 unidades en cada iteración también, sin embargo el algoritmo solo alcanza a solucionar el problema para z = 14; el programa es incapaz de continuar con las iteraciones ya que se bloquea cuando z toma el valor de 20 .

Tabla 38. Experimentos n = 3, k = 6

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,107	0,618
	8	0,250	0,577
	14	0,264	0,582
	20	**0,266**	**0,582**

Tabla 39. Experimentos n = 10, k =6

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,177	0,637
	8	0,259	0,583
	14	0,272	0,587
	20	*	*
	*****	*****	*****

La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario:

Tabla 40. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 en D

		n		
		3	**10**	**50**
K	**6**	μ* = **0,266** g(μ*) =**0,582**	μ* = * g(μ*) =*	μ* = * g(μ*) =*
	15	μ* = **0,266** g(μ*) = **0,582**	μ* =* g(μ*) =*	μ* =* g(μ*) =*
	30	μ* = **0,266** g(μ*) = **0,582**	μ* =* g(μ*) =*	μ* =* g(μ*) =*

4.2.7 Escenario E

Características

λ = 0,1

Matriz Hessenberg Superior

Costos iguales, C=1,4, C2 = 1

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver Anexo *B.2.6 Experimentos Escenario E* para observar tablas de experimentos):

Tabla 41. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 en E

		n		
		3	**10**	**50**
K	**6**	μ* = 0,217 g(μ*) = 0,678	μ* =* g(μ*) =*	μ* =* g(μ*) =*
	15	μ* = 0,217 g(μ*) = 0,678	μ* =* g(μ*) =*	μ* =* g(μ*) =*
	30	μ* = 0,217 g(μ*) = 0,678	μ* =* g(μ*) =*	μ* =* g(μ*) =*

4.2.8 Escenario F

Características

λ = 0,1

Matriz Hessenberg Superior

Costos iguales, C1 = 1 C2 = 1,4

Experimentos Realizados. La siguiente es una tabla resumen de los valores óptimos encontrados para todos los experimentos de este escenario (ver Anexo *B.2.7 Experimentos Escenario F* para observar tablas de experimentos):

Tabla 42. Valores óptimos μ* y g(μ*) encontrados mediante el algoritmo 2 en F

		N		
		3	**10**	**50**
K	**6**	μ* = 0.324 g(μ*) = 0,698	μ* = * G(μ*) =*	μ* =* g(μ*) =*
	15	μ* = 0.324 g(μ*) = 0,698	μ* =* G(μ*) =*	μ* =* g(μ*) =*
	30	μ* = 0.324 g(μ*) = 0,698	μ* =* G(μ*) =*	μ* =* g(μ*) =*

4.2.9 Observaciones Matriz Hessenberg Superior. Se puede observar como la mayoría de los experimentos no se pudieron completar como consecuencia de la incapacidad del programa para solucionar problemas con número de estados n = 10 y n = 50. En estos casos el algoritmo 2 permitía realizar las iteraciones con z menores, pero en la medida que se incrementaba su valor en k unidades, la capacidad del programa era insuficiente. Nuevamente se verifican inconvenientes con el algoritmo 2, pero causadas por las propiedades del algoritmo 1.

De la misma manera que con la matriz Tridiagonal, no se obtiene ningún beneficio adicional en el uso del algoritmo 2, simplemente se logra la programación sistemática de ciertos experimentos a partir de la modificación de z.

5. APROXIMACIÓN INICIAL A LA SOLUCIÓN DEL SISTEMA M/M/S MEDIANTE CADENAS DE MARKOV

En este capítulo se retomará el problema estudiado en el capitulo 2, en el cual se abordó la solución de un problema de diseño de dos (2) parámetros asociado *un sistema de colas* M/M/s. En ese capítulo el número esperado de unidades en el sistema se determinó mediante expresiones cerradas de la teoría de colas.

El enfoque utilizado en este capítulo será utilizar los algoritmos de Kumin para realizar una aproximación a la solución de este problema de diseño asociado al M/M/s, usando cadenas de Markov para hallar el número esperado de unidades en el sistema.

5.1 PLANTEAMIENTO DEL PROBLEMA

Recordemos que el modelo tienes las siguientes características:

- Una (1) cola
- s, número de servidores
- Tasas de llegada λ y de servicio μ con distribución de probabilidad exponencial.
- Disciplina de la cola: FCFS.

El problema de diseño consiste en hallar el valor óptimo de s y μ, tal que minimicen la función objetivo del costo total.

$$F.O: \quad Min\ f(s, \mu) = c_1 * s + c_2 * \mu + c_3 * En$$

Sujeta a: $\lambda / (s * \mu) < 1$

$\lambda > 0 , \ \mu > 0$

Donde,

- c1 es el costo asociado al número de servidores en el sistema.
- c2, es el costo asociado a la tasa de servicio μ.
- c3, es el costo asociado al número esperado de clientes en el sistema.
- c1, c2 y c3 son constantes arbitrarias.
- μ, es la tasa de servicio y es un parámetro de diseño (variable de decisión continua)
- s, es el número de servidores en el sistema y es un parámetro de diseño (variable de decisión discreta).
- λ, es la tasa de llegadas, y se le dará un valor predeterminado.

Para este enfoque, ***En***, es el número esperado de clientes en el sistema de colas y será hallado usando las propiedades de una matriz de transición asociada a un sistema de colas M/M/s. Esta matriz depende de λ, μ, s y n (número de estados).

5.2 NÚMERO ESPERADO DE UNIDADES EN EL SISTEMA M/M/S

Como ya se explicó con anterioridad, el número esperado de clientes en el sistema para el M/M/1 es un proceso markoviano de tiempo continuo; es bien conocido que de este tipo de procesos, pueden ser derivadas varias cadenas de markov de tiempo discreto. En la aplicación de los algoritmos de Kumin a la solución del sistema M/M/1, se utilizaron dos (2) matrices de transición de tiempo discreto; tridiagonal y Hessenberg Superior.

Para la aplicación de los algoritmos a la solución del M/M/s se hace indispensable una matriz de transición de tiempo discreto para el número esperado de clientes en el sistema. Para este caso la matriz presenta una mayor complejidad ya que depende de una variable más: el número de servidores, s.

Según Stewart(1994) en un cadena de markov homogénea de tiempo continuo, CTMC (*continuos time markov chain*), el tiempo gastado en el sistema en cada estado esta distribuido exponencialmente. Si ignoramos este tiempo empleado en cada estado y consideramos solo las transiciones realizadas por el sistema, podremos definir una nueva, cadena de markov, de tiempo discreto, llamada la cadena de markov encajada, EMC (embedded markov chain). Esta demostrado que muchas propiedades de una cadena de markov de tiempo continuo pueden ser deducidas de su cadena de markov encajada correspondiente.

Comúnmente la CTMC se le conoce como matriz-Q y su EMC como S. De acuerdo a Stewart(1994):

$$S = DQ^{-1}(DQ - Q)$$

Donde Q es la matriz de transición de tiempo continuo, DQ es una matriz que contiene solo los elementos de la diagonal de Q y el resto de elementos son todos cero y S es la matriz de transición encajada, que usaremos como modelo de tiempo discreto para el número esperado de clientes en el sistema para el M/M/s.

Según Winkel(2004) la matriz- Q para el sistema M/M/s es:

Figura 19. Matriz- Q para el sistema M/M/s

$$\begin{pmatrix} -\lambda & \lambda & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\ \mu & -\mu-\lambda & \lambda & 0 & \cdots & 0 & 0 & 0 & \cdots \\ 0 & 2\mu & -2\mu-\lambda & \lambda & \ddots & 0 & 0 & 0 & \ddots \\ 0 & 0 & 3\mu & -3\mu-\lambda & \ddots & 0 & 0 & 0 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & \ddots & -s\mu-\lambda & \lambda & 0 & \ddots \\ 0 & 0 & 0 & 0 & \ddots & s\mu & -s\mu-\lambda & \lambda & \ddots \\ 0 & 0 & 0 & 0 & \ddots & 0 & s\mu & -s\mu-\lambda & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

(Winkel, 2004)

Como se puede observar, la matriz-Q para el sistema M/M/s, depende de: el número de estados en el sistema que se seleccionen n, que determina el tamaño de la matriz; del número de servidores estudiados s, y de la tasa de llegadas λ. Internamente los elementos de la matriz mantienen un comportamiento previsible:

Para la diagonal, el primer elemento es – λ; luego –**1**μ-λ, y empieza aumentar –**2**μ-λ, –**3**μ-λ y así sucesivamente, hasta que se alcanza el valor estudiado para s. De ahí en adelante –**s**μ-λ es lo que va en el resto de la diagonal. Debajo de la diagonal pasa lo mismo, comienza como μ, luego empieza aumentar a 2μ, 3μ y así sucesivamente hasta que se alcanza el valor de seleccionado para s. De ahí en adelante sμ es lo que se pone. Arriba de la diagonal siempre va λ. Para terminar, la suma de las filas da igual a cero.

Procedimiento para hallar En. Para obtener el número esperado de unidades en un sistema, digamos con seis (6) servidores, seleccionando de forma arbitraria 10 estados, usando cadenas de Markov se recurre a la matriz anterior de la siguiente manera:

Q =

-200	200	0	0	0	0	0	0	0	0
μ	(-μ-200)	200	0	0	0	0	0	0	0
0	2μ	(-2μ-200)	200	0	0	0	0	0	0
0	0	3μ	(3-μ-200)	200	0	0	0	0	0
0	0	0	4μ	(-4μ-200)	200	0	0	0	0
0	0	0	0	5μ	(-5μ-200)	200	0	0	0
0	0	0	0	0	6μ	(-6μ-200)	200	0	0
0	0	0	0	0	0	6μ	(-6μ-200)	200	0
0	0	0	0	0	0	0	6μ	(-6μ-200)	200
0	0	0	0	0	0	0	0	6μ	(-6μ)

DQ =

-200		0	0	0	0	0	0	0	0
0	(-μ-200)		0	0	0	0	0	0	0
0	0	(-2μ-200)	0	0	0	0	0	0	0
0	0	0	(3-μ-200)	0	0	0	0	0	0
0	0	0	0	(-4μ-200)	0	0	0	0	0
0	0	0	0	0	(-5μ-200)	0	0	0	0
0	0	0	0	0	0	(-6μ-200)	0	0	0
0	0	0	0	0	0	0	(-6μ-200)	0	0
0	0	0	0	0	0	0	0	(-6μ-200)	0
0	0	0	0	0	0	0	0		(-6μ)

Aplicando **S = DQ^-1(DQ – Q) =**

0	1	0	0	0	0	0	0	0	0
1/(μ+200)*μ	0	200/(μ+200)	0	0	0	0	0	0	0
0	1/(μ+100)*μ	0	100/(μ+100)	0	0	0	0	0	0
0	0	3/(3*μ+200)*μ	0	200/(3*μ+200)	0	0	0	0	0
0	0	0	1/(μ+50)*μ	0	50/(μ+50)	0	0	0	0
0	0	0	0	1/(μ+40)*μ	0	40/(μ+40)	0	0	0
0	0	0	0	0	3/(3*μ+100)*μ	0	100/(3*μ+100)	0	0
0	0	0	0	0	0	3/(3*μ+100)*μ	0	100/(3*μ+100)	0
0	0	0	0	0	0	0	3/(3*μ+100)*μ	0	100/(3*μ+100)
0	0	0	0	0	0	0	0	1	0

El tamaño de estas matrices es determinado por el número de estados seleccionados; para este ejemplo n= 10 estados, creando matrices de tamaño 10x10. Ahora a partir de estas matrices se encuentra el número esperado de unidades en el sistema, usando el algoritmo 1 de Kumin(1968).

$$En = P S^{z} F =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

x

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{(\mu+200)*\mu} & 0 & \frac{200}{(\mu+200)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{(\mu+100)*\mu} & 0 & \frac{100}{(\mu+100)} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{3}{(3*\mu+200)*\mu} & 0 & \frac{200}{(3*\mu+200)} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{(\mu+50)*\mu} & 0 & \frac{50}{(\mu+50)} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{(\mu+40)*\mu} & 0 & \frac{40}{(\mu+40)} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{3}{(3*\mu+100)*\mu} & 0 & \frac{100}{(3*\mu+100)} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{(3*\mu+100)*\mu} & 0 & \frac{100}{(3*\mu+100)} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{(3*\mu+100)*\mu} & 0 & \frac{100}{(3*\mu+100)} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}^{50}$$

x

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{bmatrix}$$

A partir de la multiplicación de estas matrices y vectores se obtiene una expresión bastante larga y compleja, pero fácil de obtener mediante los sistemas actuales,

cuya variable independiente es solo μ. Esta expresión puede ser utilizada en la función objetivo asociada a un sistemas de colas M/M/s=6.

Problema de Diseño M/M/s=6: si consideramos un sistema en el cual el número de servidores es s = 6, el problema es:

$$F.O: \quad Min \ f(s = 6, \mu) = 6 + \mu + En$$

$$F.O: \quad Min \ f(s = 6, \mu) = 6 + \mu + (P\, S^{z=50}\, F)$$

$$F.O: \quad Min \ f(s = 6, \mu) = 6 + \mu + E(\mu)$$

$$Sujeta\ a: \quad 200 / (6 * \mu) < 1$$

$$\mu > 0$$

El objetivo es encontrar el valor óptimo para la tasa de servicio que permita encontrar el valor mínimo de la función objetivo cuya única variable independiente es μ. Para solucionar este problema se recurre de nuevo a la función ***fmincon*** del Optimization Toolbox de Matlab, y se obtiene el siguiente resultado: tasa de servicio óptima μ = 33.36, valor mínimo de la función objetivo = 46.26.

5.3 EXPLICACIÓN DE LA SOLUCIÓN

Se consideró el mismo problema de diseño estudiado en el capitulo 2, asociado a un sistema de colas M/M/s, en el cual la función objetivo a minimizar es:

$$F.O: \quad Min \ f(s, \mu) = s + \mu + En$$

$$Sujeta\ a: \quad 200 / (s * \mu) < 1$$

$$\mu > 0$$

Donde:

- c1 = c2 = c3 = 1
- λ = 200
- μ y s son las variables de decisión
- μ > 0

- La minimización esta sujeta a la siguiente restricción: ρ = (λ / s * μ) < 1. Es decir, (200 / s) * μ < 1.

Se desarrollo un programa *(ver Anexo A.4 DISEÑO SISTEMA M/M/S – APLICACIÓN ALGORITMOS)* que permite hallar el valor óptimo de μ, para cualquier valor de s; este programa recibe como valores de entrada el número de estados n, el número de servidores s, el valor de la tasa de llegadas λ, los costos (c1,c2, c3), y el valor del exponente z; y arroja como resultado el valor óptimo de la tasa de servicio para ese sistema y valor mínimo de la función objetivo.

En el capitulo 2 mediante expresiones cerradas se determinaron la tasas de servicio óptimas, utilizando s servidores(s = 2 hasta s = 20) en un sistema M/M/s con las condiciones anteriores. Para cada s seleccionado (número de servidores), se obtuvo el valor correspondiente al valor óptimo de μ y su respectivo valor mínimo para la función objetivo. La combinación de s y μ que logra el menor valor de la función es considerada como el punto óptimo. Mediante ese enfoque se determino que el número óptimo de servidores en este sistema es de 13 servidores y la tasa de servicio óptima es de 20.57.

A través de la utilización de los algoritmos de Kumin y la aplicación de las matrices de transición del sistema M/M/s se obtuvieron los valores óptimos de μ y sus respectivos valores mínimos para la función objetivo, probando con diferentes servidores desde s= 0 hasta s =25.

De acuerdo a los resultados obtenidos en la validación de los algoritmos se pudo establecer que estos convergen a la solución óptima del problema de diseño eficazmente cuando los parámetros n y z de los algoritmos toman valores altos, es decir, cuando el sistema en estudio alcanza el estado estable. Por tal razón se estableció n = 30 y z = 150 para garantizar que se llegara al estado estable con la aplicación de los algoritmos.

Los resultados obtenidos fueron:

Tabla 43. Resultados numéricos de los Algoritmos para el sistema M/M/s

s	μ*	f(μ)
3	66,28	81,28
6	33,36	52,79
10	22,70	45,33
12	19,78	44,87
13	**18,65**	**45,04**
14	17,69	45,4
16	16,2	46,55
20	14,48	49,85
25	13,96	54,72

Como se puede observar el valor mínimo de la función objetivo, f = 45.04, se obtiene cuando s = 13 y μ =18.65. La función al igual que la obtenida mediante expresiones cerradas es convexa; cuando el número de servidores toma valores pequeños digamos 3, el costo total es muy alto, en la medida que el número de servidores va aumentando el costo se disminuye hasta el valor óptimo de 13 servidores, de ahí en adelante el costo empieza a aumentar poco a poco.

Figura 20. Función Convexa M/M/s. Enfoque: Cadenas de Markov

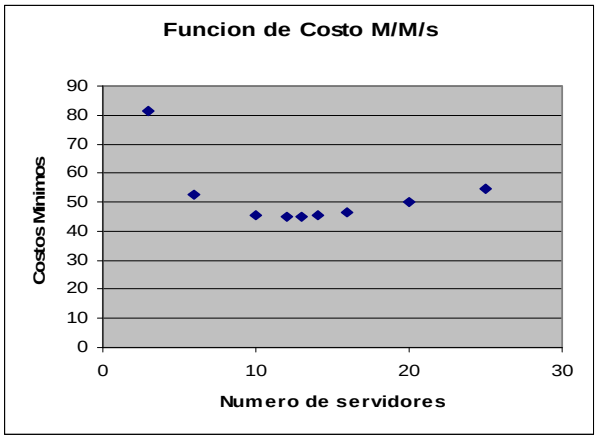
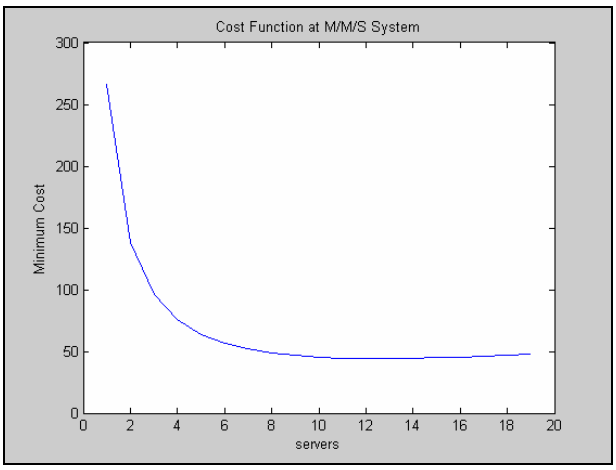


Figura 21. Función Convexa M/M/s. Enfoque: Expresiones Cerradas



Existe una gran similitud entre las 2 funciones, las dos establecen que para el sistema M/M/s , el valor óptimo de servidores es 13, sin embargo los dos enfoques difieren en la tasa de servicio más apropiada. Estos resultados también permiten inferir que los algoritmos de Kumin pueden ser una gran alternativa para el diseño de sistemas de colas más complejos.

6. CONSIDERACIONES FINALES

6.1 CONCLUSIONES

La limitación del software disponible cuando se formularon los algoritmos dificultó su implementación en el diseño de sistemas de colas. Los programas desarrollados en esta investigación permiten probar que estos algoritmos convergen a la solución óptima de diversos problemas de diseño efectivamente.

Mediante el desarrollo de esta investigación se pudo verificar que el estudio sobre la teoría de colas ha estado predominado principalmente por modelos descriptivos; modelos prescriptivos de diseño basados en técnicas de optimización son muy escasos, y la mayoría de estos utilizan expresiones cerradas para hallar las medidas de congestión del sistema. La implementación de los algoritmos de Kumin representan una metodología alterna para el diseño de sistemas colas; por medio de esta se consigue obtener los valores óptimos de los parámetros de un sistema aplicando las propiedades de las cadenas de markov y sin necesidad hallar expresiones cerradas.

La fase de experimentación permite la obtención de las siguientes conclusiones con respecto a los algoritmos:

Los dos algoritmos convergen a la solución óptima del problema de diseño, sin embargo se concluye que el algoritmo 2 es una reutilización del algoritmo 1. El algoritmo 1 es realmente el conjunto de operaciones que permiten la convergencia y la obtención del punto óptimo, sin embargo el algoritmo 2 puede ser útil ya que permite aplicar el algoritmo 1

automáticamente para ciertos z que se fijan de acuerdo al crecimiento del factor k.

Los algoritmos de Kumin están basados en una variedad de parámetros, sin embargo los factores principales que afectan la convergencia de los algoritmos y la obtención de los valores óptimos del problema de diseño son el número de estados y el exponente z.

La convergencia del algoritmo es consistente con las propiedades de las Cadenas de Markov; en la medida que el exponente z es más alto, hay que realizar menos iteraciones para que el algoritmo converja a la solución óptima y la convergencia se obtiene más rápido, esto es consistente con la convergencia de ciertas matrices de transición que cuando son elevadas a un exponente alto alcanzan la distribución de estado estable.

A pesar de que los programas diseñados permitieron la implementación y validación de los algoritmos para la mayoría de los escenarios, el software presento dificultad para solucionar ciertos problemas relacionados con la matriz Hessenberg Superior debido a su complejidad. Estudios posteriores se concentraran en ampliar la capacidad de los programas desarrollados.

La aplicación de los algoritmos de Kumin en el diseño óptimo del sistema de colas M/M/1 fue bastante satisfactoria; en todos los escenarios de la matriz tridiagonal se obtuvo el mismo resultado que se obtiene mediante expresiones cerradas. Tal exactitud permite darle validez a la implementación de los algoritmos en el diseño de sistemas de colas markovianos. Esto se confirma en la tablas 10 a la 20.

A partir de la aplicación de los algoritmos de Kumin en el diseño óptimo del sistema de colas M/M/s se confirma la flexibilidad de esta metodología para ser implementada en la solución de problemas de diseño mucho más complejos. En la

comparación entre los resultados obtenidos mediante los algoritmos y aquellos obtenidos mediante expresiones cerradas se puede observar una gran similitud, tanto así que los dos métodos concuerdan en el número óptimo de servidores (n = 13) como parámetro de diseño. Las graficas en las figuras 21 y 22 confirman estos resultados.

6.2 CONTRIBUCIONES DEL TRABAJO DE GRADO

Se realizó una recopilación bibliografica de las diferentes aproximaciones propuestas sobre el diseño óptimo, la cual servirá como fuente de consulta para proyectos similares en el futuro. Se mencionan gran cantidad de autores y se explican sus diferentes enfoques a lo largo del tiempo. También se realizó una explicación detallada de los algoritmos propuestos en esta investigación.

Se diseñaron e implementaron 4 programas por medio del software de optimización de Matlab: El primero permite la solución del problema de diseño óptimo de un sistema M/M/s mediante expresiones cerradas. El segundo y el tercero permiten la aplicación de los algoritmos 1 y 2 en la solución del problema de diseño de un sistema M/M/1 respectivamente. El último permite realizar una aproximación a la solución del problema de diseño óptimo del sistema M/M/s mediante la aplicación de los algoritmos. Estos programas permitieron la validación de los algoritmos propuestos y por la tanto de esta metodología como un enfoque alterno de diseño óptimo.

6.3 RECOMENDACIONES

Ya se demostró que para el diseño óptimo de los sistemas M/M/1 y M/M/s mediante los algoritmos se obtienen resultados bastante acertados. De esta

manera se recomienda ampliar los modelos existentes de diseño óptimo basados en cadenas de Markov considerando que esta metodología omite la necesidad de hallar expresiones cerradas para las medidas de congestión del sistema facilitando la solución del problema de diseño.

Se recomienda verificar la validez de la matriz Q usada para obtener la solución en el diseño M/M/s, así como de los resultados obtenidos mediante la aproximación a la solución realizada. Por ultimo se recomienda mejorar el código existente para permitir la solución de problemas mucho más complejos y de mayor tamaño.

BIBLIOGRAFÍA

Ausín Olivera, M. C. (2004). Análisis bayesiano de sistemas de colas. Universidad Carlos III de Madrid. Departamento de Estadística y Econometría.

Bariş, A., Shiri S. (2006). Dynamic Control of an M/M/1 Service System with Adjustable Arrival and Service Rates. Management Science. Institute for Operations Research and the Management Sciences (INFORMS).

Bohm, S. G. (1994). Transient Analysis of M/M/1 Queues in Discrete Time with General Server Vacations. *Journal of Applied Probability*, Vol. 31.

Evans, R. V. (1981). Markov Chain Design Problems. *Operations Research*. Vol. 29, No. 5. Operations Management.

Gracia, J. M. (2002). Matrices no negativas, paseos aleatorios y cadenas de Markov. Matemática Aplicada y Estadística. Universidad del País Vasco.

Isaacson, D. y Madsen, R. (1976). Markov Chains Theory and Applications. John Wiley and Sons, 1976.

Kumin H. (1968). The Design of Markovian Congestion Systems. Technical memorandum No. 115. Case WRU University. Cleveland, OH.

Neuts (2002). Two M/M/1 Queues with Transfers of Customers. Queueing Systems, ABI/INFORM Global.

Serra, D. (2002). Métodos cuantitativos para la Toma de Decisiones.

Stern, H. I.(2005) Optimal control of a facility with periodic interrupted demand. Journal of Optimization Theory and Applications. Springer Netherlands. Springer Link.

Stewart, W. (1994). Introduction to the Numerical Solution of Markov Chains. Princeton University Press.

Stidham, S. Jr., Bagajewicz, M. (2006). Optimal Design of Queuing Systems Crc Press Llc. 1 edition.

Tadj y Choudhury. (2005). Optimal Design and Control of Queues. Sociedad de Estadística e Investigación Operativa. Vol. 13, No. 2, pp. 359-412.

Taha, H. (1997). Investigación de Operaciones. Una Introducción. Prentice Hall.

Thomas, B., Donald G., Michael, J. (1977). Magazine. A Classified Bibliography of Research on Optimal Design and Control of Queues. Operations Research, Vol. 25, No. 2.

Venkataram, P. (2002). Applied Optimization with Matlab Programming. John Wiley and Sons.

Winkel, M. (2004). Applied Probability, Department of Statistics, University of Oxford

Wuyi, Y. (2004). Modeling and Methodology for Performance Evaluation in Next-Generation Wireless Communication Networks. Department of Information Science and Systems Engineering. Konan University. Kobe, 658-8501 Japan.

ANEXOS

ANEXO A. CÓDIGOS DE PROGRAMACIÓN

A.1 DISEÑO SISTEMA M/M/S – EXPRESIONES CERRADAS

A.1.1 Archivo Project1.m

Project1.m

```
% DISEÑO SISTEMA M/M/S MEDIANTE EXPRESIONES CERRADAS
% PROGRAMA
% GUILLERMO AUGUSTO VESGA ACEVEDO

global s ;
global lam ;
global C1 ;
global C2 ;
global C3 ;

lam = 200;
s = 1;
C1 = 1;
C2 = 1;
C3 = 1;
sopt = 0;
uopt = 0;
funmin = 99999999;

while s < 20

t = (lam/s)+10;
lb = [0];
x0 = [t];
s
[x, val] = fmincon('fun',x0,[],[],[],[],lb,[],'nonlin')
vec1(s)=val;
```

```
    if val < funmin
       funmin = val;
       sopt = s;
       uopt = x;
    end

s = s+1;

end

funmin
sopt
uopt
figure
plot(vec1)
xlabel('servers');
ylabel('Minimum Cost');
title('Cost Function at M/M/S System')
```

A.1.2 Archivo fun.m

fun.m

```
% DISEÑO SISTEMA M/M/S MEDIANTE EXPRESIONES CERRADAS
% FUNCION OBJETIVO
% GUILLERMO AUGUSTO VESGA ACEVEDO

function g=fun(x);

global s ;
global lam ;
global C1 ;
global C2 ;
global C3 ;

n = 0;
```

```
sum = 0;

while n < s
   sum = sum + [((lam/x)^n)/factorial(n)];
   n = n+1;
end

g = C1*s + C2*x + C3*[ [ [1/[sum + (((lam/x)^s)/factorial(s))*(1/(1-(lam/s*x)))]] * [(lam/x)^s] *
[lam/(s*x)] ] / [ (factorial(s)) * (1-(lam/(s*x)))^2 ] ] + (lam/x) ];
```

A.1.3 Archivo nonlin.m

nonlin.m

```
% DISEÑO SISTEMA M/M/S MEDIANTE EXPRESIONES CERRADAS
% RESTRICCION NO LINEAL
% GUILLERMO AUGUSTO VESGA ACEVEDO

function [c,ceq]=nonlin(x)

global s ;
global lam ;

c = (lam)/(s*x) - 1;
ceq=[];
```

A.2 DISEÑO SISTEMA M/M/1 – ALGORITMO 1

A.2.1Archivo program1.m

rogram1.m

```
% DISEÑO SISTEMA M/M/1 - ALGORITMO 1
% PROGRAMA 1
% GUILLERMO AUGUSTO VESGA ACEVEDO
```

```
%function vxg=program(CA,CB,Z,n,LAMBD,x0,lb,ub,CC,ToH)
clc
clear all
disp('Este programa minimiza una función objetivo de la forma:')
disp('              g1(x) =  [C1*x + C2*(F*A^z*P)]')
disp('- C1, C2 y Z son valores enteros.')
disp('- F es un  vector creciente de una sola fila (1xn)')
disp('- A es una matriz de nxn, que puede ser Tridiagonal o Hessenberg.')
disp('- P es un vector de una sola columna (nx1).  ')

global CA CB Z LAMBD miu
global  A F P
%introduciendo variables
CA=input('Digite el valor de C1: ');
CB=input('Digite el valor de C2: ');
Z=input('Digite el valor de z: ');
n=input('Digite el numero de estados, n: ');
LAMBD=input('Digite el valor de Lambda: ');
x0=LAMBD;
lb=0.1;
ub=0.9;
CC=0.0001;
disp('A continuacion digite 1 si desea trabajar con una matriz Tridiagonal');
disp('O  2 si desea trabajar con una matriz tipo Hessenberg');
ToH=input(' \n\n');
%Se crea la matriz A y los vectores F y P
syms miu
if ToH==1 % Si la matriz es Tridiagonal
   A=sym(zeros(n,n));
   A(1,1)=1-LAMBD;
   A(2,1)=LAMBD;
   A(n,n)=1-miu;
   A(n-1,n)=miu;
   for i=2:n-1
      for j=1:n
         if i==j
```

```
            A(j,i)=1-LAMBD-miu;
        elseif j==i-1
            A(j,i)=miu;
        elseif j==i+1
            A(j,i)=LAMBD;
        end
    end
end

elseif ToH==2 % Si la matriz es hessemberg
    A=sym(zeros(n,n));
    for i=1:n
        for j=1:n
            if (i==j && j~=n && j~=1)
                A(j,i)=1-LAMBD-(j-1)*miu;
            elseif (i==n && j==n)
                A(j,i)=1-(n-1)*miu;
            elseif (i==1 && j==1)
                A(j,i)=1-LAMBD;
            elseif j==i+1
                A(j,i)=LAMBD;
            elseif j<i
                A(j,i)=miu;
            end
        end
    end
else        % Si la opción escogida no esta dentro de las anteriores.
    disp('              OPCIÓN NO VALIDA');
    return
end
P=zeros(n,1);
F=zeros(1,n);
 for i=1:n
     F(i)=i-1;
     if i==1
         P(i,1)=1;
```

```
    else P(i,1)=0;
    end
end
A;
F;
P;
a=1;%Variable para almacenar los valores de x y g en un vector
t1= clock;%Vector inicial para calcular el tiempo de optimización
err=CC+1;% Asiganación del error para iniciar la optimización
while (err>CC)% Mientras err es mayor a CC se minimiza la función
  [x,feval]= fmincon('fun',x0,[ ],[ ],[ ],[ ],lb,ub,'nonlin');
   clc
  vxg(a,1)=x;       % Se guarda el valor minimo x
  vxg(a,2)=feval;   %Se guarda el valor minimo g(x)
  miu=vxg(a,1);
  A1=eval(A);        % Se evalua la matriz A con el x obtenido
  P=A1*P;            % Se multiplica la A numérica con el vector P: g(x)=[c1x + c2( FA^z,A1*P ).
  miu=sym('miu');
  if a>=2            % Si se ha minimizado la función dos o mas veces
     err=abs(vxg(a,1)-vxg(a-1,1));% Se calcula el error
     if err<0.0000000001 % Si el error es muy pequeño se asigna como 1
        err=1;         %Debido a que al principio de la minimización el resultado tiende a ser el mismo
     end
  else          %Si solo se ha realizado la minimización una vez
    err=CC+1;  % Se asigna err=1;
  end
  a=a+1;
end
t2= clock; % Vector 2 para el calculo del tiempo de optimización
tp = t2-t1; % Calculo del tiempo de optimización
tiempo=tp(1,4)*3600+tp(1,5)*60+tp(1,6); % Conversión del tiempo a segundos.
disp('  ***************************************************************')
fprintf(' *   El valor óptimo obtenido es %6f                    *',x);
fprintf('\n *   Para el cual el valor mínimo de la función es %6f       *', feval);
fprintf('\n *   La optimización se realizó en %6f segundos.            *\n',tiempo);
disp('  ***************************************************************')
```

```
fprintf('\n\n')
disp(' Los valores óptimos obtenidos en cada iteración y el respectivo' )
disp(' valor mínimo de la función son: ')
%vxg es la matriZ que contiene los valores de x y g obtenidos en
%cada iteracion
```

A.2.2 Archivo fun.m

fun.m

```
% DISEÑO SISTEMA M/M/1 - ALGORITMO 1
% FUNCION OBJETIVO
% GUILLERMO AUGUSTO VESGA ACEVEDO

function g=fun(x,A,Z,CA,CB,F,P);
global A Z CA CB F P
miu=x;
M = (eval(A))^Z;
g = CA*x + CB*(F*M*P);
```

A.2.3 Archivo nonlin.m

nonlin.m

```
% DISEÑO SISTEMA M/M/1 - ALGORITMO 1
% RESTRICCION NO LINEAL
% GUILLERMO AUGUSTO VESGA ACEVEDO

function [c,ceq]=nonlin(x,LAMBD)
global LAMBD
c = LAMBD/x - 1;
ceq=[];
```

A.3 DISEÑO SISTEMA M/M/1 – ALGORITMO 2

A.3.1Archivo program2.m

program2.m

```
% DISEÑO SISTEMA M/M/1 - ALGORITMO 2
% PROGRAMA 2
% GUILLERMO AUGUSTO VESGA ACEVEDO

function vxg2=program2(CA,CB,Z,K,n,LAMBD,x0,lb,ub,CC,ToH)
clc
clear all
disp('Este programa minimiza una función objetivo de la forma:')
disp('              g1(x) =  [C1*x + C2*(F*A^z*P)]')
disp('- C1, C2 y Z son valores enteros.')
disp('- F es un  vector creciente de una sola fila (1xn)')
disp('- A es una matriz de nxn, que puede ser Tridiagonal o Hessenberg.')
disp('- P es un vector de una sola columna (nx1).  ')
disp('En este caso se define un valor de K, el cual se suma al actual valor de Z ')
disp('despues que se haya obtenido el valor deseado en la actual minimización.')

global CA CB Z LAMBD miu
global  A F P
%introduciendo variables
CA=input('Digite el valor de C1: ');
CB=input('Digite el valor de C2: ');
Z=2;
K=input('Digite el valor de K: ');
n=input('Digite el numero de estados, n: ');
LAMBD=input('Digite el valor de Lambda: ');
x0=LAMBD;
lb=0.1;
ub=0.9;
CC1=0.01;
CC2=0.01;
disp('A continuacion digite 1 si desea trabajar con una matriz Tridiagonal');
```

```
disp('O  2 si desea trabajar con una matriz tipo Hessenberg');
ToH=input(' \n\n');
%Se crea la matriz A y los vectores F y P
syms miu
if ToH==1 % Si la matriz es Tridiagonal
   A=sym(zeros(n,n));
   A(1,1)=1-LAMBD;
   A(2,1)=LAMBD;
   A(n,n)=1-miu;
   A(n-1,n)=miu;
   for i=2:n-1
      for j=1:n
         if i==j
            A(j,i)=1-LAMBD-miu;
         elseif j==i-1
            A(j,i)=miu;
         elseif j==i+1
            A(j,i)=LAMBD;
         end
      end
   end

elseif ToH==2 % Si la matriz es hessemberg
   A=sym(zeros(n,n));
   for i=1:n
      for j=1:n
         if (i==j && j~=n && j~=1)
            A(j,i)=1-LAMBD-(j-1)*miu;
         elseif (i==n && j==n)
            A(j,i)=1-(n-1)*miu;
         elseif (i==1 && j==1)
            A(j,i)=1-LAMBD;
         elseif j==i+1
            A(j,i)=LAMBD;
         elseif j<i
            A(j,i)=miu;
```

```
        end
      end
   end
else        % Si la opción escogida no esta dentro de las anteriores.
   disp('            OPCIÓN NO VALIDA');
   return
end
F=zeros(1,n);
 for i=1:n
    F(i)=i-1;
 end
 b=1;%Variable para almacenar los valores de x y g en un vector
 t3= clock;%Vector inicial para calcular el tiempo de optimización
 err2=CC2+1;% Asiganación del error para iniciar la optimización
 while (err2>CC2)
    a=1;%Variable para almacenar los valores de x y g en un vector
    t1= clock;%Vector inicial para calcular el tiempo de optimización
    err1=CC1+1;% Asiganación del error para iniciar la optimización
    P=zeros(n,1);
    P(1,1)=1;
       while (err1>CC1)% Mientras err es mayor a CC se minimiza la función
         [x,feval]= fmincon('fun',x0,[ ],[ ],[ ],[ ],lb,ub,'nonlin');
          clc
         vxg(a,1)=x;       % Se guarda el valor minimo x
         vxg(a,2)=feval;   %Se guarda el valor minimo g(x)
         miu=vxg(a,1);
         A1=eval(A);        % Se evalua la matriz A con el x obtenido
         P=A1*P;            % Se multiplica la A numérica con el vector P: g(x)=[c1x + c2( FA^z,A1*P ).
         miu=sym('miu');
         if a>=2            % Si se ha minimizado la función dos o mas veces
            err1=abs(vxg(a,1)-vxg(a-1,1));% Se calcula el error
            if err1<0.0000000001 % Si el error es muy pequeño se asigna como 1
              err1=1;         %Debido a que al principio de la minimización el resultado tiende a ser el
mismo
            end
         else          %Si solo se ha realizado la minimización una vez
```

```
      err1=CC1+1;  % Se asigna err=1;
    end
    a=a+1;
    end
  t2= clock; % Vector 2 para el calculo del tiempo de optimización
  tp = t2-t1; % Calculo del tiempo de optimización
  tiempo=tp(1,4)*3600+tp(1,5)*60+tp(1,6); % Conversión del tiempo a segundos.
  disp('  ***************************************************************')
  fprintf(' *   Para un valor de Z= %6f                          *',Z);
  fprintf('\n *   El valor óptimo obtenido es %6f                  *',x);
  fprintf('\n *   Para el cual el valor mínimo de la función es %6f      *', feval);
  fprintf('\n *   La optimización se realizó en %6f segundos.           *\n',tiempo);
  disp('  ***************************************************************')
  fprintf('\n\n')
  disp(' Los valores óptimos obtenidos en cada iteración y el respectivo' )
  disp(' valor mínimo de la función son: ')
  disp(vxg)
  input('\n\n Oprima ENTER para continuar \n')
  vxg2(b,2)=vxg(a-1,1);
  vxg2(b,3)=vxg(a-1,2);
  vxg2(b,1)=Z;
  clear vxg
  if b>=2          % Si se ha minimizado la función dos o mas veces
  err2=abs(vxg2(b,2)-vxg2(b-1,2));% Se calcula el error
  else
     err2=CC2+1
  end
  b=b+1;
  Z=Z+K;
  clc
end
t4= clock; % Vector 2 para el calculo del tiempo de optimización
tp2 = t4-t3; % Calculo del tiempo de optimización
tiempo2=tp2(1,4)*3600+tp2(1,5)*60+tp2(1,6); % Conversión del tiempo a segundos.
fprintf('\n *   La optimización se realizó en %6f segundos.           *\n',tiempo2);
fprintf('\n\n')
```

```
disp(' Los valores óptimos obtenidos para cada valor de Z y el respectivo' )
disp(' valor mínimo de la función son: ')

%vxg es la matriZ que contiene los valores de x y g obtenidos en
%cada iteracion
```

A.3.2 Archivo fun.m

fun.m

```
% DISEÑO SISTEMA M/M/1 - ALGORITMO 2
% FUNCION OBJETIVO
% GUILLERMO AUGUSTO VESGA ACEVEDO

function g=fun(x,A,Z,CA,CB,F,P);
global A Z CA CB F P
miu=x;
M = (eval(A))^Z;
g = CA*x + CB*(F*M*P);
```

A.3.3 Archivo nonlin.m

nonlin.m

```
% DISEÑO SISTEMA M/M/1 - ALGORITMO 2
% RESTRICCION NO LINEAL
% GUILLERMO AUGUSTO VESGA ACEVEDO

function [c,ceq]=nonlin(x,LAMBD)
global LAMBD
c = LAMBD/x - 1;
ceq=[];
```

A.4 DISEÑO SISTEMA M/M/S – APLICACIÓN ALGORITMOS

A.4.1 Archivo program3.m

Program3.m

```
% DISEÑO SISTEMA M/M/S - APLICACION ALGORITMOS
% PROGRAMA3
% GUILLERMO AUGUSTO VESGA ACEVEDO
%function [x,feval]=program(C1,C2,C3,Z,n,s,LAMBD)
clc
clear all
global s C1 C2 C3 Z Q DQ LAMBD miu
global  A F P
%introduciendo variables
C1=input('Digite el valor de C1: ');
C2=input('Digite el valor de C2: ');
C3=input('Digite el valor de C3: ');
Z=input('Digite el valor de z: ');
n=input('Digite el numero de estados, n: ');
s=input('Digite el numero deseado de servidores, s: ');
LAMBD=input('Digite el valor de Lambda: ');
%Se crea la matriz Q
syms miu
Q=sym(zeros(n,n));
for i=1:n
   for j=1:n
      if (i==j && j~=n && j~=1)
         if (i<=s+1) % Se toma s+1 debido a que el valor de s se empieza a contabilizar a partir de
Q(2,2) esto porque en matlab no existe Q(0,0)
            Q(j,i)=-(j-1)*miu-LAMBD;
         else
            Q(j,i)=-s*miu-LAMBD;
         end
      elseif (i==n && j==n)
         Q(j,i)=-s*miu;
```

```
        elseif (i==1 && j==1)
            Q(j,i)=-LAMBD;
        elseif i==j-1
            if (j<=s+1)
                Q(j,i)=i*miu;
            else
                Q(j,i)=s*miu;
            end
        elseif i==j+1
            Q(j,i)=LAMBD;
        end
    end
end
% Se crea la matriz DQ
DQ=sym(zeros(n,n));
for i=1:n
    for j=1:n
        if (i==j)
            DQ(j,i)=Q(j,i);
        end
    end
end
% Se crean los vectores F y P
P=zeros(1,n);
F=zeros(n,1);
 for i=1:n
     F(i,1)=i;
     if i==1
         P(1,i)=1;
     else P(1,i)=0;
     end
 end
 % Se minimiza la función
 x0=(LAMBD/s)+10;
 lb=0;
[x,feval]= fmincon('fun',x0,[],[],[],[],lb,[],'nonlin');
```

```
clc
 disp('  ********************************************************************')
 fprintf(' *   El valor óptimo obtenido es %6f                    *',x);
 fprintf('\n *   Para el cual el valor mínimo de la función es %6f       *\n', feval);
 disp('  ********************************************************************')
 fprintf('\n\n')
 %vxg es la matriZ que contiene los valores de x y g obtenidos en
 %cada iteracion
```

A.4.2 Archivo fun.m

fun.m

```
% DISEÑO SISTEMA M/M/S - APLICACION ALGORITMOS
% FUNCION OBJETIVO
% GUILLERMO AUGUSTO VESGA ACEVEDO

function g=fun(x,s,A,Z,C1,C2,C3,F,P);
global s A Z C1 C2 C3 F P Q DQ
miu=x;
A=eval((DQ^(-1))*(DQ-Q));
g = C1*s+ C2*x + C3*(P*(A)^Z*F);
```

A.4.3 Archivo nonlin.m

nonlin.m

```
% DISEÑO SISTEMA M/M/S - APLICACION ALGORITMOS
% RESTRICCION NO LINEAL
% GUILLERMO AUGUSTO VESGA ACEVEDO

function [c,ceq]=nonlin(x,s,LAMBD)
global LAMBD s
c = (LAMBD/(s*x)) - 1;
ceq=[];
```

ANEXO B. EXPERIMENTOS REALIZADOS

B.1 EXPERIMENTOS ALGORITMO 1

B.1.1 Experimentos Escenario A1

3 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	1,0	0,147	0,897
0,1	5	1,0	0,289	0,688
0,1	20	0,9	0,344	0,678
0,1	35	0,5	0,345	0,678
0,1	50	0,5	0,345	0,678
0,1	100	0,5	0,345	0,678

4 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	1,4	0,171	1,046
0,1	5	1,4	0,319	0,736
0,1	20	1,3	0,390	0,717
0,1	35	0,5	0,391	0,717
0,1	50	0,5	0,392	0,717
0,1	100	0,5	0,392	0,717

5 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	1,4	0,181	1,136
0,1	5	1,4	0,327	0,753
0,1	20	1,6	0,405	0,728
0,1	35	0,5	0,407	0,728
0,1	50	0,5	0,408	0,728
0,1	100	0,5	0,408	0,728

10 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	4,7	0,188	1,249
0,1	5	2,5	0,330	0,762
0,1	20	2,3	0,412	0,732
0,1	35	1,2	0,415	0,732
0,1	50	0,6	0,416	0,732
0,1	100	0,6	0,416	0,732

15 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	4,5	0,189	1,252
0,1	5	3,6	0,330	0,762
0,1	20	3,4	0,412	0,732
0,1	35	1,6	0,415	0,732
0,1	50	0,7	0,416	0,732
0,1	100	0,7	0,416	0,732

20 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	5,7	0,189	1,252
0,1	5	4,9	0,330	0,762
0,1	20	4,8	0,412	0,732
0,1	35	2,3	0,415	0,732
0,1	50	0,9	0,416	0,732
0,1	100	0,8	0,416	0,732

50 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	26	0,189	1,252
0,1	5	25	0,330	0,762
0,1	20	24	0,412	0,732
0,1	35	10	0,415	0,732
0,1	50	3	0,416	0,732
0,1	100	3	0,416	0,732

100 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,1	2	317	0,189	1,252
0,1	5	236	0,330	0,762
0,1	20	304	0,412	0,732
0,1	35	200	0,415	0,732
0,1	50	99	0,416	0,732
0,1	100	23	0,416	0,732

B.1.2 Experimentos Escenario A2

3 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	0,8	0,267	1,077
0,2	5	1,1	0,439	0,963
0,2	20	0,5	0,473	0,960
0,2	35	0,5	0,473	0,960
0,2	50	0,5	0,473	0,960
0,2	100	0,5	0,473	0,960

4 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	0,7	0,322	1,262
0,2	5	1,2	0,519	1,056
0,2	20	0,8	0,575	1,049
0,2	35	0,5	0,576	1,049
0,2	50	0,5	0,576	1,049
0,2	100	0,5	0,576	1,049

5 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	1,1	0,345	1,373
0,2	5	1,5	0,545	1,091
0,2	20	1,0	0,617	1,079
0,2	35	0,6	0,618	1,079
0,2	50	0,5	0,618	1,079
0,2	100	0,5	0,618	1,079

10 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	2,5	0,365	1,539
0,2	5	1,4	0,545	1,091
0,2	20	1,7	0,644	1,094
0,2	35	0,7	0,646	1,094
0,2	50	0,5	0,647	1,094
0,2	100	0,7	0,647	1,094

15 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	3,8	0,365	1,547
0,2	5	3,5	0,558	1,115
0,2	20	2,5	0,644	1,094
0,2	35	0,9	0,646	1,094
0,2	50	0,6	0,647	1,094
0,2	100	0,6	0,647	1,094

20 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	5,0	0,365	1,547
0,2	5	4,9	0,558	1,115
0,2	20	3,4	0,644	1,094
0,2	35	1,1	0,646	1,094
0,2	50	0,7	0,647	1,094
0,2	100	0,7	0,647	1,094

50 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	23	0,365	1,547
0,2	5	23	0,558	1,115
0,2	20	16	0,644	1,094
0,2	35	4	0,646	1,094
0,2	50	2	0,647	1,094
0,2	100	2	0,647	1,094

100 ESTADOS				
λ	**z**	**Tiempo**	**μ***	**g(μ*)**
0,2	2	231	0,365	1,547
0,2	5	139	0,558	1,115
0,2	20	100	0,644	1,094
0,2	35	32	0,646	1,094
0,2	50	20	0,647	1,094
0,2	100	15	0,647	1,094

B.1.3 Experimentos Escenario B

3 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,0	0,219	0,831
0,1	5	1,0	0,219	0,831
0,1	20	1,0	0,287	0,804
0,1	100	0,4	0,288	0,804

4 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,6	0,106	1,567
0,1	5	1,3	0,250	0,911
0,1	20	1,2	0,334	0,862
0,1	100	0,5	0,338	0,862

5 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	2,4	0,120	1,802
0,1	5	1,8	0,259	0,946
0,1	20	1,9	0,351	0,880
0,1	100	0,5	0,356	0,880

10 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	4,4	0,134	2,322
0,1	5	3,0	0,264	0,971
0,1	20	3,0	0,359	0,888
0,1	100	0,6	0,367	0,888

15 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	6,0	0,134	2,361
0,1	5	4,3	0,264	0,971
0,1	20	4,6	0,359	0,888
0,1	100	0,7	0,367	0,888

20 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	8,6	0,135	2,365
0,1	5	6,2	0,264	0,971
0,1	20	6,7	0,359	0,888
0,1	100	0,9	0,367	0,888

50 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	39,3	0,135	2,365
0,1	5	30,5	0,264	0,971
0,1	20	33,9	0,359	0,888
0,1	100	3,4	0,367	0,888

B.1.4 Experimentos Escenario C

3 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,5	0,223	0,945
0,1	5	1,2	0,367	0,804
0,1	20	0,7	0,409	0,800
0,1	100	0,5	0,410	0,800

4 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,9	0,246	1,047
0,1	5	1,1	0,397	0,845
0,1	20	0,7	0,453	0,836
0,1	100	0,4	0,454	0,836

5 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,2	0,253	1,096
0,1	5	1,2	0,404	0,858
0,1	20	0,9	0,467	0,845
0,1	100	0,4	0,468	0,845

10 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	2,8	0,257	1,134
0,1	5	2,2	0,406	0,862
0,1	20	1,3	0,472	0,848
0,1	100	0,6	0,474	0,848

15 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	2,8	0,257	1,134
0,1	5	3,0	0,406	0,862
0,1	20	2,1	0,472	0,848
0,1	100	0,6	0,474	0,848

20 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	4	0,257	1,134
0,1	5	4	0,406	0,862
0,1	20	3	0,472	0,848
0,1	100	0,79	0,474	0,848

50 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	18,9	0,257	1,134
0,1	5	21,5	0,406	0,862
0,1	20	14,4	0,472	0,848
0,1	100	2,7	0,474	0,848

B.1.5 Experimentos Escenario D

3 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,7	0,160	0,637
0,1	5	0,8	0,246	0,584
0,1	20	0,4	0,266	0,582
0,1	35	0,4	0,266	0,582
0,1	50	0,4	0,266	0,582
0,1	100	0,4	0,266	0,582

4 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,4	0,172	0,636
0,1	5	0,9	0,254	0,588
0,1	20	0,6	0,273	0,587
0,1	35	0,7	0,273	0,587
0,1	50	0,6	0,273	0,587
0,1	100	0,6	0,273	0,587

5 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,7	0,174	0,635
0,1	5	1,0	0,255	0,589
0,1	20	0,4	0,274	0,587
0,1	35	1,3	0,274	0,587
0,1	50	0,4	0,274	0,587
0,1	100	0,4	0,274	0,587

10 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,0	0,175	0,635
0,1	5	1,3	0,255	0,589
0,1	20	0,4	0,274	0,587
0,1	35	*	*	*

15 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,3	0,175	0,635
0,1	5	1,6	0,255	0,589
0,1	20	0,6	0,274	0,587
0,1	35	*	*	*

20 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,6	0,175	0,635
0,1	5	2,5	0,255	0,589
0,1	20	0,6	0,274	0,587
0,1	35	1,1	0,274	0,588
0,1	50	*	*	*
0,1	100	*	*	*

50 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	9,5	0,175	0,635
0,1	5	14,4	0,255	0,589
0,1	20	2,5	0,274	0,587
0,1	35	*	*	*

B.1.6 Experimentos Escenario E

3 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,8	0,110	0,779
0,1	5	0,9	0,191	0,682
0,1	20	0,5	0,217	0,678
0,1	100	0,5	0,217	0,678

4 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,8	0,125	0,773
0,1	5	1,0	0,201	0,689
0,1	20	0,7	0,225	0,686
0,1	100	0,6	0,225	0,686

5 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,9	0,128	0,771
0,1	5	1,1	0,202	0,690
0,1	20	0,8	0,226	0,687
0,1	100	0,7	0,226	0,687

10 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,1	0,128	0,771
0,1	5	1,5	0,202	0,690
0,1	20	*	*	*

15 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,5	0,128	0,771
0,1	5	2,1	0,202	0,690
0,1	20	*	*	*

20 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	2,0	0,128	0,771
0,1	5	3,0	0,202	0,690
0,1	20	*	*	*

50 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	8,8	0,128	0,771
0,1	5	19,6	0,202	0,690
0,1	20	*	*	*

B.1.7 Experimentos Escenario F

3 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,1	0,223	0,736
0,1	5	0,9	0,310	0,699
0,1	20	0,4	0,324	0,698
0,1	100	0,5	0,324	0,698

4 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,8	0,234	0,736
0,1	5	0,9	0,317	0,703
0,1	20	0,6	0,330	0,702
0,1	100	0,7	0,330	0,702

5 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,7	0,235	0,736
0,1	5	1,0	0,318	0,703
0,1	20	*	*	*

10 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	0,9	0,235	0,736
0,1	5	1,3	0,318	0,703
0,1	20	*	*	*
0,1	100	*	*	*

15 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,1	0,235	0,736
0,1	5	2,0	0,318	0,703
0,1	20	*	*	*

20 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	1,4	0,235	0,736
0,1	5	2,8	0,318	0,703
0,1	20	*	*	*

50 ESTADOS				
λ	**z**	**Tiempo (s)**	**μ***	**g(μ*)**
0,1	2	5,9	0,235	0,736
0,1	5	14,1	0,318	0,703
0,1	20	*	*	*

B.2 EXPERIMENTOS ALGORITMO 2

B.2.1 Experimentos Escenario A1

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,143	0,891
	8	0,307	0,669
	14	0,332	0,675
	20	0,342	0,678
	26	**0,344**	**0,678**

3 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,143	0,891
	17	0,339	0,677
	32	**0,345**	**0,678**

3 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,1431	0,8908
	32	0,3447	0,6783
	62	**0,3448**	**0,6783**

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,102	0,961
	8	0,348	0,714
	14	0,379	0,719
	20	0,402	0,728
	26	**0,410**	**0,731**

10 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,102	0,961
	17	0,393	0,725
	32	0,413	0,732
	47	**0,416**	**0,732**

10 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,102	0,961
	32	0,413	0,732
	62	**0,416**	**0,732**

50 ESTADOS K = 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,102	0,961
	8	0,348	0,714
	14	0,379	0,719
	20	0,402	0,728
	26	**0,410**	**0,731**

50 ESTADOS K = 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,102	0,961
	17	0,393	0,725
	32	0,413	0,732
	47	**0,416**	**0,732**

50 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,102	0,961
	32	0,413	0,732
	62	**0,416**	**0,732**

B.2.2 Experimentos Escenario A2

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,265	1,076
	8	0,451	0,955
	14	0,470	0,960
	20	**0,473**	**0,960**

3 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,265	1,076
	17	0,472	0,960
	32	**0,473**	**0,960**

3 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,265	1,076
	32	0,473	0,960
	62	**0,473**	**0,960**

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,320	1,325
	8	0,581	1,077
	14	0,617	1,085
	20	0,637	1,092
	26	**0,643**	**1,094**

10 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,320	1,325
	17	0,630	1,090
	32	0,646	1,094
	47	**0,647**	**1,094**

10 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,320	1,325
	32	0,646	1,094
	62	**0,647**	**1,094**

50 ESTADOS K = 6			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,320	1,325
	8	0,581	1,077
	14	0,617	1,085
	20	0,637	1,092
	26	**0,643**	**1,094**

50 ESTADOS K = 15			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,320	1,325
	17	0,630	1,090
	32	0,646	1,094
	47	**0,647**	**1,094**
50 ESTADOS K = 30			
λ	**z**	**μ***	**g(μ*)**
0,2	2	0,320	1,325
	32	0,646	1,094
	62	**0,647**	**1,094**

B.2.3 Experimentos Escenario B

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	*	*
	8	0,239	0,788
	14	0,266	0,793
	20	0,283	0,802
	26	**0,287**	**0,804**

3 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	*	*
	17	0,277	0,799
	32	0,288	0,804
	47	**0,288**	**0,804**

3 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	*	*
	32	0,288	0,804
	62	**0,288**	**0,804**

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,116	1,928
	8	0,279	0,855
	14	0,310	0,855
	20	0,342	0,876
	26	0,355	0,883
	32	**0,361**	**0,886**

10 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,116	19281,000
	17	0,330	0,869
	32	0,361	0,886
	47	**0,366**	**0,888**

10 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,116	1,928
	32	0,361	0,886
	62	**0,367**	**0,888**

50 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,116	1,929
	8	0,279	0,855
	14	0,310	0,855
	20	0,342	0,876
	26	0,355	0,883
	32	**0,361**	**0,886**

50 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,116	19289,000
	17	0,330	0,869
	32	0,361	0,886
	47	**0,366**	**0,888**

50 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,116	1,929
	32	0,361	0,886
	62	**0,367**	**0,888**

B.2.4 Experimentos Escenario C

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,213	0,930
	8	0,382	0,793
	14	0,404	0,798
	20	**0,409**	**0,800**

3 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,213	0,930
	17	0,407	0,799
	32	**0,410**	**0,800**

3 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,213	0,930
	32	0,410	0,800
	62	**0,410**	**0,800**

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,223	1,000
	8	0,423	0,834
	14	0,452	0,841
	20	0,467	0,846
	26	**0,472**	**0,848**

10 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,223	1,000
	17	0,462	0,845
	32	0,473	0,848
	47	**0,474**	**0,848**

10 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,223	1,000
	32	0,473	0,848
	62	**0,474**	**0,848**

50 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,223	1,000
	8	0,423	0,834
	14	0,452	0,841
	20	0,467	0,846
	26	**0,472**	**0,848**

50 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,223	1,000
	17	0,462	0,845
	32	0,473	0,848
	47	**0,474**	**0,848**

50 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,223	1,000
	32	0,473	0,848
	62	**0,474**	**0,848**

B.2.5 Experimentos Escenario D

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,107	0,618
	8	0,250	0,577
	14	0,264	0,582
	20	**0,266**	**0,582**

3 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,108	0,618
	17	0,266	0,582
	32	**0,266**	**0,582**

3 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,107	0,618
	32	0,266	0,582
	62	**0,266**	**0,582**
10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,177	0,637
	8	0,259	0,583
	14	0,272	0,587
	*	*	*

10 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,177	0,637
	*	*	*

10 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,177	0,637
	*	*	*

50 ESTADOS K = 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,177	0,637
	8	0,259	0,583
	14	0,272	0,587
	*	*	*

50 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,177	0,637
	17	0,273	0,587
	*	*	*

50 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,177	0,637
	*	*	*

B.2.6 Experimentos Escenario E

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,109	0,780
	8	0,198	0,670
	14	0,213	0,677
	20	**0,217**	**0,678**

3 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,109	0,780
	17	0,216	0,678
	32	**0,217**	**0,678**

3 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,109	0,780
	32	0,217	0,678
	62	**0,217**	**0,678**

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,130	0,774
	8	0,208	0,679
	14	0,222	0,685
	*	*	*

10 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,130	0,774
	17	0,225	0,686
	*	*	*

10 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,130	0,774
	*	*	*

50 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,130	0,774
	8	0,208	0,679
	14	0,222	0,685
	*	*	*

50 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,130	0,774
	17	0,225	0,686
	*	*	*

50 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,130	0,774
	*	*	*

B.2.7 Experimentos Escenario F

3 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,226	0,738
	8	0,312	0,695
	14	0,324	0,698
	20	**0,324**	**0,698**

3 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,226	0,738
	17	0,324	0,698
	32	**0,324**	**0,698**

3 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,226	0,738
	32	0,324	0,698
	62	**0,324**	**0,698**

10 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,238	0,738
	8	0,319	0,699
	14	0,330	0,702
	*	*	*

10 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,238	0,738
	17	0,330	0,703
	*	*	*

10 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,238	0,738
	*	*	*

50 ESTADOS K= 6			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,238	0,738
	8	0,319	0,699
	14	0,330	0,702
	*	*	*

50 ESTADOS K= 15			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,238	0,738
	*	*	*

50 ESTADOS K= 30			
λ	**z**	**μ***	**g(μ*)**
0,1	2	0,238	0,738
	*	*	*