UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

Estimación del tiempo de respuesta de proxies web en redes de comunidad utilizando algoritmos de factorización matricial

Autor:

D. Diego Bores Quijano

Tutor:

Dr. D. Eduardo Gómez Sánchez
Dr. D. Miguel Luis Bote Lorenzo

Valladolid, 11 de julio de 2019

TÍTULO:	**Estimación del tiempo de respuesta de proxies web en redes de comunidad utilizando algoritmos de factorización matricial**
AUTOR:	**D. Diego Bores Quijano**
TUTOR:	**Dr. D. Eduardo Gómez Sánchez** **Dr. D. Miguel Luis Bote Lorenzo**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

Tribunal

PRESIDENTE:	**Juan Ignacio Asensio Pérez**
VOCAL:	**Pablo Casaseca de la Higuera**
SECRETARIO:	**Ignacio de Miguel Jiménez**
FECHA:	**11 de julio de 2019**
CALIFICACIÓN:	

Resumen del TFM

Las redes comunitarias inalámbricas son redes con topología en malla en las que es frecuente que el acceso a la Web se haga a través de un conjunto de *proxies*, siendo los usuarios quienes eligen su *proxy* preferido. Esto puede hacer que la carga se concentre en algunos *proxies*, aumentando su tiempo de respuesta y la carga de sus enlaces de entrada y salida, provocando la degradación de la calidad de servicio percibida por los usuarios finales. Existen varias alternativas para la selección automática del *proxy* por parte del cliente, pero para que sean compatibles con el entorno de las redes comunitarias hay que evitar modificar los *proxies* y los protocolos de comunicación de los clientes con los *proxies*. En este sentido, una posible medida es la monitorización, obteniendo información de calidad de servicio mediante sondeos que conformarían una matriz de visibilidad. Una alternativa para aproximar la visibilidad completa, en la que todos los clientes sondean a todos los *proxies* para conocer su tiempo de respuesta, consiste en la estimación de los valores desconocidos de la matriz de visibilidad a partir de los valores que sí se conocen. Estos valores conocidos los obtiene cada cliente o bien mediante sondeos directos a los *proxies* o bien mediante el intercambio de la información con otros clientes. En este trabajo se explora la posibilidad de estimación de estos valores desconocidos mediante el uso de algoritmos de factorización matricial. Para ello, se obtiene en primer lugar un conjunto de datos de una red de comunidad emulada en un banco de pruebas, y a continuación se evalúan los resultados obtenidos por varios algoritmos de factorización matricial. Los re-

sultados obtenidos muestran una mejora sustancial respecto a los algoritmos de referencia utilizados en la mayoría de los casos, lo que indica que la aproximación a la estimación de los valores desconocidos de la matriz de visibilidad mediante la factorización matricial puede ser una contribución interesante en el problema de la elección del *proxy* de acceso a la Web en redes comunitarias.

Palabras clave

Redes comunitarias, cliente, *proxy*, tiempo de respuesta, Web, filtrado colaborativo, factorización matricial, PlanetLab, estimación, monitorización.

Abstract

Community wireless networks are mesh networks in which web access is typically acomplished through a set of proxies, with users choosing their preferred proxy. This may overload a few proxies, thus increasing their response time and load on both their inbound and outbound links. This would result into quality of service degradation, as perceived by end users. There are several alternatives for the client to automatically choose a proxy, but in order to assure compatibility with the community network's environment it is necessary to avoid modifying both the proxies and the communication protocols used by clients to communicate with them. In this sense, a potential approach is monitorization, which consists of obtaining measures of quality of service through probes that make up a visibility matrix. An alternative to build a full visibility matrix, with all clients probing all proxies, is estimating the unknown values of the visibility matrix from the values that each client knows. Clients obtain the known values either through direct probes to the proxies or through the exchange of information with other clients. In this work, the possibility of determining the unknown values through the use of matrix factorization algorithms is explored. In this sense, a dataset is first obtained from a community network emulated in a testbench, and then the results of several matrix factorization algorithms are evaluated. The results obtained show a significant improvement over the baseline algorithms used in most cases, showing that the estimation of the unknown values of the visibility matrix through matrix factorization can be an interesting contribution in the problem of choosing a web access proxy in community networks.

Keywords

Community networks, client, proxy, response time, Web, colaborative filtering, matrix factorization, PlanetLab, estimation, monitorization.

Agradecimientos

Para empezar, quisiera agradecer su apoyo a todos aquellos que han estado ahí cuando lo he necesitado. En lo académico, me gustaría dar las gracias a Miguel y a Eduardo, tutores de este trabajo fin de máster, por su dedicación y su ayuda, y al Dr. Roc Meseguer por su disponibilidad para proponer alternativas y resolver dudas. También me gustaría dar las gracias a mis compañeros de clase, por haberme acompañado en este año tan intenso y estresante.

Me gustaría agradecer también a mi familia más cercana, haciendo especial incapié en mi hermana, que siempre sabe hacerme ver las cosas de otra manera cuando todo parece muy negro. También a mis amigos por haberme ayudado a escapar del estrés en momentos críticos.

Muchísimas gracias a todos.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las redes comunitarias [Bra13] son redes distribuidas, descentralizadas y a gran escala. En estas redes los usuarios comparten recursos como la infraestructura o el acceso a Internet [Par14, Uni09], lo que permite a comunidades locales crear su propia infraestructura de red [Veg15]. Permite, además, abaratar los costes y hacer que la conexión a Internet sea asequible incluso en áreas rurales [RM13]. En este tipo de redes, con topología en malla, un aspecto fundamental es la cooperación entre los miembros, controlada por reglas de participación como acuerdos de comunicación entre pares o licencias de pertenencia [Neu16].

Existen varias redes comunitarias inalámbricas en Europa, como la red comunitaria Funk-Feuer [1] en Austria, la red Ninux desplegada en Italia [Mac13], o la red Guifi.net [2], desplegada en España. En estas redes, el acceso a la Web se hace típicamente a través de *proxies*. Para iniciar una conexión a través de un *proxy* [Dim17], los usuarios eligen su *proxy* preferido de entre todos los disponibles de la red.

En las redes comunitarias, un aspecto clave que hay que tener en cuenta es el hecho de que los enlaces los aportan y organizan los participantes. Esto implica que los caminos entre nodos, o entre un cliente y un *proxy*, pueden no ser fiables [Aky05]. Existen, por lo tanto, dos retos principales en las redes de comunidad inalámbricas. El primero de ellos surge por la volatilidad de los enlaces, y en este sentido la predicción de la calidad del enlace es un aspecto importante en este tipo de redes [Bot18, Mil15]. El segundo reto aparece debido a que son los usuarios los que eligen el *proxy* que utilizan para acceder a la Web. Si todos los usuarios escogen el mismo conjunto de *proxies*, ya sea porque son los más cercanos o porque son los primeros de la lista, existe la posibilidad de que aparezca congestión en estos nodos de salida, dando lugar a un rendimiento degradado en el acceso a la Web [Dim17].

Por lo tanto, en las redes de comunidad en las que son los usuarios quienes eligen el *proxy* a través del que se conectan a la Web, puede ocurrir que mientras unos *proxies* permanecen subocupados, otros estén sobrecargados y ofrezcan una tiempo de respuesta superior, entendiendo éste como el tiempo transcurrido desde que el cliente realiza una petición hasta que recibe la respuesta completa. Además, si aumenta el número de clientes que realizan peticiones a un mismo *proxy*, aumentará la sobrecarga de sus enlaces de

[1] https://www.funkfeuer.at/

[2] http://guifi.net

entrada, lo que afectaría también al tiempo de respuesta que percibe el cliente. Dada la importancia del rendimiento de los *proxies* en el acceso a la Web, el de la elección de *proxy* es un problema abierto que puede afectar en gran medida a la calidad de servicio que percibe el usuario final.

Mientras que la estimación de la calidad de los enlaces se aborda en [Bot18], este trabajo se centra en el segundo problema, el de la elección automática del *proxy*. Para ello, se hace uso de métricas tomadas por la parte cliente que permiten elegir los mejores *proxies* de entre los disponibles o, al menos, evitar aquellos con un rendimiento bajo. Pero la solución a este problema debe ser incremental y compatible hacia atrás, de forma que funcione tanto con los clientes ya existentes como con los mejorados para la elección automática. Además, debería ser independiente de los algoritmos de enrutado y transporte, de forma que no implique cambios que sean incompatibles con el entorno de red existente [Dim17]. Esto implica que las métricas en función de las que se escoge el *proxy* se deben obtener a partir de respuestas del *proxy* a sondeos del cliente, que sean como las obtenidas en la interacción normal entre ambos, para evitar modificar el protocolo de comunicación con el *proxy*. Es decir, se deben obtener las métricas de la información disponible en la respuesta del *proxy* a una petición HTTP.

Por otro lado, se puede mejorar el conocimiento sobre el funcionamiento de los *proxies* y, por tanto, la probabilidad de que el cliente elija uno adecuado será mayor si se aumenta el número de *proxies* de los que se conocen métricas indicativas de la calidad de servicio. Hay que tener en cuenta que los sondeos implican un aumento del tráfico en la red, y el hecho de que cada cliente sondease a un número elevado de *proxies* supondría un incremento tanto en la carga de los clientes y de los *proxies* como del tráfico en la red [Bat18].

Este problema podría solucionarse si cada cliente realizase sondeos a un subconjunto reducido de *proxies*, elegidos de forma aleatoria de entre el conjunto de todos los *proxies* de la red, y compartiese esta medida con el resto de clientes. De esta forma, las medidas propias junto con las recibidas de otros clientes servirían para que el cliente estimase la medida correspondiente a aquellos *proxies* que no sondea directamente, pero de los que recibe medidas tomadas por otros clientes. Así, el cliente tendría información suficiente, formada por medidas reales y estimadas, para elegir de entre todos los *proxies* aquellos con mejor rendimiento o, al menos, para evitar aquellos cuyo comportamiento fuese peor. Pero para que el uso de estos algoritmos sea efectivo, las estimaciones deben ser precisas.

Una posible aproximación para obtener estas estimaciones es el uso de algoritmos de factorización matricial. Estos algoritmos han sido utilizados con buenos resultados para la estimación de métricas de calidad de servicio en trabajos de la literatura como [Zhu17], [Zhe13] o [Lo12]. En estos trabajos se proponen diferentes algoritmos de factorización matricial para la estimación de parámetros de calidad de servicio para elegir servicios Web. Pese a tratarse de servicios diferentes a los ofrecidos por los *proxies* en las redes comunitarias, el objetivo en los tres casos es estimar los valores de calidad de servicio desconocidos en un entorno variante. Ésto podría encajar bien en el caso de la estimación de métricas de calidad de servicio en las redes comunitarias. Mientras que en [Zhu17] se propone un algoritmo de factorización matricial adaptativa, en [Zhe13] y en [Lo12] se describen dos métodos de factorización matricial que emplean información de similitud entre clientes. La principal ventaja de este tipo de algoritmos es su gran precisión [Kor09],

así como su carga computacional reducida respecto a otras técnicas de estimación como el aprendizaje profundo (*deep learning*).

1.1. Objetivos

El objetivo de este trabajo es **abordar el problema de la selección de *proxy* para el acceso a la Web en redes comunitarias inalámbricas mediante la estimación de métricas indicativas de calidad de servicio haciendo uso de algoritmos de factorización matricial**. Para ello, se emula una red comunitaria en la plataforma PlanetLab, que hace uso de nodos reales distribuidos por Europa y Oriente Próximo, en lugar de recoger datos de una red comunitaria real. A partir de los datos obtenidos se realizan una serie de experimentos de estimación haciendo uso de varios algoritmos de factorización matricial.

De esta forma, y mediante el intercambio de las medidas obtenidas por cada cliente al subconjunto de *proxies* que sondea, los clientes podrían tener una información sobre el comportamiento de los *proxies*, que será tan válida como precisas sean las estimaciones, y que les permitiría elegir un *proxy* con un buen rendimiento. Se pretende con ello reducir el número de sondeos necesarios de cada cliente a cada *proxy* respecto a un escenario en el que todos los clientes sondean a todos los *proxies*, reduciendo también el tráfico de la red y la carga tanto en los nodos clientes como en los *proxies*. Hay que tener en cuenta que el intercambio de medidas entre clientes vecinos supondrá cierto tráfico, aunque se estima que será menor que el que supondrían los sondeos a los *proxies*.

El principal motivo por el que se utiliza la emulación es la gran dificultad para disponer de un número considerable de clientes diferentes y situados en distintas máquinas. En concreto, la red emulada está formada por 9 clientes que realizan sondeos periódicos a 5 *proxies* y, si bien el número de *proxies* y clientes no es elevado, es suficiente para realizar una primera aproximación y verificar la validez de los algoritmos de factorización matricial para el problema que abordamos. Como línea de trabajo futura, habría que determinar en qué medida los resultados obtenidos a partir de la emulación son extrapolables a un contexto de red comunitaria real. En este caso se podría aumentar la carga de los *proxies* de forma artificial durante la captura.

1.2. Metodología

El trabajo realizado durante este proyecto se ajusta al método de la ingeniería. Este método es un paradigma evolutivo en el que, de forma iterativa, se obtienen sistemas eficientes que resuelven diferentes problemas. Por otro lado, el método de ingeniería para la resolución de problemas se divide en las etapas de observación de las soluciones ya existentes, de propuesta de soluciones que mejoran las anteriores, de desarrollo de la propuesta y de medición y análisis de la solución obtenida. Este proceso se repite hasta que la mejora deje de ser apreciable [Adr93].

El punto de partida es el trabajo realizado en [Bor18], en el que se propusieron dos aproximaciones, la estimación mediante factorización matricial y la predicción de series temporales, y se vieron algunas de las deficiencias encontradas. Estas deficiencias eran: un conjunto de datos que presentaba un número excesivamente pequeño de clientes (muy

parecidos entre ellos) y *proxies*, y un algoritmo de base que no era adecuado para compararlo con la factorización matricial.

Por cada una de las limitaciones planteadas, se propone una solución, que serán descritas a lo largo de este documento. Para la elaboración de estas propuestas, se realizó un proceso iterativo de evaluación del avance y de los problemas surgidos con los tutores, determinando al finalizar cada reunión la línea de trabajo futuro a seguir.

Una vez completadas las diferentes mejoras, se ha realizado un análisis de los resultados obtenidos para evaluar, a partir de un entorno emulado de red de comunidad, si se consigue una mejora respecto al caso inicial. Al final de este trabajo se recogen los diferentes problemas que se detectaron durante las iteraciones, algunas de las cuales serán tratadas en trabajos futuros.

1.3. Estructura del documento

En el capítulo 2 se exponen de forma detallada las diferentes alternativas para la elección del *proxy* de forma automática por parte del cliente en redes comunitarias. También se explica qué es el filtrado colaborativo, y qué es la factorización matricial, incluyendo la formulación genérica del problema y la descripción de los algoritmos que se utilizan en los experimentos de estimación. Por último, se discute la adecuación del uso de estos algoritmos, y se discuten algunos ejemplos de la literatura, sobre su uso en problemas similares al tratado en este trabajo.

En el capítulo 3 se describen los datos utilizados en los experimentos de estimación, así como la forma en que se obtuvieron. Se describe también la implementación de los algoritmos empleada, así como la de los propios experimentos realizados. Se explica la métrica de rendimiento utilizada, justificando su uso frente a otras métricas conocidas. Por último, se exponen los algoritmos de base utilizados, cuyos resultados se compararán con los obtenidos por los algoritmos probados.

En el capítulo 4 se presentan y discuten los resultados obtenidos, justificando en cierta medida su relación con los datos utilizados. Por último, en el capítulo 5 se exponen las conclusiones extraídas del trabajo realizado y se plantean posibles vías de trabajo futuras. En los apéndices se muestran las tablas de resultados no incluidas en el capítulo 4.

Capítulo 2

Estado del arte

2.1. Introducción

Como se indicó en el capítulo anterior, la elección del *proxy* influye en gran medida en la calidad de servicio que percibe el usuario final. De acuerdo con [Bat18], no es tan importante elegir el mejor *proxy* como evitar aquellos *proxies* que presentan un rendimiento degradado de cara a obtener una calidad de servicio aceptable.

En este sentido, la monitorización es un aspecto clave a la hora de asegurar el rendimiento de un sistema distribuido complejo, como es el caso de las redes comunitarias. Gracias al uso de monitorización es posible tanto controlar la calidad de servicio, como realizar un buen reparto de los recursos disponibles y detectar anomalías. En las redes comunitarias, para escoger el *proxy* de acceso a la Web, los clientes pueden utilizar como referencia alguna de las métricas de calidad de servicio disponibles en este tipo de redes.

En este capítulo se presentan las diferentes alternativas que se pueden encontrar en la literatura para la monitorización de redes en el contexto de la elección del *proxy* en redes comunitarias. También se describen el filtrado colaborativo, y dentro de éste la factorización matricial. Ambas son alternativas interesantes para la estimación de métricas de rendimiento de los *proxies* que pueden ayudar en la elección automática del *proxy* de acceso a la Web en redes comunitarias.

2.2. El problema de la selección de *proxy* en redes comunitarias

En las redes comunitarias, una primera forma de monitorización podría ser la visibilidad completa entre clientes y *proxies*, de forma que todos los clientes obtengan métricas indicativas de la calidad de servicio de todos los *proxies*. Si tenemos P *proxies* y C clientes, la visibilidad completa se puede ver como una matriz R, de dimensiones $C \times P$, en la que cada elemento r_{ij} contiene la información de calidad de servicio percibida por el cliente i del *proxy* j. En la figura 2.1 se puede ver de forma gráfica esta matriz. Un ejemplo de visibilidad completa es la existente en la infraestructura de red utilizada por Google para la distribución de contenidos [Kri09]. El principal problema de esta aproximación es que habría que implicar tanto a los *proxies*, como a la infraestructura y a los

	C_1	C_2	C_3	...	C_i	...	C_c
P_1	r_{11}	r_{21}	r_{31}	...	r_{i1}	...	r_{c1}
P_2	r_{12}	r_{22}	r_{32}	...	r_{i2}	...	r_{c2}
P_3	r_{13}	r_{23}	r_{33}	...	r_{i3}	...	r_{c3}
...	...	...	...	...	...	...	...
P_j	r_{1j}	r_{2j}	r_{3j}	...	r_{ij}	...	r_{cj}
...	...	...	...	...	...	...	...
P_p	r_{1p}	r_{2p}	r_{3p}	...	r_{ip}	...	r_{cp}

Figura 2.1: Figura explicativa de la matriz de visibilidad. Cada elemento r_{ij} contiene la información de calidad de servicio percibida por el cliente *i* del *proxy j*.

clientes [Kau14] en la toma de medidas para que ésta fuese sencilla y, como hemos visto, en una red comunitaria esto no es posible.

Este problema se puede solucionar si los clientes toman medidas de los *proxies*, sin que éstos tengan que participar de forma activa. Una forma de conseguirlo es mediante fuerza bruta, en la que cada cliente sondea, de forma independiente del resto, a todos los *proxies* del sistema [Sal18]. Esta solución puede ser útil si el número de clientes o *proxies* existentes en la red es pequeño, pero no es escalable a las redes comunitarias, que cuentan con un gran número de clientes, ya que el número de sondeos que recibiría cada *proxy*, y que viajaría a través de la red, sería muy elevado. Para conseguir esta escalabilidad se puede reducir el número de *proxies* que cada cliente sondea a un subconjunto del total, formado o bien por los más cercanos [Bat18] o bien por algunos escogidos de forma aleatoria del conjunto total [Lum09]. Pero esto implica que la visibilidad deja de ser completa.

Para conseguir una aproximación a la visibilidad completa, se puede hacer uso de mecanismos de intecambio de las medidas tomadas por los clientes. De esta forma, cada cliente mantendría la visibilidad completa de un subconjunto de los *proxies* que compartiría con el resto de los clientes. Una opción para realizar esta distribución de la información es el uso de coordenadas virtuales, como en el caso de Vivaldi [Dab04], que permite la estimación de latencias entre los nodos que colaboran.

Para el uso de estos sistemas en el contexto de la elección de *proxy*, en [Led08] se propone un sistema, basado en Vivaldi, que permite estimar latencias entre cliente y *proxy*. Pero en esta aproximación las medidas que se obtienen de una red Vivaldi no son muy precisas, y son costosas de conseguir [Lum09, Kau14], puesto que requieren un gran número de sondeos periódicos, que si bien son menores que en el caso de fuerza bruta, no dejan de ser considerables. En [Dim17] se propone un sistema basado en Vivaldi que

permite a los clientes compartir de forma periódica tanto la latencia con otros nodos, como la latencia a los *proxies*.

Una forma de cooperación entre clientes alternativa a las basadas en coordenadas virtuales es la basada en reutilizar los paquetes de sondeo. Así, en [Ko13], los clientes que se encuentran en el camino entre el cliente que realiza el sondeo y el *proxy* sondeado, capturan los paquetes empleados en el sondeo para calcular su propia información, evitando así realizar un nuevo sondeo. Esto implica que solo se pueden reutilizar aquellos sondeos que pasan a través del cliente que los reutiliza.

Otra estrategia de cooperación es la que se basa en el sondeo aleatorio de algunos *proxies*. En [Bat19] se propone un sistema en el que los clientes sondean al último *proxy* utilizado y a dos más, escogidos de forma aleatoria. A continuación, los clientes comparten los resultados del sondeo con los clientes topológicamente más cercanos. Sin embargo, esta aproximación no garantiza que todos los clientes tengan información de todos los *proxies*, esto es, habrá elementos de la matriz R de valor desconocido. En esos casos, se parte de la suposición de que las medidas tomadas al mismo *proxy* por clientes cercanos será similar y se aproximan los valores desconocidos por los medidos por los clientes más cercanos.

Otras técnicas que pueden permitir obtener los elementos desconocidos de la matriz de medidas R pueden ser las basadas en estimación. En este sentido, en [Nie16] se utilizan técnicas de aprendizaje profundo (*deep learning*) para obtener estos valores. Sin embargo, el coste computacional de estos algoritmos es elevado, de forma que su ejecución debería limitarse a equipos dedicados, apareciendo así el problema de la recolección de los datos empleados para la estimación, y la transferencia de los resultados.

En el contexto de la estimación de métricas de calidad de servicio (QoS, *Quality of Service*), otra opción es la del uso del filtrado colaborativo, que implica un coste computacional menor que el del aprendizaje profundo, permitiendo que pueda ser ejecutado en los propios clientes. Además, el filtrado colaborativo únicamente necesita la matriz de medidas, no es necesaria otra información contextual como la localización geográfica de los usuarios [Ric15]. La principal desventaja de los algoritmos de filtrado colaborativo es el arranque frío (*cold start*) [Su09], que en nuestro problema se traduciría en que sería necesario un número relevante de mediciones de un cliente o de un *proxy* antes de poder realizar estimaciones relativas a ellos. En artículos de la literatura como [Zhe17], [Luo16] o [Vad16] se emplean algoritmos de filtrado colaborativo en el contexto de la recomendación de servicios mediante la estimación de métricas QoS. Dentro de estas técnicas de filtrado colaborativo se encuentran los algoritmos de factorización matricial, que ofrecen una gran precisión en matrices con un bajo porcentaje de valores conocidos, y que han sido utilizados con éxito en algunos casos como [Lo12] o [Zhe13] para la estimación de los valores restantes de la matriz de medidas R.

2.3. Introducción al filtrado colaborativo y a la factorización matricial

Como se indica en la sección 2.2, el filtrado colaborativo es una alternativa interesante para la estimación de los valores desconocidos de la matriz de medidas. Los algoritmos

de filtrado colaborativo son un subconjunto de los conocidos como sistemas de recomendación. Estos sistemas son utilizados típicamente por los grandes proveedores web de contenido como películas, series, música o libros, y por los grandes comercios en línea que ofrecen infinidad de productos, para facilitar la elección del producto por parte del usuario de acuerdo con sus gustos, que son inferidos por estos sistemas para maximizar el grado de satisfacción del usuario [Res97, Kor09, Lop11].

Un sistema de recomendación se puede describir como un sistema de procesado de información que realiza recomendaciones a partir de varios tipos de datos [Ric15]. Estos sistemas tratan de obtener un perfil que describa las preferencias de los usuarios sobre los artículos, modelando también la relación entre usuarios y artículos [Tak09], para realizar recomendaciones de nuevos artículos a los usuarios. Los artículos son aquellos objetos que el sistema recomienda, y pueden ser caracterizados por su valor, su utilidad o su complejidad. Por su parte, los usuarios tienen asociadas una serie de características que el sistema puede utilizar para realizar las recomendaciones. Por último, las valoraciones indican la relación entre los usuarios y los artículos, y a su vez son la forma de interacción entre el usuario y el sistema de recomendación.

Ejemplos de estos sistemas [Agg16] son el utilizado por la plataforma de vídeo bajo demanda Netflix o el sistema de recomendación utilizado por la web de ventas *online* Amazon.com. En ellos, los usuarios puntúan artículos, películas y series en el caso de Netflix, o artículos a la venta en el caso de Amazon. A partir de estas puntuaciones y de la información de los patrones de búsqueda y compra, realizan recomendaciones de nuevos productos al usuario.

En el contexto de la selección del *proxy* de acceso Web en redes comunitarias, se podrían identificar los artículos con los *proxies* y los usuarios con los clientes. Así, la puntuación que relaciona un artículo con un usuario se podría identificar con la métrica indicativa sobre el rendimiento del *proxy*, que será utilizada para elegir uno de los *proxies* disponibles. El objetivo de esta sección es, en primer lugar, explicar qué son los algoritmos de filtrado colaborativo y qué tipos hay y, en segundo lugar, explicar más en profundidad qué es la factorización matricial y cuáles son los algoritmos que se han empleado en los experimentos de estimación realizados.

2.3.1. El filtrado colaborativo

Los métodos de filtrado colaborativo se basan en cómo los usuarios realizan las puntuaciones para recomendarles artículos, sin hacer uso de información de otro tipo como el lugar donde vive un usuario o la categoría a la que pertenece un artículo. Una aproximación sencilla a esta idea es la de recomendar a un usuario aquellos artículos que gustaron a otros usuarios con gustos parecidos en el pasado. Para calcular la similitud entre los gustos de diferentes usuarios se hace uso su historial de puntuaciones. En la figura 2.2 se puede ver de forma esquemática cómo funcionan estos algoritmos. En función de la similitud de las puntuaciones de diferentes usuarios a diferentes artículos, el sistema recomienda a cada usuario artículos que aún no han sido puntuados por ellos.

Los algoritmos de filtrado colaborativo se pueden clasificar en basados en memoria y basados en modelo [Su09]. Los primeros calculan el parecido entre los usuarios a partir de las puntuaciones y realizan predicciones o recomendaciones a partir de esta información. Algunos ejemplos son los basados en usuario, que predicen los intereses de un usuario

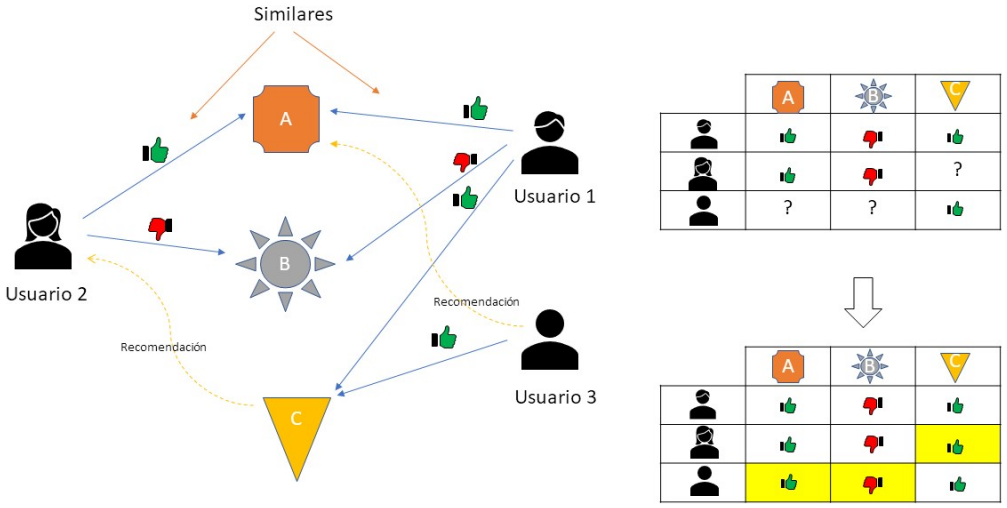


Figura 2.2: Figura explicativa del filtrado colaborativo. Los dos primeros usuarios puntúan de forma similar a los artículos A y B, y el usuario 3 puntúa de forma similar al usuario 1 el artículo C. Una vez establecida la similitud entre usuarios y artículos, el sistema estima la puntuación que darían los usuarios 2 y 3 a los artículos que no puntúan, y recomiendan aquellos con una puntuación favorable.

por un artículo en función de información de valoraciones de usuarios con características similares, o los basados en artículo, que emplean también la similitud pero en este caso entre artículos.

Por su parte, los basados en modelo realizan estimaciones sobre los gustos de los usuarios por artículos que aún no han valorado, haciendo uso de un modelo que se genera a partir de las valoraciones. Se basan en el concepto del *factor latente*, que trata de caracterizar a los usuarios y los artículos con factores inferidos a partir de la forma en que puntúan los usuarios [Kor09, Han11]. Se puede interpretar el concepto de *factor latente* como algo que explica la relación entre las valoraciones de distintos usuarios sobre distintos artículos. Destacan los algoritmos basados en agrupamiento o *clustering*, los basados en reglas de asociación, los basados en redes neuronales recurrentes, los basados en máquinas de Boltzman restringidas y los basados en factorización matricial. En este trabajo se han utilizado algoritmos basados en factorización matricial, por el moderado coste computacional de algunas de sus variantes y su uso exitoso en problemas semejantes de estimación de métricas QoS, como se detalla en secciones posteriores. No obstante, este trabajo ha dejado fuera de su alcance considerar la conveniencia de otros tipos de filtrado colaborativo.

2.3.2. La factorización matricial

Como se ha visto en la sección anterior, la factorización matricial es un tipo de filtrado colaborativo. Se basa en la inferencia de una serie de *vectores latentes* a partir de la factorización de la matriz de puntuaciones. Así, si se considera un número de usuarios m y un número de artículos n, a partir de la matriz R de puntuaciones, de tamaño $m \times n$,

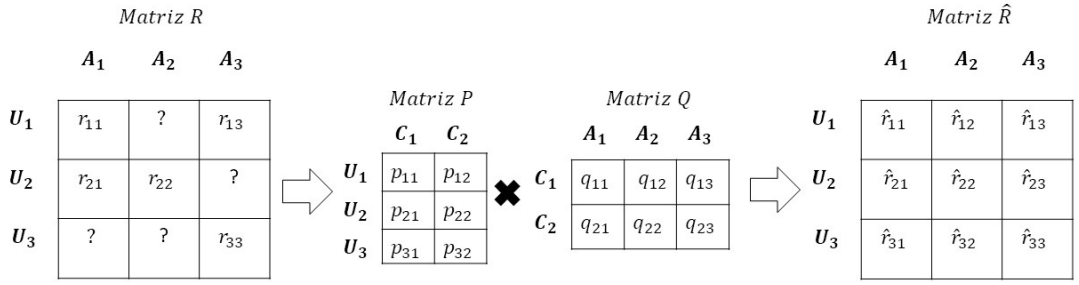


Figura 2.3: Figura explicativa de la factorización matricial. A partir de la matriz *R* con elementos desconocidos, se obtienen las matrices *P* y *Q*, suponiendo un número de conceptos igual a 2. Multiplicando estas matrices se obtiene la matriz estimada $\hat{R}$.

se obtienen las matrices *P* y *Q*, de tamaños $m \times k$ y $k \times n$ respectivamente, mediante factorización. De esta forma:

$$R \approx PQ$$

Se conocen como *vectores latentes* las columnas de la matriz *P* y las filas de la matriz *Q*. Por su parte, las filas de *P* y las columnas de *Q* son los *factores latentes*. Estos *factores latentes* se pueden dividir en *factores de usuario* y *factores de artículo*. Los primeros son las filas de *P*, donde cada elemento contiene la preferencia de cada usuario a los *k* conceptos en los que se puede dividir la matriz de puntuaciones *R*. Por otro lado, las columnas de *Q* son los *factores de artículo*, y representan la afinidad de cada artículo por los *k* conceptos de la matriz *R*. En la figura 2.3 se puede observar de forma gráfica este proceso.

La mejor forma de entender qué es un concepto es a través de un ejemplo: imaginemos una matriz con valoraciones de usuarios a varios grupos musicales. Estos conceptos serían, por ejemplo, los géneros musicales en los que se puede incluir cada grupo. De este modo, al hablar de la preferencia de un usuario por un concepto, estamos hablando de en qué medida le gusta a dicho usuario un determinado género musical. Por otro lado, cuando se habla de la afinidad de un artículo por un concepto, estamos hablando del grado en que un grupo musical se puede englobar dentro de un género. Aunque estos conceptos no suelen ser tan intuitivos como en el ejemplo, siempre representan un patrón de correlación dominante en la matriz R [Agg16]. Así, se puede obtener una estimación de cada valor r_{ui} de la matriz R [Agg16] mediante:

$$r_{ui} \approx \sum_{s=1}^{k} p_{us} \cdot q_{si}$$

donde p_{us} es cada elemento de la matriz P, y q_{si} es cada elemento de la matriz Q. Esta forma de obtener las puntuaciones se puede ver como una suma de los productos de las afinidades, por un lado del usuario i, y por el otro del artículo j, a cada uno de los conceptos de la matriz R. Algunos algoritmos de factorización matricial son:

- Factorización matricial probabilística (PMF, *Probabilistic Matrix Factorization*): es un algoritmo de factorización matricial explicado en [Sal07]. El procedimiento es el siguiente: en primer lugar se define la función de distribución de probabilidad de la matriz R condicionada a las matrices P y Q, y se emplea un modelo lineal con ruido Gaussiano para modelarla. A continuación, se obtiene la función de distribución a posteriori, y se trata de minimizar su logaritmo neperiano o, equivalentemente, la función objetivo, que sería la suma de los errores cuadráticos.

- Factorización tensorial no negativa (NTF, *Non-negative Tensor Factorization*): algoritmo propuesto en [Zha14]. En este caso, en lugar de factorizar la matriz de puntuaciones en un instante temporal, se factoriza el conjunto de todas las matrices de puntuaciones en todos los instantes de medida, que forman una matriz tridimensional donde la tercera dimensión es el tiempo. Este tipo de matrices son conocidas como tensores. Se trata de un algoritmo que, si bien tiene en cuenta la evolución temporal de las valoraciones, requiere recalcular el modelo completo con cada nueva matriz de puntuaciones. Por lo tanto, es un algoritmo útil cuando la frecuencia de actualización de los datos es baja.

- Factorización matricial adaptativa (AMF, *Adaptative Matrix Factorization*): se trata del algoritmo propuesto en [Zhu17], en el que se realiza una factorización matricial en línea, esto es, se actualiza el modelo con los últimos datos, no requiere volver a obtener el modelo completo a partir del conjunto de datos actualizado. Los modelos utilizados son adaptativos, lo que quiere decir que se tiene en cuenta la evolución temporal de los valores de la matriz *R* a la hora de realizar las estimaciones. Dado que el modelo se actualiza de forma incremental con cada nueva actualización de los datos, implica una menor carga computacional y un menor almacenamiento en memoria que el algoritmo NTF.

2.4. Estimación de métricas QoS mediante filtrado colaborativo y factorización matricial

En la literatura se pueden encontrar ejemplos del uso del filtrado colaborativo para la estimación de métricas en problemas similares al que se aborda en este trabajo. Así, en [Zhe17] se utiliza una aproximación al filtrado colaborativo basado en usuario para obtener recomendaciones de servicios en la nube. En [Luo16] se emplea un mecanismo de filtrado colaborativo basado en el coeficiente de correlación de Pearson con el fin de obtener los valores desconocidos en una matriz de QoS. Por su parte, en [Vad16] se introduce un sistema de recomendación basado en filtrado colaborativo que hace uso de las preferencias de los usuarios sobre características no funcionales de los servicios.

En cuanto al uso específico de algoritmos de factorización matricial en el contexto de la estimación de métricas de calidad de servicio, en [Zhu17] se describe una aproxima-

ción a la estimación en línea de QoS. Ésta se basa en factorización matricial adaptativa, y gracias a ello se tiene en cuenta al realizar la estimación la evolución de las medidas con el tiempo. La evaluación del funcionamiento del algoritmo propuesto se hace comparándolo con otros algoritmos de factorización matricial como PMF o NTF. Los resultados indican que el algoritmo propuesto obtiene mejores resultados que el resto. Los datos sobre los que se realizan los experimentos se obtienen de clientes desplegados en nodos de PlanetLab[1] que realizan pruebas a servicios accesibles públicamente de forma periódica cada 15 minutos.

Por su parte, en [Zhe13] se propone un método de factorización matricial que emplea información de similitud entre clientes, obtenida como el coeficiente de correlación de Pearson entre los vectores de medidas de QoS dos clientes, y lo combina con la factorización matricial para estimar los valores desconocidos de la matriz de medidas. De nuevo, se obtienen los datos mediante clientes desplegados en nodos de PlanetLab, que acceden a servicios Web públicos. En este caso se obtiene una única matriz en un instante, no se repiten las medidas de forma periódica. El algoritmo presentado, por lo tanto, no tiene en cuenta la evolución temporal de las medidas que cada cliente obtiene de cada servicio. Otro ejemplo es [Lo12] donde, sobre los mismos datos que [Zhe13], de nuevo se emplea la similitud entre las medidas de dos usuarios en forma de coeficiente de correlación de Pearson. Se realiza un filtrado en el que se eliminan aquellos usuarios que no son similares formando vecindarios, y todo esto se combina con el modelo clásico de factorización matricial. De forma adicional, se añade un término de regularización basada en usuario, que trata de minimizar las diferencias entre los factores latentes de los usuarios pertenecientes al mismo vecindario. Los datos utilizados son los mismos que en [Lo12].

En [Bor18] se realiza la estimación de el TTFB (tiempo hasta el primer *byte*, *Time to First Byte*) de *proxies*usados en el acceso a la Web en redes de comunidad.Para ello, se hace uso del algoritmo de factorización matricial BRISMF (*Biased Regularized Incremental Simultaneous Marix Factorization*, Factorización de Matrices Simultánea Incremental e Insesgada) [Tak09], con una precisión elevada . Se utilizan datos procedentes de una red comunitaria real, Guifi.net [2], pero el número de clientes utilizados es muy bajo, y las medidas que toma cada uno son muy similares a las que toma el resto, por lo que el conjunto de datos no permite extrapolar los resultados al contexto de la red comunitaria, donde el número de clientes diferentes y la variabilidad de las medidas serán mucho mayores.

Los resultados obtenidos en [Bor18] son buenos en términos de error absoluto medio, si bien la metodología presenta algunas deficiencias. En primer lugar, se trata de estimar la matriz completa, y a continuación se utiliza esta matriz para actualizar el modelo. Esto no permite evaluar en qué medida se puede utilizar el filtrado colaborativo para reducir el número de sondeos necesarios, puesto que se supone que todos los clientes sondean a todos los *proxies*. Además, se comparan los resultados obtenidos por el algoritmo BRISMF con un algoritmo de base que consiste en estimar en un instante cada muestra de la matriz de medidas como las muestras correspondientes a cada pareja cliente-*proxy* en el instante anterior . Este algoritmo de base no es el más adecuado, puesto que también requiere conocer la matriz de visibilidad al completo. Los resultados obtenidos indican que el algoritmo presenta mejores resultados en los casos en los que los datos presentaban una

[1] https://www.planet-lab.eu,Última visita: 3 de julio de 2019

[2] https://www.guifi.net, Última visita: 3 de julio de 2019

variabilidad mayor, lo que a priori podría indicar una mayor dificultad en la estimación.

2.5. Conclusiones

En este capítulo se han descrito las diferentes alternativas de monitorización para la elección automática por parte de los clientes de su *proxy* de acceso a la Web en el contexto de las redes comunitarias. Una alternativa interesante es la de aproximar la matriz de visibilidad completa mediante la medición de un subconjunto de los elementos de la matriz mediante sondeos de los clientes, y la estimación de los elementos restantes. Para mantener esta información actualizada es necesario repetir los sondeos y realizar las estimaciones de forma periódica, por lo que es importante que los algoritmos de estimación no impliquen una gran carga computacional.

Los algoritmos de filtrado colaborativo destacan en este contexto por su adecuación para la estimación de los valores desconocidos de la matriz de visibilidad. La carga computacional que implica el uso de estos algoritmos es mucho menor que la que requerida por otros, por lo que son adecuados para su uso en el cliente, de modo que no implique una sobrecarga en el mismo, y para la obtención de la matriz de visibilidad estimada de forma periódica, manteniendo así la información sobre el rendimiento de los *proxies* actualizada. Dentro de esta familia de algoritmos, destacan los de factorización matricial por su precisión. Por último, se han visto ejemplos que se pueden encontrar en la literatura del uso de estos algoritmos en contextos similares al tratado en este trabajo.

Capítulo 3

Entorno experimental

3.1. Introducción

Como se indica en el capítulo 2, los algoritmos de factorización matricial son una buena opción para la estimación de medidas de calidad de servicio en redes comunitarias. En concreto, este trabajo se centra en la estimación de métricas de calidad de servicio referidas al acceso a la Web a través de *proxies*. En este sentido, un primer paso es escoger la métrica a utilizar de entre todas las disponibles, atendiendo a qué beneficios puede tener cada una y si se quiere optimizar algún parámetro como la distribución de la carga entre los *proxies* o la calidad de servicio percibida por el usuario.

Para la realización de los experimentos de estimación, es importante elegir qué algoritmos utilizar. En este sentido, criterios a tener en cuenta para agilizar las pruebas son que exista una implementación del algoritmo escogido o su facilidad de uso. Además, es necesario escoger o diseñar algoritmos de referencia sencillos, para comparar sus resultados con los obtenidos por los algoritmos a evaluar y ver en qué medida la mayor complejidad de estos nuevos algoritmos se ve compensada en una mejora de los resultados respecto a los algoritmos de referencia. También es importante definir una métrica de error de las estimaciones, que será utilizada para evaluar y comparar los resultados de los diferentes algoritmos.

En este capítulo se describen, en primer lugar, los datos utilizados en los experimentos de estimación, incluyendo una pequeña discusión sobre cuál de las métricas disponibles es la más adecuada para la elección del *proxy*, la descripción de la red en que se obtuvieron los datos, y un análisis de las muestras tomadas. A continuación, se explican los algoritmos de estimación utilizados, tanto de factorización matricial como de referencia. En la sección 3.5 se indican los experimentos de estimación realizados, y en la sección 3.6 se explican algunas de las principales métricas de rendimiento utilizadas, exponiendo cuales son las ventajas más importantes de la métrica escogida.

3.2. Conjunto de datos

Para poder evaluar el rendimiento de los algoritmos de factorización matricial en el contexto de la estimación de métricas de calidad de servicio en las redes comunitarias, es preciso en primer lugar disponer de un conjunto de datos sobre los que realizar las

estimaciones y evaluar el error cometido por los algoritmos. En esta sección se describen y justifican los datos utilizados en los experimentos de estimación realizados en este trabajo.

3.2.1. Métricas indicativas de calidad de servicio

Existen multitud de métricas indicativas de calidad de servicio que pueden ser utilizadas en la selección de un *proxy*, como la duración del saludo TCP, el tiempo hasta el primer *byte* de una petición HTTP, el tiempo hasta el último *byte* de una petición HTTP o el número de saltos hasta el *proxy* [Bat18, Dim17]. El tiempo hasta el primer *byte* es el tiempo que transcurre desde que el cliente inicia la petición al servidor hasta que recibe el primer byte de la respuesta [Dim17]. Por su parte, el tiempo de respuesta o tiempo hasta el último *byte* (TTLB, *Time to Last Byte*) es el tiempo transcurrido desde que el cliente inicia la petición del contenido hasta que recibe el último *byte* de la respuesta [Bat18].

En [Dim17] se argumenta que hay cierta correlación entre el TTFB y la carga del *proxy*. Utilizando esta métrica para la elección del *proxy*, se podría contribuir a que la carga se repartiese entre los *proxies* de la red. Sin embargo, como se observó en [Bor18], en algunas redes de comunidad el TTFB no se mantiene alto durante largos periodos de tiempo, es decir, la carga de los *proxies* se mantiene baja. En [Bat18] se argumenta, por su parte, que el tiempo de respuesta tiene una alta correlación con el rendimiento real del *proxy*, entendiendo como rendimiento el conjunto formado por la carga del *proxy*, la congestión del camino hasta el *proxy* y la congestión en su enlace de salida. Por lo tanto, el uso del tiempo de respuesta permite optimizar la calidad de servicio percibida por el usuario final. Es por esto que, en los experimentos de estimación, trataremos de estimar el tiempo de respuesta de los *proxies*.

3.2.2. Emulación de una red comunitaria

En las redes comunitarias, los *proxies* son aportados por entidades o usuarios que deciden compartir su conexión a Internet con el resto de usuarios de la red [Dim17]. Por este motivo, para poder obtener un conjunto de datos suficientemente grande para los experimentos de estimación en una red comunitaria, es necesario disponer de un número de clientes suficientemente grande y diverso, que además tengan acceso a un gran número de *proxies*. Además, en este tipo de redes la captura de los datos está limitada, ya que no se tiene acceso a todos los *proxies* como administrador, lo que permitiría tomar otro tipo de medidas o alterar su comportamiento para estudiar cómo afecta esto a las estimaciones. Cabe destacar que, en las redes comunitarias, los *proxies* son compartidos con el resto de usuarios, de modo que una alteración intencionada en el comportamiento de un *proxy* afectaría al resto de usuarios.

Como paso previo a la obtención de datos en una red comunitaria, el uso de bancos de pruebas (*testbenches*) es interesante, ya que ofrece ventajas como la posibilidad de acceso o control de todos los nodos (dentro de un entorno real en redes comunitarias no se tiene control sobre los *proxies*). Además, esta técnica es utilizada en varios trabajos de la literatura, como podría ser [Zhu17]. Por este motivo, para la obtención del conjunto de datos utilizados en este trabajo, se emula una red comunitaria en la plataforma PlanetLab [Chu03].

La red comunitaria emulada está formada por 8 clientes y 5 *proxies*. Cabe destacar que este número es bajo, debido a problemas encontrados en la instalación del *software* empleado tanto en los clientes como en los *proxies* en los nodos de PlanteLab. Tanto clientes como *proxies* se desplegaron sobre nodos de PlanetLab distribuidos por Europa y parte de Oriente Próximo. La relación entre estos clientes y *proxies* con los nodos de PlanetLab se puede observar en la tabla 3.1. En la elección de los nodos se trató de crear zonas de clientes próximos geográficamente, con un *proxy* cada una. Para ello se supuso que las máquinas, pertenecientes a organizaciones, estaban situadas en su sede, y se trató de crear 3 zonas distintas de nodos cercanos.

Cada zona estaría formada por dos *proxies* y tres clientes. Sin embargo, una vez tomados los datos, los parecidos entre las medidas de los clientes de cada zona no eran tan evidentes. Además, uno de los *proxies* y uno de los clientes no funcionaron bien en el periodo de toma de muestras, por lo que fueron excluidos del análisis y del conjunto de datos. El número de clientes y de *proxies* utilizado, aunque pequeño, permite formar una matriz de puntuaciones lo suficientemente grande y diversa como para poder evaluar el funcionamiento de los algoritmos de factorización matricial. La principal desventaja del bajo número de nodos es la dificultad encontrada para verificar hasta qué punto se pueden reducir el número de *proxies* que cada cliente sondea mientras sin reducir la precisión de las estimaciones.

Durante el proceso de obtención de los datos, cada cliente sondeó a todos los *proxies* de forma periódica cada 10 segundos durante 2 días. Este periodo de muestreo fue elegido para garantizar que los datos en cada cliente estén actualizados, manteniendo una sobrecarga baja. Además, es el periodo utilizado en [Dim17] para el intercambio de información entre clientes. En cada instante de muestreo se realizó una ronda de medidas, en la que todos los clientes sondeaban a todos los *proxies*, dando lugar a una matriz de medidas completa.

Para la obtención de los datos, los nodos que llevaron a cabo el papel de *proxy* corrían un *script*[1] de Python, que implementa un *proxy* ligero. Por su parte, los clientes realizaron los sondeos mediante el comando `curl` de Linux accediendo siempre a la misma página Web[2], que contiene un fichero de 1Mb, y siempre que se logró descargar el fichero se registró el tiempo de respuesta.

3.3. Descripción de las muestras

En la tabla 3.2 se puede observar el número de muestras tomadas por cada cliente a cada *proxy*. Dado el tiempo en el que se estuvieron tomando medidas, el número de muestras para pareja cliente-*proxy* debería ser de en torno a 17.280. Sin embargo, debido a problemas de comunicación entre cliente y *proxy*, o por dificultades para acceder al contenido por parte del *proxy*, algunas muestras no pudieron ser tomadas. Además, sólo se recogieron las muestras cuando el código de la respuesta HTTP fue 200 (OK), por lo que algunas muestras fueron descartadas. Este efecto se puede ver en el menor número de medidas recogidas por los 8 clientes para el *proxy* 5 respecto al resto de *proxies*, que se

[1] "Lightweigth HTTP, HTTPS, WebSockets Proxy Server in a single Python file", GitHub Repository, http://github.com/abhinavsingh/proxy.py, Última visita: 20 de junio de 2019

[2] http://ovh.net/files/1Mb.dat

Identificador	Nodo
Proxy 1	ple3.planet-lab.eu
Proxy 2	planet4.cs.huji.ac.il
Proxy 3	cse-white.cse.chalmers.se
Proxy 4	planetlab1.informatik.uni-kl.de
Proxy 5	planetlab1.cs.aueb.gr
Cliente 1	cse-yellow.cse.chalmers.se
Cliente 2	mars.planetlab.haw-hamburg.de
Cliente 3	pl2.uni-rostock.de
Cliente 4	planet3.cs.huji.ac.il
Cliente 5	planetlab3.di.unito.it
Cliente 6	planetlab-1.ing.unimo.it
Cliente 7	planetlab-2.cs.ucy.ac.cy
Cliente 8	ple43.planet-lab.eu

Tabla 3.1: Relación entre clientes y *proxies* utilizados, y los nodos de PlanteLab donde se implementan.

puede deber a fallos en la conexión con dicho *proxy*. Cabe destacar que estos problemas se dan también en las redes de comunidad.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Total
Proxy 1	15.476	15.486	15.474	14.697	14.742	14.678	15.472	15.502	121.519
Proxy 2	15.474	15.484	15.474	14.697	14.742	14.678	15.470	15.500	121.519
Proxy 3	15.474	15.485	15.473	14.697	14.742	14.678	15.470	15.500	121.525
Proxy 4	15.476	15.486	15.474	14.697	14.742	14.678	15.471	15.501	121.525
Proxy 5	11.664	11.666	11.665	14.697	14.742	14.678	11.658	11.667	102.437
Total	73.564	73.607	73.560	73.485	73.710	73.390	73.541	73.670	588.527

Tabla 3.2: Número de medidas registradas por pareja cliente-*proxy*. En la columna y fila "Total"se indican el número de muestras por cliente y *proxy*, respectivamente. En la intersección se indica el número de muestras totales.

Para entender mejor las medidas, en la figura 3.1 se puede observar la evolución temporal del tiempo de respuesta medido por cada cliente para cada *proxy*. Es fácil apreciar diferencias entre las medidas tomadas para un mismo *proxy* por distintos clientes. Esto da sentido a nuestro problema, ya que si las medidas fuesen parecidas no sería necesaria la estimación. Se puede ver cómo las medidas tomadas por el cliente 4 presentan una gran variabilidad, asi como las tomadas para el *proxy* 2. En las figuras 3.2, 3.3 y 3.4 se puede ver la evolución del tiempo de respuesta a lo largo del tiempo para algunas parejas cliente-*proxy*. En general el tiempo de respuesta se mantiene bastante bajo, en torno al segundo, con algún pico de valor superior que no dura más de una muestra, como en el caso de la figura 3.2 justo antes de la ronda de medidas 6750.

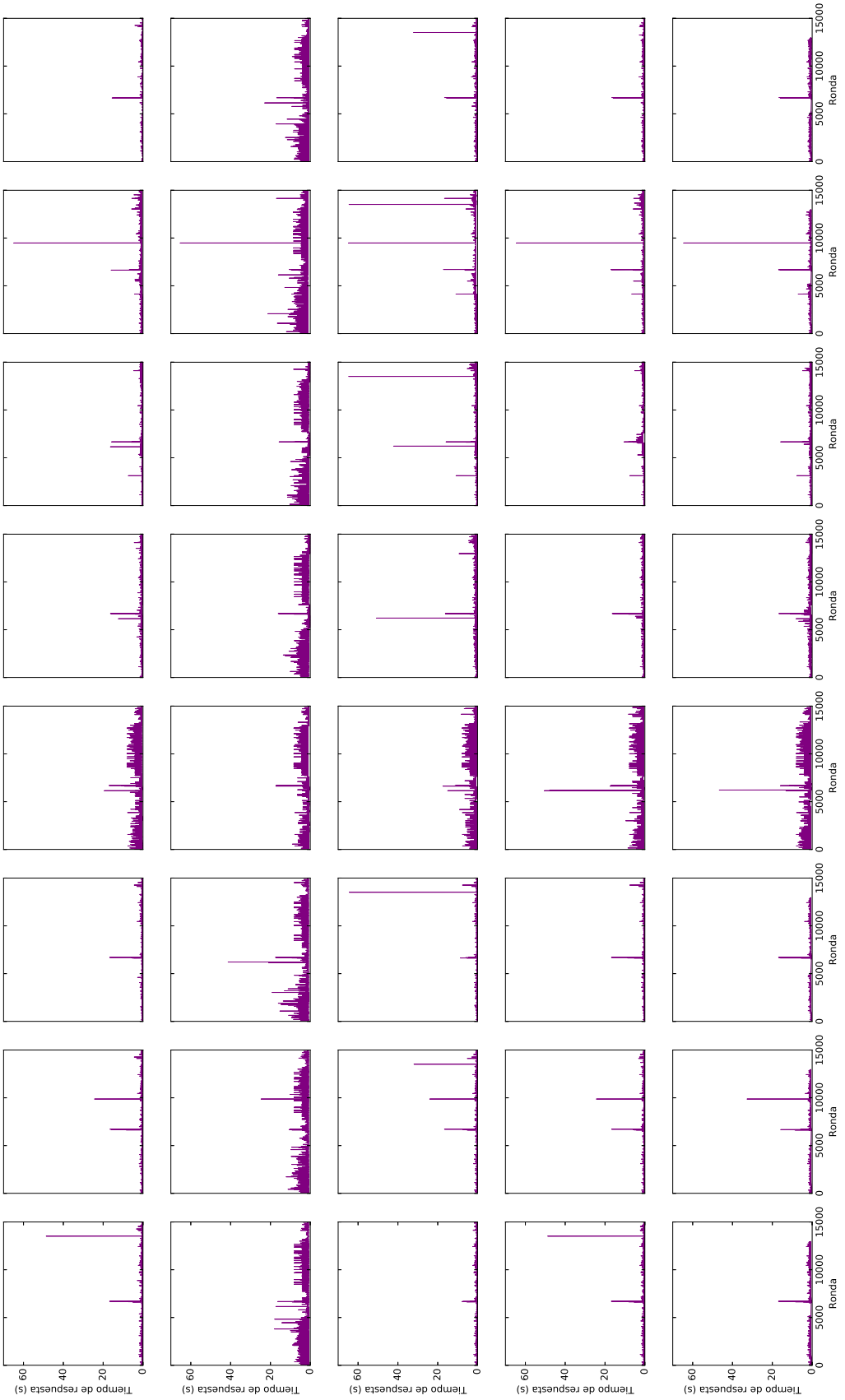


Figura 3.1: Evolución temporal del tiempo de respuesta para cada pareja cliente-*proxy*. En filas, la evolución del valor medido para cada cliente del 1 al 8 de iquierda a derecha. En columnas, el valor para cada *proxy* del 1 al 5 de arriba a abajo

Por otro lado, en la figura 3.5 se muestra la distribución estadística de las medidas. Se puede comprobar que se aleja del caso normal en la mayoría de las parejas cliente-*proxy*, como en el caso del cliente 1 y el *proxy* 2, o el del cliente 6 y el *proxy* 4. Además, en algunos casos la variabilidad entendida como el tamaño de la caja, que representa la distancia entre cuartiles, es elevada, como en el caso de las medidas tomadas por el cliente 4 para los *proxies* 1, 2 y 3, o las medidas tomadas por el cliente 7 para los *proxies* 1 y 3. Esto coincide con la idea intuitiva extraída a partir de la evolución temporal observada en la figura 3.1, y podría implicar una dificultad mayor para los algoritmos de factorización matricial a la hora de realizar la estimación, suposición que trataremos de comprobar en el capítulo 4.

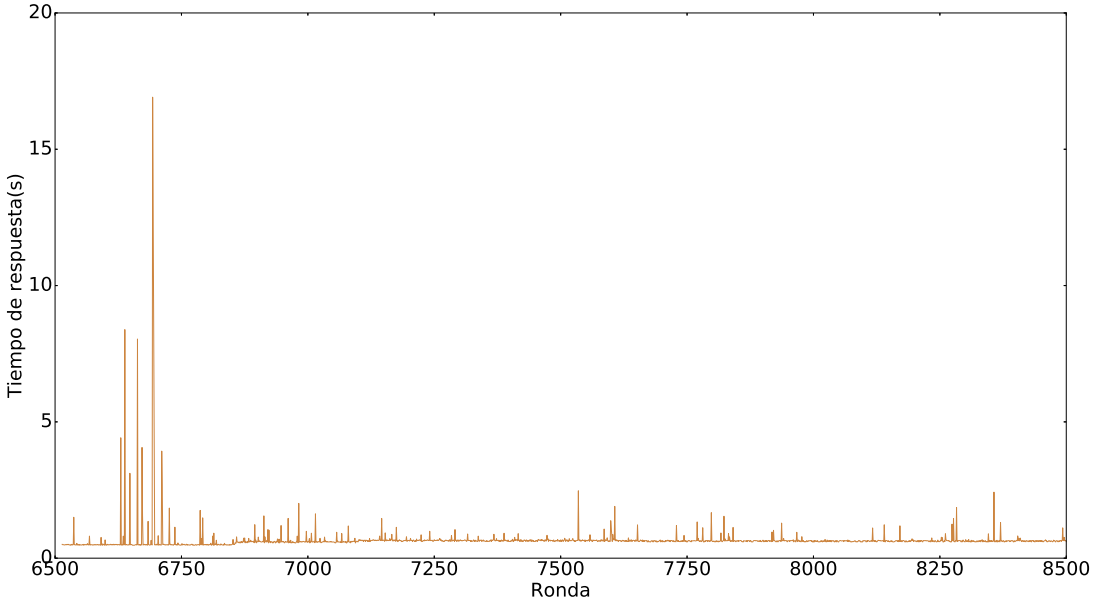


Figura 3.2: Porción ampliada de la evolución del valor medido para la pareja cliente 1 *proxy* 5.

3.4. Algoritmos de estimación empleados

Para la estimación del tiempo de respuesta en el contexto de las redes comunitarias, podemos formular el problema de forma similar como hacíamos en el capítulo 2 para la factorización matricial en un caso general. Así, tendríamos una matriz $M \in \mathbb{R}^{c \times p}$, siendo c el número de clientes y p el número de *proxies*. De esta forma, el elemento m_{jk} pasaría a ser ahora el valor del tiempo de repuesta que el cliente j mide del *proxy* k. Por lo tanto, una vez expresada la matriz de medidas de forma compatible con la factorización matricial, el proceso sería el mismo que el explicado en la sección 2.3. Como antes, en la matriz M habría elementos de valor desconocido, que corresponderían en un entorno real con aquellas parejas cliente-*proxy* para las que no habría valores medidos. Es precisamente el valor de estos elementos el que se pretende estimar.

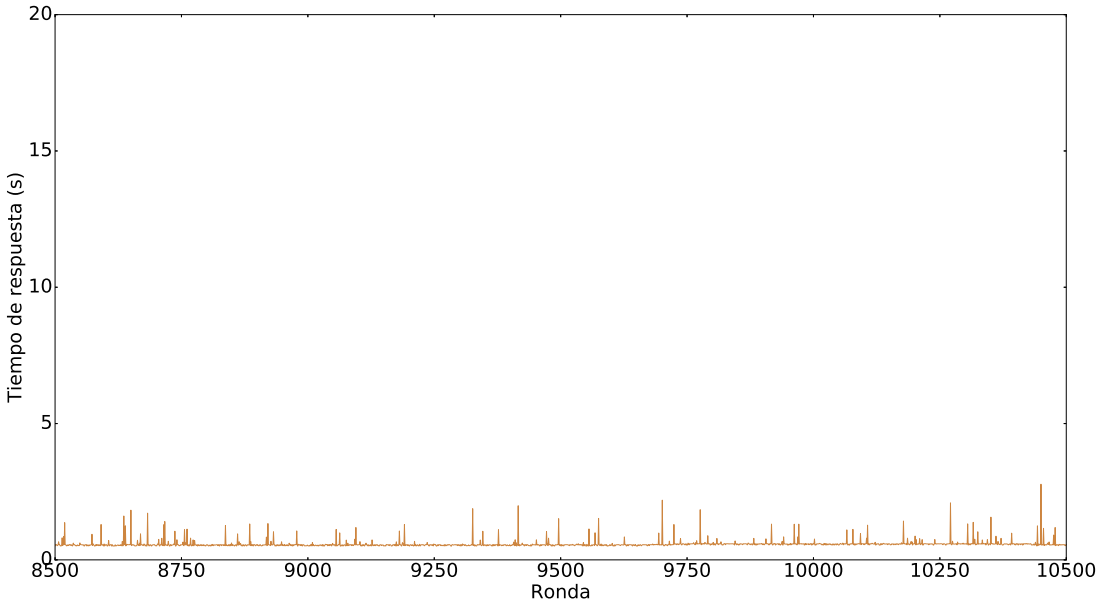


Figura 3.3: Porción ampliada de la evolución del valor medido para la pareja cliente 5 *proxy* 5.

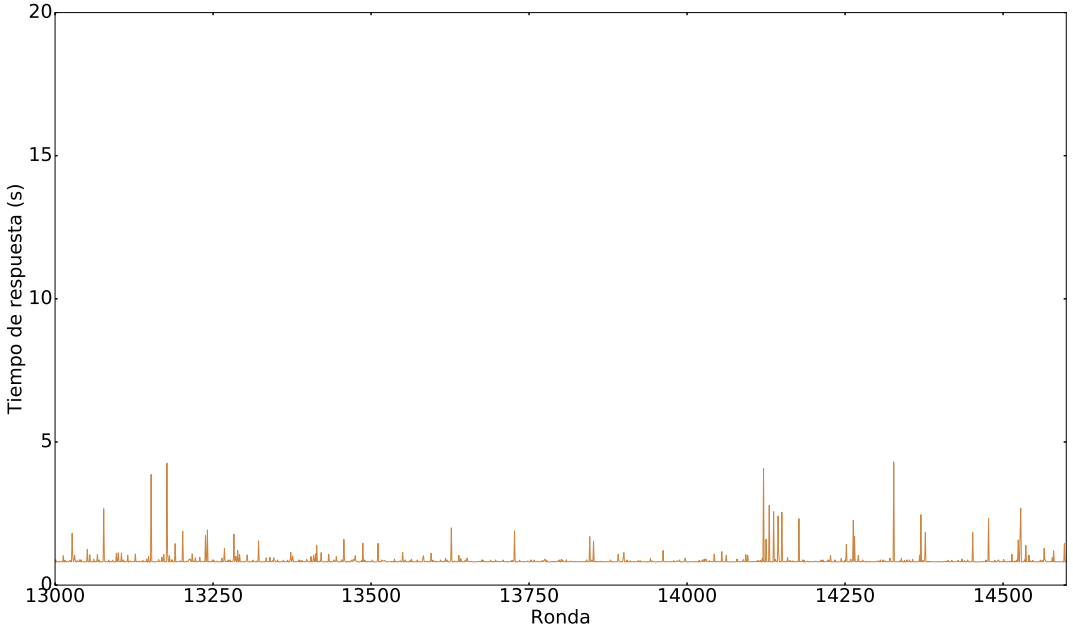


Figura 3.4: Porción ampliada de la evolución del valor medido para la pareja cliente 8 *proxy* 2.

3.4.1. Algoritmos de factorización matricial

Los algoritmos utilizados son AMF, NTF y PMF. AMF es el algortimo desarrollado por [Zhu17], y por sus características adaptativas que permiten tener en cuenta la evolución de las medidas, y por el hecho de ser un algoritmo en línea, parece idóneo para el

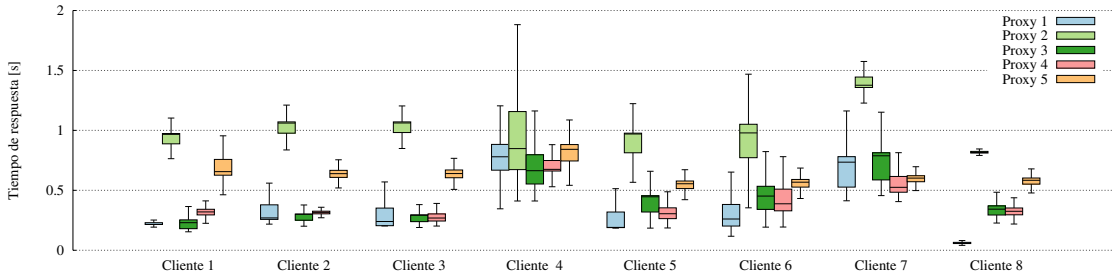


Figura 3.5: Distribución estadística de las medidas por cada clientes para cada *proxy* en forma de diagramas de cajas.

problema tratado en este trabajo. Por su parte, NTF es un algoritmo que también tiene en cuenta la evolución temporal de las medidas, por lo que se ha elegido para comparar sus resultados con AMF. Por último, PMF es un algoritmo clásico de factorización matricial para el que, aunque no permita incluir la evolución temporal de las medidas, es interesante ver su rendimiento en un problema como el de la estimación del tiempo de respuesta de los *proxies* en una red comunitaria.

La formulación matemática de AMF es la siguiente [Zhu17]: en primer lugar, se modelan los datos a estimar como la suma de un sesgo de usuario, q_{ui}, un sesgo de de servicio, $q_{s,j}$, y el producto de las dos matrices fruto de la factorización matricial, $U_i(t)^T S_j(t)$:

$$\hat{R}_{ij}(t) = q_{ui}(t) + q_{s,j}(t) + U_i(t)^T S_j(t)$$

A continuación se emplea una función logística (*logistic function*), $g(x) = 1/(1 + e^{-x})$ para trasladar el valor de la estimación entre 0 y 1. Se define también la función de costes en cada instante como la suma del cuadrado de los errores de estimación en aquellos elementos r_{ij} de la matriz que han sido observados, junto con un término de regularización en el que influyen los valores de $U_i(t)$, $S_j(t)$, $q_{s,j}$ y q_{ui}. A partir de esta función de costes se tratan de obtener los parámetros mediante un descenso de gradiente estocástico, que se actualiza de forma incremental con cada nueva matriz de medidas. Si llamamos g_{ij} a $g(\hat{R}_{ij}(t))$, en cada instante de medida los parámetros se actualizarían como:

$$U_i \longleftarrow U_i - \eta[(g_{ij} - r_{ij})g'_{ij}S_j/r_{ij}^2 + \lambda U_i]$$

$$S_j \longleftarrow S_j - \eta[(g_{ij} - r_{ij})g'_{ij}U_i/r_{ij}^2 + \lambda S_j]$$

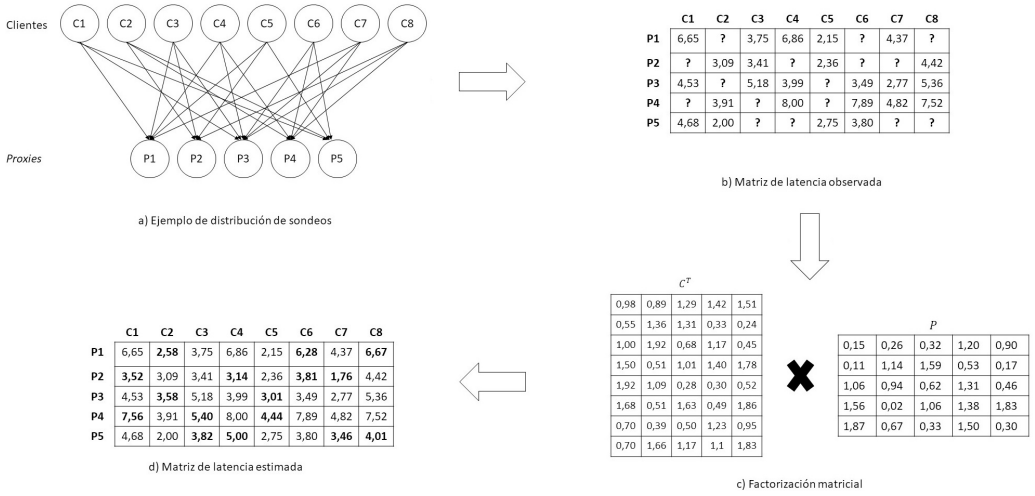
$$q_{ui} \longleftarrow q_{ui} - \eta[(g_{ij} - r_{ij})g'_{ij}/r_{ij}^2 + \lambda q_{ui}]$$

$$q_{sj} \longleftarrow q_{sj} - \eta[(g_{ij} - r_{ij})g'_{ij}/r_{ij}^2 + \lambda q_{sj}]$$

Para permitir la adición de nuevos artículos y usuarios, la tasa de aprendizaje no puede ser fija. Por esto se emplean una serie de pesos adaptativos que permiten controlar el tamaño del paso en la dirección contraria a la dirección de máximo crecimiento de la matriz de costes. Por su parte, las formulaciones matemáticas de PMF y de NTF aparecen descritas en [Sal07] y [Zha14], respectivamente.

Una implementación de estos algoritmos está disponible en GitHub [3]. Es la utilizada en [Zhu17], y se trata de un entorno en C++ con un envoltorio que permite su uso como si se tratase de una función de Python.

[3]WS-DREAM GitHub Repository, http://wsdream.github.io, Última visita: 20 de junio de 2019.



	C1	C2	C3	C4	C5	C6	C7	C8
P1	6,65	?	3,75	6,86	2,15	?	4,37	?
P2	?	3,09	3,41	?	2,36	?	?	4,42
P3	4,53	?	5,18	3,99	?	3,49	2,77	5,36
P4	?	3,91	?	8,00	?	7,89	4,82	7,52
P5	4,68	2,00	?	?	2,75	3,80	?	?

C^T

0,98	0,89	1,29	1,42	1,51
0,55	1,36	1,31	0,33	0,24
1,00	1,92	0,68	1,17	0,45
1,50	0,51	1,01	1,40	1,78
1,92	1,09	0,28	0,30	0,52
1,68	0,51	1,63	0,49	1,86
0,70	0,39	0,50	1,23	0,95
0,70	1,66	1,17	1,1	1,83

P

0,15	0,26	0,32	1,20	0,90
0,11	1,14	1,59	0,53	0,17
1,06	0,94	0,62	1,31	0,46
1,56	0,02	1,06	1,38	1,83
1,87	0,67	0,33	1,50	0,30

	C1	C2	C3	C4	C5	C6	C7	C8
P1	6,65	**2,58**	3,75	6,86	2,15	**6,28**	4,37	**6,67**
P2	**3,52**	3,09	3,41	**3,14**	2,36	**3,81**	**1,76**	4,42
P3	4,53	**3,58**	5,18	3,99	**3,01**	3,49	2,77	5,36
P4	**7,56**	3,91	**5,40**	8,00	**4,44**	7,89	4,82	7,52
P5	4,68	2,00	**3,82**	**5,00**	2,75	3,80	**3,46**	**4,01**

Figura 3.6: Figura explicativa del proceso seguido en los experimentos de estimación.Los clientes sondean a algunos de los *proxies*, dando lugar a una matriz de medidas con elementos desconocidos, y se obtiene una estimación de la matriz de medidas completa mediante factorización matricial.

En la figura 3.6 se puede observar de forma gráfica el proceso seguido en los experimentos. En primer lugar, los clientes sondean a algunos de los *proxies*, dando lugar a una matriz de medidas donde algunos de los elementos son desconocidos. A continuación se descompone la matriz de medidas por factorización en dos matrices cuyo producto da lugar a la estimación de la matriz de medidas completa.

3.4.2. Algoritmos de referencia

Para poder evaluar la precisión de los algoritmos de factorización matricial utilizados para la estimación de las medidas del tiempo de respuesta, es necesario comparar los resultados obtenidos con los de algún algoritmo de referencia sencillo. Con este fin, se proponen dos algoritmos de estimación básicos. Para explicarlos, partamos de la matriz tridimensional de medidas, o tensor. Por cada instante de medida, tenemos una matriz $M \in \mathbb{R}^{c \times p}$, siendo c el número de clientes y p el número de *proxies*. En cada instante de medida habrá $k \leq c \times p$ medidas reales tomadas, y $l = (c \times p) - k$ valores desconocidos. Teniendo todo esto en mente, los algoritomos de base se definen como:

- **Algoritmo de referencia 1**: En este algoritmo se estiman los l valores desconocidos como el valor medio de las k muestras tomadas. Este algoritmo no tiene en cuenta el *proxy* para el que se estiman las medidas, sino que asigna el mismo valor a todos los elementos desconocidos de la matriz de muestras correspondiente al instante actual. Es utilizado en [Zhu17] entre otros para comparar los resultados obtenidos con los del algoritmo propuesto.

- **Algoritmo de referencia 2**: En este algoritmo se tiene en cuenta a qué *proxy* pertenecen los valores desconocidos. Así, por cada ronda de medidas, se estiman los

valores desconocidos correspondientes a cada *proxy* como el valor medio de las medidas tomadas para ese *proxy* por el resto de clientes. Se conoce como IMEAN(*Item MEAN*, media de artículo), y es utilizado en [Zhe13] y en [Lo12] evaluar los resultados obtenidos por el algoritmo propuesto.

3.5. Descripción de los experimentos de estimación

Para la realización de los experimentos de estimación, se divide la matriz de medidas M, que contiene todos los valores medidos, en una matriz de test T y una de entrenamiento W. Los elementos que pertenecen a cada una de las matrices se obtienen de forma aleatoria de la matriz M, rellenando con un número negativo[4] el resto de posiciones de las matrices T y W. Como se indica en la sección 3.2.2, por cada instante de muestreo se realiza una ronda de medidas, dando lugar a una matriz de medidas. Con cada nueva ronda de medidas se obtienen las matrices de test y entrenamiento a partir de la de medidas. A continuación, con la matriz W se actualiza el modelo, y se trata de estimar aquellos valores mayores que 0 contenidos en la matriz T. A partir de estas estimaciones, el error se calcula como la diferencia, en valor absoluto, entre el valor real contenido en la matriz de test y el valor estimado. Por cada experimento, se obtiene el MAE(error absoluto medio, *Mean Absolute Error*) como el promedio de estos errores.

Por cada ronda de medidas, los *proxies* que cada cliente sondea se eligen de forma aleatoria, si bien el número de *proxies* que cada cliente sondea es fijo para el experimento completo. Cabe destacar que todos los algoritmos emplean las mismas matrices de entrenamiento y test. En una red comunitaria, cada cliente sondearía a un conjunto reducido de *proxies*, lo que implicaría una densidad baja de la matriz de entrenamiento. Para evaluar la influencia de este parámetro, se han hecho pruebas con 2, 3 y 4 *proxies* sondeados por cada cliente, que dan lugar a densidades de la matriz de entrenamiento del 40 %, 60 % y 80 %, respectivamente. Se puede observar que ninguna de ellas es muy baja, ya que disponemos de un número bajo de *proxies*. Por último, las medidas que no se emplean para el entrenamiento son empleadas para calcular el error de las estimaciones. Cabe destacar que, para el caso de AMF, se realiza una transformación de los datos antes y después de actualizar el modelo, con el fin de adecuar su distribución de probabilidad a la factorización matricial. Esto es así porque estos algoritmos funcionan mejor con distribuciones de tipo normal [Zhu17], que son las que normalmente describen los datos agregados de valoración que los usuarios hacen de artículos en, por ejemplo, un comercio electrónico. Pero las distribuciones de parámetros de calidad de servicio como el tiempo de respuesta son asimétricas y con cola pesada. En [Zhu17] se demuestra que una transformación de los datos mejora mucho la precisión de las estimaciones.

Puesto que se escogen los *proxies* a sondear en cada instante de forma aleatoria, se han realizado 5 experimentos completos, para evaluar la dispersión del error entre experimentos. De esta forma, se ha computado el valor medio del error obtenido para los 5 experimentos de estimación, y la desviación estándar del error entre experimentos. El código empleado tanto para la realización de los experimentos de estimación como para la

[4]Esto es así en la implementación ofrecida en GitHub de los distintos algoritmos usados por [**?**], y nosotros hemos seguido el mismo convenio. Un valor negativo debe entenderse como un valor desconocido.

Parámetro de AMF	Valor
Dimensionalidad de factores latentes	10
Tasa de aprendizaje	0,8
Parámetro de regularización	0,0003
Número máximo de iteraciones	50
Umbral de convergencia	6e-3
Peso de la media exponencial móvil	0,3

Parámetro de NTF	Valor
Dimensionalidad de factores latentes	10
Parámetro de regularización	40
Número máximo de iteraciones	300

Parámetro de PMF	Valor
Dimensionalidad de factores latentes	10
Parámetro de regularización	0
Valor inicial de la tasa de aprendizaje	0,01
Número máximo de iteraciones	600

Tabla 3.3: Parámetros utilizados para cada algoritmo de factorización matricial en los experimentos de estimación del valor del tiempo de respuesta.

obtención de las tablas de valor medio y desviación estándar del error está disponible[5], para la comprobación de los resultados. Hay que mencionar que este código parte del utilizado en [Zhu17] para la implementación de los algoritmos, con modificaciones en la parte de tratamiento de los datos y experimentos de estimación. En concreto, se han hecho los cambios oportunos para controlar la densidad de la matriz de entrenamiento mediante el número de *proxies* sondeados por cada cliente. También se ha cambiado la forma en que se obtiene la métrica de error, obteniendo la media de todos los errores en cada experimento y, a continuación, el valor medio y la desviación típica. Por último, se han hecho los cambios necesarios para que todos los algoritmos empleen la misma matriz de test y de entrenamiento, generadas de forma aleatoria en cada experimento.

Otro aspecto importante a tener en cuenta son los parámetros de los algoritmos, que determinan en cierto modo su rendimiento. En este trabajo se han obtenido mediante prueba y error unos valores para cada parámetro con el fin de mejorar el funcionamiento de los algoritmos. Sin embargo, no se ha hecho un barrido exhaustivo para obtener las mejores combinaciones de parámetros. La tabla 3.3 indica los parámetros utilizados por cada uno de los algoritmos.

3.6. Métricas de rendimiento

Para medir el rendimiento de los diferentes algoritmos de factorización matricial, es necesario computar los errores que se producen en las estimaciones. Las opciones más populares son el MAE y el RMSE (*Root Mean Square Error*, raíz del error cuadrático medio). Si consideramos un vector de errores de tamaño N, y que e_i es el i-ésimo elemento de dicho vector, cuyo valor se calcula como la diferencia entre el valor real y el estimado,

[5]Github Repository: https://github.com/gsic-emic/tfm_2019_bores

se puede definir el MAE como:

$$MAE = \frac{\sum_{i=0}^{N-1} |e_i|}{N} \tag{3.1}$$

y el RMSE como:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1} |e_i|^2}{N}} \tag{3.2}$$

Algunas otras métricas son [Zhu17] la mediana del error relativo (MRE, *Median Relative Error*) o el percentil 90 del error relativo (NPRE, *Ninety-Percentile Relative Error*). En este trabajo se emplea el MAE, dado que se trata una métrica más sencilla de calcular (el cálculo del RMSE implica el cálculo de raíces cuadradas y de potencias cuadráticas), ya que solo requiere del cálculo de sumas y una división. Además, se trata de una métrica ampliamente utilizada (ej. [Lo12, Zhe13]) para la evaluación de la precisión de algoritmos de estimación empleados en problemas similares al tratado en este trabajo. Por otra parte, de acuerdo con [Her04], es más sencilla de entender que otras y ha sido probada su utilidad para comparar el resultado de diferentes algoritmos.

3.7. Conclusiones

En este capítulo se han descrito tanto los datos empleados en la estimación como la forma en que han sido obtenidos. Cabe destacar en este sentido los beneficios del uso de bancos de prueba para la obtención de datos, ya que permite la alteración del funcionamiento de los *proxies* artificialmente sin interferir en la interacción del resto de usuarios con los *proxies* de una red comunitaria real. Los experimentos así realizados permiten por lo tanto, obtener un conjunto de datos lo suficientemente bueno como para poder realizar una primera aproximación a la estimación del tiempo de respuesta de los *proxies* en redes comunitarias. Sin embargo, sería interesante repetir los experimentos en un entorno de red comunitaria real para comprobar que los algoritmos de estimación ofrecen buenos resultados también en el caso real.

Por otro lado, se ha confirmado la necesidad de la estimación de los valores desconocidos de la matriz de muestras en el caso de que cada cliente sondee a un subconjunto de *proxies* y comparta estas medidas con el resto. Si las muestras tomadas por todos los clientes a un mismo *proxy* fuesen parecidas, la estimación no tendría sentido, ya que los clientes podrían utilizar directamente las medidas que reciben del resto de clientes para elegir el *proxy* de acceso a la Web. Sin embargo, se ha visto que las medidas tomadas para un mismo *proxy* por varios clientes son diferentes en muchos casos. Además, las diferencias y similitudes entre las muestras tomadas por diferentes clientes permiten explotar las ventajas de los algoritmos de factorización matricial.

También se han descrito los algoritmos de referencia utilizados, así como la métrica de rendimiento, con los que evaluar el rendimiento de los algoritmos de factorización utilizados. Por último, se han detallado los experimentos de estimación y el tratamiento de los datos realizado.

Capítulo 4

Resultados y análisis

4.1. Introducción

Una vez presentado el problema a tratar en este trabajo y los algoritmos utilizados, y una vez descritos tanto los datos empleados como los experimentos y las métricas de evaluación de los resultados, se realizan los experimentos de estimación. como se indica en el capítulo anterior. A partir del conjunto de datos inicial, con todas las medidas, se obtienen cinco conjuntos de entrenamiento y de test eligiendo de forma aleatoria muestras para cada densidad de matriz de entrenamiento.

Con la matriz de entrenamiento se actualiza el modelo, y se trata de estimar los valores contenidos en la matriz de test. De esta forma, por cada uno de los experimentos y densidad de la matriz de entrenamiento se obtiene un conjunto de estimaciones cuyo error se puede obtener calculando la diferencia con el valor medido real contenido en la matriz de test. A partir de estos errores se puede calcular tanto el MAE global como por pareja cliente-*proxy*, por cliente y por *proxy* para cada experimento y densidad. Se calcula a continuación el valor medio entre experimentos y la desviación típica, para validar la consistencia de los resultados. También se pueden representar las estimaciones en cada ronda para compararlas visualmente con los valores medidos reales.

En este capítulo se muestran y analizan, en primer lugar, los valores medios entre rondas de experimentos para cada densidad de la matriz de entrenamiento, tanto para el caso global como para cada pareja cliente-*proxy*, para cada cliente y para cada *proxy*. También se describe y comenta la evolución de las estimaciones con el tiempo para cada algoritmo empleado comparándolas con las medidas reales, para alguna pareja cleinte-*proxy*. Por último, se plantean algunas de las conclusiones extraídas de este análisis.

4.2. Análisis de resultados

En esta sección se analizan los resultados obtenidos en los experimentos de estimación descritos en el capítulo anterior en términos de error en la estimación. En primer lugar se describen y discuten los valores de MAE obtenidos de forma global por cada algoritmo y para cada densidad de la matriz de entrenamiento. A continuación se muestra en detalle el MAE obtenido por cada pareja cliente-*proxy*.

4.2.1. Resultados globales

Una vez realizados los experimentos de estimación descritos sobre el conjunto de datos utilizado, en la tabla 4.1 se muestran los resultados globales de cada algoritmo para cada densidad considerada. Puesto que los conjuntos de entrenamiento y de test se obtienen de forma aleatoria en cada ronda, se ha computado en primer lugar el MAE de cada ronda, para después calcular el valor medio y la desviación típica entre experimentos, que es el dato que se reporta.

Se observa un mejor funcionamiento de AMF en todos los casos, con una precisión elevada y mejor que la del resto de algoritmos. Esto podría ser debido, en gran medida, a la transformación de los datos que se hace antes y después de su uso, que permite adecuarlos al algoritmo de factorización. Esta transformación es propia de AMF, y no ha sido utilizada con el resto de algoritmos.

El comportamiento del algoritmo NTF también es bueno, siendo el peor el de PMF, cuyos resultades son peores incluso que los de los algoritmos de referencia. Este buen comportamiento podría justificarse con la sensibilidad que tiene NTF a la evolución temporal de las medidas. Sin embargo, el coste computacional de este algoritmo es superior al de AMF, ya que requiere factorizar el tensor completo por cada nueva ronda de medidas. En cuanto a los algoritmos de referencia, el comportamiento del algoritmo de referencia 2 es mejor que el del 1, puesto que tiene en cuenta a qué *proxy* corresponden las medidas reales y las estimadas, mientras que el algoritmo de referencia 1 asigna el mismo valor a todos los *proxies*, cometiendo un gran error cuando las medidas correspondientes a varios *proxies* son muy diferentes en una misma ronda de medidas.

En cuanto al efecto de la densidad de la matriz de entrenamiento, se puede observar cómo el aumento del número de *proxies* sondeados por cada cliente y ronda revierte en una mejora del resultado, que es más evidente para el algoritmo PMF, y que es menor en los otros casos. Esto es debido, probablemente, a que contamos con un número reducido de clientes y *proxies*, en comparación con los típicamente utilizados por este tipo de algoritmos, y la densidad es elevada todos los casos.

Densidad	AMF (x)	NTF	PMF	Alg. de referencia 1	Alg. de referencia 2
40 %	$\mathbf{0,104 \pm 4,6e-4}$	$0,134 \pm 2,36e-3$	$0,569 \pm 1,37e-3$	$0,315 \pm 1,2e-4$	$0,231 \pm 5,2e-4$
60 %	$\mathbf{0,100 \pm 4,4e-4}$	$0,124 \pm 5,75e-3$	$0,438 \pm 7e-4$	$0,313 \pm 4,5e-4$	$0,220 \pm 2,9e-4$
80 %	$\mathbf{0,096 \pm 5,9e-4}$	$0,115 \pm 2,18e-3$	$0,360 \pm 1,8e-3$	$0,313 \pm 3,2e-4$	$0,214 \pm 4,2e-4$

Tabla 4.1: Media y desviación estándar del MAE global obtenido cada algoritmo y cada densidad.

4.2.2. Resultados por parejas de cliente-*proxy*

Para poder analizar más en profundidad el funcionamiento de estos algoritmos, en las tablas 4.2, 4.3, 4.4, 4.5 y 4.6 se muestra el rendimiento de cada algoritmo para cada pareja cliente-*proxy* para densidad 40 %. El resultado para el resto de densidades está recogido en el apéndice B. Estos resultados se pueden relacionar con lo que se observaba en la figura 3.5.

En el caso de AMF, NTF y el algoritmo de referencia 1, los valores medios de MAE más elevados que se obtienen para los clientes 1, 2, 3, 5, 6 y 7 son los relacionados con el *proxy* 2, que presentaba una mayor variabilidad en la figura 3.5, pero también un

mayor valor de tiempo de respuesta medio. Este *proxy* es también el que más se diferencia del resto, con lo que será más difícil estimar de forma certera sus valores de tiempo de respuesta. En el caso del algoritmo de referencia 2, el MAE obtenido para el *proxy* 2 no es tan elevado en comparación con los otros. La causa más probable es que en el resto de algoritmos influyen las medidas tomadas a otros *proxies*, mientras que en el algoritmo de referencia 2 sólo influyen las medidas que otros clientes realizan a ese *proxy*.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,050	0,055	0,051	0,170	0,109	0,119	0,108	0,203	$0{,}108 \pm 0{,}00070$
Proxy 2	0,136	0,141	0,139	0,138	0,183	0,193	0,209	0,137	$0{,}159 \pm 0{,}00178$
Proxy 3	0,043	0,044	0,041	0,163	0,107	0,116	0,100	0,051	$0{,}082 \pm 0{,}00050$
Proxy 4	0,057	0,050	0,057	0,158	0,113	0,125	0,087	0,056	$0{,}087 \pm 0{,}00052$
Proxy 5	0,059	0,053	0,056	0,151	0,093	0,091	0,073	0,053	$0{,}080 \pm 0{,}00049$
Media	0,069	0,069	0,070	0,156	0,122	0,130	0,118	0,102	$0{,}104 \pm 0{,}00038$

Tabla 4.2: Media del MAE obtenido por AMF para cada pareja de cliente y *proxy* con densidad 40 %.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,079	0,072	0,077	0,183	0,153	0,154	0,150	0,254	$0{,}140 \pm 0{,}00442$
Proxy 2	0,153	0,172	0,181	0,232	0,243	0,247	0,237	0,138	$0{,}200 \pm 0{,}00825$
Proxy 3	0,092	0,064	0,052	0,156	0,140	0,136	0,153	0,057	$0{,}105 \pm 0{,}00405$
Proxy 4	0,057	0,056	0,074	0,156	0,124	0,160	0,117	0,064	$0{,}100 \pm 0{,}00174$
Proxy 5	0,164	0,120	0,134	0,184	0,125	0,096	0,098	0,075	$0{,}125 \pm 0{,}00786$
Media	0,106	0,096	0,102	0,182	0,157	0,160	0,154	0,119	$0{,}134 \pm 0{,}00236$

Tabla 4.3: Media del MAE obtenido por NTF para cada pareja de cliente y *proxy* con densidad 40 %.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,736	0,682	0,707	0,514	0,693	0,684	0,533	0,787	$0{,}668 \pm 0{,}00117$
Proxy 2	0,466	0,452	0,453	0,637	0,564	0,547	0,497	0,587	$0{,}524 \pm 0{,}00097$
Proxy 3	0,672	0,633	0,636	0,490	0,504	0,511	0,451	0,643	$0{,}569 \pm 0{,}00431$
Proxy 4	0,576	0,597	0,630	0,473	0,570	0,518	0,527	0,641	$0{,}567 \pm 0{,}00353$
Proxy 5	0,413	0,444	0,444	0,512	0,543	0,548	0,597	0,533	$0{,}506 \pm 0{,}00166$
Media	0,581	0,567	0,581	0,525	0,575	0,562	0,517	0,643	$0{,}569 \pm 0{,}00137$

Tabla 4.4: Media del MAE obtenido por PMF para cada pareja de cliente y *proxy* con densidad 40 %.

De vuelta al algoritmo AMF, para los clientes que miden un valor parecido de tiempo de respuesta de un *proxy*, y que podrían ser considerados vecinos por el algoritmo de factorización, se obtienen los valores medios del MAE más bajos. Ejemplos de esto son los clientes 1, 2 y 3 para los *proxies* 1, 3 y 4. En la figura 3.5 se puede observar que estos clientes miden un valor muy similar de cada uno de estos *proxies*. En resumen, AMF es capaz de aprender cuáles son los clientes más parecidos para tener más en cuenta las medidas tomadas por estos a la hora de realizar las estimaciones.

Una discusión similar se podría realizar al respecto de los resultados de NTF, puesto que se puede observar en la tabla 4.3 que el valor medio del MAE es muy parecido para las parejas cliente-*proxy* destacados en el párrafo anterior. Sin embargo, en el resto de algoritmos esto no es tan evidente.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,383	0,306	0,343	0,257	0,367	0,344	0,181	0,536	$0{,}340 \pm 0{,}00076$
Proxy 2	0,454	0,528	0,532	0,366	0,457	0,447	0,927	0,322	$0{,}506 \pm 0{,}00209$
Proxy 3	0,388	0,329	0,338	0,190	0,219	0,208	0,226	0,277	$0{,}273 \pm 0{,}00082$
Proxy 4	0,291	0,297	0,342	0,205	0,313	0,269	0,152	0,286	$0{,}270 \pm 0{,}00071$
Proxy 5	0,153	0,122	0,124	0,322	0,164	0,129	0,112	0,109	$0{,}157 \pm 0{,}00079$
Media	0,343	0,327	0,347	0,267	0,307	0,282	0,330	0,316	$0{,}315 \pm 0{,}00012$

Tabla 4.5: Media del MAE obtenido por el algoritmo de referencia 1 para cada pareja de cliente y *proxy* con densidad 40 %.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,239	0,188	0,207	0,490	0,230	0,216	0,364	0,362	$0{,}286 \pm 0{,}00043$
Proxy 2	0,181	0,180	0,183	0,297	0,277	0,242	0,528	0,250	$0{,}267 \pm 0{,}00103$
Proxy 3	0,254	0,194	0,201	0,363	0,158	0,162	0,393	0,160	$0{,}236 \pm 0{,}00088$
Proxy 4	0,151	0,149	0,190	0,416	0,174	0,190	0,229	0,146	$0{,}205 \pm 0{,}00119$
Proxy 5	0,109	0,086	0,086	0,338	0,173	0,133	0,098	0,106	$0{,}144 \pm 0{,}00084$
Media	0,191	0,163	0,178	0,382	0,203	0,190	0,334	0,209	$0{,}231 \pm 0{,}00052$

Tabla 4.6: Media del MAE obtenido por el algoritmo de referencia 2 para cada pareja de cliente y *proxy* con densidad 40 %.

Cabe destacar los casos en los que las medidas de un cliente a un *proxy* se encuentran muy alejadas a las del resto de parejas cliente-*proxy*. Este es el caso del cliente 8 y el *proxy* 1 o del cliente 7 y el *proxy* 2. Como se puede ver en la figura 3.5, para estos casos las medidas se encuentran fuera del rango del resto de parejas cliente-*proxy*. Por lo tanto, los algoritmos estiman un valor próximo a los medidos por el resto de clientes y pese a presentar una variabilidad en las medidas reducida, el error que se comete es elevado. Esto se puede ver en las tablas de resultados, donde el error para las parejas cliente 8 *proxy* 1 y cliente 7 *proxy* 2 presentan el máximo valor medio de MAE del cliente y del *proxy*. En el único algoritmo en que esto no ocurre de forma tán evidente es en el algoritmo de referencia 2, puesto que las medidas del resto de *proxies* tomadas por el cliente no tienen efecto.

PMF se comporta bastante mal en todos los casos, aunque presenta un mejor rendimiento para el *proxy* 2, que por otro lado es el más variable. Por su parte, el algoritmo de referencia 1, por definición, no tiene en cuenta parecidos ni entre clientes ni entre *proxies*. Puesto que predominan los *proxies* con un valor bajo de tiempo de respuesta, el valor medio de cada ronda de medidas es más cercano a este valor, y los resultados obtenidos son mejores para los *proxies* de este tipo. Por último, el algoritmo de referencia 2 no tiene en cuenta el parecido entre clientes, de forma que los resultados obtenidos dependen únicamente de la variabilidad entre medidas de distintos clientes a cada *proxy*.

4.3. Ejemplos de estimaciones

Una vez analizados los resultados en términos de MAE obtenido por cada algoritmo, se puede continuar el análisis a través de figuras que muestren la evolución con el tiempo de las medidas reales y de las estimaciones. Así, en las figuras 4.1, 4.2 y 4.3 se muestra

la evolución de las estimaciones frente a los valores reales para tres parejas cliente-*proxy* en una región ampliada. Corresponden a una de los conjuntos experimentales para una densidad de la matriz de entrenamiento del 40 %. Se puede observar que, por lo general, AMF es capaz de seguir bastante bien la evolución de la tiempo de respuesta, adaptándose incluso a picos espontáneos como el que se puede observar en la ronda 7798 en la figura 4.2. El comportamiento de NTF es bastante bueno, siendo el peor el de PMF, lo que concuerda con lo comentado sobre las tablas de resultados.

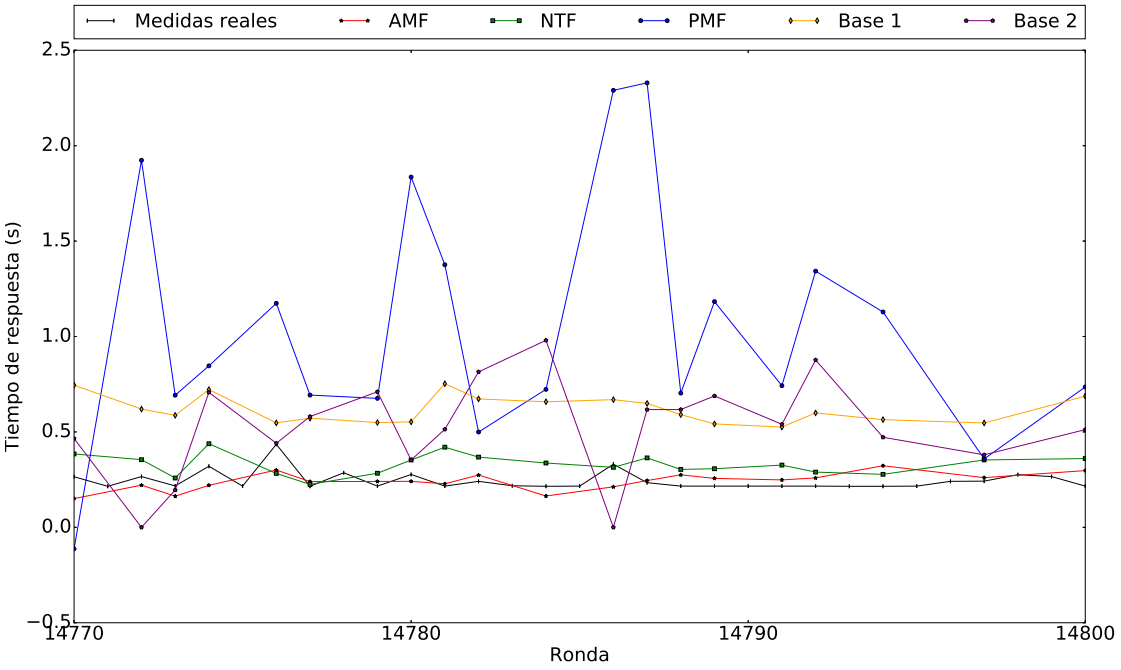


Figura 4.1: Fragmento de los resultados de estimación para cada uno de los algoritmos utilizados en el caso del cliente 1 y el *proxy* 1.

En la figura 4.1 se puede ver el contraste entre el buen seguimiento que hace AMF de los valores medidos y la distancia relativamente elevada entre los valores reales y los estimados por el resto de algoritmos. Esto se podría deber a la asimetría de la distribución de los datos, a la que AMF hace frente mediante una transformación de los mismos. Además, en la figura 4.2 se puede observar cómo casi todos los algoritmos son capaces de seguir algunos picos con bastante precisión los valores medidos. De nuevo se observa que, de entre todos los algoritmos, el que mejores resultados presenta en esta situaciones es AMF, seguido por NTF.

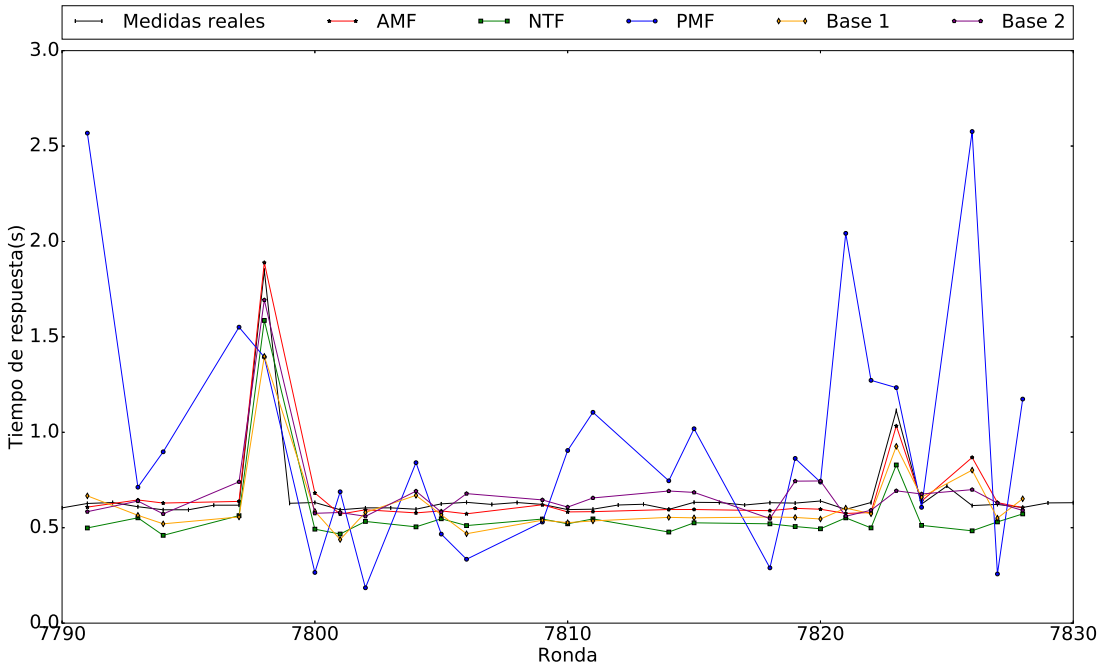


Figura 4.2: Fragmento de los resultados de estimación para cada uno de los algoritmos utilizados en el caso del cliente 2 y el *proxy* 5.

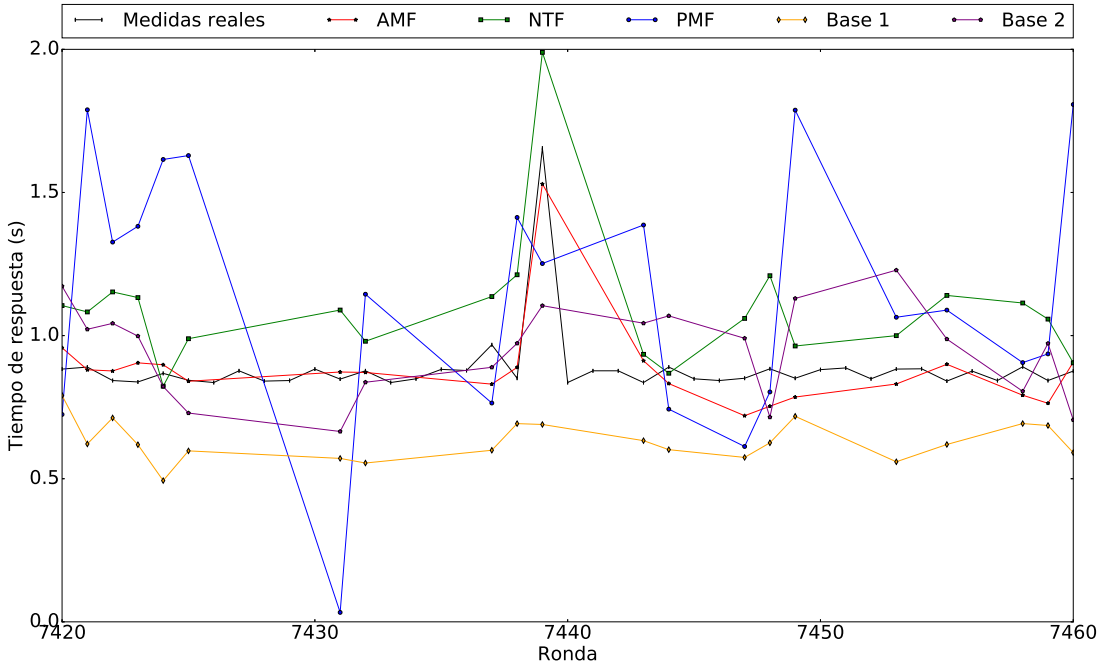


Figura 4.3: Fragmento de los resultados de estimación para cada uno de los algoritmos utilizados en el caso del cliente 4 y el *proxy* 2.

4.4. Conclusiones

Analizados los resultados, se pueden extraer alguna conclusión sobre el comportamiento de los diferentes algoritmos utilizados. En primer lugar, cabe destacar el algoritmo AMF como el que mejores resultados obtiene en todos los casos, seguido por NTF, y PMF como el algoritmo con peores resultados. Cabe destacar, a su vez, que un incremento en el número de *proxies* sondeados por cada clientes mejora el resultado, si bien no en gran medida. La mayor mejora se produce con el algoritmo PMF. Esto se puede deber a que tratamos con densidades de la matriz de entrenamiento demasiado elevadas, de entre el 40 % y el 80 %, cuando lo normal serían densidades inferiores al 20 %.

También destaca el seguimiento de las medidas reales que AMF y NTF hacen en algunos casos como los descritos en este capítulo. Sin embargo, PMF realiza estimaciones con error elevado, que varían mucho más que las medidas reales y que se podrían deber a la falta de simetría en la distribución de los datos. Este efecto podría apreciarse, aunque en menor medida, en el resto de algoritmos, a excepción de AMF, que realiza una transformación de los datos antes y después de la actualización del modelo para acercarlos a la distribución normal y adecuarlos al algoritmo.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

La elección de un *proxy* es un reto importante en las redes comunitarias, puesto que de su rendimiento depende la calidad de servicio percibida por el usuario final. Un aspecto clave para esta elección es la monitorización, aunque el hecho de que cada cliente sondee a cada *proxy*conlleva una sobrecarga elevada que a su vez afecta a la calidad de servicio. En este trabajo se ha abordado una alternativa para aproximar la visibilidad completa a partir de las medidas que los clientes comparten entre ellos. Dicha alternativa consiste en estimar los valores desconocidos de la matriz de visibilidad mediante el uso de algoritmos de factorización matricial.

A partir de los resultados obtenidos, se puede extraer que la factorización es un método eficaz para la estimación del tiempo de respuesta de los *proxies* en las redes de comunidad. Además, de entre los algoritmos estudiados, destaca el comportamiento de AMF debido al seguimiento que hace de los valores medidos, bastante exacto. Esto se podría deber a varios motivos. En primer lugar, la transformación de los datos que se realiza permite que su distribución de probabilidad sea más adecuada para las características del algoritmo. Por otro lado, se trata de un algoritmo capaz de tener en cuenta la evolución temporal de las medidas tomadas.

De entre todos los algoritmos que se han probado, el que podría competir con AMF es NTF. El seguimiento de los valores medidos es también muy bueno, y presenta un valor medio de MAE bajo. Sin embargo, este algoritmo no permite actualizar el modelo de estimación con cada ronda de medidas, sino que requiere factorizar el tensor completo de nuevo. Esto implica que, para una frecuencia de actualización tan alta como la del problema abordado, el uso de este algoritmo implicaría una carga computacional demasiado elevada. Puesto que se pretende que sea el cliente quien realice las estimaciones, en este sentido el algoritmo más adecuado sería AMF.

Además, para la obtención de los datos de partida se ha emulado una red comunitaria en un entorno controlado como es un banco de pruebas. Se ha visto que el uso de este tipo de sistemas supone ventajas importantes, como la posibilidad de alterar la carga de los *proxies* de forma artificial sin afectar a los usuarios de una red real, o la capacidad de utilizar un buen número clientes situados en localizaciones separadas y con características distintas. También permite el acceso a métricas de los *proxies*, como el uso de CPU o el

número de mensajes en cola, a las que sería difícil acceder en una red comunitaria real. Estas medidas no se han utilizado en este trabajo, pero pueden ser útiles para profundizar en las relaciones entre la carga y el tiempo de respuesta del *proxy* [Dim17, Bat18]. Sin embargo, habría que probar la validez de los resultados repitiendo los experimentos con datos obtenidos de una red comunitaria real. En una red comunitaria, la volatilidad de los enlaces influye en las medidas, pero esta característica no se da en el banco de pruebas utilizado. Por otro lado, si bien el número de clientes y de *proxies* utilizado en los experimentos es suficiente para una primera aproximación, sería necesario aumentar el número de clientes y de *proxies* para poder estudiar el comportamiento de los algoritmos cuando la matriz de entrenamiento es menos densa, esto es, cuando el número de *proxies* sondeados por cada cliente es más pequeño frente al número total de *proxies*.

5.2. Trabajo futuro

Respecto al trabajo futuro, los resultados obtenidos mediante la factorización matricial plantean varias vías de trabajo a seguir. En primer lugar, el criterio seguido para la elección de los parámetros de los algoritmos de factorización matricial se basó en ensayo y error. Si bien los resultados son buenos, si se optimizasen estos parámetros clave se podrían llegar a mejorar.

Por otro lado, el número de *proxies* utilizados no permite realizar pruebas con densidades bajas, puesto que el mínimo número razonable de *proxies* que un cliente debería sondear para que tuviese sentido el uso de la factorización matricial es 2, y este número en el conjunto de datos utilizado corresponde a una densidad del 40 %. Por lo tanto, aumentando el número de *proxies* en nuestra red de comunidad emulada se podría verificar el buen funcionamiento de los algoritmos de factorización matricial con densidades de la matriz de entrenamiento menores.

Otra prueba a realizar podría ser introducir de forma artificial carga en los *proxies*, para evaluar cómo afectaría esta variación del tiempo de respuesta en los resultados. En una red comunitaria real es difícil introducir este tipo de carga sin alterar la calidad de servicio que perciben los usuarios. Puesto que la red de la que obtenemos datos es emulada, no hay restricciones de este tipo a la hora de alterar el comportamiento de los *proxies*.

Como se vio en el capítulo 4, podría existir una gran influencia de la transformación de los datos que se realiza en AMF sobre los resultados. Una posible vía de trabajo sería realizar la misma transformación en todos los algoritmos probados, para asegurarnos de que el buen comportamiento de AMF respecto del resto de algoritmos no se debe en exclusiva a esta transformación.

Por último, y puesto que la red de la que se obtienen los datos es emulada, otra vía de investigación podría ser probar el funcionamiento de los algoritmos utilizados en este trabajo para estimar el tiempo de respuesta a partir de datos obtenidos de una red comunitaria real. De esta forma se podría verificar que los resultados son favorables también en el caso real, y no únicamente en un entorno emulado.

Referencias

[Adr93] Rick Adrion. Research methodology in software engineering. En *Summary of the Dagstuhl Workshop on Future Directions in Software Engineering*, volumen 18, páginas 36–37. ACM Software Engineering Nootes, SIG Soft, 1993. 3

[Agg16] Charu C. Aggarwal. *An Introduction to Recommender Systems*, páginas 1–28. Springer International Publishing, Cham, 2016. 8, 10

[Aky05] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: A survey. *Computer Networks*, 47(4):445–487, Marzo 2005. 1

[Bat18] Khulan Batbayar, Emmanouil Dimogerontakis, Roc Meseguer, Leandro Navarro, Esunly Medina, and Rodrigo M. Santos. The RIMO gateway selection approach for mesh networks: Towards a global internet access for all. *Conference on Ubiquitous Computing and Intelligence (UCAmI)*, 2(1258), 2018. 2, 5, 6, 15, 34

[Bat19] Khulan Batbayar, Roc Meseguer, Emmanouil Dimogerontakis, Leandro Navarro, and Ramin Sandre. Collaborative informed gateway selection in large-scale and heterogeneous networks. En *Symposium on Integrated Network Management (IM)*, páginas 337–345, 2019. 7

[Bor18] Diego Bores Quijano. Contribuciones a la selección distribuida de proxy en el acceso a la web a través de redes de comunidad inalámbricas. Trabajo Fin de Grado, Universidad de Valladolid, Valladolid, España. 2018. 3, 12, 15

[Bot18] Miguel L. Bote Lorenzo, Eduardo Gómez Sánchez, Carlos Mediavilla Pastor, and Juan I. Asensio Pérez. Online machine learning algorithms to predict link quality in community wireless mesh networks. *Computer Networks*, 132:68–80, 2018. 1, 2

[Bra13] Bart Braem, Chris Blondia, Christoph Barz, Henning Rogge, Felix Freitag, Leandro Navarro, Joseph Bonicioli, Stavros Papathanasiou, Pau Escrich, Roger Baig Vi nas, Aaron L. Kaplan, Axel Neumann, Ivan Vilata i Balaguer, Blaine Tatum, and Malcolm Matson. A case for research with and on community networks. *SIGCOMM Comput. Commun. Rev.*, 43(3):68–73, Julio 2013. 1

[Chu03] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, Julio 2003. 15

[Dab04] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. *SIGCOMM Computer Communication Review*, 34(4):15–26, Agosto 2004. 6

[Dim17] Emmanouil Dimogerontakis, João Neto, Roc Meseguer, Leandro Navarro, and Luís Vega. Client-side routing-agnostic gateway selection for heterogeneous wireless mesh networks. En *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Mayo 2017. 1, 2, 6, 15, 16, 34

[Han11] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edición, 2011. 9

[Her04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Enero 2004. 25

[Kau14] Jasleen Kaur, Qianwen Yin, and F. Donelson Smith. Can bandwidth estimation tackle noise at ultra-high speeds? páginas 107–118, 2014. 6

[Ko13] Bong-Jun Ko, Sisi Liu, Murtaza Zafer, Ho Yin Starsky Wong, and Kang-Won Lee. Gateway selection in hybrid wireless networks through cooperative probing. En *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, páginas 352–360, Mayo 2013. 7

[Kor09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Agosto 2009. 2, 8, 9

[Kri09] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize CDN performance. En *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, páginas 190–201, New York, NY, USA, 2009. ACM. 5

[Led08] Jonathan Ledlie, Nokia Research, Margo Seltzer, and Peter Pietzuch. Proxy network coordinates. 22:25, 2008. 6

[Lo12] Wei Lo, Jianwei Yin, Shuiguang Deng, Ying Li, and Zhaohui Wu. An extended matrix factorization approach for QoS prediction in service selection. En *2012 IEEE Ninth International Conference on Services Computing*, páginas 162–169, 2012. 2, 7, 12, 23, 25

[Lop11] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, páginas 73–105. Springer US, Boston, MA, Junio 2011. 8

[Lum09] Cristian Lumezanu, Randy Baden, Neil Spring, and Bobby Bhattacharjee. Triangle inequality variations in the internet. En *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, páginas 177–183, New York, NY, USA, 2009. ACM. 6

[Luo16] Xiong Luo, Ji Liu, Dandan Zhang, and Xiaohui Chang. A large-scale web QoS prediction scheme for the industrial Internet of things based on a kernel machine learning algorithm. *Computer Networks*, 101:81 – 89, 2016. Industrial Technologies and Applications for the Internet of Things. 7, 11

[Mac13] Leonardo Maccari. An analysis of the Ninux wireless community network. En *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, páginas 1–7, Octubre 2013. 1

[Mil15] Pere Millan, Carlos Molina, Esunly Medina, Davide Vega, Roc Meseguer, Bart Braem, and Chris Blondia. Time series analysis to predict link quality of wireless community networks. *Computer Networks*, 93(P2):342–358, Diciembre 2015. 1

[Neu16] Axel Neumann, Ester Lopez, Llorenc Cerda-Alabern, and Leandro Navarro. Securely-entrusted multi-topology routing for community networks. En *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, páginas 1–8, Jan 2016. 1

[Nie16] Laisen Nie, Dingde Jiang, Lei Guo, and Shui Yu. Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks. *Journal of Network and Computer Applications*, 76:16 – 22, 2016. 7

[Par14] European Parliament and Council. Directive 2014/61/eu of the European parliament and the council of 15 may 2014 on measures to reduce the cost of deploying high-speed electronic communication networks. Informe técnico 2017/61/eu, Mayo 2014. 1

[Res97] Paul Resnick and Hal R. Varian. Recommender systems. *Magazine Communications of the ACM*, 40(3):56–58, Marzo 1997. 8

[RM13] Carlos Rey-Moreno, Zukile Roro, William D. Tucker, Masbulele Jay Siya, Nicola J. Bidwell, and Javier Simo-Reigadas. Experiences, challenges and lessons from rolling out a rural WiFi mesh network. En *Proceedings of the 3rd ACM Symposium on Computing for Development*, ACM DEV '13, páginas 11:1–11:10, New York, NY, USA, 2013. ACM. 1

[Ric15] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*. Springer Publishing Company, Incorporated, 2nd edición, 2015. 7, 8

[Sal07] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. En *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, páginas 1257–1264, USA, 2007. Curran Associates Inc. 11, 21

[Sal18] Stefano Salsano, Fabio Patriarca, Francesco Lo Presti, Pier Luigi Ventre, and Valerio Maria Gentile. Accurate and efficient measurements of IP level performance to drive interface selection in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 17(10):2223–2235, Oct 2018. 6

[Su09] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4:2–4:2, Enero 2009. 7, 8

[Tak09] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, Junio 2009. 8, 12

[Uni09] International Telecommunication Union. Trends in telecommunication reform 2008: Six degrees of sharing. Informe técnico D-PREF-TTR.10, Julio 2009. 1

[Vad16] G. Vadivelou. Collaborative filtering based web service recommender system using users' satisfaction on QoS attributes. En *2016 International Conference on Inventive Computation Technologies (ICICT)*, volumen 3, páginas 1–5, Agosto 2016. 7, 11

[Veg15] Davide Vega, Roger Baig, Llorenç Cerdà-Alabern, Esunly Medina, Roc Meseguer, and Leandro Navarro. A technological overview of the guifi.net community network. *Computer Networks*, 93:260 – 278, 2015. Community Networks. 1

[Zha14] Wancai Zhang, Hailong Sun, Xudong Liu, and Xiaohui Guo. Temporal QoS-aware web service recommendation via non-negative tensor factorization. En *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, páginas 585–596, New York, NY, USA, 2014. ACM. 11, 21

[Zhe13] Zbin Zheng, Hao Ma, Michael R. Lyu, and Irwin King. Collaborative web service QoS prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, 6(3):289–299, Julio 2013. 2, 7, 12, 23, 25

[Zhe17] Xianrong Zheng, Li Da Xu, and Sheng Chai. Qos recommendation in cloud services. *IEEE Access*, 5:5171–5177, 2017. 7, 11

[Zhu17] Jieming Zhu, Pinjia He, Zibin Zheng, and Michael R. Lyu. Online QoS prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):2911–2924, Octubre 2017. 2, 11, 15, 20, 21, 22, 23, 24, 25

Apéndice A

Evolución de tiempo de respuesta para cada pareja cliente-*proxy*

En la el capítulo 3 se mostraba la evolución temporal del tiempo de respuesta en alguna región temporal ampliada para tres parejas cliente-*proxy*. En este capítulo se muestra la evolución de esta métrica para cada pareja cliente-*proxy*.

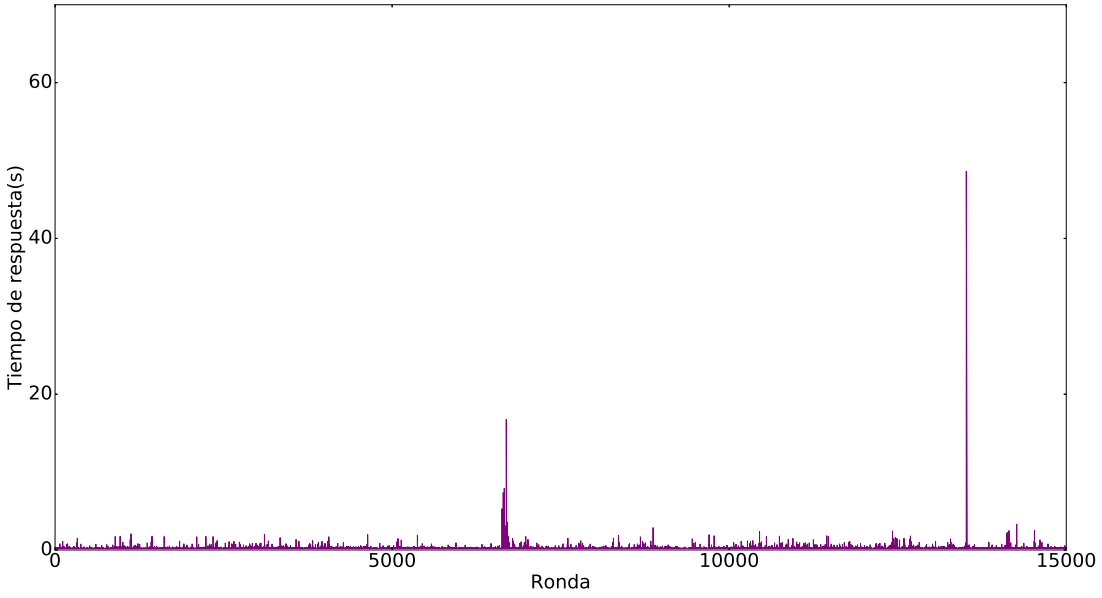


Figura A.1: Evolución del tiempo de respuesta medido para la pareja cliente 1 *proxy* 1.

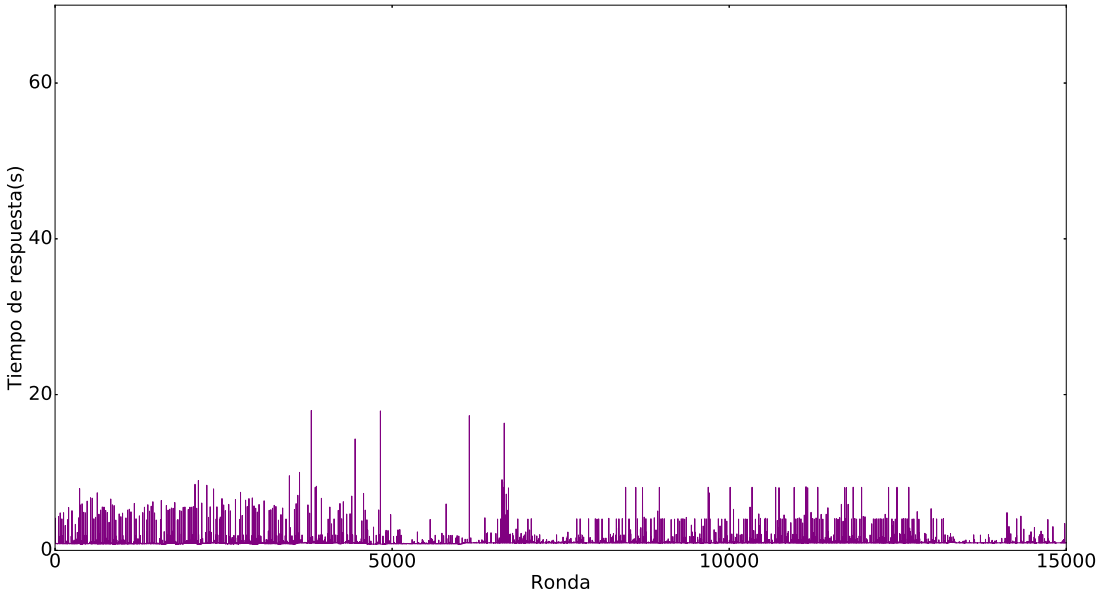


Figura A.2: Evolución del tiempo de respuesta medido para la pareja cliente 1 *proxy* 2.

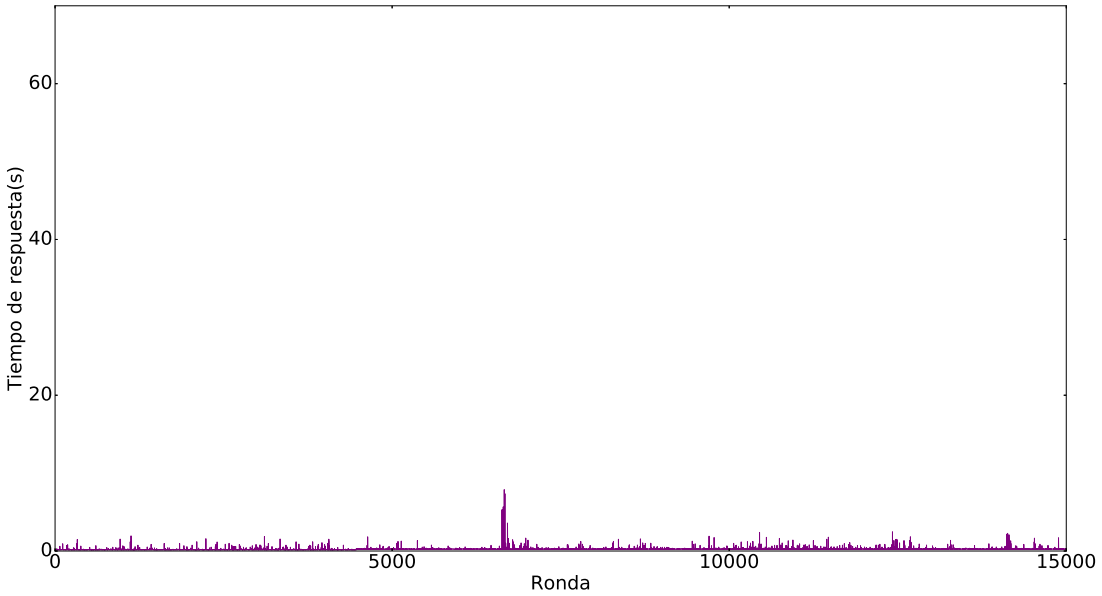


Figura A.3: Evolución del tiempo de respuesta medido para la pareja cliente 1 *proxy* 3.

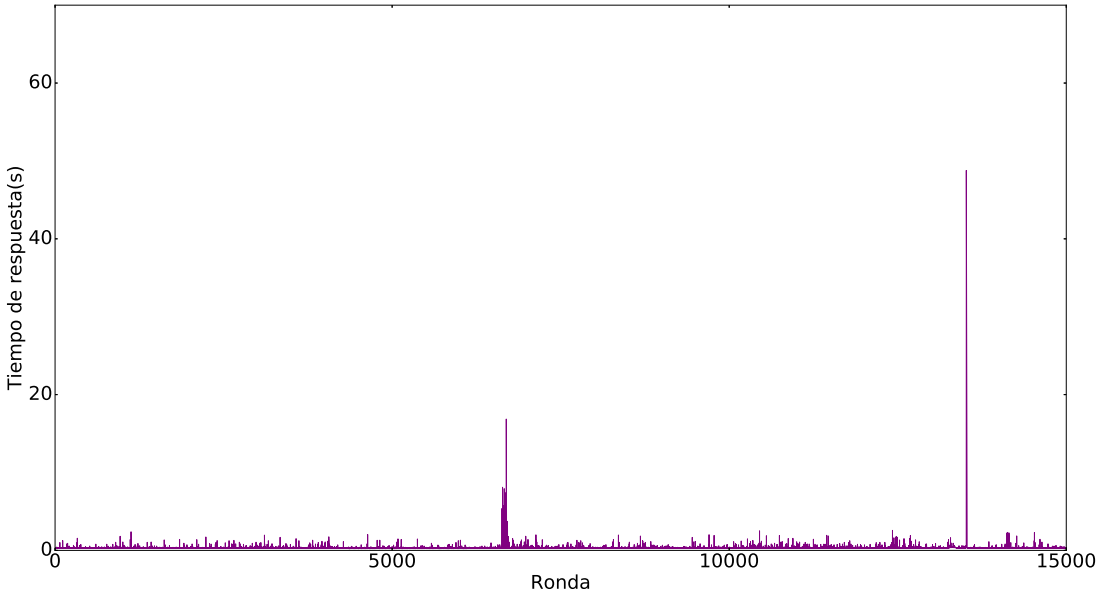


Figura A.4: Evolución del tiempo de respuesta medido para la pareja cliente 1 *proxy* 4.

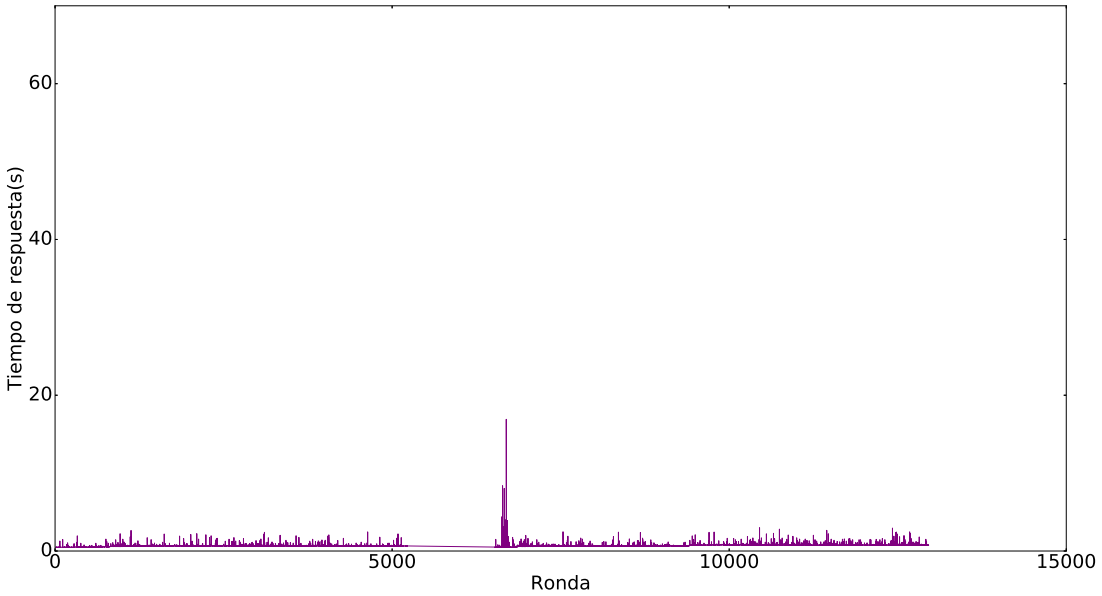


Figura A.5: Evolución del tiempo de respuesta medido para la pareja cliente 1 *proxy* 5.

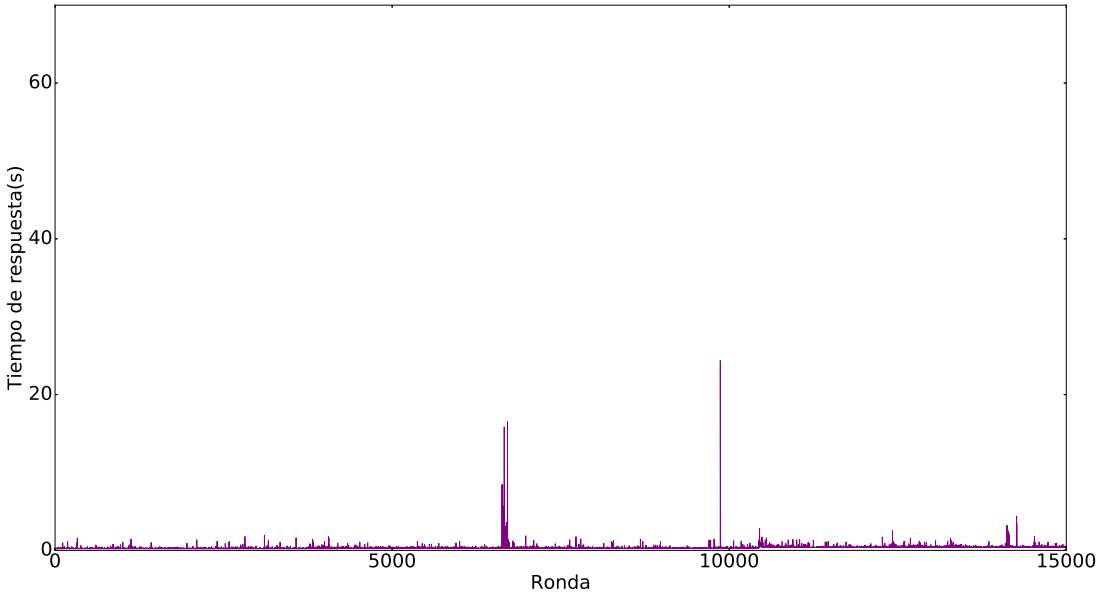


Figura A.6: Evolución del tiempo de respuesta medido para la pareja cliente 2 *proxy* 1.

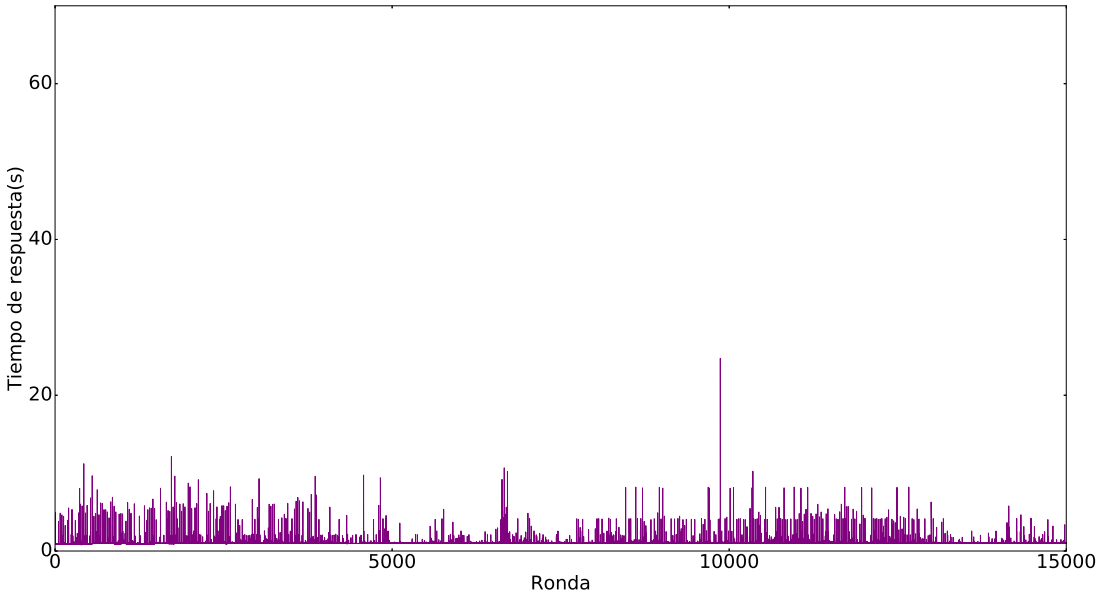


Figura A.7: Evolución del tiempo de respuesta medido para la pareja cliente 2 *proxy* 2.

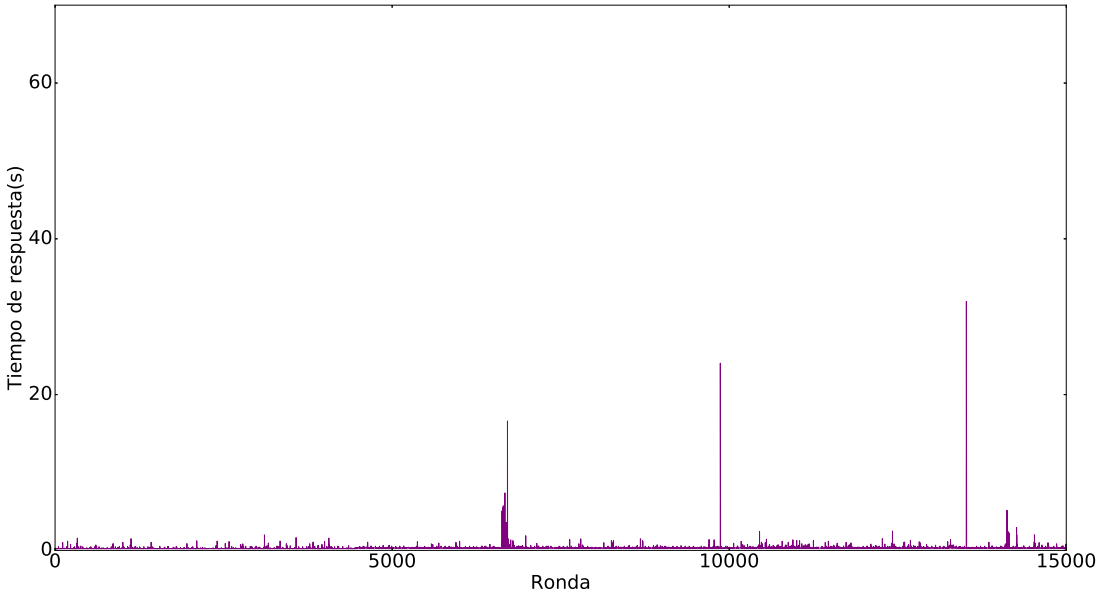


Figura A.8: Evolución del tiempo de respuesta medido para la pareja cliente 2 *proxy* 3.

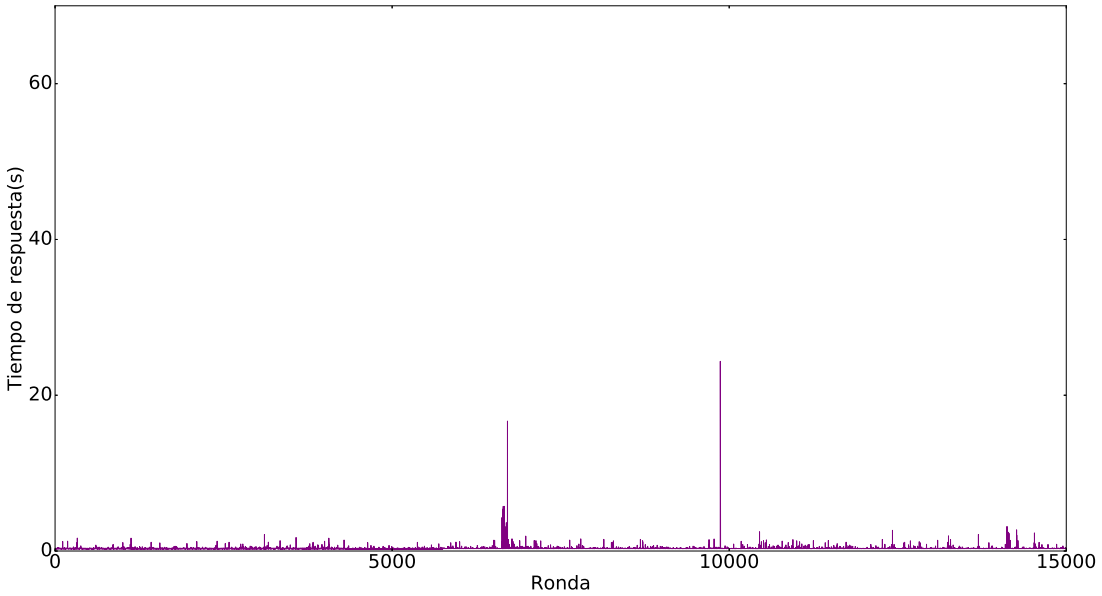


Figura A.9: Evolución del tiempo de respuesta medido para la pareja cliente 2 *proxy* 4.

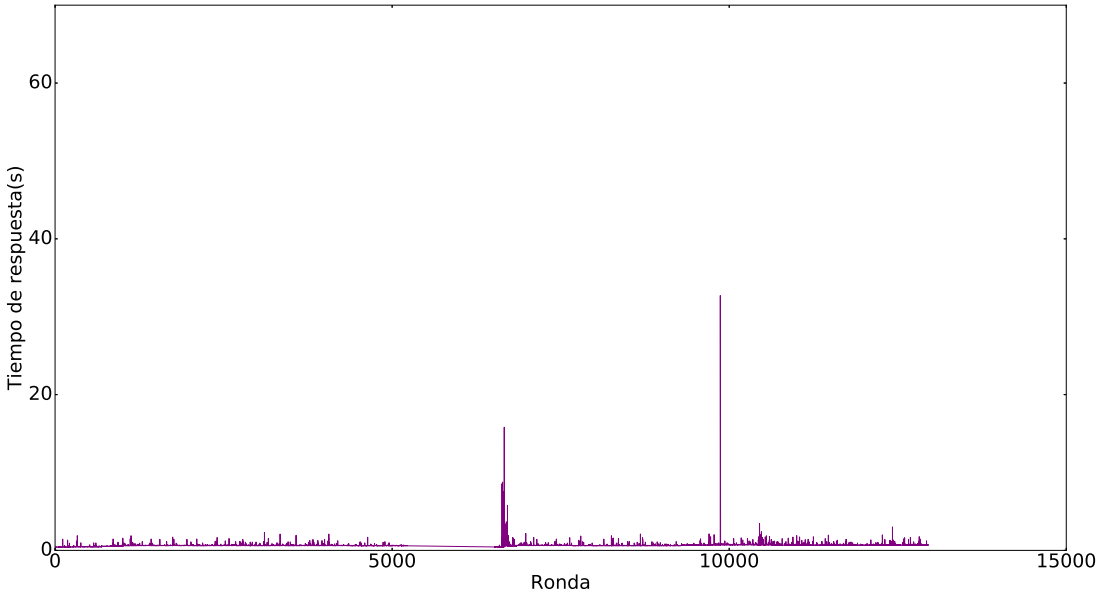


Figura A.10: Evolución del tiempo de respuesta medido para la pareja cliente 2 *proxy* 5.

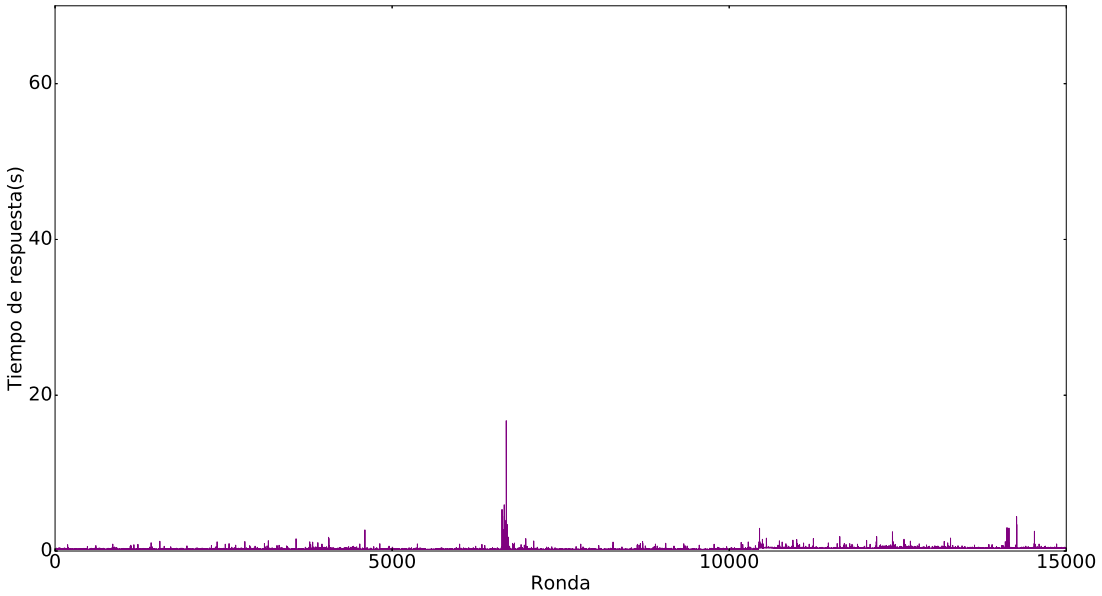


Figura A.11: Evolución del tiempo de respuesta medido para la pareja cliente 3 *proxy* 1.

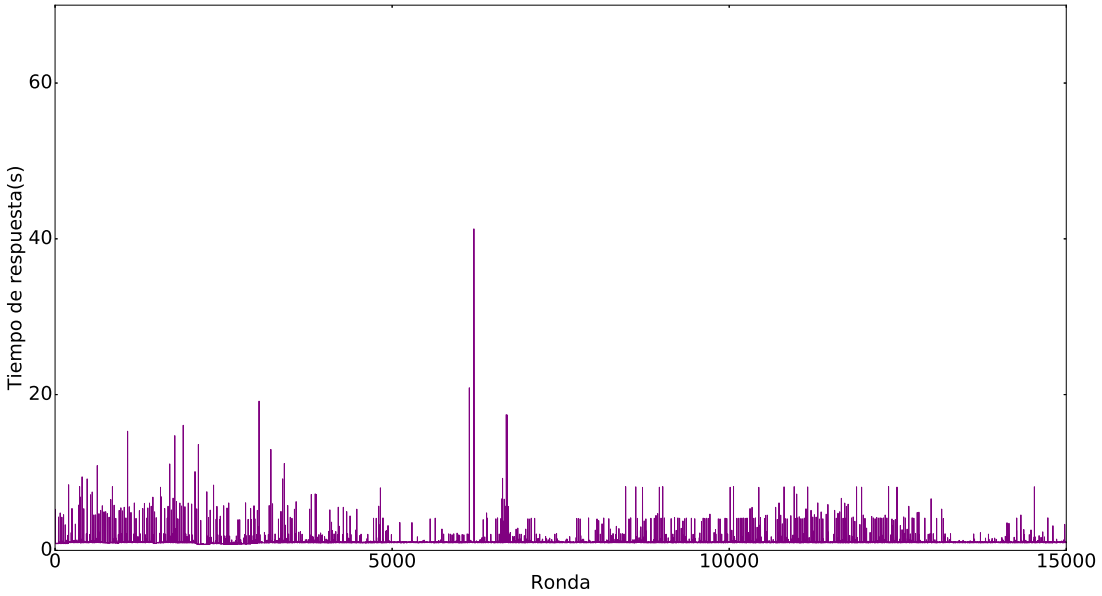


Figura A.12: Evolución del tiempo de respuesta medido para la pareja cliente 3 *proxy* 2.

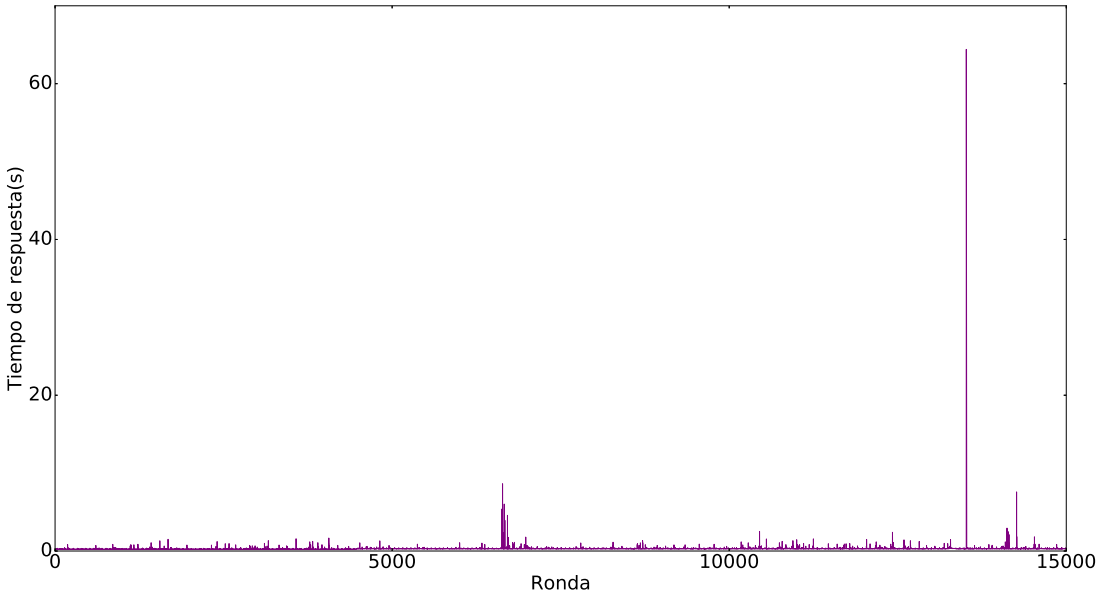


Figura A.13: Evolución del tiempo de respuesta medido para la pareja cliente 3 *proxy* 3.

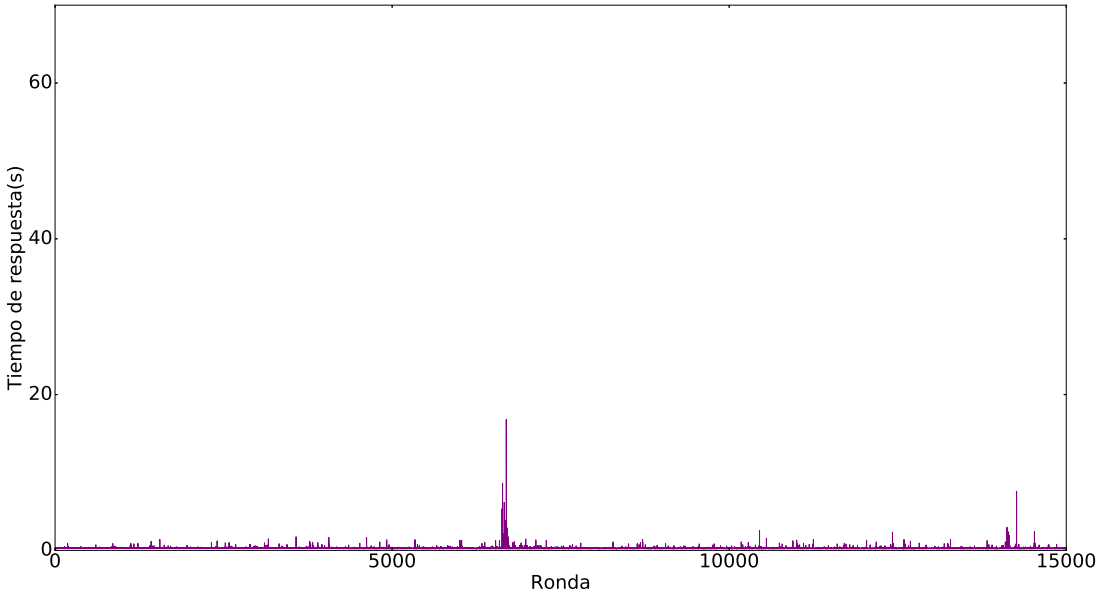


Figura A.14: Evolución del tiempo de respuesta medido para la pareja cliente 3 *proxy* 4.

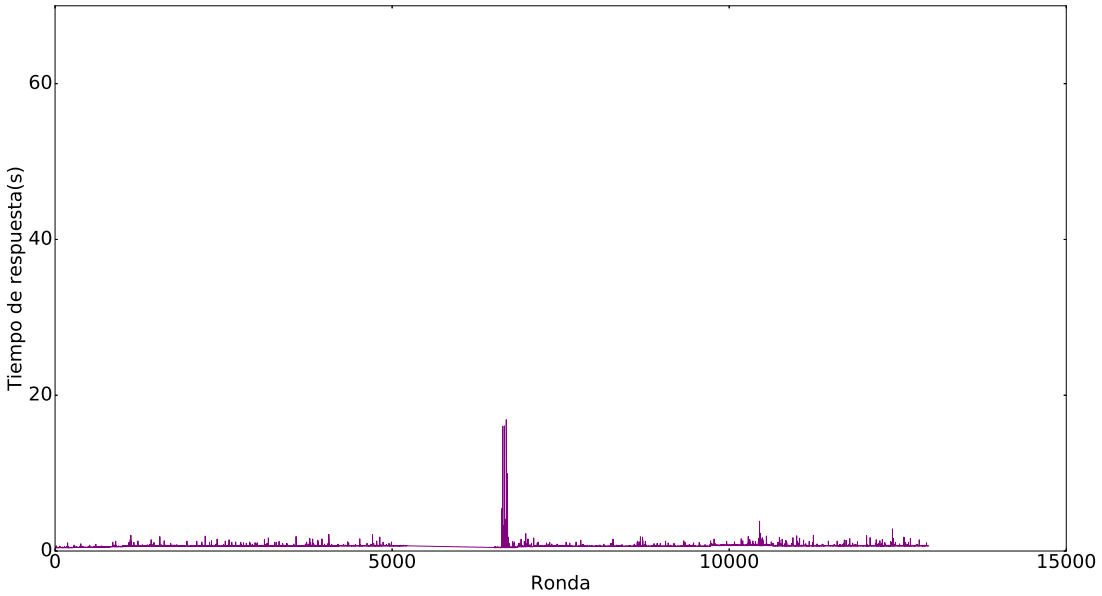


Figura A.15: Evolución del tiempo de respuesta medido para la pareja cliente 3 *proxy* 5.

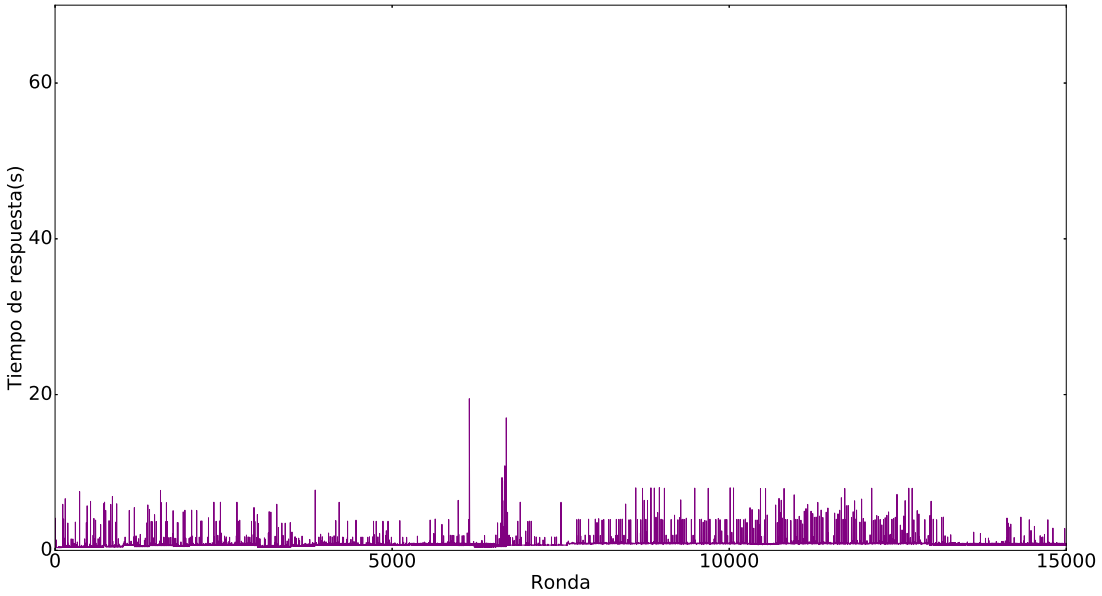


Figura A.16: Evolución del tiempo de respuesta medido para la pareja cliente 4 *proxy* 1.

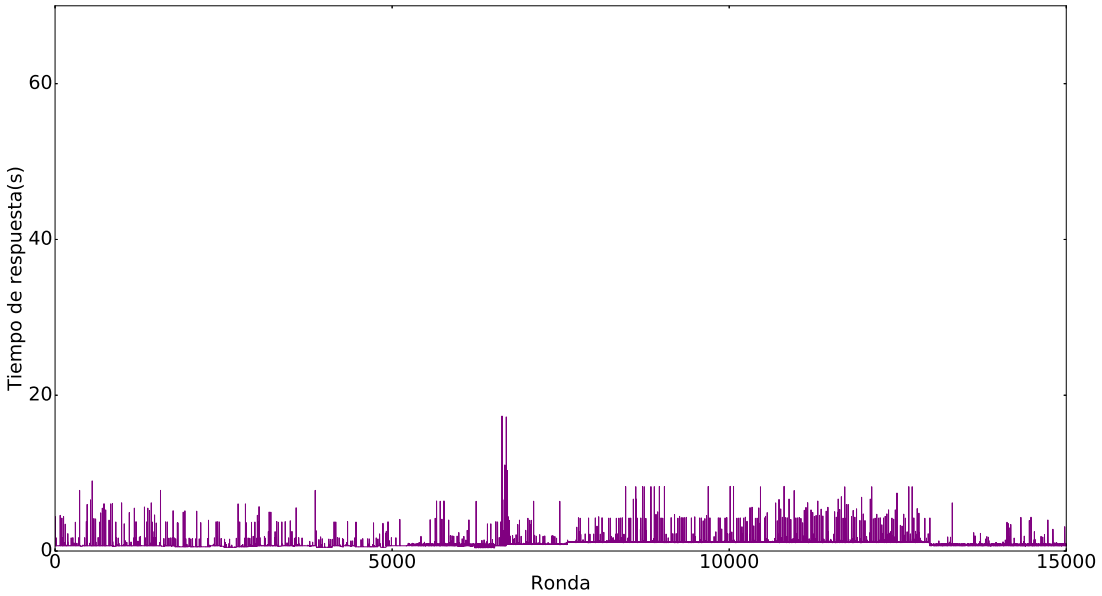


Figura A.17: Evolución del tiempo de respuesta medido para la pareja cliente 4 *proxy* 2.

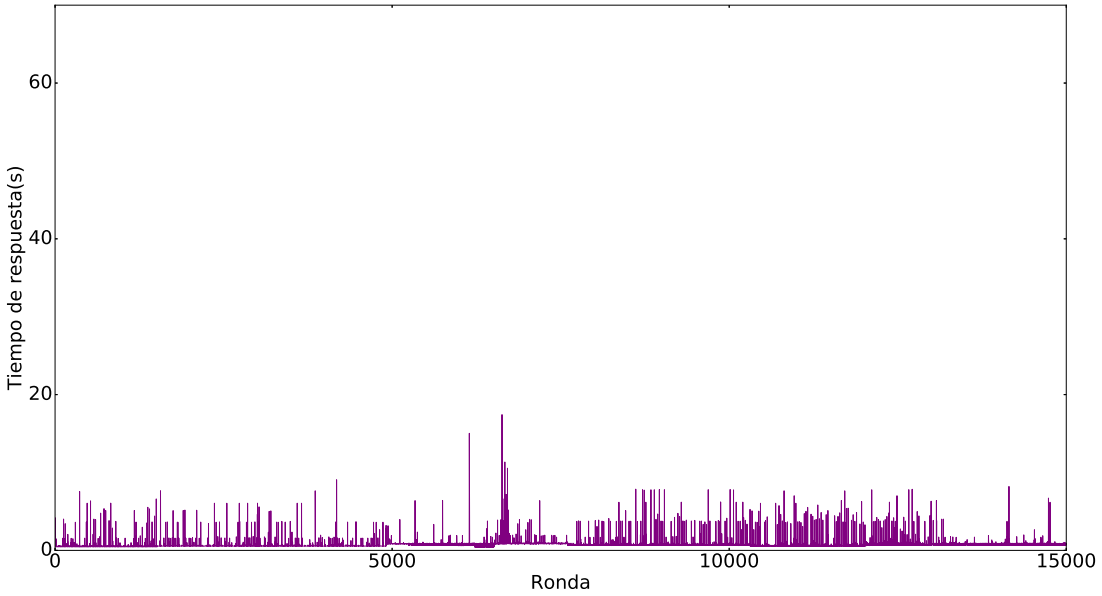


Figura A.18: Evolución del tiempo de respuesta medido para la pareja cliente 4 *proxy* 3.

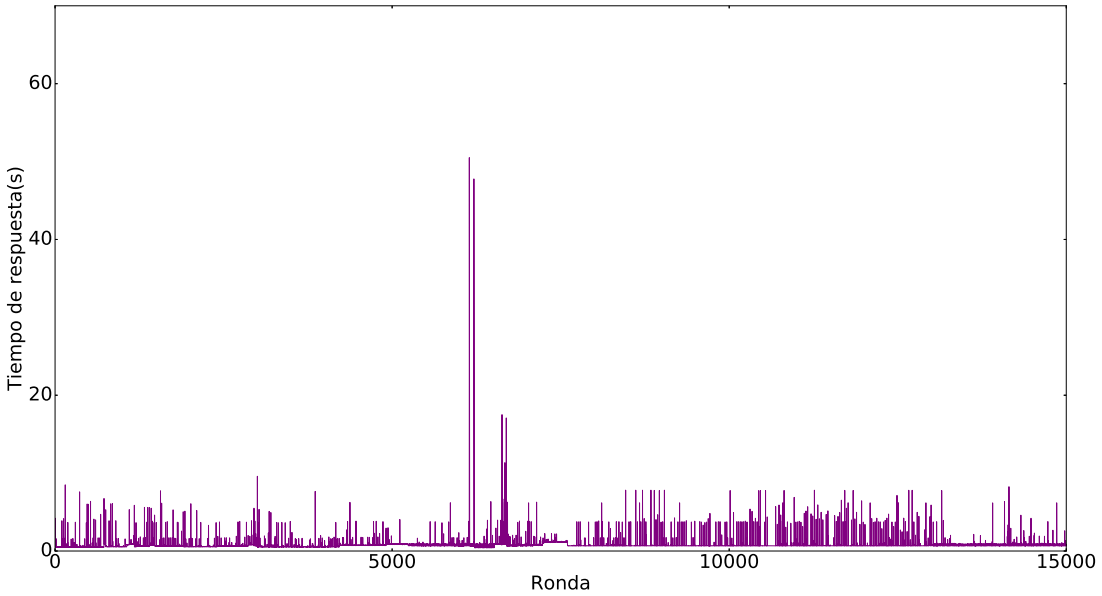


Figura A.19: Evolución del tiempo de respuesta medido para la pareja cliente 4 *proxy* 4.

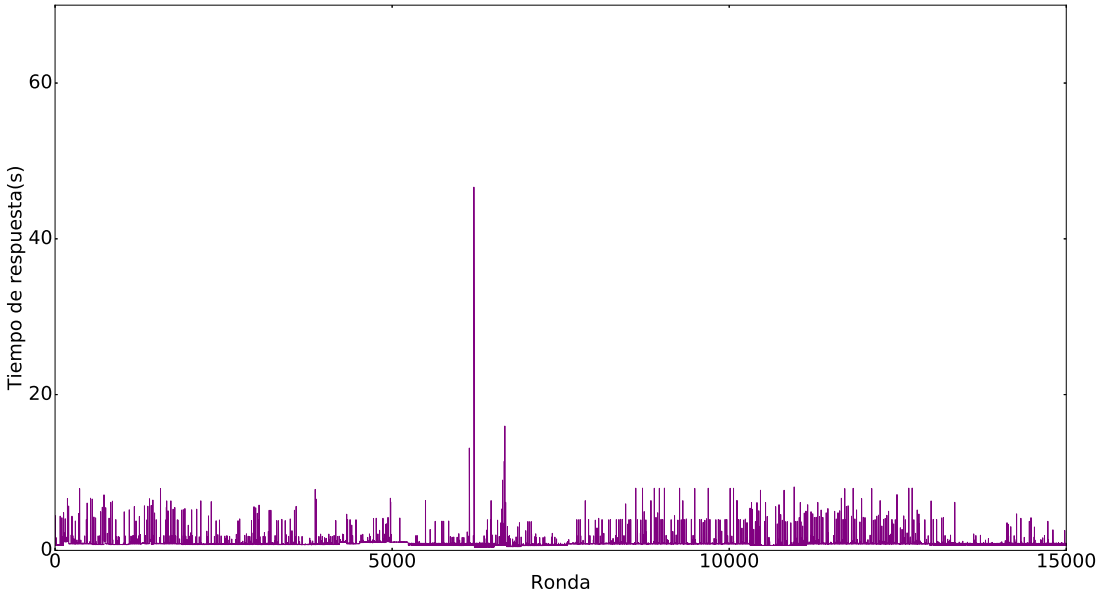


Figura A.20: Evolución del tiempo de respuesta medido para la pareja cliente 4 *proxy* 5.

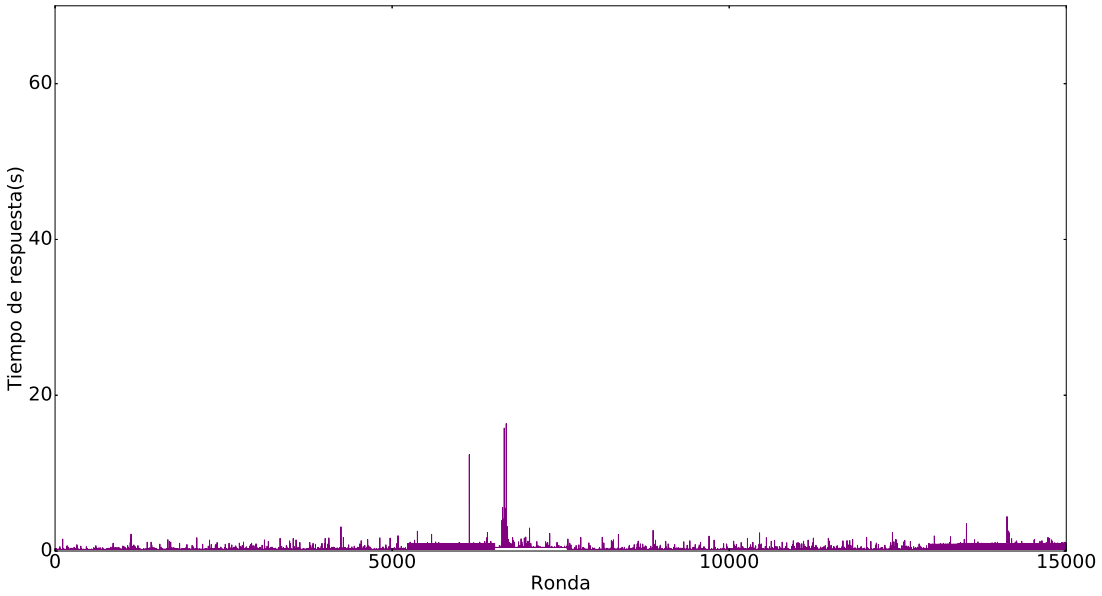


Figura A.21: Evolución del tiempo de respuesta medido para la pareja cliente 5 *proxy* 1.

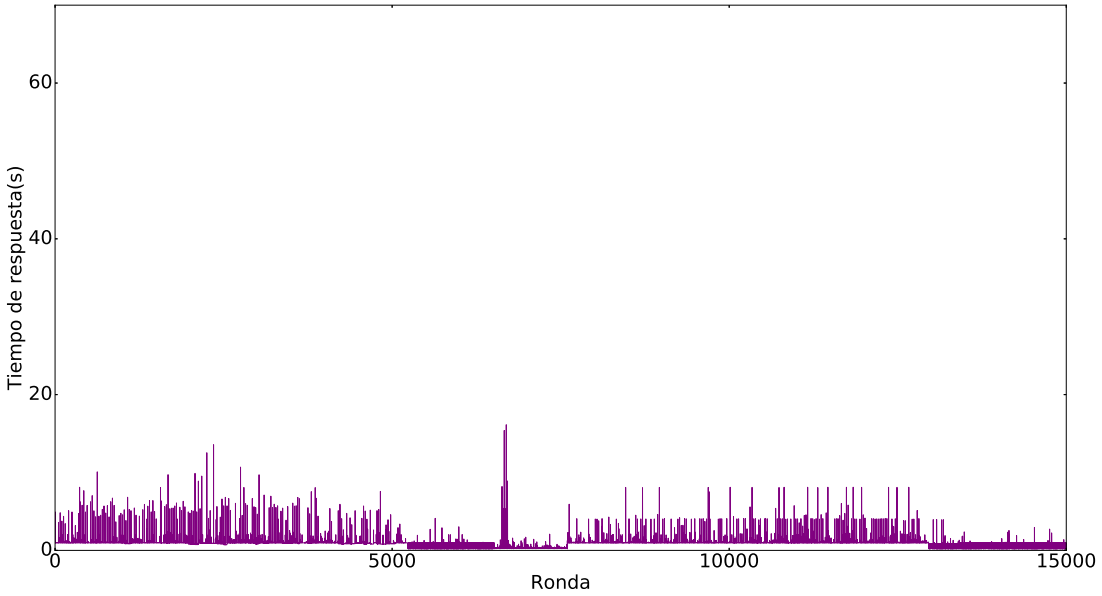


Figura A.22: Evolución del tiempo de respuesta medido para la pareja cliente 5 *proxy* 2.

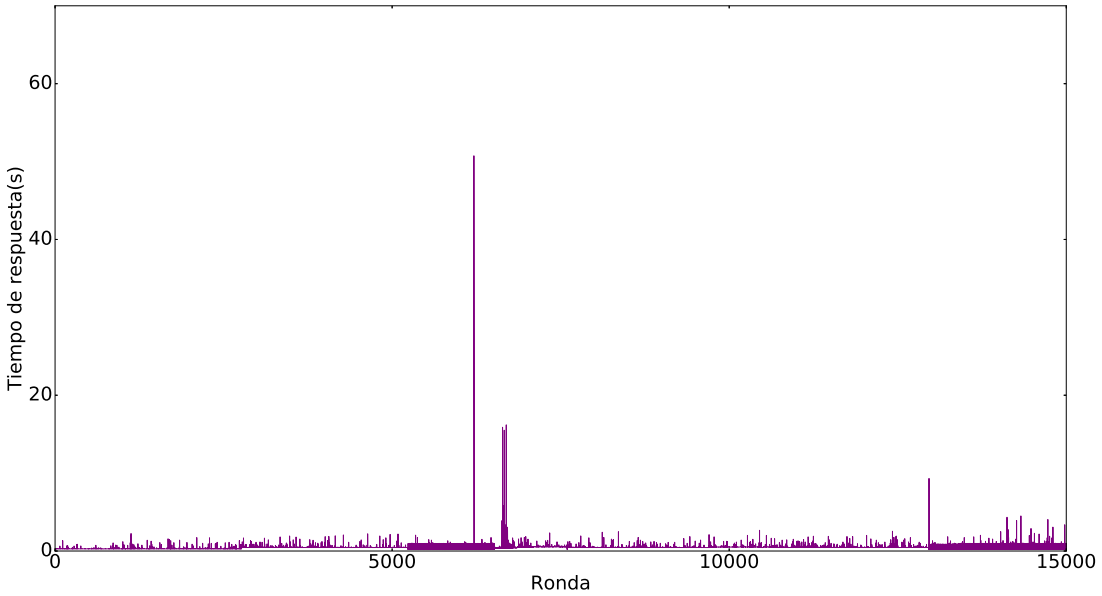


Figura A.23: Evolución del tiempo de respuesta medido para la pareja cliente 5 *proxy* 3.

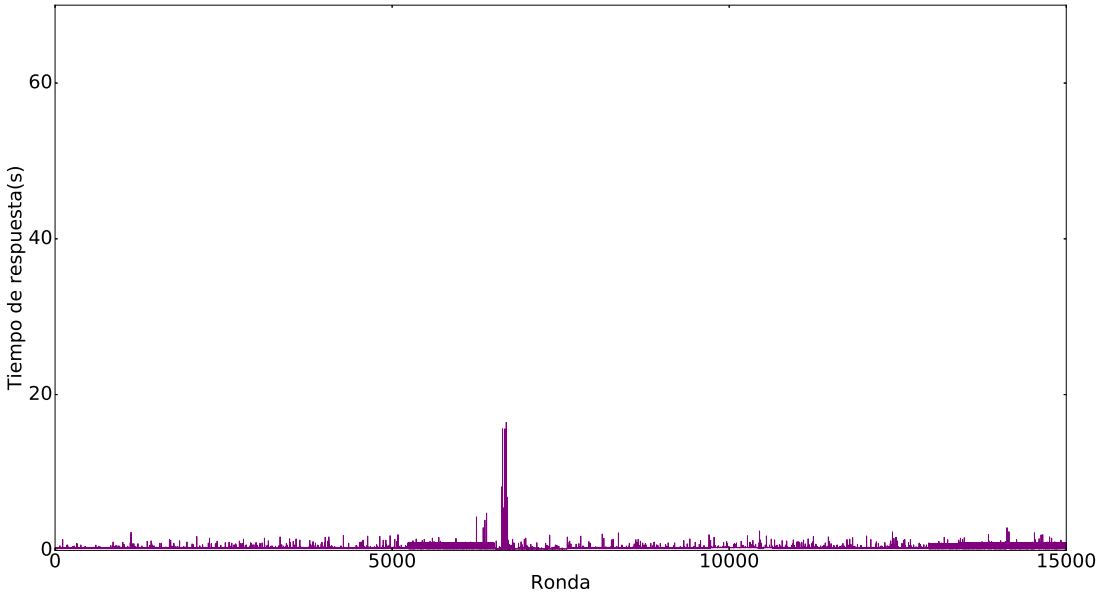


Figura A.24: Evolución del tiempo de respuesta medido para la pareja cliente 5 *proxy* 4.

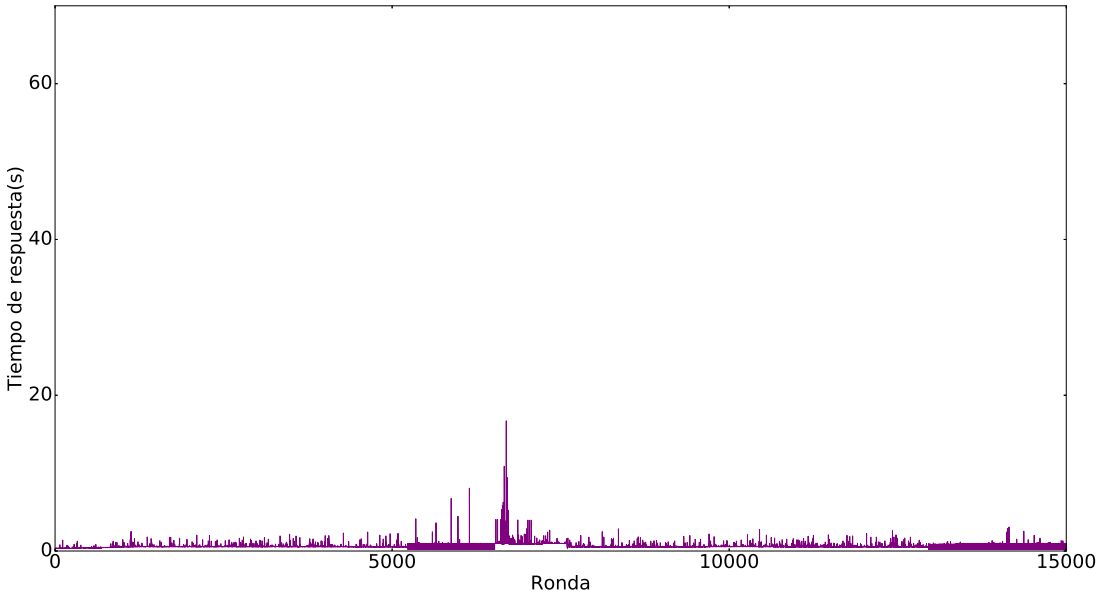


Figura A.25: Evolución del tiempo de respuesta medido para la pareja cliente 5 *proxy* 5.

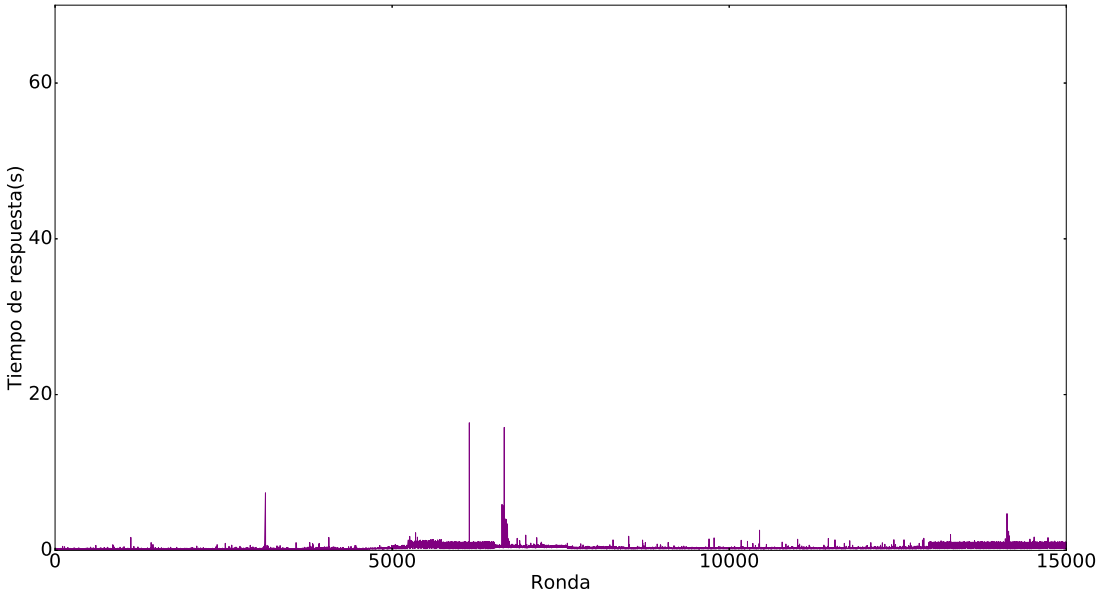


Figura A.26: Evolución del tiempo de respuesta medido para la pareja cliente 6 *proxy* 1.

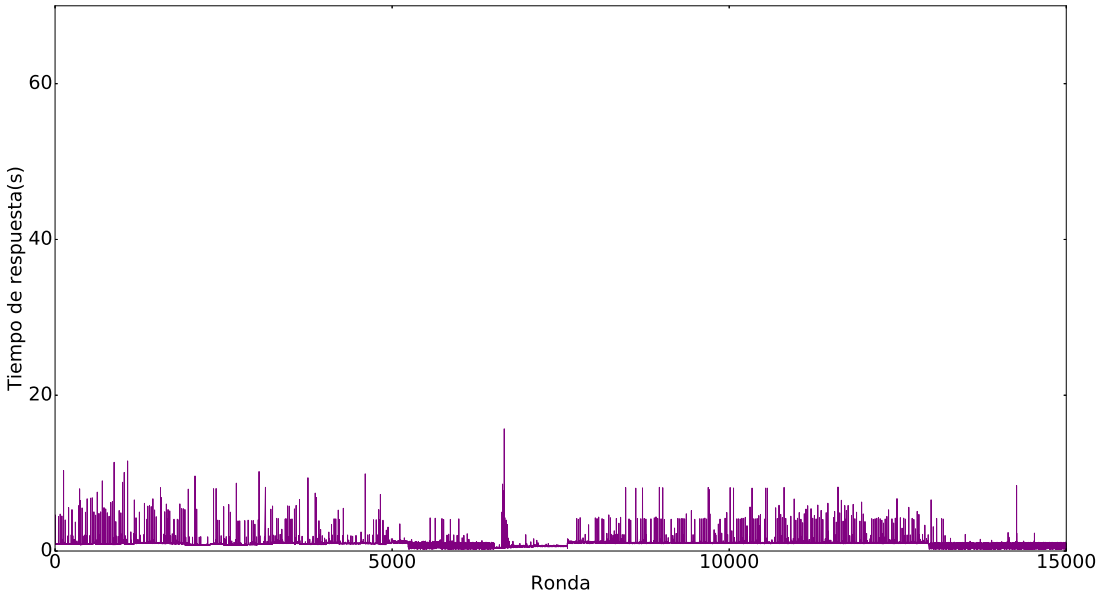


Figura A.27: Evolución del tiempo de respuesta medido para la pareja cliente 6 *proxy* 2.

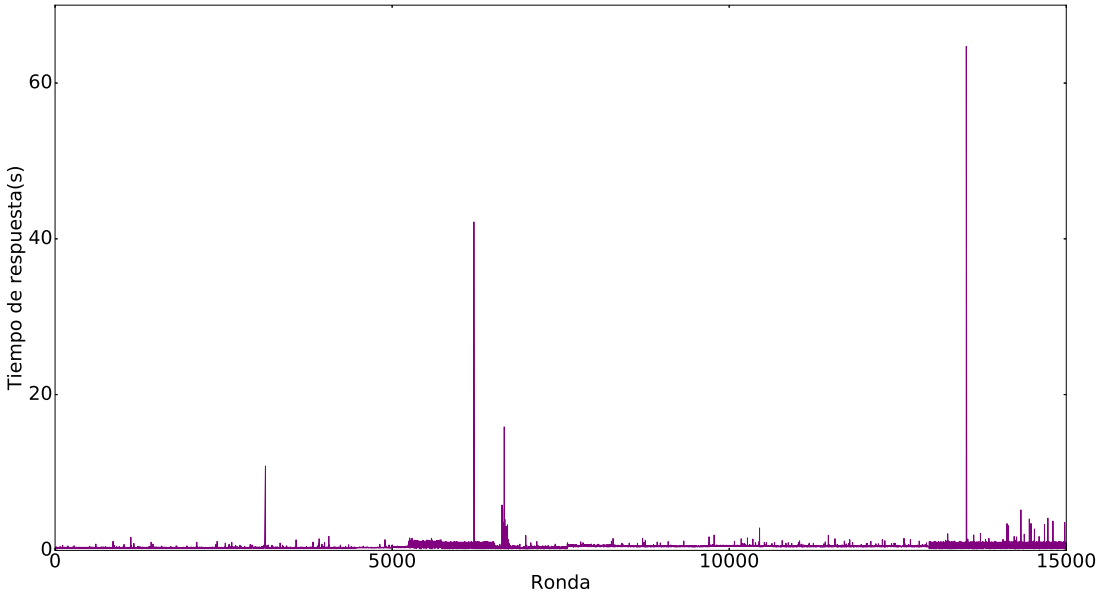


Figura A.28: Evolución del tiempo de respuesta medido para la pareja cliente 6 *proxy* 3.

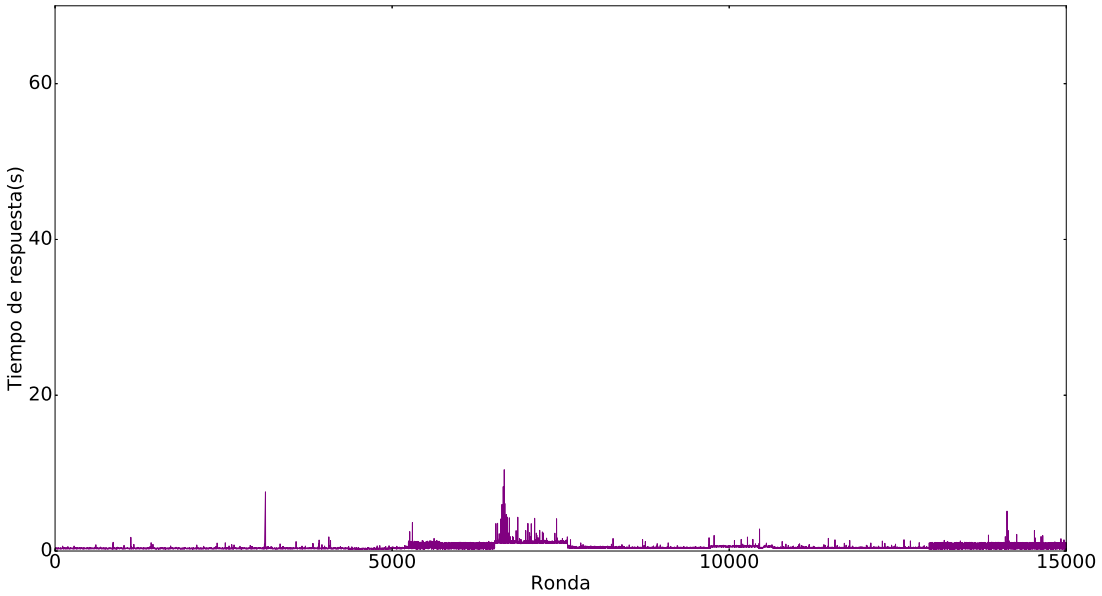


Figura A.29: Evolución del tiempo de respuesta medido para la pareja cliente 6 *proxy* 4.

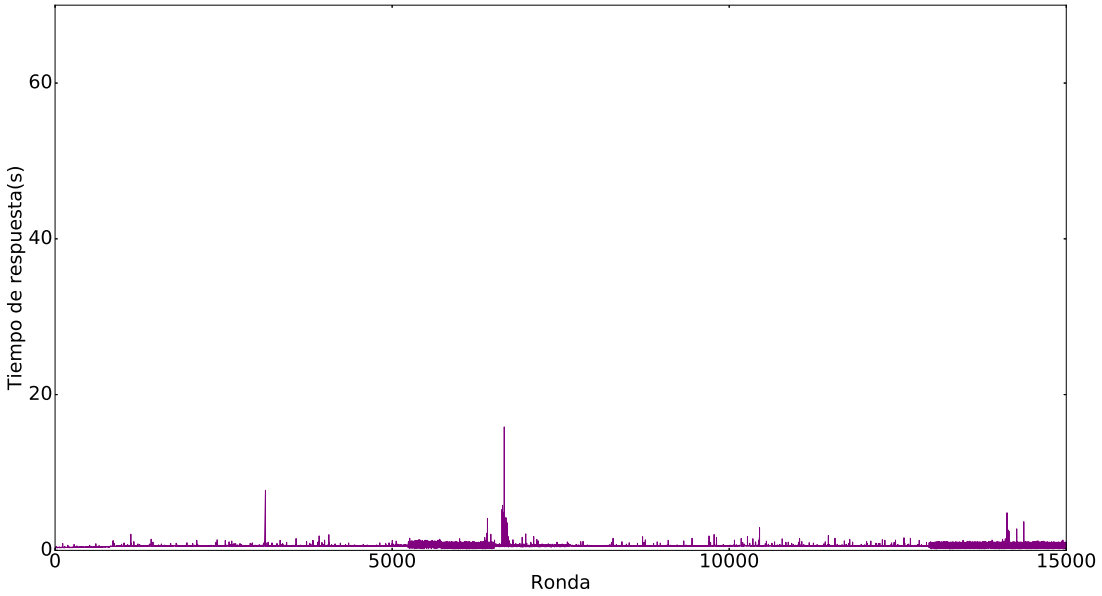


Figura A.30: Evolución del tiempo de respuesta medido para la pareja cliente 6 *proxy* 5.

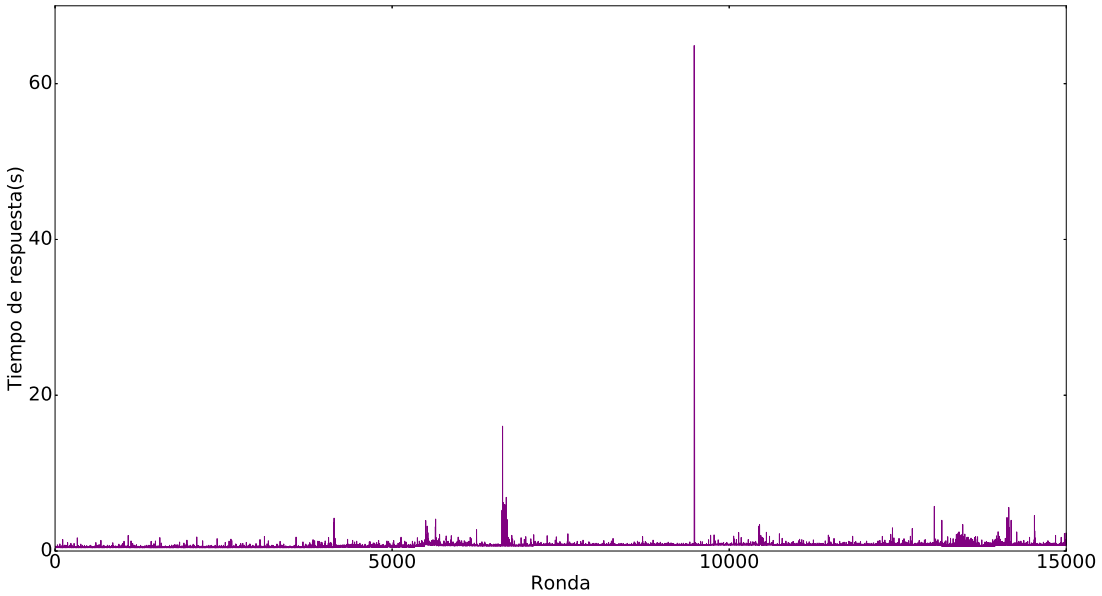


Figura A.31: Evolución del tiempo de respuesta medido para la pareja cliente 7 *proxy* 1.

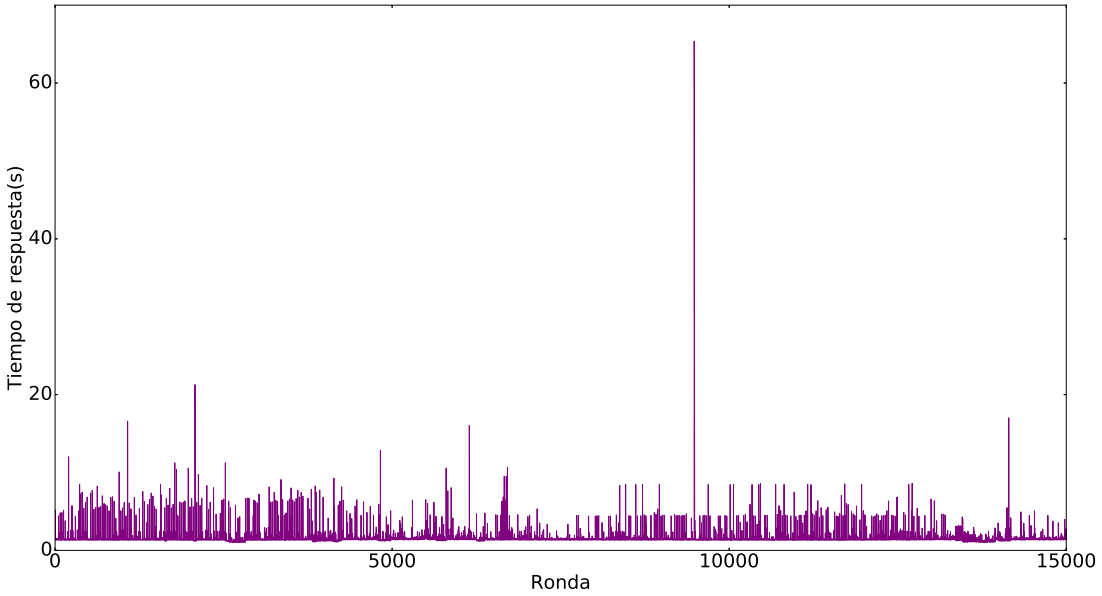


Figura A.32: Evolución del tiempo de respuesta medido para la pareja cliente 7 *proxy* 2.

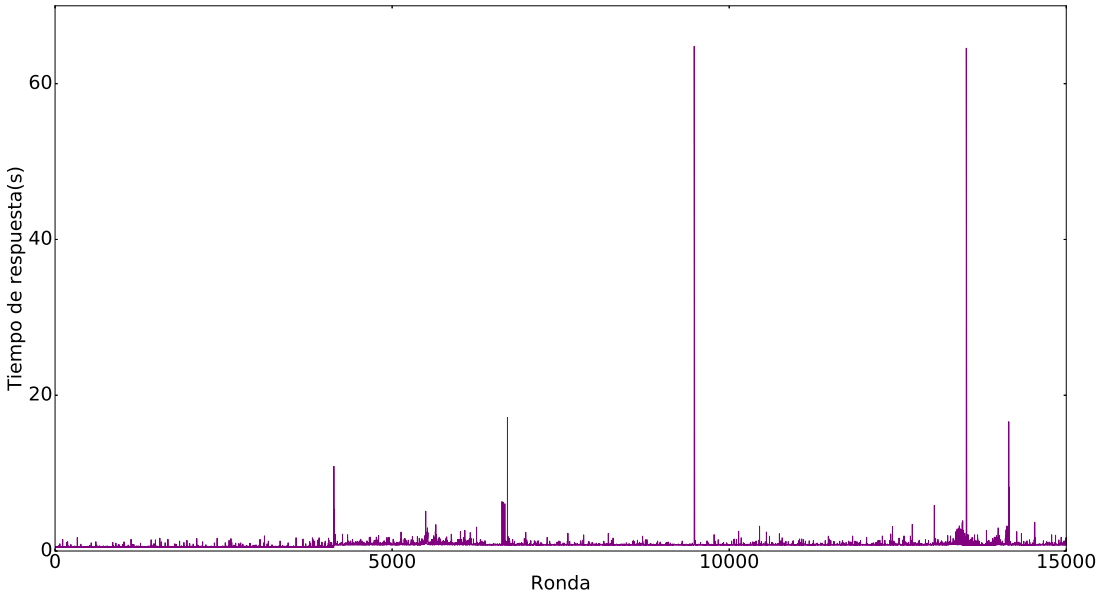


Figura A.33: Evolución del tiempo de respuesta medido para la pareja cliente 7 *proxy* 3.

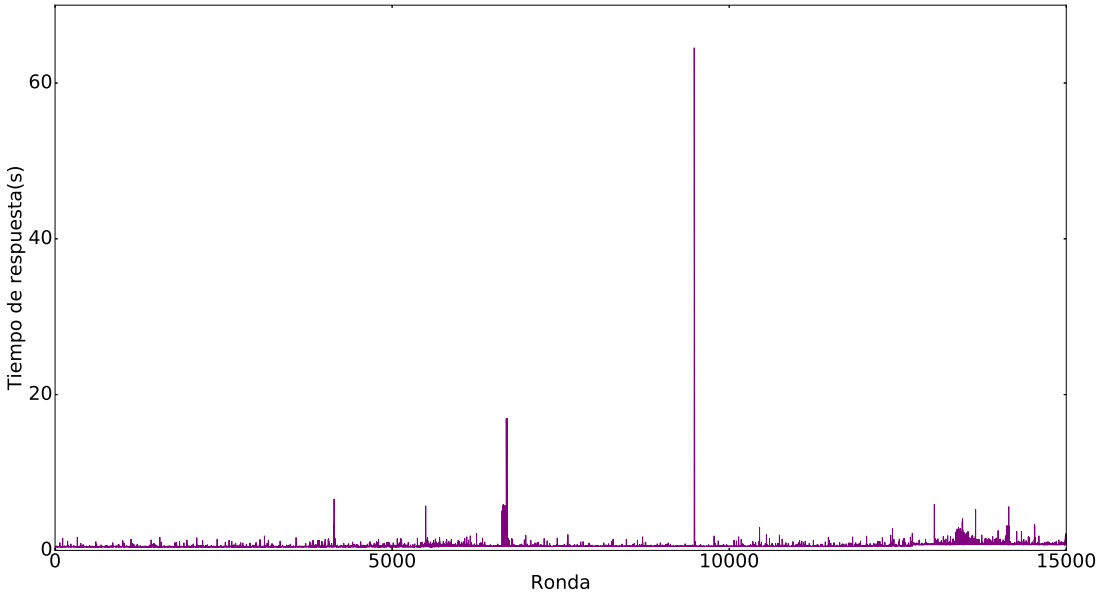


Figura A.34: Evolución del tiempo de respuesta medido para la pareja cliente 7 *proxy* 4.

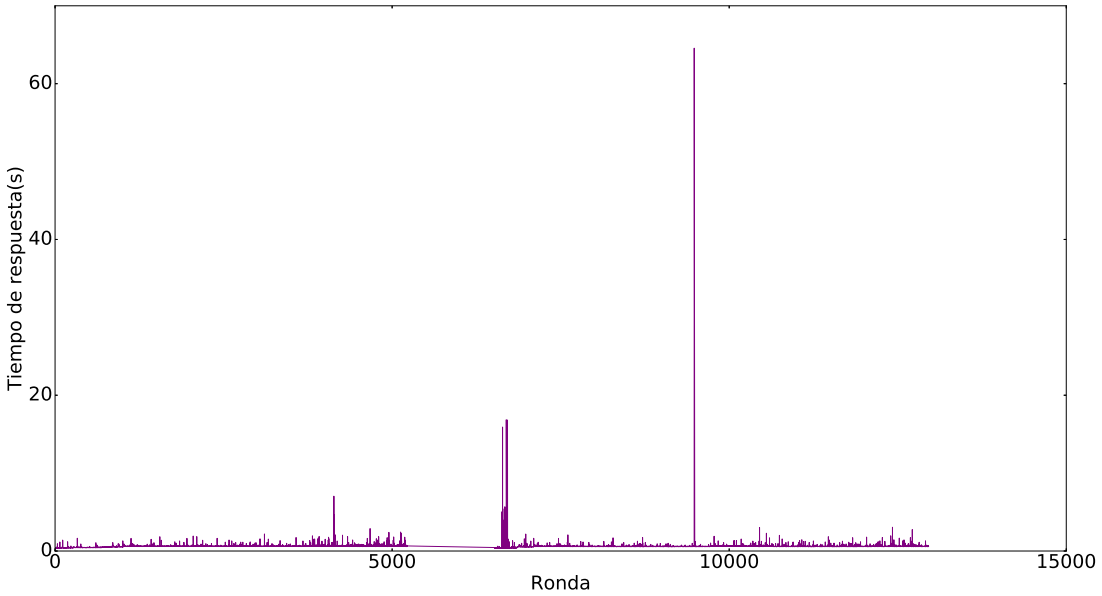


Figura A.35: Evolución del tiempo de respuesta medido para la pareja cliente 7 *proxy* 5.

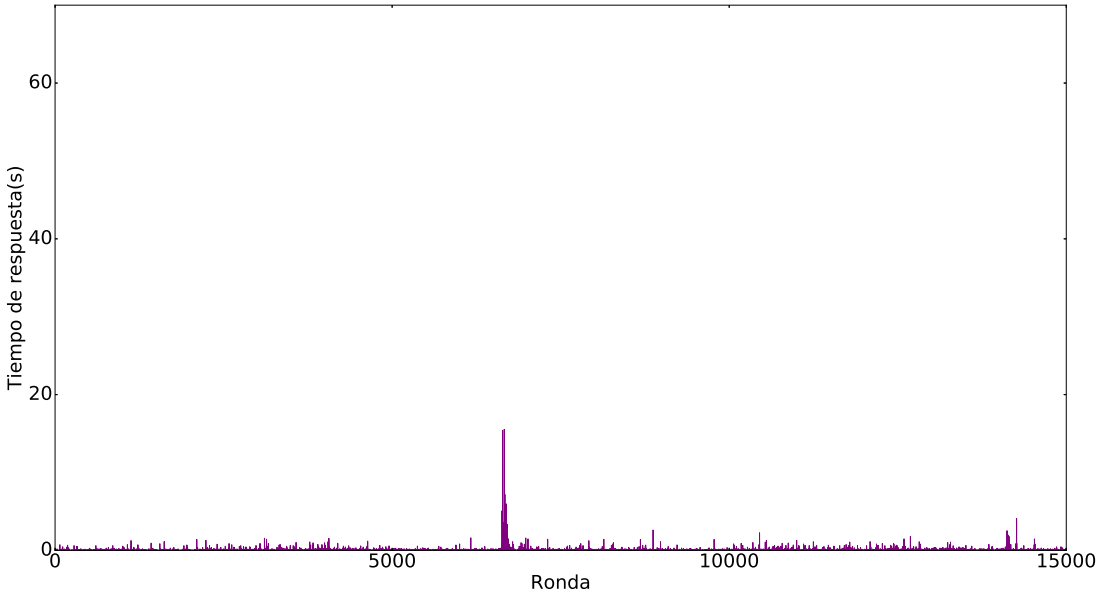


Figura A.36: Evolución del tiempo de respuesta medido para la pareja cliente 8 *proxy* 1.

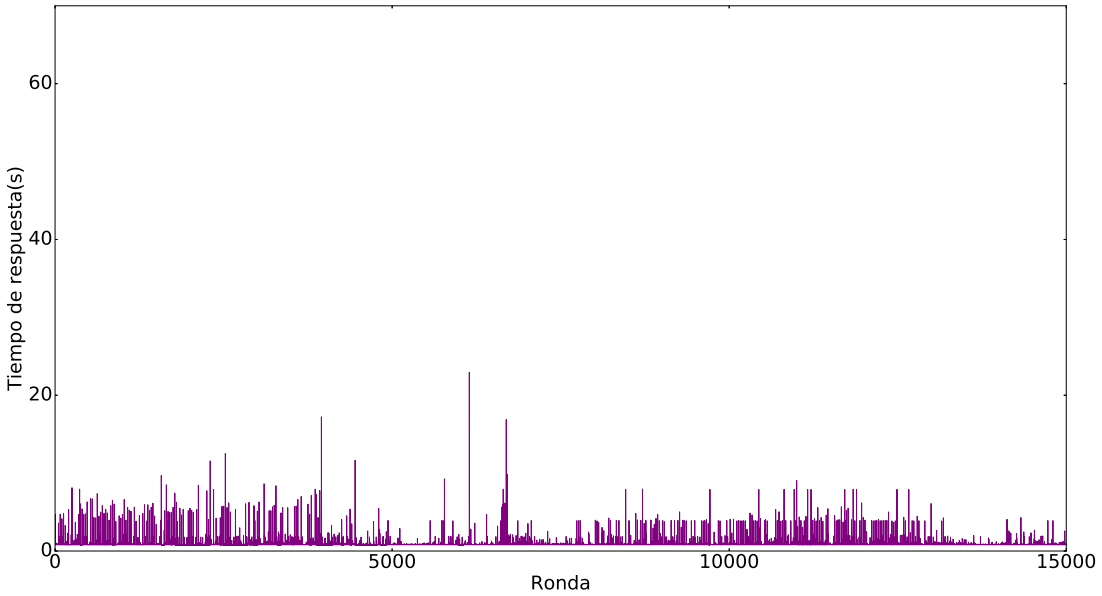


Figura A.37: Evolución del tiempo de respuesta medido para la pareja cliente 8 *proxy* 2.

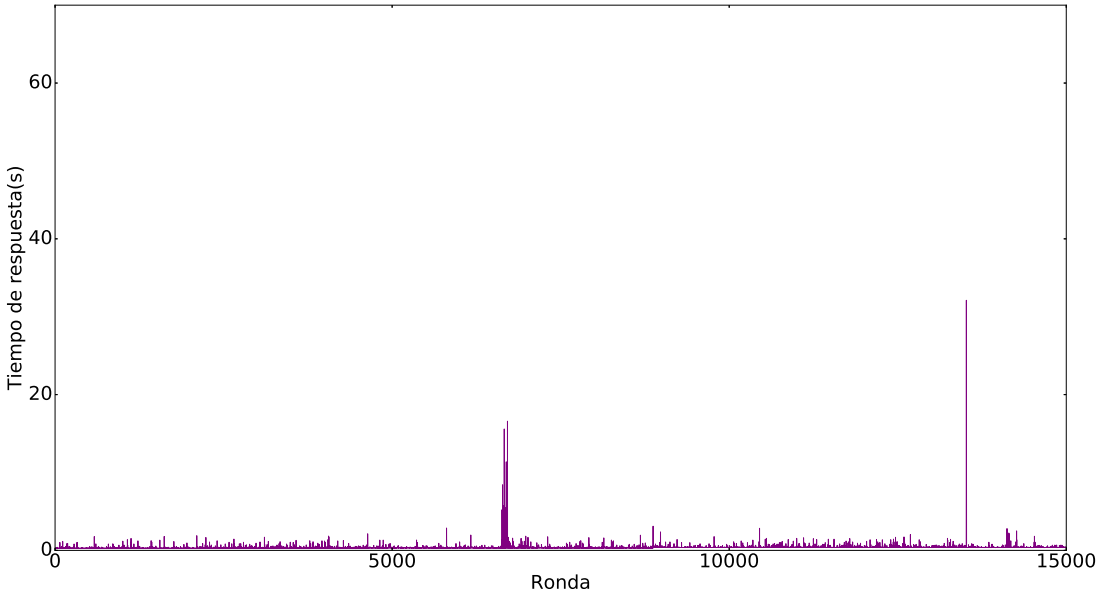


Figura A.38: Evolución del tiempo de respuesta medido para la pareja cliente 8 *proxy* 3.

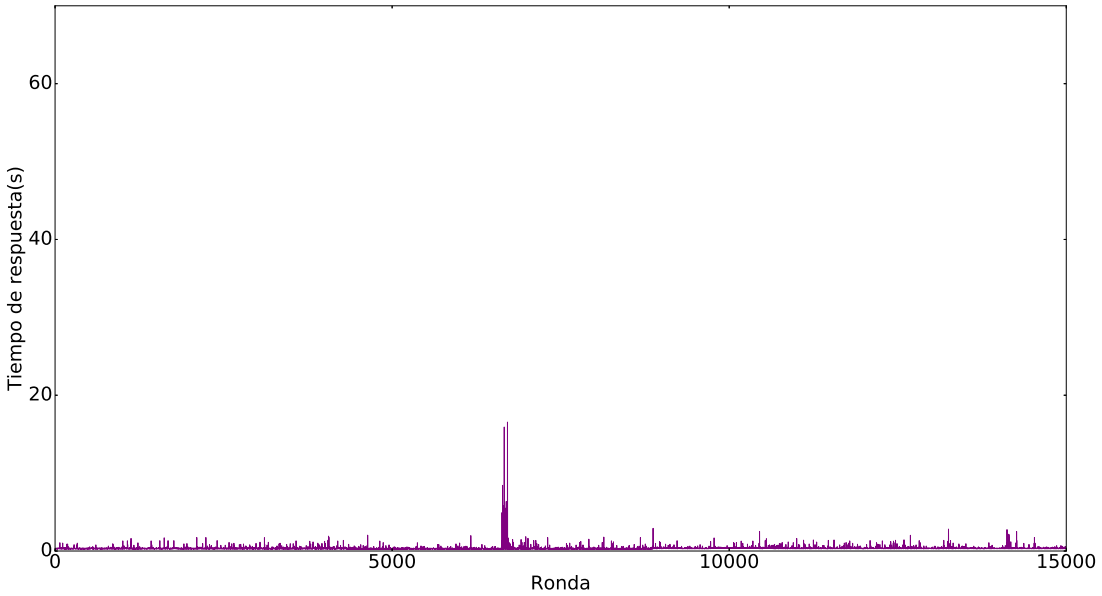


Figura A.39: Evolución del tiempo de respuesta medido para la pareja cliente 8 *proxy* 4.

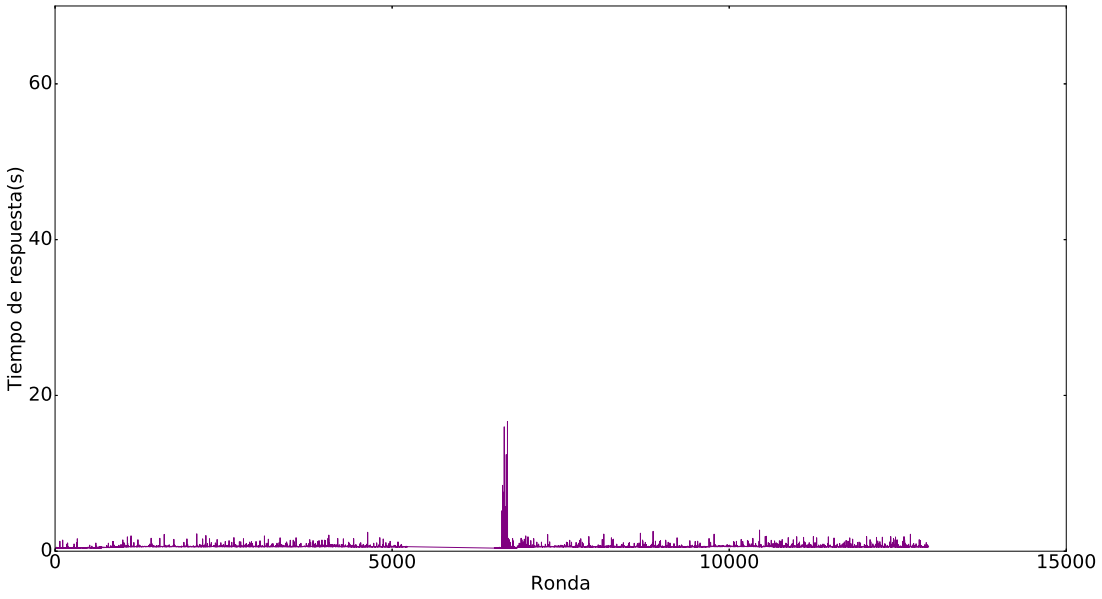


Figura A.40: Evolución del tiempo de respuesta medido para la pareja cliente 8 *proxy* 5.

Apéndice B

Tablas de resultados para todas las densidades

En el capítulo 4 se muestran los resultados para una densidad del 40 %, correspondiente al caso en que cada cliente sondea a 2 *proxies*. En esta sección se muestran los resultados para los otros dos casos, en los que cada cliente sondea a 3 y 4 *proxies* dando lugar a densidades de la matriz de entrenamiento del 60 % y 80 %, respectivamente.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,048	0,052	0,049	0,160	0,108	0,117	0,107	0,201	0,105 ± 0,00080
Proxy 2	0,128	0,133	0,129	0,132	0,173	0,184	0,210	0,129	0,152 ± 0,00146
Proxy 3	0,040	0,040	0,037	0,152	0,101	0,109	0,095	0,046	0,077 ± 0,00039
Proxy 4	0,053	0,046	0,055	0,153	0,114	0,125	0,081	0,055	0,084 ± 0,00026
Proxy 5	0,052	0,048	0,050	0,144	0,096	0,096	0,065	0,047	0,077 ± 0,00077
Media	0,065	0,064	0,065	0,148	0,119	0,126	0,114	0,098	0,100 ± 0,00037

Tabla B.1: Media del MAE obtenido por AMF para cada pareja de cliente y *proxy* con densidad 60 %.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,072	0,069	0,074	0,165	0,143	0,147	0,127	0,251	0,131 ± 0,00668
Proxy 2	0,141	0,147	0,174	0,224	0,235	0,227	0,202	0,112	0,182 ± 0,01503
Proxy 3	0,086	0,066	0,049	0,150	0,147	0,140	0,124	0,052	0,101 ± 0,00319
Proxy 4	0,053	0,051	0,070	0,140	0,123	0,161	0,108	0,060	0,095 ± 0,00216
Proxy 5	0,135	0,095	0,095	0,171	0,121	0,098	0,095	0,067	0,111 ± 0,00953
Media	0,095	0,085	0,092	0,170	0,154	0,155	0,133	0,110	0,124 ± 0,00575

Tabla B.2: Media del MAE obtenido por NTF para cada pareja de cliente y *proxy* con densidad 60 %.

	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Media
Proxy 1	0,543	0,493	0,517	0,403	0,508	0,506	0,401	0,604	0,497 ± 0,00226
Proxy 2	0,377	0,373	0,379	0,632	0,467	0,458	0,448	0,469	0,449 ± 0,00172
Proxy 3	0,505	0,470	0,471	0,382	0,364	0,367	0,346	0,460	0,422 ± 0,00125
Proxy 4	0,407	0,429	0,460	0,379	0,414	0,385	0,396	0,462	0,417 ± 0,00238
Proxy 5	0,308	0,323	0,325	0,436	0,441	0,442	0,497	0,379	0,397 ± 0,00060
Media	0,434	0,423	0,436	0,447	0,439	0,432	0,414	0,480	0,438 ± 0,00070

Tabla B.3: Media del MAE obtenido por PMF para cada pareja de cliente y *proxy* con densidad 60 %.

	Cl, 1	Cl, 2	Cl, 3	Cl, 4	Cl, 5	Cl, 6	Cl, 7	Cl, 8	Media
Proxy 1	0,383	0,306	0,343	0,251	0,369	0,342	0,179	0,539	0,339 ± 0,00114
Proxy 2	0,456	0,531	0,537	0,365	0,453	0,445	0,931	0,325	0,506 ± 0,00247
Proxy 3	0,390	0,329	0,339	0,179	0,217	0,206	0,222	0,277	0,271 ± 0,00043
Proxy 4	0,292	0,298	0,344	0,201	0,316	0,269	0,145	0,289	0,269 ± 0,00057
Proxy 5	0,146	0,116	0,116	0,315	0,163	0,128	0,103	0,099	0,152 ± 0,00113
Media	0,343	0,326	0,347	0,262	0,305	0,279	0,326	0,316	0,313 ± 0,00045

Tabla B.4: Media del MAE obtenido por el algoritmo de referencia 1 para cada pareja de cliente y *proxy* con densidad 60 %.

	Cl, 1	Cl, 2	Cl, 3	Cl, 4	Cl, 5	Cl, 6	Cl, 7	Cl, 8	Media
Proxy 1	0,234	0,165	0,194	0,478	0,221	0,202	0,350	0,360	0,275 ± 0,00095
Proxy 2	0,159	0,154	0,157	0,281	0,261	0,223	0,525	0,230	0,248 ± 0,00191
Proxy 3	0,253	0,188	0,197	0,353	0,136	0,139	0,389	0,144	0,225 ± 0,00112
Proxy 4	0,141	0,142	0,186	0,415	0,169	0,178	0,214	0,138	0,197 ± 0,00065
Proxy 5	0,097	0,078	0,074	0,337	0,176	0,135	0,092	0,103	0,141 ± 0,00136
Media	0,181	0,149	0,166	0,373	0,193	0,176	0,325	0,200	0,220 ± 0,00029

Tabla B.5: Media del MAE obtenido por el algoritmo de referencia 2 para cada pareja de cliente y *proxy* con densidad 60 %.

	Cl, 1	Cl, 2	Cl, 3	Cl, 4	Cl, 5	Cl, 6	Cl, 7	Cl, 8	Media
Proxy 1	0,045	0,049	0,047	0,154	0,097	0,108	0,102	0,202	0,100 ± 0,00168
Proxy 2	0,117	0,127	0,128	0,131	0,172	0,175	0,215	0,130	0,149 ± 0,00262
Proxy 3	0,036	0,037	0,034	0,148	0,100	0,107	0,090	0,043	0,074 ± 0,00114
Proxy 4	0,050	0,044	0,052	0,136	0,112	0,121	0,079	0,053	0,080 ± 0,00154
Proxy 5	0,047	0,042	0,044	0,142	0,101	0,101	0,060	0,042	0,076 ± 0,00280
Media	0,060	0,061	0,062	0,142	0,116	0,123	0,112	0,096	0,096 ± 0,00059

Tabla B.6: Media del MAE obtenido por AMF para cada pareja de cliente y *proxy* con densidad 80 %.

	Cl, 1	Cl, 2	Cl, 3	Cl, 4	Cl, 5	Cl, 6	Cl, 7	Cl, 8	Media
Proxy 1	0,066	0,067	0,072	0,150	0,124	0,138	0,120	0,248	0,123 ± 0,00466
Proxy 2	0,122	0,128	0,143	0,221	0,219	0,200	0,185	0,109	0,165 ± 0,00566
Proxy 3	0,072	0,060	0,045	0,133	0,144	0,136	0,110	0,050	0,093 ± 0,00442
Proxy 4	0,053	0,048	0,066	0,125	0,117	0,159	0,108	0,058	0,091 ± 0,00078
Proxy 5	0,106	0,090	0,078	0,175	0,121	0,097	0,072	0,057	0,102 ± 0,00717
Media	0,083	0,078	0,081	0,161	0,145	0,146	0,121	0,106	0,115 ± 0,00218

Tabla B.7: Media del MAE obtenido por NTF para cada pareja de cliente y *proxy* con densidad 80 %.

	Cl, 1	Cl, 2	Cl, 3	Cl, 4	Cl, 5	Cl, 6	Cl, 7	Cl, 8	Media
Proxy 1	0,419	0,369	0,396	0,364	0,398	0,399	0,322	0,477	0,393 $\pm$ 0,00126
Proxy 2	0,328	0,334	0,342	0,666	0,429	0,406	0,428	0,399	0,415 $\pm$ 0,00657
Proxy 3	0,392	0,363	0,364	0,330	0,287	0,291	0,286	0,342	0,332 $\pm$ 0,00174
Proxy 4	0,304	0,329	0,350	0,326	0,323	0,306	0,321	0,341	0,325 $\pm$ 0,00380
Proxy 5	0,248	0,250	0,246	0,400	0,367	0,370	0,444	0,280	0,330 $\pm$ 0,00345
Media	0,343	0,333	0,344	0,418	0,361	0,354	0,356	0,372	0,360 $\pm$ 0,00180

Tabla B.8: Media del MAE obtenido por PMF para cada pareja de cliente y *proxy* con densidad 80 %.

	Cl, 1	Cl, 2	Cl, 3	Cl, 4	Cl, 5	Cl, 6	Cl, 7	Cl, 8	Media
Proxy 1	0,381	0,305	0,341	0,254	0,368	0,345	0,176	0,538	0,339 $\pm$ 0,00206
Proxy 2	0,452	0,529	0,543	0,370	0,474	0,443	0,941	0,330	0,512 $\pm$ 0,00441
Proxy 3	0,387	0,330	0,341	0,174	0,216	0,206	0,219	0,276	0,270 $\pm$ 0,00255
Proxy 4	0,294	0,299	0,341	0,190	0,312	0,268	0,146	0,288	0,267 $\pm$ 0,00219
Proxy 5	0,146	0,111	0,112	0,314	0,163	0,129	0,092	0,093	0,149 $\pm$ 0,00198
Media	0,342	0,325	0,347	0,260	0,307	0,279	0,327	0,315	0,313 $\pm$ 0,00032

Tabla B.9: Media del MAE obtenido por el algoritmo de referencia 1 para cada pareja de cliente y *proxy* con densidad 80 %.

	Cl, 1	Cl, 2	Cl, 3	Cl, 4	Cl, 5	Cl, 6	Cl, 7	Cl, 8	Media
Proxy 1	0,233	0,152	0,187	0,480	0,212	0,194	0,342	0,360	0,269 $\pm$ 0,00113
Proxy 2	0,144	0,142	0,143	0,276	0,261	0,211	0,532	0,224	0,241 $\pm$ 0,00381
Proxy 3	0,252	0,187	0,198	0,351	0,122	0,129	0,385	0,133	0,219 $\pm$ 0,00208
Proxy 4	0,140	0,140	0,185	0,402	0,164	0,168	0,212	0,131	0,191 $\pm$ 0,00136
Proxy 5	0,091	0,067	0,065	0,340	0,175	0,132	0,085	0,101	0,138$\pm$ 0,00190
Media	0,176	0,141	0,160	0,369	0,187	0,167	0,322	0,194	0,214 $\pm$ 0,00042

Tabla B.10: Media del MAE obtenido por el algoritmo de referencia 2 para cada pareja de cliente y *proxy* con densidad 80 %.