UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

Predicción de calidad de enlace en redes comunitarias inalámbricas basadas en OLSR a partir de datos obtenidos con una frecuencia alta de muestreo

Autor:

D. Carlos Mediavilla Pastor

Tutores:

Dr. D. Miguel Luis Bote Lorenzo

Dr. D. Eduardo Gómez Sánchez

Valladolid, 3 de septiembre de 2019

TÍTULO:	**Predicción de calidad de enlace en redes comunitarias inalámbricas basadas en OLSR a partir de datos obtenidos con una frecuencia alta de muestreo**
AUTOR:	**D. Carlos Mediavilla Pastor**
TUTORES:	**Dr. D. Miguel Luis Bote Lorenzo** **Dr. D. Eduardo Gómez Sánchez**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. D. Juan Ignacio Asensio Pérez**
VOCAL:	**Dr. D. Ignacio de Miguel Jiménez**
SECRETARIO:	**Dr. D. Juan Pablo Casaseca de la Higuera**
PRESIDENTE	
SUPLENTE:	**Dr. D. Santiago Aja Fernández**
VOCAL	
SUPLENTE:	**Dr. Dª. María Jesús Verdú Pérez**
SECRETARIO	
SUPLENTE:	**Dr. D. Manuel Rodríguez Cayetano**
FECHA:	**12 de septiembre de 2019**
CALIFICACIÓN:	

Resumen de TFM (150 palabras)

Las redes comunitarias inalámbricas son redes malladas, creadas por varios usuarios que residen en una zona común, permitiendo a estos usuarios conectarse a Internet o acceder a otros servicios. En estas redes los enlaces pueden ser poco fiables, por lo que se utilizan protocolos de encaminamiento como OLSR, que utiliza la calidad de enlace como métrica de coste. El uso de técnicas de aprendizaje automático puede ser útil a la hora de predecir el estado futuro de los enlaces, y consecuentemente, mejorar el encaminamiento. A este respecto, trabajos previos han estudiado cómo el aprendizaje automático en línea puede predecir con éxito la calidad de los enlaces de una red. No obstante, estos trabajos parten de datos con frecuencias de muestreo muy bajas, que pueden no ser realistas. El objetivo de este TFM es analizar algoritmos de aprendizaje automático en línea que realicen buenas predicciones en el contexto de un muestreo de alta frecuencia.

Palabras clave

Redes de comunidad, redes en malla inalámbricas, predicción de calidad de enlace, aprendizaje automático en línea, alta frecuencia de muestreo.

Abstract (150 words)

Wireless Community Networks are mesh networks, built by several users living in the same area, providing access to services such as Internet. In this kind of networks, the links can be unreliable, and thus routing protocols like OLSR, which uses link quality as cost metric, are used. Machine learning techniques can be useful for forecasting future link state, and therefore, improving the operation of the network. Previous works have studied how online machine learning can successfully forecast link quality of a network. Nevertheless, these works are based on data collected with low sampling rates, which can be unrealistic. The objective of this project is to analyze online machine learning algorithms which make good predictions in a high frequency sampling context.

Keywords

Community networks, wireless mesh networks, link quality prediction, online machine learning, high frequency sampling.

Agradecimientos

Antes de comenzar este trabajo, quería dar las gracias a todas las personas, que, de una u otra manera, me han ayudado a terminar esta etapa de mis estudios. En primer lugar, en lo académico, gracias a todos los profesionales de la ETSIT que han puesto todo su empeño para formarnos como Ingenieros. Especialmente quería dar las gracias a los tutores de este Trabajo Fin de Máster, Miguel y Eduardo, que previamente fueron mis tutores del Trabajo Fin de Grado y mis profesores, por todas las dudas resueltas a lo largo de sus asignaturas, así como durante estos Trabajos, que no han sido pocas. Por otro lado, y sin alejarme de la Universidad, quería dar las gracias a mis ya excompañeros de clase, ya que sin ellos estos años se habrían hecho mucho más largos.

En lo personal, quería dar las gracias en primer lugar a mi familia más cercana, mi padre, mi madre, mi hermano. También quería dar las gracias a mi novia, por entenderme como nadie y darme su apoyo en los momentos que más lo necesitaba, gracias de corazón.

Gracias a todos.

Índice

Lista de figuras

Lista de tablas

"It's like everyone tells a story about themselves inside their own head. Always. All the time. That story makes you what you are. We build ourselves out of that story.

Patrick Rothfuss, The Name of the Wind

Capítulo 1. Introducción

1.1 Motivación

Una red comunitaria es una infraestructura de red extendida a lo largo de varias localizaciones en un vecindario, y a la cual los miembros de la comunidad tienen permiso de acceso [1]. Las redes comunitarias o redes de comunidad son infraestructuras distribuidas, descentralizadas y a gran escala compuestas de varios nodos, enlaces y servicios cuyos recursos están disponibles a un grupo de personas viviendo en la misma zona [2]. Las redes comunitarias son construidas por una gran variedad de razones. Algunas proporcionan acceso a Internet a localizaciones remotas donde el coste es demasiado alto para los proveedores comerciales de conexión a Internet [1], otras simplemente son creadas debido a las posibilidades técnicas de las redes [3]. Hay redes comunitarias construidas con unos pocos nodos y otras con miles de ellos. En muchas ciudades europeas se están creando redes de comunidad inalámbricas con decenas de nodos. En Atenas una única red comunitaria inalámbrica (AWMN) incluye más de 2400 nodos, mientras que en España la red Guifi es una composición de varias redes comunitarias inalámbricas con más de 33000 nodos y creciendo [2]. Esto permite que miles de nodos conecten decenas de miles de familias, asociaciones y oficinas públicas, con un enfoque no lucrativo y basado en la organización de las comunidades. Las redes comunitarias crecen y cambian con el paso del tiempo. Los nodos y los enlaces pueden fallar y necesitan mantenimiento, además de que nuevos nodos y enlaces pueden ser añadidos. Aunque algunas redes comunitarias utilizan profesionales a sueldo para instalaciones y mantenimiento, la mayoría de las redes de comunidad son administradas por voluntarios [3].

Aunque la variedad de redes comunitarias es alta, la mayoría de ellas tienen objetivos y desafíos comunes [3]. Los principales retos son en primer lugar los organizacionales, fundamentalmente debidos a la necesidad de obtener fondos para mantener y expandir la red. Por otra parte, el mantenimiento también destaca como uno de los problemas principales, incluyendo monitorización, actualizaciones y configuración. Debido a la naturaleza de estas redes, muchos de los enlaces son inalámbricos, y por ello presentan algunos retos como la alta variabilidad de la calidad del enlace (LQ, del inglés *link quality*), lo que supone el principal problema para muchas de ellas. El protocolo de encaminamiento utilizando en este tipo de redes es mayoritariamente OLSR (*Optimized Link State Routing*) [3], el cual calcula las mejores rutas desde cada nodo hasta el resto de los nodos de la red, minimizando el coste de cada una de ellas. Este coste se puede definir de varias formas, entre ellas, a partir de la calidad de los enlaces, es decir, su valor de LQ.

A este respecto, trabajos previos [4] afirman que el seguimiento de la LQ permite a los protocolos de encaminamiento evitar los enlaces asimétricos (dado que la métrica de coste los penaliza), aunque puede que el seguimiento no sea suficiente, y se usen datos anticuados para encaminar los nuevos paquetes [5]. En este sentido, la predicción de LQ se ha demostrado como una mejor técnica frente al seguimiento de la LQ [6], ya que permite tomar las decisiones de encaminamiento con el conocimiento del futuro próximo de la red, que es cuando los paquetes van a ser encaminados, y no con la información actual, que es posible que varíe hasta dicho momento [7]. En otro contexto semejante [8], el de las redes inalámbricas de sensores (WSN, *Wireless Sensor Networks*), también se ha propuesto la predicción de la calidad del enlace como una aproximación interesante de cara a disminuir la probabilidad de pérdida de datos. Hay que hacer notar que las redes de comunidad y las redes de sensores, pese a tener objetivos diferentes (unas comunican usuarios, otras transmiten parámetros medidos), comparten la misma problemática (topología dinámica, comunicación multisalto, dispositivos poco potentes). Otro contexto semejante en el que se están utilizando estas técnicas es el de las redes de robots [9]. En un trabajo reciente [10] se propone el uso de teoría de juegos y análisis de supervivencia para predecir la existencia o no de un enlace en un momento del futuro (*temporal link prediction*). En [11] se analiza este mismo problema utilizando modelos no lineales.

Dentro de los trabajos que se enfocan en el problema de la predicción del valor futuro de LQ, [7] sigue un enfoque basado en aprendizaje automático en lote (*Batch*) para la predicción de la LQ en la red comunitaria Funkfeuer de Austria, utilizándose técnicas aplicadas a la predicción de series temporales, concretamente, técnicas de regresión (*Regression*) y agrupación (C*lustering*). En dichos experimentos se obtienen buenos resultados utilizando técnicas de regresión como Árboles de Regresión (*Regression Trees*) o Máquinas de Soporte Vector (*Support Vector Machines*).

No obstante, como muestra [7], para poder mantener la calidad de la predicción a lo largo del tiempo se hace necesario construir modelos actualizados cada cierto tiempo, con una frecuencia tal que puede hacer inviable su aplicación en entornos reales dado el alto coste computacional necesario. Una de las principales características de un predictor es su adaptabilidad [12], dado que en redes muy dinámicas el predictor debería ser capaz de adaptarse a los cambios. Otra característica importante es la capacidad del predictor de ser *plug and play*, es decir, que sea capaz de funcionar en cualquier red sin necesidad de realizar un proceso previo de adaptación, o lo que es lo mismo, que no sea necesario que el predictor esté varios días recogiendo datos para formar un modelo. Esto último es muy

relevante, ya que en redes de comunidad donde los enlaces pueden aparecer y desaparecer de manera constante, nos interesa ser capaces de poder predecir lo antes posible. Los dispositivos de red en las redes comunitaria no son grandes ordenadores de altas prestaciones, sino pequeños encaminadores económicos con poca capacidad de procesamiento. Es en este contexto en el que el planteamiento de una predicción incremental y de bajo coste computacional tiene sentido, con el objetivo de ser implementable. Si finalmente se logra proponer un sistema de predicción incremental de LQ suficientemente preciso, los algoritmos de encaminamiento serán capaces de evitar los enlaces que vayan a fallar en el futuro, y de esta manera, minimizará el número de paquetes perdidos, dando un mejor servicio a los usuarios de la red de comunidad.

A este respecto, en [13] , se reprodujo en primer lugar el trabajo realizado por [7], llegando a unas conclusiones semejantes (no existe un valor óptimo de ventana, se puede predecir a varios instantes vista sin mucha degradación, el modelo se degrada con el tiempo). No obstante, el trabajo de [7] presentaba varias limitaciones, entre las que se encontraban el estudio de la carga computacional de los algoritmos, la falta de un *baseline* (algoritmo básico) con el que comparar los resultados o la posibilidad de reentrenamiento de los algoritmos, por lo que en [13] se atajaron estos problemas. Los resultados obtenidos fueron esclarecedores, en tanto en cuanto ninguno de los algoritmos propuestos por [7] mejoraban el *baseline*, si bien alguno de ellos se mantenía en niveles semejantes. Además, la carga computacional resultó ser extremadamente elevada para algunos de los algoritmos, concretamente, para el que mejores resultados producía. Por estos motivos, se analizaron algoritmos de aprendizaje en línea, que implicaban una carga computacional baja y se obtuvieron mejores resultados que el *baseline*, pese a no ser estadísticamente significativos.

Posteriormente, se extendió este trabajo en [14] con resultados igualmente satisfactorios, proponiéndose un algoritmo híbrido que combinaba un método de aprendizaje automático en línea (*Online Perceptron*) con el *baseline*, mejorando a este último en un 12.5%. No obstante, en todos estos trabajos previos existía una limitación para el estudio, ya que los datos de partida estaban tomados cada 5 minutos, lo que en una red real puede ser un tiempo demasiado elevado para que las predicciones lleguen a tener utilidad práctica.

1.2 Objetivos

En este trabajo se pretende continuar el previamente realizado en [13] y [14], centrándose en el estudio de algoritmos de aprendizaje automático en línea a partir de datos capturados con una alta tasa de muestreo. Es por ello por lo que el objetivo principal de la investigación será el siguiente:

- Analizar el resultado de algoritmos de aprendizaje automático en línea sobre datos reales de una WCN, proponer algoritmos incrementales de predicción de LQ que muestreen con alta frecuencia la LQ.

Para poder llevar a cabo el objetivo propuesto, hay varios objetivos parciales que han de tenerse en cuenta:

- En primer lugar, tenemos que obtener datos reales de una WCN que hayan sido capturados con una alta tasa de muestreo.
- En segundo lugar, se analizarán estos datos desde el punto de vista descriptivo, para saber de qué manera están distribuidos los datos, y de esta manera obtener información relevante de los mismos que pueda resultar útil de cara a las predicciones.
- En tercer lugar, se realizarán experimentos que nos permitan obtener predicciones de LQ para distintos escenarios, utilizando para ello la información que nos haya dado previamente el análisis descriptivo.
- Finalmente, se podrán probar otras alternativas derivadas del trabajo precedente, como utilizar otros parámetros en los algoritmos, o realizar predicciones diferentes a la de LQ.

1.3 Metodología de trabajo

Durante el desarrollo de este Trabajo Fin de Máster se ha seguido la metodología de trabajo propuesta por el Método de Ingeniería [15], adaptándola a nuestro contexto particular, teniendo en cuenta que el objetivo de nuestro trabajo no es crear un producto sino el estudio de nuevos métodos para el problema planteado.

El método de ingeniería es una aproximación sistemática que da soporte a un ingeniero o a un equipo en obtener la solución deseada a un problema, que ha sido especificado por clientes, patrocinadores o inversores que perciben valor en resolver el problema. El método de ingeniería tiene seis etapas:

1. Idea:
   La fase de "Idea" suele empezar con un problema. El problema requiere investigación sobre su viabilidad y su factibilidad. La viabilidad sugiere que existe un valor significativo en obtener la solución. La factibilidad sirve para comprobar si la idea puede ser realizada.

2. Concepto:
   La fase de "Concepto" consta de la generación de numerosos modelos (matemáticos, físicos, simulaciones, simples dibujos o esquemas), todos los cuales deberían expresar los requerimientos del cliente.

3. Planificación:
   La fase de "Planificación" se centra en definir el plan de implementación: identificar el capital humano, las tareas, la duración de las tareas, las dependencias de las tareas, las interconexiones de las tareas, y el presupuesto necesario para conseguir realizar el proyecto.

4. Diseño:
   En la fase de "Diseño" se especifican los detalles, las especificaciones son establecidas. El propósito de esta fase es trasladar los requerimientos del cliente en especificaciones de ingeniería con las que un ingeniero (diseñador) pueda trabajar para diseñar y construir un prototipo.

5. Desarrollo:
   El propósito de la fase de desarrollo es generar la documentación ingenieril: esquemas, dibujos, código fuente, y cualquier otra cosa que demuestre la solución al problema. La solución puede ser un prototipo funcional tangible o bien una simulación funcional intangible. Específicamente, consiste del ciclo iterativo: diseño, testeo, depuración y rediseño.

6. Lanzamiento:
   El "Lanzamiento" incluye la emisión del diseño ingenieril y el paquete de documentación a instalaciones de fabricación para producción. En este punto, todas las pruebas de calificación están completas y el prototipo de trabajo ha demostrado funcionalidad.

En nuestro caso particular, no todas las etapas del método de ingeniería aparecen reflejadas en nuestro trabajo. A continuación, se muestran las distintas etapas del método de ingeniería y cómo se ha particularizado el método para los objetivos de esta investigación.

1. Idea:
   En la etapa de "Idea" se ha entendido el problema existente presentado en la Motivación y detallado en los Objetivos, consistente en obtener una predicción incremental de la LQ a partir de datos capturados con una alta tasa de muestreo. Es una idea viable, ya que proporciona un valor añadido sobre el estado del arte actual y es factible, ya que los medios materiales necesarios son únicamente un ordenador con las herramientas *software* requeridas.

2. Concepto:
   Sobre la etapa de "Concepto" no se ha hecho un gran hincapié, sino que ha sido a lo largo de todo el proceso cuando se han planteado diferentes ideas sobre qué mejoras o variaciones había que realizar sobre la iteración anterior para mejorar los resultados.

3. Planificación:
   En la etapa de "Planificación" no se ha incidido en gran medida, dado que el trabajo realizado ha sido desarrollado de manera individual, y no había costes económicos que financiar.

4. Diseño:
   De manera paralela a la etapa de "Desarrollo", en la etapa de "Diseño" se han diseñado los experimentos que posteriormente se han implementado. Dado que se han realizado varios experimentos, sobre esta etapa se ha iterado aproximadamente el mismo número de veces que sobre la etapa de "Desarrollo", exceptuando aquellos casos de depuración de errores.

5. Desarrollo:
   En la etapa de "Desarrollo" se ha centrado la mayor parte del tiempo de este Trabajo Fin de Máster. Para lograr completar esta etapa, se han realizado varias subtareas que vale la mención reseñar:

- En primer lugar, se ha obtenido acceso a la WCN Funkfeuer Graz y se han capturado los datos relativos a la topología de la red con una alta tasa de muestreo.
- Después se han analizado los datos capturados para tener información previa a los experimentos que pudiera ser de interés para los mismos.
- A continuación, se han realizado los experimentos relativos a la predicción de la LQ de los diferentes enlaces seleccionados, entrenando y testeando de manera simultánea ya que los algoritmos de aprendizaje en línea nos lo permiten.
- Finalmente, para cada uno de los experimentos realizados, se han sacado las conclusiones pertinentes, que han permitido o bien realizar nuevos experimentos con diferentes parámetros, o bien extraer información sobre el problema en cuestión.

En esta etapa se han realizado varias iteraciones, tanto para ajustar diversos parámetros de los experimentos en cuestión, como para depurar errores. Además, se han realizado diferentes experimentos para extraer información relevante sobre diferentes aspectos de la predicción de la LQ.

6. Lanzamiento:
   La etapa de "Lanzamiento" no ha existido como tal en el desarrollo de este trabajo dado que no va a pasar a producción, se podría considerar este documento como la etapa de "Lanzamiento" de este Trabajo Fin de Máster.

1.4 Estructura del documento

En la parte restante del documento se detallará el trabajo realizado en relación con la predicción de la LQ.

En el capítulo 2 se analizará el estado del arte desde diferentes perspectivas. Por una parte, se estudiarán las redes de comunidad, su desarrollo, sus utilidades y sus problemáticas También se analizarán los protocolos de encaminamiento utilizados en las redes de comunidad, Además, se analizarán algunos tipos de aprendizaje automático populares de la actualidad, así como los conjuntos de datos disponibles sobre los que poder realizar aprendizaje. Por otra parte, se analizarán los estudios previos existentes sobre la predicción de la LQ, que incluyen tanto el punto de partida de nuestro trabajo (condiciones de partida), como los resultados con los que hemos de compararnos.

En el capítulo 3 se describirá y analizará el *dataset* obtenido, de cara a realizar posteriormente unas mejores y/o más contextualizadas predicciones.

En el capítulo 4 se realizarán las predicciones de los valores de LQ a partir del *dataset* capturado.

Finalmente, en el capítulo 5 desarrollaremos las conclusiones extraídas del estudio realizado. Además, se plantearán las posibles líneas de trabajo futuro para obtener una perspectiva global donde se localice el trabajo.

Capítulo 2. Estado del Arte

2.1 Introducción

Como se ha mencionado anteriormente, el objetivo de este Trabajo Fin de Máster es proponer algoritmos de aprendizaje automático en línea que mejoren el *baseline* en el contexto de unos datos obtenidos con una alta tasa de muestreo. Para poder cumplir este objetivo, es necesario en primer lugar profundizar un poco más en las características del problema en cuestión.

En primer lugar, en este capítulo se va a hablar de las redes de comunidad, sus problemáticas y los protocolos de encaminamiento que se utilizan en ellas. Por otra parte, se va a profundizar en el problema de la predicción de la calidad del enlace, para lo cual se va a estudiar el aprendizaje automático, así como los *toolkits* existentes en la actualidad. Finalmente, se comentarán brevemente los conjuntos de datos disponibles y se extraerán unas breves conclusiones

2.2 Redes de Comunidad

Para las comunicaciones inalámbricas de área local, la emergencia del estándar IEEE 802.11 [16], conocido comercialmente como Wi-Fi, supuso un gran paso hacia adelante [17]. Conducido por su bajo coste y su facilidad de despliegue, los dispositivos compatibles con el estándar 802.11 se convirtieron en equipamiento estándar en portátiles y dispositivos móviles, presentándose como la tecnología predominante para conexiones inalámbricas locales. La operación en bandas del espectro sin licencia facilitó el despliegue de Wi-Fi, dado que era un camino sencillo para los operadores comerciales, instituciones académicas, o incluso entusiastas de las radiocomunicaciones.

En las modernas zonas urbanas con mucha densidad de población, la cobertura que las WLAN proporciona es en constante crecimiento. Una simple búsqueda de redes inalámbricas nos revela un elevado número de puntos de acceso (AP, *Access Point*), por lo que actualmente el problema de la cobertura no es muy relevante. La facilidad de despliegue de Wi-Fi lo ha hecho omnipresente en áreas urbanas densamente pobladas, y solo era cuestión de tiempo que las redes comunitarias inalámbricas emergieran.

Las redes comunitarias inalámbricas (WCN, *Wireless Community Networks*) han sido desarrolladas gracias a grupos de entusiastas, que, usando equipamiento de redes barato para realizar interconexiones gratuitas, crean redes inalámbricas autónomas [17]. Estas redes proporcionan una gran variedad de servicios,

destacando entre ellos las llamadas VoIP entre los miembros de la comunidad y la compartición de ficheros.

El incremento de cobertura Wi-FI y la emergencia de las WCN hace que surja la cuestión de si alternativas de bajo coste a los servicios tradicionales ofrecidos por los sistemas de telefonía móvil (3G, 4G) pueden ser conseguidas. Además, las WCN podrían utilizarse como proveedor de acceso residencial a Internet, compartiendo el acceso entre varios hogares. No obstante, antes de que este escenario pueda ser realidad, han de resolverse algunos problemas, como los incentivos a los propietarios privados de WLAN para que den acceso libre a sus AP o algunos problemas de seguridad. A continuación, clasificaremos las WCN en función de quién sea el organizador de la iniciativa, y también en función de qué alternativa de diseño se haya adoptado.

En primer lugar, la red puede ser iniciativa de la comunidad, es decir, ser el resultado de los esfuerzos de voluntarios individuales, sin ánimo de lucro. Por otra parte, a veces la iniciativa es comercial, siendo en este caso las empresas mediadoras para el desarrollo de WCN. A este respecto, han surgido iniciativas que permiten a los usuarios compartir sus WLAN a cambio de una pequeña compensación económica, donde podría destacar FON [18]. Finalmente, los gobiernos municipales o regionales a veces ponen AP en espacios públicos, permitiendo acceso gratuito o económico a los usuarios.

Por otra parte, la red puede estar diseñada como una malla inalámbrica, típicamente cuando es iniciativa de la comunidad, dando alguno de los nodos acceso a Internet. Por otra parte, puede estar basada en *hotspots*. En la Figura 1 podemos apreciar la diferencia entre las dos arquitecturas presentadas.

Independientemente de quien organice la iniciativa y de la arquitectura de la WCN, existe una problemática común, que es el cálculo de las rutas que han de seguir los paquetes. Cuando haya más de un salto entre el AP al que se conecta el usuario y la salida a Internet, se hace necesario el uso de protocolos de encaminamiento que calcule la mejor ruta en cada caso.

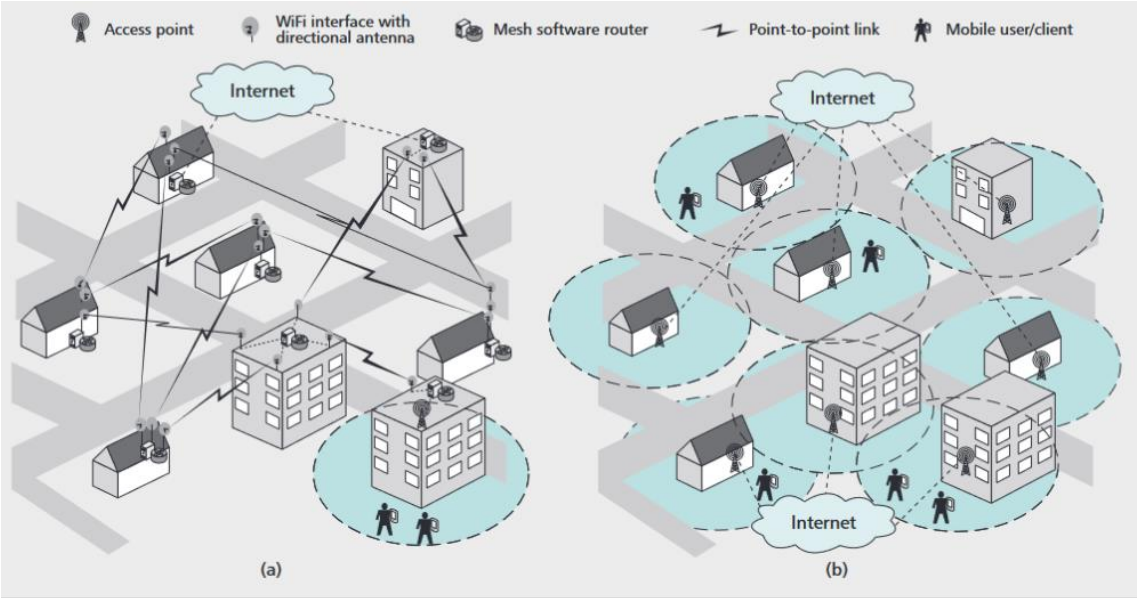

Figura 1: alternativas estructurales para redes comunitarias inalámbricas. Una arquitectura basada en una malla inalámbrica se muestra en (a), mientras que la alternativa basada en hotspots se muestra en (b). Figura tomada de [17].

2.2.1. Protocolos de encaminamiento

Para poder comunicar dos nodos en una red, es necesario encontrar rutas que atraviesen nodos intermedios, optimizando algún criterio relacionado con el coste económico o con la calidad de servicio ofrecida (por ejemplo, menor número de saltos, menor retardo extremo a extremo, menor probabilidad de pérdida, etc.).

Este problema, que en una red cableada podría resolverse de forma estática o semi-estática, se complica en gran medida cuando los nodos que se comunican son inalámbricos, pudiéndose destacar dos posibles fuentes de problemas. En primer lugar, la ubicación de los nodos esta potencialmente sujeta a cambios de posición, por lo que pueden salir de la cobertura del resto de nodos. Por otra parte, dado que en una WCN normalmente los equipos los ponen los usuarios, puede que no estén disponibles todo el tiempo, ya que estos usuarios pueden apagarlos.

Respecto al primer problema, en una WCN no habría que preocuparse, dado que los equipos suelen estar fijos, y solo los terminales finales son los que se mueven, si se estuviese estudiando otro tipo de redes, como las MANET (*Mobile Ad hoc NETwork*), que son redes malladas en las que todos los dispositivos están potencialmente en movimiento, sí que habría que tener esta consideración.

Sea como fuere, en las WCN se hace necesario establecer algún mecanismo con el que poder establecer las rutas más fiables, estables y con menor probabilidad de pérdida, entre el origen y el destino. De esto se encargan los protocolos de

encaminamiento, que a continuación vamos a explicar. No obstante, en primer lugar, se hace preciso hablar de los algoritmos de encaminamiento.

Para formular los problemas de encaminamiento se utilizan grafos [19]. Un grafo *G = (N, E)* es un conjunto de *N* nodos y una colección de *E* aristas, donde cada arista une dos nodos de N. En el contexto del encaminamiento de la capa de red, los nodos del grafo representan los encaminadores (es decir, el lugar donde las decisiones de reenvío de paquetes son tomadas) y las aristas representan los enlaces físicos entre dichos encaminadores. En la Figura 2 se puede ver un ejemplo de un grafo que representa una red de ordenadores, donde cada arista también tiene un valor que representa su coste, que podría reflejar la longitud física de un enlace (por ejemplo, un enlace transoceánico debería tener un mayor coste que un corto enlace terrestre), pero también podría tener en cuenta otras consideraciones como la velocidad del enlace, o incluso el coste económico asociado al uso del enlace. No obstante, de cara a los algoritmos de encaminamiento es irrelevante de qué manera se haya calculado. Un algoritmo de encaminamiento optimiza el cálculo de rutas entre cualquier par de todos en función de dicho coste.

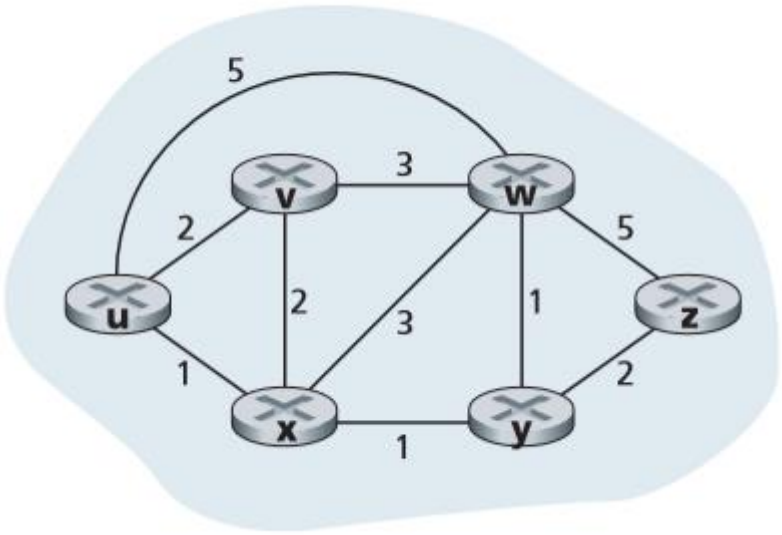

Figura 2: grafo de una red de ordenadores. Figura tomada de [19].

Por otra parte, los algoritmos de encaminamiento pueden calcularse de manera distribuida (cada nodo obtiene la información de la parte de la red más cercana a él, y optimiza las rutas de su entorno con la esperanza de que esto optimice la red de manera global), Típicamente a los algoritmos distribuidos se les conoce como algoritmos de vector de distancias (DV, *Distance-Vector*), dado que cada nodo mantiene un vector de los costes estimados (distancias) al resto de nodos en la red. Uno de más importante es el algoritmo de Bellman-Ford distribuido. También

pueden calcularse de manera centralizada (un nodo obtiene la información de la red completa, calcula las rutas aplicando el algoritmo de encaminamiento sobre el grafo, y finalmente distribuye las rutas). Además, podría seguirse un enfoque centralizado en cada nodo, por lo que cada uno de ellos debería obtener la información completa de la red, pero después ya no sería necesario distribuir las rutas. En la práctica, los algoritmos con información de estado global son mencionados como de estado de enlace (LS, *Link-State*), dado que el algoritmo debe ser consciente del coste de cada enlace en la red. Uno de más importante es el algoritmo de Dijkstra.

Por otra parte, los algoritmos de encaminamiento pueden ser estáticos (cuando las rutas cambian de manera muy lenta a lo largo del tiempo) o dinámicos (las rutas varían a medida que varía el tráfico o la topología de la red). Al contrario que los algoritmos utilizados por los protocolos más frecuentes de Internet, como RIP [20], OSPF [21], o BGP [22], que únicamente intercambian información sobre el número de saltos de cada ruta, en una WCN se hace necesario considerar métricas que reflejen la calidad de los enlaces presentes en la misma, con el objetivo de maximizar la fiabilidad y la estabilidad de las rutas.

2.3 OLSR (*Optimized Link State Routing Protocol*)

Como se ha comentado anteriormente, los protocolos que se utilizan mayoritariamente en Internet son RIP, OSPF y BGP. No obstante, estos protocolos no están optimizados para tipos concretos de redes, en concreto, para las WCN, debido a que las técnicas de inundación utilizadas para transmitir la información no son eficientes, y pueden sobrecargar la red. En esta sección se describirá detalladamente el funcionamiento del protocolo OLSR, el principal protocolo de encaminamiento utilizado en WCN [3], pese a que se utilicen otras alternativas, como pueden ser BGP - ya mencionado anteriormente - o BATMAN (*Better Approach to Mobile Adhoc Networking*), que pretende ser el reemplazo futuro de OLSR. No obstante, algunos autores no comparten la idea de crear un nuevo protocolo para este propósito, por lo que hay otro tipo de propuestas que pretenden extender el funcionamiento del protocolo OSPF, como por ejemplo las extensiones de OSPF para MANET, que se pueden consultar en los RFC 5614 [23] y 5820 [24]. Se explicarán tanto los estándares de OLSR como las implementaciones existentes.

Originalmente, OLSR fue propuesto en [25], como un protocolo para redes móviles inalámbricas. Esta primera concepción, así como la que acabó convirtiéndose en estándar en el RFC 3626 [26], no empleaban el valor de la calidad del enlace a la hora de realizar el encaminamiento [27]. No obstante, primero se explicará el funcionamiento primigenio del protocolo y después de qué manera se añadió esta característica.

De acuerdo a la publicación original [25], OLSR es un protocolo de encaminamiento de estado de enlace proactivo (va recalculando periódicamente las rutas, aunque no se necesiten). El protocolo OLSR es una optimización de un protocolo de estado de enlace, pensado para redes móviles *ad hoc* (MANET). En primer lugar, cada nodo en la red selecciona un conjunto de nodos de entre sus nodos vecinos, los cuales retransmitirán sus paquetes. Este conjunto de nodos vecinos seleccionados es llamado conjunto de *multipoint relays* (MPRs) de este nodo. El resto de los vecinos de un nodo que no estén dentro de su conjunto MPR, leen y procesan el pero no retransmiten los paquetes de *Broadcast*. Para esta finalidad, cada nodo mantiene un conjunto de sus vecinos que son llamados el MPR *Selectors* del nodo. Cada mensaje de *Broadcast* con origen un vecino del MPR *Selectors* es retransmitido. Este conjunto puede cambiar a lo largo del tiempo, que es indicado por los nodos selectores en los mensajes HELLO que se explicarán a continuación.

Cada nodo elige su conjunto MPR de entre sus nodos vecinos de manera que cubra (en términos de cobertura de radio) todos los nodos que están a una distancia de dos saltos. Es decir, que cada nodo en el vecindario de dos saltos de un nodo tiene que tener un enlace bidireccional con el conjunto MPR. Cuando menor es el conjunto MPR, mejor es el protocolo de encaminamiento. En la Figura 3 se muestra el conjunto MPR alrededor del nodo N.

La idea de los *multipoint relays* es minimizar el reenvío de los paquetes de *Broadcast*, reduciendo la duplicidad de retransmisiones en la misma zona. Solamente los *multipoint relays* de un nodo retransmiten los mensajes *Broadcast*. Esta técnica reduce de manera significativa el número de retransmisiones en un procedimiento de inundación. El protocolo es particularmente apropiado para redes grandes y densas, ya que la optimización realizada con los *multipoint relays* funciona bien en este contexto. Cuanto más grande y densa sea la red, más optimización se consigue en comparación con el algoritmo normal de estado de enlace. El protocolo está diseñado para funcionar de una manera totalmente distribuida, sin depender de ninguna entidad central.

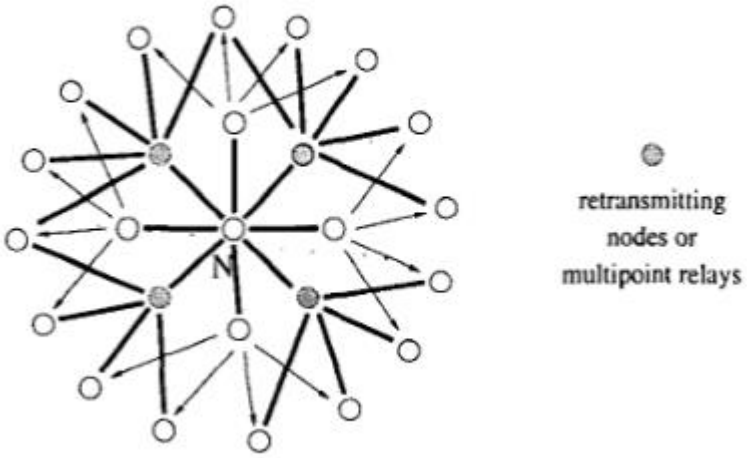

Figura 3: multipoint relays del nodo N. Figura tomada de [25].

Cada nodo debe detectar los nodos vecinos con los que tiene un enlace directo y bidireccional. Para lograr esto, cada nodo periódicamente difunde sus mensajes HELLO, conteniendo información acerca de sus vecinos y el estado de sus enlaces. Estos mensajes son recibidos por los vecinos, pero no son reenviados más allá. Además de los mensajes HELLO, el protocolo utiliza los mensajes TC (*Topology Control*). Los mensajes TC son reenviados por *Broadcast* por la red completa. Esta técnica es similar a la técnica de estado de enlace utilizada por ARPANET, pero con la ventaja de los MPR, que permite una mayor escalabilidad en el encaminamiento intra-AS.

Un mensaje TC es enviado periódicamente por cada nodo de la red, en el que declara su conjunto de selectores MPR, es decir, el mensaje contiene la lista de nodos vecinos que el nodo emisor del mensaje TC ha elegido como un *multipoint relay*. La información difundida a lo largo de la red por los mensajes TC permite a cada nodo construir la topología completa de la red. A partir de esta información, ya se puede ejecutar el algoritmo de encaminamiento correspondiente (Dijkstra) para obtener las rutas al resto de nodos de la red.

Como ya se comentó anteriormente, en los orígenes del protocolo OLSR no se utilizaba la información de calidad del enlace para establecer las rutas, sino que se utilizaba la métrica del número de saltos (como en RIP). Esto quiere decir que una implementación que cumpla el RFC minimizará el número de saltos entre el nodo origen y el resto de los nodos de la red, incluso si esto significa que se prefiera una ruta a través de un enlace muy "malo" frente a otra ruta a través de dos enlaces muy "buenos" [27]. Esta elección probablemente no sea óptima. Por ello, en la implementación 0.4.8 de `olsrd` (implementación del protocolo OLSR del proyecto

OLSR.org [28]) se introdujo una implementación experimental de una métrica basada en ETX (*Expected Transmission Count*).

Para solucionar el problema mencionado anteriormente, se planteó una medida de la calidad del enlace basada en medir la pérdida de paquetes. Como cada nodo periódicamente recibe mensajes HELLO de sus nodos vecinos (por defecto el estándar especificaba cada 2 segundos), existe suficiente información como para determinar la tasa de paquetes perdidos. Si por ejemplo 3 de cada 10 paquetes son perdidos existe un 70% de transmisión satisfactoria, valor conocido como calidad del enlace (LQ, *Link Quality*). Es importante conocer la LQ del enlace en la dirección opuesta, es decir, cuántos de los paquetes que enviamos son recibidos por nuestros nodos vecinos. A esta calidad la llamamos calidad del enlace del vecino (NLQ, *Neighbor Link Quality*).

Estos dos valores, tanto la LQ como la NLQ, oscilan entre 0 (pérdida total de paquetes) y 1 (transmisión correcta de todos los paquetes). Estos valores pueden tener sentido por separado, pero la idea es combinarlos, debido a que el tráfico de información puede ir en ambos sentidos (petición/respuesta), nos interesa la probabilidad de éxito de un viaje de ida y vuelta. La probabilidad se calcula simplemente como el producto de la LQ y la NLQ, es decir $P_{éxito}$ = LQ x NLQ. A partir de este valor se puede calcular el número medio de transmisiones necesarias para que pueda producirse un intercambio de ida y vuelta entre dos nodos, valor conocido como número esperado de transmisiones, ETX, definido por primera vez en [29].

Para poder utilizar ETX como métrica de coste de los enlaces, es necesario además de saber el valor de LQ, calculado por el mismo nodo, el valor de NLQ, del nodo vecino. Por ello, en esta implementación se introdujeron variaciones a los mensajes HELLO que hicieran posible a cada nodo enviar los valores de LQ a sus nodos vecinos. Nótese que debido a esto una versión que cumpliese el estándar, a priori no sería compatible.

Años después, se publicó la segunda versión de OLSR en el RFC 7181 [30], que ya permitió más flexibilidad a la hora de definir la métrica de coste. A raíz de esta publicación, el proyecto OLSR.org comenzó a desarrollar la nueva versión de su implementación, `olsrd2`. Cabe destacar que existen otras implementaciones del protocolo como por ejemplo NLR-OLSR [31] y NET-OLSR [32], pero parece que la que más acogida ha tenido por parte de la comunidad es la primeramente mencionada.

2.4 Predicción de Calidad del Enlace

Tal y como se ha comentado en la sección anterior, OLSR utiliza como valor de coste el valor de ETX obtenido a la hora de calcular las rutas para el encaminamiento. La variabilidad a lo largo del tiempo de la LQ dificulta la optimización del encaminamiento de los paquetes y, por tanto, dificulta la reducción de la pérdida de estos. Esta variabilidad es debida en parte al carácter inalámbrico de los enlaces y a su carácter asimétrico en ocasiones.

A este respecto, trabajos previos [4] afirman que el seguimiento de la LQ permite a los protocolos de encaminamiento evitar los enlaces asimétricos. Esto es debido a que la métrica de coste (por ejemplo, ETX, presentado en la sección anterior) penaliza a este tipo de enlaces. No obstante, puede que en el tiempo necesario para que llegue la información de LQ de nodos lejanos, se reconstruya la topología de la red, y se ejecute el algoritmo de encaminamiento, las rutas calculadas pasen por enlaces cuya calidad haya variado desde que se midió, siendo por ello la ruta inadecuada. Es decir, que las rutas óptimas hace un tiempo pueden no ser las óptimas ahora.

En este sentido, parece razonable que, si la LQ varía a lo largo del tiempo, se puedan realizar predicciones a corto plazo que mejoren el desempeño de los protocolos de encaminamiento, ya que permitiría tomar las decisiones de encaminamiento con el conocimiento del futuro próximo de la red, que es cuando los paquetes van a ser encaminados, y no con la información actual, que es posible que varíe hasta dicho momento [7].

En [7] se realiza un estudio basado en la predicción de LQ utilizando técnicas de aprendizaje automático aplicadas a series temporales, sobre los enlaces de la WCN Funkfeuer de Austria. Concretamente, se utilizan técnicas de regresión (*Regression*) y agrupación (*Clustering*). En dichos experimentos, se obtienen buenos resultados utilizando técnicas de regresión como Árboles de Regresión (*Regression Trees*) o Máquinas de Soporte Vector (*Support Vector Machines*).

Por otra parte, también se analiza la degradación a lo largo del tiempo de los modelos construidos con aprendizaje automático, mostrando claramente la necesidad de actualizar los modelos a lo largo del tiempo. Este problema no se resuelve, dejando pendiente los autores su resolución siguiendo el enfoque planteado en su estudio. Si se utilizasen los mismos algoritmos de aprendizaje propuestos en [7] y se pretendiesen reconstruir los modelos cada cierto tiempo, el coste computacional sería prohibitivo para dispositivos poco potentes como puedan ser los encaminadores. Por otro lado, en [12] se realiza una investigación acerca de la predicción de la calidad del enlace en redes de sensores. El enfoque seguido es

similar al de [7]. Sin embargo, en este caso se utilizan técnicas de aprendizaje automático en línea, para superar las limitaciones que pueden presentar los algoritmos de aprendizaje en lote, como ya se ha comentado anteriormente.

A este respecto, en mi Trabajo Fin de Grado [13] , en primer lugar, se reprodujo el trabajo realizado por [7], llegando a unas conclusiones semejantes. La primera de ellas, que no existe un valor óptimo de ventana tempora, ya que no había diferencias estadísticamente significativas entre los resultados de distintos tamaños. Por otra parte, se pueden realizar predicciones a varios instantes vista sin verse los resultados muy degradados, pese a que el modelo en sí, sí que se degrade a medida que pasa el tiempo. No obstante, el trabajo analizado [7] presentaba varias limitaciones, entre las que se encontraban el estudio de la carga computacional de los algoritmos, la falta de un *baseline* con el que comparar los resultados o la posibilidad de reentrenamiento de los algoritmos, por lo que en [13] se trabajó sobre estos problemas. Los resultados obtenidos fueron esclarecedores, en tanto en cuanto ninguno de los algoritmos propuestos por [7] mejoraban el *baseline,* si bien alguno de ellos se mantenía en niveles semejantes. Además, la carga computacional resultó ser extremadamente elevada para algunos de los algoritmos, concretamente, para el que mejores resultados producía. Dado que ninguno de los algoritmos mejoraba el *baseline* y además la carga computacional era demasiado elevada (tan elevada que un reentrenamiento del modelo de manera frecuente se hacía inviable), se descartó continuar por esta línea, y se pasó a analizar algoritmos de aprendizaje en línea, que implicaban una carga computacional mucho menor y se obtuvieron mejores resultados que el *baseline*, pese a no ser estos resultados estadísticamente significativos en todas las situaciones. En concreto, los experimentos realizados sobre los conjuntos de datos con más variabilidad demostraron que los algoritmos de aprendizaje automático en línea se comportaban mejor que los algoritmos de aprendizaje en lote y el *baseline*.

Posteriormente, se extendió este trabajo en [14] con resultados igualmente satisfactorios, proponiéndose un algoritmo híbrido que combinaba un método de aprendizaje automático en línea (*Online Perceptron*) con el *baseline*, mejorando a este último en un 12.5%. No obstante, en todos estos trabajos previos existía una limitación para el estudio, ya que los datos de partida estaban tomados cada 5 minutos, lo que en una red real puede ser un tiempo demasiado elevado para que las predicciones lleguen a tener sentido. Teniendo un *dataset* capturado con una alta frecuencia de muestreo (p. ej., cada segundo) nos proporcionaría más información de cara a analizar la propia red, así como a presentar posibles algoritmos o combinación de estos que aprovechen esta mayor cantidad de datos (si fuese posible). Es en este problema en el que se centra este Trabajo Fin de Máster.

2.5 Aprendizaje Automático

El aprendizaje automático (*Machine Learning*), según Arthur L. Samuel, pionero en el campo de la inteligencia artificial y considerado como el creador del término *Machine Learning*, se puede definir como "el campo de estudio que da a los ordenadores la capacidad de aprender sin ser explícitamente programados para ello" [33]. Más formalmente, Tom Mitchell [34] dijo que "un programa de ordenador aprende de la experiencia E respecto a una tarea T con una medida del rendimiento P, si su rendimiento en la tarea T medido por P mejora con E". El aprendizaje automático tiene muchas aplicaciones actualmente [35], como por ejemplo en robótica, biología, minería de datos, reconocimiento de escritura, visión por ordenador, buscadores para clasificar las páginas web, etcétera. En resumen, el aprendizaje automático está formado por un conjunto de algoritmos que permiten extraer modelos a partir de datos u observaciones de un sistema de cualquier tipo, con el fin de realizar tareas de clasificación o predicción de ese sistema.

El aprendizaje automático se ha vuelto muy importante debido al crecimiento de la cantidad de datos disponible [36]. En el año 2000, la cantidad total de datos disponible en la Web variaba entre 25 y 50 terabytes. En 2005, el tamaño aproximado era de 600 terabytes. Hoy en día, la cantidad total es casi incalculable.

En primer lugar, se puede hacer una primera distinción entre los diferentes tipos de aprendizaje automático entre los que se ejecutan sobre un conjunto de datos fijo, que se conoce como aprendizaje en lote (*Batch*) y sobre los que se ejecutan sobre un flujo de datos (*Stream* u O*nline*). El aprendizaje en lote se emplea para aprender modelos sobre un conjunto de datos fijo, que se ha recolectado a lo largo de un periodo de tiempo. Este enfoque puede ser suficiente, pero existen aplicaciones en las que se vuelve insuficiente. Por una parte, los datos pueden llegar a una velocidad tan elevada, que el almacenarlos conlleve un gasto inasumible. Además, actualizar periódicamente los modelos puede ser demasiado costoso, por lo que una alternativa el tiempo real (*Stream*) es la solución ideal, en la que los datos no se almacenen, y el modelo se actualice progresivamente a medida que se obtienen nuevos datos.

Por otra parte, se puede hacer una distinción entre aprendizaje supervisado y no supervisado. El aprendizaje supervisado es el tipo más común de aprendizaje automático [35], en el cual se le entregan datos correctos al algoritmo (es decir, se conoce la correspondencia entre la entrada y la salida) y tiene que ser capaz de realizar predicciones cuando se le alimenta con nuevos datos. Por otro lado, el aprendizaje no supervisado se le entregan datos al algoritmo y el algoritmo tiene que ser capaz encontrar estructura en ellos, formando distintos grupos. Esto es lo

que se conoce como problema de agrupamiento (*Clustering*). El aprendizaje no supervisado es utilizado en multitud de situaciones, como por ejemplo clasificación automática de noticias, genoma humano, análisis de redes sociales, astronomía, etcétera

Finalmente, cabe destacar que todos estos algoritmos se pueden ejecutar de manera distribuida. Tradicionalmente, el cuello de botella que impedía el desarrollo de sistemas más inteligentes era la cantidad limitada de datos disponibles [36]. No obstante, ahora el factor limitante es la imposibilidad de los algoritmos de aprendizaje de usar todos los datos para aprender en un tiempo razonable. En este contexto, el aprendizaje distribuido parece ser una línea prometedora de investigación, dado que distribuye el proceso de aprendizaje a lo largo de varias estaciones de trabajo, como una manera natural de escalar los algoritmos.

2.6 *Toolkits* de aprendizaje automático

Existe un gran número de *toolkits* o herramientas que permiten probar un conjunto elevado de algoritmos de aprendizaje automático. En este documento vamos a destacar tanto los más importantes en la actualidad, como aquellos que han sido utilizados para el desarrollo de este Trabajo Fin de Máster. Vamos a exponer las distintas alternativas en función de la clasificación realizada en el apartado anterior, de acuerdo a lo que se puede ver en la Figura 4.

- En lote: existe una gran variedad de *toolkits* de aprendizaje automático en lote no distribuido, ya que se trata del caso "más sencillo" de todos los posibles en este campo. De entre todas las alternativas posibles, vamos a destacar tres:
    - WEKA [37]: es una de las herramientas que se utilizó en el desarrollo de mi Trabajo Fin de Grado, ya que es la herramienta que los autores de [7] utilizaron en sus experimentos. Dispone de interfaz gráfica, API Java, e interfaz de línea de comandos, por lo que es una herramienta muy flexible en tanto en cuanto puede ser utilizada con varias finalidades y por distintos tipos de usuarios. Incluye una gran variedad de algoritmos para regresión, clasificación y agrupación, así como muchos módulos instalables para aumentar las funciones base.
    - R [38]: es un lenguaje de programación enfocado a la computación estadística (modelado lineal y no lineal, tests estadísticos, análisis de series temporales, clasificación, agrupación, etc.) y a los gráficos. Pese a ser un lenguaje

creado en 1993, su popularidad se ha incrementado enormemente en los últimos años con el auge de la minería de datos.

  - scikit-learn [39]:es una biblioteca de Python enfocada al aprendizaje automático, construida sobre las bibliotecas NumPy, SciPy y matplotlib. Últimamente, ha gozado de una gran popularidad, siendo incluso utilizada por empresas como Spotify y Booking [40] para realizar las recomendaciones de música y hoteles,

- En línea:
  - MOA [41]: es la herramienta que se ha utilizado en el desarrollo de este Trabajo Fin de Máster. Es similar a WEKA, pero enfocado en el aprendizaje automático sobre flujos de datos. Implementa una gran variedad de algoritmos de aprendizaje automático en línea de "última generación", para regresión, clasificación y agrupación. Al igual que WEKA, dispone de interfaz gráfica, API Java e interfaz de línea de comandos.
  - scikit-multiflow [42]: esta herramienta está inspirada en MOA, MEKA (extensión multietiqueta de WEKA [43]) y scikit-learn. Está implementado en Python y complementa a scikit-learn en tanto en cuanto éste se centra en aprendizaje en lote. Actualmente contiene generadores de flujos de datos, clasificadores multisalida/multiobjetivo, detectores de cambio y métodos de evaluación.

A modo de resumen, en la Figura 4 se puede ver un gráfico que ilustra la clasificación expuesta en esta sección. Nótese que en la referencia empleada se habla de minería de datos y no de aprendizaje automático, debido a que el aprendizaje automático es solo una de las muchas técnicas que se utilizan en la minería de datos. En este caso, las herramientas de aprendizaje automático serían las inferiores de cada rama (Mahout, SAMOA, R, WEKA, MOA).

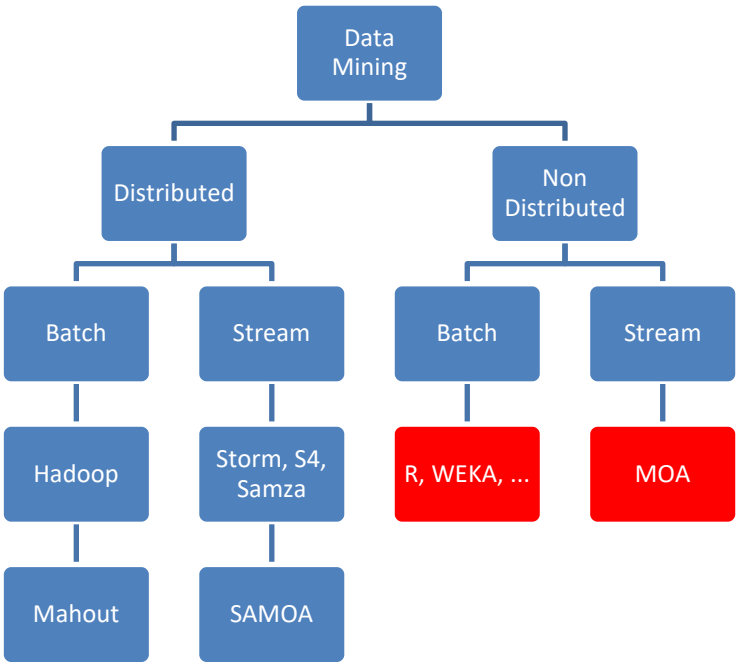

Figura 4: clasificación del aprendizaje automático. Figura tomada de [44].

2.7 Conjuntos de datos disponibles

FunkFeuer es una WCN experimental desplegada en varias ciudades de Austria. En el estudio realizado por [7], existían unos datos disponibles a través del proyecto CONFINE [45], que lamentablemente ya no estaban disponibles en el momento del inicio del trabajo previo a este Trabajo Fin de Máster, mi Trabajo Fin de Grado. No obstante, sí que había datos disponibles de la misma red (Austria), de un periodo más reciente, por lo que se pudieron realizar los experimentos sin ningún problema. Al término de ese Trabajo Fin de Grado, dichos datos ya no estaban disponibles.

Por otra parte, existe un sitio web [46], correspondiente a la red Funkfeuer de la ciudad de Graz, en el que van se actualizando los datos de calidad de enlace correspondientes a sus enlaces, pero lamentablemente, la web tiene prohibido el acceso a robots de recogida de datos, por lo que sin permiso no se podría acceder al uso de estos datos. Además, la periodicidad de los datos no era la adecuada para este Trabajo Fin de Máster. Por ello, se obtuvo acceso directo a esta red y se pudo construir por nuestra parte un *dataset* que sí cumpliera los requisitos deseados respecto a su periodicidad. Para ello, se conectó una máquina virtual a la red de Graz mediante un túnel VPN para el que nos dieron acceso. Una vez conectado, esta máquina virtual ejecutó el cliente del protocolo OLSR actuando como un nodo de dicha red. Dado que OLSR es un protocolo de estado de enlace, se puede obtener en un único nodo la información de LQ de toda la red. Realizando consultas periódicas a este cliente OLSR se fue construyendo durante 2 semanas (14 días) el *dataset*

completo, publicado en [47]. El procedimiento completo de cómo se realizó está detallado en el Anexo A.

2.8 Conclusiones

Durante este capítulo se han presentado diferentes problemáticas relativas al encaminamiento en redes comunitarias. En concreto, se ha incidido en el problema de cálculo de rutas, que en cualquier red variable es muy relevante. Este cálculo de rutas puede llevar a rutas malas o ineficientes si se emplean datos no actualizados. En este sentido, es necesaria una predicción de la calidad del enlace, para realizar el encaminamiento con datos del futuro inmediato. Esta predicción es viable mediante técnicas de aprendizaje automático en lote, pero los modelos obtenidos se degradan con el tiempo, y su reentrenamiento tiene un coste computacional elevado (y, por tanto, un tiempo elevado). La solución a este problema está en el aprendizaje automático en línea, como ha quedado demostrado en trabajos previos. No obstante, estos estudios tienen la principal limitación de que, pese a ser académicamente interesantes, desde el punto de vista práctico telemático no son muy relevantes debido a la baja tasa de muestreo.

En la parte restante del documento se presentará el trabajo realizado, partiendo de los resultados obtenidos en [13] y [14], empezando por el análisis del *dataset* capturado, y se continuará con nuevos experimentos de predicción que nos permitan obtener buenos resultados en el contexto de un muestreo de alta frecuencia.

Capítulo 3. Estudio del nuevo conjunto de datos.

3.1 Introducción

En el capítulo anterior se han estudiado las principales problemáticas existentes en cuanto al encaminamiento en las redes de comunidad, destacando la predicción de la calidad del enlace entre todas ellas. Para hacer frente a este problema, se va a continuar el trabajo realizado en [13] y [14]. Para ello, tal y como se ha dicho con anterioridad, es preciso disponer de un *dataset* con datos capturados a una alta frecuencia de muestreo.

En primer lugar, en este capítulo se describirá la red de la que se han obtenidos los datos. Posteriormente, se realizará un análisis global de la red, tanto de manera geográfica como descriptiva. Después, se analizarán maneras para segmentar el análisis de la red (por horas y seleccionando enlaces). A continuación, se aplicarán estas segmentaciones a los datos y se realizarán nuevos análisis. Se analizará el periodo de muestreo óptimo utilizando técnicas de análisis frecuencial y finalmente se extraerán las conclusiones de este capítulo.

3.2 Funkfeuer Graz

Funkfeuer Graz es una red inalámbrica experimental libre en la ciudad de Graz (Austria) englobada dentro de la iniciativa Funkfeuer [48] orientada a redes abiertas y no comerciales, cuyo objetivo es crear una red desregulada con el potencial de disminuir la brecha digital entre clases. Esta iniciativa se encuentra presente en Viena, Graz, Wels, Innsbruk, Salzburg. En el caso que nos concierne, la mayoría de los nodos se encuentran en la ciudad de Graz y alrededores. Además, algunos nodos más lejanos se encuentran conectados mediante VPN (lo cual podría "romper" un poco la idea de WCN, pero al ser minoría no es muy representativo).

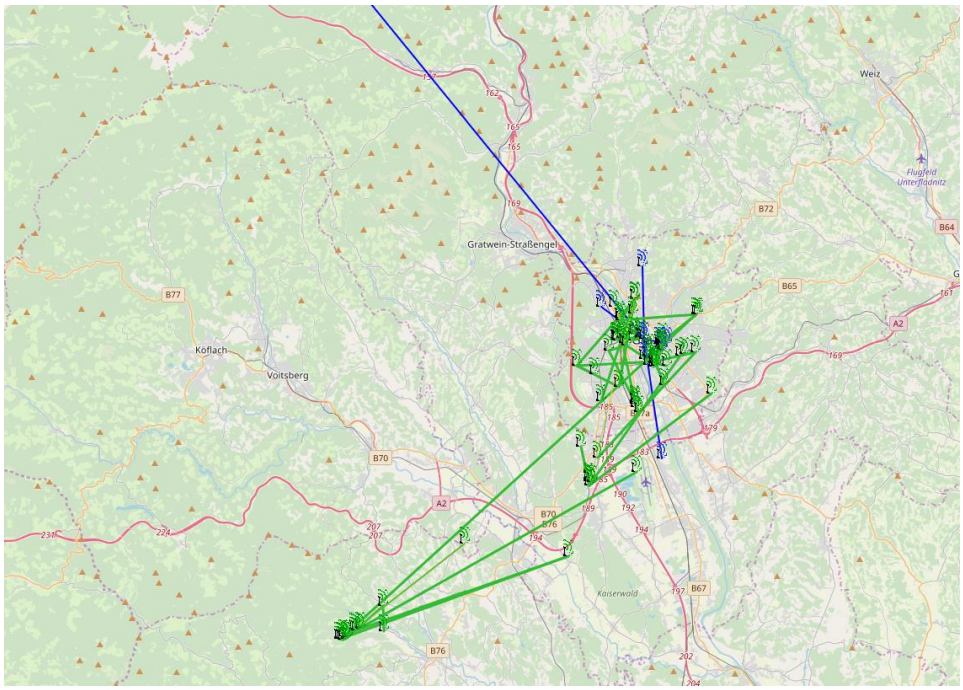

Figura 5: nodos y enlaces de la red Funkfeuer Graz. Verde enlaces reales y Azul enlaces a través de VPN. Captura realizada el 28/05/2019.

3.3 Análisis global de la red

A partir de los datos capturados, se va a proceder a realizar un análisis. En primer lugar, consideramos útil realizar una representación geográfica de los nodos y enlaces capturados en nuestro *dataset*, de manera similar a lo mostrado en la Figura 5, pero concernientes al periodo de los datos obtenidos, representando todos los enlaces que aparecieron en dicho periodo. La posición de cada uno de los nodos se ha obtenido de manera manual a partir de la información del sitio *web*, solo disponible para usuarios. A continuación, vamos a obtener distintas métricas de los enlaces; a saber, número de días de actividad de los enlaces, número de enlaces activos cada día, número de enlaces con respecto al porcentaje de tiempo activo en días con actividad, porcentaje de tiempo activo para enlaces con actividad en cada día del *dataset*, número de enlaces respecto a la media de los valores de LQ, número de enlaces respecto a la desviación típica de valores de LQ, *boxplots* de LQ promedio por día y *boxplots* de desviación típica de LQ por día.

3.3.1. Representación geográfica de la red

Como se ha dicho anteriormente, en primer lugar, vamos a realizar una representación gráfica de todos los enlaces que han aparecido alguna vez en la red durante los 14 días de captura de datos. En la Tabla 1 se puede ver el número de enlaces únicos que hay en cada día del *dataset*, así como el total de enlaces únicos a lo largo de este periodo. No obstante, analizando dichos datos de enlaces y

cotejándolos con la información obtenida en el sitio *web*, hemos descubierto que en la captura de datos hay 20 direcciones IP que no se corresponden a ningún nodp. Muchas de estas direcciones IP son erróneas en formato (número superior a 255, falta de números, exceso de puntos). Comprobando manualmente los enlaces formados por estas direcciones IP (52 enlaces), vemos que algunos de los ficheros que incluyen la información de los enlaces solo tienen una línea de información, por lo que es achacable a algún *bug* de alguna de las partes relativas a la captura de datos. Para el resto la información de ubicación no estaba presente en el sitio *web*. Por tanto, eliminamos los nodos erróneos y al resto les hemos añadido una ubicación dentro del mapa, en la esquina superior izquierda para tenerlos localizados. A este respecto hay que aclarar que algunos de los nodos disponibles en la *web* ya tenían ubicaciones ficticias, ya fuese por motivos de testeo, por ser nodos móviles, u otros motivos. En total, hemos eliminado 16 nodos, con la correspondiente eliminación de 16 enlaces (como hemos indicado anteriormente, estas direcciones IP erróneas solo aparecían en una ocasión).

Día	**Número de enlaces únicos**	**Número de enlaces únicos - corregido**
13 de nov de 2017	1719	1718
14 de nov de 2017	1748	1743
15 de nov de 2017	1730	1728
16 de nov de 2017	1730	1730
17 de nov de 2017	1745	1744
18 de nov de 2017	1731	1728
19 de nov de 2017	1725	1723
20 de nov de 2017	1726	1726
21 de nov de 2017	1735	1734
22 de nov de 2017	1727	1726
23 de nov de 2017	1724	1724
24 de nov de 2017	1686	1686
25 de nov de 2017	1678	1678
26 de nov de 2017	1676	1676
TOTAL	**1816**	**1800**

Tabla 1: Número de enlaces únicos en cada día del dataset y número de enlaces únicos en el dataset completo.

Con la información previa ya limpiada, podemos realizar los primeros mapas de la red. Los mapas se han realizado utilizando `basemap` [49], que es una biblioteca de Python que funciona sobre `matplotlib` [50]. En concreto, se ha utilizado el servicio proporcionado por la API REST de `arcgis` [51] [52], incluido en `basemap`.

En primer lugar, vamos a representar una vista global de Europa para ver donde se encuentra localizada la red y nuestro nodo conectado a través de VPN. Esto

Figura 6: mapa de la red Funkfeuer Graz con todos los nodos y enlaces activos durante el periodo de captura; vista global.

es lo que vemos en la Figura 6. En esta vista no se aprecia nada, por lo que realizamos una nueva representación centrada en la ciudad de Graz, mostrada en la Figura 7. Aquí podemos apreciar que la mayor parte de los nodos se encuentran concentrados en la ciudad, otros en el suroeste, y en la esquina superior están los nodos de los que no teníamos posición. Esta representación nos será útil para compararla con la de secciones posteriores.

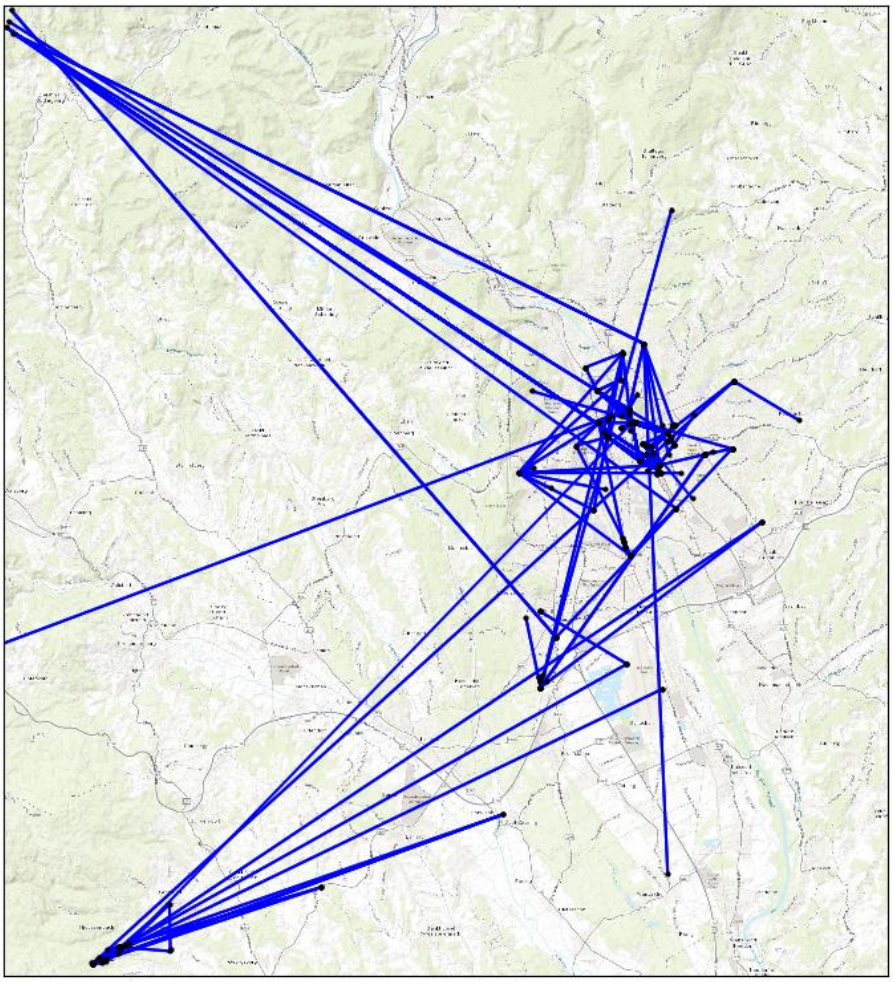

Figura 7: mapa de la red Funkfeuer Graz con todos los nodos y enlaces activos durante el periodo de captura; vista cercana.

3.3.2. Análisis descriptivo de la red

3.3.2.a. Número de días de actividad de los enlaces

En la Figura 8 vemos que la mayor parte de los enlaces de nuestro *dataset* tienen actividad durante los 14 días que lo forman. Existe una cantidad pequeña de enlaces que tiene actividad un número diferente de 14, pero no hay ninguno que destaque en gran medida.

3.3.2.b. Número de enlaces activos cada día.

En la Figura 9 podemos apreciar que el número de enlaces activos cada día se mantiene más o menos constante a lo largo de todo el *dataset*, con unos casi 1750 cada día. Vemos que los últimos 3 días hay un ligero decrecimiento, pero no es muy acusado.

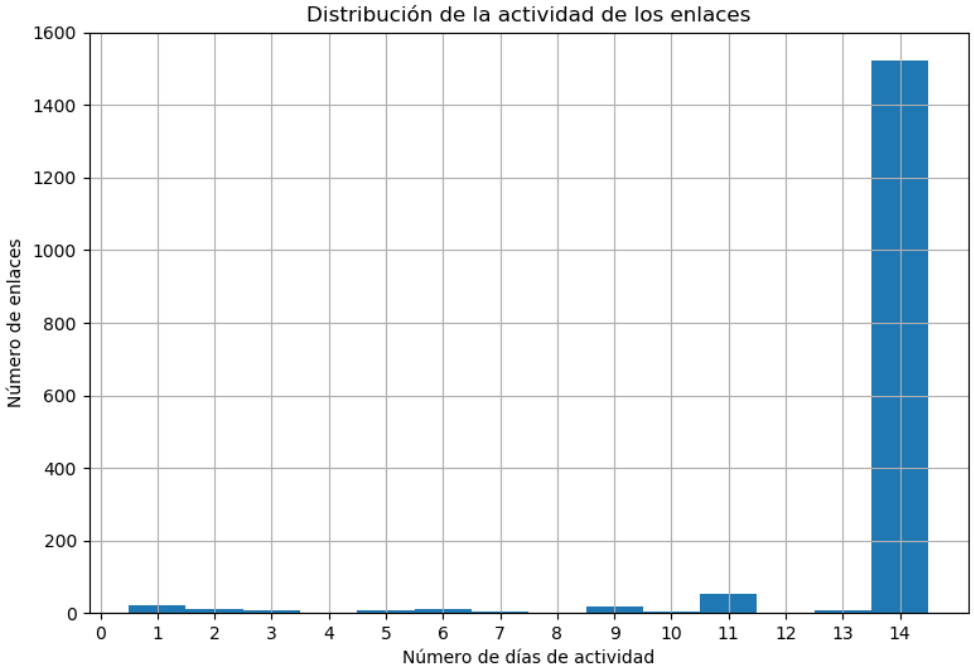

Figura 8: número de días de actividad de los enlaces.

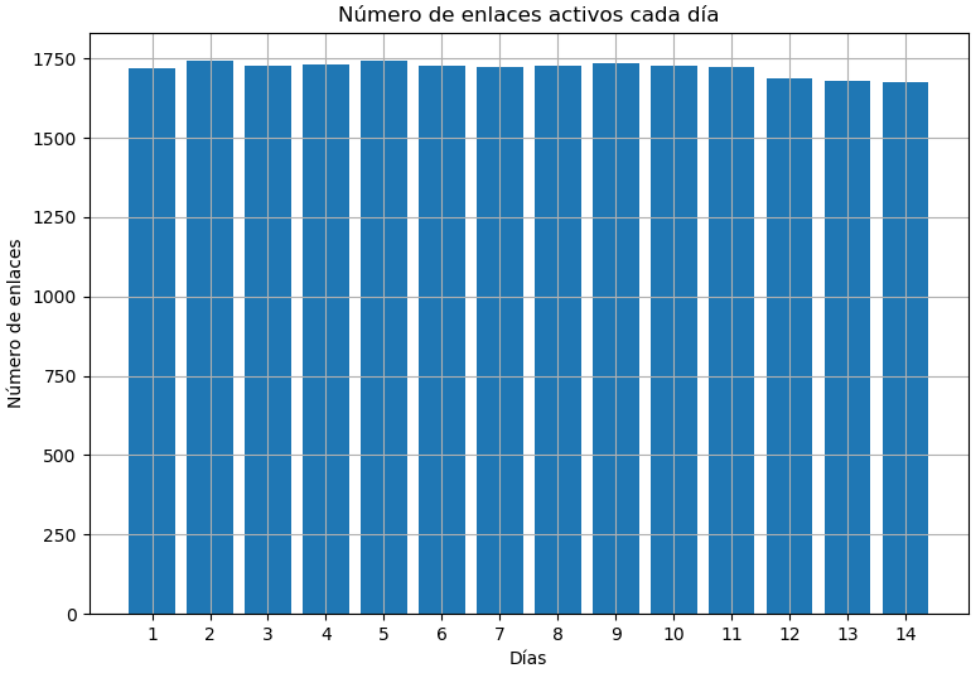

Figura 9: número de enlaces activos cada día.

3.3.2.c. Número de enlaces con respecto al porcentaje de tiempo activo en días con actividad.

En la Figura 10 observamos que la gran mayoría de enlaces están activos entre el 90 y el 100% del tiempo, lo cual es coherente con los resultados apreciados en las figuras anteriores. Además, vemos que hay un pequeño conjunto de enlaces (72, un 4% del total) que están activos poco tiempo, entre un 0 y un 10 % del total. Esto podría ser de utilidad para análisis posteriores.

3.3.2.d. Porcentaje de tiempo activo para enlaces con actividad en cada día del dataset

La Figura 11 complementa a la Figura 10. Podemos ver a lo largo de los 14 días del *dataset* como la mayoría de los enlaces están activos en un porcentaje muy alto del tiempo (la media es el rombo amarillo), pero también hay una pequeña concentración en valores pequeños

3.3.2.e. Número de enlaces respecto a la media de los valores de LQ

En la Figura 12 observamos que la mayoría de los enlaces tienen valores de LQ promedio entre 0.9 y 1, y que el resto se distribuyen de manera monótona con respecto a este valor (a más LQ promedio, más enlaces hay).

3.3.2.f. Número de enlaces respecto a la desviación típica de valores de LQ

En la Figura 13 observamos que la mayoría de los enlaces tienen valores de desviación típica LQ entre 0 y 0.1, y que el resto se distribuyen de manera monótona con respecto a este valor (a mayor desviación típica de LQ, menos enlaces hay, no habiendo ningún enlace con una desviación mayor a 0.5).

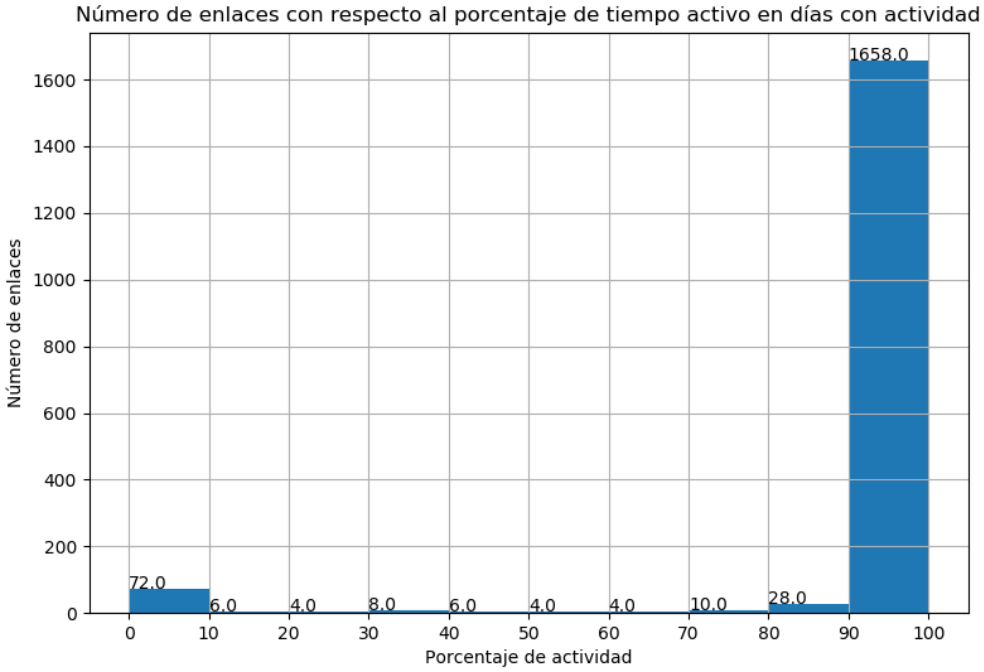

Figura 10: número de enlaces con respecto al porcentaje de tiempo activo en días con actividad.

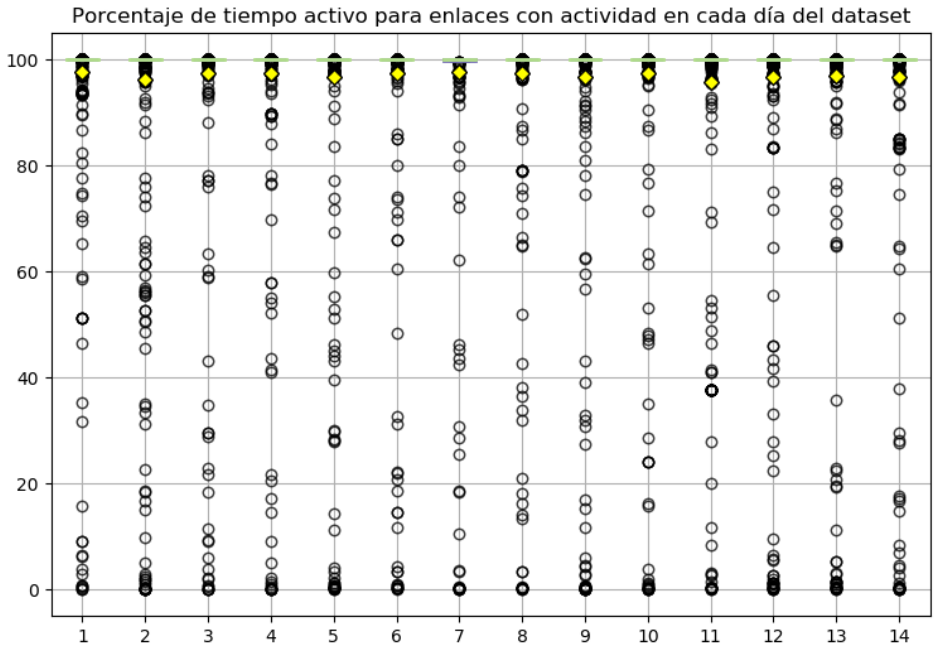

Figura 11: porcentaje de tiempo activo para enlaces con actividad en cada día del dataset.

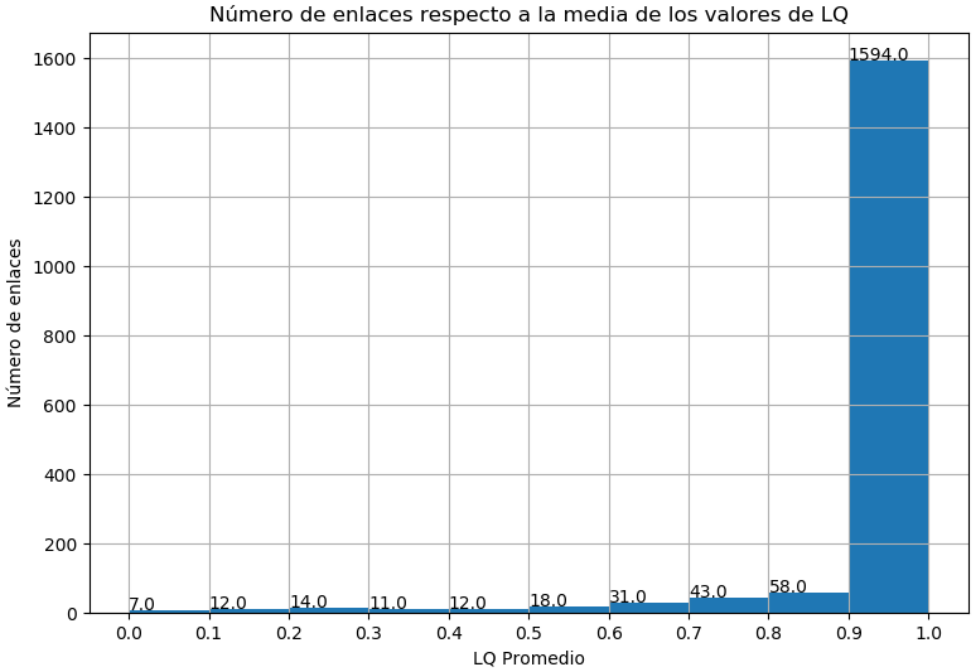

Figura 12: número de enlaces respecto a la media de los valores de LQ.

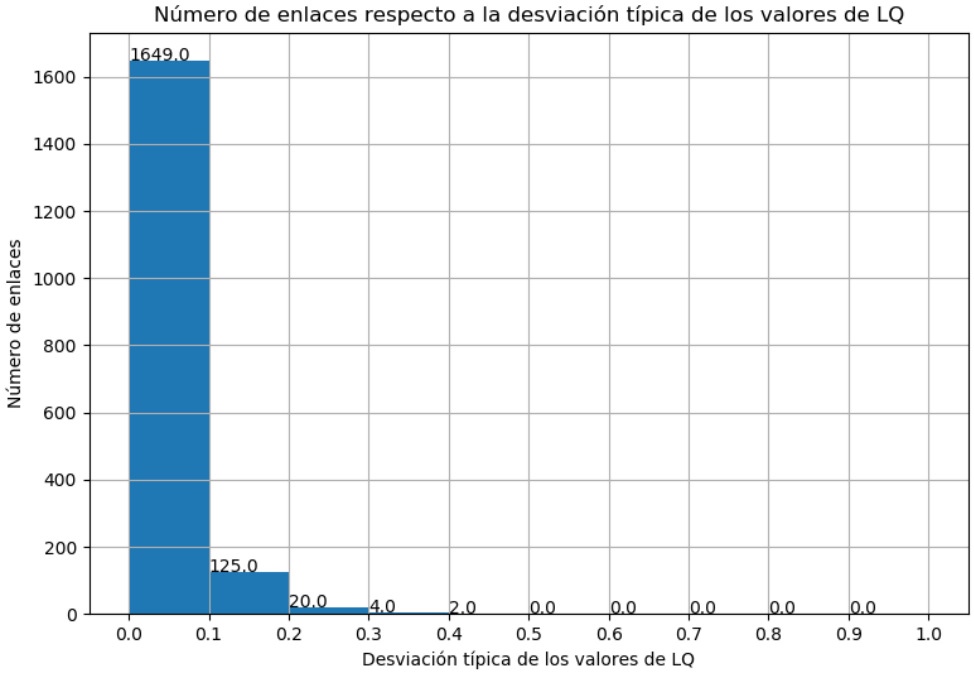

Figura 13: número de enlaces respecto a la desviación típica de valores de LQ.

3.3.2.g. Boxplots *de LQ promedio por día*

La Figura 14 complementa a la Figura 12 y nos muestra la evolución a lo largo de los 14 días del *dataset* de los valores promedio de LQ de los enlaces. El resultado es consistente.

3.3.2.h. Boxplots *de desviación típica de LQ por día*

La Figura 15 complementa a la Figura 13 y nos muestra la evolución a lo largo de los 14 días del *dataset* de la desviación típica de LQ de los enlaces. El resultado es consistente.

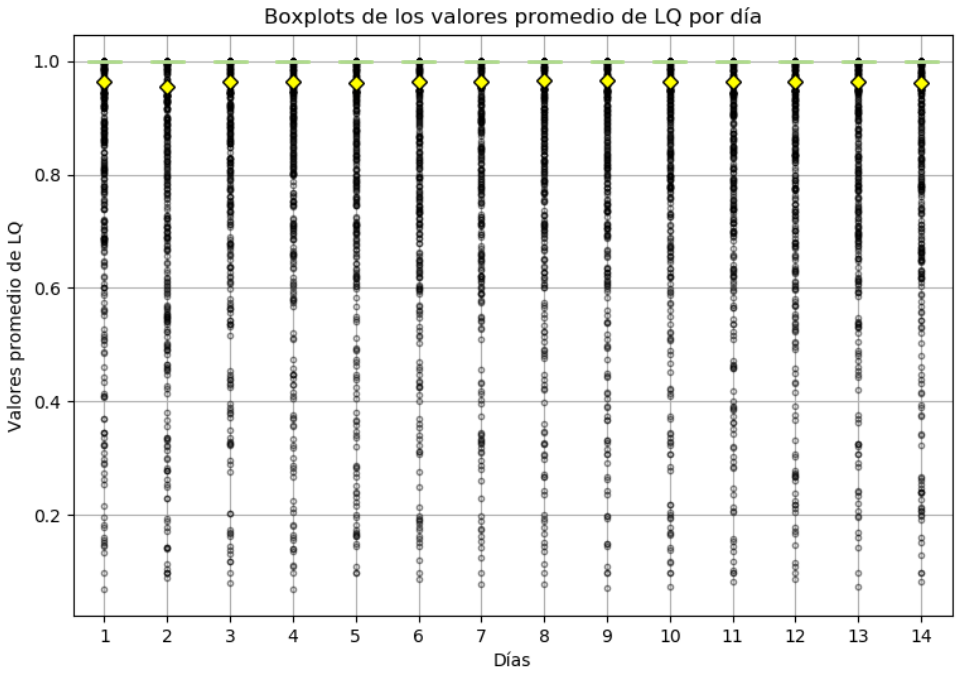

Figura 14: Boxplots *de LQ promedio por día.*

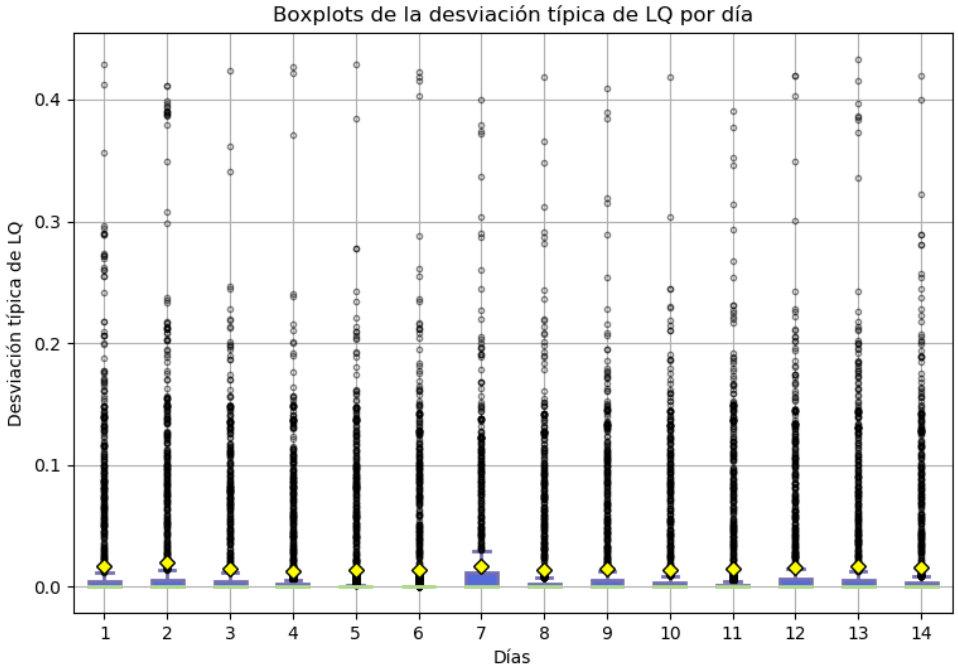

Figura 15: Boxplots *de desviación típica de LQ por día.*

3.3.2.i. Conclusiones

Como conclusiones, obtenemos que la mayoría de los enlaces están activos la mayor parte del tiempo, con valores de LQ promedio elevados y de desviación típica de la misma pequeños. Además, hay un pequeño conjunto de enlaces con baja actividad.

3.4 Criterios para segmentar el análisis de la red

A la hora de dividir el análisis de la red, podemos tener en consideración varios aspectos. En primer lugar, parece interesante realizar un análisis horario en lugar de únicamente realizar un análisis día a día. Por otra parte, puede resultar de utilidad de cara a las predicciones únicamente analizar los enlaces que presenten variaciones, ya que si no puede que la información relevante para nuestro problema quede enmascarada con estos enlaces no variables. En primer lugar, realizaremos el análisis horario, ya que se extiende de manera natural respecto al análisis diario realizado anteriormente. Después, realizaremos una selección de enlaces variables que nos otorguen más información sobre la red, que serán los enlaces sobre los que posteriormente se realizarán las predicciones.

3.5 Análisis pormenorizado de la red – Análisis horario

Se ha realizado un análisis horario de los 14 días del *dataset*. Los resultados son similares al estudio diario, por lo que, por no sobrecargar de información, estas Figuras se muestran en el Anexo B (B.1. Figuras del análisis horario del primer día). Además, en el mismo Anexo B (B.2. Comparativa de desviación típica de LQ a lo largo de los 14 días del *dataset* en los enlaces variables cada día de este) se compara con *boxplots* la desviación típica cada hora los 14 días del *dataset*, no viéndose ninguna correlación importante horaria/semanal.

3.6 Análisis pormenorizado de la red – Selección de enlaces

Pese a que el análisis realizado anteriormente es completo, debido a la alta estabilidad de los enlaces, tanto en valor alto de LQ con poca desviación, como en tiempo de actividad, este análisis puede estar ocultando aspectos interesantes del *dataset*. Además, utilizar este *dataset* para las predicciones puede resultar contraproducente, ya que la alta estabilidad de estos perjudicaría a los algoritmos frente al *baseline*, cuando realmente en esos enlaces no tendría sentido la predicción. Así pues, hemos seleccionado los enlaces que tienen variaciones dentro de cada día.

3.6.1. Representación geográfica de la red

Una vez hecha la selección, obtenemos el resultado observable en Tabla 2. Como se puede apreciar, es un número sensiblemente menor al total.

Día	**Número de enlaces seleccionados**
13 de nov de 2017	554
14 de nov de 2017	784
15 de nov de 2017	725
16 de nov de 2017	500
17 de nov de 2017	449
18 de nov de 2017	429
19 de nov de 2017	795
20 de nov de 2017	522
21 de nov de 2017	604
22 de nov de 2017	609
23 de nov de 2017	482
24 de nov de 2017	589
25 de nov de 2017	515
26 de nov de 2017	530
TOTAL	**1172**

Tabla 2: número de enlaces seleccionados en cada día del dataset y número de enlaces totales seleccionados.

Una vez hecha esta selección, procedemos a realizar la representación geográfica de los mismos para ver si existe algún tipo de distribución especial de interés, lo que vemos en la Figura 16. Debido a la alta densidad de enlaces, vamos a realizar una nueva representación de los enlaces eliminados, lo que se ve en la Figura 17. Pese a que no lo parezca, hay 628 enlaces representados. Las conclusiones que se pueden sacar de esto son las siguientes: la mayoría de los enlaces eliminados son cortos, lo que tiene sentido que sean estables. Solo hay 3 enlaces largos, y 2 son ficticios (recordemos que colocamos estos nodos sin localización en la esquina para tenerlos identificados). El otro, dado que une los dos principales núcleos de nodos, suponemos que sea un enlace de altas prestaciones para no romper esta conexión. No extraemos ninguna conclusión adicional que pueda ser de utilidad.

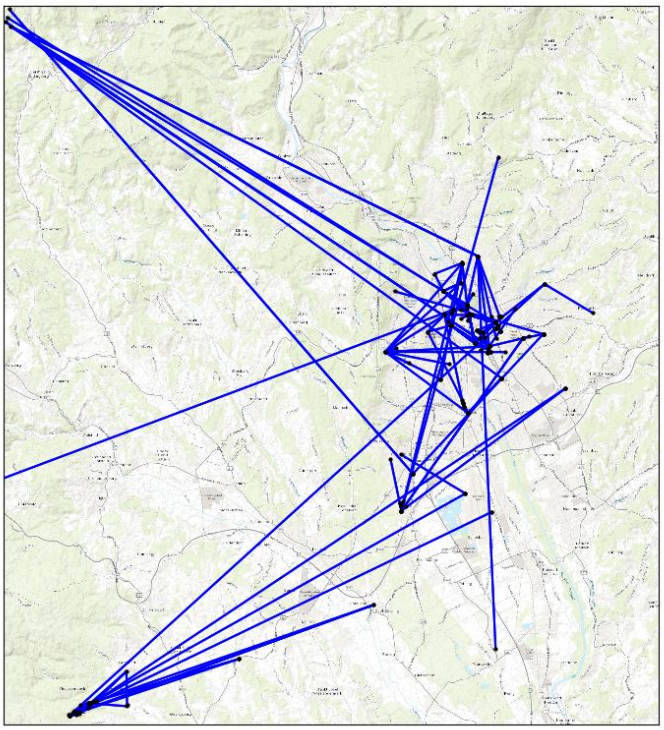

Figura 16: selección de enlaces.

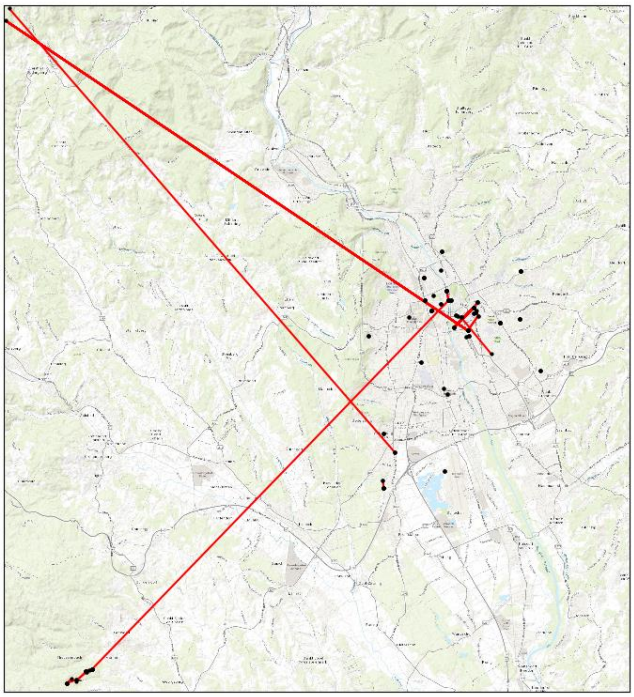

Figura 17: enlaces eliminados en la selección.

3.6.2. Análisis descriptivo de la red

Una vez realizado este análisis, semejante al de la sección 3.3.1 obtenemos resultados en la misma línea de los anteriormente mostrados, solo que aumentando

la importancia de las variaciones. Por no saturar el documento, solo se van a representar las más interesantes del primer día del *dataset*. En las Figura 18 y Figura 19 se observa el promedio de LQ y su desviación típica por hora. Se aprecia un ligero aumento en las horas de la tarde, por lo que se analizó este comportamiento a lo largo de las 2 semanas del *dataset*, lo cual hemos colocado en el Anexo B (B.2. Comparativa de desviación típica de LQ a lo largo de los 14 días del *dataset* en los enlaces variables cada día de este). Pese a que en este día si se apreciaba este aumento, no se sostiene a lo largo del resto de días, por lo que descartamos una periodicidad diaria. Además, tampoco se aprecia en esa imagen (Figura 42) alguna periodicidad semanal.

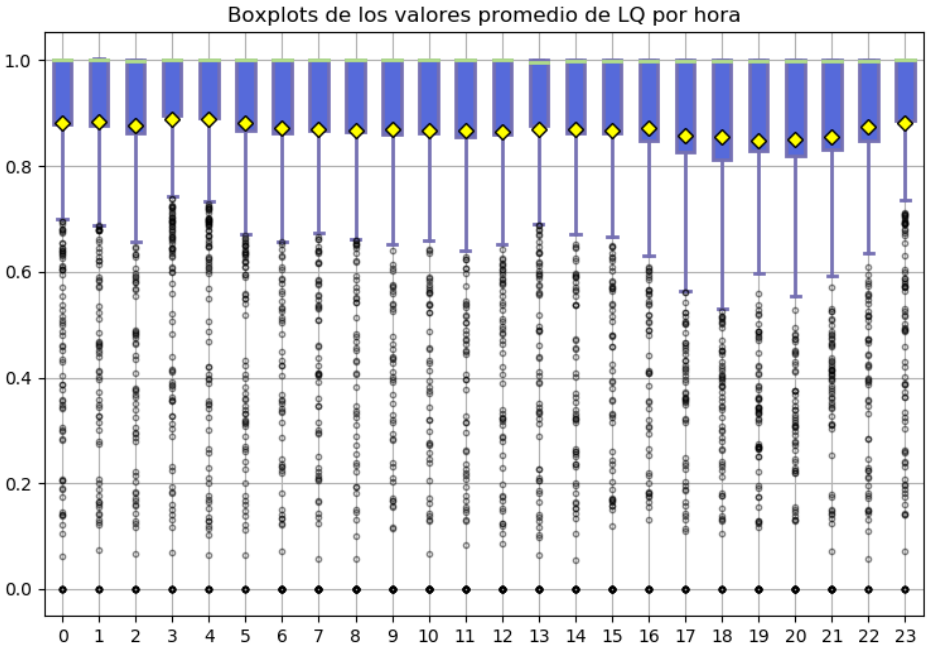

Figura 18: boxplots *de los valores promedio de LQ por hora en los enlaces seleccionados del primer día del* dataset.

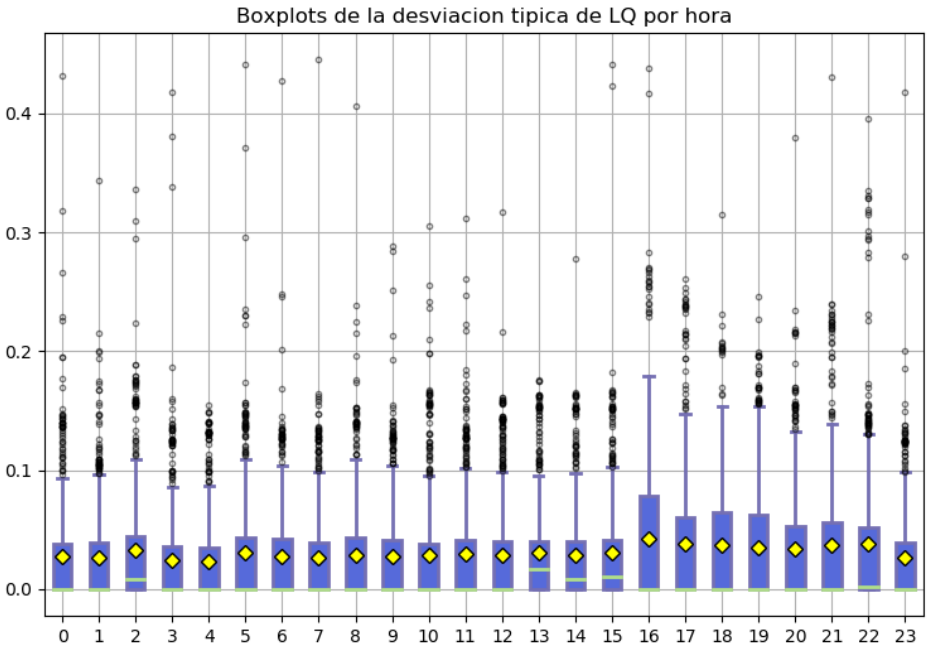

Figura 19: boxplots *de la desviación típica de LQ por hora en los enlaces seleccionados del primer día del dataset.*

3.7 Fijación de un periodo de muestreo óptimo

En esta sección se va a escoger un valor óptimo de muestreo para realizar los análisis posteriores. A priori, podría parecer que con haber muestreado cada segundo es ya suficiente. No obstante, en este muestreo no se tiene en cuenta cada cuando tiempo suele variar el valor de LQ. Si este tiempo fuese muy elevado, estaríamos muestreando largos periodos de tiempo el mismo valor, y a la hora de analizar la precisión de los predictores podría ser engañoso. Además, sería un gasto computacional más elevado de lo necesario, tanto en el caso de nuestro análisis como en el de una implementación real.

Para ello, en primer lugar, vamos a realizar un análisis frecuencial de los valores que va tomando LQ a lo largo del tiempo. Antes de nada, se presentarán la herramienta que se va a utilizar y sus bases teóricas. A continuación, se analizará de manera global el espectro de LQ, y finalmente se dividirá el análisis en *clusters* en función de la desviación típica de la LQ de cada enlace.

Por otra parte, se compararán estos resultados con los resultados esperables en función de los temporizadores del protocolo OLSR utilizados.

3.7.1 Familia de Transformadas de Fourier

La Transformada continua de Fourier (FT, *Fourier Transform*) es una operación matemática que descompone una función dependiente del tiempo (o señal) en las frecuencias que la forman; es decir, permite la representación de la información de una señal en el dominio de la frecuencia. Estas ecuaciones transforman un dominio temporal continuo en uno frecuencial también continuo. No obstante, nuestro dominio temporal es discreto, ya que tenemos capturas de datos cada 1 segundo. Por lo tanto, esta versión no es válida.

La Transformada de Fourier de Tiempo Discreto (DTFT, *Discrete-Time Fourier Transform*) es una versión de la Transformada de Fourier que se aplica a secuencias de valores. En este caso sí que nos valdría esta transformada, debido a que nuestra señal es una secuencia de valores. El problema está en que la transformada es una función continua, y no podemos tener este tipo de señales de manera digital. Para ello, se utiliza una discretización de esta transformada.

La Transformada de Fourier Discreta (DFT, *Discrete Fourier Transform*) es una transformada que surge al discretizar la DTFT en el dominio frecuencial. Para calcularla, existen diferentes algoritmos con diferente eficiencia, entre los que destacan los de la familia de la Transformada Rápida de Fourier (FFT, *Fast Fourier Transform*). La mayoría de ellos dependen de la factorización de N, el número de valores de la transformada en frecuencia. Estos algoritmos son especialmente rápidos cuando N es una potencia de 2, por lo que siempre se procurará utilizar un N apropiado.

3.7.2 Uso de la FFT para el análisis frecuencial del valor de LQ durante 1 día

Para analizar las variaciones de la LQ en el dominio de la frecuencia, en primer lugar, tenemos que elegir un valor de N adecuado. Dado que tenemos medidas de LQ tomadas cada 1 segundo, en un día hay como mucho 86400 (teniendo en cuenta que en algunos instantes no hay ninguna muestra capturada y en otros en los que algunos enlaces no están), tenemos que elegir un valor de N común para todos los enlaces, de manera que podamos sumar las transformadas y obtener una visión global de las variaciones. Tal y como se ha dicho en el apartado anterior, si N es una potencia de 2 el algoritmo es mucho más rápido, por lo que escogemos la potencia de 2 inmediatamente superior, que en nuestro caso es 2^{17} (131072). La DFT transforma una señal de tamaño N en una transformada al dominio de la frecuencia de también tamaño N, por lo que tenemos que rellenar con ceros hasta obtener el valor de N elegido (esto se conoce como *zero-padding* y equivale a una

interpolación en el dominio frecuencial). Esta decisión tiene algunas implicaciones, pero en nuestro caso no parece que nos vaya a afectar (solo queremos ver si hay alguna frecuencia en la que haya claramente más energía que en otras).

En la Figura 20 podemos ver la FFT de la LQ agregada para todos los enlaces variables del primer día del *dataset* con el eje de ordenadas en unidades logarítmicas (el valor concreto del eje y no nos importa ahora). Como podemos apreciar, la transformada tiene mayor valor a frecuencias bajas, y rápidamente disminuye a medida que aumenta la frecuencia. Esto nos indica que la mayor parte de los enlaces varían cada mucho tiempo, y los cambios cada menos tiempo son menos frecuentes. No obstante, de cara a elegir una frecuencia óptima de muestreo no es suficiente, ya que no hay ningún valor relativamente alto de frecuencia que destaque sobre el resto. Por ello, es preciso realizar un análisis pormenorizado que nos permita encontrar algún valor frecuencial que destaque. En concreto, vamos a realizar un análisis frecuencial dividido por *clusters* en función de la desviación típica.

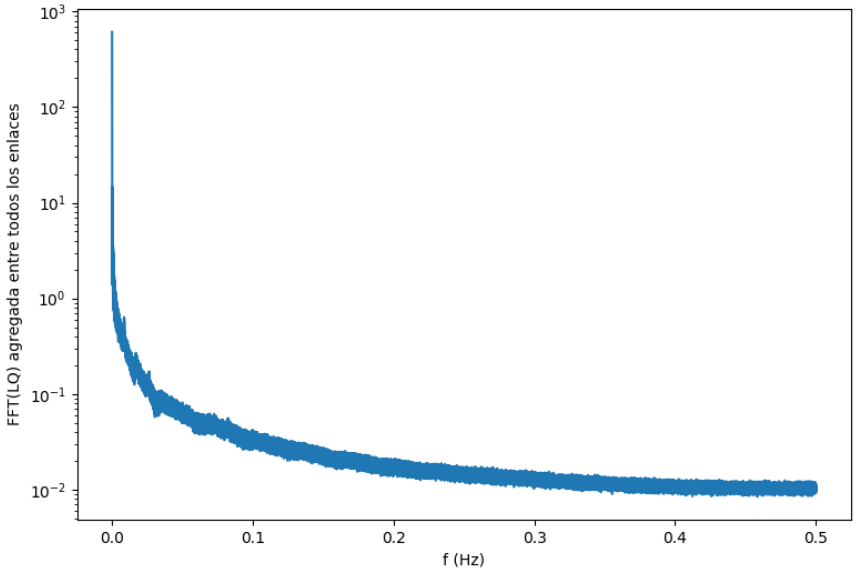

Figura 20: FFT de la LQ agregada para todos los enlaces variables del primer día del dataset.

3.7.3 Segmentación en *clusters* en función de la desviación típica de cada enlace

Tal y como se dijo en el apartado anterior, se va a realizar una desagregación de los enlaces en función de su desviación típica. En concreto, se van a realizar 4 *clusters* del total de los enlaces del primer día (1719), separados por los percentiles 25, 50 (mediana) y 75. Posteriormente, se realizarán una desagregación más fina en la parte donde la desviación típica es mayor. Después de realizar dicho experimento,

dado que la mayor parte de los enlaces del *dataset* durante el primer día tienen una desviación típica nula, solo vamos a mostrar la FFT de los *clusters* entre la mediana y el percentil 75 y los percentiles 75 y 100, que se pueden ver en la Figura 21 y Figura 22. A la vista de la Figura 22, nos interesaría ver con más detalle el último *cluster*, así que realizamos otros cinco, entre los percentiles 75, 80, 85, 90,95 y 100. Solo vamos a mostrar los que hemos obtenido resultados interesantes (80-85 y 85-90)

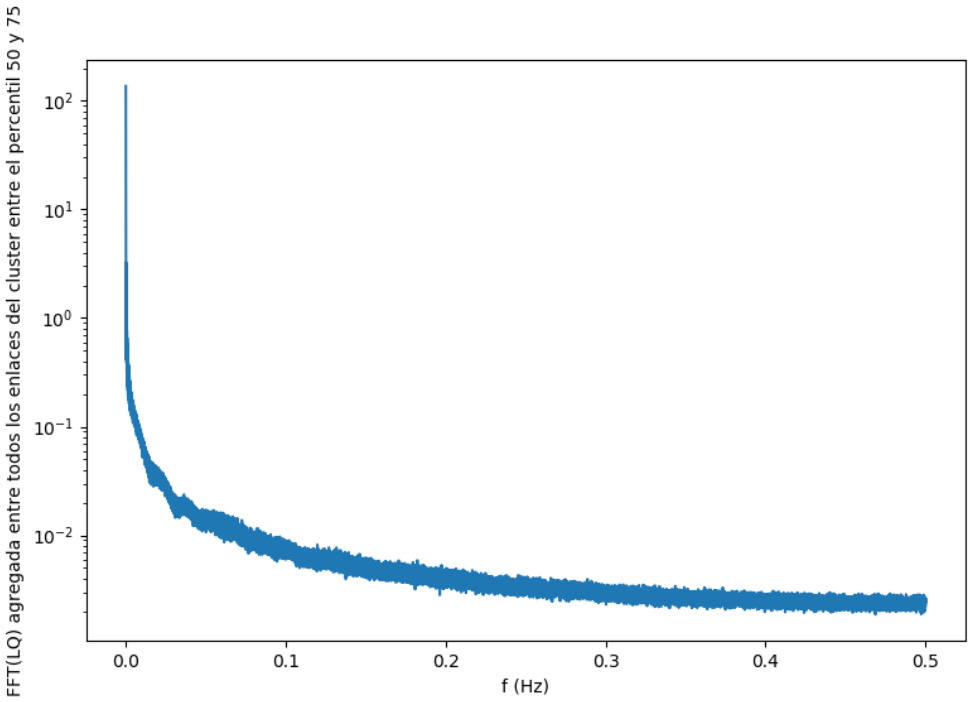

Figura 21: FFT de la LQ agregada para todos los enlaces variables del primer día del dataset *que están entre los percentiles 50 y 75 respecto a su desviación típica.*

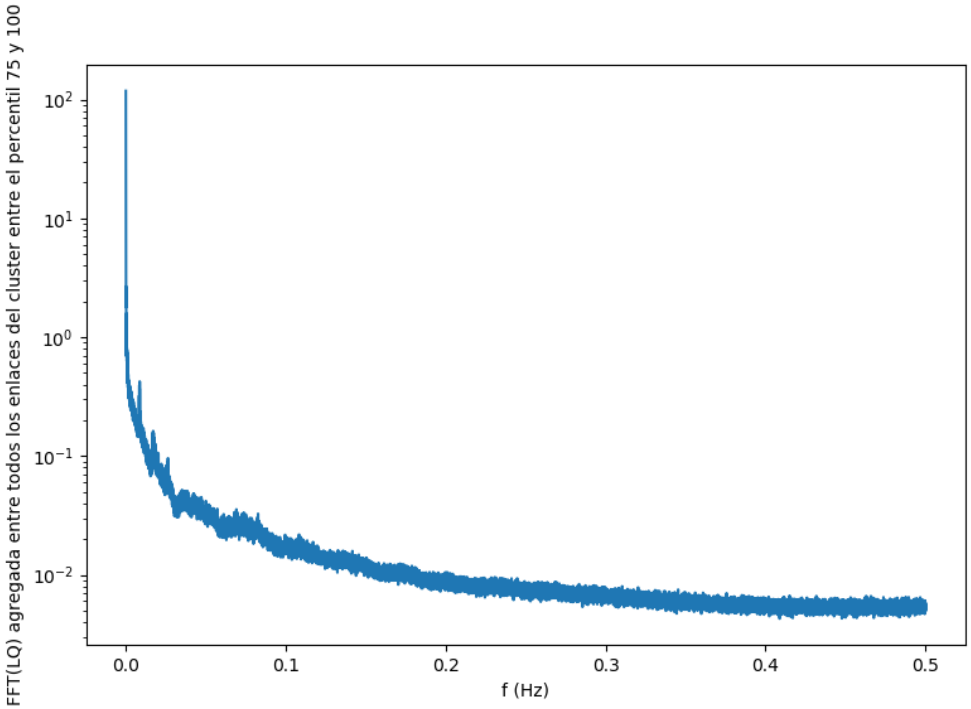

Figura 22: FFT de la LQ agregada para todos los enlaces variables del primer día del dataset *que están entre los percentiles 75 y 100 respecto a su desviación típica.*

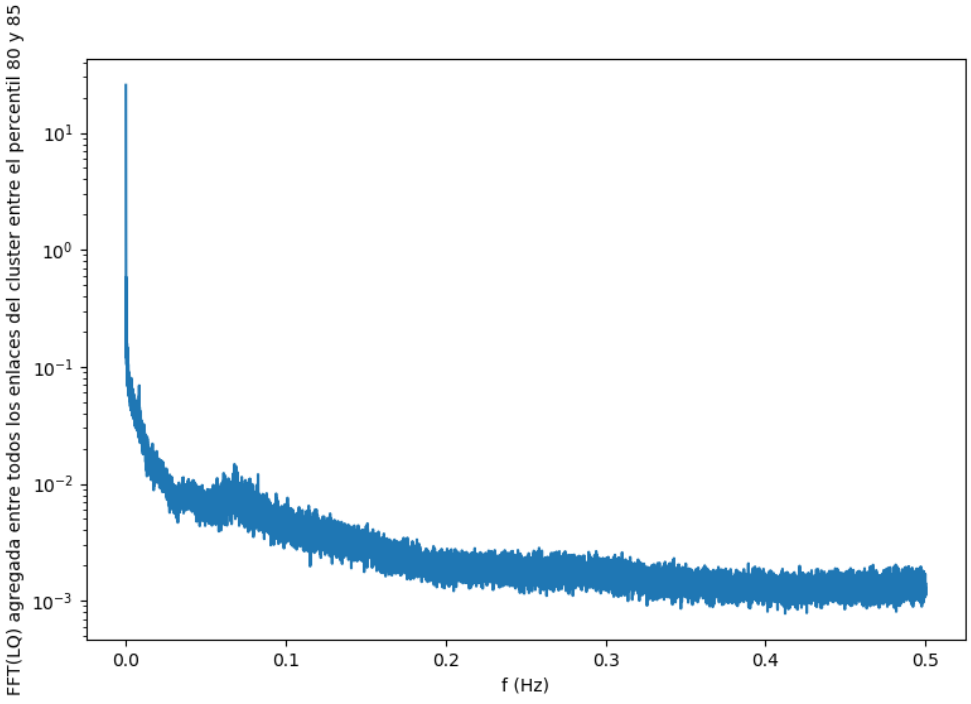

Figura 23: FFT de la LQ agregada para todos los enlaces variables del primer día del dataset que están entre los percentiles 80 y 85 respecto a su desviación típica.

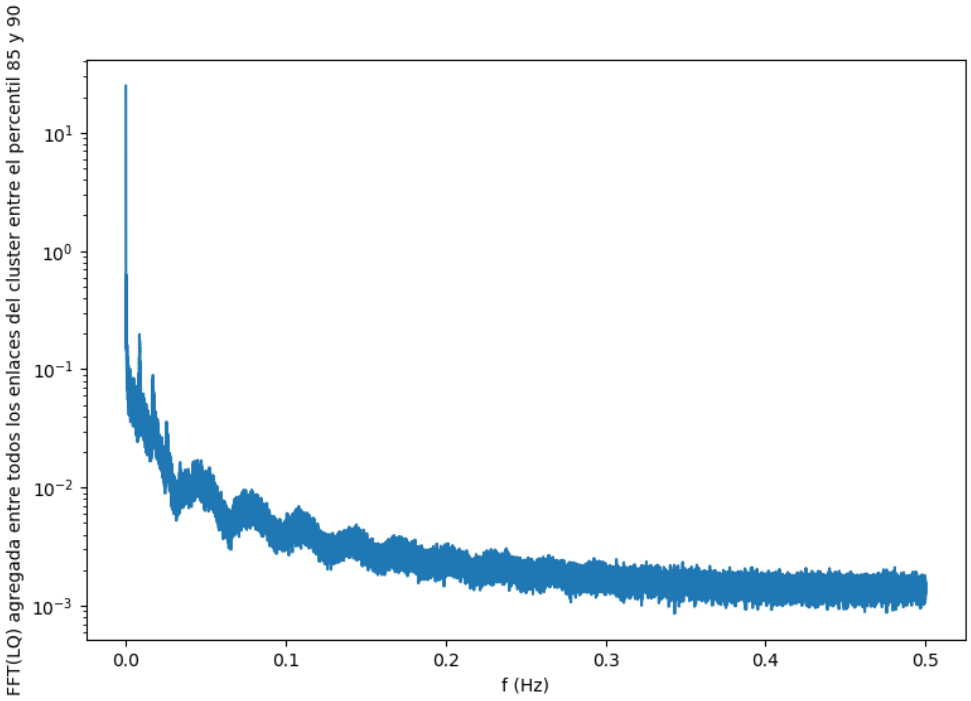

Figura 24: FFT de la LQ agregada para todos los enlaces variables del primer día del dataset *que están entre los percentiles 85 y 90 respecto a su desviación típica.*

La Figura 23 y la Figura 24 son las más interesantes hasta ahora en lo que respecta al análisis frecuencial. En la primera se puede ver cómo se intuye un máximo en f ≈ 0.066 Hz → T ≈ 15 s. La segunda nos da más información, y observamos picos en: en f ≈ 0.04 Hz → T ≈ 25 s; en f ≈ 0.08 Hz → T ≈ 12.5 s; en f ≈ 0.11 Hz → T ≈ 9 s; en f ≈ 0.14 Hz → T ≈ 7 s; y en f ≈ 0.175 Hz → T ≈ 5 s. Este último valor es el que vamos a utilizar a partir de ahora para realizar los experimentos, pues consideramos que es una frecuencia de muestreo suficientemente alta sin llegar a ser cada 1 segundo como en el *dataset*, y además tiene más sentido desde el punto de vista telemático, ya que los temporizadores de los mensajes HELLO y TC están configurados por defecto cada 3 y 2 segundos, respectivamente.

3.8 Conclusiones

En este capítulo se ha realizado un análisis del *dataset* de forma previa a la ejecución de experimentos. Hemos identificado que la mayor parte de enlaces se concentran en la ciudad de Graz y algún municipio cercano, así como que la mayor parte de los enlaces no variables son cortos. La mayoría de los enlaces están activos casi todo el tiempo con alta promedio de LQ y baja desviación típica, habiendo un 4% de enlaces activos menos de un 10% del tiempo, los cuales podrían ser muy difíciles de predecir. El comportamiento diario es equivalente al horario, no habiendo una periodicidad diaria ni semanal. Se han identificado diferentes periodos utilizando la FFT en los enlaces más variables, que a priori son los que más se beneficiarían de una predicción, por lo que se ha tomado este periodo (5 segundos) como periodo general de muestreo para los experimentos posteriores.

Capítulo 4. Predicciones de la calidad del enlace con algoritmos de *Online Machine Learning*

4.1. Introducción

En el capítulo anterior se ha descrito el conjunto de datos utilizado. En este capítulo, en primer lugar, se explicarán las consideraciones iniciales que se han tenido en cuenta a la hora de realizar los experimentos, así como las herramientas y técnicas utilizadas. Posteriormente, se realizarán unos primeros experimentos con los parámetros por defecto utilizando el primer día del conjunto de datos, para cuyo análisis se utilizarán los *clusters* empleados en el capítulo previo. Después, se intentarán optimizar los resultados de cada uno de los algoritmos utilizados de manera individual, para después repetir el experimento inicial con los nuevos parámetros. A continuación, se realizarán otro tipo de experimentos, de cara a realizar una predicción a medio plazo. Finalmente, se extraerán unas breves conclusiones.

4.2. Consideraciones iniciales, herramientas y técnicas

Para la realización de los experimentos han utilizado los valores de LQ de los enlaces variables el primer día del *dataset* (554 enlaces) remuestreados cada 5 segundos. No obstante, a la hora de realizar los experimentos vimos que de 3 enlaces había muy pocos datos, por lo que no se podían realizar con éxito todos los experimentos, así que fueron eliminados. Esto es debido a que no había suficientes datos de LQ de estos enlaces como para construir como mínimo un vector de entrenamiento y uno de test para algunas de las ventanas temporales utilizadas.

Por otra parte, en los instantes en los que un enlace está apagado, se ha optado por no asignar ningún valor a la LQ del enlace, y continuar con los valores de LQ de la próxima conexión, a diferencia de lo realizado en [13], donde se asignaba un valor igual a 0. Esto simplificaba el problema, pero en enlaces con pocos datos las predicciones son muy buenas, debido a que se predice 0, y los resultados quedan mejorados artificialmente. En un principio, se siguió este enfoque dado que [7] daba a entender que era así como realizaron sus experimentos. En [14] ya se siguió esta nueva aproximación, que es más realista y no maquilla los resultados. Lo más adecuado sería una combinación de esta solución con la predicción de encendido/apagado de los enlaces (*temporal link predition*), como la que se presenta en [10].

Para los experimentos se ha utilizado la versión de Java 8, actualización 121 y la versión 2018.6.0 de MOA. Se han empleado 5 algoritmos de aprendizaje automático en línea, a saber:

- *Perceptron*
- *FIMT-DD*
- *ORTO*
- *AMRulesRegressor*
- *FadingTargetMean*

La descripción del funcionamiento de estos algoritmos, así como sus parámetros y sus valores por defecto, queda recogida en el Anexo C.

Para realizar los experimentos, se ha construido un predictor por cada enlace, siguiendo la filosofía de *test-then-train*, es decir, por cada nueva instancia del flujo de datos, en primer lugar, se testea el nuevo valor con el que predecía el algoritmo, y a continuación se actualiza el predictor con este valor. Por lo tanto, a partir de la segunda instancia ya se han empezado a generar datos de error. En la Figura 25 se describe de manera esquemática lo anteriormente explicado.

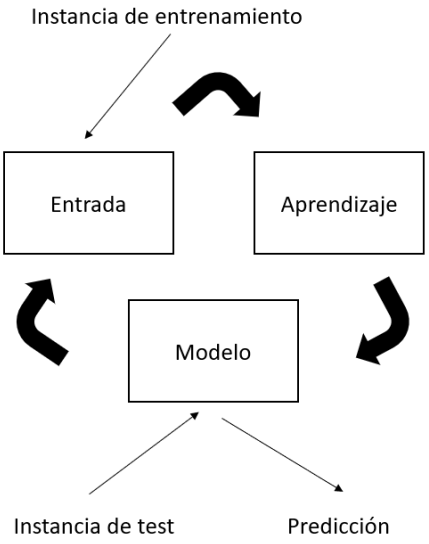

Figura 25: explicación de la filosofía test-then-train. Elaboración propia a partir de [53] .

Para los experimentos iniciales se ha utilizado una ventana temporal (*lag window*), que se ha ido variando desde 1 a 12 unidades. La ventana temporal es el número de valores pasados que constituyen el vector de entrada de la serie temporal. La predicción se ha realizado a 1 instante vista, o sea, por cada vector de tamaño *lag* que se le introduce al predictor, nos proporciona un valor de salida que es la predicción para el instante siguiente. La Figura 26 aclara esta explicación.

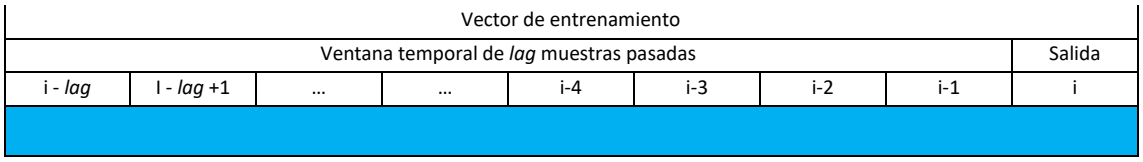

Figura 26: vector de entrenamiento para los experimentos.

Los resultados obtenidos se van a mostrar de dos formas. Por una parte, obtenidos los valores de MAE (*Mean Absolute Error*, error medio absoluto) de cada uno de los enlaces, vamos a realizar el MAE promedio por cada combinación de algoritmo y ventana utilizada, que se representará en forma de tabla, por no saturar de gráficas el documento.

A la hora de analizar los resultados, es necesario un test estadístico para verificar que los resultados son estadísticamente significativos. En [13] se discutió sobre el tema, y se llegó a la conclusión de que se puede aplicar el Z-test de acuerdo a lo descrito en [54], debido a que las muestras (el resultado de cada algoritmo, es decir, el conjunto de MAEs de todos los enlaces) estén distribuidas de manera Normal (Gaussiana). De acuerdo al teorema del límite central, la suma de variables aleatorias independientes tienden a tener una distribución Gaussiana, aunque las variables en sí mismas (en este caso, el resultado de un algoritmo en un conjunto de enlaces) no se distribuyan de manera Gaussiana (en [13] se determinaba que las funciones de densidad de probabilidad de los errores de cada enlace no eran Gaussianas, sino algo similar a exponenciales). Además, para poder aplicar el Z-test, es necesario poder obtener la media y la varianza de cada una de las muestras de manera precisa. Como las muestras son grandes (muchos enlaces), esto es viable.

4.3. Primeros experimentos

Estos experimentos se han realizado utilizando los parámetros por defecto de todos los algoritmos. Además, se ha añadido un algoritmo adicional de *baseline*, para poner en perspectiva los resultados con un algoritmo básico. El *baseline* se definió previamente en [13] y [14] como un algoritmo que predice simplemente el último valor recibido. En este caso, se ha añadido la componente de ventana

temporal, haciendo el promedio de los *lag* últimos valores (tomando como referencia la Figura 26, el promedio del valor de las posiciones desde la i-ésima hasta la (i-*lag*+1)-ésima.

En la Tabla 3 se muestran los resultados para los 5 algoritmos y para el *baseline* de manera agregada para todos los enlaces, utilizando los 12 valores de ventana temporal. En negrita se marca el mejor resultado de cada algoritmo y subrayado en verde el mejor en general. A simple vista ya se aprecia que ninguno de los algoritmos supera al *baseline*. No obstante, realizamos un Z-test entre el mejor resultado del *baseline* y el mejor de cada uno del resto de algoritmos. Para una significatividad estadística del 5%, obtenemos que se puede rechazar la hipótesis nula, y, por tanto, las medias de MAE de *baseline* son distintas (obtenemos un p-valor de 0.00 para todos los casos). Por otra parte, se puede ver que todos los algoritmos excepto *Fading Target Mean* obtienen sus mejores valores con ventanas pequeñas. Además, ORTO parece funcionar extremadamente mal, lo cual es raro ya que es una mejora de FIMTDD. Esto se tratará en el apartado de optimización.

Lag/ Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.01187	0.02281	**0.01309**	**0.02439**	0.01222	**0.00641**
Lag 2	0.01094	0.02281	0.01665	0.10071	0.01257	0.00833
Lag 3	**0.01093**	0.02281	0.02452	0.11554	0.01215	0.00988
Lag 4	0.01101	0.02280	0.03348	0.11305	**0.01178**	0.01126
Lag 5	0.01101	0.02281	0.03649	0.12153	0.01353	0.01247
Lag 6	0.01117	0.02281	0.06528	0.12181	0.01204	0.01353
Lag 7	0.01146	0.02282	0.06932	0.12707	0.01187	0.01443
Lag 8	0.01162	0.02281	0.06938	0.12695	0.01211	0.01517
Lag 9	0.01156	0.02275	0.07670	0.13236	0.01214	0.01584
Lag 10	0.01158	0.02272	0.07411	0.12897	0.01236	0.01644
Lag 11	0.01180	**0.02271**	0.08116	0.12974	0.01265	0.01698
Lag 12	0.01209	0.02272	0.07749	0.12916	0.01292	0.01745

Tabla 3: resultados iniciales de los experimentos. 5 algoritmos + baseline *sobre todos los enlaces variables en el primer día, con ventana temporal de 1 a 12 unidades.*

Una vez analizado el comportamiento global, se ha analizado el comportamiento utilizando la misma división en *clusters* en los que se dividió el análisis de la FFT, añadiendo un *cluster* adicional entre los percentiles 99 y 100 (6 enlaces). En ninguno de ellos se obtienen mejores valores que los del *baseline*, por lo que no se muestran aquí los resultados y quedan recogidos en el Anexo D (D.1. Resultados iniciales divididos por *clusters*).

4.4. Optimización de algoritmos

A la vista de los resultados de la sección anterior, a partir de ahora se trabajará con ventanas de 1 a 6 unidades. Se va a proceder a intentar optimizar algunos de los parámetros de los algoritmos de aprendizaje automático en línea utilizados. Se probará con aquellos parámetros cuyo entendimiento sea sencillo, ya que muchos de ellos requerirían un conocimiento demasiado exhaustivo de los algoritmos para el ámbito de este trabajo. Únicamente se mostrarán los resultados globales, sin dividirlos en *clusters*. En la sección siguiente, con los algoritmos con los parámetros finales ya sí que se detallará este análisis.

4.4.1 Optimización de *Perceptron*

De este algoritmo nos puede interesar optimizar el valor de su factor de aprendizaje (`learningRatio`). El resto de los parámetros son relativos también al factor de aprendizaje, pero el más importante es el que vamos a explorar. Un valor demasiado bajo puede hacer que no se llegue nunca o se tarde mucho en llegar al mínimo de la función de coste. A su vez, un valor demasiado alto tampoco permitiría llegar al mínimo, ya que el algoritmo de descenso de gradiente "se pasaría" el mínimo. El valor por defecto es 0.025, por lo que vamos a probar varios valores 0.01, 0.05, 0.1, 0.5, 1 y 5. A la vista de los resultados en la Tabla 12, ningún valor del parámetro mejora el comportamiento del *baseline*. El valor con el que mejor resultado obtenemos es 1, y queda demostrado lo explicado anteriormente. Valores más pequeños (0.01, 0.025, 0.05, 0.1 y 0.5), así como valores mayores (5) al óptimo obtienen peores resultados.

Lag*\ *Parámetro	**0.01**	**0.025**	**0.05**	**0.1**	**0.5**	**1**	**5**	**Baseline**
Lag 1	0.0127	0.0122	0.0121	0.0120	0.0114	**0.0110**	**0.0374**	**0.0064**
Lag 2	0.0146	0.0126	0.0119	0.0115	**0.0113**	0.0120	0.0427	0.0083
Lag 3	0.0135	0.0121	**0.0117**	**0.0114**	0.0126	0.0144	0.0480	0.0098
Lag 4	**0.0123**	**0.0118**	0.0117	0.0116	0.0141	0.0170	0.0530	0.0112
Lag 5	0.0165	0.0135	0.0126	0.0123	0.0158	0.0196	0.0562	0.0124
Lag 6	0.0130	0.0120	0.0119	0.0121	0.0176	0.0225	0.0603	0.0135

Tabla 4: resultados de los experimentos con Perceptron *variando su parámetro* `learningRatio`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.2 Optimización de *FIMT-DD*

El algoritmo FIMT-DD tiene una gran cantidad de parámetros, pero para nuestro análisis solo vamos a tener en cuenta los siguientes: `gracePeriod`, `splitConfidence`, `regressionTree` y `learningRatio`. Analizaremos cada uno

de ello por separado y después los combinaremos. Debemos destacar que en general, los parámetros no son independientes, sino que unos afectan a otros. No obstante, hacer un análisis multiparámetro sería demasiado costoso en este contexto, por lo que seguiremos la aproximación mencionada. El mejor resultado aparecerá subrayado en verde, el mejor resultado de los diferentes valores del parámetro, en cian, el valor por defecto del parámetro, en gris, y los mejores resultados para cada experimento aparecerán en negrita.

4.4.2.a `gracePeriod`

Este parámetro indica cuantas instancias una hoja tiene que ver antes de intentos de división. Cuanto mayor sea, menos variaciones tiene el árbol, lo cual puede ser bueno (mayor estabilidad), pero a cambio atrapamos menos los cambios del flujo de datos, es decir, podemos sufrir *underfitting*. Si es demasiado pequeño, puede que hagamos cambios demasiado rápido, y suframos *overfitting*. El valor por defecto es 200, por lo que probaremos con 20, 200, 2000 y 20000 (cabe destacar que 20000 es mayor que la cantidad de valores de LQ con los que hacemos el experimento, por lo que el modelo que se creará será muy sencillo). En la Tabla 5 se muestran los resultados.

Lag*\ *Parámetro	**20**	**200**	**2000**	**20000**	**Baseline**
Lag 1	**0.0141**	**0.0131**	**0.0120**	**0.0119**	**0.0064**
Lag 2	0.0156	0.0166	0.0170	0.0135	0.0083
Lag 3	0.0235	0.0245	0.0331	0.0144	0.0098
Lag 4	0.0313	0.0335	0.0537	0.0146	0.0112
Lag 5	0.0355	0.0365	0.0551	0.0136	0.0124
Lag 6	0.0626	0.0653	0.0704	0.0186	0.0135

Tabla 5: resultados de los experimentos con FIMT-DD variando su parámetro `gracePeriod`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.2.b `splitConfidence`

Este parámetro indica el error permitido en una decisión de división. Cuanto más cercano a 0, más tarda en decidir. El valor por defecto es 1.0E-7, por lo que probaremos con 1.0E-7, 1.0E-5, y 1.0E-3. En la Tabla 6 se muestran los resultados.

Lag\ Parámetro	**1.0E-7**	**1.0E-5**	**1.0E-3**	**Baseline**
Lag 1	**0.0131**	**0.0132**	**0.0132**	**0.0064**
Lag 2	0.0166	0.0159	0.0197	0.0083
Lag 3	0.0245	0.0249	0.0389	0.0098
Lag 4	0.0335	0.0358	0.0505	0.0112
Lag 5	0.0365	0.0406	0.0465	0.0124
Lag 6	0.0653	0.0663	0.0548	0.0135

Tabla 6: resultados de los experimentos con FIMT-DD *variando su parámetro* `splitConfidence`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.2.c `regressionTree`

Este parámetro nos deja construir un árbol de regresión en vez de un árbol de modelo. Por defecto esta desactivado (árbol de modelo), por lo que probaremos un árbol de regresión. En la Tabla 7 se muestran los resultados.

Lag\Algoritmo	**Model Tree**	**Regression Tree**	**Baseline**
Lag 1	**0.0131**	**0.0132**	**0.0064**
Lag 2	0.0166	0.0167	0.0083
Lag 3	0.0245	0.0247	0.0098
Lag 4	0.0335	0.0337	0.0112
Lag 5	0.0365	0.0367	0.0124
Lag 6	0.0653	0.0656	0.0135

Tabla 7: resultados de los experimentos con FIMT-DD *variando su parámetro* `regressionTree`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.2.d `learningRatio`

Este es el factor de aprendizaje utilizado en las hojas para entrenar los perceptrones. El valor por defecto es 0.02, por lo que probaremos con 0.01, 0.02, 0.1, 0.5 (valores de 0 a 1). En la Tabla 8 se muestran los resultados.

Lag*\ *Parámetro	**0.01**	**0.02**	**0.1**	**0.5**	**Baseline**
Lag 1	**0.0124**	**0.0131**	**0.0188**	**0.0272**	**0.0064**
Lag 2	0.0150	0.0166	0.0738	0.1248	0.0083
Lag 3	0.0174	0.0245	0.0681	0.1778	0.0098
Lag 4	0.0222	0.0335	0.0897	0.1598	0.0112
Lag 5	0.0233	0.0365	0.0906	0.1735	0.0124
Lag 6	0.0307	0.0653	0.1011	0.1878	0.0135

Tabla 8: resultados de los experimentos con FIMT-DD *variando su parámetro* `learningRatio`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.2.e Combinación y conclusiones

Una vez hechas las pruebas anteriores, se ha podido constatar que en ningún caso los resultados se acercan a los del *baseline*. Se ha probado a combinar `gracePeriod` = 20000 y `learningRatio` = 0.01, los únicos valores que no aparecían por defecto, pero el resultado ha empeorado. Por lo tanto, el mejor resultado es 0.0119 con `gracePeriod` = 20000 frente al 0.0064 del *baseline*.

4.4.3 Optimización de *ORTO*

Para optimizar ORTO en primer lugar vamos a optimizar los algoritmos propios de esta mejora de FIMT-DD, ya que obtenía resultados mucho peores. A continuación, utilizaremos la optimización realizada en FIMT-DD para ver si los resultados mejoran.

4.4.3.a `maxTrees`

Este parámetro indica el número máximo de árboles en el árbol de opción. En principio, un valor muy alto produciría *overfitting* y uno muy bajo *underfitting*. El valor por defecto es 10, por lo que probaremos con 1, 5 y 10. Como se ve en la Tabla 9, para *lag* = 1, el resultado es prácticamente idéntico en todos los casos. Para *lags* mayores, se comporta mejor cuanto menos vale, pero no nos interesa porque es peor que el *baseline*.

Lag*\ *Parámetro	**1**	**5**	**10**	**Baseline**
Lag 1	**0.0245**	**0.0245**	**0.0244**	**0.0064**
Lag 2	0.0210	0.0958	0.1007	0.0083
Lag 3	0.0241	0.0969	0.1155	0.0098
Lag 4	0.0273	0.0997	0.1130	0.0112
Lag 5	0.0290	0.1030	0.1215	0.0124
Lag 6	0.0311	0.1093	0.1218	0.0135

Tabla 9: resultados de los experimentos con ORTO *variando su parámetro* `maxTrees`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.3.b `maxOptionLevel`

Este parámetro indica la profundidad máxima a la que los nodos de opción pueden ser creados. El valor por defecto es 10, por lo que probaremos con 1, 5 y 10. Como se ve en la Tabla 10 para *lag* = 1, el resultado es prácticamente idéntico en todos los casos. Para *lags* mayores, se comporta mejor cuanto más vale, pero no nos interesa porque es peor que el *baseline*.

Lag*\ *Parámetro	**1**	**5**	**10**	**Baseline**
Lag 1	**0.0245**	**0.0245**	**0.0244**	**0.0064**
Lag 2	0.0930	0.0986	0.1007	0.0083
Lag 3	0.1209	0.1158	0.1155	0.0098
Lag 4	0.1243	0.1149	0.1130	0.0112
Lag 5	0.1238	0.1219	0.1215	0.0124
Lag 6	0.1250	0.1233	0.1218	0.0135

Tabla 10: resultados de los experimentos con ORTO *variando su parámetro* `maxOptionLevel`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.3.b `gracePeriod`

Este es el mismo parámetro que el de FIMT-DD, y podemos ver que ocurre igual que ese algoritmo. Con este parámetro el modelo se vuelve muy sencillo y es lo

único con lo que conseguimos disminuir el error, pero obtenemos el mismo valor de MAE que en FIMT-DD y no superamos al *baseline*.

Lag\ Parámetro	**200**	**20000**	**Baseline**
Lag 1	0.0244	0.0120	0.0064
Lag 2	0.1007	0.0135	0.0083
Lag 3	0.1155	0.0144	0.0098
Lag 4	0.1130	0.0146	0.0112
Lag 5	0.1215	0.0136	0.0124
Lag 6	0.1218	0.0186	0.0135

Tabla 11: resultados de los experimentos con ORTO *variando su parámetro* `gracePeriod`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.4 Optimización de *FadingTargetMean*

Este algoritmo es el más sencillo de optimizar, ya que únicamente tiene un parámetro que afecte a las predicciones, `FadingFactorOption`, que controla cuanta información del pasado se tiene en cuenta a la hora de realizar la media de los valores. Es decir, pondera el pasado más (valor 1 todas las medidas pasadas tienen la misma importancia que la nueva) o menos (valor 0 todas las medidas pasadas son rechazadas). El valor por defecto es 0.99, por lo que vamos a probar varios valores: 1, 0, 0.9, 0.5, 0.1. A la vista de los resultados en la Tabla 12, ningún valor del parámetro mejora el comportamiento del *baseline*. Solo cuando este valor es 0 o cercano, se obtienen resultados similares, dado que esto significa que los valores pasados tienen menos influencia y el algoritmo es equivalente al *baseline*.

Lag\ Parámetro	**1**	**0.99**	**0.9**	**0.5**	**0.1**	**0**	**Baseline**
Lag 1	0.03333	0.02293	**0.01691**	**0.00924**	**0.00677**	**0.00641**	**0.00641**
Lag 2	0.03332	0.02293	0.01691	0.00924	0.00677	0.00641	0.00833
Lag 3	0.03332	0.02293	0.01692	0.00924	0.00678	0.00641	0.00988
Lag 4	**0.03331**	**0.02292**	0.01692	0.00925	0.00678	0.00641	0.01126
Lag 5	0.03331	0.02293	0.01693	0.00925	0.00678	0.00642	0.01247
Lag 6	0.03331	0.02294	0.01693	0.00926	0.00679	0.00642	0.01353

Tabla 12: resultados de los experimentos con FadingTargetMean *variando su parámetro* `FadingFactorOption`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.5 Optimización de *AMRulesRegressor*

Para este algoritmo, en primer lugar, vamos a optimizar dos parámetros (`splitConfidence` y `gracePeriod`) que ya hemos visto en los algoritmos

anteriores. Después probaremos si utilizar reglas sin orden mejora el resultado (`setUnorderedRulessetOn`). Finalmente, probaremos a cambiar uno de los algoritmos de aprendizaje internos (`learnerOption`)

4.4.5.a `splitConfidence`

Al igual que en el resto de los algoritmos, este parámetro indica el error permitido en una decisión de división. Cuanto más cercano a 0, más tarda en decidir. El valor por defecto es 1.0E-7, por lo que probaremos con 1.0E-7, 1.0E-5, y 1.0E-3. En la Tabla 13 se muestran los resultados.

Lag\ Parámetro	**1.0E-7**	**1.0E-5**	**1.0E-3**	**Baseline**
Lag 1	0.0119	0.0119	0.0119	0.0064
Lag 2	0.0109	0.0111	0.0113	0.0083
Lag 3	0.0109	0.0110	0.0114	0.0098
Lag 4	0.0110	0.0111	0.0111	0.0112
Lag 5	0.0110	0.0111	0.0111	0.0124
Lag 6	0.0112	0.0113	0.0113	0.0135

Tabla 13: resultados de los experimentos con AMRulesRegressor *variando su parámetro* `splitConfidence`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.5.b `gracePeriod`

Como ya hemos visto anteriormente, este parámetro determina a partir de cuentas nuevas instancias se empiezan a tomar decisiones. Aquí, simplemente hemos probado con 200 y 20000, y, a diferencia del resto de algoritmos, aquí un valor menor es ligeramente mejor, como se ve en los resultados de la Tabla 14.

Lag\ Parámetro	**200**	**20000**	**Baseline**
Lag 1	0.0119	0.0121	0.0064
Lag 2	0.0109	0.0113	0.0083
Lag 3	0.0109	0.0115	0.0098
Lag 4	0.0110	0.0116	0.0112
Lag 5	0.0110	0.0116	0.0124
Lag 6	0.0112	0.0119	0.0135

Tabla 14: resultados de los experimentos con AMRulesRegressor *variando su parámetro* `gracePeriod`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.5.c `setUnorderedRulesOn`

Este parámetro es muy simple, y sirve para elegir si las reglas van ordenadas o no. Como vemos en la Tabla 15, no hay grandes diferencias.

Lag\ Parámetro	**ordered**	**unordered**	**Baseline**
Lag 1	0.0119	0.0119	0.0064
Lag 2	0.0109	0.0110	0.0083
Lag 3	0.0109	0.0110	0.0098
Lag 4	0.0110	0.0111	0.0112
Lag 5	0.0110	0.0111	0.0124
Lag 6	0.0112	0.0112	0.0135

Tabla 15: resultados de los experimentos con AMRulesRegressor *variando su parámetro* `setUnorderedRulesOn`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.4.5.d `learnerOption`

AMRulesRegressor funciona con algoritmos de aprendizaje internos más complejo. En concreto, por defecto se utiliza *AdaptativeMultiTargetRegressor*, que combina dos regresores y utiliza el regresor con el menor error para predecir. En la configuración inicial, utiliza *MultiTargetMeanRegressor* y *MultiTargetPerceptronRegressor*. Como hemos visto antes, *FadingTargetMean* es una mejora de *TargetMean*, por lo que hemos probado si modificar el algoritmo base *MultiTargetMeanRegressor* de *TargetMean* a *FadingTargetMean* mejora el resultado. Hemos elegido varios valores del parámetro `fadingFactor` (0.5, 0.1, y 0), y como se ve en la Tabla 16, los resultados sí que se ven mejorados.

Lag\ Parámetro	**TM**	**FTM 0.5**	**FTM 0.1**	**FTM 0.0**	**Baseline**
Lag 1	0.0119	0.0115	0.0111	0.0110	0.0064
Lag 2	0.0109	0.0107	0.0097	0.0095	0.0083
Lag 3	0.0109	0.0106	0.0096	0.0093	0.0098
Lag 4	0.0110	0.0105	0.0095	0.0093	0.0112
Lag 5	0.0110	0.0104	0.0093	0.0091	0.0124
Lag 6	0.0112	0.0103	0.0093	0.0091	0.0135

Tabla 16: resultados de los experimentos con AMRulesRegressor *variando su parámetro* `learnerOption`*, en los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

4.5. Experimentos finales con algoritmos optimizados

En esta sección, simplemente vamos a recapitular los resultados optimizados para cada uno de los diferentes algoritmos de aprendizaje automático en línea utilizados. Los parámetros utilizados son los que hemos optimizado en la sección anterior, es decir:

- *Perceptron* con `learningRatio` = 1
- *FIMT-DD* con `gracePeriod` = 20000
- *ORTO* con `gracePeriod` = 20000
- *AMRulesRegressor* con *FadingTargetMean* con `fadingFactor` a 0 dentro de `learnerOption`
- *FadingTargetMean* con `fadingFactor` a 0

Como se puede ver en la Tabla 17, sobre el total de enlaces no hay ninguno de los algoritmos optimizados que superen al *baseline*. No obstante, sí que se obtienen resultados positivos en algunos de los *clusters*, recogidos en el Anexo D (D.2. Resultados finales divididos por *clusters*). En concreto, el algoritmo *FadingTargetMean* supera al *baseline* en los *clusters* entre los percentiles 75-80 (Tabla 36), 85-90 (Tabla 38) y 90-95 (Tabla 39). En otros *clusters* se iguala el resultado, pero no se ha considerado relevante igualar al *baseline*, ya que, para eso, se usa el *baseline*. En los tres casos mencionados, utilizando Z-test obtenemos que los resultados no son estadísticamente significativos, ya que obtenemos un p-valor cercano a 1. Dado que los resultados no son especialmente relevantes, no dibujamos ningún diagrama de caja, ya que no nos aporta gran información en este caso.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.01098**	**0.01197**	**0.01197**	**0.00641**	0.01098	**0.00641**
Lag 2	0.01196	0.01346	0.01346	0.00641	0.00947	0.00833
Lag 3	0.01441	0.01435	0.01435	0.00641	0.00935	0.00988
Lag 4	0.01699	0.01462	0.01462	0.00641	0.00926	0.01126
Lag 5	0.01959	0.01364	0.01364	0.00642	0.00909	0.01247
Lag 6	0.02245	0.01865	0.01865	0.00642	**0.00906**	0.01353

Tabla 17: resultados finales de los experimentos. 5 algoritmos + baseline *sobre todos los enlaces variables en el primer día, con ventana temporal de 1 a 6 unidades.*

Una vez hecho esto, vamos a repetir los experimentos con los mejores parámetros y el mejor valor de *lag* para los 14 días del *dataset*. En este caso, hemos elegido los enlaces que tenían variaciones dentro de cada día de los que estaban activos, obteniéndose un total de 358 enlaces. No obstante, hemos eliminado 11 de ellos, obteniendo una cantidad final de 347, ya que había pocos datos para algunos de los experimentos. Como se aprecia en la Tabla 18, los resultados son equivalentes a los de los experimentos con un día de datos. Por un lado, el *baseline* es el que mejor resultado obtiene, junto a *FadingTargetMean* (Los resultados del Z-test indican que la diferencia de estos promedios de MAE no es estadísticamente significativa). En este caso, *FIMT-DD* obtiene mejores resultados que *ORTO*. Que en este caso los resultados no sean iguales es debido a que el valor de `gracePeriod` (20000) es

menor que el total de datos, por lo que ahora sí que se están realizando nuevas divisiones. Es esperable que subiendo de nuevo este valor ambos algoritmos obtuviesen los mismos resultados (y mejores, como en el caso anterior), pero dado que los resultados son mucho peores que los del *baseline*, no vamos a considerarlo.

Algoritmo y Lag	**MAE promedio**
Perceptron – Lag 1	0.01901
FIMT-DD – Lag 1	0.02110
ORTO – Lag 1	0.02359
FTM – Lag 1	0.01058
AMRR – Lag 6	0.01692
Baseline – Lag 1	0.01058

Tabla 18: resultados finales de los experimentos. 5 algoritmos + baseline *sobre el conjunto de enlaces variables en cada día que tienen actividad, con la mejor ventana temporal obtenida en los experimentos de los enlaces variables del primer día del* dataset.

4.6. Predicciones a medio plazo

Como hemos visto en secciones anteriores, casi en ningún caso los algoritmos de aprendizaje automático en línea mejoran al *baseline* en predicciones cada 5 segundos, y en los casos que lo mejoran, los resultados no son estadísticamente significativos. En [13] y [14] se vio que utilizando el *dataset* con capturas cada 5 minutos, si había algoritmos que funcionaban mejores que el *baseline*. Por tanto, el objetivo de esta sección es determinar si con tiempos también pequeños (por ejemplo: 30 segundos, 1 minuto), algún algoritmo funciona mejor que el *baseline* o no. Para ello, seguiremos utilizando el *dataset* remuestreado cada 5 segundos con la selección de enlaces variables del primer día.

Para realizar estos experimentos, realizaremos predicciones con un *horizonte* desde 5 hasta 60 segundos vista. Para ello, para cada instante t (salida) del que se tenga información, se creará un vector de entrenamiento consistente de una cantidad *lag* de medidas (entrada), que en este caso no serán las inmediatamente anteriores al instante t, sino que serán de un número de instantes anteriores correspondientes al *horizonte* de predicción deseado, de cara a crear vectores de entrada que permitan aprender esta correspondencia entrada/salida (un vector de longitud *lag* + una medida separada el horizonte de predicción deseado). En la Figura 27 se muestra un ejemplo de lo indicado anteriormente. De esta manera, una vez que los algoritmos hayan entrenado suficiente, para un instante t, teniendo los *lag - 1* valores inmediatamente anteriores (es decir, en total *lag* valores), se podrá predecir el valor futuro separado el *horizonte* deseado. En cuanto al valor de *lag*, aunque en el apartado anterior seleccionamos 6 como el valor máximo, dado que ahora el horizonte de predicción va a estar separado hasta 1 minuto (12 medidas de

valor de LQ), hemos considerado mejor utilizar una ventana temporal máxima también de hasta 12 unidades.

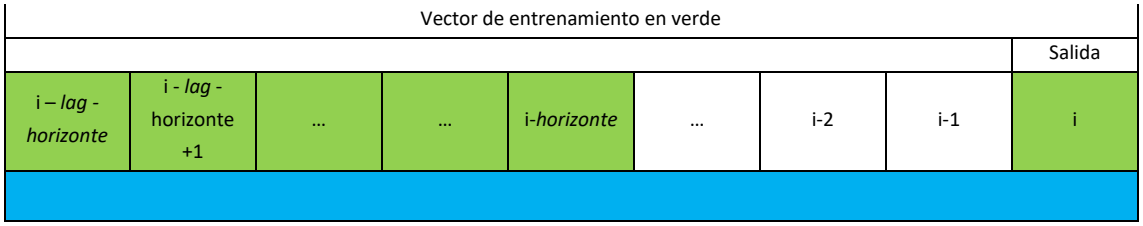

Figura 27: ejemplo de los vectores de entrenamiento para las predicciones a medio plazo.

Inicialmente se pensó realizar estos experimentos únicamente con el *baseline* y *FadingTargetMean* con el valor de `fadingFactor` a 0. No obstante a la vista de que estos resultados eran prácticamente iguales se añadieron otros dos valores de `FadingFactor` (0.1 y 0.5) y los algoritmos *Perceptron* y *FIMT-DD*. Dado que hemos realizado experimentos con ventanas temporales de 1 a 12 unidades y horizonte de 1 a 12 instantes vista, hemos decidido simplificar la exposición de los resultados y únicamente mostrar la ventana con mejor resultado para cada algoritmo. A la vista de los resultados expuestos en la Tabla 30 y en la Tabla 31 podemos extraer varias conclusiones interesantes. En primer lugar, el *baseline* no es el mejor, y para valores a corto plazo, es muy similar a *FadingTargetMean* (Aquí vemos que, para el horizonte temporal a 1 unidad, es decir, los experimentos previos, el *baseline* no es el mejor, pero dado que anteriormente solo utilizamos ventanas hasta 6 unidades, no lo vimos; no obstante, la diferencia es muy pequeña y no es significativa). En cuanto aumenta este horizonte temporal, *FIMT-DD* es el mejor, excepto para los experimentos a 12 instantes vista, que *Percepton* lo mejora ligeramente. Tanto *FIMT-DD* como *Perceptron* obtienen sus mejores resultados de manera consistente con *lag* de 1 unidad, y el *baseline* también excepto para horizontes muy alejados, donde *lag* = 2 se muestra como mejor opción. Las diferencias no son estadísticamente significativas en ningún caso. Por otro lado, podemos ver que, a horizontes cortos y medios, el valor óptimo de ventana para *FadingTargetMean* son mayores que 1, lo que nos indica que a estos horizontes un modelo más complejo puede ayudar a las predicciones. Conforme aumenta el horizonte, los valores óptimos disminuyen, llegando al valor 1 cuando el horizonte es de 60 segundos (12 instantes vista).

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	0.640	1.065	1.372	1.644	1.861	2.032	2.158	2.255	2.356	2.450	2.521	2.583
FTM (0.1)	0.676	1.087	1.389	1.656	1.868	2.034	2.157	2.254	2.355	2.446	2.517	2.578
FTM(0.5)	0.922	1.258	1.523	1.749	1.927	2.067	2.176	2.270	2.361	2.440	2.501	2.551
Perceptron	1.098	1.453	1.734	1.909	2.040	2.141	2.212	2.255	2.309	2.347	2.386	2.403
FIMT-DD	1.197	1.495	1.717	1.890	1.996	2.070	2.146	2.219	2.277	2.335	2.373	2.413
Baseline	0.641	1.066	1.377	1.650	1.866	2.037	2.163	2.257	2.357	2.370	2.378	2.438

Tabla 19: resultados de los experimentos sobre los enlaces variables del primer día. Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	10	9	10	7	5	6	4	3	2	3	2	1
FTM (0.1)	11	10	10	7	8	6	5	3	2	4	1	1
FTM(0.5)	11	9	10	7	8	6	5	4	2	1	2	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	1	2	2	2

Tabla 20: valores de ventana para los resultados de la Tabla 19.

Analizando los *clusters*, de los cuales dejamos los resultados en el Anexo D (D.3. Resultados de predicción a varios instantes vista), podemos extraer una serie de conclusiones:

- En general, para predicciones a corto plazo, lo que mejor funciona es *FadingTargetMean* (`fadingFactor` = 0) y/o el *baseline* .
- Para predicciones a más instantes vista, tanto Perceptron como FIMT-DD se demuestran como mejores alternativas (siempre con *lag* = 1), excepto en los *clusters* con más desviación típica de la LQ, en la que *FadingTargetMean* con diferentes parámetros obtienen mejor valor.
- Conforme las predicciones son a más largo plazo, *baseline* con *lag* = 2 (y mayores) empieza a ser mejor que con *lag* = 1.

Por otro lado, calculando la diferencia del error entre el mejor algoritmo y el *baseline* para cada horizonte temporal en cada uno de los *clusters*, hemos construido la Tabla 21. De aquí podemos extraer que, en los enlaces más variables, las diferencias entre el mejor algoritmo y el *baseline* son mayores, y son aún más

marcadas en los horizontes mayores. Para ver más claramente esto, hemos representado la misma información de esta tabla en unas figuras. En la Figura 28 se representan los 4 cuartiles, en la Figura 29 los *clusters* excepto los 4 cuartiles, y en la Figura 30 todo lo anterior. Los *clusters* en los que se observa más este comportamiento es en los *clusters* p80-85, p85-90y p99-100. En concreto, en p85-90 y p99-100 se puede ver una gráfica en forma de 'U', en la que vemos que las mayores diferencias respecto al *baseline* se obtienen en horizontes medios y lejanos. En cambio, en p80-85 el comportamiento es estable a partir de un horizonte de más de 6 instantes (30 segundos), obteniéndose una diferencia con el *baseline* de un 0.5%.

Cluster / *Instantes vista*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Q1	0	0	0	0	0	0.002	0.003	0.004	0.004	0.005	0.005	0.005
Q2	0	0	0	0	0	0.007	0.013	0.014	0.016	0.017	0.019	0.021
Q3	0.001	0.001	0.001	0	0	0	0.003	0.07	0.126	0.094	0.1	0.11
Q4	0.002	0.003	0.019	0.027	0.022	0.035	0.17	0.186	0.295	0.147	0.02	0.098
p75-80	0	0.002	0.002	0	0.001	0.072	0.199	0.291	0.374	0.329	0.27	0.243
p80-85	0	0	0	0	0	0,22	0.491	0.504	0.574	0.471	0.481	0.475
p85-90	0.076	0.096	0.137	0.145	0.12	0.593	0.311	0.239	0.204	0.147	0.162	0.554
p90-95	0.002	0.004	0.005	0.004	0.002	0.077	0.235	0.235	0.316	0.152	0.018	0.154
p95-100	0	0	0	0.028	0.059	0.055	0.041	0.05	0.026	0.006	0.018	0.058
p99-100	0	0	0	0.042	0.606	0.409	0.192	0.207	0.093	0.019	0.112	0.241

Tabla 21: diferencia de error (en tanto por ciento) entre el mejor algoritmo y el baseline*, para cada uno de los* clusters *con horizonte temporal de 1 a 12 unidades.*

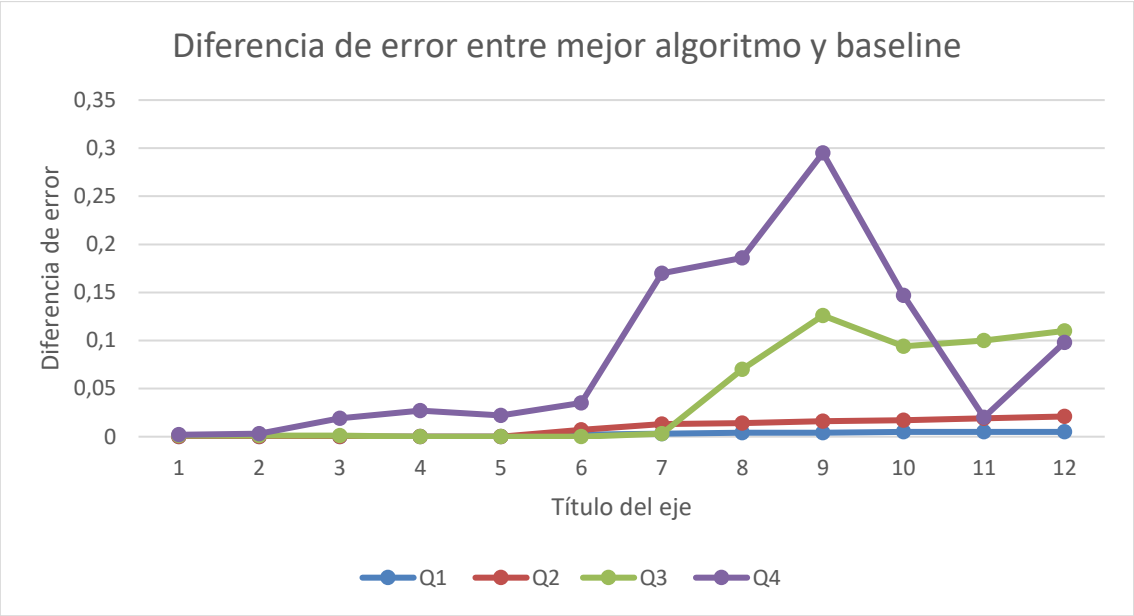

Figura 28: representación de la diferencia de error (en tanto por ciento) entre el mejor algoritmo y el baseline*, para cada uno de los cuatro cuartiles con horizonte temporal de 1 a 12 unidades.*

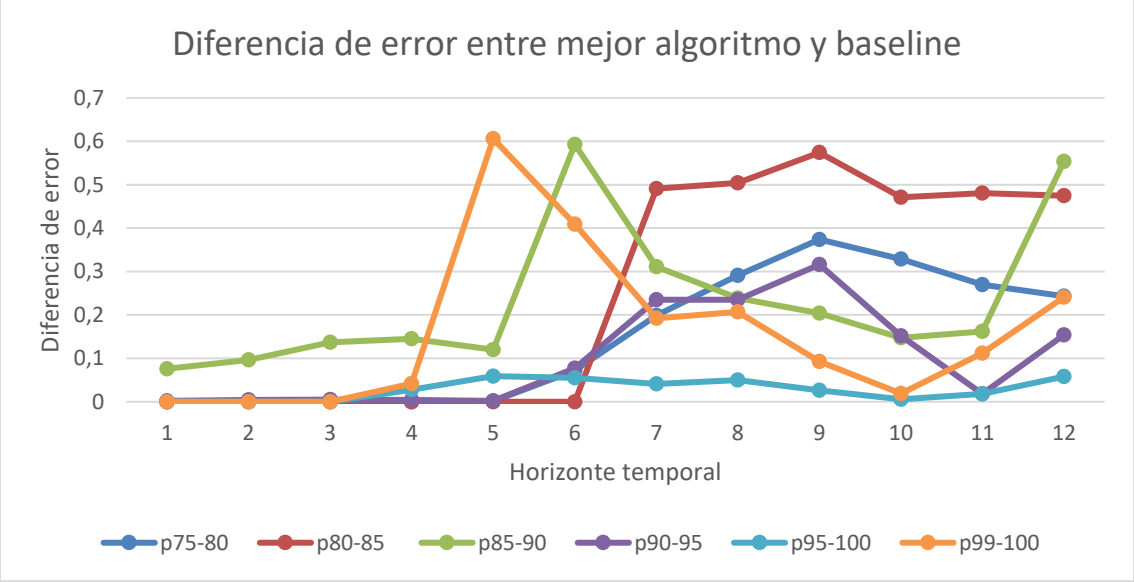

Figura 29: representación de la diferencia de error (en tanto por ciento) entre el mejor algoritmo y el baseline, para los clusters (p75-80, p80-85, p85-90, p90-95, p95-100, p99-100) *con horizonte temporal de 1 a 12 unidades.*

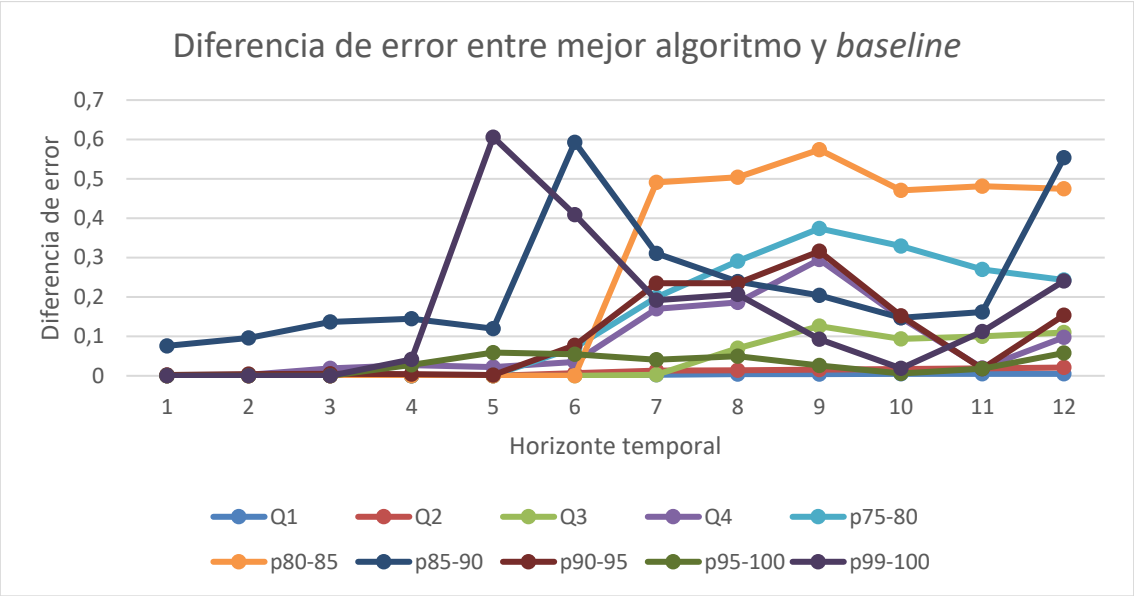

Figura 30: representación de la diferencia de error (en tanto por ciento) entre el mejor algoritmo y el baseline, para cada uno de los clusters *con horizonte temporal de 1 a 12 unidades.*

4.7. Conclusiones

Tras realizar los experimentos, hemos podido constatar que en una red como la de nuestro *dataset*, algoritmos simples como el *baseline* y *FadingTargetMean* funcionan muy bien para instantes cortos. De hecho, los primeros resultados que obtuvimos eran mucho peores que los del *baseline*, y tras modificar los parámetros de los algoritmos (simplificando los modelos), hemos obtenido mejores resultados, lo cual es coherente con que el *baseline* obtenga tan buenos resultados. Analizando estos resultados por *clusters*, pudimos ver que en alguno de ellos los resultados ya mejoraban al *baseline*, pero no eran estadísticamente significativos. Posteriormente, y a modo de nexo con los trabajos [13] y [14], se realizaron experimentos con un horizonte temporal más alejado, pero sin llegar a los 5 minutos que se utilizaba en estos trabajos. Se pudo ver que los resultados mejoraban de forma general, y pese a que los resultados no eran estadísticamente significativos, ya nos indica que, a mayores horizontes temporales del escogido, el *baseline* ya no será el que mejor se comporte. Además se observó que en los *clusters* más variables, la diferencia es más acusada y que se acentúa en horizontes más alejados.

En la parte restante del documento se extraerán las conclusiones de este Trabajo Fin de Máster. También se describirán posibles vías de trabajo futuro.

Capítulo 5. Conclusiones

El objetivo de este Trabajo Fin de Máster era analizar el resultado de algoritmos de aprendizaje automático en línea sobre datos reales de LQ obtenidos de una WCN capturados con una alta frecuencia de muestreo, para superar la limitación de [7], [13] y [14] respecto a la periodicidad de los datos capturados.

En primer lugar, se han obtenido datos reales de la WCN Funkfeuer Graz. Para ello, fue necesario obtener acceso directo a esta red y construir por nuestra parte un *dataset* que sí cumpliera los requisitos deseados respecto a su periodicidad. Para ello, se realizó una conexión a la WCN de Graz con un túnel VPN. Una vez conectado, esta máquina virtual ejecutó el cliente del protocolo OLSR actuando como un nodo de dicha red. Dado que OLSR es un protocolo de estado de enlace, se puede obtener en un único nodo la información de LQ de toda la red. Realizando consultas periódicas a este cliente OLSR se fue construyendo el *dataset* completo.

Por otra parte, se ha realizado un análisis descriptivo de la red. Haciendo un análisis geográfico, se identificó que la mayor parte de los enlaces de esta WCN estaban concentrados en la ciudad de Graz y los alrededores cercanos. Apoyándonos en este análisis geográfico, se ha constatado que la mayoría de los enlaces que no varían tienen una longitud física corta. En cuando a la actividad de los enlaces, la mayor parte de ellos tienen una actividad prácticamente del 100%, con una LQ promedio elevada y una desviación típica reducida. No obstante, hay un 4% de enlaces activos menos de un 10% del tiempo. Dado que el *dataset* construido constaba de 14 días, se realizó un análisis horario y semanal, para poder encontrar algún tipo de periodicidad., que finalmente no había. Para identificar periodos de variación importantes de los enlaces, se ha empleado la transformada matemática FFT. Aunque en la mayoría de los enlaces no se apreciaba nada relevante (la mayor parte de cambios ocurren con periodos muy grandes, es decir, frecuencia muy pequeña), en los *clusters* de enlaces más variables sí que se detectó algún periodo interesante, que se tomó como referencia para el resto del estudio.

Una vez hecho este análisis descriptivo, se utilizó parte de la información extraída para poder enfocar mejor los experimentos. En concreto, se decidió analizar únicamente el primer día, ya que no había diferencias significativas entre diferentes días y no se veían periodicidades. Además, se remuestreó el *dataset* cada 5 segundos. Se han realizado experimentos con 5 algoritmos de aprendizaje automático en línea, contra un *baseline* que sirve como algoritmo de referencia. Los resultados iniciales de estos experimentos han sido muy negativos, y el *baseline* los superaba con facilidad. Buscando una solución mejor, se ha optimizado de manera

individual cada algoritmo, analizando algunos de sus parámetros y variando los que se consideraban interesantes. En general, los parámetros que mejoraban los resultados han sido aquellos que simplificaban los modelos. Esto es coherente con que el *baseline*, que es un algoritmo muy sencillo, sea el que mejores resultados obtenía. Con estas mejoras, en algunos de los *clusters* sí se obtenían mejores resultados que los del *baseline*, no obstante, estos resultados no eran estadísticamente significativos.

Finalmente, hemos considerado interesante cambiar el objetivo de predicción a varios instantes vista, y hemos realizado predicciones con horizontes desde 5 hasta 60 segundos. En casi todos los *clusters* el resultado ha sido satisfactorio, ya que se ha mejorado el resultado del *baseline*. También se vio que en los *clusters* más variables, la diferencia es más grande y que se acentúa en horizontes más alejados. Los resultados no son estadísticamente significativos, pero nos indica que para valores mayores como los de [13] y [14] el *baseline* ya no será el que mejor se comporte.

5.1. Trabajo Futuro

A la vista de los resultados obtenidos, se nos abren varias líneas de investigación. En primer lugar, en los trabajos previos utilizaban un muestreo de 5 minutos, y aquí hemos empezado con 1 segundo remuestreados después a 5, para realizar finalmente predicciones a varios segundos vista (de 5 a 60). El problema de los trabajos anteriores era que 5 minutos podría ser un valor demasiado elevado desde el punto de vista telemático. En este caso, hemos visto como aproximadamente a partir de predicciones a 30 segundos vista los resultados del *baseline* (y de *FadingTargetMean*) dejan de ser los mejores, excepto en enlaces con mucha desviación típica. No obstante, nada de esto significa nada sin testearlo en una red real, por lo que el siguiente paso debería ser analizar las necesidades reales de una red de este estilo, para determinar finalmente si el trabajo realizado es útil en un caso práctico. Para ello, lo más sencillo sería simular una red a partir de los datos de topología capturados en este trabajo, de manera que se pudiese verificar de manera fidedigna si las predicciones mejoran el encaminamiento, y en qué medida.

Por otra parte, dado que el objetivo en sí es mejorar el encaminamiento en redes comunitarias, sería interesante utilizar los datos de topología y rutas capturados para realizar predicciones de la tabla de reenvío de cada uno de los nodos (o en su lugar, predecir el siguiente salto para un destino en concreto). En este caso, la métrica ya no sería el error, sino las métricas utilizadas en problemas de clasificación, como *accuracy*, *precision-recall*, F1-score, etc.

Finalmente, podría resultar útil emplear los datos geográficos de los nodos para crear conjuntos de entrenamiento distintos en función de variables físicas como altitud, distancia entre nodos, etcétera.

Referencias

[1] B. Braem, C. Blondia, C. Barz, H. Rogge, F. Freitag, L. Navarro, J. Bonicioli, S. Papathanosiou, P. Escrich, R. Baig Viñas, A. L. Kaplan, A. Neumann, I. Vilata i Balaguer, B. Tatum and M. Matson, "A case for research with on community networks," vol. 43, no. 3, pp. 68-73, jul 2013.

[2] L. Maccari and R. Lo Cigno, "A week in the life of three large Wireless Comunnity Networks," *Ad Hoc Networks,* vol. 24, pp. 175-190, jan 2015.

[3] J. Avonts, B. Braem, Blondia and Chris, "A Questionnaire based Examination of Community Networks," in *International Workshop on Community Networks and Bottom-up-Broadband*, 2013.

[4] A. Woo, T. Tong and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," in *International Conference on Embedded Networked Sensor Systems*, 2003.

[5] R. Draves, J. Padhye and B. Zill, "Comparison of Routing Metrics for Static Multi-Hop Wireless Networks," in *SIGCOMM*, 2004.

[6] K. Farkas, T. Hossmann, F. Legendre, B. Plattner and S. K. Das, "Link quality prediction in mesh networks," *Computer Communications,* vol. 31, no. 8, pp. 1497-1512, 25 may 2008.

[7] P. Millan, C. Molina, E. Medina, D. Vega, R. Meseguer, B. Braem and C. Blondia, "Time series analysis to predict link quality of Wireless community networks," *Computer Networks,* vol. 93, no. 2, pp. 342-358, dec 2015.

[8] Y. Zhao, S. Li and J. Hou, "Link Quality Prediction via a Neighborhood-Based Nonnegative Matrix Factorization Model for Wireless Sensor Networks," *International Jorunal of Distributed Sensor Networks,* vol. 11, no. 10, jan 2015.

[9] C. J. Lowrance and A. P. Lauf, "An active and incremental learning framework for the online prediction of link quality in robot networks," *Engineering Applications of Artificial Intelligence,* vol. 77, pp. 197-211, Jan. 2019.

[10] Z. Bu, Y. Wang, H.-J. Li, J. Jiang, Z. Wu and J. Cao, "Link prediction in temporal networks: Integrating survival analysis and game theory," *Information Sciences,* vol. 498, pp. 41-61, Sep. 2019.

[11] K. Lei, M. Qin, B. Bai, G. Zhang and M. Yang, "GCN-GAN: A Non-linear Temporal Link Prediction Model for Weighted Dynamic Networks," in *IEE Infocom 2019*, 2019.

[12] T. Liu and A. Cerpa, "Temporal Adaptive Link Quality Prediction with Online Learning," *ACM Transactions on Sensor Networks (TOSN),* vol. 10, no. 3, apr 2014.

[13] C. Mediavilla Pastor, *Predicción de calidad de enlace en redes comunitarias inalámbricas basadas en OLSR,* 2017.

[14] M. L. Bote Lorenzo, E. Gómez Sánchez, C. Mediavilla Pastor and J. I. Asensio Pérez, "Online machine learning algorithms to predict link quality in community wireless mesh networks," *Computer Networks,* vol. 132, pp. 68-80, 26 February 2018.

[15] R. Lasser, "Engenieering Method, Electrical and Computer Engineering Design Handbook," Tufts University School of Engineering, [Online]. Available: https://sites.tufts.edu/eeseniordesignhandbook/2013/engineering-method/. [Accessed 28 apr 2017].

[16] IEEE, "IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". 14 Dec. 2016.

[17] P. A. Frangoudis, G. C. Polyzos and V. P. Kemerlis, "Wireless Community Networks: An Alternative Approach for Nomadic Broadband Network Access," *IEEE Communications Magazine,* vol. 49, no. 5, pp. 206-2013, may 2011.

[18] FON, "FON," [Online]. Available: https://fon.com//. [Accessed 3 Jun. 2019].

[19] J. F. Kurose and K. W. Ross, Computer Networking: A Top-Down Approach, 6th ed., Boston: Pearson, 2013.

[20] T. (. N. G. Malkin, "RIP Version 2". Patent Request for Comments 2453.

[21] J. Moy, "OSPF Version 2". Patent Request for Comments 2328, Apr. 1998.

[22] Y. Rekhter, T. Li and S. Hares, "A Border Gateway Protocol 4 (BGP-4)". Patent Request for Comments 4271, Jan. 2006.

[23] R. Ogier, S. International and B. P. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding". Patent Request for Comments 5614, Aug. 2009.

[24] A. Roy, C. Systems and M. Chandra, " Extensions to OSPF to Support Mobile Ad Hoc Networking". Patent Request for Comments 5820, Mar. 2010.

[25] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum and L. Viennot, "Optimized Link State Routing Protocol for Ad Hoc Networks," in *Multi Topic COnference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, Lahore, Pakistan, 2001.

[26] T. Clausen, P. Jacquet and I. Project Hipercom, "Optimized Link State Routing Protocol (OLSR)". Patent Request for Comments 3626, Oct. 2003.

[27] T. Lopatic and A. Tonnesen, "olsrd Link Quality Extensions," 5 oct 2004. [Online]. Available: http://www.olsr.org/docs/README-Link-Quality.html. [Accessed 26 jun 2017].

[28] olsr.org, "OLSR Project," [Online]. Available: http://www.olsr.org/mediawiki/index.php/Projects. [Accessed 3 Jun. 2019].

[29] D. S. De Couto, High-Throughput Routing for Multi-Hop Wireless Networks, 2004.

[30] T. Clausen, E. P. LIX, C. Dearlove, B. S. ATC, P. Jacquet, A.-L. B. Labs, U. Herberg and F. L. o. America, "The Optimized Link State Routing Protocol Version 2". Patent Request for Comments 7181, Apro. 2014.

[31] US Naval Research Laboratory: Networks and Communication Systems Branch, "The NRL OLSR Routing Protocol Implementation," [Online]. Available: https://www.nrl.navy.mil/itd/ncs/products/olsr. [Accessed 3 Jun. 2019].

[32] moviquity, "Windows Mobile OLSR Daemon," 22 Jan. 2018. [Online]. Available: https://sourceforge.net/projects/wmolsr/. [Accessed 3 Jun. 2019].

[33] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development,* vol. 3, no. 3, p. 210, 1959.

[34] T. Mitchell, Machine Learning, McGraw-Hill, 1997.

[35] A. Ng, "Machine Learning. Stanford University," [Online]. Available: https://www.coursera.org/learn/machine-learning. [Accessed 14 apr 2017].

[36] D. Peteiro- Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Progress in Artificial Intelligence,* vol. 2, no. 1, pp. 1-11, mar 2013.

[37] The University of Waikato, "Weka 3: Machine Learning Software in Java," [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/. [Accessed 3 Jun. 2019].

[38] The R Project for Statistical Computing, "R version 3.6.0 (Planting of a Tree)," 26 Apr. 2019. [Online]. Available: https://www.r-project.org/. [Accessed 3 Jun. 2019].

[39] scikit-learn, "scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org/stable/. [Accessed 3 Jun. 2019].

[40] scikit-learn, "Who is using scikit-learn?," [Online]. Available: https://scikit-learn.org/stable/testimonials/testimonials.html. [Accessed 3 Jun. 2019].

[41] The University of Waikato, "MOA Machine Learning for Streams," Apr. 2019. [Online]. Available: https://moa.cms.waikato.ac.nz/. [Accessed 3 Jun. 2019].

[42] Télécom ParisTech & École Polytechnique, "scikit-multiflow - A machine learning framework for multi-output/multilabel and stream data.," 2019. [Online]. Available: https://scikit-multiflow.github.io/. [Accessed 3 Jun. 2019].

[43] jread82_nz, "MEKA: A Multi-label Extension to Weka," 29 Mar. 2018. [Online]. Available: https://sourceforge.net/projects/meka/. [Accessed 3 Jun. 2019].

[44] G. De Francisci Morales and A. Bifet, "SAMOA: Scalable Advanced Massive Online Analysis," *Journal of Machine Learning Research,* vol. 16, pp. 149-153, 2015.

[45] CONFINE, "CONFINE project," 2019. [Online]. Available: https://confine-project.eu/. [Accessed 3 Jun. 2019].

[46] Funkfeuer Graz, "Funkfeuer Graz Stats," 2019. [Online]. Available: http://stats.ffgraz.net/index.php?module=olsr&view=topo. [Accessed 3 Jun. 2019].

[47] C. Mediavilla Pastor, E. Gómez Sánchez and M. L. Bote Lorenzo, *Funkfeuer Graz topological and routing raw data. [Data set]. Zenodo.,* Valladolid, Valladolid, 2019.

[48] Funkfeuer, "Funkfeuer," [Online]. Available: https://www.funkfeuer.at/. [Accessed 3 Jun. 2019].

[49] J. Whitaker and T. m. d. t. team, "The Matplotlib Basemap Toolkit User's Guide," 4 May. 2017. [Online]. Available: https://matplotlib.org/basemap/users/index.html. [Accessed 30 Jun. 2019].

[50] J. Hunter, D. Dale, E. Firing, M. Droettboom and T. m. d. t. team, "matplotlib," 18 May. 2019. [Online]. Available: https://matplotlib.org/. [Accessed 30 Jun. 2019].

[51] R. Veciana i Rovira, "Basemap Background methods - arcgisimage," 2014. [Online]. Available: https://basemaptutorial.readthedocs.io/en/latest/backgrounds.html#arcgisimage. [Accessed 30 Jun. 2019].

[52] ArcGIS, "ArcGIS REST Services Directory," [Online]. Available: http://server.arcgisonline.com/arcgis/rest/services. [Accessed 30 Jun. 2019].

[53] A. Bifet, G. Holmes, R. Kirkby and B. Pfahringer, "DATA STREAM MINING: A Practical Aproach," 2011.

[54] W. Navidi, Statistics for Engineers and Scientists, 3rd ed., McGraw-Hill, 2011.

[55] Funkfeuer Graz, "Tutorial de configuración de una conexión a través de un túnel VPN a la red FunkFeuer Graz (para Debian) (alemán)," 25 Aug. 2017. [Online]. Available: https://wiki.graz.funkfeuer.at/FunkInselDebian. [Accessed 3 Jun. 2019].

[56] Debian Bug reports logs, "Fallo al no existir el directorio /etc/openvpn/tmp," 24 Nov. 2014. [Online]. Available: https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=770811. [Accessed 3 Jun. 2019].

[57] Canonical, "Configuración de OpenVPN (Ubuntu).," [Online]. Available: https://help.ubuntu.com/lts/serverguide/openvpn.html. [Accessed 3 Jun. 2019].

[58] Debian Bug reports logs, "Opciones LinkQualityDijkstraLimit y LinkQualityWinSize eliminadas," 19 Dec. 2011. [Online]. Available: https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=652690. [Accessed 3 Jun. 2019].

[59] olsr.org, "olsrd.conf," 29 Sep. 2005. [Online]. Available: http://www.olsr.org/docs/olsrd.conf.5.html. [Accessed 3 Jun. 2019].

[60] A. Bifet, G. Holmes, B. Pfahringer and E. Frank, "Fast Perceptron Decision Tree Learning from Evolving Data Streams," in *Advances in Knowledge Discovery and Data Mining. PAKDD 2010.*, 2010.

[61] L. Torgo and J. Gama, "Regression using classification algorithms," *Intelligent Data Analysis,* vol. 1, no. 1-4, pp. 275-292, 1997.

[62] E. Ikonomovska, J. Gama and S. Džeroski, "Learning model trees from evolving data streams," *Data Mining and Knowledge Discovery,* vol. 23, no. 1, pp. 128-168, Jul. 2011.

[63] A. Wong, "Introduction to Model Trees from scrath," 2 Jul. 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-model-trees-6e396259379a. [Accessed 30 Jun. 2019].

[64] E. Ikonomovska, J. Gama, B. Ženko and S. Džeroski, "Speeding-up Hoeffding-based regression trees with options," in *Proceedings of 28th International Conference on Machine Learning, ACM*, 2011.

[65] E. Almeida, C. Ferreira and J. Gama, "Adaptive Model Rules from Data Streams," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Prague, 2013.

Anexo A

Conexión a la WCN Funkfeuer Graz y captura de datos

En este Anexo se explica la configuración necesaria que se ha utilizado para conectarse a la WCN Funkfeuer Graz y posteriormente obtener los datos utilizados en este Trabajo Fin de Máster. Se recuerda que se dispuso de una máquina virtual conectada a la red de Graz mediante un túnel VPN para el que nos dieron acceso. Una vez conectado, esta máquina virtual ejecutó el cliente del protocolo OLSR actuando como un nodo de dicha red. Dado que OLSR es un protocolo de estado de enlace, se pudo obtener en un único nodo la información de LQ de toda la red. Realizando consultas periódicas a este cliente OLSR se fue construyendo durante 2 semanas (14 días) el dataset completo.

A.1. Creación del túnel OpenVPN

Antes de empezar, es necesario obtener los ficheros necesarios para poder conectarse a la red VPN. Estos ficheros son:

- `ca.crt`: Certificado raíz SSL/TLS de Autoridad Certificadora (CA, *Certification Authority*).
- `research.crt`: Certificado.
- `research.key`: Clave Privada.

Cada cliente y el servidor deben tener sus propios ficheros de Certificado y de Clave Privada. El fichero de CA se puede compartir. Una vez obtenidos estos ficheros gracias a un miembro de la WCN Funkfeuer Graz, se puede realizar la configuración partiendo del tutorial [55] proporcionado por este mismo miembro, con algunas ligeras modificaciones.

En primer, se instala el servicio de cliente de `openvpn`:

```
sudo apt-get install openvpn
```

Una vez instalado, se crea el fichero de configuración `/etc/openvpn/client.conf` y se copian los tres ficheros proporcionados a la misma ruta. En el tutorial del sitio *web* el fichero se crea con el nombre `ff-tunnel.conf`, pero para ello hay que cambiar el nombre del fichero por defecto en la configuración de `openvpn`, así que se ha mantenido el nombre por defecto. El contenido del fichero es el siguiente:

```
client // Especifica que somos el cliente.

dev tap0 // Define el nombre de la interfaz del túnel.
```

```
proto udp // Define el tipo de protocolo que se utiliza para
conectar al servidor.

remote tun.graz.funkfeuer.at 1194 // host y puerto al que nos
conectamos (esto a veces da problemas de DNS, en ese caso mejor
cambiar el nombre del host por su dirección IP, 217.29.149.73 en
este caso en concreto)

resolv-retry infinite // Sigue intentando resolver el nombre del
host indefinidamente (útil para máquinas no conectadas
permanentemente a Internet).

nobind // La mayoría de los clientes no necesitan asociarse a un
puerto local específico.

persist-key // Mantiene el estado entre reinicios.
persist-tun // Mantiene el estado entre reinicios.

// Comandos
daemon ovpn-ff-tunnel
cd /etc/openvpn
chroot /etc/openvpn
user nobody
group nogroup

// Ficheros necesarios (SSL/TLS)
ca "ca.crt"
cert "research.crt"
key "research.key"

ns-cert-type server

cipher none // No utiliza cifrado

verb 3 // Verbosidad del fichero de log
```

La explicación de cada una de las opciones se ha realizado en base al fichero `/usr/share/doc/openvpn/examples/sample-config-files/client.conf`.

Después, se ha creado un directorio `/etc/openvpn/tmp`, ya que al emplear el comando `chroot` se necesita un directorio temporal dentro del directorio (que el programa no crea [56] cuando debería):

```
sudo mkdir /etc/openvpn/tmp
```

En paralelo se ha consultado la documentación de Ubuntu sobre OpenVPN [57] por si había algo adicional que en la configuración de FunkFeuer Graz no se realizase.

Finalmente, se arranca el servicio:

```
sudo systemctl start openvpn@client
```

Se comprueba que se ha creado la interfaz:

```
ifconfig tap0
```

Cuyo resultado aparece en la Figura 31, en la que también se observa que la dirección IP asociada a esta interfaz en la red VPN es la 10.12.11.23.

```
ubuntu@maquina-carlos:/usr/share/doc/openvpn/examples/sample-config-files$ ifconfig tap0
tap0      Link encap:Ethernet  HWaddr 72:c0:d9:50:54:d9
          inet addr:10.12.11.23  Bcast:10.12.11.255  Mask:255.255.255.0
          inet6 addr: fe80::70c0:d9ff:fe50:54d9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1610698 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:2227727628 (2.2 GB)  TX bytes:1418 (1.4 KB)
```

Figura 31: Interfaz tap0 relativa al túnel openvpn.

Comprobamos que podemos hacer ping a la dirección del servidor VPN (10.12.11.1), como podemos ver en la Figura 32.

```
ping 10.12.11.1
```

```
ubuntu@maquina-carlos:/usr/share/doc/openvpn/examples/sample-config-files$ ping 10.12.11.1
PING 10.12.11.1 (10.12.11.1) 56(84) bytes of data.
64 bytes from 10.12.11.1: icmp_seq=1 ttl=64 time=75.7 ms
64 bytes from 10.12.11.1: icmp_seq=2 ttl=64 time=104 ms
64 bytes from 10.12.11.1: icmp_seq=3 ttl=64 time=74.7 ms
64 bytes from 10.12.11.1: icmp_seq=4 ttl=64 time=71.4 ms
^C
--- 10.12.11.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 71.400/81.666/104.754/13.430 ms
ubuntu@maquina-carlos:/usr/share/doc/openvpn/examples/sample-config-files$
```

Figura 32: ping al servidor VPN.

Comprobamos la tabla de reenvío, como se ve en la Figura 33:

```
netstat -rn
```

```
ubuntu@maquina-carlos:/usr/share/doc/openvpn/examples/sample-config-files$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.10.250  0.0.0.0         UG        0 0          0 ens3
10.12.11.0      0.0.0.0         255.255.255.0   U         0 0          0 tap0
169.254.169.254 192.168.10.250  255.255.255.255 UGH       0 0          0 ens3
192.168.10.0    0.0.0.0         255.255.255.0   U         0 0          0 ens3
```

Figura 33: tabla de reenvío.

A.2. Configuración de `olsrd`

Tras conseguir conectarse a la red, es necesario ejecutar un cliente OLSR. Para ello, en primer lugar, se instala el servicio:

```
sudo apt-get install olsrd
```

Una vez instalado, se crea el fichero de configuración. Como ya existe uno, lo guardamos con otro nombre e introducimos el nuestro, `/etc/olsrd/olsrd.conf`. La configuración proporcionada en el sitio *web* de Funkfeuer Graz está obsoleto para la versión actual de `olsrd`, ya que dos parámetros que se indican, *LinkQualityWinSize* y *LinkQualityDijstraLimit* ya no existen [58]. Es probable que la versión de la *web* no esté actualizada o que algunos de los equipos de la red usen versiones más antiguas. No obstante, como con la versión utilizada se han obtenido satisfactoriamente los datos, no es un tema del que debamos preocuparnos. El fichero final de configuración se presenta a continuación. La descripción de las opciones de configuración se ha tomado de la página del manual de Linux, `olsrd.conf(5)` [59].

DebugLevel 0 // Controla la cantidad de datos enviada a la salida estándar. Por defecto es 0 (ninguna salida, funciona en segundo plano. Valor máximo 9.

IpVersion 4 // Elige entre IPv4 e IPv6. Valor por defecto 4.

AllowNoInt yes // Permite que se añadan y quiten interfaces en caliente. Valor por defecto "yes".

Pollrate 0.025 // Marca el tiempo en segundos para sondear en busca de nuevos eventos. Por defecto 0.1 (Según el manual los valores deberían ser entre 0.1 y 10, así que el 0.025 igual lo interpreta como 0.1).

TcRedundancy 2 // Controla la redundancia de TC usada por el nodo local en la generación de mensajes TC:

- 0(d): advertised link set = MPR selectors
- 1: advertised link set = MPR set + MPR selector set
- 2: advertised link set = fully symmetric neighbor set.

MprCoverage 7 // Decide cuántos MPR debe intentar seleccionar un nodo en cada vecindario de dos saltos. Por defecto a 1, cualquier otro valor puede reducir seriamente la optimización introducida por el esquema de MPR. En este caso debe dar buen resultado este valor.

LinkQualityFishEye 1 // Activa (1) o desactiva (0) el uso del algoritmo experimental de ojo de pez (Fish Eye).

LinkQualityLevel 2 // Decide que esquema de calidad del enlace utilizar:

- 0: no se utiliza la calidad del enlace y OLSR cumple la RFC 3626.
- 1: se utiliza la calidad del enlace para calcular los MPR.
- 2: se utiliza la calidad del enlace para calcular los MPR y para calcular las rutas.

Los valores 1 y 2 rompen el estándar y solo deben utilizarse si toda la red lo utiliza (como es el caso).

UseHysteresis no // Si se activa la histéresis se utiliza tal y como se explica en la sección 14 de la RFC 3636.

FIBMetric "approx" // Controla el valor de la métrica de los conjuntos de rutas. Esto no está muy bien explicado:

- "flat" (d): indica que el valor de métrica siempre es 2. Se prefiere este valor porque ayuda al kernel a limpiar rutas antiguas.
- "correct": usa el número de saltos como valor de métrica.
- "approx": usa el número de saltos, pero solo lo actualiza si cambia el nexthop.

ClearScreen yes // Si está activado y se está utilizando un nivel de depuración > 0, la pantalla se borra antes de cualquier actualización para ver mejor los datos.

LinkQualityAging 0.1 // Solo para LinkQualityLevel = 2. Parámetro para etx_float y etx_fpm.

LinkQualityAlgorithm "etx_ff" // Solo para LinkQualityLevel = 2:

- "etx_float": utiliza un valor de ETX en coma flotante con envejecimiento exponencial.
- "etx_fpm": igual que el anterior, pero con aritmética entera.
- "etx_ff (d)": ETX Freifunk, utiliza TODO el tráfico OLSR para calcular el ETX en vez de solo los mensajes HELLO.
- "etx_ffetch": una versión incompatible de etx_ff que permite enlaces ethernet con ETX 0.1 (Calidad "infinita").

```
RtTable 111 // Especifica la tabla que utiliza olsrd para las rutas de los hosts

RtTableDefault 112 // Especifica la tabla que utiliza olsr para la ruta al Gateway por defecto.


Interface "tap0"

{

        // Temporizadores

        HelloInterval 3.0

        HelloValidityTime 125.0

        TcInterval 2.0

        TcValidityTime 500.0

        MidInterval 25.0

        MidValidityTime 500.0

        HnaInterval 10.0

        HnaValidityTime 125.0

}

// Plugins, son iguales, pero en diferente formato (Uno lo devuelve en formato texto y otro en JSON)

LoadPlugin "olsrd_txtinfo.so.0.1"

{

        PlParam "port" 9091

        PlParam "accept" 0.0.0.0

}

LoadPlugin "olsrd_jsoninfo.so.0.0"

{

        PlParam "port" 9090

        P1Param "accept" 0.0.0.0
```

```
}
```

Una vez realizada esta configuración, es necesario configurar el encaminamiento con políticas para que funcione, ya que, si no, según la página del tutorial, el túnel `openvpn` se rompe. Para ello ya simplemente seguimos el resto de las instrucciones de la web. En primer lugar, se crea el fichero `/etc/network/policyrouting.sh` con el siguiente contenido:

```
#! /bin/sh
#
# Policy-Routing Script

// Interfaces involucradas (tap0 la creada y ens3 la real)

OLSR_IFACES="tap0 ens3"

case "$1" in
  up)
        ip rule add lookup olsr pref 20000
        ip rule add lookup main pref 30000
        ip rule del pref 32766

        PREF=32000
        for iface in $OLSR_IFACES; do
                ip rule add iif $iface pref $PREF lookup olsr-default
                PREF=$((PREF+1))
        done
    ;;
  down)
        PREF=32000
        for iface in $OLSR_IFACES; do
                ip rule del iif $iface pref $PREF lookup olsr-default
                PREF=$((PREF+1))
        done

        ip rule add lookup main pref 32766
        ip rule del pref 30000
        ip rule del pref 20000
    ;;
  *)
    echo "Usage: $0 {up|down}"
    exit 1
    ;;
esac

exit 0
```

Una vez creado el fichero, se modifican sus permisos para que se pueda ejecutar:

```
chmod +x /etc/network/policyrouting.sh
```

Para que este script funcione, hay que poner nombre a las tablas de encaminamiento de `olsr`. Para ello, al final del fichero `/etc/iprtoute2/rt_tables` añadimos lo siguiente:

```
111 olsr

112 olsr-default
```

Finalmente, es preciso modificar el fichero de interfaces /etc/network/interfaces:

Para la interfaz que está conectada a Internet, en ningún caso debe establecerse una puerta de enlace. Por lo que, si estuviese, la siguiente línea:

```
gateway 192.2.0.1
```

Debería ser sustituida por:

```
up ip router add default via 192.2.0.1 table default

down ip route del default via 192.2.0.1 table default
```

Y ahora, con la interfaz para la red OLSR, se debe añadir lo siguiente:

```
up /etc/network/policyrouting.sh up

down /etc/network/policyrouting.sh down
```

En este caso tampoco debe haber configuración de puerta de enlace.

Después de reiniciar, el túnel debe estar establecido y el encaminamiento con políticas configurado correctamente, por lo que no habrá ningún problema para capturar los datos deseados de la red FunkFeuer.

A.3. Captura de datos

Una vez configurado y ejecutado el cliente OLSR, es necesario extraer los datos relevantes que nos proporciona. Tal y como se ha detallado en la configuración de olsrd, se han configurado dos *plugins*, `txtinfo` y `jsoninfo`. Son equivalente, excepto que el primero nos lo muestra en texto plano y el segundo en formato JSON. Por simplicidad, se han tomado los datos en texto plano. Dentro de los datos recuperables con `txtinfo` tenemos:

- *Config*
- *Gateways*
- HNA
- *Interfaces*
- *Links*
- MID
- *Neighbors*
- *Routes*
- *Topology*
- *2-hop neighbors*

De los anteriores, los que son más importantes de cara a realizar predicciones son dos, *Routes* y *Topology*. La primera indica la ruta a una red o máquina, con su coste total (ETX suma de todos los enlaces), su *next-hop* (que nos da igual porque siempre es el mismo), su interfaz (ídem). A continuation, se extrae un fragmento.

```
Table: Routes
Destination     Gateway IP      Metric  ETX     Interface
0.0.0.0/0       10.12.11.1      2       2.000   tap0
10.12.0.2/32    10.12.11.1      8       8.220   tap0
10.12.0.9/32    10.12.11.1      6       6.080   tap0
10.12.0.10/32   10.12.11.1      2       2.000   tap0
10.12.0.38/32   10.12.11.1      12      13.236  tap0
10.12.0.47/32   10.12.11.1      12      13.236  tap0
10.12.0.52/32   10.12.11.1      4       4.000   tap0
```

Por otra parte, *Topology*, que incluye LQ, NLQ, ETX, destino y *Last-hop*. A continuación, se muestra un ejemplo:

```
Table: Topology
Dest. IP    Last hop IP LQ    NLQ   Cost
10.12.4.204 192.168.1.1 1.000 1.000 1.000
10.12.91.23 10.12.3.1   1.000 1.000 1.000
10.12.91.25 10.12.3.1   1.000 1.000 1.000
10.12.0.62  10.12.3.1   0.736 0.831 1.631
```

```
10.12.4.81  10.12.3.1   0.650 0.890 1.726
10.12.0.139 10.12.3.1   0.721 0.705 1.963
10.12.2.186 10.12.3.1   0.549 0.991 1.835
```

En cuanto a la obtención de los datos en tiempo real, es relativamente sencillo, ya que estos *plugins* proporcionan acceso simple mediante línea de comandos, por lo que se puede realizar una llamada cada *n* segundos sin ningún problema:

```
watch -n 1 ./recogeDatos.sh
```

Donde este script contiene:

```
curl http://localhost:9091/topo >"topologia/topo_`date +%d-%m-%Y-%H.%M.%S`"
```

De esta manera se han podido capturar los datos. No obstante, no se obtienen datos el 100% del tiempo, sino que hay algunos instantes en los que no se llega a escribir ningún fichero. No se ha encontrado una explicación a este fenómeno, ya que ni aumentando al máximo la prioridad del proceso se ha conseguido subsanar. Además, en otros *datasets* como el utilizado para mi Trabajo Fin de Grado también faltaban ficheros, solo que en este caso las medidas estaban tomadas cada 5 minutos y era más probable encontrar un periodo sin pérdidas.

Anexo B

Figuras adicionales del análisis descriptivo del conjunto de datos

En la Capítulo 3 se presenta el análisis descriptivo del conjunto de datos. En este Anexo se presentan figuras complementarias. En la sección 3.5 se decidió no mostrarlas, ya que no aportaban nada relevante de cara al estudio, pero se muestran aquí por completitud.

B.1. Figuras del análisis horario del primer día

Estas figuras muestran los resultados de un análisis horario realizado sobre el primer día del *dataset*.

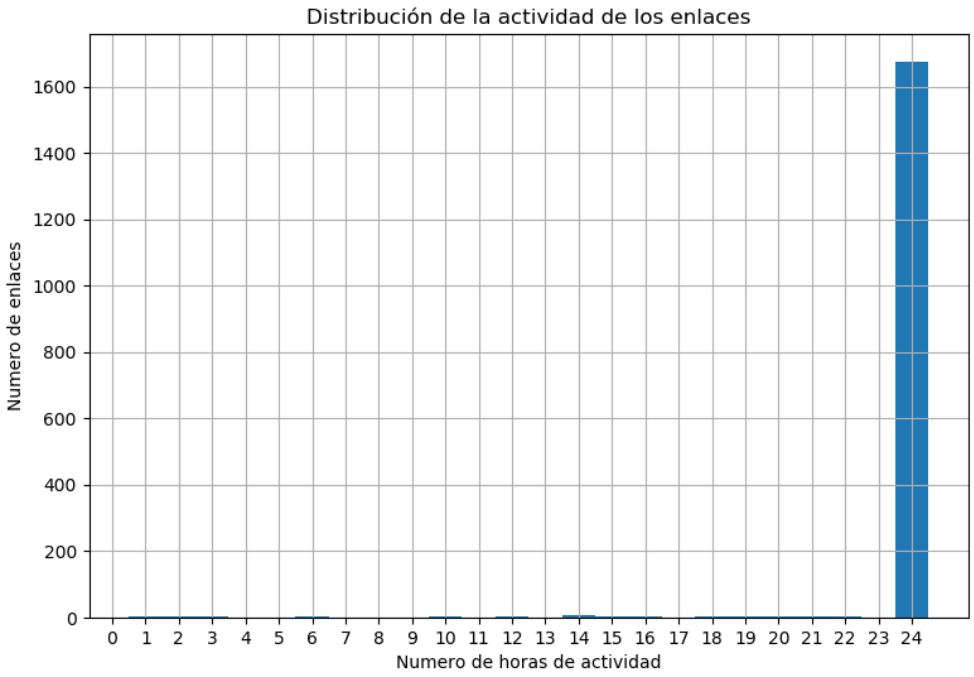

Figura 34: número de horas de actividad de los enlaces

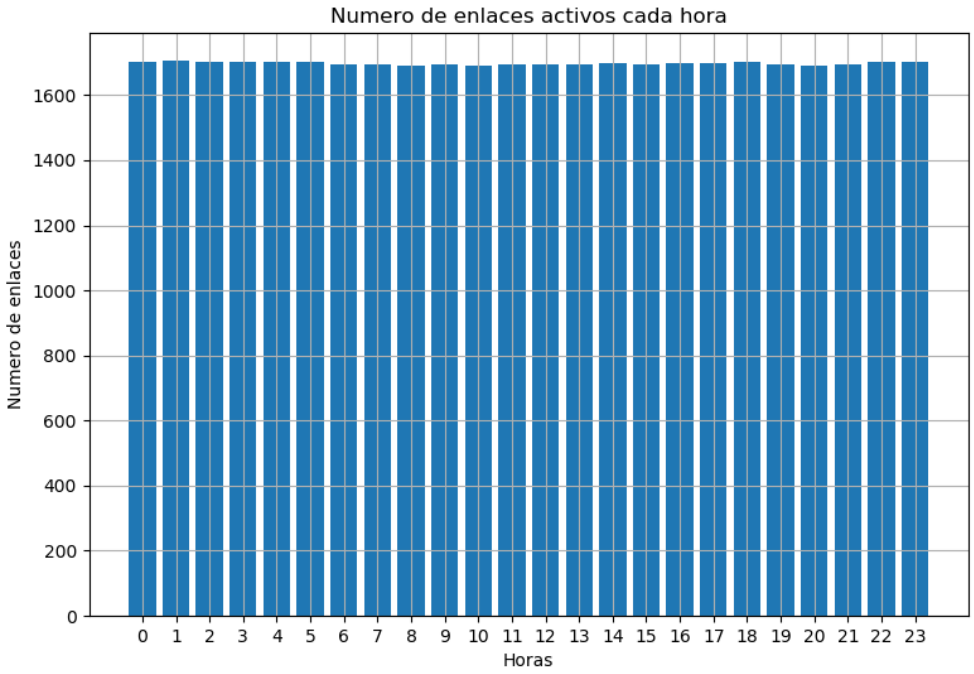

Figura 35: número de enlaces activos cada hora.

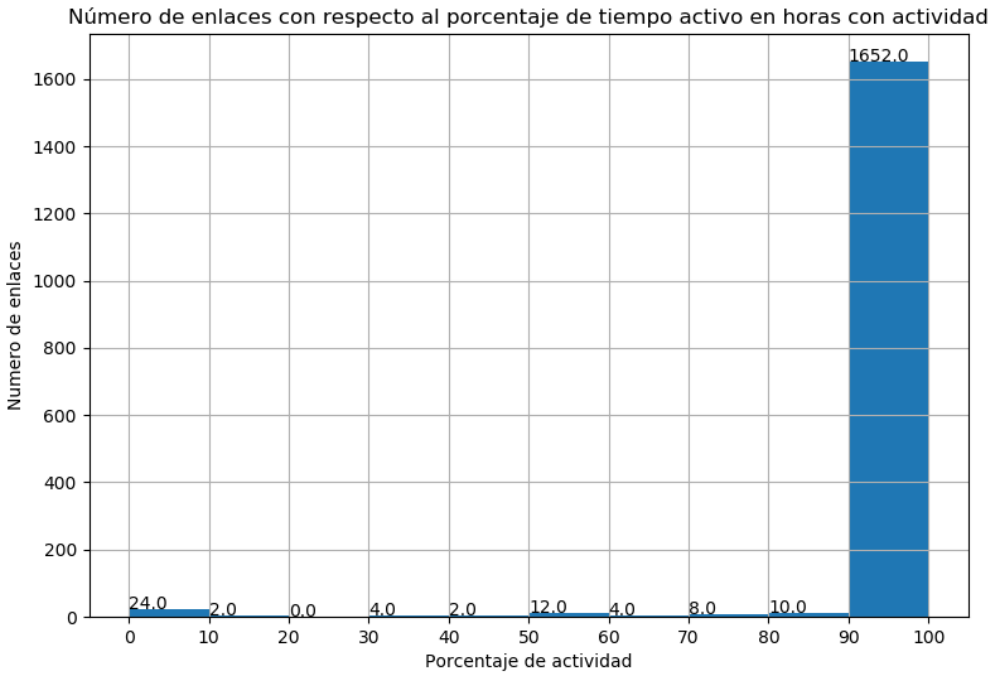

Figura 36: número de enlaces con respecto al porcentaje de tiempo activo en horas con actividad

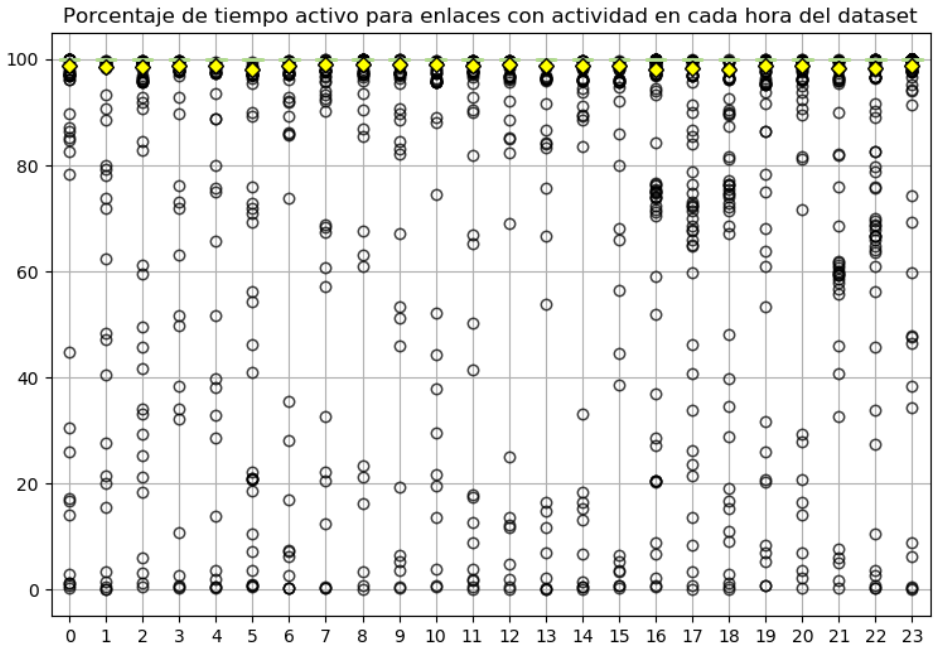

Figura 37: porcentaje de tiempo activo para enlaces con actividad en cada hora del dataset

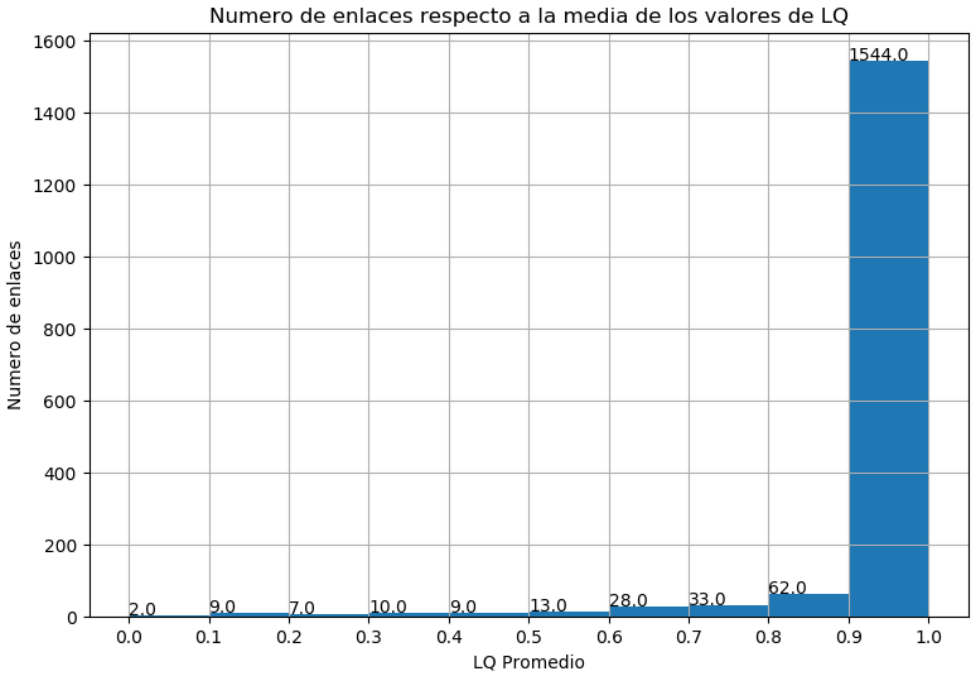

Figura 38: número de enlaces respecto a la media de los valores de LQ.

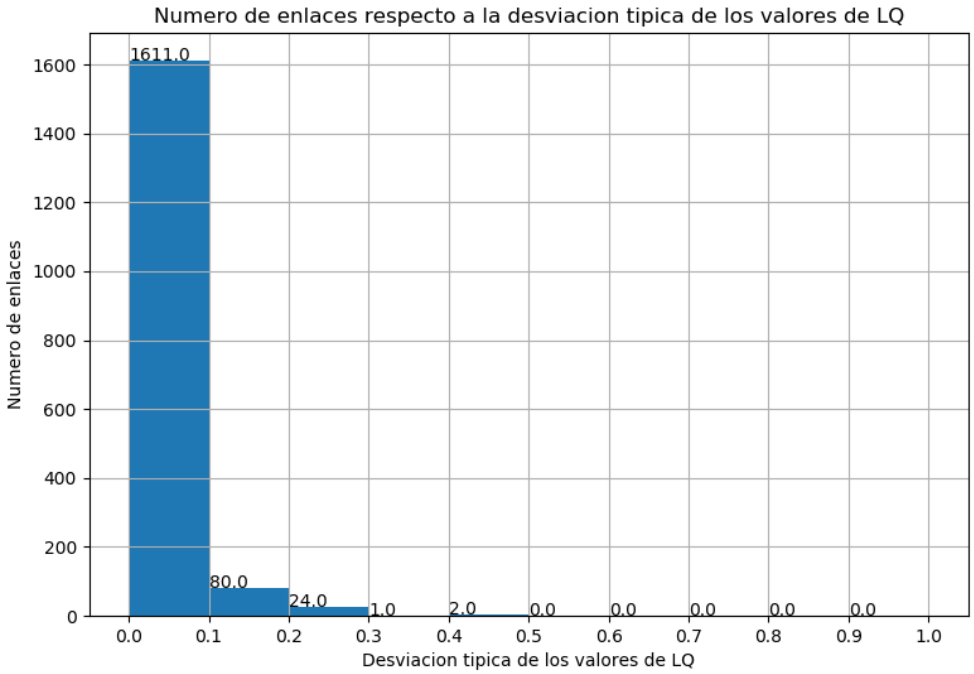

Figura 39: número de enlaces respecto a la desviación típica de valores de LQ

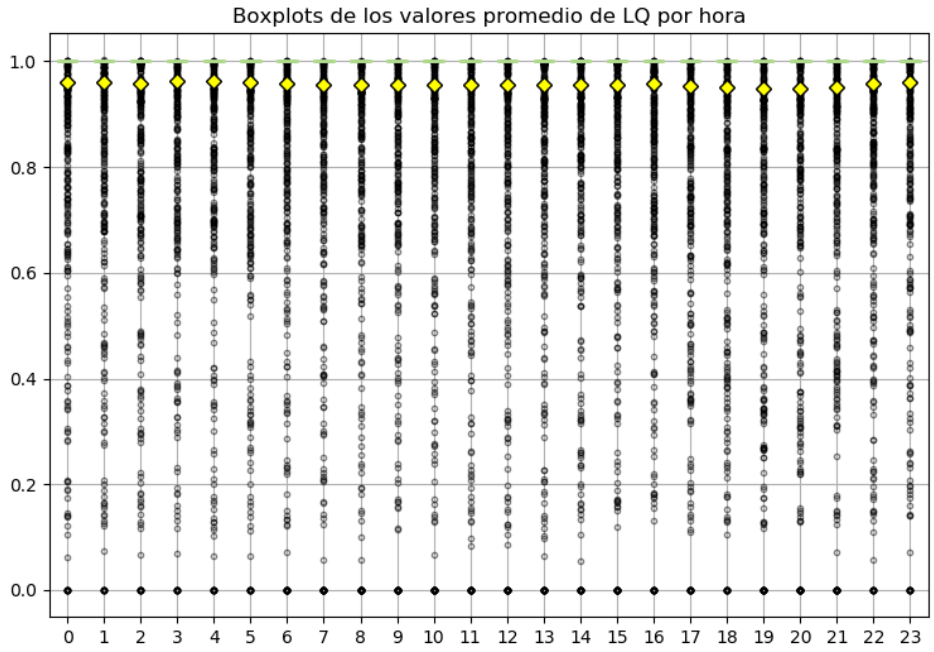

Figura 40: boxplots *de LQ promedio por hora.*

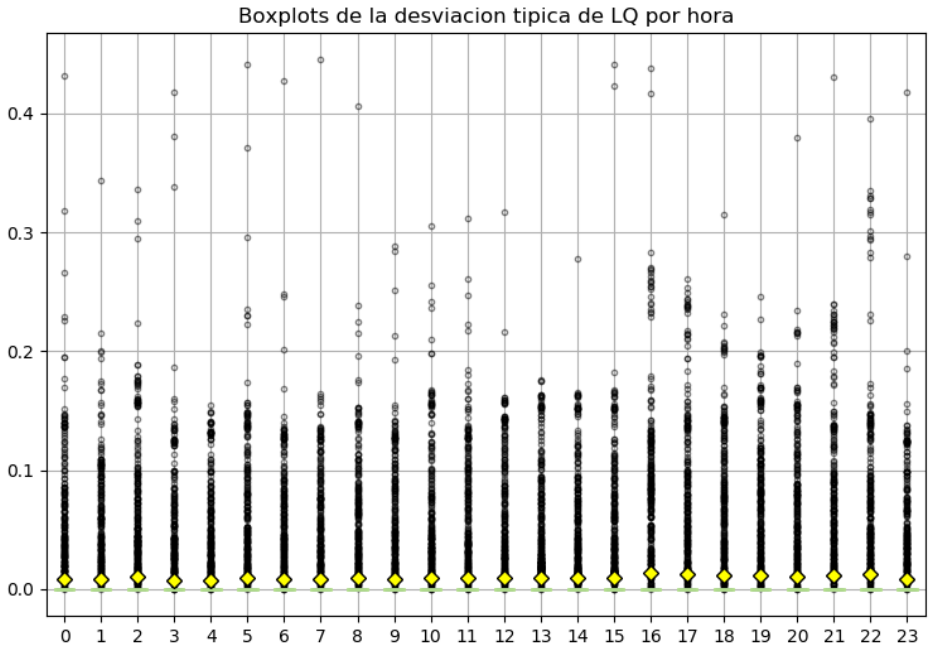

Figura 41: boxplots *de desviación típica de LQ por día.*

B.2. Comparativa de desviación típica de LQ a lo largo de los 14 días del *dataset* en los enlaces variables cada día de este

En esta figura se compara con *boxplots* la desviación típica cada hora los 14 días del *dataset*, no viéndose ninguna correlación importante horaria/semanal (la primera semana es la superior y la segunda la inferior)

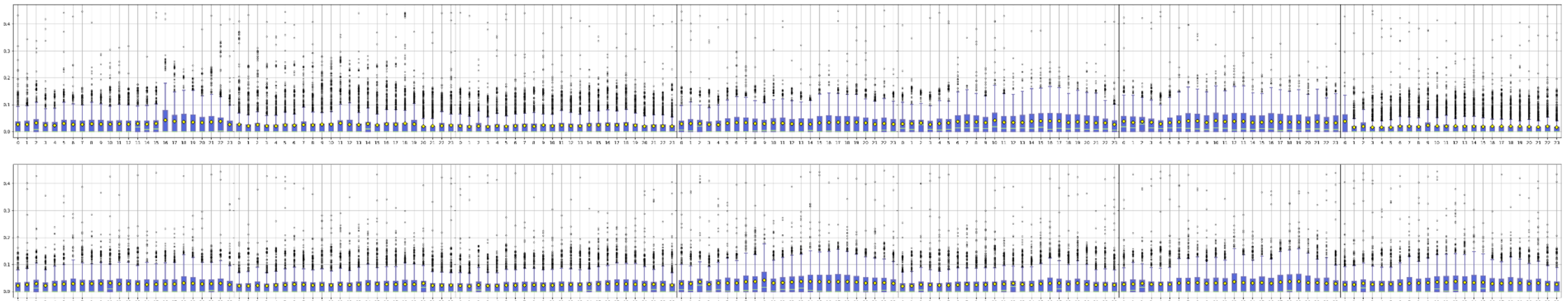

Figura 42: comparativa de la desviación típica de LQ a lo largo de los 14 días de los enlaces variables del dataset.

Anexo C

Descripción de los algoritmos de aprendizaje automático en línea y de sus parámetros

En este Anexo se describen los algoritmos de aprendizaje automático en línea utilizados, así como los parámetros que permite fijar cada uno de los algoritmos utilizados. La descripción de cada uno de ellos, así como de los valores por defecto ha sido extraído de la interfaz de usuario, ya que lamentablemente la API de Java no lo incluye.

C.1. *Perceptron* (`moa.classifiers.rules.functions.Perceptron`)

Este algoritmo se trata de una red neuronal formada por únicamente una neurona. En [60] se presentan árboles de decisión en línea que utilizan esta versión del perceptrón. Utiliza la función sigmoide en vez de un umbral. Dado que se emplea para un caso en línea, en vez de hacer actualizaciones en lote, se utiliza descenso de gradiente estocástico (con cada nueva instancia se actualiza el modelo). Para tratar con un problema multiclase, utiliza un perceptrón por clase. En nuestro caso se utiliza para un problema de regresión, y el artículo no hace ninguna mención al respecto, pero es de suponer que utiliza las técnicas habituales [61] para utilizar algoritmos de clasificación en casos de regresión. Los parámetros que nos permite modificar la interfaz son los siguientes:

- `-d learningRatio_Decay_set_constant` (valor por defecto: **desactivado**): establece si el decrecimiento del factor de aprendizaje es constante.
- `-l learningRatio` (valor por defecto: **0.025**): factor de aprendizaje constante usada para entrenar los Perceptrones en las hojas.
- `-m learningRateDecay` (valor por defecto: **0.001**): decrecimiento del factor de aprendizaje usado para entrenar el Perceptrón.
- `-e fadingFactor` (valor por defecto: **0.99**, valores que deja fijar la interfaz entre 0 y 1): factor de desvanecimiento del error acumulado del Perceptrón.
- `-r randomSeed` (valor por defecto: **1**): semilla para controlar el comportamiento aleatorio del clasificador.

C.2. *FIMT-DD* (`moa.classifiers.trees.FIMTDD`)

FIMT-DD (Fast Incremental Model Trees with Drift Detection) [62] es un algoritmo basado en los árboles modelo (*model tree*) . Un árbol modelo es un algoritmo de aprendizaje que combina un árbol de decisión con otro algoritmo, como por ejemplo regresión logística. La combinación se hace de tal manera que en cada una de las hojas del árbol se crea un modelo. Esto queda más claro en la Figura 43.

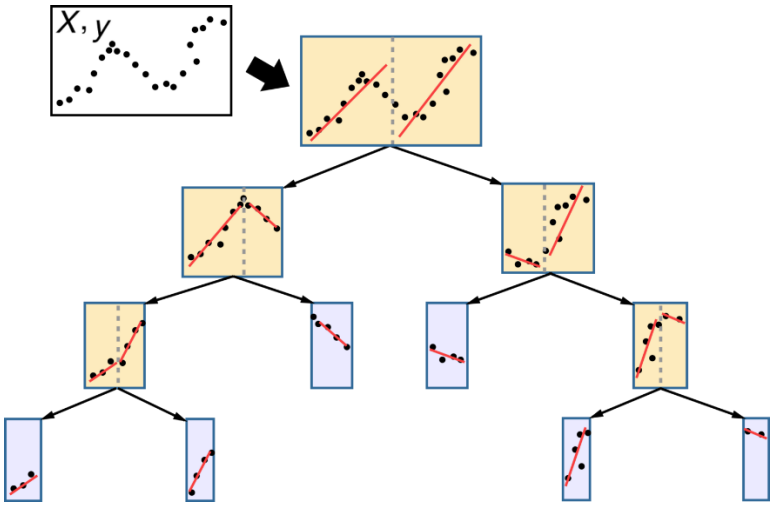

Figura 43: ejemplo de árbol modelo [63].

Los parámetros que nos permite modificar la interfaz son los siguientes:

- `-s splitCriterion` (valor por defecto: **`moa.classifiers.core.splitcriteria.VarianceReductionSplitCriterion`**): criterio de división a usar. Existen varios criterios diferentes que aquí no se detallarán.
- `-g gracePeriod` (valor por defecto: **200**, valores que deja fijar la interfaz mayores o iguales a 0): número de instancias una hoja debe observar antes de intentos de división.
- `-c splitConfidence` (valor por defecto: **1.0E-7**, valores que deja fijar la interfaz entre 0 y 1): error permitido en una decisión de división, valores cercanos a 0 requerirán más tiempo para decidir.
- `-t tieThreshold` (valor por defecto: **0.05**, valores que deja fijar la interfaz entre 0 y 1): umbral por debajo del cual una división será forzada para romper vínculos.
- `-a PageHinckleyAlpha` (valor por defecto: **0.005**, valores que deja fijar la interfaz entre 0 y 1): valor de Alpha a usar en los test de detección de cambio de Page Hinckley.

- `-h PageHinckleyThreshold` (valor por defecto: **50**, valores que deja fijar la interfaz mayores o iguales a 0): valor de umbral usado en los test de detección de cambio de Page Hinckley.
- `-f alternateTreeFadingFactor` (valor por defecto: **0.995**, valores que deja fijar la interfaz entre 0 y 1): factor de desvanecimiento usado para decidir si un árbol alternativo debería sustituir al original).
- `-y alternateTreeTmin` (valor por defecto: **150**, valores que deja fijar la interfaz mayores o iguales a 0): valor Tmin usado para decidir si un árbol alternativo debería sustituir al original.
- `-u alternateTreeTime` (valor por defecto: **1500**, valores que deja fijar la interfaz mayores o iguales a 0): el número de instancias usadas para decidir si un árbol alternativo debería ser descartado.
- `-e regressionTree` (valor por defecto: **desactivado**): construye un árbol de regresión en vez de un árbol de modelo.
- `-l learningRatio` (valor por defecto: **0.02**, valores que deja fijar la interfaz entre 0 y 1): factor de aprendizaje usado para entrenar a los perceptrones en las hojas.
- `-d learningRatioDecayFactor` (valor por defecto: **0.001**, valores que deja fijar la interfaz entre 0 y 1): decrecimiento del factor de aprendizaje (no usado cuando el factor de aprendizaje es constante).
- `-p learningRatioConst` (valor por defecto: **desactivado**): mantiene un factor de aprendizaje constante en lugar de que disminuya.

C.3. *ORTO* (`moa.classifiers.trees.ORTO`)

ORTO surge como mejora de *FIMT-DD*. *ORTO* (*Online Regression/Model Trees with Options*) [64] introduce la idea de árbol de opción (*option tree*). Un árbol de opción es una variación de árbol de decisión, en la cual además de los nodos de división (*splitting nodes*), existen otros nodos, los nodos de opción (*option nodes*), que se comportan como una función lógica OR. Es decir, cuando una instancia llega a uno de estos nodos, avanza por todas las ramas para las que se cumpla la condición especificada, llegando a múltiples hojas. La principal motivación de usar este tipo de

nodos es que elimina la necesidad de elegir el mejor atributo para realizar la división de ramas (*split*). En la Figura 44 se puede ver un ejemplo de árbol de opción.

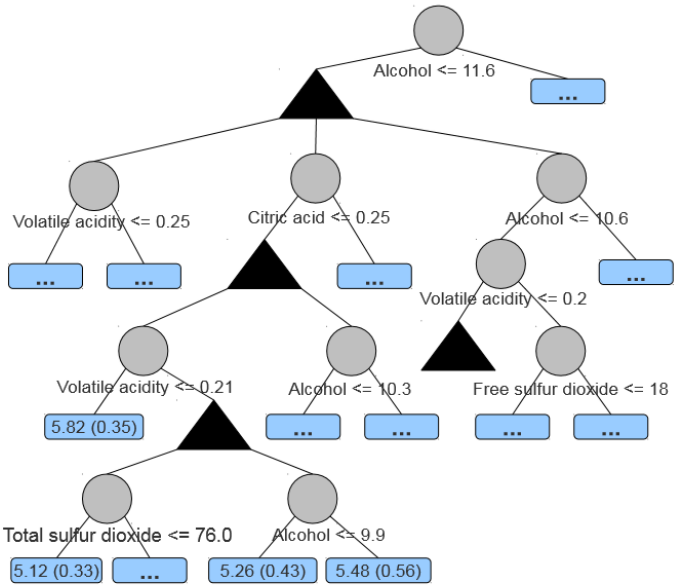

Figura 44: árbol de opción para el conjunto de datos de la calidad del vino [64].

ORTO tiene los mismos parámetros que *FIMT-DD* y, además, los siguientes:

- `-m maxTrees` (valor por defecto: **10,** valores que deja fijar la interfaz mayores o iguales a 1): número máximo de árboles contenidos en el árbol de opción.
- `-x maxOptionLevel` (valor por defecto: **10,** valores que deja fijar la interfaz mayores o iguales a 0): profundidad máxima a la que los nodos de opción pueden ser creados.
- `-z optionDecayFactor` (valor por defecto: **0.9,** valores que deja fijar la interfaz entre 0 y 1): determina cuantas opciones pueden ser seleccionadas a un nivel dado.
- `-o optionNodeAggregation` (valor por defecto: **`average`**): método de agregación usado para combinar predicciones en los nodos de opción. Otra opción: **`bestTree`**.
- `-q optionFadingFactor` (valor por defecto: **0.9995**, valores que deja fijar la interfaz entre 0 y 1): el factor de desvanecimiento usado para comprar subárboles en un nodo de opción). (Parece equivaler al parámetro `alternateTreeFadingFactor` de FIMT-DD.)

C.4. *AMRulesRegressor* (`moa.classifiers.rules.AMRulesRegressor`)

AMRulesRegressor (*Adaptative Model Rules from High-Speed Data Streams*) es un algoritmo incremental para aprender un modelo de reglas [65]. Un modelo de reglas es similar a un árbol de regresión o decisión, de hecho, muchas veces se construyen a partir de estos últimos. Tienen ventajas parecidas, como la robustez ante datos poco comunes. Los parámetros que nos permite modificar la interfaz son los siguientes:

- `-c splitConfidence` (valor por defecto: **1.0E-7**, valores que deja fijar la interfaz entre 0 y 1): parámetro de la cota de Hoeffding. Error permitido en una decisión de división, valores cercanos a 0 requerirán más tiempo para decidir.
- `-t tieThreshold` (valor por defecto: **0.05**, valores que deja fijar la interfaz entre 0 y 1): parámetro de la cota de Hoeffding. Umbral por debajo del cual una división será forzada para romper vínculos.
- `-g gracePeriod` (valor por defecto: **200**, valores que deja fijar la interfaz mayores o iguales a 0): parámetro de la cota de Hoeffding. Número de instancias una hoja debe observar antes de intentos de división.
- `-L learnerOption` (valor por defecto: **`moa.classifiers.rules.multilabel.functions.AdaptiveMultiTargetRegressor`**): algoritmo de aprendizaje. Existen varios algoritmos diferentes que aquí no se detallarán.
- `-U setUnorderedRulesOn` (valor por defecto: **desactivado**): activa las reglas sin orden.
- `-D dropOldRuleAfterExpansion` (valor por defecto: **desactivado**): descarta la regla antigua si se expande. Por defecto, la regla se guarda para el conjunto de salidas no seleccionadas para expansión.
- `-H changeDetector` (valor por defecto: **`PageHinkleyDM -d 0.05 -l 35.0`**): Detector de cambios. Existen varios detectores diferentes que aquí no se detallarán.
- `-A anomalyDetector` (valor por defecto: **`moa.classifiers.rules.core.anomalydetection.OddsRatioScore`**): detector de anomalías. Existen varios detectores diferentes que aquí no se detallarán.
- `-S splitCriterionOption` (valor por defecto: **`moa.classifiers.rules.multilabel.core.splitcriteria.MultiTargetVarianceRatio`**): criterio de división usado para

evaluar el mérito de una división. Existen varios criterios diferentes que aquí no se detallarán.

- `-e errorMeasurer` (valor por defecto: **`moa.classifiers.rules.multilabel.errormeasurers.RelativeMeanAbsoluteDeviationMT`**): medida de error para decidir que algoritmo de aprendizaje debería predecir. Existen varios medidores diferentes que aquí no se detallarán.
- `-w weightedVoteOption` (valor por defecto: **`moa.classifiers.rules.multilabel.core.voting.InverseErrorWeightedVoteMultiLabel`**): tipo de voto ponderado. Existen varios tipos diferentes que aquí no se detallarán.
- `-y numericObserver` (valor por defecto: **`MultiLabelBSTree`**): observador numérico. Existen varios observadores diferentes que aquí no se detallarán.
- `-z nominalObserver` (valor por defecto: **`MultiLabelNominalAttributeObserver`**): observador nominal. Existen varios observadores diferentes que aquí no se detallarán.
- `-v verbosity` (valor por defecto: 1): controla el nivel de verbosidad de la salida, de 1 (menos) a 5 (más).
- `-O outputSelector` (valor por defecto: **`moa.classifiers.rules.multilabel.outputselectors.SelectAllOutputs`**): selector de atributos de salida. Existen varios selectores diferentes que aquí no se detallarán.
- `-I inputSelector` (valor por defecto: **`moa.classifiers.rules.multilabel.inputselectors.SelectAllInputs`**): selector de atributos de entrada. Existen varios selectores diferentes que aquí no se detallarán.
- `-r randomSeedOption` (valor por defecto: 1): semilla para controlar el comportamiento aleatorio.
- `-F featureRanking` (valor por defecto: **`moa.classifiers.rules.featureranking.NoFeatureRanking`**): algoritmo de clasificación (en el sentido de orden) de características. Existen varios selectores algoritmos que aquí no se detallarán.

C.5. *Fading Target Mean* (`moa.classifiers.rules.functions.FadingTargetMean`)

Este algoritmo es el más simple, y no existe documentación como tal para él, ya que se utiliza en otros algoritmos. No obstante, leyendo su código fuente, se puede entender fácilmente como funciona. El algoritmo del que deriva, *TargetMean*,

simplemente calcula el promedio de todos los valores que ha ido recibiendo. *FadingTargetMean* añade un parámetro (`-f`) que hace que el pasado tenga mayor o menor relevancia. Los parámetros que nos permite modificar la interfaz son los siguientes:

- `-f fadingFactor` (valor por defecto: **0.99**, valores que deja fijar la interfaz entre 0 y 1): factor de desvanecimiento para el error acumulado de FadingTargetMean.
- `-e fadingErrorFactor` (valor por defecto: **0.99**, valores que deja fijar la interfaz entre 0 y 1): factor del desvanecimiento del error para el error acumulado de TargetMean.

Anexo D

Resultados adicionales de los experimentos

En este Anexo se incluyen resultados adicionales de los experimentos con algoritmos de aprendizaje automático en línea. En negrita se marca el mejor resultado de cada algoritmo y subrayado en verde el mejor en general.

D.1. Resultados iniciales divididos por *clusters*

En esta sección se muestran los resultados relativos a los experimentos iniciales dividiendo los resultados en diferentes *clusters* en función de la desviación típica. Los *clusters* son: los 4 cuartiles, percentiles 75-80, 80-85, 85-90, 90-95, 95-100 y 99-100.

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.00011	**0.00021**	**0.00180**	**0.00246**	0.00051	**0.00004**
Lag 2	**0.00010**	0.00021	0.00659	0.02238	0.00150	0.00005
Lag 3	0.00011	0.00021	0.01853	0.08987	0.00091	0.00007
Lag 4	0.00011	0.00021	0.03058	0.09444	**0.00013**	0.00008
Lag 5	0.00012	0.00021	0.04079	0.09549	0.00251	0.00009
Lag 6	0.00012	0.00021	0.07886	0.10301	0.00016	0.00010
Lag 7	0.00012	0.00021	0.08104	0.09888	0.00018	0.00011
Lag 8	0.00013	0.00021	0.08241	0.10140	0.00020	0.00012
Lag 9	0.00013	0.00021	0.10474	0.10285	0.00022	0.00013
Lag 10	0.00013	0.00021	0.07992	0.10145	0.00073	0.00013
Lag 11	0.00015	0.00021	0.12528	0.09304	0.00099	0.00014
Lag 12	0.00013	0.00021	0.09879	0.09939	0.00127	0.00014

Tabla 22: resultados iniciales de los experimentos. 5 algoritmos + baseline *sobre los enlaces variables del primer cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.*

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.00044	**0.00086**	0.00285	**0.00399**	0.00079	**0.00018**
Lag 2	**0.00040**	0.00086	**0.00214**	0.02333	0.00176	0.00025
Lag 3	0.00042	0.00086	0.01150	0.04356	0.00122	0.00032
Lag 4	0.00044	0.00086	0.03668	0.05337	**0.00048**	0.00038
Lag 5	0.00045	0.00086	0.04032	0.07094	0.00282	0.00043
Lag 6	0.00046	0.00086	0.10056	0.07196	0.00057	0.00048
Lag 7	0.00047	0.00086	0.10898	0.07690	0.00061	0.00052
Lag 8	0.00048	0.00086	0.11687	0.08892	0.00067	0.00055
Lag 9	0.00048	0.00086	0.11845	0.07108	0.00071	0.00058
Lag 10	0.00049	0.00086	0.13050	0.07157	0.00124	0.00061
Lag 11	0.00052	0.00086	0.10450	0.07514	0.00153	0.00063
Lag 12	0.00049	0.00086	0.10951	0.07221	0.00185	0.00064

Tabla 23: resultados iniciales de los experimentos. 5 algoritmos + baseline *sobre los enlaces variables del segundo cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.*

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.01243	0.02504	**0.01256**	**0.02498**	**0.01198**	**0.00689**
Lag 2	**0.01160**	0.02504	0.01497	0.18565	0.01238	0.00885
Lag 3	0.01168	0.02502	0.02060	0.15977	0.01216	0.01042
Lag 4	0.01168	**0.02499**	0.01853	0.14736	0.01207	0.01182
Lag 5	0.01179	0.02500	0.02174	0.15205	0.01267	0.01309
Lag 6	0.01188	0.02500	0.03370	0.14764	0.01242	0.01420
Lag 7	0.01200	0.02500	0.04181	0.15665	0.01230	0.01519
Lag 8	0.01204	0.02500	0.03073	0.14395	0.01247	0.01606
Lag 9	0.01216	0.02499	0.03614	0.16368	0.01252	0.01682
Lag 10	0.01218	0.02500	0.03701	0.15578	0.01260	0.01752
Lag 11	0.01235	0.02500	0.04452	0.16702	0.01282	0.01816
Lag 12	0.01253	0.02499	0.04976	0.15520	0.01289	0.01875

Tabla 24: resultados iniciales de los experimentos. 5 algoritmos + baseline *sobre los enlaces variables del tercer cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.*

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.03442	0.06498	**0.03505**	**0.06601**	0.03551	**0.01834**
Lag 2	0.03160	0.06499	0.04277	0.17152	0.03456	0.02392
Lag 3	0.03182	0.06500	0.04733	0.16877	**0.03421**	0.02845
Lag 4	0.03172	0.06499	0.04803	0.15683	0.03437	0.03245
Lag 5	**0.03144**	0.06501	0.04304	0.16749	0.03605	0.03592
Lag 6	0.03215	0.06503	0.04802	0.16448	0.03494	0.03894
Lag 7	0.03316	0.06506	0.04554	0.17571	0.03429	0.04149
Lag 8	0.03376	0.06504	0.04758	0.17336	0.03500	0.04352
Lag 9	0.03338	0.06479	0.04751	0.19162	0.03504	0.04539
Lag 10	0.03342	0.06468	0.04913	0.18687	0.03479	0.04706
Lag 11	0.03411	**0.06464**	0.05024	0.18365	0.03517	0.04851
Lag 12	0.03512	0.06468	0.05193	0.18961	0.03559	0.04979

Tabla 25: resultados iniciales de los experimentos. 5 algoritmos + baseline *sobre los enlaces variables del cuarto cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.*

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.02458	0.04938	**0.02415**	**0.04909**	**0.02407**	**0.01324**
Lag 2	**0.02319**	0.04932	0.02713	0.16226	0.02440	0.01759
Lag 3	0.02332	0.04933	0.03290	0.14988	0.02422	0.02115
Lag 4	0.02410	0.04932	0.03172	0.16609	0.02665	0.02421
Lag 5	0.02386	0.04933	0.02810	0.16231	0.02669	0.02685
Lag 6	0.02484	0.04933	0.03474	0.15591	0.02704	0.02913
Lag 7	0.02523	0.04933	0.03273	0.15207	0.02643	0.03113
Lag 8	0.02552	0.04928	0.03303	0.15132	0.02649	0.03285
Lag 9	0.02568	0.04928	0.03340	0.17915	0.02665	0.03436
Lag 10	0.02532	**0.04931**	0.03988	0.17433	0.02596	0.03570
Lag 11	0.02613	0.04932	0.04048	0.16637	0.02613	0.03690
Lag 12	0.02579	0.04935	0.04167	0.19311	0.02626	0.03801

Tabla 26: resultados iniciales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 75 y 80 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.

Lag\ Algoritmo	AMRR	FTM	FIMT-DD	ORTO	Perceptron	Baseline
Lag 1	0.03106	0.05615	**0.03444**	**0.05635**	0.03125	**0.02159**
Lag 2	0.02991	0.05615	0.03912	0.18440	0.03156	0.02493
Lag 3	0.03010	0.05615	0.04829	0.20313	0.03111	0.02790
Lag 4	0.02996	0.05615	0.04279	0.18694	0.03074	0.03118
Lag 5	0.02992	0.05615	0.04040	0.21825	0.03119	0.03396
Lag 6	**0.02985**	**0.05614**	0.04387	0.22900	0.03096	0.03635
Lag 7	0.03005	0.05614	0.04454	0.22503	**0.03070**	0.03857
Lag 8	0.03001	0.05614	0.04826	0.24546	0.03089	0.04049
Lag 9	0.03011	0.05614	0.04909	0.23527	0.03095	0.04210
Lag 10	0.02998	0.05614	0.04806	0.23702	0.03112	0.04356
Lag 11	0.03021	0.05614	0.04931	0.23504	0.03128	0.04486
Lag 12	0.03037	0.05614	0.05066	0.21986	0.03142	0.04601

Tabla 27: resultados iniciales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 80 y 85 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.

Lag\ Algoritmo	AMRR	FTM	FIMT-DD	ORTO	Perceptron	Baseline
Lag 1	0.04057	0.07561	**0.03975**	**0.06915**	**0.04086**	**0.02001**
Lag 2	0.03856	0.07553	0.04461	0.13383	0.04134	0.02743
Lag 3	0.03796	0.07541	0.04422	0.09911	0.04192	0.03410
Lag 4	0.03701	0.07522	0.05170	0.10478	0.04424	0.04022
Lag 5	0.03664	0.07526	0.04278	0.10885	0.04211	0.04578
Lag 6	0.03575	0.07524	0.05327	0.10562	0.04519	0.05079
Lag 7	0.03716	0.07540	0.05035	0.13458	0.04334	0.05480
Lag 8	0.03692	0.07531	0.05170	0.12662	0.04404	0.05747
Lag 9	0.03607	0.07407	0.04995	0.15132	0.04294	0.05988
Lag 10	**0.03552**	0.07398	0.04607	0.11898	0.04192	0.06187
Lag 11	0.03592	0.07387	0.04807	0.12217	0.04273	0.06354
Lag 12	0.03627	**0.07372**	0.04877	0.13075	0.04233	0.06495

Tabla 28: resultados iniciales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 85 y 90 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.03497	0.07405	**0.03414**	**0.07857**	0.03894	**0.02019**
Lag 2	0.03277	0.07405	0.03719	0.19354	0.03783	0.02682
Lag 3	**0.03250**	0.07405	0.03922	0.19977	0.03805	0.03241
Lag 4	0.03300	0.07405	0.04156	0.15366	**0.03778**	0.03711
Lag 5	0.03302	0.07404	0.03946	0.16763	0.03790	0.04096
Lag 6	0.03291	0.07402	0.04464	0.16739	0.03861	0.04423
Lag 7	0.03388	0.07396	0.03890	0.18399	0.03792	0.04697
Lag 8	0.03385	0.07394	0.04230	0.17344	0.03804	0.04925
Lag 9	0.03326	0.07394	0.04138	0.21888	0.03826	0.05120
Lag 10	0.03310	**0.07393**	0.04482	0.23212	0.03837	0.05289
Lag 11	0.03378	0.07393	0.04739	0.21386	0.03908	0.05440
Lag 12	0.03451	0.07393	0.04893	0.22367	0.03912	0.05576

Tabla 29: resultados iniciales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 90 y 95 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.04115	**0.07009**	**0.04296**	**0.07699**	0.04264	**0.01673**
Lag 2	0.03382	0.07026	0.06584	0.18224	0.03790	0.02297
Lag 3	**0.03356**	0.07041	0.07189	0.18949	0.03602	0.02687
Lag 4	0.03472	0.07055	0.07249	0.17084	**0.03282**	0.02983
Lag 5	0.03485	0.07066	0.06445	0.17832	0.04257	0.03243
Lag 6	0.03754	0.07077	0.06378	0.16240	0.03327	0.03462
Lag 7	0.03962	0.07085	0.06133	0.18142	0.03339	0.03643
Lag 8	0.04262	0.07089	0.06274	0.16832	0.03589	0.03804
Lag 9	0.04185	0.07085	0.06380	0.17206	0.03670	0.03993
Lag 10	0.04324	0.07040	0.06673	0.16949	0.03683	0.04180
Lag 11	0.04456	0.07027	0.06591	0.17859	0.03692	0.04339
Lag 12	0.04868	0.07058	0.06949	0.17854	0.03905	0.04477

Tabla 30: resultados iniciales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 95 y 100 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.

Lag*\ *Algoritmo	**AMRR**	**FTM**	**FIMT-DD**	**ORTO**	**Perceptron**	**Baseline**
Lag 1	0.10968	**0.14283**	**0.10007**	**0.12885**	0.10089	**0.04537**
Lag 2	0.09190	0.14359	0.14446	0.23913	0.10590	0.06177
Lag 3	**0.09054**	0.14433	0.15378	0.27564	0.09722	0.06898
Lag 4	0.09496	0.14502	0.15732	0.21590	0.08397	0.07320
Lag 5	0.09404	0.14565	0.14823	0.20038	0.12613	0.07718
Lag 6	0.10673	0.14620	0.13531	0.18360	**0.08355**	0.08029
Lag 7	0.11401	0.14661	0.13071	0.20725	0.08513	0.08248
Lag 8	0.12620	0.14683	0.13432	0.22024	0.09503	0.08443
Lag 9	0.12089	0.14668	0.13304	0.18491	0.09766	0.08830
Lag 10	0.12858	0.14460	0.12352	0.23787	0.09712	0.09257
Lag 11	0.13178	0.14409	0.12227	0.26516	0.09532	0.09592
Lag 12	0.14689	0.14561	0.13319	0.27884	0.10329	0.09865

Tabla 31: resultados iniciales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 99 y 100 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 12 unidades.

D.2. Resultados finales divididos por *clusters*

En esta sección se muestran los resultados relativos a los experimentos finales (algoritmos optimizados) dividiendo los resultados en diferentes *clusters* en función de la desviación típica. Los *clusters* son: los 4 cuartiles, percentiles 75-80, 80-85, 85-90, 90-95, 95-100 y 99-100.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.00008**	**0.00016**	**0.00016**	0.00004	0.00021	**0.00004**
Lag 2	0.00010	0.00019	0.00019	0.00004	**0.00017**	0.00005
Lag 3	0.00011	0.00021	0.00021	0.00004	0.00017	0.00007
Lag 4	0.00012	0.00023	0.00023	0.00004	0.00018	0.00008
Lag 5	0.00014	0.00028	0.00028	0.00004	0.00018	0.00009
Lag 6	0.00015	0.00080	0.00080	0.00004	0.00018	0.00010

Tabla 32: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables del primer cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.00034**	**0.00048**	**0.00048**	**0.00018**	0.00056	**0.00018**
Lag 2	0.00039	0.00055	0.00055	0.00018	0.00042	0.00025
Lag 3	0.00044	0.00060	0.00060	0.00018	0.00042	0.00032
Lag 4	0.00050	0.00062	0.00062	0.00018	0.00042	0.00038
Lag 5	0.00055	0.00067	0.00067	0.00018	**0.00041**	0.00043
Lag 6	0.00062	0.00191	0.00191	0.00018	0.00043	0.00048

Tabla 33: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables del segundo cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.01145**	**0.01229**	**0.01229**	0.00699	0.01220	**0.00689**
Lag 2	0.01252	0.01397	0.01397	0.00699	0.01062	0.00885
Lag 3	0.01486	0.01497	0.01497	0.00699	0.01054	0.01042
Lag 4	0.01757	0.01485	0.01485	**0.00698**	0.01042	0.01182
Lag 5	0.02056	0.01421	0.01421	0.00698	0.01039	0.01309
Lag 6	0.02390	0.01814	0.01814	0.00699	**0.01035**	0.01420

Tabla 34: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables del tercer cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.03214**	**0.03483**	**0.03483**	**0.01847**	0.03103	**0.01834**
Lag 2	0.03493	0.03923	0.03923	0.01848	0.02678	0.02392
Lag 3	0.04233	0.04174	0.04174	0.01850	0.02635	0.02845
Lag 4	0.04991	0.04288	0.04288	0.01852	0.02612	0.03245
Lag 5	0.05727	0.03949	0.03949	0.01853	0.02546	0.03592
Lag 6	0.06532	0.05386	0.05386	0.01855	**0.02537**	0.03894

Tabla 35: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables del cuarto cuartil (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.02241**	**0.02484**	**0.02484**	0.01324	0.02252	**0.01324**
Lag 2	0.02615	0.02820	0.02820	**0.01322**	0.01962	0.01759
Lag 3	0.03158	0.03044	0.03044	0.01322	0.01929	0.02115
Lag 4	0.03755	0.03138	0.03138	0.01322	0.01924	0.02421
Lag 5	0.04285	0.02759	0.02759	0.01322	0.01906	0.02685
Lag 6	0.04808	0.03450	0.03450	0.01322	**0.01911**	0.02913

Tabla 36: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 75 y 80 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.03065**	**0.03225**	**0.03225**	**0.02159**	0.03076	**0.02159**
Lag 2	0.03120	0.03489	0.03489	0.02159	0.02844	0.02493
Lag 3	0.03589	0.03636	0.03636	0.02159	0.02858	0.02790
Lag 4	0.04122	0.03421	0.03421	0.02159	0.02819	0.03118
Lag 5	0.04697	0.03333	0.03333	0.02159	0.02803	0.03396
Lag 6	0.05304	0.03785	0.03785	0.02159	**0.02790**	0.03635

Tabla 37: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 80 y 85 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.03548**	**0.04210**	**0.04210**	0.02001	0.03806	**0.02001**
Lag 2	0.03936	0.04902	0.04902	0.01997	0.03480	0.02743
Lag 3	0.04521	0.04952	0.04952	0.01999	0.03435	0.03410
Lag 4	0.05240	0.05637	0.05637	0.01995	0.03349	0.04022
Lag 5	0.05881	0.04672	0.04672	0.01997	0.03090	0.04578
Lag 6	0.06690	0.06136	0.06136	**0.01994**	**0.03043**	0.05079

Tabla 38: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 85 y 90 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.03528**	**0.03876**	0.03876	0.02019	0.03329	**0.02019**
Lag 2	0.03993	0.04230	0.04230	0.02019	0.02912	0.02682
Lag 3	0.04741	0.04522	0.04522	0.02019	0.02867	0.03241
Lag 4	0.05580	0.04401	0.04401	**0.02018**	0.02892	0.03711
Lag 5	0.06355	0.04046	**0.04046**	0.02018	0.02845	0.04096
Lag 6	0.07316	0.04812	0.04812	0.02018	**0.02841**	0.04423

Tabla 39: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 90 y 95 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag*\ *Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.03718**	**0.03653**	**0.03653**	**0.01735**	0.03078	**0.01673**
Lag 2	0.03829	0.04218	0.04218	0.01745	0.02201	0.02297
Lag 3	0.05202	0.04766	0.04766	0.01755	0.02096	0.02687
Lag 4	0.06316	0.04915	0.04915	0.01765	0.02086	0.02983
Lag 5	0.07484	0.04999	0.04999	0.01774	**0.02091**	0.03243
Lag 6	0.08622	0.08897	0.08897	0.01785	0.02102	0.03462

Tabla 40: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 95 y 100 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

Lag\ Algoritmo	**Perceptron**	**FIMT-DD**	**ORTO**	**FTM**	**AMRR**	**Baseline**
Lag 1	**0.07242**	**0.09270**	**0.09270**	**0.04537**	0.06465	**0.04537**
Lag 2	0.07818	0.10830	0.10830	0.04582	0.05069	0.06177
Lag 3	0.10129	0.12396	0.12396	0.04628	**0.04764**	0.06898
Lag 4	0.11204	0.12447	0.12447	0.04675	0.04813	0.07320
Lag 5	0.12880	0.12476	0.12476	0.04723	0.04857	0.07718
Lag 6	0.14326	0.15302	0.15302	0.04772	0.04877	0.08029

Tabla 41: resultados finales de los experimentos. 5 algoritmos + baseline sobre los enlaces variables entre los percentiles 99 y 100 (respecto a su desviación típica) en el primer día, con ventana temporal de 1 a 6 unidades.

D.3. Resultados de predicción a varios instantes vista

En esta sección se muestran los resultados relativos a los experimentos de predicción a varios instantes vista dividiendo los resultados en diferentes *clusters* en función de la desviación típica. Los *clusters* son: los 4 cuartiles, percentiles 75-80, 80-85, 85-90, 90-95, 95-100 y 99-100.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	0.004	0.007	0.01	0.013	0.015	0.017	0.018	0.019	0.019	0.02	0.02	0.02
FTM (0.1)	0.004	0.007	0.01	0.013	0.015	0.017	0.018	0.019	0.019	0.02	0.02	0.02
FTM(0.5)	0.006	0.009	0.012	0.014	0.016	0.017	0.018	0.019	0.019	0.02	0.02	0.02
Perceptron	0.008	0.011	0.014	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
FIMT-DD	0.03	0.029	0.029	0.038	0.041	0.044	0.045	0.046	0.047	0.047	0.047	0.047
Baseline	0.004	0.007	0.01	0.013	0.015	0.017	0.018	0.019	0.019	0.02	0.02	0.02

Tabla 42: resultados de los experimentos sobre los enlaces variables del primer día (primer cuartil respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1	1	1	1	1	1	1	1	1	1	1	1
FTM (0.1)	1	1	1	1	1	1	1	1	1	1	1	1
FTM(0.5)	1	1	1	1	1	1	1	1	1	1	1	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	1	1	1	1

Tabla 43: valores de ventana para los resultados de la Tabla 42.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	0.018	0.033	0.047	0.059	0.069	0.077	0.083	0.085	0.087	0.089	0.091	0.093
FTM (0.1)	0.019	0.034	0.048	0.059	0.069	0.077	0.082	0.085	0.088	0.089	0.091	0.093
FTM(0.5)	0.03	0.043	0.055	0.065	0.073	0.079	0.083	0.086	0.088	0.09	0.091	0.092
Perceptron	0.034	0.05	0.062	0.066	0.069	0.07	0.07	0.071	0.071	0.072	0.072	0.072
FIMT-DD	0.064	0.082	0.094	0.125	0.140	0.155	0.159	0.161	0.163	0.163	0.163	0.163
Baseline	0.018	0.033	0.047	0.059	0.069	0.077	0.083	0.085	0.087	0.089	0.091	0.093

Tabla 44: resultados de los experimentos sobre los enlaces variables del primer día (segundo cuartil respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1	1	1	1	1	1	1	1	1	1	1	1
FTM (0.1)	1	1	1	1	1	1	1	1	1	1	1	1
FTM(0.5)	1	1	1	1	1	1	1	1	1	1	1	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	1	1	1	1

Tabla 45: valores de ventana para los resultados de la Tabla 44.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	0.698	1.155	1.485	1.779	2.028	2.223	2.396	2.542	2.667	2.773	2.876	2.970
FTM (0.1)	0.734	1.174	1.5	1.79	2.034	2.226	2.397	2.54	2.663	2.770	2.872	2.963
FTM(0.5)	0.98	1.343	1.634	1.889	2.1	2.274	2.426	2.557	2.667	2.767	2.856	2.932
Perceptron	1.145	1.518	1.827	2.055	2.246	2.395	2.514	2.592	2.662	2.714	2.754	2.788
FIMT-DD	1.251	1.554	1.808	2.018	2.154	2.286	2.393	2.472	2.535	2.616	2.659	2.697
Baseline	0.699	1.156	1.486	1.779	2.028	2.223	2.396	2.542	2.661	2.710	2.759	2.807

Tabla 46: resultados de los experimentos sobre los enlaces variables del primer día (tercer cuartil respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	4	2	2	1	1	1	1	1	1	4	2	11
FTM (0.1)	2	3	1	1	1	12	1	1	6	4	2	12
FTM(0.5)	4	3	2	1	1	1	2	1	6	3	4	12
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	2	2	2	2

Tabla 47: valores de ventana para los resultados de la Tabla 46.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1.845	3.074	3.957	4.737	5.348	5.825	6.155	6.392	6.673	6.940	7.119	7.273
FTM (0.1)	1.954	3.140	4.011	4.773	5.368	5.831	6.149	6.389	6.670	6.929	7.107	7.258
FTM(0.5)	2.680	3.645	4.403	5.040	5.534	5.914	6.194	6.439	6.690	6.905	7.060	7.182
Perceptron	3.212	4.244	5.048	5.515	5.846	6.104	6.270	6.364	6.509	6.610	6.724	6.760
FIMT-DD	3.453	4.328	4.591	5.393	5.664	5.813	6.004	6.217	6.382	6.535	6.646	6.765
Baseline	1.847	3.077	3.976	4.764	5.370	5.848	6.174	6.403	6.677	6.682	6.666	6.858

Tabla 48: resultados de los experimentos sobre los enlaces variables del primer día (cuarto cuartil respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	11	12	10	12	5	12	4	4	2	1	2	1
FTM (0.1)	11	12	10	8	5	6	5	4	2	1	2	1
FTM(0.5)	11	10	10	8	8	6	5	4	2	1	2	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	1	2	2	2

Tabla 49: valores de ventana para los resultados de la Tabla 48.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1.321	2.289	3.013	3.612	4.099	4.492	4.805	5.059	5.292	5.504	5.700	5.883
FTM (0.1)	1.402	2.338	3.048	3.635	4.114	4.499	4.806	5.057	5.287	5.498	5.693	5.873
FTM(0.5)	1.948	2.713	3.321	3.828	4.242	4.574	4.850	5.082	5.292	5.489	5.664	5.815
Perceptron	2.241	2.984	3.617	4.109	4.476	4.733	4.950	5.132	5.271	5.398	5.490	5.562
FIMT-DD	2.431	3.054	3.525	3.923	4.205	4.421	4.607	4.768	4.915	5.044	5.157	5.247
Baseline	1.321	2.291	3.015	3.612	4.100	4.493	4.806	5.059	5.289	5.373	5.427	5.490

Tabla 50: resultados de los experimentos sobre los enlaces variables del primer día (percentiles 75 a 80 respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	11	6	6	1	4	3	2	1	1	1	1	1
FTM (0.1)	8	7	6	1	1	3	2	1	1	1	1	1
FTM(0.5)	8	7	6	1	4	3	2	1	1	1	1	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	2	2	2	2

Tabla 51: valores de ventana para los resultados de la Tabla 50.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	2.159	3.008	3.656	4.452	4.948	5.386	5.819	6.126	6.347	6.589	6.779	6.931
FTM (0.1)	2.219	3.040	3.700	4.463	4.953	5.390	5.809	6.109	6.332	6.568	6.765	6.905
FTM(0.5)	2.690	3.390	4.021	4.634	5.077	5.473	5.824	6.085	6.296	6.496	6.656	6.776
Perceptron	3.065	3.689	4.378	4.899	5.264	5.559	5.766	5.839	5.908	5.972	6.007	6.084
FIMT-DD	3.150	3.989	4.358	4.743	5.029	5.166	5.328	5.534	5.524	5.684	5.733	5.819
Baseline	2.159	3.008	3.656	4.452	4.948	5.386	5.819	6.038	6.098	6.155	6.214	6.294

Tabla 52: resultados de los experimentos sobre los enlaces variables del primer día (percentiles 80 a 85 respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1	1	1	1	1	1	1	1	1	1	1	1
FTM (0.1)	1	7	1	1	1	1	1	1	1	1	1	1
FTM(0.5)	1	1	1	1	1	1	1	1	1	1	1	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	2	2	2	2	2

Tabla 53: valores de ventana para los resultados de la Tabla 52.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1.925	3.471	4.761	5.930	6.938	7.756	8.148	8.279	8.292	8.480	8.564	8.643
FTM (0.1)	2.071	3.589	4.862	6.013	7.006	7.773	8.140	8.267	8.377	8.465	8.550	8.626
FTM(0.5)	3.142	4.474	5.592	6.567	7.350	7.864	8.119	8.227	8.323	8.405	8.476	8.528
Perceptron	3.548	5.175	6.428	6.803	6.922	7.136	7.166	7.073	6.964	6.915	6.909	6.896
FIMT-DD	4.045	5.415	6.400	7.002	7.206	7.230	7.277	7.383	7.335	7.387	7.432	7.454
Baseline	2.001	3.567	4.898	6.075	7.042	7.729	7.477	7.312	7.168	7.062	7.071	7.450

Tabla 54: resultados de los experimentos sobre los enlaces variables del primer día (percentiles 85 a 90 respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	12	10	12	12	12	12	11	10	9	9	8	6
FTM (0.1)	12	11	12	12	12	12	11	10	9	10	8	11
FTM(0.5)	12	11	12	10	10	12	12	10	9	10	8	6
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	12	12	12	10	6	4	4

Tabla 55: valores de ventana para los resultados de la Tabla 54.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	2.017	3.448	4.545	5.383	5.963	6.455	6.814	7.069	7.289	7.492	7.673	7.837
FTM (0.1)	2.145	3.535	4.605	5.418	5.988	6.467	6.817	7.069	7.287	7.487	7.667	7.829
FTM(0.5)	3.006	4.140	5.022	5.687	6.177	6.575	6.872	7.102	7.304	7.487	7.649	7.787
Perceptron	3.528	4.560	5.401	5.906	6.250	6.519	6.734	6.867	6.985	7.090	7.196	7.287
FIMT-DD	3.532	4.565	5.376	5.816	6.142	6.380	6.579	6.753	6.907	7.038	7.156	7.259
Baseline	2.019	3.452	4.550	5.387	5.965	6.457	6.814	7.069	7.289	7.190	7.138	7.413

Tabla 56: resultados de los experimentos sobre los enlaces variables del primer día (percentiles 90 a 95 respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	11	6	5	5	3	2	1	1	1	1	1	1
FTM (0.1)	7	6	5	4	3	3	1	1	1	1	1	1
FTM(0.5)	7	7	7	4	3	3	2	1	1	1	1	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	1	2	2	2

Tabla 57: valores de ventana para los resultados de la Tabla 56.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1.735	3.086	3.788	4.297	4.774	5.024	5.172	5.407	6.026	6.630	6.882	7.072
FTM (0.1)	1.860	3.132	3.817	4.325	4.776	5.011	5.155	5.425	6.053	6.624	6.876	7.062
FTM(0.5)	2.548	3.462	4.041	4.480	4.833	5.072	5.308	5.685	6.230	6.648	6.867	7.014
Perceptron	3.718	4.866	5.479	5.920	6.375	6.629	6.785	6.956	7.466	7.726	8.074	8.019
FIMT-DD	4.152	4.668	5.156	5.542	5.798	5.925	5.283	6.705	7.285	7.590	7.821	8.120
Baseline	1.735	3.086	3.788	4.325	4.833	5.066	5.196	5.457	6.052	6.630	6.885	7.072

Tabla 58: resultados de los experimentos sobre los enlaces variables del primer día (percentiles 95 a 100 respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1	1	1	8	5	3	5	4	7	1	2	1
FTM (0.1)	1	1	1	8	5	5	5	4	7	1	2	1
FTM(0.5)	1	1	1	7	5	6	4	4	2	1	2	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	1	1	1	1	1	1	1	1	1

Tabla 59: valores de ventana para los resultados de la Tabla 58.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	4.537	7.914	8.850	9.362	10.034	9.923	9.530	9.693	11.699	13.708	14.228	14.514
FTM (0.1)	4.849	7.942	8.852	9.397	9.934	9.827	9.426	9.757	11.817	13.716	14.207	14.480
FTM(0.5)	6.235	8.136	8.904	9.380	9.694	9.700	9.764	10.711	12.444	13.689	14.116	14.273
Perceptron	7.242	10.578	11.715	12.429	13.345	13.620	13.546	13.693	15.411	16.152	17.294	16.674
FIMT-DD	9.503	10.730	11.429	12.296	12.655	12.521	13.041	13.834	14.864	15.556	16.149	16.854
Baseline	4.537	7.914	8.850	9.404	10.300	10.109	9.618	9.900	11.792	13.708	14.228	14.514

Tabla 60: resultados de los experimentos sobre los enlaces variables del primer día (percentiles 99 a 100 respecto a la desviación típica de LQ). Predicciones a medio plazo: de 1 a 12 instantes vista (de 5 a 60 segundos vista). Se muestra el MAE promedio en tanto por ciento.

Algoritmo / Instantes vista	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
FTM (0)	1	1	1	8	5	3	4	4	2	1	2	1
FTM (0.1)	1	1	1	8	5	3	5	4	2	4	2	1
FTM(0.5)	1	1	1	7	5	6	5	4	2	1	2	1
Perceptron	1	1	1	1	1	1	1	1	1	1	1	1
FIMT-DD	1	1	1	1	1	1	1	1	1	1	1	1
Baseline	1	1	1	2	1	1	1	1	1	1	1	1

Tabla 61: valores de ventana para los resultados de la Tabla 60.