TRABAJO DE FIN DE GRADO

Grado en Estadística

Título: Evaluación de sistemas de líneas de transporte público urbano por simulación.

Autor: Nerea Calderón Mulero

Director: Esteve Codina Sancho

Departamento: Estadística

Convocatoria: Septiembre 2020

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Agradecimientos

Deseo agradecer especialmente a mi tutor de proyecto Esteve Codina Sancho por guiarme y acompañarme a lo largo de todos estos meses y ser capaz de transmitirme sus ideas con motivación haciendo de este trabajo una experiencia muy enriquecedora de valores y conocimientos.
Gracias.

Resumen

Este trabajo desarrolla un modelo de simulación para un conjunto de líneas de transporte público basado en una secuencia de cadenas de Markov conectadas entre sí. Este modelo de simulación es capaz de reproducir las interacciones entre colas de autobuses y colas de pasajeros en la secuencia de paradas de las líneas. El modelo de simulación se desarrolla utilizando el lenguaje de programación Python. Se tienen en cuenta varias distribuciones de probabilidad para las llegadas y salidas de pasajeros hacia y desde las unidades de transporte, y también se calculan las estadísticas de simulación convencionales para los sistemas de colas, correspondiente a los niveles de demanda del sistema y a los tiempos medios de espera para poder evaluar el funcionamiento del sistema.

Palabras claves: simulación, teoría de colas, transporte público urbano, fenómenos de espera, Python, R, cadenas de Markov

Clasificación AMS(MSC2010): 60K20 Applications of Markov renewal processes

60K25 Queueing theory

68U20 Simulation

90B22 Queues and service

Abstract

This work develops a simulation model for a set of public transport lines based on a sequence of connected Markov chains. This simulation model is able to reproduce the interactions between bus queues and passenger queues at the sequence of stops on the lines. The simulation model is developed using the Python programming language. Several probability distributions for the arrivals and departures of passengers to and from the transportation units are taken into account, and also the conventional simulation statistics for the queueing systems are calculated, corresponding to the levels of demand of the system and to the average waiting times so that the operation of the system can be evaluated.

Key words: simulation, queueing theory, urban public transport, waiting phenomena, Python, R, Markov chains

Índice general

Índice de figuras

Capítulo 1

Introducción

Actualmente vivimos en una sociedad muy acostumbrada y familiarizada al uso de las redes de transporte público urbano. Del mismo modo, la población es consciente de algunos de los posibles fenómenos de espera que se pueden producir cuando se hace uso de ellas. En general, se puede pensar que en un sistema de transporte público los únicos factores que generan colas y tiempos de espera son los clientes, pero lo cierto es que es un pensamiento erróneo. En ciertas circunstancias es posible que se lleguen a generar colas de unidades de transporte, lo que estaría produciendo dos tipos de lineas de espera simultáneas dentro de una misma red.

¿Cuánto tiempo estoy dispuesto a esperar a un autobús?, ¿Cuánto estoy dispuesto a pagar?, ¿En base a qué evalúo la calidad del servicio que me ha sido ofrecido?.

Cada una de estas preguntas obtendrá una respuesta diferente dependiendo de las exigencias de cada cliente. Generalmente, lo que el usuario busca es encontrar un servicio más o menos rápido que no le haga perder mucho tiempo entre esperas.

Es por esto que, el auge del uso de estos transportes ha desencadenado una preocupación entre los profesionales del ámbito ante la diversidad de fenómenos de espera que se producen y que pueden derivar en problemas más graves. Si una línea de transporte va muy congestionada, es muy probable que los pasajeros tengan que experimentar de media tiempos de espera muy altos. A su vez, esto puede ocasionar que algunos usuarios acaben prescindiendo de ese servicio, lo que conduce a una pérdida de fuentes de ingreso.

Hoy por hoy, los modelos de simulación son una herramienta fundamental para evaluar estos fenómenos de esperas. Además, son igual de válidos para redes de transporte en funcionamiento como para la construcción de futuras líneas o paradas. Estos modelos proporcionan un análisis global de la red que permite obtener una idea de como poder mejorarlas o incluso optimizarlas, pero no todos los programas evalúan de manera conjunta las colas de pasajeros y unidades de transporte que pueden producirse.

1.1. Objetivos del trabajo

Una de las principales motivaciones que tiene este proyecto es programar un simulador específico que para una red de transporte público urbano sea capaz de evaluar y reproducir las colas de pasajeros en un sistema de líneas de autobús. Al mismo tiempo, usando el mismo simulador, se busca reproducir la evolución de las colas de las unidades de transporte vinculadas a las colas de pasajeros. La base fundamental de este proyecto es implementar una simulación que sea capaz de reproducir adecuadamente las secuencias de colas conjuntamente y sujetas unas con otras.

El programa de simulación se integrará con modelos basados en programación lineal para calcular flujos medios en redes de transporte. Para evaluar el funcionamiento del sistema se crearán distintas situaciones de demanda, de esta manera se pondrá a prueba la mecánica de la red en función de su nivel de congestión.

Otra de las motivaciones de este trabajo es poder utilizar este simulador en la universidad como parte de un proyecto de mejora en modelos ya creados denominados de asignación, de los que se hablará más adelante en un capítulo de esta memoria. La idea es utilizar la simulación para ser capaces de asignar pasajeros a líneas de transporte público urbano, y así poder tener en cuenta todos los fenómenos de espera que se pueden producir bajo congestión. Actualmente se trabaja en modelos capaces de poder evaluar este tipo de situaciones, pero los modelos anteriores no pueden evaluar de forma consistente las interacciones entre los dos tipos de colas (pasajeros y unidades de transporte), especialmente si trata de redes urbanas de gran tamaño.

1.2. Justificación del trabajo

El principal motivo por el que he decidido realizar un trabajo con estas características es la fascinación que he ido desarrollando a lo largo de la carrera por el estudio de fenómenos de espera en teoría de colas y la programación de simuladores que reproduzcan fenómenos reales. En tercero del grado de Estadística hay una asignatura llamada Teoría de Colas, gracias a la cual me decidí por estudiar un tema relacionado con los tiempos de espera y las formaciones de colas.

En este trabajo se verán conceptos nuevos que no se vieron en la carrera, y se elaborarán programas con softwares con los que no se había trabajado anteriormente. Entre las motivaciones personales está el aprender nuevos lenguajes y métodos de programación. Con este proyecto se pretende explorar desde cero una nueva herramienta de programación como Python con la cual se diseñará el principal simulador.

Capítulo 2

Metodología

En este apartado se hará un breve resumen de la estructura que presenta la memoria. Además, se explicarán las técnicas estadísticas utilizadas durante el trabajo y se hará un compendio de la utilidad que ha desempeñado cada uno de los recursos informáticos utilizados.

2.1. Estructura de la memoria

La memoria se divide en tres grandes bloques. En la primera parte se expondrán los principales conceptos de la teoría de colas y adicionalmente, se describirá la estructura de las redes de transporte público urbano. También habrá un apartado donde se explicarán los métodos comentados anteriormente sobre asignación de pasajeros a líneas de transporte. En otro se detallarán elementos de simulación previos a la construcción del simulador que son necesarios para diseñar el código, como la generación de variables aleatorias para algunas distribuciones y la validación de la aleatoriedad de estas.

En la segunda parte del trabajo se realiza la implementación del simulador, se describen todas las variables utilizadas explicando su función y se detalla la secuencia del algoritmo que se programa. Además, se describirán todos los estadísticos necesarios para realizar los cálculos para cada sistema de espera y la manera de obtenerlos.

Por último se presentarán los resultados obtenidos con la ejecución del simulador y se comentarán las conclusiones extraídas de su evaluación para diferentes situaciones, así como una conclusión global de todo el trabajo.

2.2. Recursos informáticos

Para el desarrollo de la parte práctica del proyecto se ha hecho uso de tres programas informáticos que, además, son muy conocidos en el mundo de la estadística: Sistema

AMPL, Python y R.

2.2.1. Sistema AMPL

El sistema AMPL se ha utilizado para obtener los datos de volúmenes de flujo de pasajeros que se emplean para evaluar la congestión de la red del simulador que se programa.

El código AMPL utilizado para obtener los ficheros de datos que alimentan el programa de simulación desarrollado en este trabajo, se extrae de una práctica que se realiza en Ingeniería de Caminos donde a los alumnos se les presenta un fichero de AMPL y se les explica cómo ejecutarlo y cómo mostrar los resultados, además de tener que resolver un determinado número de cuestiones a parte.

El modelo de programación lineal que AMPL resuelve es el siguiente :

$$f = (\,..., f_a, ...;\ a \in \hat{E}(b),\ b \in \hat{N}_G\,)$$

$$\begin{array}{lll}
Min_{\ v,w} & \displaystyle\sum_{q\in D}\left(\sum_{a\in A} t_a v_a^q + \sum_{b\in \hat{N}_G} w_b^d\right) & \\
s.a: & & \\
q \in D: & \displaystyle\sum_{a\in E(i)} v_a^q - \sum_{a\in I(i)} v_a^q = \begin{cases} -\sum_{(p,q)\in W} g_{pq} & \text{if } i = q, \\ g_{iq}, & \text{if } i \neq q,\ (i,q)\in W \\ 0, & \text{if } (i,q)\notin W \end{cases} & i \in N, \\
 & v_a^q \le f_a w_b^q, & a \in \hat{E}(b),\ b \in \hat{N}_G, \\
[\mathrm{PL}](f) & v_a^q \ge 0 & a \in A
\end{array}$$

Los volúmenes de flujo $(v_a, v_y, v_e...)$ que se utilizan en el simulador y que se detallarán más adelante en el capítulo 7, provienen de resolver este modelo de programación lineal. La ejecución de este programa devuelve un archivo con datos que se utilizarán para probar la simulación y extraer los cálculos estadísticos que se mostrarán en la parte de resultados.

2.2.2. Python

Python es una herramienta informátic muy potente y versátil, además posee una gran facilidad en la lectura del código, detalle muy importante a tener en cuenta cuando se trabaja con códigos de programación de muchas líneas y con muchas sentencias.

Con él se ha realizado la definición de todas las variables y se ha construido y programado el algoritmo del simulador del proyecto. También se han construido las funciones que realizan los cálculos de los estadísticos de las colas de espera y se han realizado los

gráficos de las congestiones que se comentan en los resultados. Además, este programa también se ha utilizado para construir funciones que simulan métodos de generación de variables para algunas distribuciones, entre las que destacan la Poisson y la Binomial, ambas necesarias para implementar algunas de las sentencias del código.

Se ha decidido emplear este programa porque computacionalmente es muy potente y eficaz en comparación con otros, y a la hora de realizar una simulación tan compleja es importante trabajar con un recurso informático capaz de ejecutar el programa de la manera más rápida y eficiente posible. En este caso Python ha facilitado la fluidez en la ejecución del código y ha permitido poder tratar con muchas sentencias introducidas a su vez en diferentes bucles al mismo tiempo.

2.2.3. R

El mejor programa que se puede usar cuando se quiere hacer un análisis estadístico avanzado es R. Este software presenta una herramienta muy potente en visualización de información y de datos. En este programa se han pasado los Tests de las muestras de todas las distribuciones generadas en Python para comprobar si realmente estaban bien generadas y cumplían sus especificaciones. Al trabajar con distintas distribuciones estadísticas, R facilita las instrucciones necesarias para llevar a cabo la ejecución de cada una de ellas y, a su vez, poderlas contrastar gráficamente además de con tests estadísticos.

Con él también se ha construido un programa que simula un test de independencia para evaluar la aleatoriedad de una muestra de datos. En este caso se ha utilizado R por su gran capacidad en análisis estadístico. Es muy útil a la hora tratar con probabilidades y cálculos estadísticos con el propósito de comprobar ciertos contrastes de hipótesis.

Capítulo 3

Conceptos básicos de teoría de colas

En algún momento en la vida real todos hemos observado e incluso hemos sido partícipes de un fenómeno muy habitual como son las formaciones de colas o, dicho de otra manera, líneas de espera. Este fenómeno ocurre cuando un cliente llega a un lugar buscando recibir un servicio. Este servicio necesita de al menos un servidor para poder atender a los clientes, el que a su vez dispondrá de una determinada capacidad de atención. Si se da el caso que hay más clientes esperando a recibir su servicio que servidores capaces de atenderles, se producirá una cola de clientes esperando a ser atendidos. El conjunto que forman los clientes que hacen cola y los sistemas de servicio recibe el nombre de sistema de espera.

El estudio matemático de estos sistemas de espera se conoce comúnmente como teoría de colas. A continuación se presentarán los conceptos básicos que se han de tener en cuenta si se quiere hacer un estudio con estas características (Codina y Montero, 2020).

3.1. Descripción de los componentes

La estructura del sistema de espera se puede representar como en la figura 3.1. Se diferencian claramente tres partes: la población, la cola y el sistema de servicio.

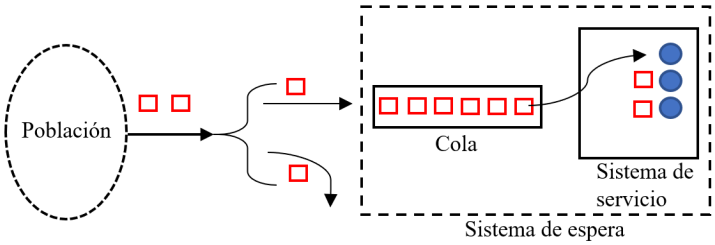

Figura 3.1: Estructura sistema de espera

3.1.1. Población

La población se compone por los elementos (denominados clientes), que solicitan de un servicio por parte del sistema de servicio o servidor. Esta población se puede visualizar como una fuente de clientes, esto quiere decir que en algún momento estos abandonarán la población (de uno en uno, o en lotes) para pasar a formar parte del sistema de espera donde acabarán recibiendo su servicio.

Es común observar como los clientes abandonan la población en instantes de tiempo diferentes entre si y de uno en uno, pero no siempre se produce del mismo modo. En algunos servicios, como por ejemplo en la subida de clientes a un transporte público urbano, los clientes abandonan la población conjuntamente, y se dice que la gestión del servicio se ha producido por lotes de clientes.

La población también puede clasificarse como finita o infinita. Diremos que una población es finita si siempre hay un número N de clientes formando parte de ella. En este caso, el usuario después de salir del sistema de espera vuelve a formar parte de la población. En cambio, se hace referencia a una población infinita si el cliente, después de salir del sistema de espera, desaparece y no vuelve a formar parte de la población.

3.1.2. Cola

Las colas son modelos de sistemas reales y pueden representar tanto a clientes como maquinaria, sistemas logísticos o flujos de trabajo. La cola forma parte del sistema de espera y crece a medida que va incrementando el número de clientes de la población que esperan por recibir su servicio.

La formación de las colas se produce cuando la capacidad de un servidor por atender a los clientes no es lo suficientemente grande como para poder abastecer la demanda de todos los usuarios al mismo tiempo. En este caso los clientes que no pueden ser atendidos porque ya hay otros que están recibiendo su servicio, deben esperar en la cola hasta poder ser atendidos.

Una cola se puede caracterizar principalmente por dos cosas: la capacidad en número de clientes que puede soportar, ya sea finita o infinita, y el orden en el que los clientes son servidos. El número de usuarios que pueden entrar por unidad de tiempo se define como la tasa de entrada, y viene caracterizado por la distribución de entradas y si la entrada de clientes se produce en lotes o no.

En cuanto al orden de llegada de los clientes, se pueden clasificar en cola de un único

servidor o múltiples y por tipo de disciplina: FIFO, LIFO, RSS, Priority, WFQ, así como por la tasa de abandono.

Los dos tipos de disciplinas más comunes son FIFO y LIFO :

- **FIFO**.*(First-in First-out)*, el primero que llega el primero que sale. En este tipo de disciplina se atiende por orden de llegada.
- **LIFO**.*(Last-in First-out)*, el último que llega el primero que sale. Esta disciplina se utiliza mucho en el registro de inventarios de productos, donde se compra primero el producto que se ha vendido el último.

En caso de no especificarse, normalmente se asume la disciplina FIFO.

3.1.3. Proceso de llegadas de los clientes

Las llegadas de los clientes se producen en general estocásticamente, es decir, de manera aleatoria. Es por esto que el tiempo entre las llegadas de dos clientes dependen de una variable aleatoria que se rige por una ley de distribución en particular, la cual se determina en el estudio del sistema de espera. Se debe de tener en cuenta que los clientes pueden llegar independientemente o en lotes. En cualquiera de los casos se debe definir la distribución probabilista que mejor los define.

Bajo la hipótesis más común, la variable aleatoria del número de clientes que salen de la población durante un determinado horizonte de tiempo se rige por la ley de Poisson. También se puede interpretar como que el intervalo de tiempo entre dos salidas de clientes sigue una ley Exponencial.

3.1.4. Sistema de servicio

El sistema de servicio es la parte del sistema de espera donde los servidores atienden a los clientes que están esperando en la cola a ser servidos. Este servicio puede constar de uno o varios servidores y estos a su vez pueden estar dispuestos en serie o paralelamente.

Cuando el cliente va a ser servido, este sale de la cola de espera y entra en el sistema de servicio. En el momento en que el cliente es servido, este abandona el sistema de espera y otro cliente que aguarda en la cola pasa a ser servido. En el caso que la cola esté vacía, cuando se produzca la llegada de un cliente este será atendido inmediatamente.

En esta parte es interesante tener en cuenta el tiempo que conlleva la realización del servicio. Este tiempo se describe como una variable aleatoria y, en los casos más comunes, su distribución se rige según la ley Exponencial constante o K-Erlang.

3.1.5. Descripción magnitudes

En este apartado se explicará y detallará la terminología y notaciones necesarias para entender los conceptos más importantes en sistemas de servicios y teoría de colas.

Notación de Kendall-Lee

Inicialmente se van a describir los conceptos más utilizados en los sistemas de servicios, para ello se utiliza la notación de Kendall-Lee:

$$F_{Y_{(y)}}/F_{S_{(s)}}/s/K/N$$

1 Los tiempos entre llegadas sucesivas de clientes $(Y_1, Y_2, Y_3, ...)$, se representan como variables aleatorias independientes e idénticamente distribuidas con función de probabilidad $F_{Y_{(y)}}$.

2 Los tiempos de servicio por cada cliente $(S_1, S_2, S_3, ...)$, se representan como variables aleatorias e idénticamente distribuidas con función de probabilidad $F_{S_{(s)}}$.

3 El número de servidores disponibles se representa con s. Se supone que estos servidores son idénticos entre ellos. Los tiempos de servicio son S.

4 La capacidad de la cola es K y puede ser finita o infinita. En caso de no especificarse se asume que es infinita (no hay un número máximo de clientes disponibles esperando en la cola).

5 La población se define como N y puede ser finita o infinita. Si no aparece se entiende que es infinita.

A la hora de describir los modelos de colas es fundamental saber identificar las distribuciones que siguen los tiempos entre llegadas de clientes y los tiempos de servicio. En términos de notación, cuando se escribe un modelo se pueden especificar las distribuciones que siguen cada uno de estos eventos de la siguiente manera:

- M Para la distribución exponencial.
- D Para distribución con tiempo entre llegadas o de servicio constante: $E[\tau] = T, Var[\tau] = 0$.
- E_K Para la distribución K-Erlang.

- G Esta notación sirve para representar una distribución cualquiera (tiempos correlacionados entre sí o no).
- GI Es una notación similar a la de antes pero, a diferencia, esta representa una distribución general de tiempo entre llegadas o de servicio independiente.

Por lo tanto, si se quiere hacer referencia a un modelo con tiempos entre llegadas y tiempos de servicios exponenciales y de más de un servidor, la manera correcta de representarlo sería: M/M/s.

Terminología y Notación.

Para describir las magnitudes habituales en teoría de colas, es preciso considerar una variable t como el tiempo desde que el sistema de espera empieza a funcionar hasta que pasa a detenerse. A continuación, teniendo en cuenta esta nueva variable, se describirán las principales magnitudes habituales en teoría de colas:

$N(t)$ es el número de usuarios en el sistema de espera o estado del sistema en el instante de tiempo t.

$P_n(t)$ supone la probabilidad de encontrar en el sistema de espera n clientes en el instante t.

λ_n es el parámetro que hace referencia al número esperado de llegadas de clientes al sistema de espera por unidad de tiempo cuando ya hay n clientes esperando en el sistema. Si esta tasa es constante para toda n, es decir, que no afecta al número esperado de llegadas , entonces se interpreta como λ.

μ_n es el número esperado de clientes que acaban el servicio por intervalo de tiempo cuando hay n clientes en el servicio. Si μ_n es constante para toda n, entonces el número medio de servicios por unidad de tiempo es μ. En el caso que n sea superior al numero de servidores s entonces $\mu_n = s * \mu$.

Un sistema de espera cuando empieza su funcionamiento se encuentra en un régimen transitorio, es decir, que se desvanece con el paso del tiempo. Cuando ya ha pasado un tiempo suficiente, normalmente pasa a ser un régimen estacionario, esto quiere decir que las variables que influyen al sistema se mantienen imperturbables a lo largo del tiempo. En este último régimen se pueden diferenciar las siguientes notaciones:

$P_n \rightarrow$ Probabilidad de que el estado del sistema sea de n clientes.
$L \rightarrow$ Nombre esperado de clientes en el sistema de espera.
$L_q \rightarrow$ Nombre esperado de clientes en la cola.

$W \rightarrow$ Tiempo medio de espera en el sistema, incluyendo el tiempo de servicio.
$W_q \rightarrow$ Tiempo medio de espera en la cola del sistema.

Una vez descritas las notaciones y terminologías propias de teorías de colas, es necesario conocer las magnitudes de interés de ser calculadas en un sistema de espera. Se pueden encontrar magnitudes que hacen referencia a tres tipos de clases diferentes: las relacionadas con el estado del sistema de espera, las relacionadas con el tiempo de espera por cliente en el sistema de espera y las relacionadas con la tasa temporal de clientes que entran en el sistema de espera.

Estas magnitudes también se pueden clasificar según si son:

- Magnitudes instantáneas. Son variables aleatorias dado el carácter estocástico de los sistemas.
- Magnitudes en régimen estacionario o para $t = \infty$.
- Magnitudes que son esperanzas matemáticas del tipo anterior.

Magnitudes de un sistema de espera

En este subapartado se examinarán un conjunto de magnitudes comunes a cualquier tipo de sistema de espera. Con esto, el objetivo es dar a conocer las *fórmulas de Little*, que se informarán más adelante.

- τ_n: Tiempo entre las llegadas de dos clientes consecutivos (el n y el $n+1$).
- w_n: Tiempo de permanencia en el sistema de espera del cliente n.
- w_{q_n}: Tiempo de permanencia en la cola del cliente n.
- x_n: Tiempo de servicio del cliente n.
- $N(t)$: Número de clientes en el sistema de espera en el instante t.
- $N_q(t)$: Número de clientes en la cola en el instante t.

Fórmulas de Little

Las anteriores magnitudes en general son aleatorias, y se ha comprobado que verifican las denominadas *fórmulas de Little*; relaciones matemáticas que fueron demostradas por John Little en 1961.

$$L = \bar{\lambda}W; \qquad L_s = \bar{\lambda}W_s; \qquad L_q = \bar{\lambda}W_q; \qquad L = L_s + L_q; \qquad W = W_s + W_q$$

La *Ley de Little*, definida con la primera fórmula que acabamos de ver, establece que el número promedio de clientes en un sistema (L) es igual a la tasa promedio de llegada de los clientes al sistema ($\bar{\lambda}$) multiplicada por el tiempo promedio que un cliente permanece en el sistema (W). La fórmula es igual de válida para sistemas como para subsistemas, de ahí fórmulas cómo $L_q = \bar{\lambda}W_q$, donde L_q es el número promedio de clientes que esperan en la cola y W_q el tiempo promedio que un cliente espera en la cola.

Descripción de los modelos de colas más conocidos

Los modelos que se presentarán a continuación son modelos de colas que basan sus teorías en procesos denominados de nacimiento y muerte. En el contexto de teoría de colas el término de nacimiento hace referencia a las llegadas de clientes que se producen al sistema de espera, mientras que el de muerte indica el de las salidas. Este proceso describe de manera probabilística como evoluciona el estado del sistema a lo largo del tiempo.

En este tipo de modelos generalmente hse asume que las distribuciones de probabilidad del tiempo hasta la próxima llegada (nacimiento) y la del tiempo hasta la próxima salida (muerte) son exponenciales de parámetros λ_n y μ_n respectivamente.

Modelo M/M/1

El modelo M/M/1 representa al tipo de sistema de espera más simple que existe. Sus principales características son:

1. Los tiempos entre las llegadas de los clientes son independientes e idénticamente distribuidos según una ley exponencial, identificada en el modelo como la primera M, de parámetro $\lambda_n = \lambda$.

2. Los tiempos de servicios son independientes e idénticamente distribuidos según una ley exponencial, identificada en el modelo como la segunda M, de parámetro $\mu_n = \mu$.

3. El sistema presenta un único servidor (s=1).

Normalmente estos modelos son representados mediante diagramas de transición que identifican las tasas de llegadas y tiempos de servicio por cliente. El diagrama de las tasas de transición en el modelo M/M/1 queda esquematizado en la figura 3.2 .

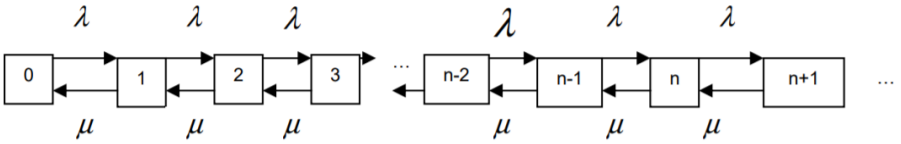

Figura 3.2: Diagrama de tasas de transición en el modelo M/M/1

En esta figura se muestran las posibles transiciones permitidas en el estado del sistema en cada instante. Cada una de esas etiquetas indica la tasa media de cada transición: número medio de ocurrencias por unidad de tiempo.

Si el cociente entre la tasa media de llegadas y la tasa media de salidas ρ, el cual recibe el nombre de factor de carga, es inferior a la unidad ($\frac{\lambda}{\mu} < 1$), entonces el sistema de espera asume un régimen estacionario.

El modelo M/M/s es parecido al sistema anterior, pero en este caso el sistema de espera dispone de más de un único servidor. Dadas las nuevas características, el diagrama de transición se representa como en la siguiente figura:

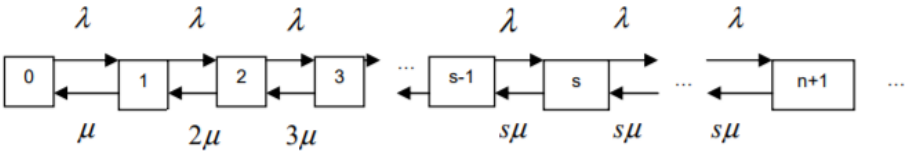

Figura 3.3: Diagrama de tasas de transición en el modelo M/M/s

En esta situación el factor de carga es igual a $\frac{\lambda}{s\mu}$, de ahí que se multiplique el número de servicios en funcionamiento por el parámetro μ entre las transiciones de clientes.

Modelo M/M/s/K

Este modelo de más de un único servidor es una ampliación del modelo anterior. La diferencia es que en estas condiciones se le añade una nueva restricción: la capacidad de la cola del sistema es limitada (K).

En este caso especial, el concepto de nacimiento y muerte queda condicionado por las limitaciones de la cola. En el momento que se complete el número total de clientes permitidos esperando en la cola, no podrán permanecer nuevos usuarios en ella hasta que se produzca, como mínimo, la salida de un cliente.

El diagrama de transiciones para las tasas en este modelo se representa como en la figura 3.4.

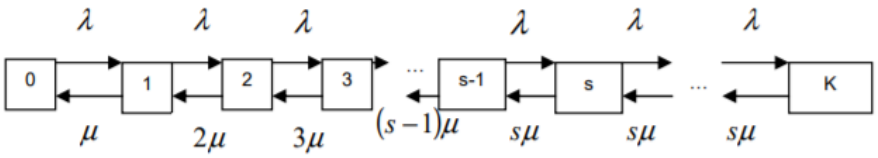

Figura 3.4: Diagrama de tasas de transición en el modelo M/M/s/k

En este caso el sistema de espera se define mediante un régimen estacionario.

Modelo M/G/1

El modelo M/G/1 es igual al M/M/1 en cuanto a las tasas de llegadas de los clientes y número de servicios. Se diferencian principalmente por la distribución que siguen los tiempos de servicio. En el ejemplo que se había visto anteriormente, estos tiempos se distribuían siguiendo una ley Poisson o, dicho de otra forma, de manera exponencial. El caso M/G/1 es un modelo de colas del tipo no exponencial donde los tiempos de servicio siguen una distribución general con esperanza $\frac{1}{\mu}$ y varianza σ^2.

En este tipo de modelos, para demostrar que el sistema es estacionario tan solo se ha de cumplir que el factor de carga ρ sea inferior a la unidad .

El diagrama de transición de las tasas de llegadas y de servicios se representa de la misma manera que un modelo M/M/1.

3.2. Tipos especiales de colas

Normalmente cuando se habla de esperar en una cola a la mayoría de personas les viene a la mente un sistema donde cada servidor atiende a un cliente a la vez. Un sistema con más de un canal o más de un servicio dispone de un cierto numero de canales paralelos y puede dar servicios a un número determinado de clientes a al mismo tiempo. Pero lo cierto es que existen otros tipo de servicios. En esta práctica se tratara con más profundidad el

tipo de cola de servicio que se produce por lotes, que es el tipo de servicio que representa a la mayoría de redes de transporte público urbano.

3.2.1. Colas de servicios por lotes

Como bien es conocido, los tiempos de llegadas de los clientes a los servicios se producen de manera estocástica. Para evitar problemas de congestión, es posible que a la hora de ser atendidos se contemple un servicio por lotes, es decir, un grupo de usuarios que espera en la cola a ser atendidos es servido conjuntamente y a la vez por un mismo servidor. La cantidad de usuarios que son atendidos por lotes dependerá, entre otras cosas, de la capacidad de atención que tenga el servicio y, además, puede tratarse de cantidades fijas o variables. Si las cantidades son fijas siempre se atiende a un número establecido de usuarios dentro de un mismo lote. En este caso, el tamaño de lote es otra variable aleatoria más del sistema y como cualquier otra se ha de definir su distribución de probabilidad.

En el concepto de red de transporte público urbano que se presenta en este artículo, los modelos de colas que reproducen este tipo de servicio pueden ser representados de la siguiente forma: $M/G^{[a,b]}/s$. En este modelo se establece que el servicio será atendido cuando al menos hayan a unidades en la cola de servicio y la capacidad máxima de atención que presente sea de b. Si el servicio está disponible pero el lote de clientes no supera las a unidades, el servicio esperará hasta alcanzar esa magnitud. Lo mismo sucede si el lote de usuarios excede la capacidad b, en este caso los últimos usuarios en llegar a la cola y sobrepasen esa limitación deberán esperar a ser atendidos en el siguiente lote. Si el número de clientes a ser atendidos se encuentra en el intervalo $[a, b]$, todos los usuarios presentes en la cola serán atendidos en un mismo lote.

En el caso de una red de transporte público, los pasajeros son atendidos en conjunto cuando la unidad de transporte llega a la parada. En estos casos no es usual encontrarse un límite inferior de cantidad de pasajeros, en cambio si existe un límite superior y este coincide con la capacidad que albergue la unidad en ese momento.

3.2.2. Fórmula de Powell

La fórmula de Powell (Powell, 1986) se basa en la implementación de un método de aproximación para modelos de colas del tipo $M^{[X]}/E_k^{[V]}/1$.

Descripción de magnitudes:

1. V - se describe como una variable aleatoria discreta y corresponde a la capacidad disponible del servidor. La capacidad máxima se indica como K, de forma que $V \leq K$; $v = E[V]$; $C_v = \sigma_v / v$.

2. S - se describe como una variable aleatoria continua y representa el tiempo entre las llegadas consecutiva de dos servicios. Su esperanza matemática es $E[S] = 1/\mu$, $C_s = \mu\sigma_s$

3. τ - se describe como una variable aleatoria continua y representa el tiempo entre llegadas consecutivas de lotes de pasajeros. Su esperanza matemática es $E[\tau] = 1/\lambda$.

4. X - se describe como una variable aleatoria discreta y representa el número de pasajeros que hay por lote. $x = E[X]$; $C_x = \sigma_x / x$.

5. Y - se describe como una variable aleatoria discreta y representa número de clientes en un ciclo. Su esperanza matemática, El número medio de clientes que llegan en un ciclo, también representado como la esperanza matemática de Y, es: $E[Y] = \frac{\lambda x}{\mu}$

$$Var[Y] = (\lambda x)^2 Var[S] + \frac{\lambda}{\mu}(Var[X] + x^2); \quad C_Y^2 = \frac{Var[Y]}{E^2[Y]} = C_s^2 + \frac{\lambda}{\mu}(1 + C_X^2)$$

El factor de carga de la cola es $\rho = \dfrac{\lambda x}{\mu v}$. Si $\rho < 1$ se dice que el sistema tiene estado estacionario.

Llegadas de clientes observadas por el servidor

El número total medio $\tilde{x}$ de clientes que llegan por lotes es:

$$\tilde{x} = \frac{(x(1 + C_x^2) - 1)}{2}$$

El número total medio $\tilde{y}$ de clientes que llegan observador por el ciclo es:

$$\tilde{y} = \frac{1}{2}\left(\frac{\lambda x}{\mu}(1 + C_S^2 + \frac{\mu}{\lambda}(1 + C_x^2)) - 1\right)$$

Ocupación máxima en promedio

Si $\hat{Q}$ es el valor medio de la cola máxima que se da (es decir, la que hay justo antes de que llegue un servicio), entonces:

$$\hat{Q} = \frac{v}{2}\left\{\frac{1+C_v^2}{1-\rho} + \frac{\rho^2(C_s^2-1)}{1-\rho}\right\} + \frac{x\rho(1+C_x^2)}{2(1-\rho)} + \frac{1}{2} - K + \epsilon(v\,\rho)$$

El término de corrección $\epsilon(v,\rho)$ es importante tenerlo en cuenta para valores de ρ bajos,

$$\epsilon(v,\rho) = a + (f + b\rho + g\rho^2)v + c(\rho v)^{\frac{1}{2}} + eK + d\breve{Y}^{\frac{1}{3}}$$

$$\begin{aligned} &a = -0{,}4358, \quad b = 0{,}6804, \quad c = -0{,}8862 \\ &d = 0{,}4155, \qquad e = 0{,}9925, \quad f = -0{,}4775, \quad g = -0{,}1892 \end{aligned}$$

($\breve{Y}$ es el momento tercero del número total de clientes llegados por ciclo y generalmente es desconocido).

Ocupación media L y demora W por cliente

$$L = \hat{Q} + \rho v\left\{\frac{(1+C_s^2)}{2} - 1\right\}$$

$$W = \frac{L}{\lambda x} = \frac{1}{\mu}\left\{\frac{\hat{Q}}{\rho v} - 1\right\} + \frac{1}{2\mu}\left\{1 + C_s^2\right\}$$

Se ha de tener en cuenta que : $\ell im_{\rho\to 0+}\dfrac{\hat{Q}}{\rho} = v$ y, por tanto, para $\rho \approx 0$ és $W \approx \dfrac{1}{2\mu}\left\{1 + C_s^2\right\}$

Cola residual aproximada

Valor medio de la cola mínima, es decir, valor medio de las colas que quedan inmediatamente después de irse el servidor.

$$R_L = L - \frac{\rho v}{2}\left(1 + \text{`}C_s^2\right)$$

Aproximación con fórmula de Powell

La formula de Powell es muy compleja y tiene una gran cantidad de parámetros de entrada, como los coeficientes de desviación de los tiempos de servicios, los de las llegadas, la capacidad del servicio, el factor de carga, etc. Bajo unos determinados valores para estos inputs, se han generado los siguientes gráficos aproximados:

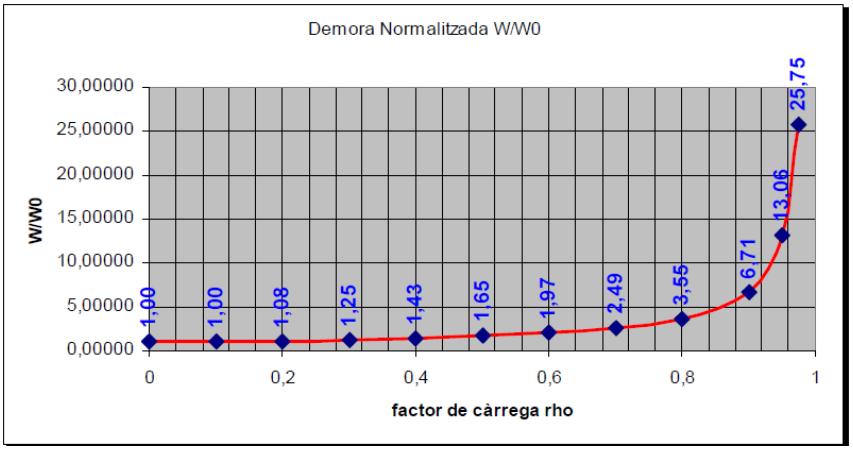

Figura 3.5: Fórmula de Powell - Demora normalizada W/W0

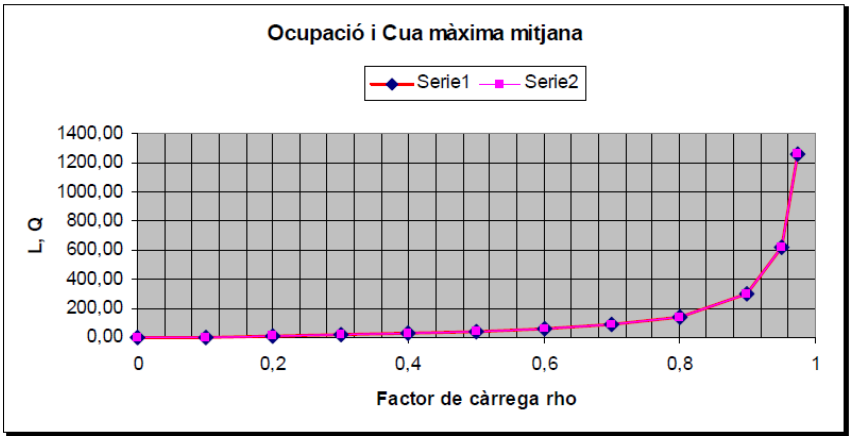

Figura 3.6: Fórmula de Powell - Ocupación y cola máxima media

Estos gráficos están relativamente aproximados para los valores provenientes de las simulaciones implementadas, y dan una subestimación de los tiempos de espera. Dicho en otras palabras, la fórmula de Powell subestima los valores de la demora. Esto reafirma la idea de crear un simulador especial que evalúe de manera correcta estas situaciones.

Capítulo 4

Sistemas de transporte público urbano

Cuando se habla de transporte público se hace referencia al término aplicado al transporte colectivo de pasajeros. A diferencia del transporte privado, el transporte público está generalmente regulado por administraciones locales y establece un conjunto de normas horarias y rutas fijas establecidas previamente. Este tipo de transporte engloba diferentes medios de desplazamiento como pueden ser buses, metros, trenes, etc.

Como bien se ha comentado anteriormente, en este trabajo se realizará una simulación de teoría de colas sobre una red de transporte público urbano. Con este fin, se ha creado este capítulo para poder detallar todos aquellos elementos que conforman este tipo de red de transporte y los fenómenos de espera que se pueden producir. Todos estos aspectos son importantes para, posteriormente, poder entender mejor la parte práctica de este trabajo.

4.1. Elementos de la estructura

Un sistema de transporte se compone de tres elementos físicos principales: el vehículo, la infraestructura y la red de transporte. En esta práctica nos centraremos principalmente en entender la estructura del sistema de transporte de autobuses y metros.

La red de transporte está compuesta por las diferentes rutas por las que se desplazan los medios de transporte. Cada ruta recibe el nombre de *línea* de transporte. Cada línea esta formada por un número determinado de paradas por las que pasará un bus o metro a realizar su servicio. Por lo tanto, si hablamos de una red de transporte en autobuses, esta se divide en distintas líneas que constituyen las diferentes rutas que pueden tomar los autobuses y, a su vez, cada línea esta segmentada en distintas paradas. Cada autobús tiene una ruta determinada, por lo tanto cada servicio recorrerá toda una línea pasando por todas sus paradas.

4.2. Fenómenos de espera

En los sistemas de transporte público se pueden identificar dos tipos de colas. El tipo de cola más común es la formada por los clientes. Este fenómeno se produce cuando los usuarios que desean utilizar el servicio de transporte público esperan en una parada a ser atendidos. En momentos de tiempo con mucha demanda de pasajeros es posible que cuando un autobús llegue a la parada, teniendo en cuenta los pasajeros que continúan en el bus y los que terminan ahí su servicio, el autobús no disponga de capacidad suficiente como para atender a todos los usuarios que estaban esperando poder subir. Al no poder recogerlos a todos, algunos de ellos deben permanecer esperando en la parada a la llegada del siguiente autobús, creándose una cola de pasajeros esperando ser atendidos, algunos de los cuales puede que tengan que esperar a más de una llegada de las unidades de transporte para poder encontrar un autobús en el que poder subir.

Pero no solo se han de tener en cuenta las colas de espera que pueden producir los clientes, si no también las que producen los autobuses. Primero de todo hay que tener en cuenta que este tipo de transporte depende de muchos factores externos, como el tráfico, el tiempo, o incluso el día del mes, lo que puede condicionar el tiempo que le lleve al autobús de realizar su servicio y le introduce estocasticidad. También puede pasar que un autobús tarde mucho tiempo en realizar su servicio, ya sea por la cantidad de pasajeros que suben y bajan o por si se producen ciertas circunstancias, como la llegada de nuevos pasajeros mientras el bus está apunto de marcharse, que hagan que el servicio demore mucho tiempo y llegue con retraso a la próxima parada. En cualquiera de estos casos, pueden coincidir al mismo tiempo más de un autobús en la misma parada, por lo tanto, el servicio de detrás debe esperar a que el de delante finalice su servicio para poder incorporarse a la plataforma. A su vez, esto provoca que el segundo autobús realice su servicio con retrasos en sus llegadas.

Capítulo 5

Concepto de asignación de pasajeros a líneas de transporte publico

Cuando se diseña una red de transporte público y se postula la idea de añadir una nueva linea de autobús, se han de tener en previo conocimiento cosas como el trazado, las secuencias de paradas que va a tener, por donde van a transcurrir el recorrido de las unidades de transporte, cuanta gente vive por la zona o cuantos van a trabajar y, en general, se ha de tener una estimación de cuál va a ser la demanda que va a atender el sistema, es decir, los movimientos por origen y destino que hay en toda la red. Por lo tanto, aunque esta linea sea nueva y no esté construida, se quiere ver si sería factible implementarla o no. Con este fin, una de las cuestiones de más interés es saber cuánta gente va a utilizar la nueva línea, substrayéndose del resto de lineas que ya están ejecutando su servicio.

En este caso solo habrían dos maneras de poder averiguarlo: o bien tenemos un método adecuado y bien construido para predecir cuál sería la absorción de pasajeros que tendría esta linea, o bien se construye la línea y se pasa a observar si los resultados son positivos o no. En el segundo caso se estaría probando a ciegas y, si todo sale mal, se acabaría perdiendo tiempo y dinero. Por lo tanto lo óptimo en este caso sería construir un modelo capaz de calcular la tasa de absorción que tendría esta nueva linea.

Esto es precisamente lo que persiguen los modelos denominados de asignación. Cuando se tiene un conjunto de lineas de transporte y se conoce la demanda global del sistema, expresada en función de flujos que van de un lado a otro sin especificar la ruta de transporte, se puede construir un modelo capaz de decir el volumen de la demanda de los pasajeros, por donde van a pasar y las lineas de transporte que van a utilizar.

Este tipo de modelos de optimización suelen ser muy simplificados, por eso la mayoría de las veces les falta poder ser afinados y contrastados con simulaciones creadas para reproducir fenómenos relacionados generalmente con la congestión en las lineas. De esta

manera se extraerían conclusiones mucho más precisas con las que posteriormente poder tomar mejores decisiones.

En la actualidad existen diversos paquetes de software comercial que permiten a los ingenieros de transporte realizar evaluaciones con modelos de diferentes tipos para la asignación de pasajeros a líneas de transporte público. En general, es una práctica extendida entre los profesionales hacer uso de estos modelos en proyectos y evaluaciones de sistemas de redes de transporte público en entornos urbanos y metropolitanos. Sin embargo, hay que tener en cuenta que estos modelos de asignación descansan en hipótesis simplificadoras para poder ser resueltos de manera ágil en el caso de redes urbanas de gran tamaño. Solo en las dos últimas décadas estos modelos han ido evolucionando para poder tener en cuenta los trastornos que la congestión puede introducir en el funcionamiento de las redes de transporte (y aún así con modelos de colas muy simplistas) , mientras que los modelos anteriores no podían ser capaces de tener en cuenta fenómenos de espera bajo congestión.

Para entender mejor todos estos conceptos sobre asignación de pasajeros, se mostrará a continuación un esquema ilustrativo de las interacciones que se producen en cada una de las diferentes unidades de servicio j a lo largo de las paradas i.

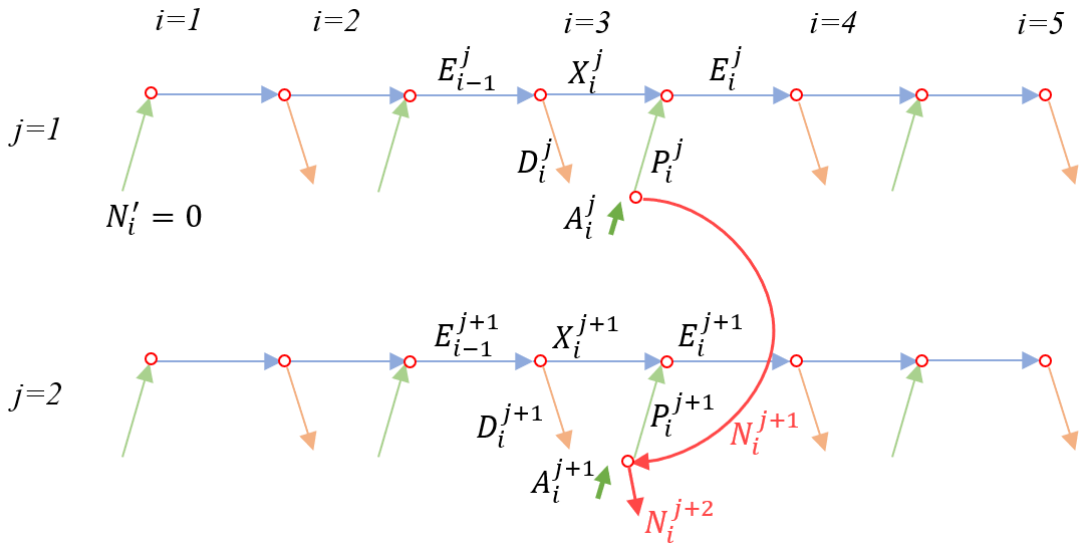

Figura 5.1: Esquema de los sucesos que se producen a lo largo de las paradas

A excepción de la primera y la última parada, se pueden observar distintos tipos de situaciones cuando se estaciona un servicio: bajadas de pasajeros D_i^j, subidas de pasajeros P_i^j, llegadas de pasajeros a la parada A_i^j y pasajeros que permanecen en el servicio antes y después de que este se pare, representados combo X_i^j y E_i^j respectivamente.

Las llegadas de pasajeros a las paradas A_i^j, si se producen de manera masiva, condi-

cionan a que no todos los usuarios que esperan en la cola puedan subir en ese servicio. Es por esto que el algoritmo para las personas que no pueden subir al servicio y han de esperar en la cola a poder subir al siguiente, queda explicado con la siguiente ecuación:

$$N_i^{j+1} = A_i^j - P_i^j + N_i^j$$

Donde el número de personas que quedará esperando poder subir al servicio $j+1$ es igual al número de llegadas que se producen a la parada i más el número de clientes que quedó esperando poder subir al servicio j y restado al número de subidas de pasajeros.

De esta manera queda definida la idea al concepto de asignación de pasajeros a líneas de transporte público, donde estos métodos buscan definir las capacidades de atención de los servicios a lo largo de las paradas.

Capítulo 6

Elementos de simulación

Cuando se realiza una simulación, es muy común utilizar datos que hayan sido generados aleatoriamente en base a una distribución. En este caso, lo que buscamos son métodos que permitan obtener valores de variables aleatorias que sigan determinadas distribuciones de probabilidad. La facilidad para la obtención de estos datos dependerá de la familia de variables aleatorias a la que se aplique.

En este capítulo se verá el proceso de generación de variables aleatorias para distintas distribuciones utilizadas en la construcción del simulador que se presentará más adelante, y algunos procedimientos que se han utilizado para validarlas (Barceló y Montero, 2016).

6.1. Procedimientos de generación de variables aleatorias

A lo largo de toda esta sección se considerará X como una variable aleatoria cuya función de probabilidad se define como $P(X = x_i) = p_i, \qquad i \in I, \sum_{x_i \leq x} p(x_i)$

La generación de cualquier tipo de variable aleatoria que siga determinada distribución de probabilidad, se realiza a partir de la generación previa de una uniforme (0,1). Existen diversos métodos con este propósito y la elección del método adecuado puede basarse en una serie de requerimientos básicos como son:

- **Exactitud**. Se puede argumentar que las distribuciones ajustadas son aproximadas de todos modos, por lo que un método de generación aproximado debería ser suficiente. Pero aun así, se deben preferir los métodos exactos con exactamente la distribución deseada.
- **Eficiencia**. Son métodos destinados a la generación de un gran número de variables aleatorias, por lo tanto se habla de eficiencia en un método cuando este contribuye

un tiempo de ejecución y un tamaño de almacenamiento competentes.

- **Complejidad**. Se debe valorar si la implementación de un algoritmo más complejo proporciona unos resultados más eficientes y si a su vez la implantación de este esfuerzo adicional resulta eficazmente rentable.
- **Robusteza**. Ha de tratarse de un método eficaz para cualquier valor del parámetro.

Los métodos más utilizados para la generación de estas variables, que se explicarán a continuación, se pueden resumir en los siguientes:

- Método de la transformada inversa.
- Método de composición.
- Método de aceptación-rechazo.
- Método de Box-Müller.

En este apartado se tratarán algunos de estos métodos para la generación de variables aleatorias para las siguientes distribuciones:

- Exponencial
- K-Erlang
- Poisson
- Gamma
- Binomial
- Normal

6.1.1. Método de la transformada inversa

Supongamos que la variable aleatoria X que queremos generar es discreta. Esta generación se lleva a cabo a través de la probabilidad acumulada *P(X)* y la generación de números aleatorios $U \sim$ *Uniforme(0,1)*.

$$F(x) = P(X \leq x) = \sum_{x_i \leq x} p(x_i)$$

La función de probabilidad viene definida por $p(x_i) = P(X = x_i)$. Se asume que X solo puede tomar valores $x_1, x_2, ...$ de modo que $x_1 < x_2 < ...$.

El algoritmo para el método sería:

1 Generar un número aleatorio $U \sim$ *Uniforme(0,1)*.

2 Si $F(X_i) \leq U$, hacer i=i+1 y volver a este paso. En caso contrario, ir al siguiente paso.

3 x_i es el valor generado de la variable X

Para generar valores de X a partir de números aleatorios, se divide el intervalo (0,1) en subintervalos de longitudes como valores tome la variable X, de modo que el intervalo i-ésimo tenga probabilidad p_i.

Una vez generadas las variables aleatorias con distribución uniforme, se ha de determinar el número entero I más pequeño tal que $U \leq F(x)$ y devuelva $X = x_I$. De esta manera y según el método de la transformada inversa se tendría:

$$X = \begin{cases} x_0 & \text{si} \quad U < p_0 \\ x_1 & \text{si} \quad p_0 \leq U < p_0 + p_1 \\ ... & \\ x_j & \text{si} \quad p_0 + ... + p_{j-1} \leq U < p_o + ... + p_{j-1} + p_j \end{cases}$$

Por lo tanto X solo tomaría valores si se cumplieran las condiciones. Se ha de tener en cuenta que una vez se ha generado el número aleatorio U, encontrar el valor generado X equivale a encontrar la inversa de la *F(x)*.

Ahora supongamos que la variable aleatoria X que queremos generar es continua. Para eso necesitamos disponer, de forma cerrada, de la función de probabilidad acumulada, la cual siempre tendrá inversa, por ser creciente asimptóticamente hasta 1.

El primer paso consiste en generar $U \sim Uniforme(0, 1)$. A través de la función de densidad de la distribución en cuestión, se obtiene la función de distribución *F(x)* . Según este método si una variable aleatoria X tiene función de distribución y esta admite inversa, la variable transformada U es igual a la función de distribución. Por lo tanto tenemos que *F(x)=U*, de manera que si realizamos la inversa de la función de distribución obtenemos el valor de X.

$$F(x) = U \rightarrow x = F_x^{-1}(U)$$

EXPONENCIAL

A través del método de la transformada inversa es posible generar variables aleatorias continuas para la distribución Exponencial. La función de densidad para la exponencial viene definida de la siguiente manera:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

Y la función de distribución es:

$$F(x) = \int_0^x f(x)dx = \int_0^x \lambda e^{-\lambda x}dx = -e^{-\lambda x}|_0^x = 1 - e^{-\lambda x}$$

Si resolvemos la ecuación *F(x)=U* substituyendo la primera parte por el resultado obtenido arriba y aplicamos la inversa se obtiene el valor de x:

$$U = 1 - e^{-\lambda x} \rightarrow \quad x = -\frac{1}{\lambda} ln(1-U) \quad o \quad x = -\frac{1}{\lambda} ln(U)$$

POISSON

También se pueden generar variables aleatorias discretas para la distribución Poisson a través de la técnica de la transformada inversa. La distribución de Poisson es una distribución de probabilidad discreta que se aplica a los eventos de algún suceso durante un intervalo de tiempo determinado t.

La distribución de probabilidad en este caso para X es:

$$f(x) = \frac{\lambda t^x e^{-\lambda t}}{x!} \quad , x = 0, 1, 2, ...$$

Los tiempos de llegada entre eventos de un proceso Poisson de parámetro λ son variables aleatorias exponenciales de media $\frac{1}{\lambda}$ y, por lo tanto, el número de eventos en un cierto periodo de tiempo que llamaremos t, es una variable aleatoria Poisson de media λt.

Como es una distribución de probabilidad discreta, se ha de evaluar *f(x)* para todos los valores de x y, de esta manera, evaluar su función de distribución acumulada *F(x)*. Por lo tanto, para generar una variable aleatoria x, se ha de crear una sentencia donde se vayan generando variables aleatorias exponenciales y_i dentro del intervalo (0,1):

- **Inicialización**

  $Y = 0; \;\; i = 1;$

- Mientras $Y < 1$:
  - $y_i = -\frac{1}{\lambda} ln(U_i)$
  - $Y = y_i + Y$
- Fin Mientras

$x = i - 1$

BINOMIAL

Sea $X \sim B(n, p)$. La función de probabilidad puntual de X es:

$$P(X = x) = \binom{n}{r} p^x (1-p)^{n-x}$$

Sabemos que una variable *Binomial(n,p)* es la repetición de n experimentos de *Bernoulli* donde p representa la probabilidad de éxito y (1-p) la probabilidad de fracaso.

El algoritmo para generar valores de esta distribución es el siguiente:

Paso 1 Se crea x=0, i=1.

Paso 2 Se genera un número aleatorio u_i. Si $u_i \leq p$, hacer $x = x + 1$.

Paso 3 Incrementamos el valor de i: $i = i + 1$.

Paso 4 Si $i \leq n$ volver al paso 3. En caso contrario x será el valor generado de la variable X, y esta contendrá una variable *Binomial(n,p)*

6.1.2. Método de la composición

Este método se usa si la variable aleatoria X puede ser expresada como la suma de n variables aleatorias $X = x_1 + x_2 + x_3 + ... + x_n$, de manera que x_i sigue una distribución y X sigue otra.

K-ERLANG

Un ejemplo para este método es la distribución *K-Erlang*. La variable aleatoria X sigue una $K-Erlang(\lambda, k)$, y se puede expresar como la suma de k variables $x_1+x_2+x_3+...+x_n$ cada una con valor esperado $\frac{1}{\lambda}$, donde x_i sigue una distribución exponencial.

La función de densidad de la distribución *K-Erlang* viene definida de la siguiente manera:

$$f(x) = \lambda e^{-\lambda x} \frac{(\lambda k)^{k-1}}{(k-1)!}, \quad x, \lambda \geq 0$$

Cuando el parámetro k=1 , la distribución de la variable aleatoria X es la exponencial puesto que la fracción de la función de densidad de la *K-Erlang* es equivalente a 1.

Por lo tanto si se quiere obtener el valor de X, es necesario sumar los valores simulados de k variables aleatorias exponenciales con media $\frac{1}{\lambda}$, de esta manera se ha de aplicar la siguiente ecuación:

$$x = \sum_{i=1}^{k} x_i = -\frac{1}{\lambda} ln(\Pi_{i=1}^{k} U_i)$$

Donde todas las x_i han sido generadas por el método de la transformada inversa y siguen una distribución exponencial.

6.1.3. Método de aceptación-rechazo

Este método no es tan directo como los métodos anteriores y a la vez es algo más complejo. Supongamos *f(x)* como la función de densidad de la distribución deseada, de esta manera se ha de encontrar una función *e(x)* exponencial con la misma media. El método dice que se ha de calcular una constante que llamaremos a. Con este objetivo se debe de encontrar el máximo de $\frac{f(x)}{e(x)}$ derivando esta división e igualándola a cero. De esta manera se encuentra el punto donde la constante a es máxima.

Una vez obtenido el valor de esta constante, se aplica la fórmula $\frac{f(x)}{ag(x)}$ necesaria para realizar el algoritmo de creación de variables aleatorias para la distribución en cuestión. Los pasos del algoritmo son los siguientes:

1 Generar un número aleatorio $U_1 \sim$ *Uniforme(0,1)*.

2 Con esta U_1 generar una variable aleatoria exponencial: $Y = -\frac{1}{\lambda} ln(U_1)$

3 Generar un número aleatorio $U_2 \sim$ *Uniforme(0,1)*.

4 Si $U_2 \leq \frac{f(Y)}{ag(Y)}, entonces X = Y$. En otro caso, se vuelve al paso 1.

GAMMA

Es posible generar variables aleatorias para la distribución $Gamma(\alpha, \beta)$ través de este método. Su función de densidad es:

$$f(x) = \frac{1}{\Gamma(\alpha)}\beta^{-\alpha}x^{\alpha-1}e^{-\frac{x}{\beta}}$$

En el caso estándar se tiene que $\beta = 1$, por lo tanto:

$$f(x) = \frac{1}{\Gamma(\alpha)}\beta^{-\alpha}x^{\alpha-1}e^{-\frac{x}{\beta}} = \frac{1}{\Gamma(\alpha)}x^{\alpha-1}e^{-}x$$

La media de la función *Gamma* es $\Gamma(\alpha, \beta)$, que es igual a:

$$\Gamma(\alpha, \beta) = \alpha\beta = \alpha$$

La media de la exponencial es $\frac{1}{\lambda}$. Con este resultado se genera una función exponencial con la misma media:

$$g(x) = \lambda e^{-\lambda x} = \frac{1}{\alpha}e^{-\frac{x}{\alpha}}, x > 0$$

De esta manera se divide *f(x)* entre *g(x)*:

$$\frac{f(x)}{g(x)} = \frac{\frac{1}{\Gamma(\alpha)}x^{\alpha-1}e^{-}x}{\frac{1}{\alpha}e^{-\frac{x}{\alpha}}} = \frac{\alpha x^{\alpha-1}e^{-}x}{\Gamma(\alpha)e^{-\frac{x}{\alpha}}} = \frac{\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-x+\frac{x}{\alpha}}$$

Para calcular la constante a, se debe de encontrar el valor del punto donde se encuentra el máximo de la división anterior:

$$\begin{aligned}\frac{d}{dx}[\frac{f(x)}{g(x)}] \rightarrow \frac{\alpha}{\Gamma(\alpha)}[(\alpha-1)x^{\alpha-2}e^{-x+\frac{x}{\alpha}} + (-1+\frac{1}{\alpha})x^{\alpha-1}e^{-x+\frac{x}{\alpha}}] = 0 \rightarrow \\ \rightarrow (\alpha-1)x^{\alpha-2} + x^{\alpha-1}(-1+\frac{1}{2}) = 0 \rightarrow \\ \rightarrow x^{\alpha-2}(-\frac{1}{2}x + (\alpha-1)) = 0 \\ \rightarrow \overline{x} = 2\alpha - 2; \overline{x} = 0\end{aligned}$$

Descartamos $x = 0$ puesto que nos saldría una indeterminación y obtenemos que $x = 2\alpha - 2$. Una vez obtenido el valor $\overline{x}$ que maximiza $\frac{f(x)}{g(x)}$, se substituye en la división de tal manera que se obtiene el valor de la constante a:

$$a = max[\frac{f(x)}{g(x)}] = \frac{f(\overline{x})}{g(\overline{x})} = \frac{\alpha}{\Gamma(\alpha)}(2\alpha-2)^{\alpha-1}e^{-(2\alpha-2)+\frac{(2\alpha-2)}{\alpha}}$$

Con el valor obtenido de la constante a se aplica la siguiente fórmula:

$$\begin{aligned}\frac{f(x)}{ag(x)} &= \frac{\frac{1}{\Gamma(\alpha)}x^{\alpha-1}e^{-x}}{\frac{\alpha}{\Gamma(\alpha)}(2\alpha-2)^{\alpha-1}e^{(-2\alpha-2)+\frac{(2\alpha-2)}{\alpha}}\frac{1}{\alpha}e^{-\frac{x}{\alpha}}} = \\ &= \frac{x^{\alpha-1}e^{-}x}{(2\alpha-2)^{\alpha-1}e^{-(2\alpha-2)+\frac{(2\alpha-2)}{\alpha}}e^{-\frac{x}{\alpha}}} = \\ &= \frac{x^{\alpha-1}}{(2\alpha-2)^{\alpha-1}e^{-2\alpha^2-2-x+xa}}\end{aligned}$$

Por último se aplica el algoritmo para generar las variables aleatorias con la gamma:

1 Se genera un valor aleatorio uniforme $U_1 \sim$ *Uniforme(0,1)*. Con este valor generamos una variable aleatoria exponencial con: $Y = -\frac{1}{\lambda}ln(U_1)$

2 Se genera otro valor aleatorio uniforme $U_2 \sim$ *Uniforme(0,1)*.

3 Si $U_2 \leq \frac{x^{\alpha-1}}{(2\alpha-2)^{\alpha-1}e^{-2\alpha^2-2-x+xa}}$, *entonces* $X = Y$. En otro caso, se vuelve al paso 1.

6.1.4. Método de Box-Müller

Es un método de generación de pares de números aleatorios independientes con distribución normal estándar (esperanza cero y varianza uno), a partir de números aleatorios distribuidos uniformemente.

La idea es generar dos muestras con distribución *Uniforme* $\sim (0,1)$ y transformarlas en dos muestras con distribución normal. El método de la transformada inversa también genera números aleatorios distribuidos normalmente pero a diferencia de Box-Müller su coste computacional es superior y, por lo tanto, menos eficiente.

NORMAL

Imaginemos un sistema cartesiano bidimensional donde las coordenadas de X e Y son dadas por variables aleatorias independientes y distribuidas normalmente.

Donde los valores de R y θ corresponden a :

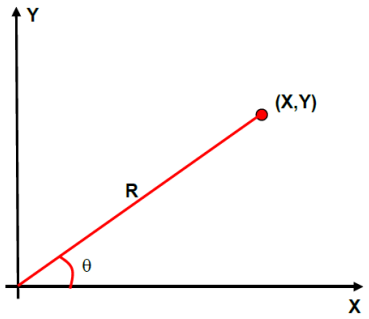

Figura 6.1

$$R^2 = X^2 + Y^2 \quad (X = Rcos\theta)$$
$$\theta = \arctan\frac{Y}{X} \quad (Y = Rsen\theta)$$

Si la función de densidad de una distribución normal para una variable x es:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$$

Dado que las variables X e Y son independientes, su función de probabilidad conjunta será el producto de las funciones de probabilidad individuales:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} = \frac{1}{2\pi} e^{\frac{-(x^2+y^2)}{2}}$$

Para determinar la función de probabilidad conjunta de R^2 y θ, se realiza el siguiente cambio de variable:

$$d = x^2 + y^2 \quad \theta = \tan^{-1}(\frac{y}{x}) \rightarrow \quad f(d,\theta) = |J|^{-1} f(x,y)$$

Donde J es el Jacobiano de la transformación.

$$J = \frac{2x^2}{x^2+y^2} + \frac{2y^2}{x^2+y^2} = 2$$

$$f(d,\theta) = \frac{1}{2}\frac{1}{2\pi} e^{-\frac{d}{2}}, \;\; 0 < dz < \text{ínf}; \;\; 0 < \theta < 2\pi$$

Esto es igual al producto de una uniforme aleatoria en $(0, 2\pi)$ y una exponencial de media 2, $\left(\frac{1}{2}e^{-\frac{d}{2}}\right)$, lo que implica que R^2 y θ son independientes con R^2 exponencial de media 2 y θ distribuida uniformemente en $(0, 2\pi)$.

Se pueden generar X, Y variables aleatorias independientes estándar normales generando R^2 y θ, coordenadas polares del punto (X, Y) y transformándolas en coordenadas cartesianas.

El algoritmo resumido para la generación de variables aleatorias de Box-Müller se presenta a continuación:

Paso 1 Generar números aleatorios uniformes independientes U_1 y U_2.

Paso 2 Generar R^2 exponencial con media 2: $R^2 = -2lnU_1$, y generar θ uniforme en $(0, 2\pi)$: $\theta = 2\pi U_2$

Paso 3 Se transforma a coordenadas cartesianas.

$$X = Rcos\theta = \sqrt{-2lnU_1}cos(2\pi U_2)$$
$$Y = Rsen\theta = \sqrt{-2lnU_1}sin(2\pi U_2)$$

6.2. Tests

6.2.1. Test Chi-cuadrada

Después de generar variables aleatorias es importante asegurarse de que los valores generados realmente siguen la distribución teórica que deberían. Para esto, existen diferentes pruebas para comprobar si una muestra de valores generados aleatóriamente bajo el supuesto de que provienen de una cierta distribución, siguen realmente la distribución para la cual están siendo evaluados.

Con este mismo propósito se ha realizado un test de chi-cuadrado para las muestras generadas de diferentes distribuciones. Este test de bondad de ajuste de la χ^2, también conocido como un test de frecuencias, es una prueba de hipótesis que compara la distribución observada de los datos con su distribución esperada. De esta manera se puede comprobar si una muestra proviene de una función de probabilidad específica.

La prueba parte con la idea de comprobar la bondad del ajuste de una muestra de n valores $t_1, t_2, ..., t_n$ con una distribución determinada. Se quiere comprobar si la distribución de esta muestra es igual a la distribución contra la que se quiere hacer el test.

En primer lugar se realiza un histograma de los datos y se calculan estadísticos básicos de la muestra como la media o la desviación estándar, de esta manera se puede ver si a *priori* se puede determinar que los valores generados de la muestra y sus respectivos

parámetros provienen de la distribución teórica. Esto se puede generalizar teniendo en cuenta varios momentos de la distribución y los que se obtienen mediante la muestra.

Después se ha de calcular el test χ^2. Como hipótesis para esta prueba se tiene que:

H_0 : *La muestra proviene de la distribución que se plantea.*

H_1 : *La muestra no proviene de la distribución que se plantea.*

Si el test no lo contradice, y además de manera contundente, aceptamos la hipótesis nula de que los valores generados provienen de la distribución teórica deseada.

Para la creación de este test se han de fijar un conjunto N de subintervalos $[x_i, x_{i+1}]$, como por ejemplo $N = 10$ que cubren todo el intervalo de valores posibles para la variable aleatoria t de manera que $P(x_i \leq t \leq x_{i+1})$. De esta manera tendríamos el intervalo de valores de la variable x_1 y x_0 correspondientes a cada decil.

El número esperado de elementos de la muestra que tendrían que estar comprendidos en cada subintervalo tendría que ser:

$$n_e = n/N$$

Si las dos distribuciones, la empírica y la teórica, no discrepan demasiado la igualdad anterior se cumpliría.

Por otro lado, en cada uno de los intervalos $[x_i, x_{i+1}]$ se ha de contabilizar el numero de elementos n_i de la muestra n que caen dentro de cada subintervalo i.

Una vez se tienen contabilizados los valores esperados n_e y observados n_i se calcula una medida global de discrepancia entre estos valores:

$$\chi^2 = \sum_{i=1}^{N} (\frac{(n_i - n_e)^2}{n_e})$$

Este estadístico de la χ^2 se distribuye aproximadamente según una ley χ^2_{N-m-1}, siendo m el número de parámetros de la distribución. Una vez se conoce el valor de χ^2 y la ley con la que se distribuye se pueden fijar los intervalos de aceptación o de rechazo. Si χ^2 cae dentro de estos intervalos de aceptación se afirma que la prueba ha pasado el test, en caso contrario no se podría asegurar que la muestra provenga de la distribución teórica.

Para comprobar si los anteriores procedimientos de generaciones de variables eran o no adecuados, se ha realizado el test de bondad del ajuste a la distribución Uniforme, Exponencial, Poisson, K-Erlang y Binomial.

Tests de bondad ajuste. Caso Variable Uniforme

El histograma para la muestra de valores generados de la uniforme se muestra en la figura 6.2.

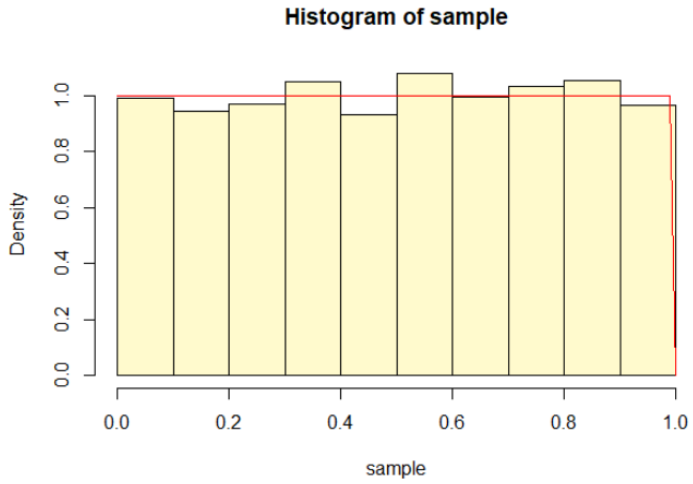

Figura 6.2: Histograma para la muestra de valores generados con la Uniforme

Se observa como para los diferentes valores de la muestra, esta se distribuye homogéneamente. Parece que la muestra de valores se ajusta a una distribución teórica uniforme como la proporcionada por la recta roja.

Antes de realizar el test, se ha fijado el número de subintervalos ($N = 25$) para los valores de la muestra. Estos vienen definidos por la siguiente secuencia de valores:

0,00 0,04 0,08 0,12 0,16 0,20 0,24 0,28 0,32 0,36 0,40 0,44 0,48
0,52 0,56 0,60 0,64 0,68 0,72 0,76 0,80 0,84 0,88 0,92 0,96 1,00

Para una muestra de 5000 valores, se espera que cada subintervalo contenga 200 observaciones. Por otro lado, el número de observaciones reales que hay en cada intervalo es:

198 205 190 179 194 206 191 195 203 214 186 196 178
215 229 204 198 188 221 200 215 200 208 193 194

Con estos datos, y teniendo en cuenta los grados de libertad $(25 - 1 = 24)$, se realiza el test de la chi-cuadrada y se obtiene un valor de $\chi^2 = 18{,}47$. Se rechazará la distribución propuesta si $P(x \geq \chi^2 = p - valor < \alpha)$.

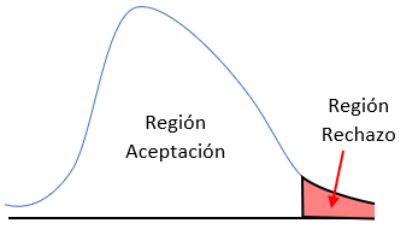

Figura 6.3: Contraste Chi-cuadrada

Por lo tanto, la región de aceptación es $1 - \alpha$, mientras que la de rechazo es el nivel de significación α. Con un nivel de confianza del 95 % ($\alpha = 0{,}05$), se obtiene un p-valor de 0,7796. Como el p-valor entra dentro de la zona de aceptación, se acepta la H_0 y decimos que la muestra de valores generada proviene realmente de la uniforme.

El mismo procedimiento se ha realizado para el resto de distribuciones, teniendo en cuenta las características particulares de cada una.

Tests de bondad de ajuste. Caso Variable Exponencial

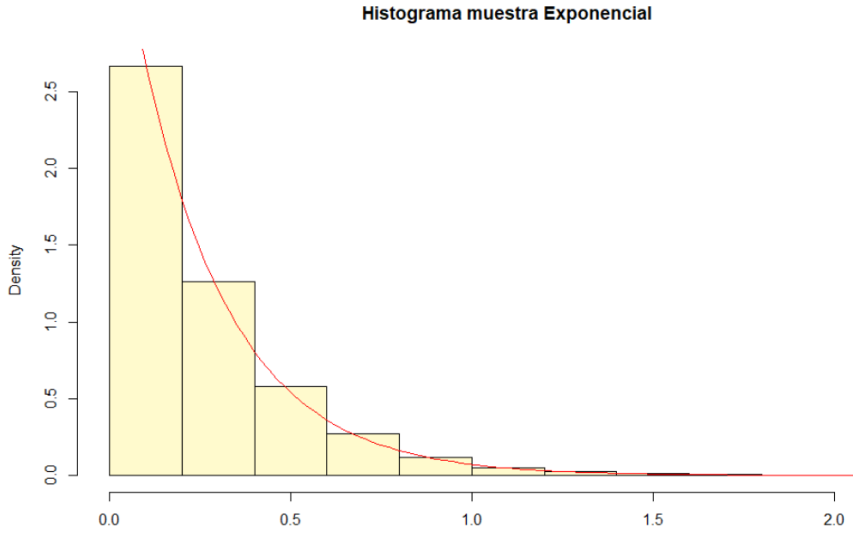

Figura 6.4: Histograma para la muestra de valores generados con la Exponencial

En la figura 6.4 es posible ver como los datos para la Exponencial se ajustan a su distribución teórica. Para los intervalos, primero se ha construido un vector de probabilidades entre 0 y 1 y se ha adoptado un valor para el parámetro de $\lambda = 4$. Con la función *qexp* de

R se han devuelto los valores correspondientes de la función cuantil de probabilidad, es decir, la inversa de la función de distribución acumulada de la distribución exponencial. Los valores que definen los subintervalos son:

0,000 0,010 0,021 0,032 0,044 0,056 0,069 0,082 0,096 0,112 0,128 0,145 0,163
0,183 0,205 0,229 0,255 0,285 0,318 0,357 0,402 0,458 0,530 0,631 0,805 2,182

Igual que en el caso para la distribución anterior, el número de observaciones que se espera que entren dentro de cada subintervalo es de 200. De la misma manera, se muestran los valores que realmente se observan dentro de esos intervalos:

201 194 193 191 180 192 201 194 213 182 192 181 211
199 194 189 195 198 219 214 216 222 193 218 218

En este caso, los grados vienen definidos por una χ^2_{25-1-1} con 23 grados de libertad. Con un nivel de significación de $\alpha = 0{,}05$, se obtiene un $\chi^2 = 19{,}44$ y un p-valor de 0.675. Puesto que este valor es superior al 0.05, se acepta la H0 y se afirma que la muestra de valores para la que se ha hecho el test proviene realmente de la distribución Exponencial.

Tests de bondad de ajuste. Caso Variable K-Erlang

El siguiente gráfico representa la distribución empírica y teórica para la muestra de datos de la K-Erlang. Se ha generado una muestra de 5000 valores, con $\alpha = 0{,}5$ y número de servidores $k = 3$.

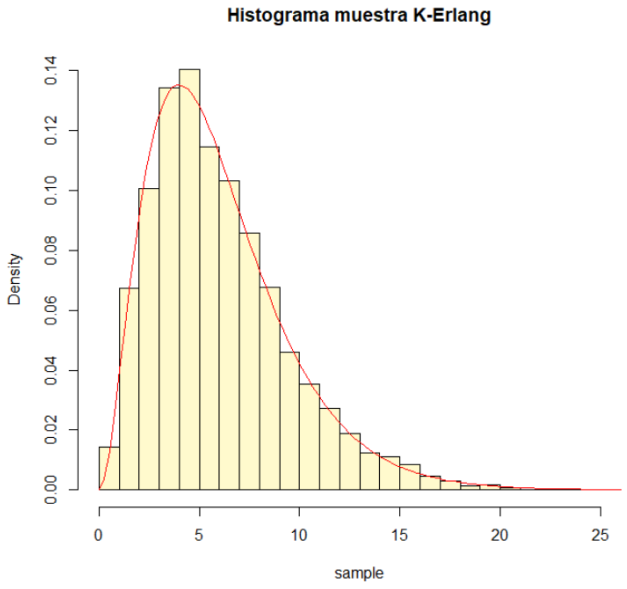

Figura 6.5: Histograma para la muestra de valores generados con la K-Erlang

Los datos generados parece que se ajustan a la curva de la distribución teórica. Con el vector de probabilidades entre 0 y 1, y la función *qgamma* de R, se ha generado el vector de cuantiles para crear los intervalos:

0,000 1,492 1,997 2,395 2,746 3,070 3,379 3,679 3,975 4,271 4,570 4,875 5,187 5,512
5,852 6,211 6,594 7,009 7,465 7,974 8,558 9,250 10,112 11,283 13,198 28,462

Como en las distribuciones anteriores, y siendo $N = 25$, se espera que hayan 200 observaciones por intervalo, pero los valores que realmente caen dentro de cada uno son:

194 206 186 207 210 202 193 188 187 186 178 217 191
214 212 203 199 214 225 179 188 220 188 200 213

Como la distribución k-Erlang cuenta con dos parámetros, los grados de libertad son $25-2-1 = 22$. Con un nivel de confianza del 95 %, se obtiene un $\chi^2 = 21{,}51$ y un p-valor de 0,4894 superior al nivel de significación, por lo que se acepta la H0 y se afirma que la muestra de valores para la que se ha hecho el test proviene realmente de la distribución k-Erlang.

Tests de bondad de ajuste. Caso Variable Poisson

Para la distribucion Poisson se han generado 5000 valores con parámetros $\lambda = 4$ y $t = 400$, donde $E[x] = \lambda t = 1600$.

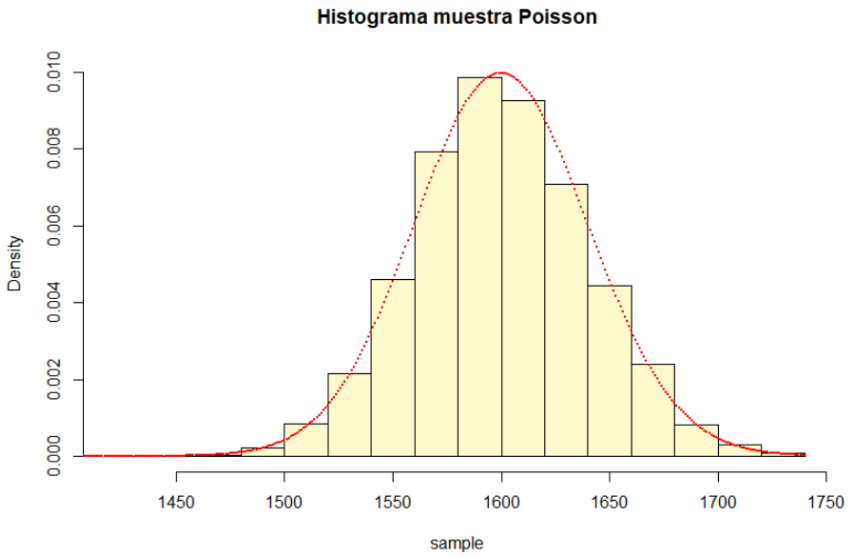

Figura 6.6: Histograma para la muestra de valores generados con la Poisson

El gráfico de la figura 6.6 presenta los resultados de comparar la muestra generado con los valores teóricos para esta distribución. Se puede apreciar como la muestra empírica se ajusta correctamente a la distribución Poisson.

Con el vector de probabilidades entre 0 y 1, y la función *qpois* de R, se ha generado el siguiente vector de cuantiles para crear los intervalos:

0 1531 1544 1553 1561 1567 1572 1577 1581 1586 1590 1594 1598 1602
1606 1610 1614 1619 1624 1628 1634 1640 1647 1657 1671 1739

Siendo $N = 25$, se espera que hayan 200 observaciones dentro de cada uno de esos intervalos. A continuación se presenta la tabla de los valores realmente observados dentro de cada intervalo:

201 194 199 223 206 214 207 170 215 217 210 201 210
199 196 173 213 200 150 203 197 193 205 190 214

Teniendo en cuenta los grados de libertad que rigen esta distribución (25-1-1=23) y un nivel de confianza del 95 %, se obtiene un valor para el estadístico de $\chi^2 = 31{,}33$ y, un p-valor superior al nivel de significación de 0,1148. De esta modo, se acepta la H_0 y se afirma que la muestra de valores contra la que se ha hecho el test proviene realmente de la distribución Poisson.

Tests de bondad de ajuste. Caso Variable Binomial

Para demostrar la procedencia de los valores generados mediante la distribución Binomial, se ha realizado una muestra de 5000 valores con $n = 5$ y probabilidad de $p = 0{,}5$. El gráfico de los valores observados se representa a continuación:

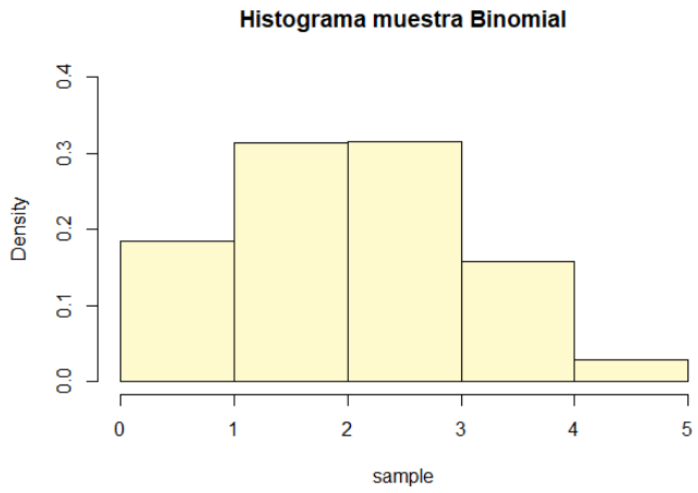

Figura 6.7: Histograma para la muestra de valores generados con la Binomial

Dado que la probabilidad con la que se ha generado la muestra es de 0.5, las frecuencias altas de los datos se acumulan en los valores de la parte central.

A continuación se verá la tabla de valores y proporciones para los datos observados y esperados de la muestra generada:

n	*V. observados*	*P. observados*	*P. esperados*	*V. esperados*
0	159	0.0318	0.03111274	155.5637
1	762	0.1524	0.15583774	779.1887
2	1567	0.3134	0.31222452	1561.1226
3	1577	0.3154	0.31277452	1563.8726
4	791	0.1582	0.15666274	783.3137
5	144	0.0288	0.03138774	156.9387

$$\chi^2 = \sum \frac{(Valores\ observados - Valores\ esperados)^2}{Valores\ esperados} = 1{,}73$$

El valor del estadístico obtenido es de 1.73. Con un nivel de confianza del 95 % y $6-1-1=4$ grados de libertad, el p-valor que se obtiene es de 0.7853, de tal manera que se acepta la H_0 y se afirma que la muestra de valores contra la que se ha hecho el test proviene realmente de la distribución Binomial.

6.2.2. Runs Test

Se ha podido ver que la generación de variables aleatorias de la gran mayoría de las distribuciones parte con la generación previa de variables aleatorias uniformes. En la sección anterior se ha utilizado un test para comprobar que los métodos utilizados realmente generaban la distribución deseada, pero todavía no se ha verificado si las variables que están siendo generadas son realmente aleatorias o no e independientes entre sí, dentro de la secuencia generada. Para ello se ha construido un programa en R que realiza el *Runs Test*de independencias sobre una muestra de variables uniformes generadas. La hipótesis que se contrasta es la siguiente:

H_0 : *Cada resultado de la muestra n es igual de probable que se convierta en el resultado observado. En este caso se concluye que la muestra es suficientemente aleatoria*

H_1: *Hay muy pocas o demasiadas fluctuaciones en la ejecución como para que se pueda considerar que provengan de una fuente totalmente aleatoria.*

Para entender mejor el uso de este test, se explicará con un ejemplo.

Póngase el supuesto que existe un número N de variables uniformes generadas. La idea es saber el número de crecimientos y decrecimientos que se producen en la secuencia. Imaginar que se dispone del siguiente vector de variables consecutivas:

0.41	0.68	0.89	0.84	0.74	0.91
+	+	+	-	-	+

Se puede ver que se han producido cambios en el comportamiento, en base a subidas y bajadas, de los valores a lo largo de toda la secuencia: los tres primeros valores crecen progresivamente, los dos siguientes decrecen, y el último vuelve a tener un comportamiento creciente. Estos cambios se denominan *runs* y en este ejemplo se diría que hay tres. Por lo tanto, si N es la longitud de la secuencia, el máximo número de runs (total, entre subidas y bajadas) es $N-1$, y el mínimo es 1.

Si se denomina a como el número total de runs en una secuencia, la media y la varianza (Meng, 2002) de a están proporcionadas por:

$$\mu_a = \frac{(2N-1)}{3}$$
$$\sigma_a^2 = \frac{16N-29}{90}$$

Para $N > 20$, la distribución de a es aproximada por una distribución Normal $N(\mu_a, \sigma_a^2)$. Esta aproximación se puede usar para probar la independencia de los números de un generador, de manera que:

$$Z_0 \sim N(0,1) = \frac{(a - \mu_a)}{\sigma_a}$$

.

Si al calcular el p-valor del test, $P(|Z| > Z_0)$, se obtiene un valor superior al nivel de significación asignado, no habría motivos para rechazar la hipótesis nula de que los valores generados son realmente aleatorios.

Con motivo de la realización de la parte práctica de este trabajo, se han generado 5000 valores aleatorios de una uniforme y se les ha pasado este test de independencia para comprobar su aleatoriedad.

La función ha calculado un total de 3324 runs a lo largo de la secuencia y, con un nivel de confianza del 95 %, se ha obtenido un valor para el estadístico $Z = -0{,}3019$.

Para seguir las reglas de decisión, se ha de tener en cuenta que se trata de un contraste bilateral (figura 6.8), es decir, la hipótesis alternativa da lugar a una región crítica a ambos lados del valor del parámetro.

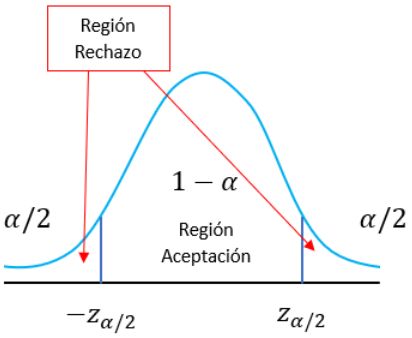

Figura 6.8: Contraste bilateral

En este caso se rechaza H_0 si el estadístico de contraste cae en la zona crítica, es decir, si el estadístico de contraste toma un valor tan grande o tan pequeño que la probabilidad de obtener un valor tan extremo o más que el encontrado es mejor que $\alpha/2$.

Si $-Z_{\alpha/2} < Z < Z_{\alpha/2}$ entonces aceptamos H0. En este caso $-1{,}95 < -0{,}3019 < 1{,}95$, por lo tanto aceptamos que la muestra de valores generadas para la distribución uniforme es realmente aleatoria.

6.3. Procedimiento de simulación de event-scheduling

La gran mayoría de los programas de simulación se basan en un enfoque de programación de eventos. El procedimiento de simulación de Event-Scheduling, traducido al castellano como eventos de organización o programación de eventos, es una metodología de simulación por eventos muy potente y computacionalmente hablando muy eficaz.

En esta metodología de simulación es importante tener en cuenta qué sucesos provocan cambios en qué variables de estado (Tyszer, 1999). En términos de simulación, un evento representa todo cambio en el valor de las variables de estado. En la simulación orientada a eventos se gestiona la variable *tiempo* y se va ajustando hasta la ocurrencia del siguiente suceso (o evento). Por ese motivo es muy importante controlar los acontecimientos del próximo hecho: es necesario saber cuáles son los posibles eventos en un futuro cercano y, de todos ellos, cuál es el más inmediato.

Para ver con más claridad su funcionamiento, se va a explicar con un ejemplo la simulación de un sistema de espera X/Y/s (por ejemplo,un M/M/1) con este método. Un sistema de espera $X/Y/s$ (Codina y Montero, 2020) supone:

- Una población finita.
- Un espacio de espera en la cola único e ilimitado.
- Una disciplina de servicio FIFO (recordar que era el primero en llegar el primero en salir).
- Tiempos de llegadas de los clientes y tiempos de servicio distribuidos respectivamente por $F_X(x)$ y $F_Y(y)$.
- Un número establecido s de servidores.

Por otro lado, las variables de estado que definen el estado de servicio serian:

- El número de clientes en el sistema de espera.
- El estado de cada servidor.
- El instante de la siguiente llegada al sistema.
- El instante de la siguiente salida del sistema.

Las llegadas de nuevos clientes al sistema y las salidas de los que ya han sido servidos, son sucesos que provocan cambios en las variables de estado, y reciben el nombre de *eventos*.

La simulación de Event-Scheduling consiste en planificar los diferentes tipos de sucesos o eventos controlados por el instante de simulación o instante de reloj (clock time). Según este tiempo de simulación, cada tipo de suceso está ordenado en una lista de sucesos, que toma el nombre de *event list*.

Inicialmente en la simulación se considera un sistema nulo donde todos los servidores se encuentran libres, no hay colas de clientes, y el tiempo de simulación parte de cero, y este puede estar determinado por la llegada del primer cliente.

El esquema de un evento de llegada se representa de la siguiente manera:

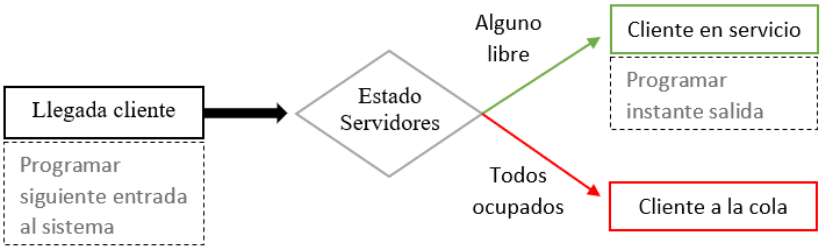

Figura 6.9: Esquema del evento llegadas

El instante de salida de un cliente que sale de la cola de espera y entra en el sistema de servicio es el tiempo actual de la simulación incrementado al tiempo de durada del servicio generado aleatoriamente según la distribución $F_Y(y)$. El instante de la llegada del siguiente cliente al sistema es el instante actual incrementado al tiempo entre llegadas generadas aleatoriamente según la distribución $F_X(x)$.

Del mismo modo, las salidas de los clientes del sistema también son considerados como eventos y se pueden representar esquemáticamente como en la figura 6.10.

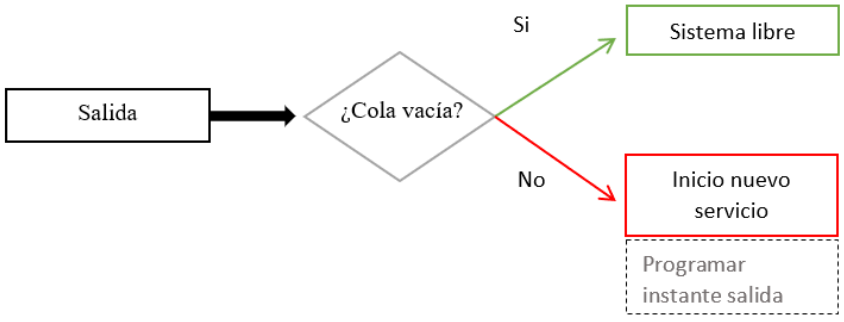

Figura 6.10: Esquema del evento salidas

Una vez se tienen en conocimientos los sucesos que se producen en el sistema, se ha de decretar el mecanismo de actualización en el tiempo de la simulación, es decir, el *clock time*. Las formas más comunes de renovación se basan en:

- Incrementar el tiempo hasta el próximo suceso. El *clock time* avanza hasta el instante del evento más cercano programado hasta el momento. Esto exige que todos los procesos que estén vinculados a cada evento provoquen a su vez la planificación de nuevos eventos.

- Incremento fijo del tiempo de simulación: $\triangle t$. El tiempo de la simulación avanza de $\triangle t$ en $\triangle t$ y si existe algún evento a ser tratado entonces procede a activar el procedimiento vinculado al evento.

- Incremento fijo del tiempo de simulación: $\triangle t$. El tiempo de la simulación avanza de manera fija de $\triangle t$ en $\triangle t$. Si en ese tiempo se producen eventos, se ejecutará el procedimiento adecuado de estos.

A pesar de la eficiencia de esta metodología, la simulación por Event-Scheduling técnicamente es muy compleja; a menudo el proceso es largo y complicado para desarrollar un modelo. Requiere de estructuras de datos muy sofisticadas y una depuración rigorosa del código para, posteriormente, implementar la agenda de simulación.

6.4. Simulación para evaluar colas de un único servidor

El simulador que se realiza en esta práctica es una extensión de este tipo de modelo. En esta simulación, donde se evalúan colas de un único servidor, se crean unos bucles de sentencias que, con pocas instrucciones (como las que se comentarán a continuación) se puede construir el simulador. En esta variante no existe la simulación por eventos como en el procedimiento anterior si no que se recurren las sentencias a la vez dentro del mismo bucle.

A continuación se presentará una relación de recurrencia para evaluar colas $G/G/1$ de un único servidor (Codina y Montero, 2020).

- x_i = tiempo de servicio del cliente i.

- θ_i = instante de salida del sistema de espera del cliente i.

- t_i^S = instante de entrada al sistema de servicio del cliente i.

- t_i = instante de entrada al sistema de espera; se obtiene a partir de la secuencia de tiempo entre llegadas.

$$\tau_i = t_{i+1} - t_i; \quad t_i = \sum_{l=1}^{i-1} \tau_l$$

- $w_i = \theta_i - t_i$ = tiempo de permanencia del cliente i en el sistema de espera.
- $w_{q,i} = t_i^S - t_i$ = tiempo de permanencia del cliente i en la cola de espera.
- $L_i = w_i$=contribución del cliente i a la ocupación.
- $t_0 = -\infty$

El siguiente algoritmo presenta un seguido de ecuaciones de recurrencia que permiten ir generando los valores de los instantes de entrada en el sistema de servicio t_i^S y de salida del sistema de espera θ_i, para $i = 1, 2, 3, 4, n$ clientes:

- **Inicialización:**

  $L = 0; \;\; W = 0; \;\; L_q = 0; \;\; W_q = 0; \;\; \theta_q = 0; \;\; t_1 = 0$

- Para $i = 1, 2, 3, ..., n$

  1. $t_i^S = max(\theta_{i-1}, t_i)$
  2. $\theta_i = t_i^S + x_i$
  3. Si $i < n$ entonces: Generar $\tau_i; t_{i+1} = t_i + \tau_i$.
  4. Recogida estadísticos:

     a) $L_i = w_i = \theta_i - t_i; L = L + L_i; L_{T_i} = \frac{L}{t_i - t_1}; W = W + w_i$

     b) $L_{q,i} = w_{q,i} = t_i^S - t_i; L_q = L_q + L_{q,i}; W_q = W_q + w_{q,i}$

     Después del cliente n:

$$W = \frac{W}{n}; W_q = \frac{w_q}{n}; L = \frac{L}{t_n - t_1}; L_q = \frac{L_q}{t_n - t_1}$$

Este tipo de simulador es uno de los más simples que se puedan pensar en escribir, y el que se describe en el proyecto está basado en una ampliación mucho más compleja de esta idea. En realidad, los procesos de llegadas de los autobuses a la parada siguen este tipo de procedimiento.

Capítulo 7

Implementación de un simulador

Antes de comenzar a explicar el simulador y su procedimiento, se hará un resumen detallado de las variables y conceptos que se han utilizado para llevar a cabo esta práctica así como la notación que se ha utilizado para construir el código.

El objetivo principal del trabajo es simular una red de transporte público urbano, es por esto que las variables definidas que se han utilizado en la simulación han sido diseñadas con la finalidad de que el problema que se resuelve se aproxime lo máximo posible a un caso real. Para ello, se han tenido en cuenta todos los fenómenos tanto internos como externos que se pueden producir por lo tanto, las variables están diseñadas específicamente para resolver esta simulación. De esta manera se consigue que el problema se asemeje a un servicio de transporte habitual en autobús.

La variabilidad de este tipo de transporte deja en el aire una amplia posibilidad de factores a tener en cuenta. En un futuro, sería interesante ir añadiendo todas estas restricciones y factores para ser mucho más aproximados con los resultados. Por el momento, a partir de la idea principal del proyecto, las variables que se han utilizado en este artículo vienen explicadas a continuación.

7.1. Descripción de las variables

7.1.1. Constantes y parámetros

Para empezar, es importante distinguir entre constante y parámetro. Cuando se habla de una constante se hace referencia a un valor fijo. En otras palabras, es un valor que ya está establecido y no varia durante la resolución de un problema a no ser que sea especificado. Un parámetro es, en general, una característica que sirve para definir un sistema en particular. Es un valor que viene incluido dentro de una función y es necesario para definirla.

Parámetros de flujo

Los primeros que van a ser definidos son los valores medios de los flujos de pasajeros durante todo el servicio, que pueden ser representados con el esquema de la figura 7.1.

Estos valores están definidos para una parada i y vienen proporcionados por un programa externo, que en este caso se trata de AMPL.

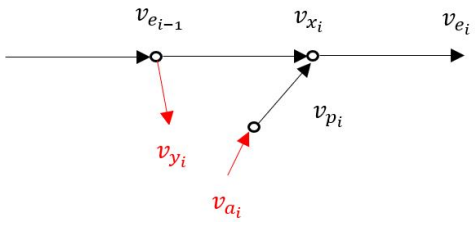

Figura 7.1: Valores medios de los flujos de los pasajeros en la parada i.

- v_{a_i}, v_{y_i} Estos valores representan el total de pasajeros que llegan a la parada i y el total de pasajeros que bajan en la parada i respectivamente durante un horizonte de tiempo $(J+1)h$. Son flujos externos al sistema en forma de entradas y salidas totales a la parada i.

  El valor de v_{a_i} sirve para determinar el número aleatorio de llegadas A_i^j a la parada i para cada bus j.

$$A_i^j \sim Poisson\ de\ tasa\ \lambda_i^j$$

  El valor medio de A_i^j viene dado por:

$$E[A_i^j] = \frac{v_{a_i}}{(J+1)h}\varphi_i^j$$

  Y la tasa λ_i^j vale:

$$\lambda_i^j = \frac{v_{a_i}}{(J+1)h}$$

  Por otro lado tenemos que el valor de v_{y_i} sirve para determinar el número aleatorio de bajadas D_i^j a la parada i para cada bus j, según una ley binomial:

$$D_i^j \sim Binomial(\alpha_i)$$

  Donde la proporción α_i, que viene dada por:

$$\alpha_i = \frac{v_{y_i}}{v_{e_{i-1}}}$$

- v_{ei}, v_{p_i}, v_{xi} Estos valores representan el total de pasajeros que salen de la parada i, el total de pasajeros que suben en bus a la parada i y el total de pasajeros que están dentro del bus en los estacionamientos de los buses a la parada i respectivamente durante un horizonte de tiempo $(J+1)h$. Estos valores sirven para posteriormente ser contrastados con los resultados de la simulación.

Parámetros derivados o secundarios

- λ_i Tasa de llegadas de los pasajeros a la parada i.
- α_i = Fracción de pasajeros que bajan a la i de entre los que están dentro del bus.

Parámetros estructurales

- I, J Numero de paradas y de servicios respectivamente.
- κ_p, κ_d Tiempo de subida y de descenso respectivamente de un pasajero en los buses. Normalmente se asume que $\kappa_p = 5s$, $\kappa_d = 2{,}5s$.
- h Tiempo (en minutos) entre dos autobuses consecutivos saliendo de la parada 1. Este tiempo recibe el nombre de *headway*. Inicialmente se considera constante.
- c Capacidad en número de plazas (máximo numero de pasajeros) que hay en los buses que operan los J servicios. Se supone común a todos los buses/servicios.
- $d^j_{i,i+1}$ Tiempo de recorrido entre la parada i y la $i+1$ para el servicio j. Inicialmente se considera como constante, pero puede pensarse en generarse de manera aleatoria.

$$0 \leq i \leq I-1; \qquad d^j_{0,1} = 0; \qquad 1 \leq j \leq J$$

- $\epsilon_i^+, \epsilon_i^-$ Tiempo de .$^{\text{en}}$cochado/"desencochado"(refiriendo al tiempo de maniobra en cuestión) en la parada i.
- π^+, π^- Tiempo de abrir y cerrar las puertas del autobús respectivamente. Se supone común a todos los buses/servicios. Si se hace una versión con diferentes tipos de buses, entonces se tendría que especificar las características de cada modelo de bus para tenerlas en cuenta.
- a_i, f_i Incremento de tiempo por frenada/arranque en llegar/salir de la parada. Se tiene que tener en cuenta que las distancias d_{i-1} se ha de considerar que están calculadas desde final de parada a final de parada.

7.1.2. Variables

Una variable representa todo aquello que este sujeto a algún cambio.

- t_i^j Instante de llegada del servicio j a la parada j.
- $t_i^{S_j}$ Instante de inicio del servicio del bus j a la parada i. Esta variable incluye el encochado a la plataforma, el tiempo de abrir y cerrar puertas, el tiempo de subidas y bajadas de los pasajeros y el abandono del bus de la parada i.
- θ_i^j Instante de salida del servicio j de la parada i.
- x_i^j Tiempo de servicio de los pasajeros del servicio j en la parada i. Es el tiempo que tardan en subir/bajar los pasajeros de los autobuses una vez este ha accedido a la plataforma y ha abierto las puertas.
- e^+, e^- Tiempo de maniobra por encochar/desencochar del bus en función de las circunstancias.
- p^+, p^- Tiempo de abrir y cerrar puertas del bus en función de las circunstancias.
- E_i^j Número de pasajeros en el servicio j después de salir de la parada i.
- D_i^j Número de pasajeros que bajan del servicio j en la parada i. Esta variable se distribuye según una $Binomial(n, p)$, por lo tanto para su obtención se utiliza un procedimiento de generación de variables aleatorias.
- X_i^j Número de pasajeros presentes en el bus j después de salir de la parada i.
- P_i^j Número de subidas al servicio j en la parada i.
- N_i^j Número de pasajeros en la parada i en salir el bus $j-1$. $N_i^1 = 0$ por convenio.
- A_i^j Número de pasajeros que llegan entre el periodo $[\theta_i^{j-1}, \theta_i^j]$, es decir, des de la salida del servicio anterior a l'instante θ_i^{j-1} hasta la salida del actual servicio hasta el instante θ_i^j. Esta variable se distribuye según una $Poisson(\lambda)$, por lo tanto para su obtención se utiliza un procedimiento de generación de variables aleatorias.
- C_i^j Capacidad en número de pasajeros del servicio j en la parada i. Se cumple la siguiente igualdad:

  $$X_i^j + C_i^j = c$$

  donde s es la capacidad física del bus en número de plazas. Se entiende como el número de pasajeros que pueden subir al bus una vez hayan bajado los D_i^j.
- R_i^j Número de pasajeros mínimo que atenderá el bus j a la parada. Más adelante se expondrán algunas cuestiones a tener en cuenta de esta variable. i.

- $tSBUS$ Tiempo de servicio del bus en función de las circunstancias.
- y_i^j Tiempo necesario para poder hacer subir la cantidad de pasajeros R_i^j
- $tSBUS$ Tiempo entre la subida del último pasajero del bus j y la salida del bus j. Esta variable ha de generarse para el procedimiento $A[]$.

Para entender mejor el concepto de ϕ_i^j se va a ver el esquema de la figura 7.2.

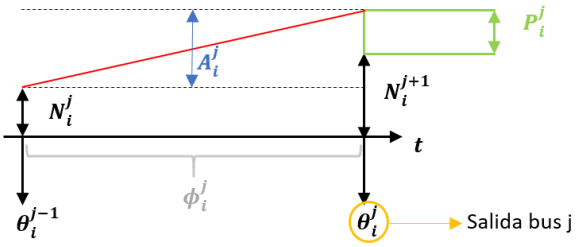

Figura 7.2: Esquema explicativo de ϕ_i^j

Explicación de ϕ_i^j y N_i^j. Se puede ver que en el periodo de tiempo comprendido entre las salidas de dos servicios consecutivos en la parada i $(\theta_i^{j-1}, \theta_i^j)$, se puede dar el caso que se queden pasajeros sin subir al servicio $j-1$ y deban esperar en la cola para ver si pueden o no subir al siguiente servicio j (N_i^j). Mientras tanto continúan llegando pasajeros (A_i^j) que incrementa el número de usuarios que esperan en la parada por su servicio. La variable ϕ_i^j comprende todo este periodo de tiempo y viene definido por el tiempo resultante de sumar a la diferencia entre el instante de inicio del servicio j a la parada i y el instante de salida del bus $j-1$ a la misma parada, el tiempo necesario para hacer subir a los pasajeros y el tiempo entre la última subida de del pasajero del bus $j-1$ y la salida del bus j.

Explicación de R_i^j. En principio es difícil de determinar, como variable en un procedimiento de simulación, cual es el tiempo de servicio de pasajeros en un bus. La razón es que, aunque lo común sería pensar que cuando el bus accede finalmente a la plataforma hay un número determinado de pasajeros para que suban y otro número de pasajeros para que bajen (y que por tanto el tiempo de operación de subidas y bajadas puede determinarse), mientras el bus está parado para que puedan subir/bajar pasajeros, puede pasar que lleguen nuevos pasajeros que incrementen el tiempo de servicio.

También puede darse el caso que, aunque el conductor haya cerrado las puertas del autobús, se puede presentar de repente un pasajero que acaba siendo atendido (haciendo que el conductor vuelva a abrir las puertas para que pueda subir), o bien simplemente, justo antes de que el conductor cierre las puertas aparezca un pasajero que se aproxima

corriendo y el conductor lo espera con las puertas abiertas durante un tiempo extra. Esta situación la denominaremos como "piedad del conductor". En un principio no se tendrá en cuenta este factor.

De todas formas, se tendrá en cuenta un alargamiento del tiempo de servicio mediante la variable y_i^j. Supongamos que el bus $j+1$ hace cola en la parada, es decir, que cuando el bus llega a la parada se encuentra la plataforma ocupada por otro bus del que están subiendo y bajando pasajeros y por tanto no puede empezar con la maniobra de acceso a la plataforma. En este caso $t_i^{j+1} < \theta_i^j$. En salir el bus j, el siguiente bus $j+1$ encontrará N_i^{j+1} pasajeros en la parada i. Si la capacidad del bus $j+1$ es de c_i^{j+1}, entonces tendrían que subir $y = minN_i^{j+1}, C_i^{j+1}$ pasajeros.

7.1.3. Ecuaciones de balance del número de pasajeros en la parada *i*

$$
\begin{array}{lll|l}
E_i^j = P_i^j + X_i^j, & 1 \le j \le J-1, & & N_i^1 = 0,\ 1 \le i \le I \\
X_i^j = E_{i-1}^j - D_i^j, & 1 \le j \le J,\ \ 2 \le i \le I-1,\ 1 \le j \le J & & D_1^j = 0,\ 1 \le j \le J \\
N_i^{j+1} = A_i^j - P_i^j + N_i^j, & 1 \le j \le J,\ \ 1 \le i \le I, & & P_I^j = 0,\ 1 \le j \le J \\
C_i^j = c - X_i^j, & 1 \le j \le J,\ \ 2 \le i \le I-1, & & X_1^j = X_I^j = 0,\ 1 \le j \le J
\end{array}
$$

$$\mathbf{P_i^j = min\{A_i^j + N_i^j, C_i^j\} \ \Rightarrow \ E_i^j \le c}$$

7.2. Creación del algoritmo

A continuación se escribirá el algoritmo básico que se ha utilizado en el desarrollo del código.

- **Inicialización**

  $N_i^1 = 0;\ \ z_i^0 = 0;\ \ \theta_i^0 = 0,\ \ 1 \leq i \leq I$

  $D_1^j = P_I^j = X_1^j = X_I^j = E_0^j = N_1^j = d_{01}^j = 0, \theta_0^j = (j-1)h,\ \ 1 \leq j \leq J.$

- Para $j = 1, .., J$

  - Para $i = 1, .., I$

    1. Calcular D_i^j:
       - Si $i < I$ entonces $D_i^j = \mathsf{Bino}(E_{i-1}^j, \alpha_i)$; contrariamente $D_I^j = E_{I-1}^j$;
    2. - $e^+ = \epsilon_i^+$; $e^- = \epsilon_i^-$; $p^+ = \pi_i^+$; $p^- = \pi_i^-$;
    3. - $X_i^j = E_{i-1}^j - D_i^j$;
    4. - $C_i^j = c - X_i^j$;
    5. - $R_i^j = \mathsf{min}\{N_i^j,\, C_i^j\}$;
    6. - $t_i^j = \theta_{i-1}^j + d_{i-1,i}^j + a_{i-1} + f_i$;
    7. - $t_i^{S_j} = \mathsf{max}\{\theta_i^{j-1}, t_i^j\}$;
    8. - Si $R_i^j > 0$ entonces $y_i^j = \kappa_p R_i^j + \mathrm{x}_i$;
    9. - Si $R_i^j = 0$ entonces $y_i^j = 0$;
    10. - $\varphi_i^j = t_i^{S_j} - \theta_i^{j-1} + y_i^j + z_i^{j-1}$;
    11. - $A_i^j = \mathsf{A}[\varphi_i^j, z_i^{j-1}]$ ($\longrightarrow\ z_i^j$)
    12. - $P_i^j = \mathsf{min}\{A_i^j + N_i^j, C_i^j\}$;
    13. - $E_i^j = P_i^j + X_i^j$;
    14. - $N_i^{j+1} = A_i^j + N_i^j - P_i^j$;
    15. - $x_i^j = \mathsf{max}\{\kappa_d D_i^j, \kappa_p P_i^j\}$;
    16. - Si $x_i^j = 0$ entonces { $p^+ = 0$; $p^- = 0$; }
    17. - $tSBUS = e^+ + p^+ + x_i^j + p^- + e^-$;
    18. - $\theta_i^j = t_i^{S_j} + tSBUS$;
    19. - Cálculo estadísticos (detallados más adelante)

- **Fin**

Este algoritmo está pensado para recorrer una sola línea de metro, por lo tanto la idea es, a partir de este código, repetir el procedimiento tantas veces para todas las líneas de buses que se tengan disponibles. Además, se ha diseñado con el fin de realizar simultáneamente más de una simulación y así poder evaluar su funcionamiento dependiendo del nivel de congestión del sistema.

Como bien se ha comentado, este algoritmo presenta la estructura simple de la simulación de una sola línea de transporte, con sus respectivos buses y paradas. El código que se implementa en Python está diseñado para recorrer estas sentencias a lo largo de cinco bucles distintos, los cuales podemos definir a continuación de la siguiente manera:

1 El bucle más externo recorre un número específico de **semillas**. Esto quiere decir que, por cada bucle de simulación que el programa ejecute de manera completa, se realizará un numero determinado de simulaciones sobre ella misma cambiando simplemente los valores aleatorios que se generan. De esta manera se obtienen distintos resultados, debiendo ser razonablemente similares entre ellos, de una misma simulación. Esto sirve para evaluar una misma simulación con distintas generaciones aleatorias, cuyos valores vendrán proporcionados por un cambio en el valor de la semilla.

2 El segundo bucle recorre un número establecido de **simulaciones**. En un primer momento se construyó una red simple con una sola simulación pero a medida que se avanzaba en la construcción del código este se ha ido implementando para poder realizar hasta un número de 10 simulaciones o más, de manera que fuese posible evaluar la red de transporte a lo largo de diferentes valores para λ y así poder calcular los estadísticos en diferentes entornos de congestión de pasajeros.

Para actualizar los valores de λ a lo largo de las diferentes simulaciones, lo que se hace es aplicar un método interpolador. Se parte de λ^0, que es la que se supone más pequeña, y a partir de ese valor se quiere llegar hasta $\hat{\lambda}$, valor correspondiente para la última simulación. Por ejemplo, si el número de simulaciones es $n = 10$, se define λ^q (con $q = 0, 1, 2, ..., n-1$) como:

$$\lambda^q = \lambda^0 + (\hat{\lambda} - \lambda^0)\frac{q}{n-1}$$

De esta manera se van obteniendo los valores de este parámetro por cada simulación. El valor $\hat{\lambda}$ para la última simulación viene descrito por la siguiente expresión:

$$\hat{\lambda} = x\frac{J * C}{(J+1)h} \tag{7.1}$$

Para evitar situaciones de acumulación de pasajeros masiva, se le aplica una reducción a la fórmula multiplicándola por la constante x, que toma valores entre 0.8 y 0.95.

3 El tercer bucle recorre las **líneas** que existen en la red de transporte.

4 El siguiente bucle va pasando por los **servicios** J que recorren la línea que se está evaluando.

5 Por último, el bucle más interno recorre las **paradas** I de cada línea por cada bus.

Una vez creada la estructura básica del simulador, el siguiente paso es determinar las 18 sentencias del algoritmo anterior de manera que se defina una tabla que vaya recogiendo los resultados por parada y bus dentro de cada línea de transporte.

De entrada se han tenido que inicializar algunas de las variables de manera que su valor inicial sea cero. Es lógico pensar que en la primera parada no pueden producirse bajadas de pasajeros, ni pueden haber usuarios presentes en el bus provenientes de una parada anterior a la primera. A pesar de ser conceptos sencillos, es muy importante que estos detalles sean especificados previamente en la simulación para evitar incongruencias en los resultados.

- $N_i^1 = 0 \quad \rightarrow$ Durante el primer servicio no pueden haber quedado pasajeros sin subir de un servicio anterior, puesto que no existe.
- $z_i^0 = \theta_i^0 = 0 \quad \rightarrow$ No existe el servicio 0, por lo tanto no puede haber tiempo entre la subida del último pasajero en el servicio 0 y el servicio 1 ni instantes de salida del servicio 0.
- $D_1^j = P_I^j = X_1^j = X_I^j = E_0^j = 0 \quad \rightarrow$ No pueden bajar (D) pasajeros en la primera parada , de igual manera que tampoco pueden subir (P) en la última. Esto hace que no existan los tiempos de servicio (X) en esta parada. Tampoco pueden haber pasajeros en el bus (E) que provengan de la parada 0.
- $d_{01}^j = 0 \quad \rightarrow$ Como no hay parada anterior a la primera, la distancia entre la parada 0 y la 1 es de 0.

Con la idea de poder contrastar los tiempos de realización del simulador por cada ejecución, se han implementado unas sentencias que, una vez que el código ha compilado correctamente, indican el tiempo total que ha llevado a cabo su ejecución.

7.2.1. Cálculo de estadísticos

Con el código de la simulación ya construido, es interesante documentar su funcionamiento probando la sentencia con datos reales, de esta manera se podrá hacer una evaluación de la actividad de la red de transporte bajo distintas situaciones de congestión.

Una de las principales inquietudes que tiene este apartado es observar los volúmenes de congestión que pueden producirse tanto de pasajeros como de unidades de transporte. Con este fin, a continuación se describirán todos los cálculos estadísticos necesarios para estimar cada tipo de línea de espera coordinados a los niveles de demanda del sistema y los tiempos medios de espera para pasajeros y para buses. De esta manera se podrá hacer una evaluación global del funcionamiento de todo el sistema.

Estadísticos para colas de pasajeros

Cada vez que llega un pasajero a una parada se genera un tiempo de espera desde que llega hasta que consigue subir a un autobús. El pretexto de estos cálculos es estimar en promedio la demanda del sistema y el tiempo medio de espera por pasajero en una parada.

El factor de carga es un indicador básico para este sistema y se define como el cociente entre el número de pasajeros que intenta subir al autobús y la capacidad libre del bus:

$$\rho_i^j = \frac{A_i^j}{C_i^j}$$

Para obtener un mejor indicador es óptimo calcularlo de manera agreda, de tal forma que se obtenga un factor de carga individual por cada parada:

$$\rho_i = \frac{\sum_{j=1}^{J} A_i^j}{\sum_{j=1}^{J} C_i^j} \quad ; \quad \rho < 1$$

Este factor indica cómo va de cargada la parada i-ésima a lo largo de todo el horizonte

de tiempo que se determine. Por tanto, como bien se ha definido antes, para una parada i-ésima, cualquier magnitud vendría representada con respecto de ρ_i. Valores de ρ pequeños señalan poca afluencia de pasajeros, lo que denota valores de λ poco significativos y lapsos entre llegadas de pasajeros muy amplio.

Con motivo de este apartado, una cuestión de interés es saber el tiempo que esperan los pasajeros en la parada hasta poder subir a un autobús. La demora media que experimenta el pasajero en momentos de baja congestión se representa de la siguiente manera:

$$W_0 = E\left[\frac{1}{2}\phi_i^j\right](1 + CV_\phi^2)$$

Este estadístico nos calcula el tiempo medio que espera un pasajero en situación de muy poca demanda, es decir, en momentos de poca congestión de pasajeros. CV_ϕ^2 es el coeficiente de variación de ϕ, y se calcula como la división entre la desviación típica de ϕ_i y su media:

$$CV_{\phi_i} = \frac{S_{\phi_i}}{\bar{\phi}_i}$$

El valor del intervalo entre la última salida y la salida del autobús en curso ha de aumentarse por la irregularidad que pueda tener el intervalo de tiempo $(1 + CV_\phi^2)$. La variable de interés es la esperanza del tiempo de vida o de renovación observada (proporcionado por ϕ_i^j). La irregularidad que tenga esta renovación hace que los pasajeros la observen incrementada.

Anteriormente se ha tenido en cuenta la demora que se observaría en ausencia o bajo muy poca demanda. De la misma manera, se pretende calcular la demora o el tiempo medio que experimenta un pasajero en su situación real.

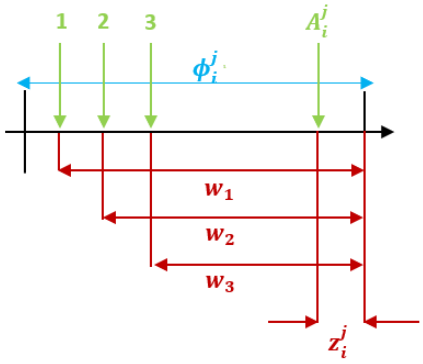

Figura 7.3: Tiempos de llegadas de los pasajeros

La figura 7.3 muestra una representación de los tiempos de espera que experimentan

los pasajeros a medida que van llegando a la parada i durante el tiempo comprendido entre la marcha del bus $j-1$ y la llegada del bus j. Durante el periodo ϕ_i^j van llegando las rutinas de pasajeros generadas aleatoriamente con la distribución Poisson. El tiempo que espera casa pasajero es el tiempo que hay entre que llega a la parada y el que consigue subir al autobús. Para el primer pasajero su tiempo de espera es w_1.

Se puede observar que desde la llegada del último pasajero a la parada, hasta que llega el autobús donde este podrá subir, existe un corto lapso de tiempo. Generalmente este tiempo se expresa con la variable z_i^j.

Si se quiere ver la demora verdadera respecto la demanda real que hay, la forma de obtenerla es a través de la misma rutina que genera las llegadas con la distribución Poisson, de esta manera se estaría generando simultáneamente el instante de llegada (w_l) de cada pasajero.

$$\sum_{l=1}^{A_i^j} w_l$$

Con este sumatorio se obtiene la demora media por pasajero. Cada vez que un pasajero llega a una parada se genera también un instante de llegada. El tiempo total esperado por todos los pasajeros que han llegado se representa con esta suma. Esto es para los pasajeros que han llegado durante ϕ_i^j. A este sumatorio hay que añadirle aquellos pasajeros que han estado esperando en la parada porque no pudieron subir en el autobús anterior (N_i^j). Con lo cual, hay que tenerlos en cuenta de la siguiente manera:

$$W_i^j = \sum_{l=1}^{A_i^j} w_l + N_i^j * \phi_i^j$$

Por lo tanto, cada vez que el autobús visita una parada, hay un incremento del tiempo total esperado que se da con esta formula .

Cuando se detallen los resultados más a delante, se verá que el sistema de espera es la parada, y se obtendrá un factor de carga por cada una de ellas. Las W_i son los tiempos de permanencia de los pasajeros para los diferentes autobuses que han servido esa parada. Cada vez que el programa de simulación recorra un servicio por las diferentes paradas, este irá calculando W_i^j en una tabla.

Cuando finalicé la sentencia de W_i^j, se debe fijar la fila correspondiente a la parada i. Para lograr calcular la demora media, se debe de dividir por el número total de pasajeros, obteniéndose la siguiente igualdad:

$$\bar{W}_i = \frac{\sum_{j=1}^{J} W_i^j}{\sum_{j=1}^{J} A_i^j} = \frac{W_i}{\sum_{j=1}^{J} A_i^j}$$

$\bar{W}_i$ es el tiempo medio por pasajero de espera hasta poder subir en la parada i-ésima. Para esta parada se obtendrá un factor de carga ρ_i y al mismo tiempo un W_i, por lo que si se dibujara gráficamente se acabaría obteniendo un punto de la recta.

Estadísticos para colas de autobuses

El cálculo de los estadísticos de los buses trata de generar o ir calculando las variables que son tiempo de permanencia en cada unas de las paradas de los autobuses y, al mismo tiempo, calcular las longitudes de cola que se pueden producir.

El tiempo de permanencia del autobús j en la parada i se calcula como el instante de salida del servicio menos el de llegada (mismo servicio en la misma parada):

$$w_{ij}^B = \theta_i^j - t_i^j$$

Esta variable se va a ir acumulando en otra más grande W_i^B , la cual va a contener el tiempo total de todos los autobuses que han parado en la parada i:

$$W_i^B = W_i^B + w_{ij}^B$$

En la parada i cada uno de los servicios habrá permanecido un tiempo determinado. Esta variable acumula todos los tiempos de todos los buses que han pasado en esa parada.

También se calcula la variable L_i^B que, aparentemente, hace lo mismo que W_i^B:

$$L_i^B = L_i^B + w_{ij}^B$$

Al finalizar los bucles de recorrido de j y i para todos los buses y todas las paradas estas variables contendrán la misma información: tiempo total de permanencia de todos los autobuses que han pasado por la i-ésima parada. En este momento, al finalizar los bucles, la variable se introduce en un algoritmo aparte y la variable W_i^B se divide entre el número total de autobuses que han pasado por la parada i:

$$W_i^B = \frac{W_i^B}{J}$$

Por lo tanto esto pasa a ser el tiempo medio de permanencia en la parada i por unidad de transporte.

La variable $L^B_{T_i}$ se divide por el tiempo entre la llegada del último autobús y la llegada del primero, con lo cual se calcula la longitud media de la cola (número medio de autobuses) que hay en la parada i-éssima.

$$L^B_{T_i} = \frac{L^B_{T_i}}{t^J_i - t^1_i}$$

Mientras el sistema esté congestionado, los tiempos de permanencia serán prácticamente la suma de los $tSBUS$ y $L^B_{T_i}$ será aproximadamente la unidad. Pero a medida que se pruebe con valores de demanda que provoquen más congestión en el sistema, el valor de W^B_i comenzará a crecer y superará los valores de $tSBUS$.

De la misma manera que se calculan los factores de carga para las colas de pasajeros, el factor de carga correspondiente de la cola de buses a la parada i es:

$$\rho^B_i = (\frac{(J-1)\sum_{j=1}^{J} x^j_i)}{J(t^J_i - t^1_i)})$$

Además, para poder representar la demora normalizada $\tilde{W}^B_i$ para la cola de autobuses, se realiza el siguiente cálculo:

$$\tilde{W}^B_i = \frac{W^B_i}{\sum_{j=1}^{J} x^j_i}$$

Capítulo 8

Resultados de la simulación

Una de las utilidades de más interés que proporciona este simulador es la capacidad de calcular los tiempos medios de espera tanto de clientes como de unidades de transporte bajo unas condiciones de servicio determinadas y de manera conjunta. Además, se han diseñado unas funciones que pueden hacer representaciones gráficas para proporcionar una visualización más clara de los resultados.

Hay ciertos valores de entrada que determinan las características del sistema de servicio. Para un sistema de transporte en autobús se han determinado los siguientes parámetros:

- Se ha fijado que la velocidad media de cada bus sea de 50km/h y, por lo tanto, el recorrido que realiza en una hora viene determinado por: $recorrido = \frac{velocidad*1000}{60}$.
- La distancia entre paradas consecutivas viene definida en los archivos de datos que proporciona AMPL. El tiempo de recorrido entre la parada i y la $i+1$ para el bus j es el cociente entre la distancia y el recorrido.
- El tiempo de subida (kp) y bajada (kd) de un pasajero es de 5 y 2.5 segundos respectivamente.
- El tiempo que tarda el bus j en encochar (e_+) y desencochar (e_-) en la parada i es de 6 y 15 segundos respectivamente.
- El tiempo que tarda el bus en abrir (π_+) y cerrar (π_-) las puertas es de 5 segundos en total.

Las tablas de datos de los flujos de pasajeros proporcionadas por AMPL para cada uno de los ejemplos que se presentarán a continuación, quedan detalladas en los anexos de esta memoria.

Teniendo en cuenta estas características, el primer ejemplo que se presenta es la simulación de un sistema de red de transporte público urbano de autobuses con una sola línea y siete paradas. Los valores medios de los flujos de pasajeros durante todo el servicio son valores externos proporcionados por AMPL con los que se han obtenido los siguientes datos:

- Servicios = 100
- Capacidad de los buses en número de pasajeros = 100
- Tiempo entre servicios (en minutos) = 30

Con estos datos se han generado unos gráficos que representan la demora normalizada de los pasajeros respecto del factor de carga. Esta magnitud, que es la demora verdadera dividida entre la demora sin congestión, representa el número de veces la demora que tiene una parada respecto la que tendría si prácticamente no hubiese afluencia de pasajeros. Esto quiere decir que, para un factor de carga determinado y una altura en la demora normalizada de h, el tiempo de espera que experimenta el pasajero es h veces el que tendría si el sistema no estuviese congestionado. Esto da una idea de lo rápido que se carga el sistema.

Como en la última parada no se generan llegadas de clientes, no tiene sentido realizar el gráfico que visualiza la demora global de esa parada. Es por esto que en las únicas paradas donde se pueden generar colas de pasajeros son en las seis primeras.

Se ha generado una ejecución del simulador con 10 semillas para 10 simulaciones, y los gráficos que se han obtenido se presentan a continuación:

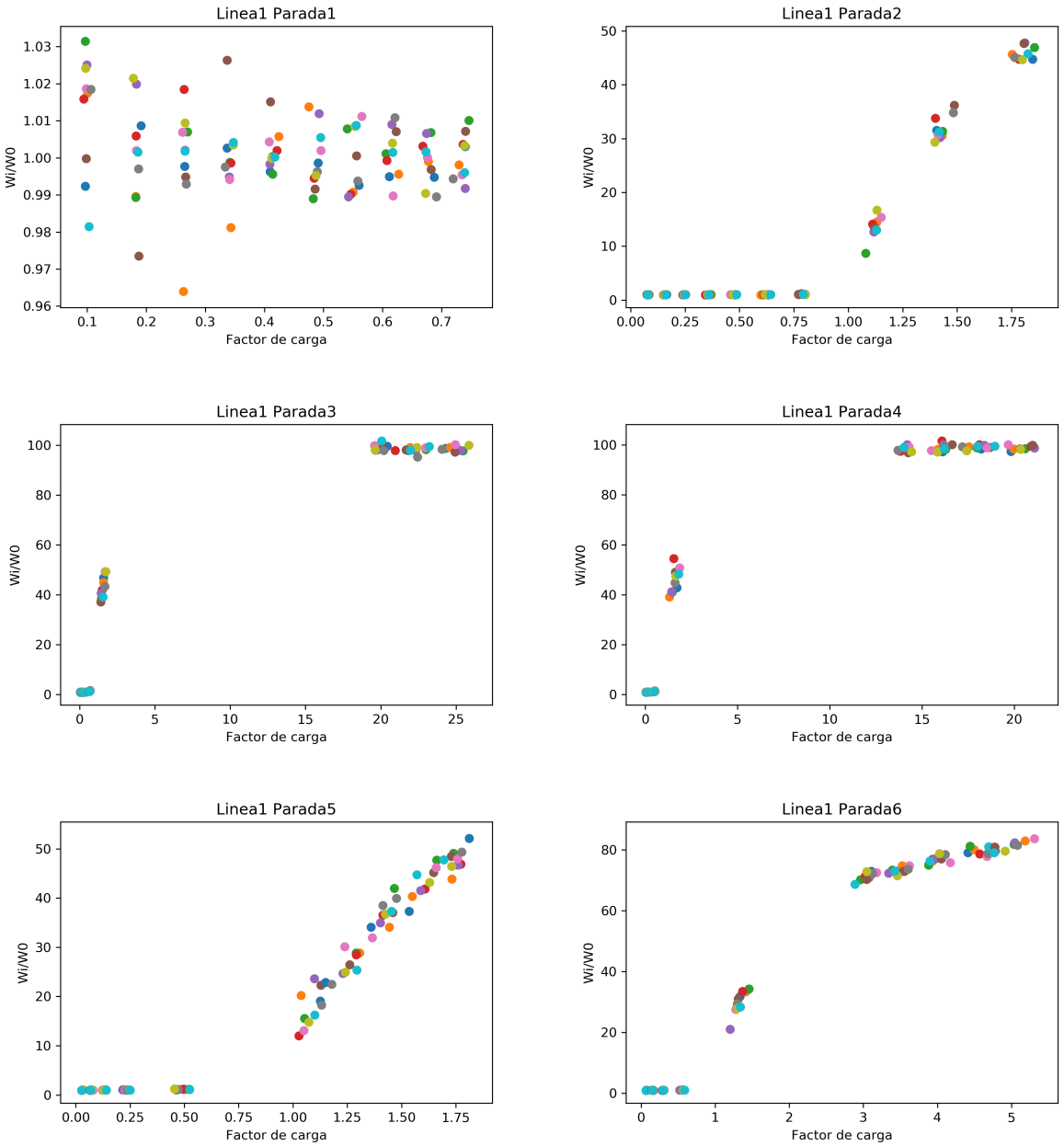

Figura 8.1: Primera prueba, cola de pasajeros: 1 línea - 6 paradas. Demora normalizada Wi/W0 versus factor de carga

La figura 8.1 presenta los resultados gráficos de la primera prueba ejecutada con el simulador. Los resultados obtenidos son a partir de realizar para un mismo valor de factor de carga 10 simulaciones con distintos valores. Esto quiere decir que para un mismo factor de carga, se han realizado 10 simulaciones con distintos datos generados para las llegadas y las salidas de los pasajeros.

En los gráficos se puede observar una tendencia plana para la primera parada, lo que indica una falta de congestión. En esta parada se puede distinguir un comportamiento más heterocedástico al principio cuando los factores de carga son más bajos, es decir, hay

una conducta en los datos a converger a medida que el factor de carga se hace más grande.

En las siguientes paradas, en cambio, se observa un comportamiento totalmente contrario. Las paradas se congestionan muy rápido a medida que el factor de carga augmenta. De hecho, en los gráficos se muestran factores de carga muy elevados. Esto es debido a la gran carga de pasajeros que está soportando la línea, debida a la llegada masiva de usuarios a las paradas, provocando una sobrecarga en el sistema y, en consecuencia, tiempos de espera muy significativos.

El siguiente ejemplo, que contiene el mismo número de servicios, misma carga del servicio y mismo tiempo entre servicios que el ejemplo anterior, presenta un sistema con menor carga de pasajeros por parada.

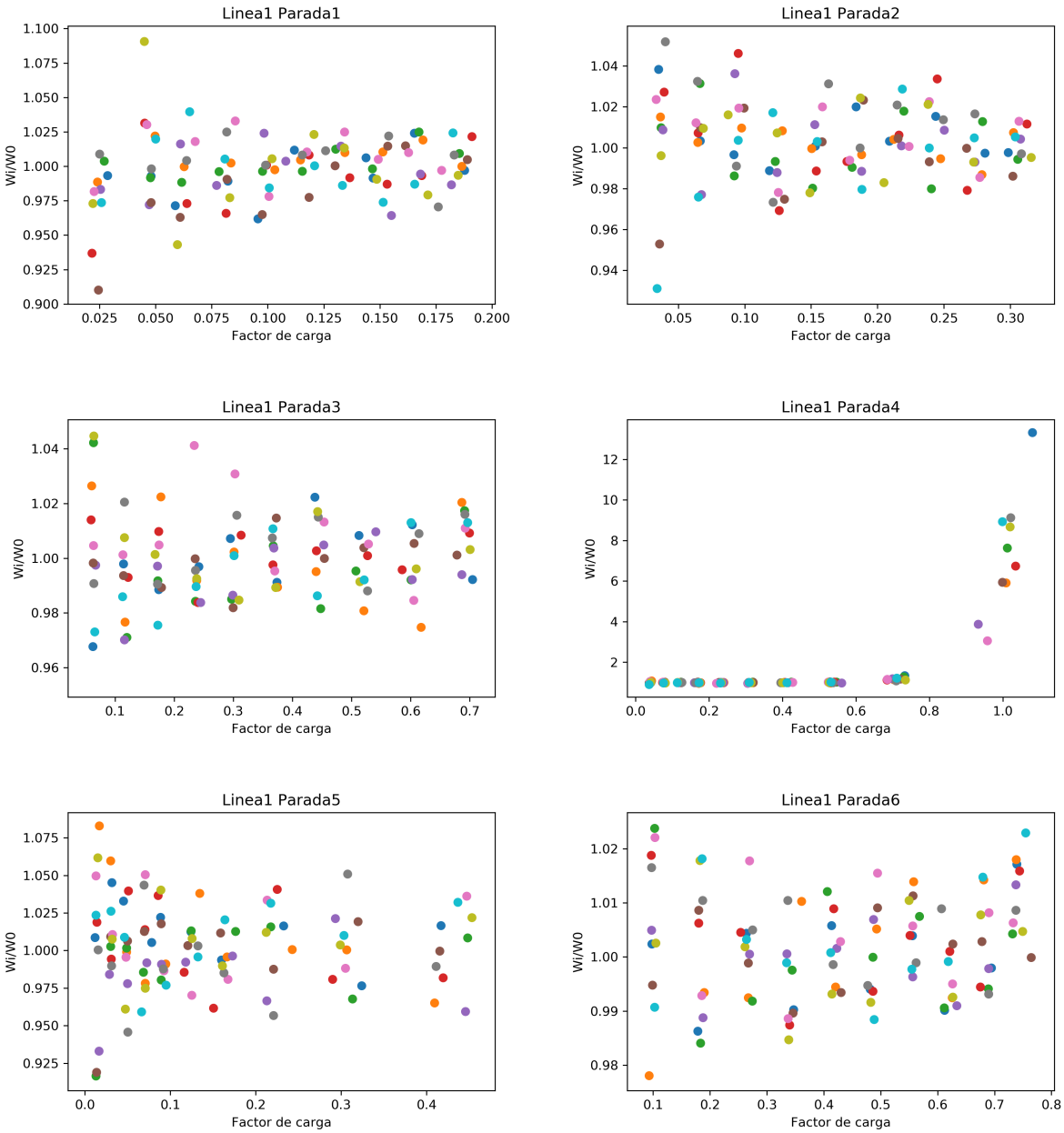

Figura 8.2: Segunda prueba, cola de pasajeros: 1 línea - 6 paradas. Demora normalizada Wi/W0 versus factor de carga

En esta segunda prueba se aprecia una tendencia lineal durante las tres primeras paradas, y unos factores de carga muy pequeños. Esto quiere decir que el sistema no sufre ningún tipo de congestión al principio de la línea a pesar de augmentar el factor de carga, por ese motivo los tiempos de espera son tan reducidos. Se observa además la misma tendencia a converger a un mismo valor a medida que el factor de carga aumenta.

En la cuarta parada aparece un cambio en la demora normalizada al augmentar el factor de carga. Además, los tiempos de espera aumentan considerablemente cuando este se acerca a la unidad. Las dos siguientes paradas vuelven a tener unos tiempos de espera

pequeños en comparación, mostrando de nuevo una tendencia plana sin congestión a lo largo de las diferentes simulaciones de los factores de carga.

El tercer ejemplo presenta exactamente la misma tabla de datos con las mismas características de servicio que el modelo anterior, la diferencia es que este simulador se ejecuta incrementando el número de servicios hasta 200. Los gráficos de las paradas se presentan a continuación:

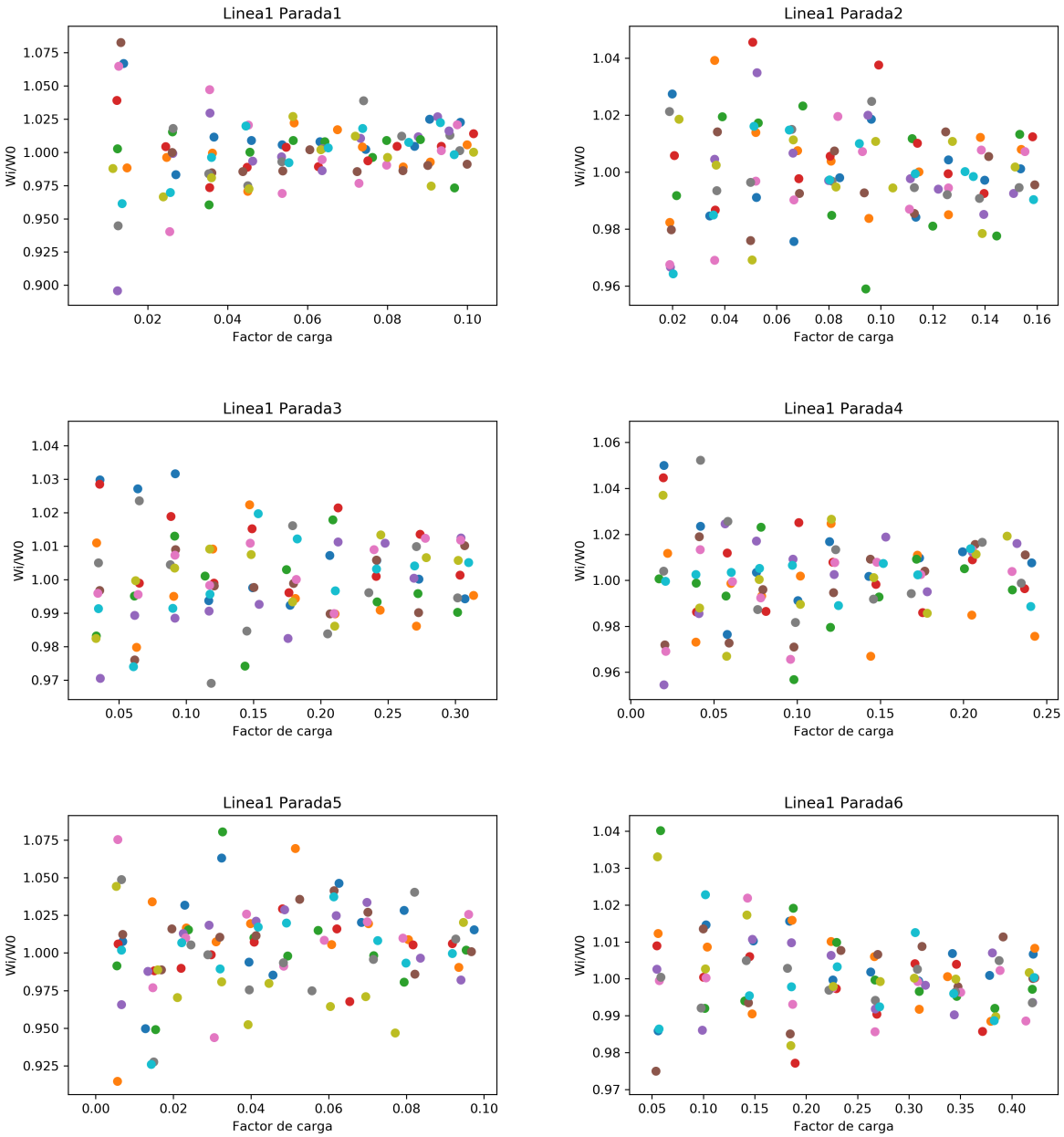

Figura 8.3: Tercera prueba, cola de pasajeros: 1 línea - 6 paradas. Demora normalizada Wi/W0 versus factor de carga

Este es un caso donde ninguna de las paradas se encuentra congestionada. Esto es debido al augmento provocado en el número de servidores. Al pasar más autobuses durante el mismo horizonte de tiempo, se producen menos colas de pasajeros y, a su vez, menores tiempos de espera hasta poder subir a un autobús.

Los dos ejemplos que se mostrarán a continuación contienen exactamente las mismas tablas de flujos para las paradas, pero presentan diferentes características de servicio.

El primer archivo contiene los siguientes registros:

- Servicios = 50
- Capacidad de los buses en número de pasajeros = 100
- Tiempo entre servicios (en minutos) = 10

Y los gráficos obtenidos para esta prueba son:

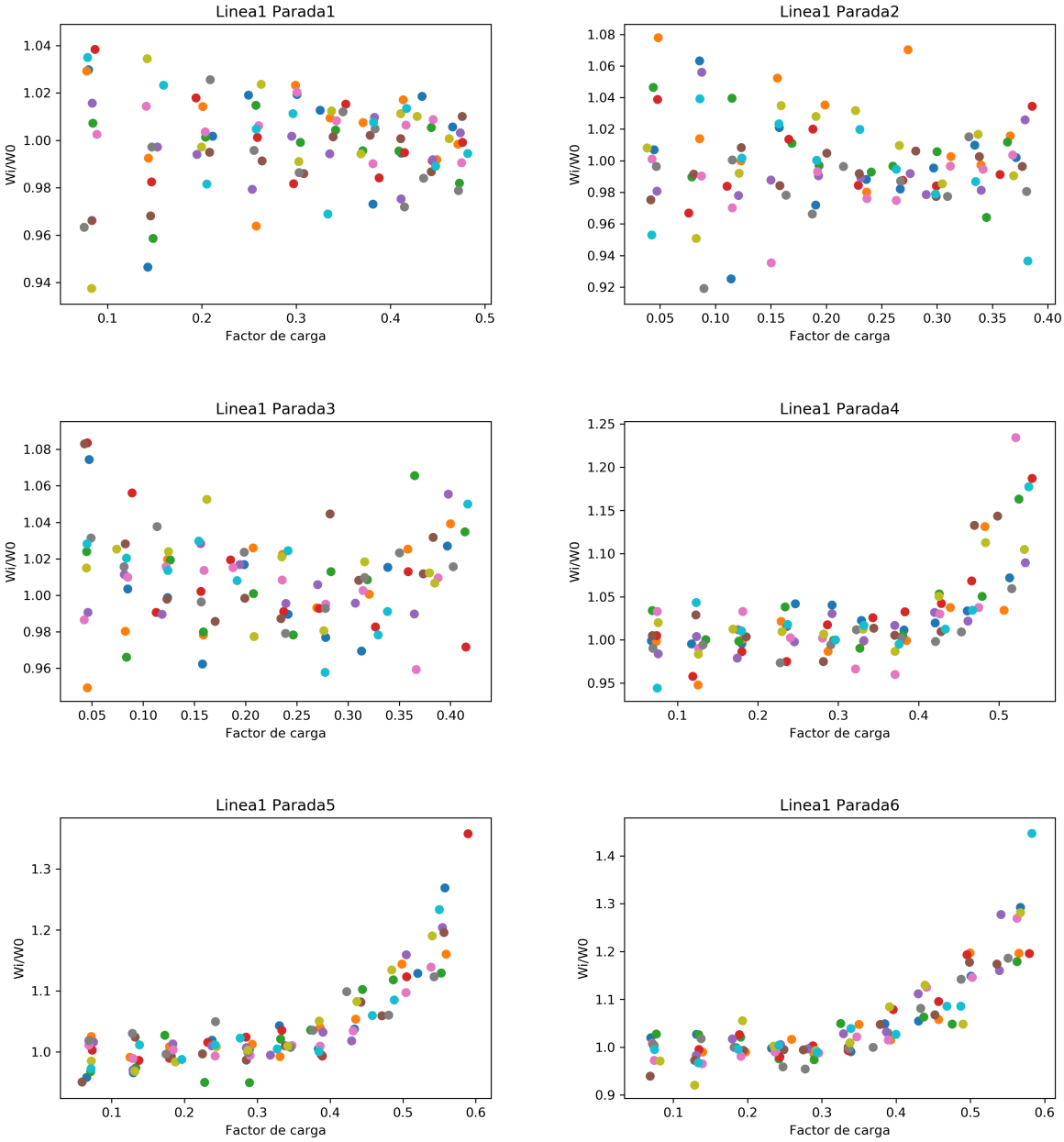

Figura 8.4: Cuarta prueba, cola de pasajeros: 1 línea - 6 paradas. Demora normalizada Wi/W0 versus factor de carga

Las primeras tres paradas no presentan congestión, pero a medida que se avanza en

la línea, se ve como las siguientes paradas empiezan a congestionarse a medida que los factores de carga se hacen más grandes.

A partir de la parada 4 se comienza a ver como los datos dibujan una curva. Esto representa el aumento de la congestión en las paradas a medida que el factor de carga se va haciendo más grande. En general, se pueden apreciar unas nubes de puntos construidas con los mismos factores de carga. Es por esto que el eje de las abscisas nunca va a ser fijo, siempre estará más o menos oscilando. Al principio, cuando el factor de carga es bajo, la oscilación es más pequeña. Si aumento el factor de carga se aumenta un poco la dispersión.

Por otro lado, las características de servicio del segundo archivo son:

- Servicios = 50
- Capacidad de los buses en número de pasajeros = 100
- Tiempo entre servicios (en minutos) = 5

Mientras que los gráficos obtenidos para estos datos son:

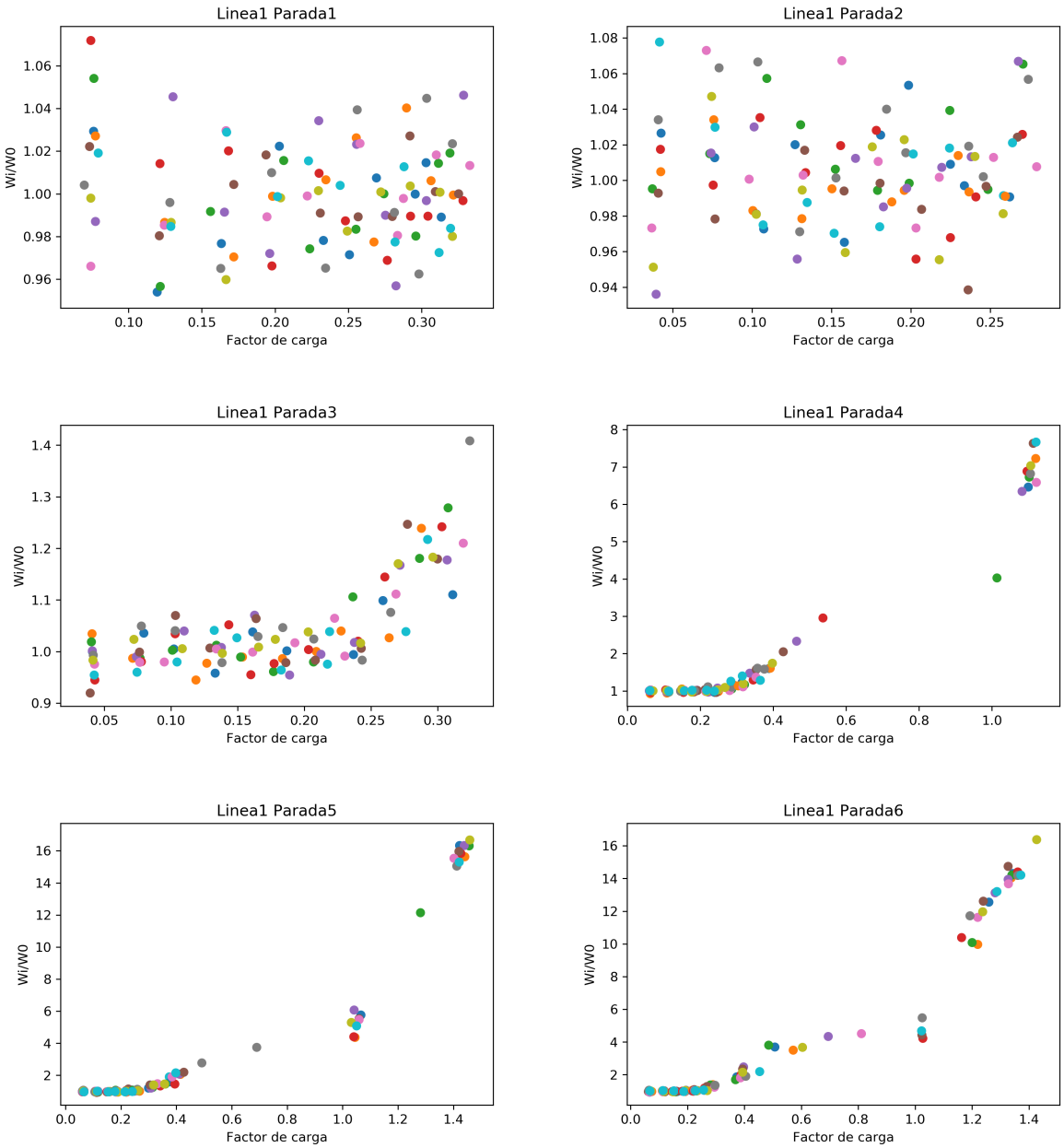

Figura 8.5: Quinta prueba, cola de pasajeros: 1 línea - 6 paradas. Demora normalizada Wi/W0 versus factor de carga

En este caso, en comparación con el ejemplo anterior, se ha reducido el tiempo entre servicios y lo que se ha obtenido es un augmento en la congestión de la línea. En los resultados del cuarto ejemplo, los servicios llegan en 500 minutos mientras que en el quinto llegan en 250 minutos, por lo que la afluencia es más alta en este último ejemplo. En consecuencia, los autobuses también llegan más rápido. Bajo este análisis superficial, debería salir que los dos ejemplos están igual de congestionados.

Además de la demora para las colas de pasajeros, se han realizado unos gráficos que representan los tiempos de espera para las colas de autobuses en los ejemplos cuatro y cinco, y los resultados son los siguientes:

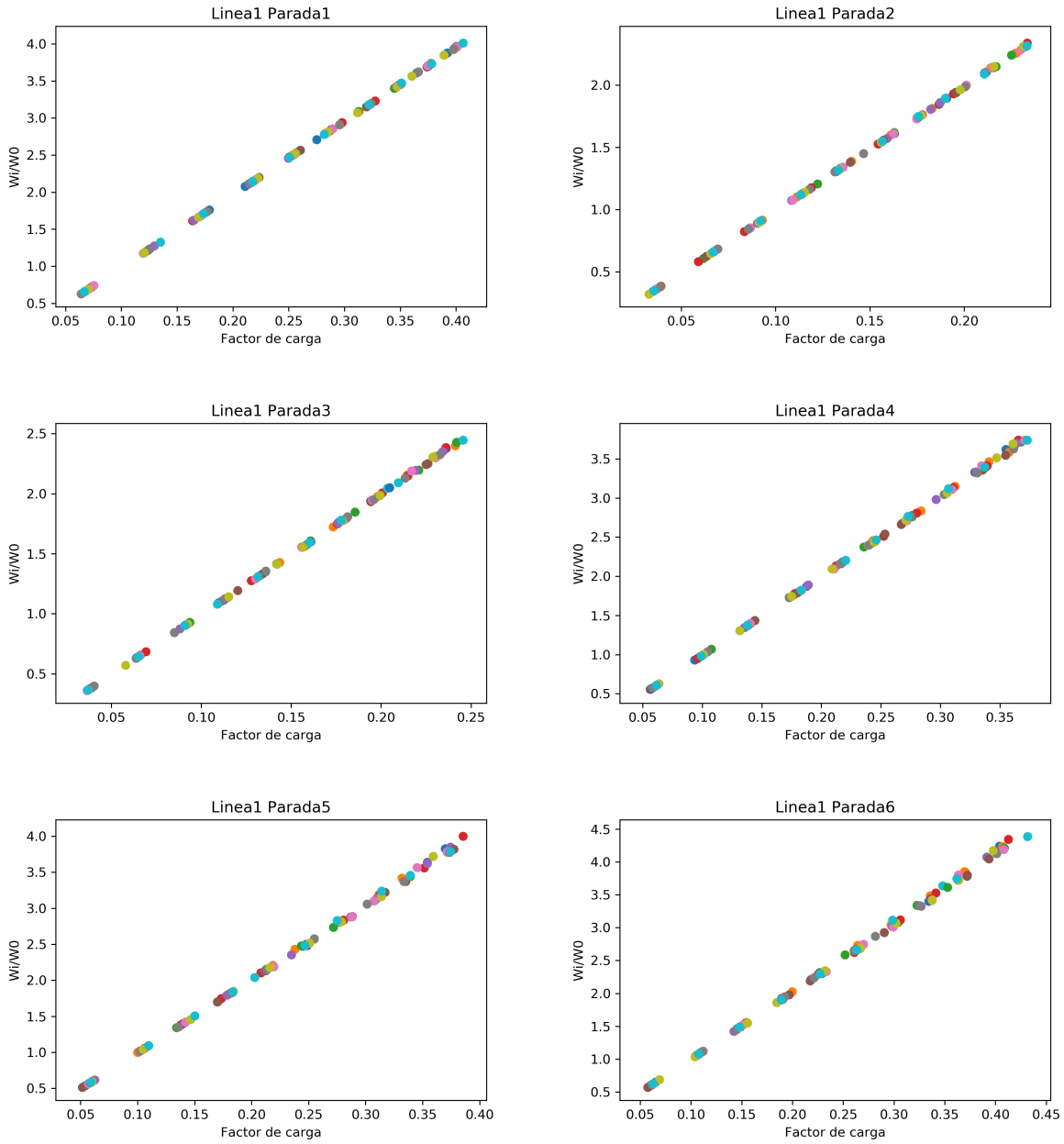

Figura 8.6: Cuarta prueba, cola de autobuses: 1 línea - 6 paradas. Demora normalizada Wi/W0 versus factor de carga

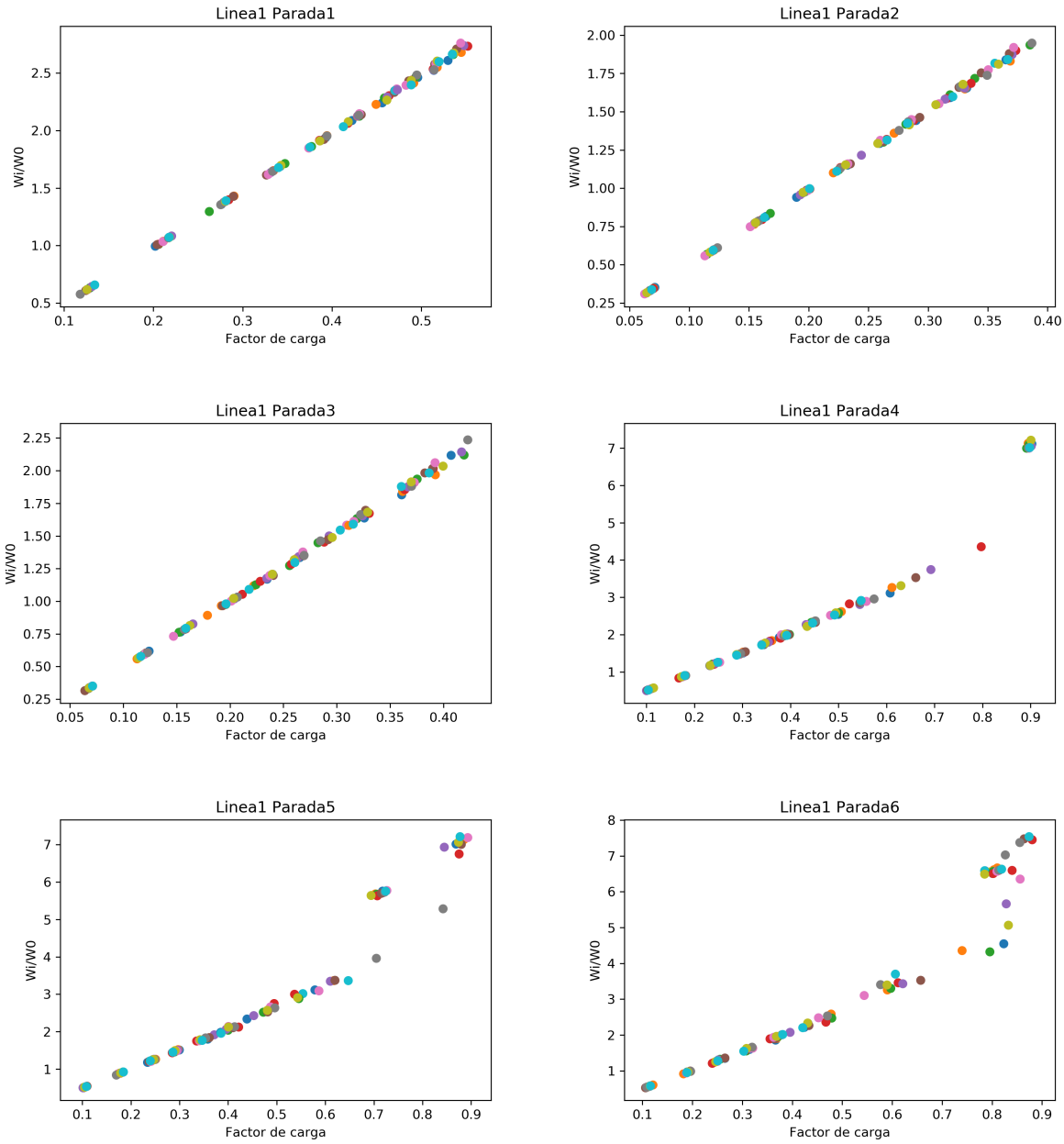

Figura 8.7: Quinta prueba, cola de autobuses: 1 línea - 6 paradas. Demora normalizada Wi/W0 versus factor de carga

Se observa claramente una tendencia lineal en los resultados. Para entender mejor el porqué de este efecto, se explicará de la siguiente manera:

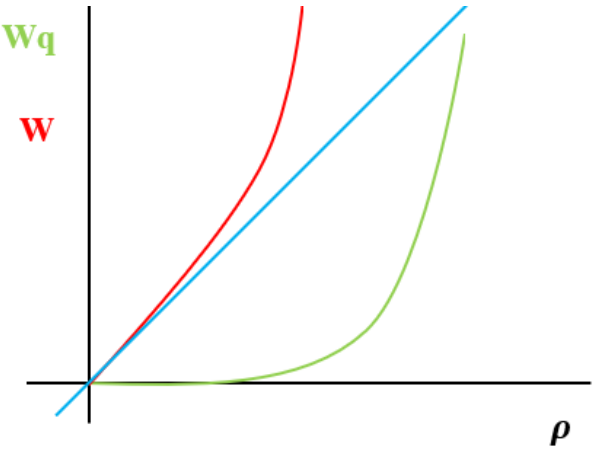

Figura 8.8: Gráfico de la demora normalizada para las colas de autobuses

La demora total del sistema se expresa como:

$$W = W_q + x = W_q + \hat{\tau}\rho$$

En la gráfica 8.8 se superponen dos tiempos: el tiempo del sistema (en rojo) y el tiempo en la cola (en verde). Se observa como el W_q es una recta plana en el origen cuando los valores de ρ son pequeños. Cuando estos empiezan a augmentar, la curva toma una forma convexa hacia arriba. El tiempo de permanencia total W, ha de ser igual que la curva verde más x.

La x representa el tiempo medio de servicio y W_q los tiempos de espera en la cola. Esta x se puede expresar como el producto del factor de carga por una constante $\hat{\tau}$ (que es el headway entre autobuses), por lo tanto $x = \hat{\tau}\rho$.

Cuando hay mucha afluencia de pasajeros, se produce un augmento en el valor de x. Por tanto, teniendo en cuenta la anterior ecuación, el término segundo es lineal en ρ, que es lo que se explica con la linea azul.

Para finalizar, adicionalmente se han representado gráficamente las longitudes medias de las colas, es decir, el número medio de autobuses, para los dos últimos ejemplos y los resultados se representan a continuación:

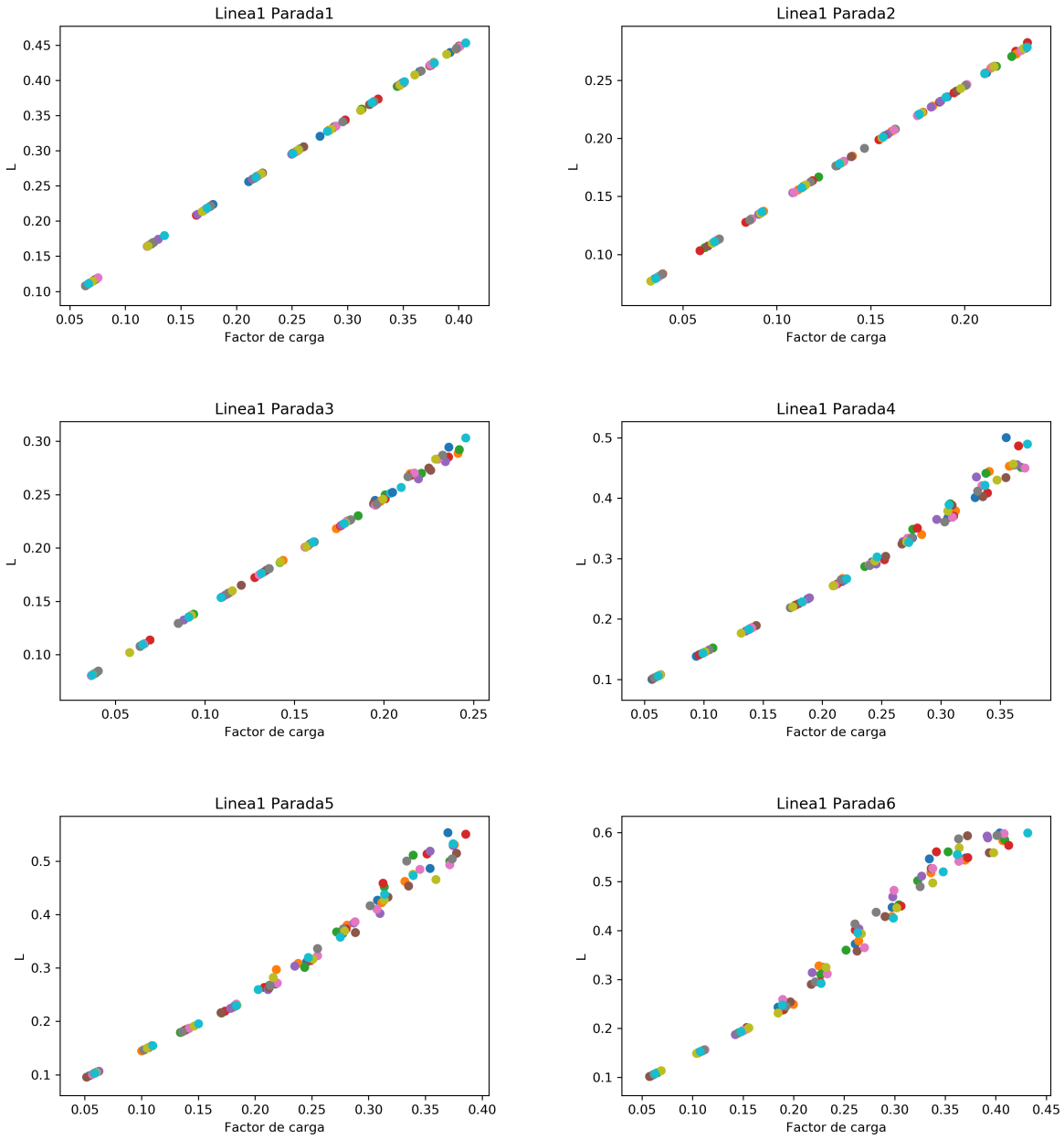

Figura 8.9: Cuarta prueba, cola de autobuses: 1 línea - 6 paradas. Longitud media de la cola versus factor de carga

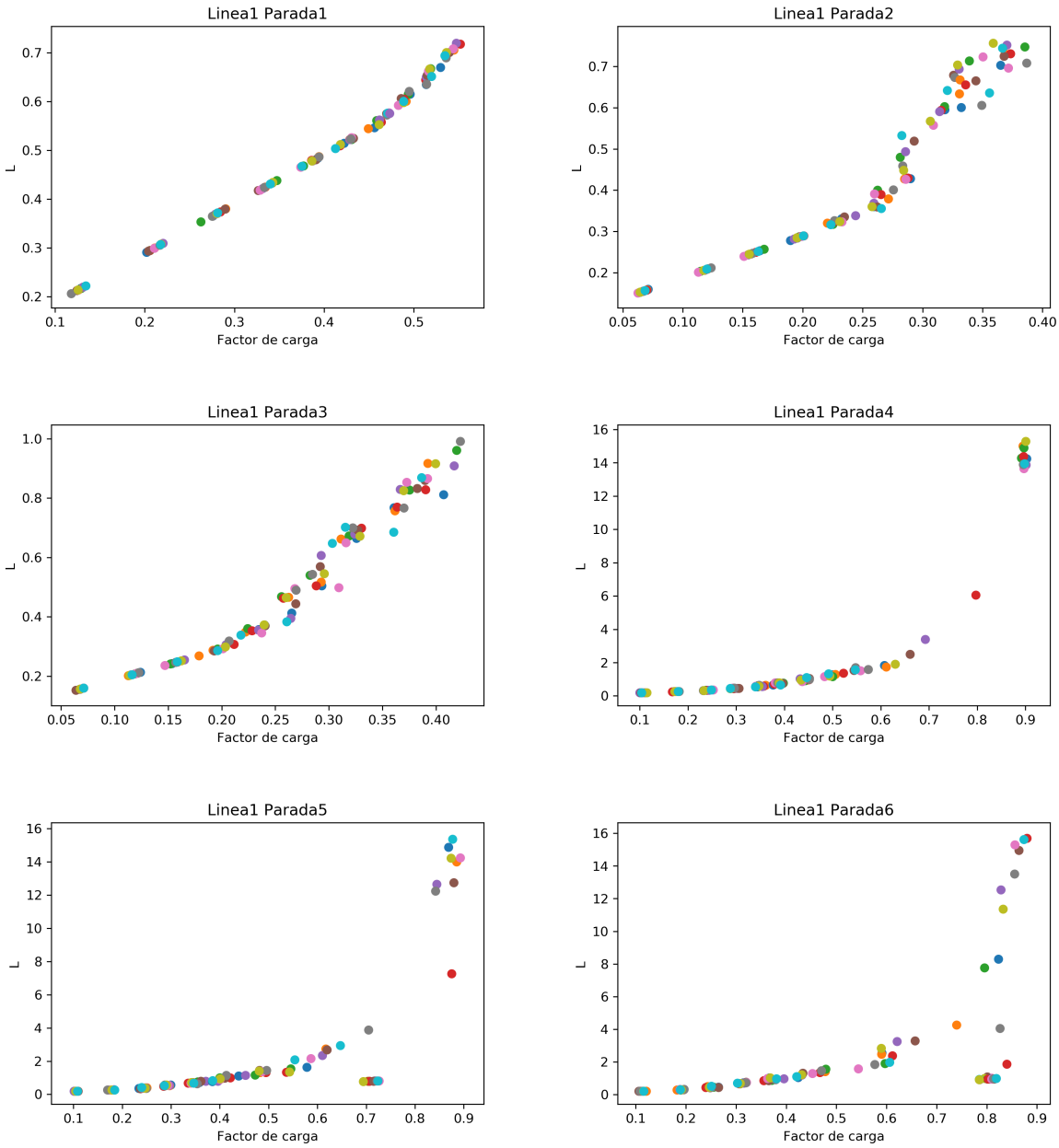

Figura 8.10: Cuarta prueba, cola de autobuses: 1 línea - 6 paradas. Longitud media de la cola versus factor de carga

Es evidente observar una diferencia en las longitudes medias de las colas de autobuses de la cuarta prueba respecto de la quinta. En este primer ejemplo se puede apreciar como apenas se encuentra congestión en la línea, pero sí se distingue un augmento en las longitudes de la cola en comparación a las de la quinta prueba. Se distingue como a medida que se avanza en la línea, las paradas comienzan a tener una mayor saturación en las colas de espera de los buses. De nuevo se describe la forma curva hacia arriba que se había podido observar anteriormente en las colas de pasajeros y esto empieza a suceder a partir de la curta parada.

Capítulo 9

Conclusiones

El principal propósito que sostiene esta práctica mantiene como objetivo la programación de un simulador capaz de replicar la actividad de una red de transporte público urbano con sus respectivas líneas, paradas y unidades de servicio. Además, bajo unas determinadas características de servicio, se busca comprender las colas que se producen tanto de pasajeros como de servicios a lo largo de cada línea mientras se calculan las demoras medias que estos experimentan.

La teoría de colas convencional afirma que la fórmula de Powell refleja una aproximación de colas con múltiples llegadas y múltiples servicios, es decir, llegadas por lotes y servicios por lotes, que no es la correcta. Esta aproximación sigue una metodología antigua y a pesar de parecer adecuada, dado que trata este tipo de colas, los tiempos de espera que se obtienen resultan estar subestimados.

Es por esto que, para poder abordar de manera apropiada una situación como la que se propone en este trabajo, se ha diseñado un simulador que reproduce los fenómenos que se ocasionan en este tipo de transporte. El código programado es capaz de gestionar diferentes volúmenes de congestión y, para cada uno de ellos, calcular y representar la demora normalizada que presenta cada red a lo largo de todas sus paradas. Para obtener el número de llegadas de pasajeros en cada parada y el de bajadas, se han implementado en el código unas funciones que simulan las distribuciones de Poisson y Binomial, las cuales generarán números aleatorios regidos por cada una de estas distribuciones.

Para poner a prueba el código y así poder evaluar si realiza las ejecuciones correctamente, el simulador, bajo unas mismas condiciones de congestión de pasajeros, realiza múltiples repeticiones con diferentes datos generados aleatóriamente. De esta manera, se puede hacer una representación de los niveles de demora que experimentan las paradas de una misma línea bajo diferentes niveles del factor de carga.

Los ejemplos que se han presentado para evaluar el simulador, ponen a prueba líneas

de transporte en distintas situaciones. Con los diferentes gráficos obtenidos se ve como la demora es totalmente aleatoria en función del factor de carga con el que se esté trabajando. Esto se debe en gran medida a que el número de autobuses que circulan en cada parada es finito. Si el intervalo de tiempo con el que se trabaja fuese muy grande, como por ejemplo de un año, las curvas que se obtendrían serían mucho más nítidas y con menos dispersión. Pero se debe tener en cuenta que no se está produciendo un trabajo de 24h continuado si no que se trabaja durante un horizonte de tiempo determinado, como por ejemplo 5h, y en consecuencia el número de servidores es limitado. Es por esto que las gráficas no son tan limpias, si no que se manifiestan en forma de nubes de puntos. Se aprecia como a medida que empieza la congestión, la nube de puntos se vuelve totalmente heterocedástica (la desviación incrementa cuando la variable explicativa se va para la derecha).

Se ha hecho una evaluación del simulador respecto distintas situaciones: primero se ha reproducido un modelo de red caótico que prácticamente permanece congestionado a lo largo de toda la línea, seguidamente se han comprobado los efectos positivos que tiene evaluar una misma línea respecto una reducción en las cantidades de servicios y por último se ha evaluado una misma línea con tiempos entre salidas de autobuses distintos.

Como bien se comenta, el primer ejemplo es una simulación de un modelo de red extremo con muchas llegadas de pasajeros a las paradas lo que provoca una sobrecarga en todo el sistema. Como las llegadas de pasajeros se producen de manera masiva, en todos los servicios queda gente sin poder subir que debe quedar esperando poder ser atendido en los próximos servicios.

Los dos siguientes ejemplos explican lo que sucede cuando se augmenta en cantidad el número de servicios que recorren la línea. El pasar de 100 a 200 autobuses ha hecho que se deje de experimentar congestión en una de sus paradas.

El cuarto ejemplo es una de las simulaciones que más se acercan a las situaciones que se observan en el día a día. Durante las primeras paradas no es usual tener que esperar tiempos muy altos para poder subir a un autobús. Tampoco es común encontrar estos servicios muy llenos al principio de la línea. Las acumulaciones de pasajeros suelen producirse a partir de las siguientes paradas hasta el final de la línea, ya que las paradas intermedias sufren acumulaciones de pasajeros provenientes de paradas anteriores. Además, los trayectos que realizan los pasajeros suelen ser normalmente largos, y un pasajero que comienza su trayectoria en la primera parada, puede finalizar su servicio en la misma parada en la que otro pasajero ha llegado subiendo al servicio tres paradas más adelante, provocando una congestión en la línea debida a la permanencia de los pasajeros hasta el final de esta.

En cuanto al último ejemplo, una intuición es que la capacidad C_i^j (número libre de plazas del servicio j en la parada i) hace que los factores de carga sean más elevados. Anteriormente se ha visto como el factor de carga resulta de dividir la suma total de llegadas A_i^j entre las capacidades C_i^j del servicio j en la parada i. Respecto los resultados obtenidos, una explicación es que estas capacidades son más bajas, observando en consecuencia valores de ρ más elevados.

Estos resultados obtenidos pueden deberse a que el número de pasajeros que han quedado esperando en la cola a poder subir en el siguiente servicio (N_i^j) es elevado respecto el ejemplo anterior. Una posible cuestión que lo explica es que el tiempo de servicio x_i^j es muy grande. La idea es que estos tiempos deberían de ser del orden de 5 minutos o más en algunos casos. Cuando se presenta un headway de 10 minutos, el tiempo de servicio es inferior a ese valor, pero puede ser, como es el caso del quinto ejemplo, que el tiempo de servicio esté sobre los 5 minutos o incluso sea superior en algunos casos. Cuando esto ocurre, los servicios llegan pegados. Esto hace que no se recojan a todos los pasajeros que han llegado, y si las colas son elevadas, se pase a atender solo a gente que ha quedado atrasada. En consecuencia, los que han llegado durante el período pasarán a formar parte del lote de pasajeros que subirán al siguiente autobús.

Respecto las representaciones de las colas de autobuses, también se han evaluado sus longitudes medias y se han observado situaciones muy distintas. Es cierto que en la vida real no es habitual encontrar colas de buses de más de dos o tres unidades de servicios, pero es importante tener este tipo de colas de espera en cuenta si se quiere simular correctamente una red de transporte. En los ejemplos se ha podido evaluar dos tipos de disposiciones distintas: uno de ellos prácticamente no presenta colas de espera de servicios mientras que el otro, a medida que la línea se congestiona, comienza a sufrir un augmento en el número de autobuses que esperan en la cola a poder servir en las paradas. De esta manera, el simulador, a parte de ser un método que representa situaciones reales, es una herramienta perfecta para poner a prueba una línea bajo condiciones muy extremas y poder anotar sus respuestas.

Anteriormente se ha hecho referencia a lo extenso que puede llegar a ser programar con total exactitud un simulador de estas magnitudes. Existen multitud de variables y condiciones que, en proyectos posteriores, deberían tenerse en cuenta en el simulador para conseguir resultados mucho más acurados. También sería interesante en un futuro poder centrarse en la optimización del código con la idea de gestionar este tipo de simulaciones en tiempos reducidos y con sistemas de transporte distintos. Cambiando las parametrizaciones pueden obtenerse resultados además de para colas de autobuses, para colas de metros, trenes, etc.

Por último, y para terminar, se quiere hacer referencia a la aplicación que tiene este proyecto. Se espera que el simulador sea de utilidad en proyectos con modelos diseñados para la asignación de pasajeros a líneas de transporte público. Antes de construir nuevas líneas de transportes se hacen predicciones sobre la absorción de pasajeros que estas pueden contemplar, de esta manera se puede estudiar si es rentable o no introducir una nueva línea en la red. Es por esto que, el simulador puede ser de gran ayuda a la hora de estimar las demoras de cada línea bajo ciertas situaciones de congestión y evaluar si se establecen las condiciones óptimas para implementarlas.

Capítulo 10

Bibliografía

Barceló, Jaume. Montero, Lídia. 2016. *«Apuntes de la asignatura Simulación. Master d'Estadística i Investigació Operativa UB-UPC (MESIO)»*. .

Bhat, U. Narayan. 1933. *«An Introduction to Queueing Theory [Recurs electrònic] : Modeling and Analysis in Applications»*. https://link.springer.com/book/10.1007%2F978-0-8176-4725-4

Burgoine, Paul. 2013. «The Testing of Random Number Generators». https://www1.maths.leeds.ac.uk/ voss/projects/2012-RNG/Burgoine.pdf

Codina S., Esteve. Montero M., Lídia. 2020. *«Teoria de cues i simulació. Grau d'estadística»*. http://www-eio.upc.edu/teaching/TCiS/

Donald Gross. 2008. *«Fundamentals of queueing theory»*.

Lipsky, Lester. 2009. *«Queueing Theory [Recurs electrònic]: A Linear Algebraic Approach»*. https://link.springer.com/book/10.1007%2F978-0-387-49706-8

Meng, Xiannong. 2002. *«Runs Tests»*. https://www.eg.bucknell.edu/ xmeng/Course/CS6337/Note/master/node44.html

Powell, W.B. 1986 *«Approximate, Closed form moment formulas for bulk arrival, bulk service queues.»* Transportation Science, Vol 20, No 1.

Prabhanhan N. Tattar, Suresh Ramaiah, B. G. Manjunath. 2016. *«A Course in Statistics with R»*. https://books.google.es/books?id=2BmPCQAAQBAJ&pg

R Core Team and contributors worlwide. 2020. *The R Stats Package*. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/

Robert G. Brown. 2004. *«Dieharder: A Random Number Test Suite»*. http://webhome.phy.duke.edu/ rgb/General/dieharder.php/

Tyszer J. 1999. *«Event scheduling. In: Object-Oriented Computer Simulation of Discrete-*

Event Systems. The Kluwer International Series on Discrete Event Dynamic Systems, vol 10. Springer, Boston, MA.» https://doi.org/10.1007/978-1-4615-5033-4_2

Weisstein EW. From MathWorld-A Wolfran Web Resource. 1999-2020. *«Statistical Distribution».* https://mathworld.wolfram.com/topics/StatisticalDistributions.html

Wolfgang Karl Härdle, Ostap Okhrin, Yarema Okhrin. 2017. *«Basic Elements of Computational Statistics».* https://books.google.es/books?id=JOI3DwAAQBAJ&pg

Wuyi Yue, Yataka Takahashi, Hideaki Takagi. 2009. *«Advances in Queueing Theory and Network Applications [Recurs electrònic]».* https://link.springer.com/book/10.1007%2F978-0-387-09703-9

Apéndice A

Anexo I: Valores para las magnitudes mostradas en la figura 7.1

Cuadro A.1: Flujo de datos primer ejemplo

	v_x	v_a	v_y	v_e	d
Parada 1	0	10000	0	10000	200
Parada 2	8000	7000	2000	15000	200
Parada 3	15000	2500	0	17500	200
Parada 4	17500	2000	0	19500	200
Parada 5	17500	2000	2000	19500	200
Parada 6	17500	5000	2000	23500	200
Parada 7	0	0	23500	0	0

Cuadro A.2: Flujo de datos segundo y tercer ejemplo

	v_x	v_a	v_y	v_e	d
Parada 1	0	2000	0	2000	200
Parada 2	1000	3000	1000	4000	200
Parada 3	3000	5500	1000	8000	200
Parada 4	7000	3000	1000	10000	200
Parada 5	8000	1000	2000	9000	200
Parada 6	0	10000	9000	9000	200
Parada 7	0	0	10000	10000	0

Cuadro A.3: Flujo de datos cuarto y quinto ejemplo

	v_x	v_a	v_y	v_e	d
Parada 1	0	4500	0	4500	200
Parada 2	2500	2000	2000	4500	200
Parada 3	2500	2000	2000	4500	200
Parada 4	1250	3500	3250	4750	200
Parada 5	1250	3500	3250	4750	200
Parada 6	1000	3750	3750	4750	200
Parada 7	0	0	4750	0	0

Apéndice B

Anexo II: Código en R - Runs Test

```
### Numero total de runs ###
funcion.runs<-function(v){
    # Se crea un vector vacio para los runs
    runs <- c()
    p <- c()
    # i recorre la longitud del vector menos la ultima posicion
    for (i in 1:length(v)-1){
        # Si el valor actual del vector es mas grande o igual al
            siguiente valor, ponemos -1, si no ponemos 1
        runs[i] <- ifelse( v[i] >= v[i+1], -1, 1)
    }
    # j recorre el vector de runs a partir de la segunda posicion
    for (j in 2:length(runs)){
        # Si el valor actual de runs no es igual al anterior, p vale
            1, si no vale 0
        p[j] <- ifelse( runs[j] != runs[j-1], 1, 0)
    }
    # Se quitan los NA y se convierten en 1
    p[is.na(p)] <- 1
    # Se devuelve la suma de p
    return(sum(p))
}


# Distribucion del numero de runs aproximados por una normal
funcion.runs.test<-function(v,a){
    # Se guarda el numero total de runs obtenido con la funcion
       anterior
    r <- funcion.runs(v)
    # Longitud del vector de entrada
    n <- length(v)
    # Calculo de la media de los runs
    mu_a <- (2*n - 1)/3
    # Calculo de la variancia de los runs
```

```
    sigma_a <- (16*n - 29)/90
    # Test de independencia de los numeros Z0
    Z <- (r - (mu_a))/sqrt(sigma_a)

    # Si el test estadistico es superior a la normal con alpha=.05
    if( (Z>qnorm(a/2)) && (Z<qnorm(a/2,lower.tail=F)) ){
        # Se acepta la H0
        return(list(r,"Aceptamos H0: generadas aleatoriamente", Z))
        # En caso contrario, se rechaza la H0
    }else{
        return(list(r,"Rechazamos H0: no generadas aleatoriamente", Z
            ))
    }
}
```

Apéndice C

Anexo III: Código en R - Test Chi-Cuadrada

```
dd <- read.csv("muestras2.csv", sep=";",header=TRUE, as.is=T)
n <- 5000
lambda=4
############
# Uniforme #
############

sample <- dd[,"Uniforme"]

# Estadisticas basicas
summary(sample)
mm <- mean(sample)
parametro <- 0.5    # Media uniforme: (a+b/2 = 1+0/2 = 1/2)

# Representacion grafica
par(mfrow=c(1,2))
hist(sample,breaks=100,col="lemonchiffon",main="Histograma para la
    muestra de valores generada con la Uniforme ")
hist(sample,freq=F, col="lightblue")
curve(dunif(x,min=0, max=1), add=T,col=2)

# Secuencia
sequence<-seq(0,1,by=0.04)
graphics.off()

perdist <- sequence
hist(sample,freq=FALSE,breaks=perdist, col="Histograma para la
    muestra de valores generada con la Uniforme ")
curve(dunif(x,min=0, max=1),col=2,add=T)

# Chi squared test
```

```
dsample<-cut(sample,breaks=perdist,include.lowest = T)
table(dsample)
summary(dsample)
iobs<-as.vector(table(dsample))
pexp<-as.vector(rep(1/length(table(dsample)),length(table(dsample))))
iexp<-n*pexp
X2<-sum(((iobs-iexp)^2)/iexp);X2
chisq.test( iobs,p=pexp )



#########################
#  Muestra Exponencial #
#########################
sample <- dd[,"Exponencial"]

# Estadisticas basicas
summary(sample)
mm <- mean(sample)
parametro <- 1/mm     # La media de una exponencial es igual a 1/
    lambda

# Representacion grafica
par(mfrow=c(1,2))
hist(sample,breaks=100,col="yellow",main="Histograma para la muestra
    de valores generada con la Exponencial ")
hist(sample,freq=F, col="lightblue")
curve(dexp(x,rate=lambda), add=T,col=2)
plot(density(sample), col = 'red')  ## Asi si sale
lines(density(rexp(n,lambda)), col= 'blue')

# Secuencia
sequence<-seq(0,1,by=0.04)
graphics.off()

#
perdist<-qexp(sequence,rate=lambda)
perdist[length(perdist)]<-max(sample)
hist(sample,freq=FALSE,breaks=perdist, col="green",main="Histograma
    para la muestra de valores generada con la Exponencial ") #
    Exemple de histograma per percentils
curve(dexp(x,rate=lambda),col=2,add=T)


# Chi squared test
dsample<-cut(sample,breaks=perdist,include.lowest = T)
table(dsample)
summary(dsample)
```

```
iobs<-as.vector(table(dsample))
pexp<-as.vector(rep(1/(length(table(dsample))),length(table(dsample))
    ))
iexp<-n*pexp
X2<-sum(((iobs-iexp)^2)/iexp);X2
pchisq(X2,23, lower.tail=FALSE)

#####################
#   Muestra Poisson #
#####################
sample <- dd[,"Poisson"]

# Estadisticas basicas
summary(sample)
mm <- mean(sample)
parametro <- mm;parametro     # La media de una poisson es igual a
    lambda por t (en este caso 4*3=12)
parametro <- parametro

# Representacion grafica
par(mfrow=c(1,2))
hist(sample,ylim =c(0,1000),breaks=100,col="yellow",main="Histograma
    para la muestra de valores generada con la Poisson")

b<-seq(-0.5,max(sample)+0.5)
hist(sample,freq=F , breaks=b,ylim=c(0,0.01), xlim = c(1000,2000),col
    ='lemonchiffon ', main="Histograma para la muestra de valores
    generada con la Poisson")
points(b+0.5,dpois(b+0.5,parametro),col=2,pch=19)
points(b+0.5,dpois(b+0.5,parametro),col=2,type='h',lwd=2)

# Secuencia
sequence<-seq(0,1,by=0.04)
graphics.off()

perdist <-qpois(sequence,parametro, lower.tail=TRUE, log.p=FALSE);
    perdist
perdist[length(perdist)]<-max(sample)
perdist <- unique(perdist)
hist(sample,freq=FALSE, ylim= c(0,0.01), reaks=perdist, col="
    lemonchiffon",main="Histograma muestra Poisson") # Exemple de
    histograma per percentils
points(b+0.5,dpois(b+0.5,parametro),col=2,pch=19, cex=0.2)

hist(sample,freq=F,breaks=perdist,ylim= c(0,0.01),xlim=c(1500,1800),
    col='lemonchiffon', main= "Histograma muestra Poisson")
points(b+0.5,dpois(b+0.5,parametro),col=2,pch=19)
points(b+0.5,dpois(b+0.5,parametro),col=2,type='h',lwd=2)
```

```
# Chi squared test
dsample<-cut(sample,breaks=unique(perdist),include.lowest = F)
table(dsample)
summary(dsample)
iobs<-as.vector(table(dsample))             # Obs in the groups
    defined by nb intervals-percentiles
pexp<-as.vector(rep(1/length(iobs),length(iobs)))               #
    Expected probability for each groups: sample size/nb intervals
iexp<-n*pexp                                # VALOR ESPERAT D'
    OBSERVACIONS
X2<-sum(((iobs-iexp)^2)/iexp);X2            # CACLCUL DE L'ESTADISTIC
    DE X2
chisq.test(iobs,p=pexp )                    # EXEMPLE QUE SI FUNCONA
pchisq(X2, length(table(dsample))-2, lower.tail = FALSE)  # 14-1-1
    grados de libertad
barplot(table(dsample), col="green")


####################
# Muestra K-Erlang #
####################
sample <- dd[,"K.Erlang"]

# Estadisticas basicas
summary(sample)
mm <- mean(sample);mm
k <- round((mm/sqrt(var(sample)))^2,0);k
(alpha <- round(sqrt(k/var(sample)),2)  )
V = k/alpha^2 #variancia aproximada
(t_m = 1/alpha) #tiempo medio de cada etapa
(c_d <- 1/sqrt(k))  # Coef_desviaciom

# Representacion grafica
par(mfrow=c(1,2))
hist(sample,breaks=100,col="yellow",main="Histograma muestra K-erlang
    ")
hist(sample,freq=F,breaks=25, col="lightblue")
curve(dgamma(x,rate=alpha,shape=k), add=T,col=2)


# Secuencia

sequence<-seq(0,1,by=0.04)
sequence

graphics.off()
```

```
# H0: Exponencial lambda = 1/16
perdist<-qgamma(sequence,shape=k, rate=alpha)
perdist[length(perdist)]<-max(sample)
hist(sample,freq=FALSE,breaks=perdist, col="green",main="Histograma
    muestra K-Erlang")
curve(dgamma(x,shape=k, rate=alpha),col=2,add=T)

# Chi squared test
dsample<-cut(sample,breaks=perdist,include.lowest = T)
table(dsample)
summary(dsample)
iobs<-as.vector(table(dsample))
pexp<-as.vector(rep(1/length(table(dsample)),length(table(dsample))))
iexp<-n*pexp
X2<-sum(((iobs-iexp)^2)/iexp);X2
pchisq(X2,22, lower.tail=)

# P-valor significativo...
chisq.test(iobs,p=pexp )
pchisq(X2, length(table(dsample))-3, lower.tail = FALSE)  # 14-2-1
    grados de libertad


#############
# Binomial #
#############
sample <- dd[,"Binomial"]
summary(sample)
nn <- max(sample);nn
mm <- mean(sample);mm
(p <- mm/nn);p


# Estadisticas basicas

t <- table(sample);t

# Multiples grafic
hist(sample,freq=F,breaks=nn,ylim=c(0,0.4),col='lemonchiffon', main="
    Histograma muestra Binomial")


f = as.vector(t)
x= names(table(sample))
r = f/5000
pdf = dbinom(0:5,5,p)
E = 5000*pdf
```

```
df <- data.frame(x=x,f=f, r=r, pdf=pdf, E=E)
df

Q = sum((f-E)^2/E);Q
qchisq(.95,6-2)
1-pchisq(Q,4)
```

Apéndice D

Anexo IV: Código en Python - Generación Variables Aleatorias

```
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
import random as rd
import numpy.random as ra
import math
import os
os.getcwd()
os.chdir('C:\\Universidad\\TFG')

n=5000
lambd = 4 #4 pasajeros por minuto

# Uniforme
def uniforme(mini,maxi,n):
    U = ra.uniform(mini,maxi,n)
    U = U.tolist()
    return(U)

unifo = uniforme(0,1,n)
unifo = [round(num, 5) for num in unifo]

# Exponencial
def exponencial(n,param):
    r = ra.uniform(0,1,n)
    x = (-1/param)*np.log(r)
    return(x)

expo = exponencial(n,lambd)
expo = expo.tolist()
expo = [round(num, 5) for num in expo]
```

```
# Poisson
def poisson(n,lambd,T):
    X = list(range(n))
    s = 0
    for k in range(n):
        i=0
        r = ra.uniform()
        t = -(1/lambd)*np.log(r)
        while t<T:
            i += 1
            r = ra.uniform()
            t += -(1/lambd)*np.log(r)
            s += 1
        X[k]= i
    return(X)

poiss = poisson(5000,4,1)

# Erlang
def kerlang(n,alpha,k):
    X = list(range(n))
    for i in range(n):
        r = ra.uniform(1,0,k)
        Y = -(1/alpha)*np.log(np.prod(r))
        X[i] = Y
    return(X)

erln = kerlang(5000,0.5,3)
erln = [round(num, 5) for num in erln]

# Binomial
def binomial (n,p):
    # Inicializacion variables
    x=0
    # Para el rango de n (numero de variables a generar)
    for i in range(n):
        # Se genera una v.a uniforme en el intervalo [0,1]
        r = ra.uniform(1,0,1)
        # Si la v.a. uniforme es inferior a la constante de
          probabilidad
        if r < p:
            # x valdra 1
            x = x + 1
            # En caso contrario, x valdra 0
    # Devuelve valor de x
    return(x)
```

```
X =list(range(5000))
for i in range(5000):
    binom = binomial(1,0.5)
    X[i] =binom


binom = X

data = {'Uniforme':unifo,'Exponencial':expo,
        'Poisson':poiss, 'K-Erlang':erln, 'Binomial':binom}
df = pd.DataFrame(data)
df.to_csv('muestras2.csv', sep=';' , float_format='%.5f')
```

Apéndice E

Anexo V: Código en Python - Simulador

```
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
import random as rd
from tabulate import tabulate
import numpy.random as ra
import math
import os
import scipy.stats as ss
import matplotlib.pyplot as plt
from datetime import date
from datetime import datetime
import matplotlib as mpl

### Tiempo de inicio de la ejecucion ###
inicial = datetime.now()

### Funcion que genera variables aleatorias poissonianas ###
def poisson(lambd,T):
    # Inicializacion variables
    ww = 0
    i=0
    zeta = 0
    # Se genera una variable aleatoria uniforme en el intervalo [0,1]
    r = ra.uniform(1,0,1)
    # Con la v.a. uniforme se genera una v.a exponencial
    t = -(1/lambd)*np.log(r)

    # Mientras el valor de t generado sea inferior a T (intervalo de
       tiempo)
    while t<T:
```

```
            # Contador para ir incrementando el bucle del while
            i += 1
            # Variable que contiene las sumas de los tiempos que esperan
                los pasajeros
            ww += (T-t)
            # Variable que contiene el tiempo que espera el ultimo
                pasajero que llega a la parada
            zeta = T-t
            # Se genera una v.a uniforme
            r = ra.uniform()
            # Se genera un nuevo valor de t exponencial con la v.a.
                uniforme r
            t += -(1/lambd)*np.log(r)
        # El numero de llegadas de clientes a la parada sera el valor del
            contador
        X = i
        # Devuelve el numero de llegadas, la suma de los tiempos de
            espera y el tiempo de espera del ultimo cliente
        return(X, ww, zeta)


### Funcion que genera variables aleatorias binomiales ###
def binomial (n,p):
    # Inicializacion variables
    x=0
    # Para el rango de n (numero de variables a generar)
    for i in range(n):
        # Se genera una v.a uniforme en el intervalo [0,1]
        r = ra.uniform(1,0,1)
        # Si la v.a. uniforme es inferior a la constante de
            probabilidad
        if r < p:
            # x valdra 1
            x = x + 1
            # En caso contrario, x valdra 0
    # Devuelve valor de x
    return(x)

### Directorio donde se encuentran los archivos ###
os.getcwd()
os.chdir('C:\\Universidad\\TFG')

### Parametros de entrada separados por lineas ###
numero = 9
archivo = 'avflows' + str(numero) + '.dat'
archivo2 = 'avflows' + str(numero) + '\\'
archivo_buses = 'buses'+str(numero)+'\\'
```

```
#Para pruebas
archivo2 = 'pruebas2' +'\\'
archivo_buses = 'pruebas2'+'\\'

# Importa el archivo a file
file = open(archivo, 'r').read()
# Divide el archivo por lineas
input_param = file.split('***')
# Crear lista vacia
ip = []

for ne in range(len(input_param)):
    # Selecciona los datos de la linea con el indice ne
    sep = input_param[ne]
    # Separa todos los datos en una lista
    sep_lista = sep.split()
    # Guarda la linea como una lista dentro de la lista ip
    ip.append(sep_lista)

# Elimina lo ultimo que haya en ip (espacio vacio)
ip.pop()

### Parametros fijos para servicio en autobus ###
# Velocidad del bus (50km/h)
#velocidad = 50
# Recorrido que realiza el bus en una hora yend a 50km/h
#recorrido = (velocidad*1000)/60
# Distancia entre paradas (en metros)
#distancia = 400
# Tiempo de recorrido entre la parada i y la i+1 para el bus j
#dis = (distancia/recorrido)

# Tiempo de subida/bajada de un pasajeros (5 i 2.5 segundos pasados a
    minutos)
kp, kd = 5/60,2.5/60
# Tiempo de encochado y desencochado en la parada i
e_mas, e_menos  = 6/60 ,15/60
# Tiempo de abrir y cerrar puertas en la parada i
pi_mas, pi_menos = 2.5/60, 2.5/60


#################
### Simulador ###
#################
# Inicializacion de algunas listas
demoras = []
Wi_barra=[]
rhom_barra =[]
```

```
vector_paradas = []
WIB = []
LIB =[]
WIB2 = []
LIB2 = []
rhoB = []
WIB_NORM = []

nombres = ["Semilla","Simulacion","Linea", "Bus", "Parada", "Ei-1", "
    Xi", "Ei", "Pi", "Di", "Ai", "Ni", "t", "theta", "tS", "phi","x",
    "tSBUS","Parametro"]
Matriz = [nombres]
lol = lambda lst, sz: [lst[il:il+sz] for il in range(0, len(lst), sz)
    ]

nombres_p = ["Semilla","Simulacion","Linea", "Bus", "Parada", "Ai", "
    phi", "Parametro"]
Matriz_p = [nombres_p]


# Numero de simulaciones que realizara el simulador
simulaciones=10
# Numero de factores de carga distintos que hara por simulacion
semillas = 10
# Contador para cambiar de semilla
contador = 0

### Inicio del simulador ###
# El primer bucle recorre las semillas
for bucle_rhos in range(semillas):

    # Comando que altera el generador de variables aleatorias
    np.random.seed(478659 + contador)

    # El segundo bucle recorre las simulaciones
    for s in range(simulaciones):

        # El tercer bucle recorre las lineas de la red de transporte
        for l in range(len(ip)):

            ########################################
            ### Informacion relativa a la linea i ###
            ########################################

            # Vector que guarda los tres primeros valores de la linea
                (numero de buses, capacidad del bus y headway)
            obs = ip[l][:3]
            # Vector que guarda los flujos totales de llegadas y
```

```
    salidas de pasajeros
info_p = ip[l][3:]
# Diferencia los datos de cada parada en sublistas
parad = lol(info_p,5)
# Numero de paradas
I = len(parad)
# Numero de buses (comun a las paradas)
J = int(obs[0])
# Capacidad en numero de pasajeros de cada bus
c = int(obs[1])
# Tiempo entre dos autobuses consecutivos
headway = float(obs[2])
# Incremento de tiempo por arranque/frenada en salir/
    llegar de la parada
a,f = 3/60, 3.5/60
a,  f, = [a] * I, [f] * I
a[I-1] , f[0] = 0,0
# Nombre indices columnas
buses = []
list1 = [ bu for bu in range(0,J+1,1)]
for b in list1:
    bus = "bus"+ str(b)
    buses += [bus]

# Nombre indices filas
paradas = []
list2 = [ pa for pa in range(0,I+1,1)]
for p in list2:
    parada = "parada"+ str(p)
    paradas += [parada]

#################
### Variables ###
#################

## Data frames vacios
# Numero de pasajeros en el servicio j despues de salir
    de la parada i
E = pd.DataFrame(index=paradas,columns=buses)
E.iloc[:,0], E.iloc[0,:]  = 0.0, 0.0

# Numero de pasajeros que bajan en el servicio j parada i
D = pd.DataFrame(index=paradas,columns=buses)
D.iloc[:,0], D.iloc[0,:], D.iloc[1,:]  = 0.0, 0.0, 0.0

# Numero de pasajeros presente en el bus j durante la
    parada del servicio j a la parada i
X = pd.DataFrame(index=paradas,columns=buses)
```

```
X.iloc[:,0:1], X.iloc[0:1,:] , X.iloc[1,:]  = 0.0, 0.0,
    0.0

# Capacidad en numero de pasajeros del servicio j en la
    parada i.
C = pd.DataFrame(index=paradas,columns=buses)
C.iloc[:,0], C.iloc[0,:]  = 0.0, 0.0

# Numero de pasajeros en la parada i en salir el bus j
N = pd.DataFrame(index=paradas,columns=buses)
N.iloc[:,0], N.iloc[0,:], N.iloc[:,1] , N.iloc[1,:] =
    0.0, 0.0, 0.0, 0.0

# Numero de pasajeros minimo que atendera el bus j a la
    parada i
R = pd.DataFrame(index=paradas,columns=buses)

# Instante de salida del servicio j de la parada i
theta = pd.DataFrame(index=paradas,columns=buses)
theta.iloc[:] =0

# Se calculan los tiempos de inicio de todos los buses
    desde la primera parada
valores = []
suma = 0
for k in range(J):
    valores.append(suma)
    suma = suma + headway

# Instante de llegada del servicio j a la parada i (la
    primera parada con valores)
t = pd.DataFrame(index=paradas,columns=buses)
t.iloc[:], t.iloc[1,1:]  = 0, valores

# Instante de inicio del servicio del bus j a la parada i
tS = pd.DataFrame(index=paradas,columns=buses)
tS.iloc[:,0], tS.iloc[0,:]  = 0, 0

# Numero de subidas de pasajeros a la parada i en salir
    el bus j-1
P = pd.DataFrame(index=paradas,columns=buses)
P.iloc[:,0], P.iloc[0,:]= 0.0, 0.0

# Tiempo necesario para hacer subir la cantidad de
    pasajeros R_i^j
y = pd.DataFrame(index=paradas,columns=buses)
y.iloc[:,0], y.iloc[0,:], y.iloc[1,:]  = 0.0, 0.0, 0.0
```

```
# Tiempo de recorrido entre la parada i y la i+1
d = pd.DataFrame(index=paradas,columns=buses)
d.iloc[1,:] = 0.0

# Numero de pasajeros que llegan entre el periodo theta[i
    ,j-1] y theta[i,j]
A = pd.DataFrame(index=paradas,columns=buses)

# Tiempo de servicio del bus en funcion de las
    circumstancias
tSBUS = pd.DataFrame(index=paradas,columns=buses)

# Tiempo de servicio de los pasajeros del servicio j a la
     parada i.
x = pd.DataFrame(index=paradas,columns=buses)
x.iloc[:,0:1], x.iloc[0:1,:] , x.iloc[1,:]  = 0.0, 0.0,
    0.0

# Intervalo de tiempo que entra como variaable input en
    la funcion poisson
phi = pd.DataFrame(index=paradas,columns=buses)
phi.iloc[:,0], phi.iloc[0,:]  = 0, 0

# Tiempo entre la subida del ultimo pasajero del bus j y
    la sortida del bus j
z = pd.DataFrame(index=paradas,columns=buses)
z.iloc[:,0], z.iloc[0,:], z.iloc[1,1]  = 0, 0, 0

# Demoras de tiempo (tiempo que esperan los pasajeros en
    las paradas)
W = pd.DataFrame(index=paradas,columns=buses)
W.iloc[:,0], W.iloc[0,:]  = 0, 0

# Demoras para el estadistico de la cola de buses
w_ij_B = pd.DataFrame(index=paradas,columns=buses)
w_ij_B.iloc[:,0], w_ij_B.iloc[0,:]  = 0, 0

# Tiempos de espera para los buses
W_i_B = pd.DataFrame(index=paradas, columns = range(1))
W_i_B.iloc[:,0] = 0

# El cuarto bucle recorre los buses
for j in range(J):

    theta.iloc[0,j+1] = (j)*headway

    # El quinto bucle recorre las paradas
    for i in range(I):
```

```
# Recoje los parametros de flujo de la parada
    correspondiente
parada_sim = parad[i]

# Parametros de la lambda de la ultima simulacion
param_max = (0.95*float(parada_sim[1]))/((J+1)*
    headway)
# Parametro de lamdda de las simulaciones
    intermedias
param_min = param_max/simulaciones

# Velocidad del bus (50km/h)
velocidad = 50
# Recorrido que realiza el bus en una hora yendo
    a 50km/h
recorrido = (velocidad*1000)/60
# Distancia entre paradas (en metros)
distancia = float(parada_sim[4])
# Tiempo de recorrido entre la parada i y la i+1
    para el bus j
dis = (distancia/recorrido)
d.iloc[i+1,j+1] = dis
d.iloc[0,:] = 0

## S01

# Si no es la ultima parada
if (i+1)<I:
    # Se calcula el parametro de alpha
    alpha = (float(parada_sim [2])/(float(
        parada_sim [3])-1))
    # Se calcula el numero de personas que
        bajaran en la parada i del bus j
    D.iloc[i+1,j+1] = binomial(int(E.iloc[i,j+1])
        ,alpha)
# Si es la ultima parada
else:
    # Bajan todos los pasajeros que hay en el bus
    D.iloc[i+1,j+1] = E.iloc[i,j+1]

## S02

# Tiempos de abrir y cerrar puertas del servicio
    j
p_mas = pi_mas
p_menos = pi_menos
```

```
## S03

# Si es la ultima parada
if i+1 == I:
    # No pueden quedar pasajeros en el bus cuando
        el servicio j se para
    X.iloc[i+1,j+1] = 0.0    #
# Si no es la ultima parada
else:
    # Se calcula el numero de personas que quedan
        en el bus cuando hace su servicio
    X.iloc[i+1,j+1] = E.iloc[i,j+1] - D.iloc[i+1,
        j+1]

## S04

# Capacidad del bus sera la capacidad real menos
    los pasajeros que han quedado en el bus
C.iloc[i+1,j+1] = c - X.iloc[i+1,j+1]

## S05

# Numero de pasajeros minimo que atendera el bus
R.iloc[i+1,j+1] = min(N.iloc[i+1,j+1],C.iloc[i+1,
    j+1])

## S06

# Instante de llegada del servicio j a la parada
    i
t.iloc[i+1,j+1] = theta.iloc[i,j+1] + d.iloc[i,j
    +1] + a[i] + f[i] #+ t.iloc[i+1,j+1]   #
    Instant de llegada del bus j a la parada i

## S07

# Instante de inicio del servicio del bus j a la
    parada i
tS.iloc[i+1,j+1] = max(theta.iloc[i+1,j], t.iloc[
    i+1,j+1])        # Instante de inicio del
    servicio del bus j a la parada j

## S08

# Si el numero minimo de pasajeros que atiende el
     bus es superior a cero
if R.iloc[i+1,j+1]  > 0.0:
```

```
            y.iloc[i+1,j+1] = kp*R.iloc[i+1,j+1]       #
                Tiempo necesario para poder hacer subir a
                las personas Rij

        ## S09

        # En caso contrario
        elif R.iloc[i+1,j+1] == 0.0:
            y.iloc[i+1,j+1] = 0.0


        ## S10

        # Calculo del intervalo de tiempo (parametro T de
            poisson)
        phi.iloc[i+1,j+1] = tS.iloc[i+1,j+1]- theta.iloc[
           i+1,j] + y.iloc[i+1,j+1] + z.iloc[i+1,j]

        ## S11

        # Si es la ultima parada
        if i+1 == I:

            # No se producen subidas de pasajeros
            A.iloc[i+1,j+1] = 0.0
            # Y el parametro lambda es 0
            param = 0

        # Si no es la ultima parada
        else:
            # Se calcula lambda
            param = param_min*(s+1)

        # Si el parametro es 0
        if param == 0:
            # No hay subidas de pasajeros
            A.iloc[i+1,j+1] = 0
        # En caso cotrario
        else:
            # Vector que contiene el numero de llegads,
                los tiempos totales de espera y z
            llegadas =  poisson(param,phi.iloc[i+1,j+1] )
                 # Numero de personas que llegan a la
                parada
            # Numero de llegadas
            A.iloc[i+1,j+1] = llegadas[0]

        # Si no llegan pasajeros
```

```
if A.iloc[i+1,j+1]==0:
    # NO hay tiempos de espera
    w = 0
    # No existe tiempo entre la subida del ultimo
        pasajero y la salida del bus
    z.iloc[i+1,j+1] = 0
# Si llegan pasajeros
else:
    # w contiene los tiempos totales de espera
    w = llegadas[1]
    # tiempo entre la subida del ultimo pasajero
       del bus j y la salida de ese bus
    z.iloc[i+1,j+1] = llegadas[2]

## S12

# Si es la ultima parada
if i+1 == I:
    # No se producen subidas de pasajeros al bus
    P.iloc[i+1,j+1] = 0.0

# Si es la ultima parada
else:
    # El numero de subidas es el minimo entre los
        que llegan mas los que quedaron colgados
       y la capacidad que queda en el bus
    P.iloc[i+1,j+1] = min( A.iloc[i+1,j+1] + N.
       iloc[i+1,j+1], C.iloc[i+1,j+1]  ) # Numero
        de personas que suben al bus

## S13
# Numero de pasajeros al servicio k despues de la
    salida de la parada i
E.iloc[i+1,j+1] = P.iloc[i+1,j+1] + X.iloc[i+1,j
   +1]

## S14

# Buses a partir del segundo servicio
if j+2 <= J:
    N.iloc[i+1,j+2] = A.iloc[i+1,j+1] + N.iloc[i
       +1,j+1] - P.iloc[i+1,j+1]
                          # Si es el primer bus
if j+1 == 1:
    N.iloc[i+1,j+1] = 0 #Numero de personas en la
        parada i en salir el bus j+1

## S15
```

```
# Tiempo de servicio de los pasajeros del
    servicio j a la parada i
x.iloc[i+1,j+1] = max (kd*D.iloc[i+1,j+1],kp*P.
    iloc[i+1,j+1])


## S16

# Si el tiempo de servicio de los pasajeros es 0
    (no hay pasajeros)
if x.iloc[i+1,j+1] == 0:
    # Los tiempos de abrir y cerrar puertas no
        existen
    p_mas = 0.0
    p_menos = 0.0

## S17

# Tiempo total de servicio del bus j en la parada
    i
tSBUS.iloc[i+1,j+1] = e_mas + p_mas + x.iloc[i+1,
    j+1] + p_menos + e_menos

## S18

# Calculo del tiempo de salida del bus actual
theta.iloc[i+1,j+1] = tS.iloc[i+1,j+1] + tSBUS.
    iloc[i+1,j+1]


## S19 Calculo estadisticos

# Si no hay llegadas de pasajeros
if A.iloc[i+1,j+1]==0:
    # Las demoras se calculan sin la suma de los
        tiempos de espera (w)
    W.iloc[i+1,j+1] = N.iloc[i+1,j+1]*phi.iloc[i
        +1,j+1]
# Si hay llegadas de pasajeros
else:
    # A las semoras se les suma el tiempo total
        de espera de los pasajeros
    W.iloc[i+1,j+1] = w + N.iloc[i+1,j+1]*phi.
        iloc[i+1,j+1]

# Lista que guarda los valores de cada bus en
    cada parada
```

```
            lista = [bucle_rhos+1, s+1, l+1, j+1, i+1, E.iloc
                [i,j+1], X.iloc[i+1,j+1], E.iloc[i+1,j+1], P.
                iloc[i+1,j+1], D.iloc[i+1,j+1], A.iloc[i+1,j
                +1], N.iloc[i+1,j+1], t.iloc[i+1,j+1], theta.
                iloc[i+1,j+1], tS.iloc[i+1,j+1],phi.iloc[i+1,j
                +1], x.iloc[i+1,j+1], tSBUS.iloc[i+1,j+1],
                param]
            # Matrix que guarda los valores de lista por bus
                y parada
            Matriz.append(lista)
            # Listas para las pruebas
            lista_p = [bucle_rhos+1, s+1, l+1, j+1, i+1, A.
                iloc[i+1,j+1], phi.iloc[i+1,j+1], param ]
            Matriz_p.append(lista_p)


            w_ij_B.iloc[i+1,j+1] = theta.iloc[i+1,j+1]- t.
                iloc[i+1,j+1]
            W_i_B.iloc[i+1,0]   += w_ij_B.iloc[i+1,j+1]



# Lista que guarda el numero de paradas que hay por linea
vector_paradas.append(i+1)


## Estadisticos pasajeros

# Esperanza de phi
esp = np.mean(phi.iloc[1:,1:],axis=1)
# Desviacion tipida de phi
desv = np.std(phi.iloc[1:,1:],axis=1)
# Demora sin congestion
W0 = 0.5*esp*(1+(desv/esp)**2)
W0[len(W0)-1]=0
demoras.append(["sim"+str(s+1),["linea"+ str(l+1),W0]])

Wi = np.sum(W.iloc[1:,1:],axis=1)/np.sum(A.iloc[1:,1:],
    axis=1)
Wi_barra.append(["sim"+str(s+1), ["linea"+ str(l+1),Wi]])


rhom = np.sum(A.iloc[1:,1:],axis=1)/np.sum(C.iloc[1:,1:],
    axis=1)
rhom_barra.append(["sim"+str(s+1),["linea"+str(l+1),rhom
    ]])

## Estadisticos buses
```

```
                W_I_B = np.sum(W_i_B.iloc[1:,:],axis=1)
                WIB.append(["sim"+str(s+1),["linea"+str(l+1),W_I_B/J]])
                #WIB2.append(["sim"+str(s+1),["linea"+str(l+1),W_I_B]])

                L_I_B = np.sum(W_i_B.iloc[1:,:], axis=1)
                LIB.append(["sim"+str(s+1),["linea"+str(l+1),L_I_B/(t.
                    iloc[1:,J]-t.iloc[1:,1])]])
                LIB2.append(["sim"+str(s+1),["linea"+str(l+1),L_I_B]])

                # Factor de carga de la cola de buses
                rho_b = np.sum(x.iloc[1:,1:],axis=1)/theta.iloc[1:,J]
                rhoB.append(["sim"+str(s+1),["linea"+str(l+1),rho_b]])

                # Demora normalizada
                WIBNORM = (W_I_B)/np.sum(x.iloc[1:,1:],axis=1)
                WIB_NORM.append(["sim"+str(s+1),["linea"+str(l+1),WIBNORM
                    ]])

        contador += 1500

vector_paradas=vector_paradas[0:(l+1)]
num_lin = len(vector_paradas)



#print(tabulate(Matriz,headers='firstrow'))
#dd = pd.DataFrame(np.array(Matriz))
#dd.columns = nombres
#df = dd.drop([0],axis=0)
#df.to_csv('df.csv', sep=';')


#dd_p = pd.DataFrame(np.array(Matriz_p))
#dd_p.columns = nombres_p
#df_p = dd_p.drop([0],axis=0)
#df_p.to_csv('df_p.csv', sep=';')

# Graficos para las demoras de pasajeros
for lin in range(len(vector_paradas)):
    for par in range(vector_paradas[lin]-1):
        ci= 0
        for sim in range(simulaciones):
            for se in range(semillas):
                W0_G = demoras[sim*num_lin+se*simulaciones*(num_lin)+
                    lin][1][1][par]
                WI_G = Wi_barra[sim*num_lin+se*simulaciones*(num_lin)
                    +lin][1][1][par]
                rho_G = rhom_barra[sim*num_lin+se*simulaciones*(
```

```
                    num_lin)+lin][1][1][par]
                nombre_linea = "Linea"+str(lin+1)+" "
                nombre_parada = "Parada"+str(par+1)
                titulo = str(nombre_linea)+str(nombre_parada)
                WI_W0 = WI_G/W0_G
                plt.scatter(rho_G,WI_W0, label="sim"+str(ci+1)+"sem"+
                    str(se+1))
                plt.title(titulo)
                plt.xlabel('Factor de carga')
                plt.ylabel('Wi/W0')
        ci +=1
        #plt.legend()
        save_results_to = 'C:\\Universidad\\TFG\\Graficos\\' +
            archivo2
        plt.savefig(save_results_to + titulo+'.png', dpi=300)
        #plt.savefig(titulo+'.svg', dpi=1000)
        plt.show()
        plt.ioff()


# Grafico para las demoras de autobuses
for lin2 in range(len(vector_paradas)):
    for par2 in range(vector_paradas[lin2]-1):
        ci2= 0
        for sim2 in range(simulaciones):
            for se2 in range(semillas):
                WIBNORM_G = WIB_NORM[sim2*num_lin+se2*simulaciones*(
                    num_lin)+lin2][1][1][par2]
                WIB_G = WIB[sim2*num_lin+se2*simulaciones*(num_lin)+
                    lin2][1][1][par2]
                rhoB_G = rhoB[sim2*num_lin+se2*simulaciones*(num_lin)
                    +lin2][1][1][par2]
                nombre_linea = "Linea"+str(lin2+1)+" "
                nombre_parada = "Parada"+str(par2+1)
                titulo = str(nombre_linea)+str(nombre_parada)
                WI2_W02 = WIB_G/WIBNORM_G
                plt.scatter(rhoB_G,WI2_W02, label="sim"+str(ci2+1)+"
                    sem"+str(se2+1))
                plt.title(titulo)
                plt.xlabel('Factor de carga')
                plt.ylabel('Wi/W0')
        ci2 +=1
        #plt.legend()
        save_results_to = 'C:\\Universidad\\TFG\\Graficos\\' +
            archivo_buses
        plt.savefig(save_results_to + titulo+'.png', dpi=300)
        #plt.savefig(titulo+'.svg', dpi=1000)
        plt.show()
```

```
            plt.ioff()



    # Grafico para las demoras de autobuses
    for lin2 in range(len(vector_paradas)):
        for par2 in range(vector_paradas[lin2]-1):
            ci2= 0
            for sim2 in range(simulaciones):
                for se2 in range(semillas):
                    LIB_G = LIB[sim2*num_lin+se2*simulaciones*(num_lin)+
                        lin2][1][1][par2]
                    rhoB_G = rhoB[sim2*num_lin+se2*simulaciones*(num_lin)
                        +lin2][1][1][par2]
                    nombre_linea = "Linea"+str(lin2+1)+" "
                    nombre_parada = "Parada"+str(par2+1)
                    titulo = str(nombre_linea)+str(nombre_parada)
                    plt.scatter(rhoB_G,LIB_G, label="sim"+str(ci2+1)+"sem
                        "+str(se2+1))
                    plt.title(titulo)
                    plt.xlabel('Factor de carga')
                    plt.ylabel('L')
            ci2 +=1
            #plt.legend()
            save_results_to = 'C:\\Universidad\\TFG\\Graficos\\' +
                archivo_buses
            plt.savefig(save_results_to + titulo+'.png', dpi=300)
            #plt.savefig(titulo+'.svg', dpi=1000)
            plt.show()
            plt.ioff()

    # Tiempo total de ejecucion
    final_time = datetime.now()
    total = final_time-inicial
    print(total)
```