D^{2}ISCO:
Diseño de Sistemas CBR Deliberativos Distribuidos con jCOLIBRI

Sergio González Sanz

dirigido por:

Belén Díaz Agudo

codirigido por:

Juan A. Recio García

Proyecto Fin de Máster en Sistemas Inteligentes

Máster en Investigación en Informática

Facultad de Informática

Universidad Complutense de Madrid

Curso Académico 2008/2009

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: D^{2}ISCO*: Diseño de Sistemas CBR Deliberativos Distribuidos con jCOLIBRI, realizado durante el curso académico 2008-2009 bajo la dirección de Belén Díaz Agudo y codirigido por Juan Antonio Recio García en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.*

Sergio González Sanz

Listado de palabras clave en inglés para su indexación bibliográfica:

- Distribute
- Collaborative
- Deliberative
- Case Based Reasoning (CBR)
- Fuzzy
- Multiagent System
- Distributed Reasoning System

Listado de palabras clave en castellano para su indexación bibliográfica:

- Distribuido
- Colaborativo
- Deliberativo
- Razonamiento Basado en Casos (CBR)
- Difuso
- Sistema multiagente
- Sistema de razonamiento distribuido

Agradecimientos

Quiero aprovechar esta oportunidad para demostrar mi gratitud hacia mis directores de proyecto, Belén Díaz Agudo y Juan Antonio Recio García por el gran esfuerzo y trabajo que han realizado, sacrificando su tiempo para permitir que, tanto este proyecto como el artículo que se escribió a partir de él salieran adelante. Sin ellos, ninguno hubiera sido posible. Desde aqui, mi más sincero agradecimiento.

Quisiera agradecer también al resto del personal docente que forma la Facultad de Informática de la UCM, es especial a aquellos que se encargan de la docencia en el Master de Investigación en Informática, su esfuerzo por trasmitirnos sus conocimientos en los distintos campos y trasmitirnos su pasión por la investigación.

Agradecer también a aquellas personas que han estado estos meses a mi lado, apoyándome en los momentos en los que el trabajo se hacía cuesta arriba, no permitiéndome caer en el desánimo y comprendiendo los sacrificios que supone el desarrollo de este proyecto. A ellos, a mi familia y a mis amigos, muchas gracias.

Por último, agradecer a mis compañeros de trabajo su apoyo, su dedicación y la ayuda prestada sin dilaciones siempre que la he necesitado.

Sergio González Sanz

Abstract

In this proyect we describe D^{2}ISCO[a]: a framework to design and implement deliberative and collaborative Case Based Reasoning (CBR) systems. Using D^{2}ISCO we design and implement distributed CBR systems where each node collaborates, arguments and counterarguments its local results with other nodes to improve the performance of the system's global response. D^{2}ISCO is integrated as a part of jCOLIBRI 2 [1] an established framework in the CBR community, enabling the developing of distributed CBR systems. We perform a case study for a collaborative music recommender system and present the results of an experiment of the accuracy of the system results using a fuzzy version of the argumentation system AMAL [2] and a network topology based on a social network.

[a] D^{2}ISCO: Deliberative DIStributed CBR systems with jCOlibri

Resumen

En este proyecto describimos D^{2}ISCO[a]: una plataforma para diseñar e implementar sistemas de Razonamiento Basado en Casos (CBR) deliberativos y colaborativos. Utilizando D^{2}ISCO podemos diseñar e implementar sistemas CBR distribuidos donde cada nodo colabora, argumenta y contraargumenta sus resultados con otros nodos para mejorar el rendimiento de la respuesta global del sistema. D^{2}ISCO se ha integrado como una parte de jCOLIBRI 2 [1] una reconocida plataforma en la comunidad CBR, permitiendo el desarrollo de sistemas CBR distribuidos. Hemos realizado un caso de estudio de un sistema de recomendación musical y presentamos los resultados de un experimento donde medimos la precisión del sistema utilizando una versión difusa del sistema de argumentación AMAL [2] y una topología de red basada en una red social.

[a] D^{2}ISCO: sistemas CBR Deliberativos DIStribuidos con jCOlibri

Índice general

Índice de figuras

Capítulo 1

Introducción

En este proyecto describimos D^{2}ISCO: una plataforma para diseñar e implementar sistemas de Razonamiento Basado en Casos (CBR) deliberativos y colaborativos. En D^{2}ISCO se integran un marco teórico para la caracterización de sistemas CBR distribuidos, así como un marco práctico para su construcción.

Utilizando D^{2}ISCO podemos diseñar e implementar sistemas CBR distribuidos donde cada nodo colabora, argumenta y contraargumenta sus resultados con otros nodos para mejorar el rendimiento de la respuesta global del sistema. D^{2}ISCO se ha integrado como una parte de jCOLIBRI 2 [1] una reconocida plataforma en la comunidad CBR, permitiendo el desarrollo de sistemas CBR distribuidos.

Para probar el funcionamiento de D^{2}ISCO, hemos realizado un caso de estudio de un sistema de recomendación musical y presentamos los resultados de un experimento donde medimos la precisión del sistema utilizando una versión difusa del sistema de argumentación AMAL [2] y una topología de red basada en una red social.

EL CBR o Case-Based Reasoning (Razonamiento Basado en Casos) es una técnica de Inteligencia Artificial que se basa en la premisa de que problemas parecidos pueden resolverse a través de soluciones similares y que los problemas tienden a repetirse [3]. Esto les permite nutrirse de la experiencia pasada para poder dar solución a los problemas que se les presentan.

Uno de los sectores donde mayor desarrollo comercial han tenido los sistemas CBR es en el de las ventas por Internet. Páginas como *Amazon*[1] o *La Casa del Libro*[2] han sabido explotar el potencial de negocio de los sistemas CBR recomendadores y se han adaptado al uso de estas tecnologías con excelentes resultados, incrementando sus ventas.

Los sistemas CBR explotan la información contenida en los perfiles de usuario. Gracias a ella, pueden mejorar el resultado de las soluciones que presentan. La información acerca del perfil de un usuario y de sus gustos resulta fundamental, por ejemplo, en los sistemas recomendadores.

[1] http://www.amazon.com/

[2] http://www.casadellibro.com/

La redes sociales, tales como *Facebook*[3], *MySpace*[4] o *Tuenti*[5] (entre otras muchas), tienen como objetivo el intercambio de información entre sus usuarios. La expansión de las redes sociales tiene su auge en los últimos años coincidiendo con la expansión de Internet, donde han cobrado una gran importancia. Las redes sociales contienen mucha información acerca de los usuarios de las mismas, tanto explícita, como los enlaces entre dos usuarios de la red, como implícita, como la existencia de similitudes entre dos usuarios enlazados.

Los sistemas CBR distribuidos pueden explotar el uso de la información contenida en las redes sociales. Nutriéndose de la información como los perfiles de usuario, los enlaces de la red, o las actividades de un usuario, pueden mejorar las respuestas generadas para problemas vinculados a estas redes sociales.

Como parte de este trabajo, se ha escrito un artículo de investigación de nombre *"DISCO: Distributed Deliberative CBR Systems with j*COLIBRI*"* que ha sido presentado y aceptado en el congreso ICCCI '09 (1^{st} International Conference on Computational Collective Intelligence).

[3] http://www.facebook.es/

[4] http://es.myspace.com/

[5] http://www.tuenti.com/

1.1. Objetivos

Los objetivos planteados a resolver dentro de este trabajo son los siguientes:

- **Proporcionar un marco teórico general para caracterizar los sistemas CBR distribuidos y deliberativos a partir del estudio de sistemas concretos de la literatura.** Gracias a este marco, podremos realizar una caracterización de los distintos sistemas CBR distribuidos existentes, así como realizar una clasificación de los mismos. Además, nos permitirá definir estos sistemas mediante el conjunto de características que presentan lo cuál ayudará a su identificación y a su comprensión.

- **Plasmar el marco teórico anterior en un marco práctico o plataforma de diseño de sistemas CBR distribuidos** Gracias a este marco práctico, se ampliará la funcionalidad de jCOLIBRI 2, permitiendo la construcción de cualquiera de los sistemas CBR distribuidos caracterizados en el marco teórico realizado.

- **Instanciar el marco teórico general para diseñar un sistema CBR distribuido con unas características concretas como un protocolo de negociación, una topología de red, un modelo de confianza, etc.** La instancia construida nos servirá además para validar el marco teórico diseñado.

- **Comprobar el funcionamiento del sistema implementado sobre un caso de estudio.** Construiremos un recomendador musical sobre el sistema CBR distribuido implementado en el objetivo anterior, comprobando su funcionamiento y su rendimiento sobre varias características de su diseño, como pueden ser el número de nodos o la topología de red utilizada.

1.2. Estructura de la memoria

En primer lugar, antes de comenzar con la resolución de cada uno de los objetivos que acabamos de presentar, en el Capítulo 2 se realiza un resumen del estado del arte en los campos más relevantes para la realización de este proyecto, como son los sistemas CBR tradicionales, los sistemas CBR distribuidos, los protocolos de argumentación existentes, los sistemas de razonamiento fuzzy o las redes sociales.

Una vez situado el estado del arte en estos campos, pasaremos a exponer en el Capítulo 3 el marco teórico realizado, resolviendo el primero de los objetivos de este trabajo.

El en Capítulo 4 se da solución al segundo de nuestros objetivos: la extensión de jCOLIBRI 2 para permitir la construcción de sistemas CBR distribuidos caracterizados mediante el marco descrito en el Capítulo 3.

Contando ya con la plataforma de desarrollo de sistemas CBR distribuidos, pasaremos a instanciar un sistema concreto en el Capítulo 5, que dará solución al tercero de los objetivos planteados al comenzar el proyecto.

El último de los objetivos de este trabajo se encuentra descrito en el Capítulo 6, en el que se utiliza el sistema creado para construir un recomendador musical distribuido, aportando medidas de su funcionamiento y rendimiento en función de algunas de sus características.

El último de los capítulos de esta memoria se utiliza para describir las conclusiones a las que nos ha llevado el desarrollo de este trabajo, así como el trabajo futuro que surge tras la finalización del mismo.

Además se incluyen los siguientes apéndices:

- **Reglas del sistema fuzzy:** Apéndice en el que se describen las reglas que utiliza el sistema fuzzy descrito en la Sección 5.3 para llevar a cabo la toma de decisiones dentro del protocolo de argumentación descrito en la Sección 5.2.

- **Tipos del sistema fuzzy:** Apéndice en el que se describen los tipos de datos de la información manejada por este sistema fuzzy.

Capítulo 2

Estado del arte

La expansión sufrida por los sistemas CBR (Case-Based Reasoning) en los últimos años los han convertido en una de las tecnologías de la Inteligencia Artificial más difundida. Su aplicación en diversos campos como los departamentos de ventas [4, 5, 6] de numerosas empresas, la planificación, por ejemplo, de viajes turísticos [7] o recursos en computación Grid [8], a la mejora de las búsquedas web [9, 10], a la ayuda al aprendizaje online [11], en los recomendadores [12, 13, 14, 15], a la clasificación de especies [2], etc., ha impulsado su desarrollo comercial y académico.

Uno de los sectores donde mayor desarrollo comercial han tenido los sistemas CBR es en el de las ventas por Internet. Páginas como *Amazon*[1] o *La Casa del Libro*[2] han sabido explotar el potencial de negocio de los sistemas CBR recomendadores y se han adaptado al uso de estas tecnologías con excelentes resultados.

El objetivo de este capítulo es plantear una visión general de las investigaciones en torno a los sistemas CBR. Por ello, en las siguientes secciones trataremos las diversas alternativas surgidas en los últimos años en el campo de los sistemas CBR tradicionales así como las nuevas líneas de investigación existentes, la plataforma jCOLIBRI para la construcción de sistema CBR, los distintos sistemas CBR distribuidos surgidos como evolución de los sistemas CBR tradicionales, los sistemas recomendadores basados en sistemas CBR y el uso de las redes sociales aplicadas a la construcción de sistemas CBR distribuidos.

2.1. Sistemas CBR tradicionales

Los sistemas CBR se basan en la premisa de que problemas parecidos pueden resolverse a través de soluciones similares y que los problemas tienden a repetirse [3]. Esto les permite nutrirse de la experiencia pasada para poder dar solución a los problemas que se les presentan. Cuando un sistema CBR se enfrenta a la resolución

[1] http://www.amazon.com/

[2] http://www.casadellibro.com/

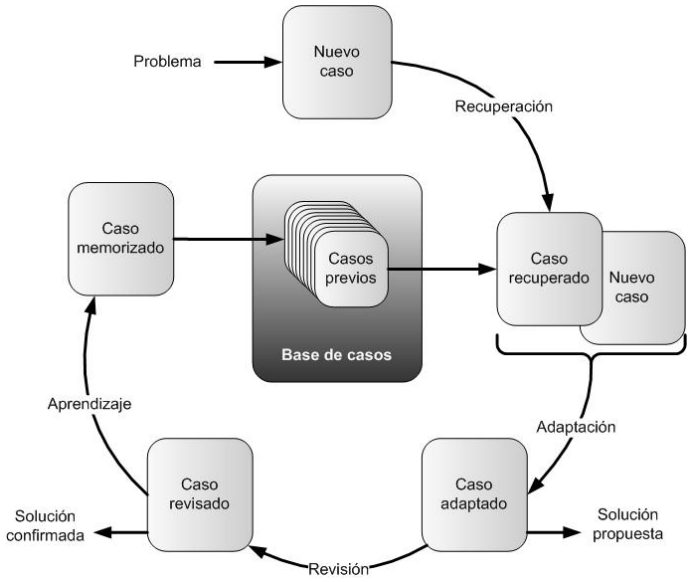

Figura 2.1: Fases del ciclo CBR

de un nuevo problema, buscará en su base de casos los problemas similares al problema actual. Una vez encontrados, adaptará las soluciones de los casos previos al problema actual para dar una respuesta válida. El razonamiento CBR surgió a raíz de los trabajos de Roger Schank [16] en memoria dinámica y el papel central que los episodios previos (casos) tenían en la resolución de problemas y el aprendizaje.

El proceso de solución de un problema en un sistema CBR consta de cuatro fases, en lo que conocemos como ciclo CBR (Figura 2.1) [17]. Las fases de este ciclo son:

1. **Recuperación:** Durante esta fase se busca en la base de casos del sistema el caso más similar al problema actual.

2. **Adaptación:** Se adapta el caso recuperado al problema actual.

3. **Revisión:** Se comprueba que la solución del caso adaptado resuelva el problema inicial.

4. **Aprendizaje:** La solución adaptada junto con la descripción del problema se incluyen a la base de casos como un nuevo caso.

El aprendizaje de los sistemas CBR se produce durante la fase de *aprendizaje*. Al introducir los nuevos casos en la base de casos se consigue que el sistema aprenda de la experiencia pasada. De esta forma, ante la llegada de un problema, el sistema CBR sabrá resolverlo si en su base de casos encuentra casos similares al

problema al que se está enfrentando. En caso contrario, no podrá resolver el problema o la solución propuesta será errónea, no siendo validada durante la ase de revisión y por lo tanto no entrando a formar parte de su base de casos.

Dado que los sistemas CBR necesitan disponer de una base de casos para trabajar, no son apropiados para resolver aquellos problemas de los que no se tiene casos previos, es decir, problemas a los que va a tener que enfrentarse el sistema CBR cubriendo todo el rango posible junto con sus soluciones. Son apropiados por tanto, para resolver problemas en los que si existe este conjunto de problemas junto con sus soluciones, como por ejemplo, problemas de diagnóstico o de clasificación.

Para determinar qué casos de los contenidos en su base de casos se parecen más al problema que debe resolver, el sistema CBR utiliza las medidas de similitud. Una medida de similitud es una función que dados dos casos es capaz de determinar cuánto de parecidos son. Las medidas de similitud resultan fundamentales para el correcto funcionamiento de los sistemas CBR, puesto que van a determinar los casos que son devueltos durante la fase de recuperación.

En muchas ocasiones, cuando el tamaño de las bases de casos crece mucho, es necesario aplicar técnicas que permitan aumentar el rendimiento del sistema. Estas técnicas son conocidas como mecanismos de indexación. Gracias a ellas, las bases de casos son indexadas y ordenadas de tal forma que las nuevas búsquedas se realizan de una forma más rápida, mejorando el rendimiento del sistema. Los atributos de indexación en torno a los cuáles se realice el proceso de indexación dependerán del algoritmo utilizado, variando desde métodos de selección manual de atributos hasta métodos automáticos.

Algunas de las aplicaciones de los sistemas CBR son la construcción de sistemas CBR textuales, para la recuperación y búsqueda de documentos mediante consultas en forma de texto, los sistemas recomendadores, que se tratarán en la Sección 2.2 o los sistemas CBR de carácter general como aplicaciones médicas, comerciales, de gestión, etc..

2.2. Sistemas recomendadores

Los sistemas recomendadores son una de las ramas de mayor aplicación de los sistemas CBR. Podemos distinguir dos clases fundamentales de sistemas recomendadores ([18]) en función de en qué basan sus recomendaciones:

- **Colaborativos:** Usan historiales, normalmente en forma de perfiles con valoraciones. Las recomendaciones se basan en el perfil de usuario y en los compañeros de recomendación (aquellos usuarios cuyas valoraciones son próximas a las del usuario actual). Recomendarán casos que no existan en el perfil de este usuario, pero que sus compañeros hayan valorado de forma elevada. No necesitan la descripción de los productos a recomendar.

- **Basados en casos:** Las descripciones de los productos a recomendar son fundamentales, dado que el conjunto de recomendaciones se formará con

los productos cuyas descripciones más se ajusten a la consulta del usuario.

- **Mixtos o híbridos:** Son aquellos recomendadores que usan las características de los dos grupos anteriores. El conjunto de los productos recomendados será aquél con los productos cuyas descripciones más se ajusten a la consulta del usuario, filtrando los resultados con las valoraciones de los compañeros de recomendación del usuario actual.

Otro aspecto fundamental dentro de los recomendadores es cómo obtienen los datos de la consulta del usuario. Podemos distinguir dos tipos de recomendadores en función de este aspecto:

- **Navigation-by-asking o navegación por preguntas:** Estos recomendadores obtienen los datos de la consulta del usuario a través de preguntas que se le van formulando secuencialmente. Utilizan métodos como, por ejemplo, los árboles de decisión para determinar el orden de realización de las preguntas al usuario a fin de minimizar, entre otros parámetros, el número de preguntas realizadas o el tiempo de recomendación.

- **Navigation-by-proposal o navegación por propuesta:** En estos recomendadores, en vez de realizar preguntas al usuario, se le muestran una serie de productos de la base de casos entre los que el usuario debe elegir el que más se ajuste al que busca. Una vez que el usuario ha seleccionado uno de los productos mostrados, el sistema mostrará productos similares al escogido, hasta que el usuario encuentre el producto buscado. En este tipo de recomendadores juega un papel importante la diversidad en los productos mostrados, para poder determinar las características de la búsqueda del usuario con mayor rapidez.

- **Mixtos o híbridos:** Se trata de aquellos recomendadores que integran las capacidades de los dos anteriores. Se trata de recomendadores en los que se alternan las preguntas con la visualización de productos para que el usuario pueda encaminarse de una manera más rápida hacia el producto deseado.

También podemos realizar una clasificación de los sistemas recomendadores en función de la actividad que realizan ante el comienzo de una nueva sesión:

- **Reactivos:** No realizan ninguna recomendación al iniciar la sesión. Esperan a que el usuario comience el diálogo.

- **Proactivos:** Realizan recomendaciones al iniciar la sesión, a pesar de que el usuario no haya realizado ninguna consulta. Suele tratarse de recomendadores colaborativos, basándose la recomendación únicamente en las valoraciones de otros usuarios. También los recomendadores basados en casos pueden mostrarse proactivos, para tener una primera selección de productos donde el usuario puede realizar sus elecciones.

Otra característica importante dentro de los sistemas recomendadores es la de proporcionar información acerca de las decisiones que se toman, como por ejemplo, el porqué de realizar una pregunta, mostrar un producto o realizar cierta recomendación. Por ejemplo, la explicación acerca de la realización de una pregunta puede ayudar al usuario a discriminar mejor entre dos productos.

2.3. Sistemas CBR distribuidos

El desarrollo de Internet y el crecimiento de la información a tratar llevaron a los sistemas CBR a dar un nuevo paso en su desarrollo. El tamaño de las bases de casos de los sistemas CBR creció significativamente lo que llevó a los expertos en el área a desarrollar sistemas CBR cuyas bases de casos se dividían en varias partes para poder hacerlas tratables, o lo que conocemos con el nombre de sistemas CBR distribuidos.

Las investigaciones en el área de los sistemas CBR distribuidos han ido dirigidas hacia dos enfoques distintos: cómo se organiza y se gestiona el conocimiento, y cómo el conocimiento es procesado por el sistema [19]. La primera de ellas muestra las diferencias entre tener una única base de casos frente a varias y la segunda se centra en el número de sistemas CBR que procesan la información y cómo combinan sus resultados. Una parte importante de las investigaciones ha ido encaminadas a estudiar diferentes mecanismos de colaboración entre agentes sobre bases de casos distribuidas, como es el caso de [20].

El crecimiento de las bases de casos de los sistemas CBR no es el único motivo que ha llevado a la aparición de los sistemas CBR distribuidos. El efecto de unión o *"ensemble efect"* expuesto en [21] demuestra que el razonamiento aportado por dos agentes trabajando sobre una base de casos distribuida mejora los resultados obtenidos por un solo agente trabajando sobre esa misma base de casos. De esta forma, se abre una nueva vía de investigación destinada a encontrar los protocolos de argumentación adecuados para permitir a agentes CBR deliberar a cerca de la solución común a un problema, así como intercambiar conocimiento en forma de argumentos o contraejemplos contra las soluciones del resto de los agentes. De esta forma surge el protocolo AMAL, desarrollado en [2], permitiendo a varios agentes CBR deliberar acerca de un problema común.

Existen diversos mecanismos para la toma de decisiones y para la generación de argumentos y contraejemplos dentro de los protocolos de argumentación. Algunos de ellos, como el descrito en [2], utilizan las Lógicas Descriptivas (DL) [22] como mecanismo a partir del cual poder realizar argumentaciones sobre los casos. Otras alternativas hacen uso de la Lógica Difusa [23, 24] para realizar la toma de decisiones necesaria para la gestión de los sistemas multiagente. Estos sistemas aprovechan el tipo de información no nítido manejado por el sistema para realizar un razonamiento difuso, similar al que podría realizar un humano, obteniendo una mejora significativa en el rendimiento del sistema.

De forma paralela se ha desarrollado otras alternativas para el otro enfoque en

los sistemas CBR distribuidos, aquél que se orienta hacia la organización y gestión del conocimiento. De esta forma, han surgido estudios [25, 26] acerca de cómo modularizar el conocimiento almacenado en los sistemas CBR y aprovechar el mismo. En esta clase de sistemas cada uno de los agentes CBR de los mismos se encarga de un área de conocimiento, tal que los casos contenidos en su base de casos cubren dicha área. Además, resulta habitual que existan agentes no CBR (aquellos que no poseen una base de casos) en el sistema, cuya misión es la de descomponer las consultas en las áreas de conocimiento que abarca y solicitar el conocimiento requerido a los agentes adecuados, componiendo después la información recuperada.

El salto de los sistemas CBR hacia los sistemas distribuidos, ha traído consigo la aparición de un nuevo parámetro que no era considerado en los sistemas CBR tradicionales: la confianza. Antes de la aparición de los sistemas CBR distribuidos este parámetro no tenía sentido al existir una única base de casos dentro del sistema. Con la inclusión de varias bases de casos aparece este nuevo parámetro, que va a determinar en qué medida los casos devueltos por una de las bases de casos del sistema satisfacen las necesidades del resto de los individuos del sistema ante una determinada consulta. Los trabajos dentro del ámbito de la confianza dentro de los sistemas CBR distribuidos se encaminan hacia el cálculo automático de medidas de confianza entre los agentes del sistema [27, 28, 29, 30], así como a la forma adecuada de introducir la confianza dentro del procesamiento de la información para mejorar los resultados obtenidos por el sistema [15, 13, 14].

2.4. jColibri

jCOLIBRI es una plataforma de desarrollo de sistemas CBR en Java desarrollada por el grupo GAIA[3] de la Facultad de Informática de la Universidad Complutense de Madrid. La primera versión de la plataforma fue desarrollada como prototipo en LISP, evolucionando después a su versión Java (jCOLIBRI).

jCOLIBRI tiene dos versiones principales:

- jCOLIBRI versión 1: jCOLIBRI 1 es la primera versión de la plataforma de desarrollo. Incluye una completa interfaz gráfica de usuario que guía al usuario en el diseño de un sistema CBR. Esta versión es recomendada para usuarios no programadores que quieran crear un sistema CBR si programar nada de código.

- jCOLIBRI versión 2: jCOLIBRI 2 es una nueva implementación con una nueva y más clara arquitectura dividida en dos capas: una orientada a desarrolladores y otra orientada a diseñadores. El nuevo diseño permite a los desarrolladores incluir las características de jCOLIBRI en sus aplicaciones CBR.

[3] Group for Artificial Intelligence Applications - http://gaia.fdi.ucm.es/

La capa de desarrollo de jCOLIBRI 2 contiene los componentes básicos de una plataforma de desarrollo. El objetivo de esta capa es que sea utilizada por los programadores dentro de los propias aplicaciones, sin incluir ninguna clase de herramientas gráficas. Incluye las herramientas necesarias para resolver la mayoría de los problemas identificados en la versión previa, de forma que se presenta como una herramienta fundamental para el desarrollo de cualquier tipo de sistema CBR.

La capa de diseño de jCOLIBRI 2 incluye multitud de herramientas con interfaces gráficas de usuario. Esta capa abstrae todo el funcionamiento de la aplicación en el manejo del sistema CBR, ayudando al usuario a diseñar un sistema CBR completo a través de un proceso de configuración sin necesidad de conocimientos previos de programación o del lenguaje específico.

Algunas de las características que incluye jCOLIBRI 2 son:

- Herramientas para la construcción de sistemas CBR textuales como recuperación textual con Apache Lucene, clustering textual con Carrot2, medidas de similitud con OpenNLP y GATE.

- Herramientas para la visualización, clasificación y mantenimiento de bases de casos, así como diversos conectores que permiten importar y exportar bases de casos desde diferentes formatos, utilizando Hibernate y JavaBeans.

- Medidas de similitud basadas en ontologías gracias al uso de OntoBridge.

- Integración en Tomcat y acceso a Wordnet.

- Dispone de 16 ejemplos totalmente desarrollados y documentados que sirven como referencia al usuario a la hora de construir un sistema CBR utilizando jCOLIBRI y le muestran el uso de cada una de las características desarrolladas en la plataforma para facilitar su comprensión y su posterior uso.

jCOLIBRI permite construir sistemas CBR de los tipos:

- **CBR de carácter general:** Incluye método que facilitan la construcción de sistemas CBR de propósito general como recuperación filtrada de casos, uso de la diversidad para la selección y recuperación de casos, métodos para obtención de consultas de forma gráfica y para la visualización de resultados, etc.

- **CBR textual:** Con los métodos comentados anteriormente para la recuperación, clustering o medidas de similitud textual, permite la construcción de sistemas CBR textuales de forma sencilla.

- **Sistemas recomendadores:** Dispone de los métodos necesarios para la construcción de recomendadores colaborativos, navegación por preguntas y navegación por propuestas, así como la obtención de consultas a través de perfiles de usuario o medidas de similitud local.

2.5. Redes sociales

La necesidad de los nuevos sistemas CBR distribuidos de crear medidas de confianza entre los distintos agentes del sistema, ha llevado a los investigadores a buscar un medio de importar estas medidas de estructuras ya existentes. De este modo, entran en juego las redes sociales dentro del ámbito de los sistemas CBR distribuidos. Estas redes, no sólo proporcionar una medida de confianza entre los diversos usuarios que forman parte de ellas, sino que nos aportar toda una estructura de red construida entre ellos. Un enlace entre dos usuarios de la red social simboliza una afinidad entre estos acerca dentro de la temática de la red.

La redes sociales, tales como *Facebook*[4], *MySpace*[5] o *Tuenti*[6] (entre otras muchas), tienen como objetivo el intercambio de información entre sus usuarios. La temática de las redes sociales es amplia, variando desde la laboral al intercambio de fotos y música. La expansión de las redes sociales tiene su auge en los últimos años coincidiendo con la expansión de Internet, donde han cobrado una gran importancia. Los usuarios de las redes sociales buscan un lugar donde encontrar gente similar a ellos dentro de la temática de la red y con quien poder compartir sus ideas.

El mundo de los sistemas CBR distribuidos ha aprovechado esta sinergia en su favor y poder utilizar toda la información contenida en las redes sociales para construir sistemas con distintos propósitos. Así, en los últimos años, se han creado sistemas que utilizaban estas redes, representando cada usuario por una base de casos y un agente CBR que la gestiona, y utilizando como topología de red y como valores de confianza entre agentes aquellos importados de éstas. Con este espíritu, se crean sistemas como [13] o [15], recomendadores en los que el ámbito de los casos recomendados coincide con la temática de la red social utilizada.

Existen otras aplicaciones de las redes sociales dentro del mundo de los sistemas CBR. Por ejemplo, en [31] se muestran diferentes sistemas que han utilizado las redes sociales como un mecanismo para recomendar personas con las que colaborar o como utilizar visualizaciones de redes sociales para poder aprovechar las diferentes colaboraciones que surgen entre usuarios de un mismo entorno laboral.

[4] http://www.facebook.es/
[5] http://es.myspace.com/
[6] http://www.tuenti.com/

Capítulo 3

Marco teórico para sistemas CBR distribuidos

La construcción de cualquier tipo de sistema requiere tomar una serie de decisiones de diseño acerca de las características que va a tener el sistema construido. Cada una de estas decisiones requiere conocer las consecuencias que tendrá, así como cuáles son las aportaciones que realiza. Por ello, antes de asumir cualquiera de las decisiones de diseño de un sistema, debemos conocer en profundidad cuáles son sus características y cómo influyen en el sistema completo.

Gracias al marco teórico expuesto en este capítulo, pretendemos que los futuros desarrolladores de sistemas CBR distribuidos dispongan de una referencia a la hora de desarrollar sus sistemas, sabiendo cómo van a afectar sus decisiones de diseño al sistema construido.

Este capítulo está estructurado de la siguiente manera: en primer lugar realizaremos una introducción a los sistemas CBR distribuidos para que el lector conozca qué son y cómo funcionan, para posteriormente poder comprender cada una de las características expuestas en la Sección 3.2 así como las consecuencias que tienen dentro de los sistemas CBR distribuidos.

3.1. Introducción

Consideramos como un sistema CBR distribuido aquél sistema CBR cuya base de casos no se encuentra localizada en un sólo punto, sino que se encuentra dividida ya sea en un mismo lugar o bien repartida a lo largo del mundo, o a aquél sistema CBR formado por varias bases de casos. De forma habitual, cada una de estas bases de casos se encuentra gestionada por un agente, que se encargará de realizar las operaciones CBR relacionadas con ella, así como de gestionar las conexiones de red y las comunicaciones con otros agentes que estén representando a otras bases de casos (Figura 3.1).

Dado que la base de casos de un sistema CBR distribuido se encuentra dividida o distribuida (de ahí el nombre de sistemas CBR distribuidos), el ciclo CBR se tiene

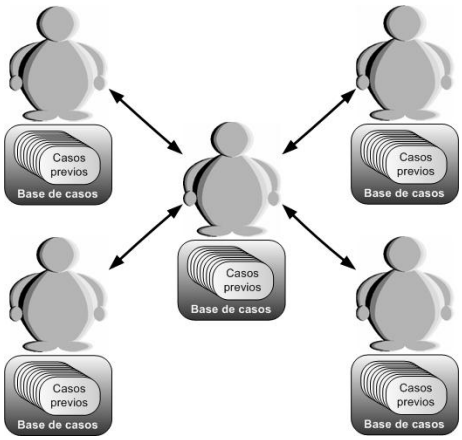

Figura 3.1: Esquema sistema CBR distribuido

que realizar de manera independiente en cada una de las partes de la bases de casos. En este punto nos plateamos un primer interrogante: ¿cómo devolverá el sistema distribuido una única solución si cada una de sus bases de casos está generando una? Al mecanismo a través del cuál el sistema CBR distribuido genera una única solución para el problema a partir de las múltiples soluciones generadas por su base de casos se le denomina *proceso de composición de resultados*.

Además, es muy posible que no todas las soluciones de las distintas bases de casos sean consecuentes entre si. ¿Qué sucederá si dos agentes presentan soluciones opuestas? Los sistemas CBR utilizan un *protocolo de argumentación* para determinar qué soluciones son más adecuadas en cada caso. Dentro de los diferentes protocolos de argumentación, los agentes del sistema podrán exponer sus argumentos a favor y en contra de las diferentes soluciones propuestas por cada uno de los agentes del sistema para resolver un problema.

Un parámetro importante dentro de los sistemas multiagente es el de la confianza entre los distintos agentes del sistema. ¿Utilizarán los sistemas CBR distribuidos parámetros de confianza entre los diversos agentes que gestionan las bases de casos?. Habitualmente sí, en lo que denominamos *modelos de confianza*, representando la confianza entre agentes la calidad de las respuestas generadas por cada uno de ellos.

Dado que en los sistemas CBR distribuidos cada agente es responsable de realizar las tareas CBR sobre su base de casos, deberemos tener en cuenta como realizaran la *recuperación de casos* cada uno de ellos. ¿Realizan todos los agentes la recuperación de casos de una misma forma o lo harán de formas distintas?. También debemos plantearnos si las distintas *bases de casos* utilizan los mismos tipos de casos, si se solapan entre ellas o el tamaño que tienen.

Por último, no debemos olvidarnos del *número de agentes* del sistema distribuido, vinculado al número de bases de casos que utilicemos, ni de la *topología de*

red, que definirá la conexiones existentes entre los distintos agentes del sistema.

3.2. Caracterización de los sistemas CBR distribuidos

El objetivo de esta sección es exponer de una forma teórica las principales variables de diseño existentes en la construcción de sistemas CBR distribuidos, de tal forma que podamos conocer en todo caso cómo influirán cada una de estas características dentro del sistema completo. Además, nos permitirá caracterizar de forma única cada uno de los sistemas que podamos desarrollar.

Una de las contribuciones de este trabajo es el marco teórico descrito en este capítulo. Actualmente no existen caracterizaciones de los sistemas CBR distribuidos, debido a su reciente expansión y a la falta de estandarizaciones entre ellos. Con este marco se pretende aportar un entorno dentro de cuál se incluyan todos los sistemas distribuidos actuales definiendo cada uno de ellos en función de sus características.

A continuación se describen las características identificadas dentro de los sistemas CBR distribuidos estudiados, centrándonos en aquellas que son concretas de los sistemas CBR distribuidos, sin incluir las características propias de los sistemas CBR tradicionales, tales como las medidas de similitud, los mecanismos de indexación y el resto de características vistas en la Sección 2.1.

3.2.1. Número de agentes del sistema distribuido

Una de las principales características de los sistemas CBR distribuidos, es el número de agentes[1] que forman el sistema. El número de agentes resulta fundamental en el tipo de problema a resolver y determina otras características como la topología de red o la composición de resultados entre distintos agentes.

La clasificación de los sistemas CBR distribuidos en función del número de agentes que los forman es la siguiente:

- **Pequeños:** Sistemas multiagente con escasa necesidad de recursos de computación, normalmente ejecutados en una única máquina con varios procesadores o núcleos harfware. Han sido utilizados, por ejemplo, en [2] para resolver problemas de clasificación de individuos dentro de familias vegetales o en [32] para modelar el comportamiento de los jugadores dentro de la RoboCup[2], donde cada agente representa a un individuo de cada uno de los equipos contendientes.

- **Medianos:** Sistemas que necesitan cierta infraestructura hardware para su funcionamiento (varios PCs), pero sin llegar a necesitar grandes instalaciones específicas. Se han utilizado dentro de un rango amplio de problemas,

[1] agentes CBR, es decir, aquellos que gestionan una base de casos

[2] www.robocup.org/

como son por ejemplo la mejora de las búsquedas web como en [9] o la recomendación de referencias bibliográficas cooperativa como en [33].

- **Grandes:** Se trata de sistemas CBR distribuidos con un número muy grande de bases de casos, requiriendo grandes infraestructuras. Pueden estar ligados a redes sociales, en las que la base de casos de cada usuario está gestionada por un agente, como es el caso de [15] que realiza recomendaciones de música para una radio por internet basándose en las preferencias y gustos de los usuarios dentro de una red social o [13] que realiza recomendaciones de películas.

El número de agentes que constituyen un sistema CBR distribuido influye de forma directa en los siguientes aspectos:

- **Topología de red:** El número de agentes constituye una limitación a la hora de elegir determinadas topologías de red. Por ejemplo, la elección de una topología de red "todos con todos" con un sistemas CBR distribuido de tamaño grande resulta ineficiente, debido al gran número de enlaces existentes y a la sobrecarga de red introducida, así como el coste en tiempo de razonamiento entre los agentes. Por otra parte la elección de topologías de "árbol" en sistemas pequeños puede carecer de sentido puesto que lo que se busca es que los agentes realicen deliberaciones en conjunto, puesto que el razonamiento de varios grupos aislados en sistemas pequeños puede desvirtuar las soluciones, siendo más adecuada por lo tanto la topología "todos con todos".

- **Composición de los resultados:** La composición de los resultados se ve influida por el número de agentes que formen el sistema CBR distribuido. Para sistemas grandes puede resultar interesante escoger únicamente la mejor de las soluciones individuales de cada uno de los agentes involucrados en el sistema, mientras que para sistemas pequeños podremos escoger quedarnos con un mayor número de soluciones individuales de cada uno de los agentes para aportar una mayor diversidad en la solución común. Para sistemas de tamaño medio, deberemos buscar un compromiso entre las dos soluciones anteriores.

3.2.2. Parámetros de las bases de casos

Las bases de casos que manejan cada uno de los agentes CBR constituyen la fuente de conocimiento de cada uno de ellos, y por tanto sus características resultan fundamentales en el comportamiento que tendrán estos agentes dentro del sistema distribuido. Las tres variables de diseño dentro de las bases de casos son las siguientes:

Tamaño de las bases de casos

Puede resultar difícil de antemano fijar el tamaño que tendrán las bases de casos de cada uno de los agentes de un sistema distribuido. Además, presumiblemente

este tamaño se incrementará a lo largo del tiempo con la inclusión de nuevos casos dentro de la base de casos de cada agente. A pesar de ello, es recomendable tener en cuenta el tamaño esperado que van a tener estas. Podemos distinguir tres posibilidades:

- **Pequeñas:** Los tiempos y recursos para su procesamiento son asumibles y no requieren técnicas específicas. Se han utilizado en sistemas como por ejemplo la mejora de las búsquedas web [9], la recomendación de películas [14] o música [15] o destinados a herramientas de ayuda al aprendizaje como en [11].
- **Grandes:** Requieren mecanismos específicos para su gestión de forma eficiente. Se trata de bases de casos construidas a priori y que son utilizadas por estos sistemas para sus propósitos como por ejemplo en [2] para la clasificación de especies vegetales.

Aunque el tamaño de las bases de casos utilizadas no afecte de manera relevante a otras características de diseño de los sistemas distribuidos, resulta fundamental en otros parámetros, principalmente el tiempo de recuperación de casos ante una consulta.

Solapamiento de las bases de casos

Otro parámetro dentro del diseño de las bases de casos de los agentes de un sistema distribuido es si un mismo caso puede presentarse de forma simultánea en dos bases de casos distintas. De esta forma, existen tres alternativas:

- **No solapadas:** Se trata de bases de casos que no comparten ningún caso entre ellas. Los sistemas CBR distribuidos que utilizan este tipo de bases de casos han sido utilizados, por ejemplo, para la gestión de bases de casos modularizadas en [25] o en [26] para la gestión del conocimiento o para la clasificación de especies vegetales en [2].
- **Solapadas:** Las bases de casos solapadas son aquellas que pueden compartir algún caso, pudiendo compartir incluso todos los casos contenidos en ellas. Son utilizadas, por ejemplo, en recomendadores distribuidos vinculados a redes sociales como los expuestos en [15, 13, 14].
- **Iguales:** Las bases de casos iguales son aquellas que contienen los mismos problemas, pudiendo variar la solución de los mismos. Se trata de sistemas en los que cada uno de los agentes tiene una solución distinta ante cada uno de los problemas que poseen, siendo fundamental el proceso de argumentación a la hora de conseguir llegar a una solución conjunta ante un problema planteado.

La utilización de los distintos tipos de solapamientos de bases de casos influyen en otros factores dentro de los sistemas distribuidos como:

- **Composición de los resultados:** El mecanismo de composición de resultados debe tener en cuenta que un mismo caso puede ser devuelto como solución por dos agentes distintos del sistema en el caso de estar utilizando bases de casos solapadas o iguales.
- **Protocolo de argumentación:** De igual forma, el protocolo de argumentación deberá considerar que, dentro de una argumentación entre agentes del sistema, varios de ellos estén utilizando como argumentaciones casos iguales.

Tipos de casos utilizados

Tanto la estructura como la semántica de la información contenida en los casos de cada uno de los agentes del sistema resulta muy importante para otros parámetros del sistema distribuido. En función de si los casos utilizados por los agentes son iguales en cuanto a su forma y su semántica, podemos realizar la siguiente clasificación:

- **Casos homogéneos:** Todos los casos de los diversos agentes del sistema tienen la misma estructura y semántica. La mayoría de los sistemas estudiados en la bibliografía utilizan casos homogéneos.
- **Casos heterogéneos:** Se trata de sistemas en los que el formato de los casos puede diferir de un agente a otro. En este caso, se necesitará un mecanismo de traducción durante los procesos de composición de resultados o de argumentación.

El tipo de los casos utilizados, ya sean homogéneos o heterogéneos influye en los siguientes aspectos del sistema distribuido:

- **Protocolo de argumentación:** La semántica de los casos utilizada, así como la estructura de los mismos, impone restricciones a la hora de utilizar un determinado protocolo de argumentación. Por ejemplo, el protocolo de argumentación AMAL descrito en [2] impone la restricción de que los casos tratados por el sistema deben estar descritos mediante el uso de Lógicas Descriptivas [17].
- **Tipo de razonamiento CBR:** El formato de los casos tratados por cada uno de los agentes puede imponer restricciones en la forma en la que los casos son recuperados de cada una de las bases de casos individuales de los agentes.

3.2.3. Topología de red

La organización existente entre los agentes de un sistema distribuido constituye otro de los parámetros de estos sistemas. Las topologías fundamentales que podemos encontrar dentro de los sistemas CBR distribuidos son:

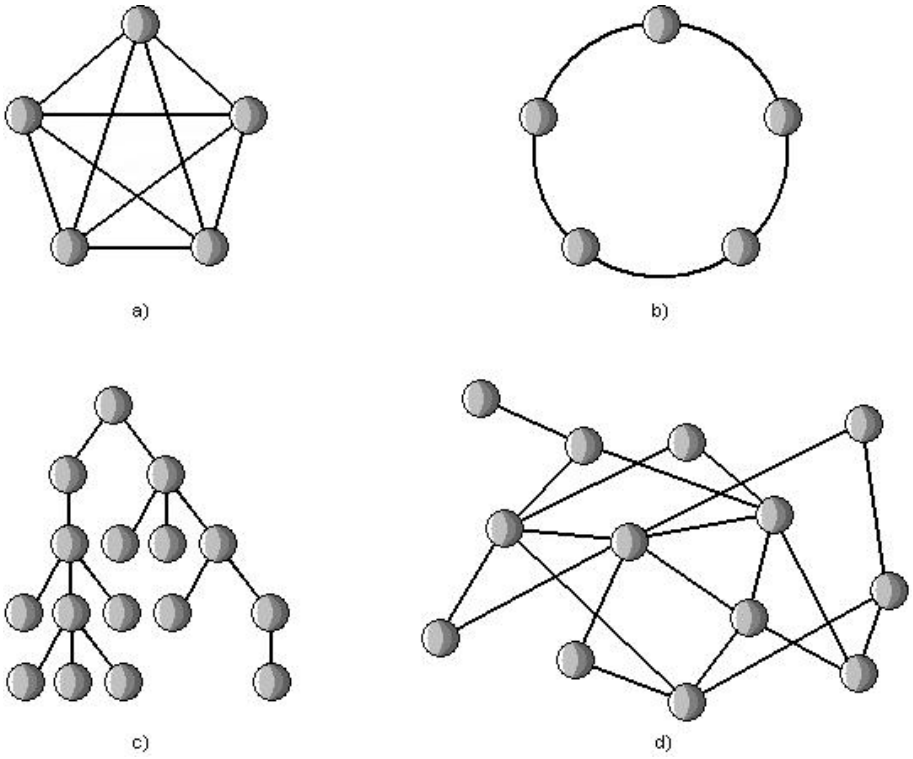

Figura 3.2: Topologías de red posibles para un sistema CBR distribuido

- **Topología "todos con todos":** Se trata de aquella topología en la que cada uno de los agentes se encuentra conectado con el resto de agentes que forman el sistema distribuido. Se utiliza con sistemas en los que el número de agentes es pequeño debido a la carga de red y de procesamiento que requiere, como por ejemplo en [2], donde se utiliza con sistemas de 2, 3, 4 y 5 agentes respectivamente. Corresponde con la Figura 3.2 a).

- **Topología de anillo:** Se trata de una topología en la que cada uno de los agentes del sistema únicamente está conectado a otros dos individuos, siguiendo el patrón de la Figura 3.2 b). Se puede utilizar en sistemas CBR distribuidos en los cuáles resulta interesante que los procesos de argumentación se hagan localmente, participando en cada uno un número pequeño de agentes y existiendo por tanto un número alto de argumentaciones, al contrario de la topología "todos con todos" en la que únicamente existe un proceso de argumentación en el que participaban todos los agentes del sistema.

- **Topología de árbol:** Corresponde con la Figura 3.2 c). Los agentes del sistema CBR distribuido se conectan siguiendo una estructura de árbol en la que, para cada uno de los agentes, podemos identificar su nodo padre (tiene uno y

solamente uno) y sus nodos hijos (puede tener varios o ninguno). Se ha utilizado, por ejemplo, en sistemas para la recuperación de información en bases de casos modularizadas en función de las áreas de conocimiento [25, 26], en las que los agentes de las capas superiores representan áreas de conocimiento más generales, y según vamos profundizando en el árbol, los agentes gestionan bases de casos referentes a áreas concretas del conocimiento.

- **Topología de red social:** Se trata de sistemas CBR distribuidos en los que la estructura de red se ha importado de la red social a la que representa. Podemos ver un ejemplo en la Figura 3.2 d). Cada uno de los agentes del sistema se encuentra conectado con aquellos agentes que representan a usuarios afines. Son utilizadas sobre todo, en el campo de los recomendadores [15, 13, 14], en la mejora de las búsquedas web [9] o para representar grupos afines en entornos laborales [33].

La influencia de la topología de red utilizada tiene una especial importancia dentro de:

- **Número de agentes del sistema:** La topología de red supone una restricción como se ha comentado en la Sección 3.2.1, dado que no es posible utilizar un número elevado de agentes con una topología, por ejemplo, "todos con todos". De esta forma, tenemos que tener en consideración la topología de red que deseamos utilizar a la hora de fijar el número de agentes del sistema CBR distribuido.

- **Composición de los resultados:** El mecanismo de composición de los resultados que estemos eligiendo variará en función del tipo de topología de red utilizada por el sistema distribuido. De esta forma, no se necesitará ningún proceso de composición por ejemplo para la topología de red "todos con todos", mientras que sin embargo, si lo necesitaremos para las restantes topologías.

3.2.4. Modelo de confianza

El modelo de confianza escogido dentro del sistema CBR distribuido determinará la confianza depositada por el sistema dentro de cada uno de los agentes que forman parte del mismo.

La forma de construcción de un modelo de confianza entre agentes puede ser:

- **Explícita:** Mediante la indicación de forma explícita de la medida de confianza entre dos agentes del sistema. Este método se utiliza cuando se conoce a priori la confianza existente entre los agentes del sistema. Es utilizado en sistemas como el descrito en [14].

- **Implícita:** La confianza entre los agentes no se indica de manera explícita al crear el sistema, sino que es calculada por este mediante algún mecanismo,

habitualmente basado en el coeficiente de correlación de Pearson [34], como los expuestos en [30, 28].

Además de la forma de construcción del modelo de confianza dentro de un sistema CBR distribuido, una parte fundamental del mismo es el mecanismo de gestión de los modelos de confianza. Existen dos posibilidades:

- **Centralizada:** Existen un único modelo de confianza para cada uno de los agentes del sistema distribuido, gestionado de forma centralizada por el mismo. De esta forma, cuando un agente desee conocer la confianza que tiene en otro de los agentes, deberá preguntar al sistema distribuido, obteniendo un valor que será el mismo, independientemente del agente que realice la pregunta, es decir, cada agente del sistema tiene un único valor de confianza. Esta alternativa es utilizada por sistemas como en [29, 13, 14, 2, 15].

- **Descentralizada:** Se trata de un método de gestión de la confianza en la que cada agente es responsable de gestionar la confianza que deposita en cada uno de los agentes a los que está enlazado. De esta forma, existen varios modelos de confianza para cada uno de los agentes del sistema CBR distribuido, tantos como agentes haya enlazados a ellos. Esta alternativa es utilizada por ejemplo en [27].

La forma de construcción del modelo de confianza así como el mecanismo de gestión de ésta, no interfieren con los otros parámetros de forma importante, si bien resultan claves en la solución final del sistema, puesto que van a marcar en que grado se confía en cada uno de los agentes, y por tanto, la solución estará sujeta a esta confianza.

3.2.5. Composición de resultados

El método de composición de resultados de un sistema distribuido es el mecanismo mediante el cual se integran las soluciones individuales de los distintos agentes dentro de un sistema CBR distribuido junto con las soluciones obtenidas de los proceso de argumentación.

Este mecanismo resulta fundamental a la hora de encontrar una solución adecuada a una consulta determinada al sistema distribuido, debido a que establece cómo las soluciones locales de cada uno de los agentes del sistema se integran dentro de la solución global. Además resulta clave en otras características de los sistemas CBR como la diversidad o la precisión, comentadas dentro del Capítulo 2.

El mecanismo básico de la composición de resultados consiste en obtener las n soluciones locales de los agentes enlazados en un punto de la red y hacerlos argumentar hasta obtener n resultados comunes. Estos n resultados comunes podrán volver a ser utilizados como soluciones locales dentro de otro proceso argumentativo en un nivel superior de la red o ser mezcladas con las soluciones locales de otros de los agentes mediante la realización de un ranking, en base a su ajuste a

la consulta a la que responden. Los distintos métodos de composición de resultados se basan, fundamentalmente, en la modificación del número de casos que se componen de cada agente.

El mecanismo de composición de resultados está influido por:

- **Número de agentes del sistema:** El número de agentes debe ser tenido en cuenta por el mecanismo de composición de los resultados para contemplar factores como la diversidad y la precisión de las soluciones. Cuantas más soluciones locales incluyamos en el proceso de composición de los resultados, probablemente aumentaremos la diversidad de las soluciones a costa de sacrificar la precisión de las mismas, mientras que si incluimos pocas soluciones locales estamos disminuyendo la diversidad y posiblemente aumentando la precisión al quedarnos con las mejores de ellas.

- **Solapamiento de las bases de casos:** Resulta fundamental durante el proceso de composición de resultados contemplar la posibilidad de que un mismo caso pueda repetirse en varias bases de casos. Durante el proceso de composición deberemos tener en cuenta esta posibilidad y aportar los mecanismos apropiados para evitar la repetición de soluciones si así lo deseamos.

- **Topología de red:** Influye en el proceso de composición determinando las soluciones de qué agentes son incluidas en la composición. Las distintas topologías darán lugar a métodos de composición diferentes que puedan ser más adecuados a cada una de ellas.

- **Protocolo de argumentación:** El número de soluciones propuestas tras el protocolo de argumentación, así como el número de casos que proporciona cada agente, debe ser tenido en cuenta por el protocolo de argumentación para devolver el número apropiado de casos.

3.2.6. Protocolo de argumentación

El protocolo de argumentación de un sistema CBR distribuido determinará la forma en la que un grupo de agentes alcanzan una solución común ante una consulta.

Resulta complicado realizar una clasificación de los distintos protocolos de argumentación existentes dentro de los sistemas CBR distribuidos. En este caso, hemos optado por realizar una clasificación basándonos en el papel que juegan los agentes dentro del protocolo de argumentación, de forma que nos encontramos dos alternativas:

- **Agentes homogéneos:** Se trata de aquellos protocolos de argumentación dentro de los cuáles todos los agentes que participan en el mismo están al mismo nivel y juegan el mismo papel, es decir, no existe ningún agente destacado o director del protocolo de argumentación. Este es el caso del protocolo AMAL descrito en [2]. Suele tratarse de protocolos asociados con

topologías de red en las que no existe ningún tipo de orden jerárquico y todos los agentes se encuentran al mismo nivel.

- **Agentes heterogéneos:** Se trata de aquellos protocolos de argumentación dentro de los cuáles los agentes pueden desarrollar papeles diferenciados, es decir, puede existir uno o varios agentes encargados de la organización del protocolo de argumentación mientras que existen otros que aportan soluciones y argumentaciones. Este tipo de protocolos suelen estar asociados a topologías de red jerárquicas con el caso de las topologías de árbol, como es el caso de los sistemas [25, 26] o asociados a redes sociales.

El protocolo de argumentación está influido por otras características como:

- **Solapamiento de las bases de casos:** En el caso de que las bases de casos del sistema puedan ser solapadas o incluso iguales, debemos tener en cuenta que varios agentes pueden utilizar casos similares como soluciones o argumentaciones dentro del proceso de argumentación. Además deberemos considerar si el proceso de argumentación puede generar como soluciones casos similares o si bien por el contrario, todas las soluciones aportadas por un proceso argumentativo deben ser distintas.

- **Tipo de casos:** Tanto la estructura como la semántica utilizada para describir los casos utilizados por el sistema influyen en el protocolo de argumentación de forma que fijan el tipo de razonamiento que este puede realizar sobre las distintas soluciones y argumentaciones aportadas por los diferentes agentes

- **Composición de los resultados:** El protocolo de argumentación deberá tener en cuenta el número de soluciones aportadas por cada uno de los agentes, así como el número de respuestas que debe generar, determinado por el proceso de composición de resultados que utilice el sistema CBR distribuido.

3.2.7. Tipo de razonamiento CBR

El tipo de razonamiento CBR consiste en el tipo de recuperación y adaptación que cada agente realiza en su sistema CBR. Distinguimos dos posibilidades:

- **Razonamiento CBR homogéneo:** Se trata de aquél sistema CBR distribuido en el que todos sus agentes utilizan el mismo sistema de razonamiento CBR, por lo que dos agentes devolverían las mismas soluciones si utilizaran la misma base de casos. La mayoría de los sistemas de la literatura utilizan esta aproximación.

- **Razonamiento CBR heterogéneo:** Se trata de aquél sistema CBR distribuido en el que cada uno de sus agentes puede utilizar un sistema de razonamiento CBR distinto. En este caso, aunque dos agentes del sistema utilicen bases de casos similares, no tienen porque devolver la misma solución ante la misma consulta.

El tipo de razonamiento CBR utilizado no va a influir en los otros parámetros del sistema CBR distribuido expuestos en este capítulo, pero resulta claro que la utilización de un sistema u otro de razonamiento CBR va a influir en las respuestas devueltas por el sistema, así como en su diversidad y precisión.

3.2.8. Iniciación de las consultas

El modo en el que las consultas son planteadas al sistema distribuido admite dos posibilidades:

- **Iniciación única:** Se trata de aquellos sistemas CBR distribuidos en los que las consultas únicamente pueden ser planteadas a uno de los agentes del sistema y este se encargará de resolverla de la forma que considere más adecuada. Es utilizado en sistemas CBR distribuidos en los que no existe una topología de red asociada por ejemplo, a una estructura de usuarios, por lo que únicamente se habilita un punto de entrada al sistema para las consultas. Este tipo de iniciación ha sido utilizada en los sistemas descritos en [2, 25, 26].

- **Iniciación múltiple:** Se trata de aquellos sistemas CBR distribuidos en los que las consultas pueden ser planteadas a cualquiera de los agentes del sistema, el cuál se encargará de resolverla de una forma adecuada. Es especialmente indicada para aquellos sistemas que utilicen, por ejemplo, una red social como topología de red en la que existan varios usuarios que puedan realizar consultas en varios puntos del sistema distribuido. En la literatura, esta iniciación de las consultas es utilizada en los sistemas descritos en [9, 33, 15, 13, 14, 11, 29]

El método de iniciación de las consultas elegido influirá en el mecanismo de propagación de las consultas elegido para el sistema CBR distribuido, puesto que debe considerar si las consultas pueden ser iniciadas en uno o en varios puntos de la topología de red y por tanto, saber gestionar estas consultas.

3.2.9. Propagación de las consultas

Existen dos vías distintas para la propagación de las consultas en los sistemas CBR distribuidos:

- **Propagación completa:** En este tipo de propagación cuando uno de los agentes del sistema distribuye una consulta lo hace a todos los agentes enlazados a él por la topología de red. Se utiliza en los sistemas descritos en [13, 14, 2, 15]

- **Propagación parcial:** En este tipo de propagación los agentes seleccionan el subconjunto de agentes a los que propagar la consulta. Este subconjunto suele estar constituido por los agentes competentes para resolver el problema planteado por la consulta. Esta metodología es utilizada en [33].

El mecanismo de propagación de las consultas influye en:

- **Composición resultados:** Dado que el número de agentes a los que va a propagarse la consulta puede ser variable, debe ser tenido en cuenta a la hora de seleccionar el modo de composición de resultados.
- **Protocolo de argumentación:** De igual forma que para la composición de resultados, el protocolo de argumentación tiene que tener en cuenta que es posible que no todos los agentes enlazados al que propaga la consulta vayan a participar en el protocolo de argumentación.

A continuación, a modo de resumen aportamos una tabla (Figura 3.3) que incluye las características anteriores de los sistemas CBR distribuidos, las distintas características o parámetros del sistema a los que afecta cada una de ellas, las diferentes posibilidades existentes para su implementación y una referencia a sistemas de la bibliografía que utilizan cada una de las distintas alternativas.

3.3. Conclusiones

En este capítulo hemos expuesto un marco teórico de integración de sistemas CBR distribuidos que permite la caracterización de los sistemas existentes en la literatura así como aquellos nuevos sistemas CBR que podamos construir.

En él, hemos identificado las principales características de los sistemas CBR distribuidos, las hemos analizado exponiendo cada una de sus principales alternativas y hemos analizado las consecuencias de la utilización de cada una de ellas sobre el sistema. Con ello, hemos construido una herramienta de referencia básica tanto a la hora de diseñar sistemas CBR distribuidos como a la hora de caracterizarlos y analizarlos, facilitando su estudio y compresión.

Por último, hemos aportado una tabla de resumen de las características descritas durante el desarrollo del capítulo.

En el capítulo siguiente trataremos la construcción de un marco práctico que extienda las funcionalidades de jCOLIBRI y que permita la construcción de cualquier sistema CBR caracterizado con el marco teórico descrito en este capítulo.

Característica	Afecta a	Posibilidades	Utilizado en
Número de agentes	Topología de red Composición de los resultados	Pequeño Mediano Grande	[2, 32] [9, 33] [15, 13]
Tamaño de la base de casos	Recuperación de los casos	Pequeña Grande	[9, 14, 15, 11] [2]
Solapamiento de las bases de casos	Composición de los resultados Protocolo de argumentación	No solapadas Solapadas Iguales	[25, 26, 2] [15, 13, 14]
Tipo de casos de las base de casos	Protocolo de argumentación Tipo de razonamiento CBR	Casos homogéneos Casos heterogéneos	
Topología de red	Número de agentes Composición de los resultados	Todos con todos Anillo Árbol Red social	[2] [25, 26] [15, 13, 14, 9, 33]
Construcción del modelo de confianza	Solución del sistema	Explícita Implícita	[14] [30, 28]
Gestión del modelo de confianza	Solución del sistema	Centralizada Descentralizada	[29, 13, 14, 15, 2] [27]
Composición de los resultados	Número de agentes Solapamiento de las bases de casos Topología de red Protocolo de argumentación	n casos de cada agente	
Protocolo de argumentación	Solapamiento de las bases de casos Tipo de casos de las bases de casos Composición de los resultados	Agentes homogéneos Agentes heterogéneos	[2] [25, 26]
Tipo de razonamiento CBR	Solución del sistema	Razonamiento CBR homogéneo Razonamiento CBR heterogéneo	
Iniciación de las consultas	Propagación de las consultas	Iniciación única Iniciación múltiple	[2, 25, 26] [9, 33, 15, 13, 14, 11, 29]
Propagación de las consultas	Composición de los resultados Protocolo de argumentación	Propagación completa Propagación parcial	[13, 14, 2, 15] [33]

Figura 3.3: Resumen de las características de los sistemas CBR distribuidos

Capítulo 4

Marco práctico para sistemas CBR distribuidos: Extensión de jCOLIBRI

La construcción de cualquiera de los sistemas CBR distribuidos expuestos en el capítulo anterior requiere de una plataforma que facilite su diseño y de soporte a su funcionalidad. Ante esta premisa, surge la necesidad de construir un marco para la implementación de este tipo de sistemas.

Para ello, partimos de todo el trabajo realizado en jCOLIBRI 2 en la creación de sistemas CBR monolíticos o tradicionales, es decir, aquellos cuya base de casos se encuentra concentrada en un único lugar. Gracias a jCOLIBRI 2, podemos crear los sistemas CBR individuales que usarán cada uno de los individuos del sistema CBR distribuido con su propia base de casos para resolver las consultas que les sean planteadas.

El objetivo de este capítulo será extender las funcionalidades contenidas en jCOLIBRI 2, creando una capa de abstracción que permita la construcción de cualquier sistema CBR distribuido descrito anteriormente, en el cuadro resumen de la Figura 3.3.

4.1. Diseño del marco práctico

El diseño del sistema multiagente para la construcción de sistemas CBR distribuidos en jCOLIBRI 2 se basa en una versión simplificada del estándar para arquitecturas multiagente de FIPA[1].

FIPA (Foundation for Intelligent Physical Agents) es una organización perteneciente a la comunidad informática del IEEE que promueve la tecnología basada en agentes mediante la creación de estándares que unifiquen los sistemas multiagentes y mejoren su calidad así como la interoperabilidad con otros sistemas.

[1] http://www.fipa.org/, estándar SC00001L

El estándar de FIPA define la arquitectura abstracta de un sistema multiagente, de forma independiente del lenguaje de implementación, fomentando la interoperabilidad y la reusabilidad. En nuestro caso, hemos decidido utilizar una versión reducida de esta arquitectura abstracta, de forma que mantenemos los beneficios de interoperabilidad y reusabilidad definidos por el estándar FIPA, aliviando algunas de las cargas de diseño.

El marco práctico de integración de sistemas CBR distribuidos diseñado consiste en una serie de interfaces que definen el comportamiento de los elementos básicos de cualquier sistema CBR distribuido multiagente y a partir del cual se pueda crear cualquiera de los sistemas caracterizados en el Capítulo 3 y en el cuadro de resumen de la Figura 3.3.

Este marco de alto nivel será pues independiente de la implementación y de la arquitectura, así como de la comunicación utilizada por los agentes que compongan el sistema distribuido.

4.1.1. Entidades

El marco práctico de integración de sistemas CBR distribuidos está formado por 6 entidades básicas, que exponemos a continuación:

AgentDirectory: El papel fundamental del *AgentDirectory* consiste en proporcionar un lugar donde un agente puede registrarse para que su ubicación sea conocida por el resto de los agentes del sistema multiagente. Cuando una de las entidades del sistema desee conocer la localización de uno de los agentes del sistema, deberá acudir al AgentDirectory, que será la entidad que le proporcione esta información. De esta forma, el AgentDirectory almacenará tuplas *<AID, localizacion>* que le servirán para mantener un registro de todos los agentes del sistema multiagente. Además, dentro del sistema CBR distribuido almacenará la información relativa a los enlaces de topología de red implementada por el sistema.

El *AgentDirectory* dispondrá de su propio *AgentIdentifier* (*AID*) puesto que se trata de un agente más del sistema y por lo tanto debe identificarse de forma única. Además hará uso del *MessageTransporter* para responder a los envíos de las peticiones que se realicen. Como ya hemos comentando, almacenará una lista de tuplas *<AID, localizacion>* como registro de los agentes del sistema y una lista de conexiones entre agentes de la forma *<AID, AID>*.

Las operaciones que a las que deberá dar soporte el *AgentDirectory* son:

- Registrar a un agente dentro del sistema multiagente.
- Eliminar a un agente del sistema multiagente.
- Establecer una conexión de red entre dos agentes previamente registrados en el sistema.
- Eliminar una conexión de red entre dos agentes previamente creada.

- Devolver las conexiones de red de uno de los agentes del sistema.

AgentDirectoryDelegate: El papel del *AgentDirectoryDelegate* dentro del sistema CBR distribuido es el de facilitar a los *CBRAgent* del sistema el acceso a la información contenida en el AgentDirectory relativa a la topología de red del sistema, es decir, les permite a los *CBRAgent* conocer a qué otros *CBRAgent* están enlazados.

La entidad *AgentDirectoryDelegate* será usada por los *CBRAgent* del sistema para acceder al *AgentDirectory*. Por lo tanto, cada uno de los *CBRAgent* contendrán un *AgentDirectoryDelegate* que será el encagardo de acceder a la información contenida en el *AgentDirectory*.

La operación que deberá realizar el *AgentDirectoryDelegate* es:

- Devolver los agentes conectados al agente que esté haciendo uso de él.

AID: La entidad *AID* o *AgentID* tiene como objetivo poder identificar a un agente del sistema multiagente de forma inequívoca. Dado que representa de forma inequívoca a uno de los agentes del sistema, va a permanecer invariable durante todo el ciclo de vida del agente. Esta identificación inequívoca puede tratarse, por ejemplo, de un identificador universal único o UUID.

La entidad *AID* no utiliza ninguna de las otras entidades dentro del marco práctico, pero juega un papel fundamental para hacer posible la identificación de los agentes en el sistema. Por ello, cada uno de los *CBRAgent* contiene una identificación *AID* y la entidad *AgentDirectory* contiene la lista de <*AID, localizacion*> con las localizaciones asociadas a cada identificación. Esto se debe a que los *CBRAgent* se identificarán unos a otros por el identificador único contenido en cada *AID*, y no por la entidad completa, que además posee una localización.

El objetivo de esto es, que si el agente cambia su localización durante la ejecución del sistema, no sean necesarias actualizar todas las referencias existentes al *AID*, sino que baste con modificar la existente en el *AgentDirectory* y la propia existente en el *AID* del agente.

Por tanto, las operaciones a las que debe dar soporte la entidad *AID* son:

- Devolver el identificador único del agente que utilice esta entidad.

- Devolver la localización actual del agente que utilice esta entidad.

CBRAgent: Un agente es un proceso computacional de una aplicación, autónomo y con capacidad de comunicarse. En el caso de los agentes CBR, se trata de agentes software que tienen a su cargo la gestión de un sistema CBR, por lo que cada agente CBR debe poseer las herramientas necesarias para el manejo de una aplicación CBR estándar de jCOLIBRI.

Cada uno de los *CBRAgent* del sistema multiagente estará asociado a un *AID* que lo identificará y distinguirá del resto de los agentes del sistema. Además, dispondrá de la ayuda de un *AgentDirectoryDelegate* para poder acceder al *AgentDirectory* y conocer los agentes a los que se encuentra enlazado dentro del sistema CBR distribuido. Para el envió y recepción de mensajes con el resto de los agentes del sistema multiagente utilizará un *MessageTransporter*.

Las funcionalidades básicas que debe implementar la entidad *CBRAgent* son:

- Proporcionar acceso a cada uno de los elementos que lo componen como, su *AID*, su *AgentDirectoryDelegate* o el *MessageTransporter* que utiliza.
- Permitir su activación y desactivación.

Message: Un *Message* es una unidad individual de comunicación entre dos o más agentes. Un *Message* incluye información acerca del tipo de comunicación del que forma parte, los *AID* de los agentes emisor y receptor y el propio contenido del mensage. Dependiendo del sistema distribuido concreto, existirán diversas implementaciones distintas de *Message*.

Es utilizado por los *CBRAgent* cuando necesitan realizar algún tipo de comunicación con el resto de los agentes del sistema, y es la entidad básica de envío y recepción del *MessageTransporter*.

Debe dar soporte a operaciones de:

- Acceso al emisor y receptor del mensaje a través de sus *AID*.
- Acceso al lenguaje y al tipo de codificación que utiliza en su contenido.
- Acceso al contenido que transmite.
- Acceso al tipo de mensaje que implementa.

MessageTransporter: La entidad *MessageTransporter* representa un servicio del sistema multiagente. Esta entidad da soporte al envío y recepción de mensajes entre agentes del sistema. La implementación concreta de esta entidad depende del sistema distribuido que creemos, así como del mecanismo de envío de mensajes que utilicemos como, por ejemplo, los *sockets*, *TCP/IP* o la *memoria compartida*.

La entidad *MessageTransporter* es utilizada por los *CBRAgent* cuando desean realizar el envío de algún *Message* a otros agentes del sistemas. El *MessageTransporter* transmitirá el contenido de cada uno de estos mensajes entre los distintos agentes del sistema y se lo hará llegar en forma de un nuevo mensaje.

Debe tener soporte para las siguientes operaciones:

- Realizar el envío de un mensaje.
- Recibir un mensaje.

4.1.2. Relaciones entre las entidades

Para completar la información acerca del marco práctico diseñado y mejorar su comprensión, en esta sección se muestran las relaciones que se establecen entre cada una de las entidades que lo forman. En primer lugar mostraremos un diagrama UML simplificado de las relaciones globales de los elementos del marco práctico, pasando después a mostrar los diagramas UML concretos para cada una de las entidades que lo forman.

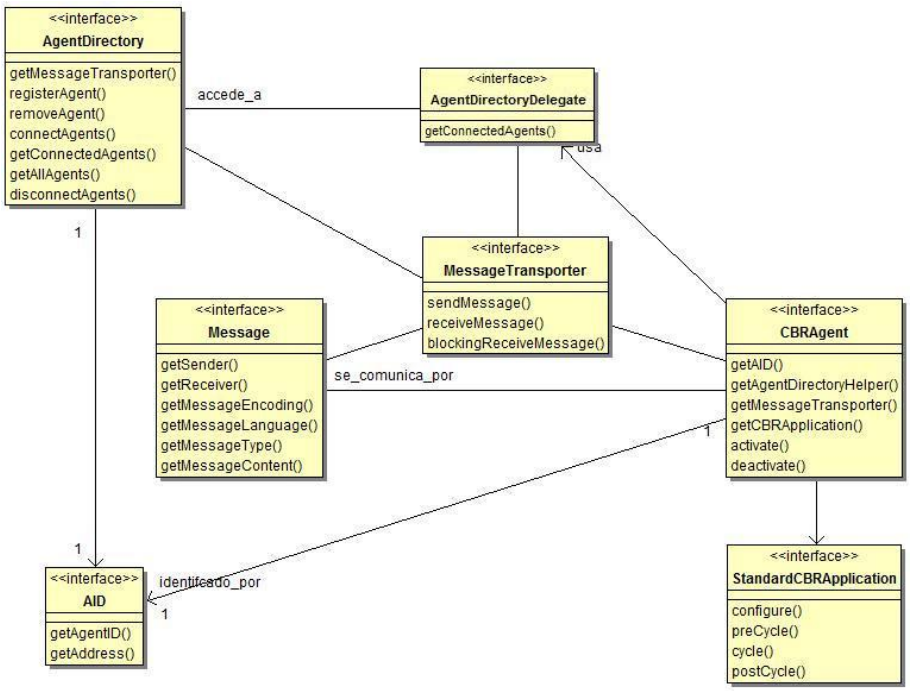

Figura 4.1: Diagrama UML global del marco práctico

Visión global: El diagrama UML simplificado de la interacción de todos los elementos que forman este marco práctico está expuesto en la Figura 4.1.

El centro del marco práctico diseñado lo constituye el *CBRAgent*, el cuál gestiona su sistema CBR a través de la entidad *StandardCBRApplication*.

Cada uno de los *CBRAgent* del sistema se comunica con los demás a través de *Message*, enviados y recibidos a través del *MessageTransporter*. Gracias al sistema de comunicación entre los agentes, se posibilita el intercambio de información entre ellos y por tanto los protocolos de argumentación que permiten llegar a soluciones conjuntas a un problema.

Todos los agentes del sistema, tanto los *CBRAgent* como el *AgentDirectory*, están representados mediante la entidad *AID*. Gracias a esta entidad, cada uno de ellos está identificado de manera única frente al resto. Además, la entidad *AID* contiene información acerca de la localización de los agentes que la poseen.

Por último, la entidad *AgentDirectoryDelegate* facilita el acceso del *CBRAgent* a la información contenida en el *AgentDirectory*. Será a través del *MessageTransporter* como el *AgentDirectoryDelegate* acceda a dicha información.

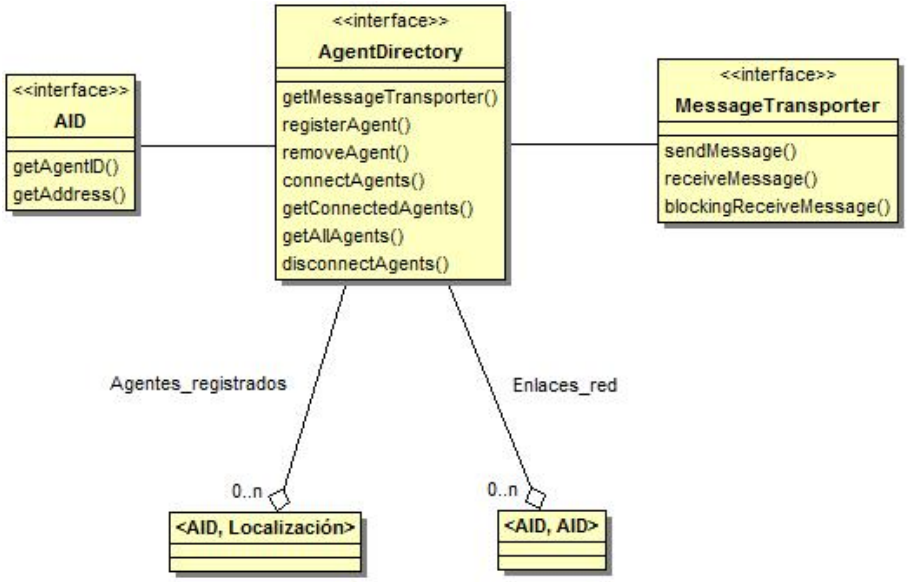

Figura 4.2: Diagrama UML de la entidad *AgentDirectory*

AgentDirectory: El *AgentDirectory* (Figura 4.2) está formado por la agregación de dos elementos:

- El conjunto de las tuplas *<AID, localizacion>* que representan el conjunto de agentes registrados en el sistema junto con sus localizaciones. Como podemos ver en la Figura, el *AgentDirectory* puede tener desde 0 de estas tuplas, si el sistema multiagente inicia su ejecución sin ningún agente registrado, hasta n agentes, los agentes registrados en cada momento en el sistema.

- El conjunto de las tuplas *<AID, AID>* que representan el conjunto de los agentes enlazados en el sistema. Cada una de estas tuplas representa una conexión entre dos agentes del sistema CBR distribuido y significa que ambos colaborarán entre ellos intercambiando información. En el momento inicial del sistema distribuido esta lista se encontrará vacía y se irá modificando a lo largo e la vida del sistema según se vayan generando y destruyendo estas conexiones.

Además, la entidad *AgentDirectory* utiliza:

- Un elemento *AID* para representarse de forma inequívoca dentro del sistema distribuido.

- Un *MessageTransporter* para poder comunicarse con los distintos elementos del sistema distribuido y atender a sus solicitudes.

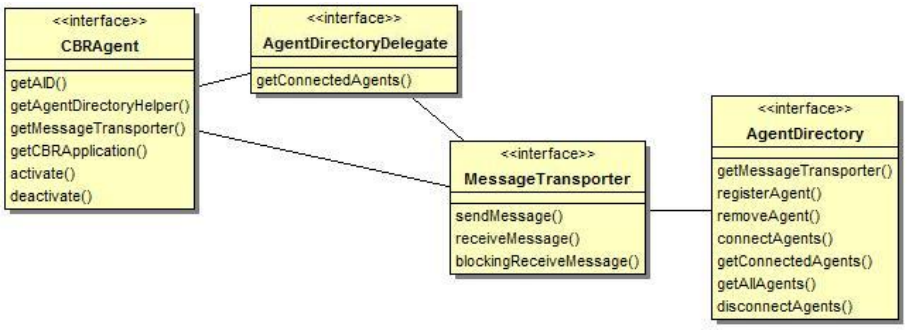

Figura 4.3: Diagrama UML de la entidad *AgentDirectoryDelegate*

AgentDirectoryDelegate: Podemos observar el diagrama UML de la entidad *AgentDirectoryDelegate* en la Figura 4.3.

Como podemos ver en el diagrama UML, la entidad *AgentDirectoryDelegate* es el mecanismo utilizado por el *CBRAgent* para poder acceder a la información almacenada en el *AgentDirectory* acerca de las conexiones de red del sistema CBR distribuido.

Ante una solicitud del *CBRAgent* el *AgentDirectoryDelegate* deberá responder con la información solicita, accediendo al *AgentDirectory* de la forma adecuada (dependiendo de la implementación concreta del sistema), habitualmente a través del *MessageTransporter* asociado a ese agente.

La entidad *AgentDirectoryDelegate* oculta por tanto la comunicación como la interfaz de acceso al *CBRAgent* a fin de facilitar su uso y servir de puente a la información necesaria acerca de los enlaces existentes entre el agente actual y el resto de los agentes del sistema CBR distribuido.

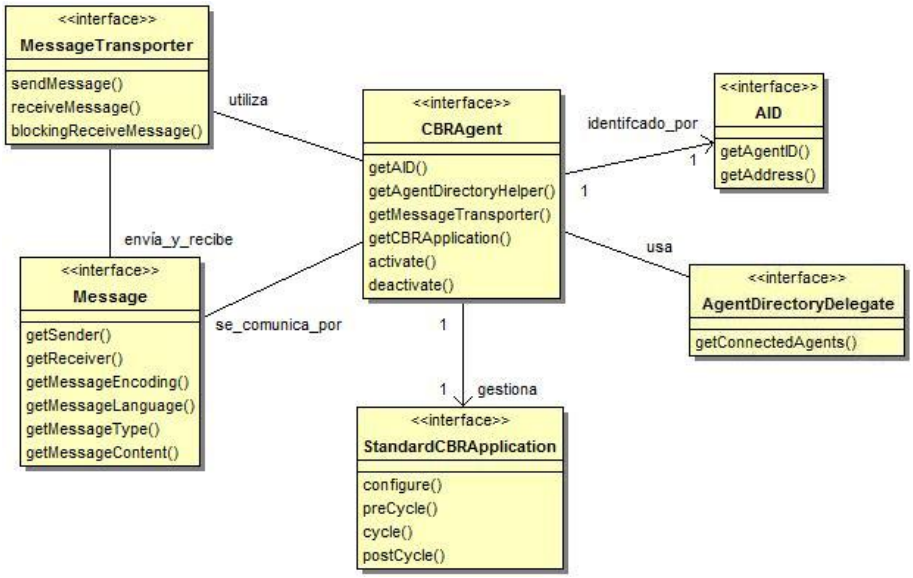

Figura 4.4: Diagrama UML de la entidad *CBRAgent*

CBRAgent: La Figura 4.4 se corresponde con el diagrama UML de la entidad *CBRAgent*.

Las relaciones del *CBRAgent* son uno de los centros del marco práctico diseñado. Como podemos observar en el diagrama, utilizan una entidad *StandardCBRApplication*, que consiste en sistema CBR tradicional o monolítico definido en jCOLIBRI 2 y que cuenta con todos los mecanismos necesarios para el manejo de una aplicación CBR. Gracias a él, el *CBRAgent* gestiona el sistema CBR al que está asignado.

Además, dado que es los *CBRAgent* son agentes dentro del sistema multiagente, tendrán asociados a ellos una entidad *AID*, que será su identificación respecto del resto de los agentes del sistema y además contendrá su localización actualizada en todo momento.

Dispondrán también de una entidad *AgentDirectoryDelegate* que le ayudará a obtener la información necesaria del *AgentDirectory* de forma trasparente cuando sea necesario. Esta información resulta clave para los agentes dado que la necesitan en procesos claves del sistema CBR distribuido como los protocolos de argumentación o los procesos de propagación de las consultas.

Para poder comunicarse con el resto de los agentes del sistema, utilizarán además un *MessageTransporter* que será el encargado del envío y recepción de los *Message* que les harán posibles desarrollar las comunicaciones con los demás agentes del sistema.

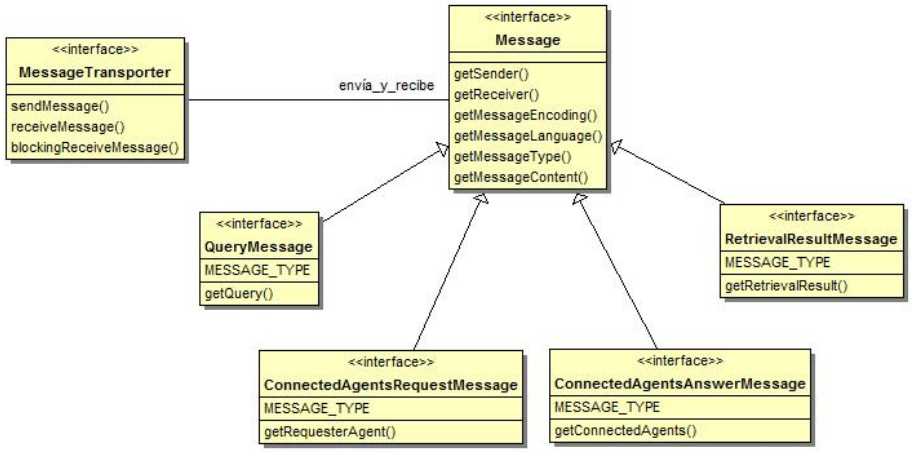

Figura 4.5: Diagrama UML de la entidad *Message*

Message: Podemos observar el diagrama de la entidad *Message* en la Figura 4.5.

La entidad *Message* constituye la unidad de transferencia básica durante la comunicación entre agentes del sistema. Gracias a ella, la comunicación entre los agentes es posible.

Es utilizada como podemos ver en el diagrama por el *MessageTransporter*, entidad encargada del envío y la recepción de *Message*. El mecanismo de envío y recepción llevado a cabo por el *MessageTransporter* es trasparente a la entidad *Message* y completamente dependiente del sistema concreto implementado. Por el contrario, el formato del mensaje enviado es dependiente del tipo de codificación y del lenguaje utilizados, también dependientes de la implementación concreta, pero deben estar especificados en la entidad *Message*.

Además de la relación entre la entidad *Message* y la entidad *MessageTransporter*, en el diagrama UML de la Figura 4.5 podemos observar cuatro entidades que heredan de la entidad *Message*. Estas entidades corresponden con mensajes de comunicación independientes del sistema CBR concreto que se quiera diseñar, por lo que hemos tomado la decisión de incluirlos dentro del marco diseñado. Estas entidades heredadas de *Message* son:

- **QueryMessage:** Este mensaje es el iniciador del proceso CBR. A través de él se envía una consulta CBR a un agente del sistema CBR distribuido para que lo resuelva utilizando los mecanismos adecuados.

- **RetrievalResultMessage:** Este mensaje surge como respuesta al mensaje anterior y contiene los casos recuperados tras un proceso CBR deliberativo en el que han podido colaborar varios nodos en su solución dependiendo de la implementación concreta del sistema.

- **ConnectedAgentsRequestMessage:** Mensaje de solicitud de los agentes

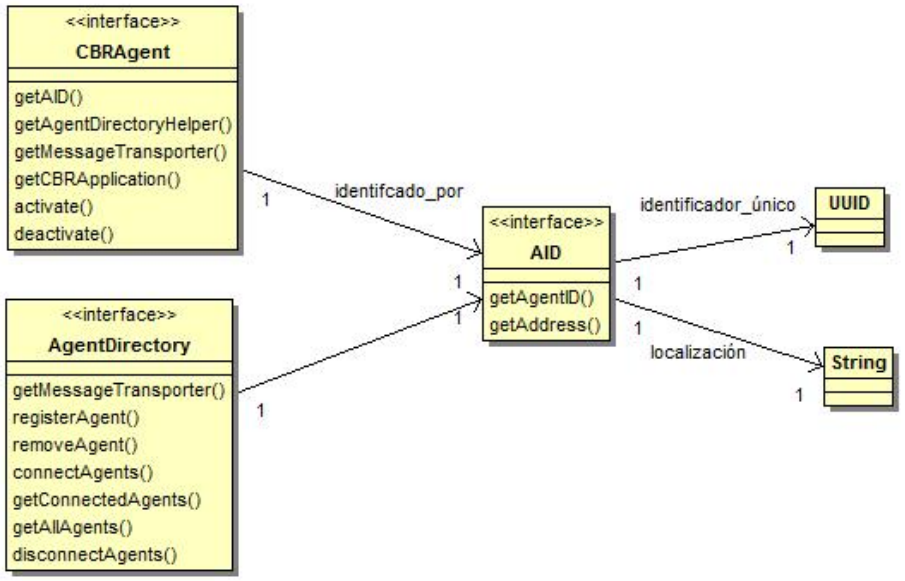

Figura 4.6: Diagrama UML de la entidad *AID*

conectados a un agente del sistema. Mediante este mensaje, la entidad *AgentDirectoryDelegate* solicitará a *AgentDirectory* los agentes enlazados al agente al que representa.

- **ConnectedAgentsAnswerMessage:** Mensaje de respuesta al mensaje anterior. Contiene los *AID* de los agentes enlazados al agente que ha realizado la solicitud.

Como hemos comentado, hemos decidido incluir estos mensajes dentro del marco práctico debido a que son independientes de la implementación concreta del sistema y son comunes a todos los sistemas CBR que se construyan.

AID: El diagrama UML de la entidad *AID* está representado en la Figura 4.6.

Cada uno de los agentes en el sistema, tanto los *CBRAgent* con el *AgentDirectory*, dispondrán de una única entidad *AID* asociada a ellos. Esta identidad es personal e intransferible y les acompañará en todo su ciclo de vida, hasta que dejen de formar parte del sistema CBR distribuido.

Cada una de las entidades *AID* del sistema tiene asociado un identificador así como una dirección únicas que sólo pueden pertenecer a esta *AID*.

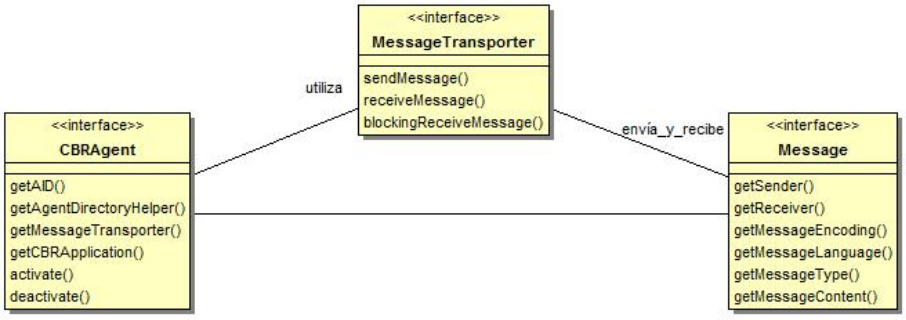

Figura 4.7: Diagrama UML de la entidad *MessageTransporter*

MessageTransporter: El diagrama UML correspondiente a la entidad *MessageTransporter* se encuentra en la Figura 4.7.

La misión fundamental del MessageTransporter es el envío y la recepción de los mensajes utilizados por los agentes del sistema multiagente. De esta entidad dependen que lleguen al destino correcto según lo esperado por su emisor.

La implementación concreta del *MessageTransporter* dependerá de los requisitos concretos del sistema CBR distribuido o del entorno en el cuál operen los agentes del sistema, pero debe resultar trasparente para los *CBRAgent* por su ocultación tras el *MessageTransporter*.

4.2. Implementación del marco práctico

La implementación realizada de esta marco práctico tiene como finalidad la realización de uno de los objetivos de este proyecto: construir un marco práctico que extienda la funcionalidad de jCOLIBRI 2 permitiendo la construcción de sistemas CBR distribuidos.

Por ello, este marco práctico ha sido implementado en el lenguaje de programación Java (lenguaje en el que está implementado jCOLIBRI), como un nuevo paquete de nombre *jcolibri.distributed.core* y que extiende la funcionalidad de jCOLIBRI 2. Este paquete contiene las seis entidades descritas anteriormente incluyendo un subpaquete *jcolibri.distributed.core.messages* que contiene la implementación de las entidades de los mensajes expuestos en la Sección 4.1.2.

Cada una de estas entidades incluidas en la nueva distribución de jCOLIBRI distribuido ha sido implementada como una interfaz, como podemos observar en cada uno de los diagramas UML de la sección anterior. También podemos observar cada uno de los métodos que incluye cada interfaz, y cómo dan respuesta a las operaciones exigidas durante la descripción de cada una de las entidades del marco práctico.

4.3. Conclusiones

Durante el desarrollo de este capítulo hemos expuesto el marco práctico de construcción de sistemas CBR distribuidos desarrollado para jCOLIBRI. Gracias al marco práctico desarrollado permitimos aprovechar las excelentes funcionalidades de jCOLIBRI extendiendo su uso al campo de los sistemas CBR distribuidos.

El estándar SC00001L de la FIPA ha sido utilizado para definir la estructura del sistema multiagente que define el marco práctico, de tal forma que contamos con las ventajas de interoperabilidad y reusabilidad objetivo de la FIPA al definir este estándar.

Para completar la información del diseño del marco práctico hemos incluido una detallada descripción de cada una de las entidades que forman parte de él, así como de las relaciones existentes entre las cada una de ellas, aportando los diagramas UML correspondientes.

En el siguiente capítulo trataremos la instanciación realizada del marco teórico del Capítulo 3 a un sistema CBR concreto.

Capítulo 5

Instanciación del marco teórico a un sistema CBR distribuido concreto

En el capítulo anterior hemos estudiado y caracterizado las diversas alternativas existentes para el diseño de sistemas CBR distribuidos. En este capítulo y como uno de los objetivos, vamos a diseñar un sistema basado en las alternativas anteriores, realizando una aportación significativa a los sistemas ya existentes.

Existen cuatro funcionalidades claves dentro del sistema diseñado:

1. **Propagación de la consulta:** El sistema de propagación de la consulta en sistemas CBR distribuidos hace posible la llegada de una consulta a distintos nodos de la red y se encarga de la composición de los resultados de los distintos procesos deliberativos así de como de la integración de las soluciones individuales de los distintos agentes dentro del sistema.

2. **Protocolo de argumentación:** Aporta a los individuos del sistema las capacidades necesarias para desarrollar un proceso deliberativo. Gracias a él, los individuos del sistema pueden aportar argumentaciones a favor y en contra de las distintas soluciones a una consulta CBR planteada al sistema CBR distribuido. Como veremos en la Sección 5.2, para la creación del protocolo de argumentación se ha realizado una adaptación del protocolo AMAL definido en [2].

3. **Sistema de razonamiento difuso:** Permite la toma de decisiones dentro del protocolo de argumentación y elimina las restricciones impuestas por el mismo acerca de los casos. Mediante el sistema de razonamiento difuso, un individuo dentro de un protocolo de argumentación puede saber cuando uno de sus casos constituye un contraejemplo contra otro o una defensa a favor de sus soluciones. Además, determinará cuando los contraejemplos y las defensas son aceptadas.

4. **Sistema de aprendizaje del modelo de confianza:** Además del sistema de aprendizaje tradicional CBR que consiste en la inclusión de nuevos casos, se ha diseñado un sistema de actualización de los valores de credibilidad o confianza entre los distintos nodos del sistema distribuido, de tal forma que estos valores sean más altos cuanto mejores sean las respuestas dadas por un nodo a otro. Los valores de credibilidad del sistema distribuido, influirán en la aceptación de contraejemplos y defensas dentro del protocolo de razonamiento, de forma que confiaremos más en las argumentaciones de los individuos cuya mayor credibilidad tengan.

El sistema diseñado es independiente del lenguaje de programación concreto así como del formato de los casos que maneje, así que puede ser implementado sobre cualquier lenguaje de programación así como utilizado sobre cualquier base de casos.

En las diferentes secciones de este capítulo se irá tratando con detalle el diseño de cada una de estas funcionalidades. En la Sección 5.1 se tratará el mecanismo de propagación de las consultas y soluciones dentro del sistema distribuido, en la Sección 5.2 se explicará el protocolo AMAL modificado, en la Sección 5.3 se tratará el sistema de razonamiento fuzzy y, por último, en la Sección 5.4 se expondrá el sistema de aprendizaje distribuido y la actualización de los valores de credibilidad.

5.1. Funcionamiento de los sistemas CBR deliberativos

Como ya describimos en el Capítulo 3 de este trabajo, consideramos como un sistema CBR distribuido aquél sistema CBR cuya base de casos no se encuentra localizada en un sólo punto, sino que se encuentra dividida ya sea en un mismo lugar o bien repartida a lo largo del mundo, o a aquél sistema CBR formado por varias bases de casos. Cada una de estas bases de casos se encuentra gestionada por un agente, que se encargará de realizar las operaciones CBR relacionadas con ella, así como de gestionar las conexiones de red y las comunicaciones con otros agentes que estén representando a otras bases de casos. Consideramos cada uno de estos agentes CBR junto con su base de casos como un nodo del sistema distribuido.

Cada uno de los nodos es accesible desde una serie de nodos con los que se encuentra conectado. Todos los nodos del sistema distribuido son accesibles al menos desde otro nodo del sistema, por lo que ninguno de ellos se encuentra aislado.

El objetivo de esta sección es exponer la forma en la que una consulta iniciada en uno de los nodos del sistema es propagada a través del sistema distribuido, cómo influye en la propagación la topología del sistema, como son devueltas las soluciones y finalmente, de qué manera podemos enfrentarnos a los problemas introducidos por la topología.

5.1.1. Proceso de propagación de la consulta

El primer paso en el funcionamiento del sistema distribuido es la llegada de una consulta a uno de los nodos del sistema. Ante esta petición, el nodo propagará la consulta de forma homogénea a todos los nodos enlazados a él. La llegada de la consulta dará lugar a su vez a otro proceso deliberativo en cada uno de estos nodos, y esto a su vez a otra consulta de estos nodos a los nodos conectados a ellos. Si una petición llega a un nodo del sistema distribuido y este debe generar a su vez otra consulta, nunca se enviará por el enlace por el que se ha recibido, de forma que no consultamos al nodo de la red que nos la ha solicitado.

Para cada nodo del sistema (salvo para el que ha iniciado la solicitud, puesto que no utilizará su base de casos para resolver el problema, salvo que así lo queramos expresamente), una vez se han obtenido los casos devueltos por el proceso argumentativo con los nodos enlazados, se procederá a realizar un ranking con estos casos y los casos recuperados de la base de casos del propio nodo. Los casos devueltos por este nodo y los que defenderá durante el siguiente proceso de argumentación, serán los n primeros del ranking, siendo n el número de casos solicitados.

De esta forma, la propagación de la consulta en el sistema distribuido se realizará en dos sentidos:

- Un primer sentido (hacia delante), el que se deriva de propagar la consulta desde el nodo inicial a los nodos enlazados a él y desde estos a los nodos enlazados a ellos, etc. En este sentido de propagación, se enviará de un nodo a otro la consulta al sistema distribuido (Figura 5.1 (a))

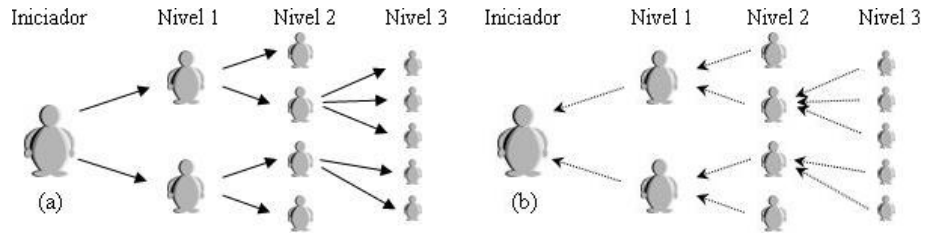

Figura 5.1: Propagación de una consulta en un sistema distribuido

- Un segundo sentido de propagación (hacia atrás), que se obtiene cuando los nodos han resuelto la consulta (como resultado del proceso argumentativo expuesto en la Sección 5.2) y deben devolver los resultados a los nodos que les enviaron las consultas (Figura 5.1 (b))

5.1.2. Límite en la propagación de la consulta

Con las restricciones actuales la consulta se extendería completamente por el sistema distribuido. Por ello, debemos fijar un límite en la propagación en forma de número máximo de pasos que puede realizar dentro del sistema. Este parámetro va a limitar el tiempo de respuesta, el número de casos dentro de los sistemas CBR que son valorados, y por lo tanto, la calidad de la respuesta del sistema (como se demostrará más adelante en la Sección 6.4).

Una vez introducido el número máximo de pasos que puede dar una consulta dentro del sistema, existen dos formas distintas en las que los agentes devuelven sus casos ante una consulta:

1. Si no se ha alcanzado el número máximo de pasos de la consulta dentro del sistema, se iniciará un nuevo proceso argumentativo, enviando la consulta a los nodos adyacentes, por lo que el número de pasos de la consulta se incrementará en uno. Este caso corresponde a los agentes del nivel 1 y a los agentes centrales del nivel 2 de la Figura 5.1

2. Si se ha alcanzado el número máximo de pasos (o si los nodos únicamente están enlazados a aquellos que les envían la consulta), se resolverá la consulta dentro de ese nodo de forma aislada utilizando únicamente la base de casos de este. Este caso corresponde a los agentes del nivel 3 y a los agentes superior e inferior del nivel 2 de la Figura 5.1

El número de pasos de la consulta va a marcar también el número de nodos visitados dentro del sistema distribuido:

- Suponiendo un sistema en el que cada nodo está conectado a otros 10 nodos, y siempre son nodos distintos, una consulta en 5 pasos recorrería $10 + 102 + 103 + 104 + 105 = 111{,}110$ nodos.

- Suponiendo un sistema en la que cada nodo está conectado a otros 20 nodos, y 5 de ellos son iguales en cada caso, una consulta en 7 pasos recorrería $15 + 152 + 153 + 154 + 155 + 156 + 157 = 183{,}063{,}615$ nodos

Es importante ver que el tiempo de respuesta del sistema no depende del número de nodos visitados, sino del número de pasos dentro del sistema, dado que todas las consultas dentro de un mismo paso se pueden resuelven de forma simultánea, con lo que conseguiríamos obtener una respuesta en la que habrían participado una gran cantidad de nodos (depende del número de conexiones de cada nodo y de las repeticiones dentro de los nodos conectados) en un tiempo que únicamente corresponde a realizar el proceso de razonamiento tantas veces como pasos realice la consulta dentro del sistema.

Además, descargamos el procesamiento realizado por el nodo que realiza la consulta (en cada nodo se realiza un proceso argumentativo que contribuye a calcular una solución distribuida entre todos los nodos del sistema) así como su ancho de banda (las conexiones no se realizan únicamente con el nodo que realiza la consulta como en [2][1], sino que están repartidas por toda la red, de forma que cada nodo se comunica únicamente con los nodos que están conectados con él).

5.1.3. Problemas de las topologías de los sistemas distribuidos

Analizando la topología de los sistemas distribuidos podemos observamos dos problemas existentes para el proceso de propagación de las consultas: los bucles, y los caminos múltiples a un nodo. Hasta en sistemas sencillos como el de la Figura 5.2 aparecen estos problemas. Podemos observar como existe un bucle entre A1, A2 y A3 y entre A0, A1 y A2 y como existe un camino múltiple para llegar desde A0 a A1, bien directamente (1 paso) o bien a través de A2 (2 pasos), por lo que la misma petición llegará a A1 en momentos distintos el proceso de razonamiento.

Como primera aproximación para resolver los problemas en los sistemas distribuidos se plantea la siguiente alternativa: Dado que hemos establecido un número máximo de pasos para el proceso de razonamiento, llegará el momento en el que se alcance el máximo número de pasos y por tanto, el proceso de vuelta atrás para la devolución de los casos recomendados comenzará. El caso de los caminos múltiples no resultaría tan fácil. Dado que se produce la llegada de una misma petición de recomendación por varios puntos distintos de la red, el agente podría tener dificultades a la hora de distinguir entre los dos procesos de razonamiento y cuando es devuelta cada una de las consultas.

A pesar de que el coste computacional del proceso de argumentación no es elevado, no resulta interesante que un mismo proceso se reitere en un nodo. Para poder solucionar este problema, en primer lugar debemos tener una forma de identificar las distintas consultas que realiza un agente. La forma escogida para representar esta información es la siguiente:

[1] las conexiones se hacen desde todos los nodos al nodo iniciador, lo que supone una gran carga, tanto de red como de procesamiento para ese nodo

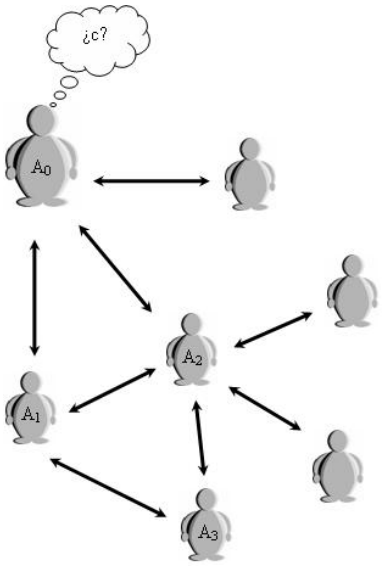

Figura 5.2: Problemas de las topologías de los sistemas distribuidos

$$id_{consulta} =< id_{nodo}, n_{peticion} > \quad \text{donde:} \tag{5.1}$$

- id_{nodo}: Identificador del nodo que realiza la consulta original
- $n_{peticion}$: Número de la consulta realizada por ese nodo de la red social

Una primera solución al problema de los bucles sería, para cada uno de los nodos de la red social, tener constancia de cada una de las consultas que ha procesado, de tal forma que para cada nodo de la red, almacenaría únicamente la última consulta procesada de cada nodo, suponiendo que no pueden realizar una nueva consulta antes de obtener los resultados de la última. De esta forma, ante la llegada de una consulta a un nodo existirían dos posibilidades:

- Si no ha procesado previamente esa consulta, responderá a la misma siguiendo el proceso establecido, añadirá el identificador de consulta a la lista de consultas procesadas y borrará la última consulta procesada para ese nodo si existiera
- Si ya la ha procesado, puede tratarse de una de las dos opciones siguientes:
  - Que se trate de un bucle, y por tanto estará esperando en algún punto del proceso argumentativo su propia solución, por lo tanto su base de casos será utilizada más adelante, así que devolverá n soluciones vacías

- Que la consulta haya llegado por un camino distinto. Este caso es más complejo. Dado que las credibilidades de los nodos de los distintos caminos pueden ser distintas, no podemos dejar de incluir las soluciones de este nodo en cada una de las dos consultas, puesto que es posible que sean valoradas de forma distinta en cada uno de los dos caminos, y por tanto sean ambas necesarias.

Analizando detenidamente esta propuesta de solución nos encontramos con dos problemas fundamentales:

- Coste del almacenamiento del registro de consultas: Con los datos del número de nodos de los sistemas distribuidos presentadas como ejemplo en la Sección 5.1.2, para una red de 183.063.615 nodos, suponiendo que cada nodo haya realizado una consulta, necesitamos en cada uno de los nodos del sistema un registro de consultas procesadas de 183.063.615 entradas, lo que resulta muy ineficiente tanto en coste de almacenamiento como en el procesamiento y la búsqueda de nuevas consultas en el registro

- Indistinguibilidad entre bucles y caminos múltiples: Ante la llegada de una nueva consulta resulta imposible distinguir si se trata de un bucle o de un camino múltiple, por lo que no podemos realizar la distinción entre el procesamiento de ambas consultas en el caso de caminos múltiples y no procesar las consultas cuando se trate de bucles

  Todo ello lleva a rechazar la proposición actual y a buscar una nueva forma de representación de las consultas y el registro de las mismas.

5.1.4. Solución a los problemas de la topología de los sistemas distribuidos

Una vez planteado el problema y una solución parcial que no satisfacía completamente el problema planteado, se plantea la búsqueda de una nueva solución. La alternativa tomada para resolver los problemas de los sistemas distribuidos es la siguiente:

- Eliminación de la tabla de registros en cada uno de los nodos del sistema

- Extensión de la identificación de la consulta de la siguiente forma:

$$id_{consulta} = < id_{nodo}, n_{peticion}, traza > \quad \text{donde:} \tag{5.2}$$

- id_{nodo}: Identificador del nodo que realiza la consulta originaria

- $n_{peticion}$: Número de la consulta realizada por ese nodo del sistema distribuido

- $traza$: Lista de los nodos por los que ha pasado la consulta. Esta lista se irá completando a medida que esta se propaga por la red de forma que se irán añadiendo los nodos por los vaya procesándose.

Las ventajas que nos presenta esta identificación de las consultas son las siguientes:

- Permite prescindir del registro de consultas procesadas en cada nodo, con lo que ello supone en ahorro en coste de almacenamiento y cálculo.
- Resuelve el problema de la identificación entre los bucles y los caminos múltiples en la red: Para saber si una consulta ha sido ya procesada por un nodo, este sólo necesitará comprobar si se encuentra en la traza de la consulta, por lo tanto:
  - Si el nodo se encuentra en la traza de la consulta, indicará que se trata de un bucle y por tanto devolverá n recomendaciones vacías.
  - Si el nodo no está en el traza de la petición, simplemente procesará la petición de forma normal independientemente de que se trate de un camino múltiple.
- Permite la identificación de peticiones iguales en el caso de que se haya tratado de caminos múltiples. Se mantendrá un registro sólo con las peticiones en proceso, es decir, de aquellas a las que todavía no se haya devuelto una recomendación y bastará identificar parte de la traza con el registro para saber a qué petición estamos contestando.
- Permite conocer el destinatario de nuestra recomendación, simplemente comprobando el elemento situado antes del nodo en la traza de la petición.

5.2. Protocolo de razonamiento multiagente

El diseño de un sistema multiagente en el que cada uno de los nodos del sistema posee una información que puede ser distinta o incluso contradictoria con la información de los restantes agentes del sistema, conlleva la necesidad de buscar un mecanismo que permita a los agentes compartir la información de su base de casos con los otros agentes así como poder deliberar acerca de la solución correcta a un problema.

Este mecanismo de deliberación resulta una pieza clave en la construcción de un sistema deliberativo distribuido, puesto que el correcto funcionamiento del sistema dependerá, en gran medida, de la utilización de un protocolo de argumentación que permita a los agente intercambiar información de forma rápida y fiable, alcanzando solución válida al problema que se está abordando.

En esta sección se muestra la adaptación detallada realizada del protocolo de argumentación AMAL descrito en [2] para ajustarlo al problema al que nos enfrentamos, mostrando las principales diferencias con el protocolo AMAL original, así como un ejemplo guía para aclarar el funcionamiento del mismo.

5.2.1. Protocolo de argumentación AMAL modificado

Como inicio del proceso de argumentación, un agente A_0 realiza una consulta al conjunto de agentes con los que está relacionado. Cada uno de estos agentes obtendrá por su cuenta n casos de respuesta a la consulta como hemos visto en la Sección 5.1.1. Una vez los han obtenido, se realizará el proceso de argumentación *n* veces (tantas como casos requiera el agente que realiza la consulta), durante cada una de las cuáles, los agentes defenderán uno de los casos que han devuelto a A_0 (Figura 5.3). De esta argumentación obtendremos *n* casos, uno por cada uno de los procesos de argumentación que se han realizado (Figura 5.4).

Como podemos observar en la Figura 5.4, para cada una de las *n* argumentaciones, se coge la recomendación i de cada uno de los agentes, y se realiza el proceso de argumentación *i* sobre ellas. De cada argumentación se obtiene un caso, de donde conseguimos los n casos que hace el conjunto de agentes devuelve al agente A_0

Es importante señalar que el agente que realiza la consulta no aporta casos al proceso de argumentación (no se responde a sí mismo). A pesar de ello, juega un papel fundamental en este proceso:

- Define la credibilidad del resto de agentes que participan en el proceso argumentativo.
- Decide si un contraejemplo dado por un agente frente al caso de otro es aceptado o no en función de las credibilidades de los agentes implicados y las valoraciones de los casos devueltos.
- Decide si la defensa presentada por un agente frente a un contraejemplo es

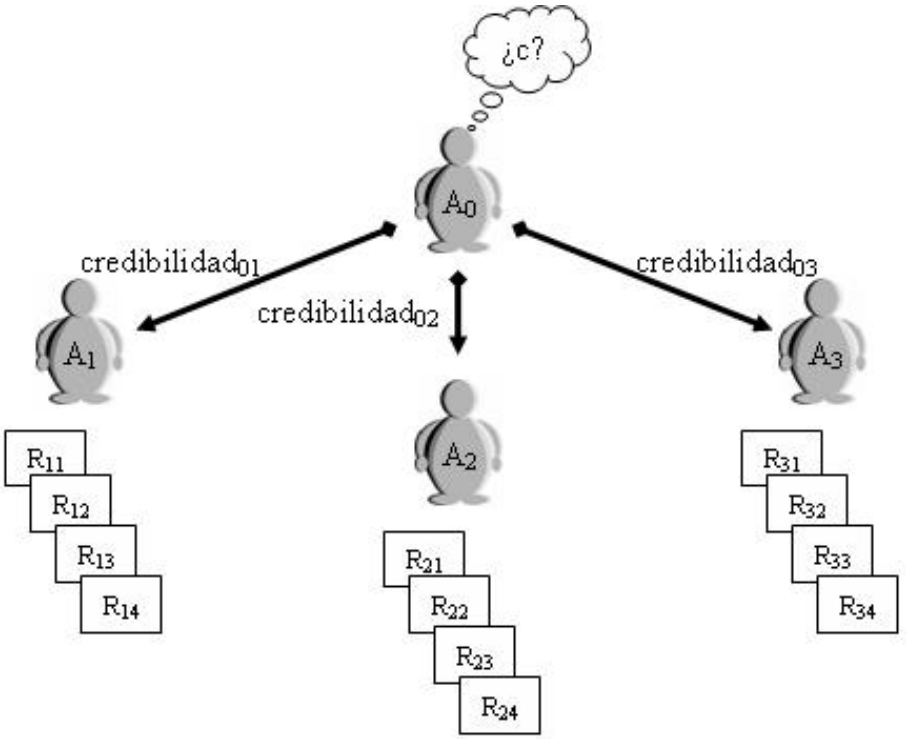

Figura 5.3: Esquema del protocolo AMAL modificado

aceptada o no en función de la credibilidad de los agentes implicados y las valoraciones de los casos.

Una vez establecidas las características de la argumentación para sistemas deliberativos, veamos cómo se desarrolla el protocolo paso a paso para un proceso argumentativo *i*:

1. En la ronda t = 0, cada agente A_a escoge su caso R_{ai} asertándolo a los otros agentes mediante el mensaje assert(R^0_{ai}). Al final de la ronda cada agente conoce $H^0_i =< R^0_{1i}, ..., R^0_{mi} >$, que representa el conjunto de recomendaciones que cada agente defiende en la ronda 0 para el proceso argumentativo i. Se envía un token que servirá para dar la palabra al primer agente del proceso argumentativo.

2. Para todas las rondas t > 0, A_0 comprueba si ha finalizado la argumentación por alguno de los siguientes motivos:

   - Si únicamente existe un caso en H^t_i (el resto ha sido refutado mediante contraejemplos).
   - Si existe un acuerdo en la argumentación (los casos existentes en H^t_i son iguales).

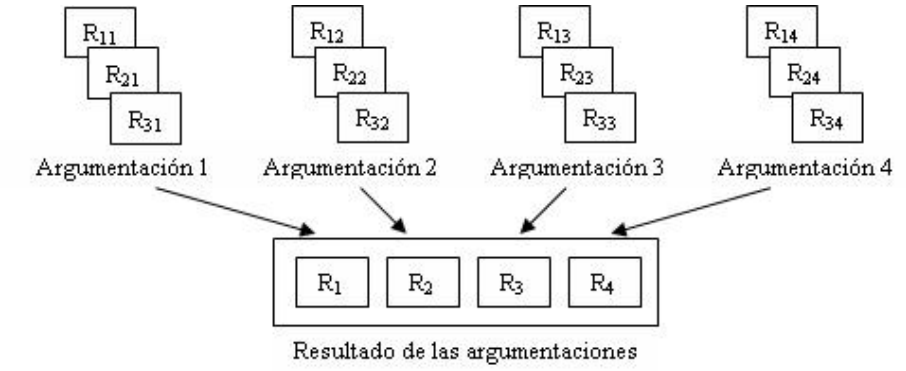

Figura 5.4: Composición de los resultados de las argumentaciones

- Si no se ha generado ningún contraejemplo durante las últimas m rondas (siendo m el número de agentes que participan en el proceso argumentativo, es decir, que están enlazados con A_0).

Si se da alguna de estas condiciones, el algoritmo continuará en el paso 6. En caso contrario, el agente A_a poseedor del token trata de generar un contraejemplo para cada uno de los casos en H_i^t que no coincidan con el suyo. Se elegirá el contraejemplo β_{ai}^t contra el caso R_{ji}^t que tenga el mayor peso $C(\beta_{ai}^t, R_{ji}^t)$, siempre y cuando no se haya realizado ya otro contraejemplo contra el agente A_j en ese proceso argumentativo. Cabe destacar que puede que A_a no esté defendiendo ninguna recomendación en ese momento, como resultado de que haya sido eliminada por un contraejemplo de otro agente. Sin embargo, el agente A_a puede seguir generando contraejemplos para refutar casos de otros agentes, hasta que sólo quede un caso en H_i^t, momento en el que el algoritmo se detendría por las condiciones anteriores. En el caso que el agente A_a pudiera generar contraejemplos para todos los casos del resto de agentes, nos aseguramos de que el último que quede sea el caso contra el que existe un contraejemplo más débil, y por tanto, el mejor desde el punto de vista del agente A_a, puesto que vamos eligiendo los contraejemplos del más fuerte al más débil (gracias al valor $C(\beta_{ai}^t, R_{ji}^t)$, calculado como veremos en la Sección 5.3 gracias a un sistema fuzzy). En este punto, se nos presentan dos opciones:

- Si A_a puede generar el contraejemplo β_{ai}^t contra R_{ji}^t, siendo este el más fuerte de los contraejemplos generados y no ha generado ya otro ejemplo contra R_{ji}^t entonces A_a envía rebut(β_{ai}^t, R_{ji}^t) a A_0 para que lo evalúe. El algoritmo pasa al estado 3.
- Si A_a no puede generar contraejemplos, el token es pasado al siguiente agente, comienza la ronda t+1 y el protocolo pasa al estado 2.

3. A_0 recibe un contraejemplo de A_a contra el caso R_{ji}^t de A_j (mensaje rebut-(β_{ai}^t, R_{ji}^t)). Tras ello, evaluará la credibilidad de los agentes implicados, de la recomendación y del contraejemplo a través del sistema fuzzy de la Sección 5.3. Se abren dos posibilidades:

   - Si A_0 acepta el contraejemplo de A_a, el protocolo pasará al estado 4.
   - Si A_0 no acepta el contraejemplo de A_a, el agente A_a no podrá dar otro contraejemplo contra A_j, enviará el token al siguiente agente, el protocolo pasará al estado 2 y se iniciará una nueva ronda *t+1*.

4. Un agente A_j ha recibido un contraejemplo de un agente A_a considerado por A_0 como válido. En este caso, el agente A_j deberá buscar una defensa (contraejemplo) contra β_{ai}^t en su base de casos. Existen dos opciones:

   - Si A_j encuentra una defensa para β_{ai}^t en su base de casos, lo notificará a A_0 mediante defend(δ_{ji}^t, β_{ai}^t). El protocolo pasará al estado 5.
   - Si A_j no ha podido encontrar una defensa en su base de casos para β_{ai}^t, el caso de A_j queda rechazado y eliminado de H_i^t mediante el envío del mensaje reject(R_{ji}^t) a todos los agentes por parte de A_j. En este caso, el protocolo pasará al estado 2, se iniciará una nueva ronda t+1 y el token pasará al siguiente agente.

5. A_0 recibe una defensa de A_j contra un contraejemplo del agente A_a (mensaje defend(δ_{ji}^t, β_{ai}^t)). A_0 deberá valorar las credibilidades de los agentes en disputa y los pesos de sus casos. Tras ello, existirán dos opciones:

   - A_0 acepta la defensa de A_j, por lo que seguirá aceptando su caso. Para comunicar su decisión mandará el mensaje accept(δ_{ji}^t, β_{ai}^t) a los agentes A_j y A_i. En este caso, el agente A_i no podrá volver a generar ejemplos contra A_j.
   - A_0 no acepta la defensa de A_j, con lo que rechaza su caso aceptando los argumentos de A_i. Para comunicar a los agentes su decisión, manda reject(R_{ji}^t) a todos ellos.

     En cualquiera de los dos casos, el protocolo continuará en el paso 2, con una nueva ronda *t+1*.

6. El protocolo puede finalizar por una de las siguientes tres situaciones:

   - Únicamente existe un caso en H_i^t. Esto se debe a que los casos del resto de los agentes han sido refutados. El resultado del protocolo de argumentación será este caso restante.
   - Existe un acuerdo en la argumentación (los casos existentes en H_i^t son iguales). Estoy se debe a que dos agentes han devuelto el mismo caso y este no ha sido refutado. El resultado del protocolo de argumentación será este caso.

- No se haya generado ningún contraejemplo durante las últimas m rondas (siendo m el número de agentes que participan en el protocolo). Esta situación se alcanza cuando existe más de un caso en H_i^t distinto, y los agentes del comité no pueden generar contraejemplos para los casos o los que han generado han sido defendidos. El caso devuelto será un caso resultado de un proceso que combinará la credibilidad de los agentes con las valoraciones de sus casos.

Cabe destacar que, durante el proceso de argumentación, A_0 puede encontrar un contraejemplo contra una de los casos. En este caso, no necesitaría mandarse así mismo el mensaje rebut(β_{0i}^t, R_{ji}^t). Además, dado que consideramos que el agente A_0 tiene una credibilidad del 100 % consigo mismo, bastará con que envíe el mensaje reject(R_{ji}^t) a los agentes para informarles que el caso que le ha devuelto A_j no es de su agrado, dado que en este caso no cabe defensa ninguna por parte de A_j.

5.2.2. Limitaciones del protocolo AMAL original

Una vez descrito el protocolo de argumentación modificado, analicemos las limitaciones que tenía el protocolo AMAL tradicional:

- En la topología de red del protocolo AMAL tradicional (Figura 5.5, izquierda), todos los nodos se encuentran conectados con todos. Esta topología presenta problemas de escalabilidad en sistemas reales con una gran cantidad de nodos. Todos los nodos del sistema se encuentran participando en el mismo proceso de argumentación y esto provoca problemas de ineficiencia.

- En problemas reales, los nodos se encuentran organizados siguiendo topologías de usuarios similares, en los que existe normalmente una relación en sus bases de casos (calculada mediante el coeficiente de Pearson [34]). La topología todos con todos utilizada en el protocolo AMAL original no refleja estas relaciones entre nodos.

- AMAL está basado en el uso de Lógicas Descriptivas[2] (para mayor información acerca de las Lógicas Descriptivas consultar [22]). Muchos de los problemas reales a los que nos podemos enfrentar no toleran la construcción de razonamientos sobre sus casos con este tipo de lógicas, ya se o bien por la representación o por la propia naturaleza (por ejemplo, un registro que representa una canción en un sistema recomendador) de los mismos.

- AMAL no tiene en cuenta la confianza entre nodos en el protocolo de razonamiento.

[2] DLs

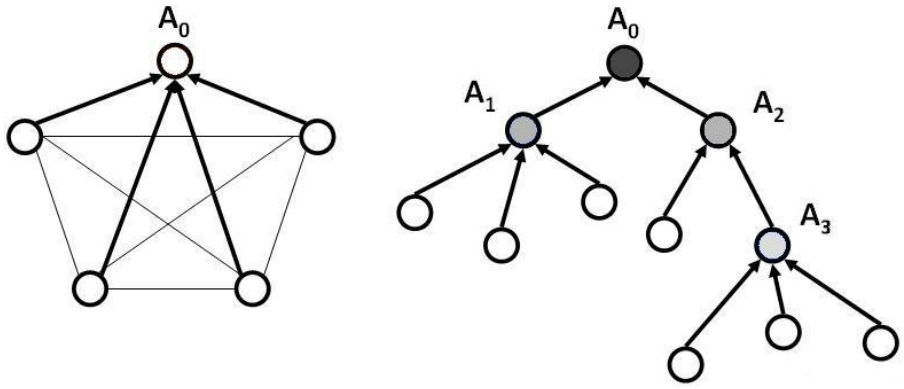

Figura 5.5: Topologías en AMAL original y AMAL modificado

5.2.3. Diferencias con el protocolo AMAL original

A continuación, para poder apreciar de manera más clara las modificaciones que se han realizado sobre el protocolo original, se muestran las principales diferencias entre los dos protocolos:

- El protocolo modificado permite la utilización del sistema sobre casos en los que no se puede aplicar Lógicas Descriptivas y sólo requiere que los casos sean valorados. Las capacidades de razonamiento de las DLs serán sustituidas por el sistema de razonamiento difuso descrito en la Sección 5.3.

- Permite su aplicación sobre casos no nítidos. En el protocolo original sólo se contempla la posibilidad de argumentaciones sobre conjuntos nítidos o clases.

- En el protocolo AMAL modificado se introduce el concepto de *defensa*. Una defensa es un caso de la base de casos de uno de los agentes que participa en la argumentación y que sirve para justificar una solución ante un contraejemplo. El concepto de defensa se introduce debido a que existen argumentaciones en el protocolo AMAL original que necesitan la utilización de Lógicas Descriptivas y que en el caso del protocolo modificado no pueden ser generadas.

- El protocolo de argumentación sigue la estructura del sistema distribuido. Esto permite incrementar el número de nodos que participan en el proceso de argumentación evitando los problemas de escalabilidad. Además permite la utilización de los valores de confianza entre los nodos.

- En el protocolo modificado la recuperación de los casos es jerárquica. Cuando estamos respondiendo a una consulta, el agente que realiza la consulta se convierte en la raíz de la jerarquía de agentes que están enlazados a él (sus

hijos en el árbol de la Figura 5.5). Cuando ha acabado el proceso de argumentación con los nodos enlazados a él, devolverá los resultados al nodo que le ha enviado la consulta (su padre en el árbol de la Figura 5.5). El proceso de propagación de la consulta se reitera hasta que se alcanzan las hojas del árbol y la propagación hacia atrás de los resultados hasta que se alcanza el nodo raíz que ha realizado la primera consulta. Las flechas en la Figura 5.5 representan la dirección del envío de las soluciones. Por el contrario, el protocolo original no realiza ninguna recuperación jerárquica y todos los nodos envían directamente las soluciones al nodo que ha originado la consulta. De esta forma, hemos distinguido dos papeles diferenciados que juegan los agentes dentro de un proceso de argumentación:

- El agente que manda la consulta a sus nodos hijos, cuyo papel dentro de la argumentación es definir las credibilidades de los diferentes agentes, así como decidir cuando las defensas y los contraejemplos son aceptados o rechazados. Además puede rechazar casos de los restantes agentes, pero no puede aportar soluciones a la argumentación. Los resultados generados por el proceso argumentativo, junto con los recuperados de su base de casos (después de realizar el ranking comentado en la Sección 5.1.1, serán los que defienda en el proceso argumentativo con su nodo padre en el que jugará el otro papel que se describe a continuación.
- Los agente a los cuales les a llegado una consulta y la han resuelto y cuyo comportamiento es similar. Su objetivo es aportar casos para solucionar la consulta así como aportar contraejemplos frente a los casos aportados por otros agentes que juegan su mismo papel. Por lo tanto, dentro de un proceso argumentativo, siempre existirá un único agente que juegue el primero de los papeles, pudiendo existir uno o varios (al menos uno) agentes que jueguen el segundo de los papeles.

- En el protocolo modificado es posible aprovechar la información contenida de forma implícita o explícita en un sistema distribuido en forma de confianza y similaridad entre nodos.

5.2.4. Ejemplo de funcionamiento del protocolo AMAL modificado

La última sección en la exposición del protocolo de argumentación diseñado es un ejemplo que sirva para aclarar el funcionamiento del mismo.

Situemos el ejemplo: supongamos un recomendador musical distribuido y un agente A_0 enlazado a otros tres agentes A_1, A_2 y A_3. El agente A_0 realiza una consulta a los agentes enlazados a él pidiendo que se le devuelvan tres soluciones. En concreto, el agente A_0 pide discos cuyo artista sea Mike Oldfield (Figura 5.6).

Ante la consulta de A_0 los agentes A_1, A_2 y A_3 generarán tres soluciones. En cada uno de los tres procesos deliberativos que se realizan, los agentes defienden

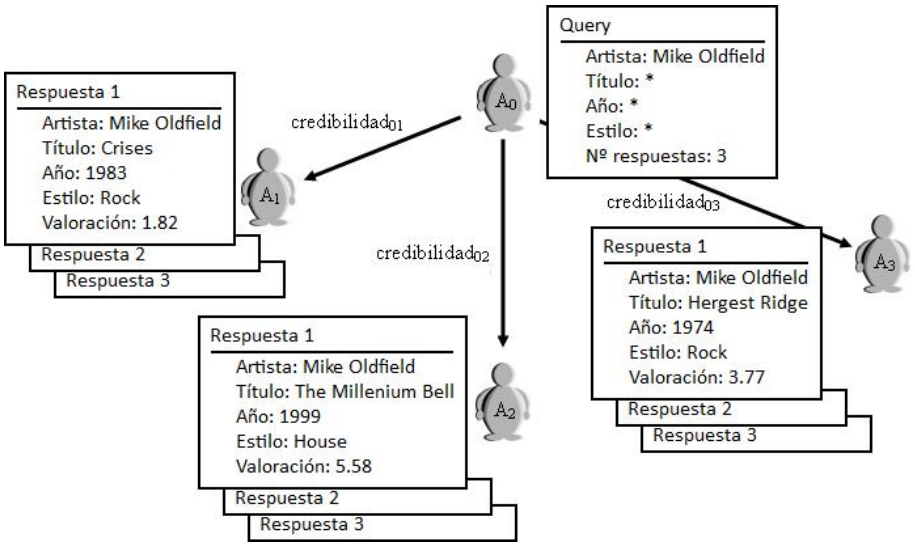

Figura 5.6: Ejemplo del protocolo de argumentación

uno de sus casos. En este ejemplo sólo nos vamos a centrar en el primer proceso argumentativo, en el cual los agentes defenderán la primera de sus soluciones y cuyo resultado será el primero de los casos que obtiene A_0 como solución de la consulta. Los otros dos casos restantes los obtiene de los procesos deliberativos segundo y tercero, en los cuales los agentes defiende sus segundas y terceras soluciones respectivamente.

Cuando los tres agentes enlazados con A_0 devuelvan los casos con los que participarán en el proceso deliberativo, el protocolo comenzará. El orden que seguirán los agentes a la hora de realizar sus argumentaciones será: A_2, A_3 y A_1 (escogido en función del orden en el que han ido obteniendo sus respuestas). En primer lugar, el agente A_0 enviará el token al agente A_2.

A_2 revisará los casos de su base de casos que constituyan un contraejemplo contra las soluciones de A_1 y A_3 (gracias al sistema de decisión fuzzy explicado en la Sección 5.3). De todos los contraejemplos obtenidos, elegirá aquel que considere más fuerte. En concreto, como se puede observar en la tabla de la Figura 5.7, A_2 presenta un contraejemplo contra la solución aportada por A_1, que es un caso similar a la solución de A_1 y con una valoración baja. A_0 valora la credibilidad de A_2 y el contraejemplo presentado y decide aceptar el contraejemplo. Dado que se ha aceptado el contraejemplo contra C_1, A_1 debe intentar buscar una defensa en su base de casos a favor de su solución C_1. Una defensa será un caso similar al contraejemplo presentado por A_2 pero con una valoración alta. A_1 no puede encontrar una defensa en su base de casos a favor de C_1 por lo que su solución es eliminada del proceso argumentativo. El token pasará al agente A_3.

A_3 buscará un contraejemplo contra la solución de A_2 (no busca contraejem-

	Valoración	Artista	Título	Año	Estilo
Consulta (Q)	10	Mike Oldfield	*	*	*
Caso A_2 (C_2)	5,58	Mike Oldfield	The Millennium Bell	1999	House
Caso A_3 (C_3)	3,77	Mike Oldfield	Hergest Ridge	1974	Rock
Caso A_1 (C_1)	1,82	Mike Oldfield	Crises	1983	Rock
Ronda 1: A_2 tiene el token					
Contraejemplo de A_2 contra C_1	0,5	Mike Oldfield	Incantations	1978	Rock
A_0 valora la credibilidad de A_2 y el contraejemplo y decide aceptarlo					
A_1 no puede generar una defensa					
Resultado de la argumentación tras la ronda 1:					
C_2	5,58	Mike Oldfield	The Millennium Bell	1999	House
C_3	3,77	Mike Oldfield	Hergest Ridge	1974	Rock
Ronda 2: A_3 tiene el token					
Contraejemplo de A_3 contra C_2	1,36	Deep Dish	George is on	2005	House
A_0 valora la credibilidad de A_3 y el contraejemplo y decide aceptarlo					
Defensa de A_2 (C_e)	5,3	Deep Dish	George is on	2005	House
A_0 valora la credibilidad de A_2 y A_3 y el contraejemplo y la defensa presentada y decide aceptar la defensa					
Resultado de la argumentación tras la ronda 2:					
C_2	5,58	Mike Oldfield	The Millennium Bell	1999	House
C_3	3,77	Mike Oldfield	Hergest Ridge	1974	Rock
Ronda 3: A_1 tiene el token					
Contraejemplo de A_1 contra C_2	1,67	Mike Oldfield	Earth Moving	1989	Pop
A_0 valora la credibilidad de A_1 y el contraejemplo y decide aceptarlo					
A_2 no puede generar una defensa					
Resultado de la argumentación tras la ronda 3:					
C_3	3,77	Mike Oldfield	Hergest Ridge	1974	Rock

Figura 5.7: Esquema del proceso argumentativo

plos contra su propia solución) en su base de casos. A_3 consigue encontrar un contraejemplo y lo presenta a A_0 quien tras evaluarlo y evaluar la credibilidad de A_03 decide aceptarlo. A_2 busca en su base de casos una defensa a favor de su caso C_2 y encuentra un caso que puede actuar como tal y lo presenta a A_0. A_0 lo evalua junto con el contraejemplo de A_3 y las credibilidades de ambos agentes y decide aceptar la defensa de A_2, por lo que la solución de A_2 permanece en el protocolo y A_3 no podrá volver a presentar contraejemplos contra A_2. El token pasa al agente A_1.

A_1, como se ha comentado en las secciones anteriores, a pesar de no defender ya ninguna solución en la argumentación, puede seguir presentando contraejemplos frente a las soluciones de los otros agentes que participan en el proceso. A_1 revisa su base de casos y decide que su mejor contraejemplo es un caso frente a la solución del agente A_2. A_0 evaluando el contraejemplo de A_0 y su credibilidad decide aceptarlo. Dado que A_2 no puede generar ninguna defensa a favor de su caso C_2, su solución queda eliminada de la argumentación.

En este momento, sólo existe un caso en las soluciones del proceso argumentativo, por lo que este finaliza, obteniendo como primera respuesta para A_0 el caso C_3. En el caso de que todavía quedaran varias soluciones en el proceso, el token pasaría al agente A_0, quién podría eliminar de forma automática uno de los casos de la argumentación, aquel contra el que tuviera el contraejemplo más fuerte. Si en este momento todavía no hubiese acabado la argumentación, el token pasaría de

nuevo al agente A_2 y comenzaría un nuevo ciclo.

Cabe destacar que el caso resultante de la argumentación no es el caso con la mayor valoración. Esto, como se verá más adelante en el caso práctico (ver el Capítulo 6), se debe a que no sólo se quiere recomendar los casos que están valorados de forma más alta en las bases de casos de los distintos usuarios, sino que se busca un compromiso entre la similaridad del caso recomendado y la consulta, la similitud entre los perfiles del usuario que realiza la consulta y que hace la recomendación, la valoración del caso y, de forma muy importante y que supone una de las grandes aportaciones del caso práctico, que las valoraciones de los casos recomendados sean similares a las valoraciones que realizaría el usuario que realiza la consulta sobre esos casos.

5.3. Sistema de decisión fuzzy

En esta sección se expone el sistema de decisión fuzzy diseñado y utilizado para la toma de decisiones dentro del protocolo AMAL modificado expuesto en la Sección 5.2. Como se comentaba en esa sección, la misión del sistema de decisión fuzzy es sustituir a la lógica descriptiva utilizada en el protocolo AMAL original con el objetivo de poder utilizar el sistema deliberativo sobre casos que no admitan la utilización de lógicas descriptivas.

A continuación se detalla el sistema creado, realizando primero una breve introducción acerca de por qué se ha decidido utilizar un sistema fuzzy y qué decisiones se espera que resuelva dentro del protocolo de argumentación, pasando después a la descripción de cómo se ha modelado la información manejada por el sistema de decisión y terminando con una visión global del diseño del mismo.

Al final de este trabajo se incluyen dos apéndices: uno (Apéndice A) incluye el conjunto de reglas utilizado por el sistema fuzzy en cada uno de sus subsistemas descritos en la Sección 5.3.2. El otro (Apéndize B), incluye el conjunto de tipos definidos para posibilitar al sistema fuzzy el manejo de la información.

5.3.1. Introducción al sistema de decisión fuzzy

El principal objetivo del sistema de decisión fuzzy es el de sustituir el razonamiento sobre los casos que en el protocolo de argumentación original realizaba la lógica descriptiva. Gracias a ello, extendemos el abanico de casos que pueden ser tratados por nuestro sistema deliberativo imponiendo únicamente la restricción de que los casos deben estar valorados.

Esta valoración, viene a representar el grado de confianza entre el nodo del sistema distribuido y el caso residente en su base da casos. Cuanto más alto sea este valor, más de acuerdo estará el nodo con la información contenida en este caso y por el contrario, más en desacuerdo estará cuanto menor sea este valor. Por ejemplo, en un recomendador de música, este valor representará cuanto le gusta al usuario un disco o, por ejemplo, en un clasificador distribuido de plantas, una valoración alta de un caso representará que para ese usuario, las características de un caso concuerdan con la especie vegetal asignada al mismo.

¿Por qué utilizar un sistema de decisión fuzzy?

Las principales razones para utilizar un sistema de decisión basado en el lógica fuzzy son las siguientes:

- Eliminar las restricciones que supone tener que utilizar Lógicas Descriptivas sobre los casos. Como acabamos de comentar, la única restricción que existe sobre los casos para que pueda aplicarse el sistema de decisión fuzzy es que sean valorados.
- Poder realizar deliberaciones sobre bases de casos no nítidos. El protocolo de argumentación original sólo permitía la aplicación del protocolo sobre

casos nítidos, es decir, los casos pertenecían a una clase o no sin poder estar en medio de dos clases distintas.

- Nos abre la posibilidad de utilizar contraejemplos y defensas no absolutas contra los casos contra los que se presentan, es decir, que la credibilidad de estos contraejemplos y defensas sea parcial y que abra la posibilidad de deliberar acerca de su idoneidad.

- Buscamos un sistema que se parezca, en la medida de los posible, en la forma de razonamiento que realizaría un humano valorando distintas argumentaciones y alternativas antes de decidir sobre la validez de un caso.

- Queremos que las salidas obtenidas por el sistema de decisión presenten comportamientos suaves y no presenten grandes oscilaciones ante pequeñas variaciones de los parámetros de entrada.

¿Qué necesitamos del sistema de decisión?

Las funciones del sistema fuzzy son:

- Valorar un caso que un nodo agente del sistema recomienda a otro.

- Decidir cuando un caso de la base de casos de un agente constituye un contraejemplo contra la solución presentada por otro agente.

- Decidir si un contraejemplo presentado por un agente contra la solución de otro es aceptado o no.

- Decidir cuando un caso de la base da casos de un agente constituye una defensa para su solución.

- Decidir si una defensa es aceptada o no.

5.3.2. Diseño del sistema de decisión fuzzy

Existen cinco subsistemas distintos para dar solución a cada una de las necesidades planteadas. A continuación, se detalla cada uno de estos sistemas, exponiendo los argumentos que utilizan así como las salidas que se espera que generen.

Subsistema de evaluación de casos

El subsistema de evaluación de casos (Figura 5.8) tiene como objetivo dar respuesta a la primera de las necesidades de la Sección 5.3.1. Mediante este subsistema, un agente puede conocer en qué grado los casos de su base de casos se adaptan a una consulta. Es utilizado cuando los agentes del sistema distribuido quieren resolver la consulta que les han realizado utilizando su propia base de casos.

Necesita la siguiente información:

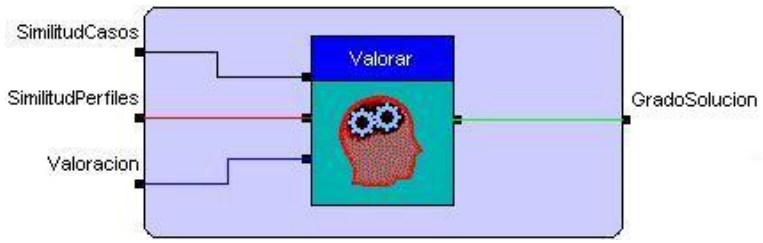

Figura 5.8: Subsistema de evaluación de casos

- La similitud entre el caso que se quiere evaluar y la consulta que se quiere resolver.
- La similitud entre los perfiles del nodo que está evaluando los casos y el nodo que realiza la consulta.
- La valoración del nodo sobre el caso que se está evaluando.

Como resultado de la evaluación del caso, el subsistema generará el grado con el que ese caso es una solución a la consulta realizada.

Gracias a este valor, un agente del sistema distribuido podrá evaluar todos los casos de su base de casos ante una consulta, realizando un ranking de los mismos. Los casos que mejor resuelvan esta consulta, serán aquellos que ocupen las primeras posiciones dentro del ranking realizado.

Subsistema de evaluación de contraejemplos

Para poder determinar cuando un agente del sistema distribuido puede presentar un contraejemplo contra la solución de otro agente, se utiliza el subsistema de evaluación de contraejemplos (Figura 5.9), que resuelve el segundo de los puntos de las necesidades de la Sección 5.3.1.

Este subsistema de razonamiento es utilizado por los agentes durante el proceso de argumentación cuando son poseedores del token (turno de palabra) y deben decidir si presentar un contraejemplo contra alguna de las soluciones del resto de agente que intervienen en el proceso. Gracias a este sistema, pueden buscar en su base de casos cual es el caso que constituye el contraejemplo más fuerte contra alguna de las soluciones.

Este sistema tiene como parámetros de entrada:

- La similitud entre el caso que se está evaluando y la solución contra la que se presenta.
- La similitud entre los perfiles del nodo que está evaluando el contraejemplo y el nodo que ha realizado la consulta.
- La valoración del nodo sobre el caso que se está evaluando.

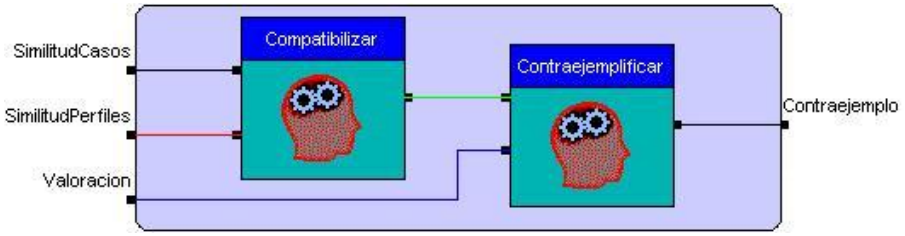

Figura 5.9: Subsistema de evaluación de contraejemplos

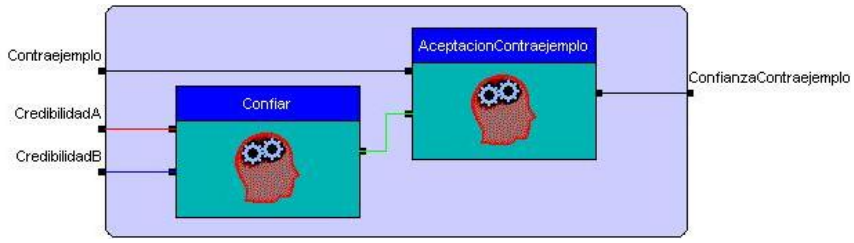

Figura 5.10: Subsistema de aceptación de contraejemplos

Como resultado, el subsistema devuelve el grado en el que ese caso constituye un contraejemplo contra la solución contra la que se está presentado.

Subsistema de aceptación de contraejemplos

El sistema de aceptación de un contraejemplo (Figura 5.10) da respuesta a la tercera de las necesidades planteadas en la Sección 5.3.1.

Gracias a él, el agente coordinador de un proceso deliberativo (el agente A_0 en la Sección 5.2.1), podrá decidir si acepta el contraejemplo presentado por un agente del sistema distribuido contra la solución de otro. En el momento en el que A_0 reciba un contraejemplo de uno de los agentes del proceso, deberá hacer uso de este subsistema para decidir si acepta el contraejemplo presentado.

Para poder realizar esta operación, el subsistema necesita:

- El grado de contraejemplo que supone este caso. Este valor es el valor calculado por el subsistema anterior. Le indica al agente en qué medida este caso supone un contraejemplo contra la solución contra la que se presenta.
- La credibilidad que el agente A, que presenta el contraejemplo, tiene con respecto al agente A_0, es decir, cuanto confía el agente A_0 en el agente que ha presentado el contraejemplo.
- La credibilidad que el agente B, que ha presentado la solución, tiene con respecto al agente A_0, es decir, cuanto confía el agente A_0 en el agente que ha presentado la solución.

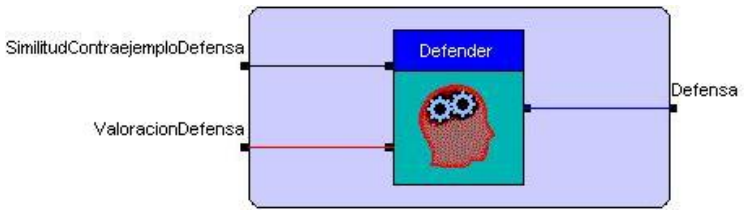

Figura 5.11: Subsistema de evaluación de defensas

Cuando es consultado el subsistema de aceptación de un contraejemplo, devuelve un valor que indica cuanto el agente A_0 confía en el contraejemplo que le han presentado. Se debe establecer un valor mínimo o umbral que indique el valor mínimo de confianza que tenemos en un contraejemplo para considerarlo válido. Este umbral debe ser un valor dependiente del sistema que se quiera implementar y por tanto deberá ser definido por el mismo.

Subsistema de evaluación de defensas

Para poder atender la cuarta de las necesidades presentadas en la Sección 5.3.1, se diseña el subsistema de evaluación de las defensas (Figura 5.11), que permite a los agentes del sistema difuso encontrar en su base de casos una defensa para defender alguna de sus soluciones durante un proceso deliberativo.

Ante la llegada de un contraejemplo a un agente contra una de sus soluciones, el agente recorrerá su base de casos en busca de la mejor defensa frente a ese contraejemplo. Para ello, evaluará todos sus casos mediante el sistema de evaluación de defensas, realizando un ranking con los casos. El caso que presente como defensa, será aquel que mayor valoración consiga por parte de este subsistema.

Para poder realizar la operación, el subsistema de evaluación de defensas necesita la información siguiente:

- La similitud entre el contraejemplo presentado y el caso de defensa que se quiere evaluar.
- La valoración del caso que se quiere utilizar como defensa.

Como hemos comentado, el subsistema de razonamiento nos devolverá el grado en el que el caso representa una defensa frente a ese contraejemplo.

Subsistema de aceptación de defensas

La ultima de las operaciones a resolver por parte del sistema de decisión difuso es la de decidir cuando una defensa es aceptada o no. El subsistema de aceptación de defensas (Figura 5.12) se encarga de dar respuesta a este problema. Cuando una defensa es enviada al agente A_0, a favor de una solución de una agente B y en

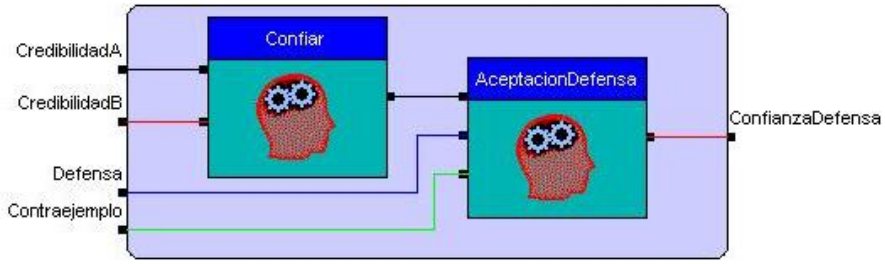

Figura 5.12: Subsistema de aceptación de defensas

contra de un contraejemplo presentado por un agente A, el agente A_0 deberá decidir si acepta el contraejemplo o, si por el contrario, acepta la defensa.

Para ello, el subsistema de aceptación de defensas utiliza la siguiente información:

- El grado de contraejemplo del caso presentado como tal. Este valor es el valor obtenido como resultado del subsistema de evaluación de contraejemplos.
- El grado de defensa del caso de defensa. Este valor es el valor obtenido como resultado del subsistema anterior.
- La credibilidad que el agente A, que presenta el contraejemplo, tiene con respecto al agente A_0, es decir, cuanto confía el agente A_0 en el agente que ha presentado el contraejemplo.
- La credibilidad que el agente B, que ha presentado la solución, tiene con respecto al agente A_0, es decir, cuanto confía el agente A_0 en el agente que ha presentado la solución.

Como resultado obtendremos el grado de confianza que el agente A_0 pone en la defensa. Como en el caso de subsistema de aceptación de contraejemplos, se debe fijar un umbral para la aceptación de la defensa. Este valor será dependiente del sistema concreto que se implemente. Si el valor devuelto por el subsistema de evaluación supera este umbral, la defensa será aceptada y por tanto, el contraejemplo desechado. Si por el contrario la defensa es rechazada, la solución de este agente se eliminará el proceso deliberativo.

5.4. Aprendizaje del modelo de confianza

En este capítulo presentamos el sistema de aprendizaje del modelo de confianza que hemos diseñado. En primer lugar realizaremos una introducción al sistema de aprendizaje en la Sección 5.4.1 describiendo en qué consiste el sistema de aprendizaje del modelo de confianza del sistema distribuido. En la Sección 5.4.2 estableceremos cómo podemos conocer los agentes que aportan cada una de las soluciones a las consultas y cómo saber como una solución llega hasta el agente que realiza la consulta. En la Sección 5.4.3 expondremos como modificamos los valores de la confianza entre los agentes del sistema para ajustarlos al modelo y por último, en la Sección 5.4.4 daremos una solución a los problemas en el sistema de aprendizaje.

5.4.1. Introducción

El aprendizaje del modelo de confianza entre los agentes del sistema CBR distribuido se basa en la actualización de los valores de confianza entre los agentes del sistema. El aprendizaje que exponemos en esta sección es un sistema de aprendizaje basado en un sistema de gestión de la confianza descentralizado, en el que la confianza es una relación asimétrica entre dos individuos, puesto que la confianza que un individuo A tiene sobre otro individuo B está basada en la validez de las soluciones que presente B a A y puede que estas no sean igual de válidas de A para B que de B para A.

Para poder realizar el aprendizaje del modelo de confianza necesitamos una realimentación por parte del usuario del sistema cuando se generen las soluciones. Esta realimentación puede ser de forma manual por parte del usuario indicando sus valoraciones sobre las respuestas que le ha realizado el sistema o bien de forma automática, siendo el sistema encargado de generar estas valoraciones (como realizaremos en el experimento del Capítulo 6).

5.4.2. Caracterización de la ruta seguida por las soluciones generadas

Una vez conocida la valoración realizada por un usuario ante una solución del sistema, podemos proceder a actualizar los valores de la confianza en la red. Para ello, es necesario que conozcamos el recorrido que una solución ha realizado en la red hasta llegar al destinatario. Optamos por resolver el problema de forma similar a la adoptada para conocer las trazas de las peticiones: utilizaremos una traza para representar el camino seguido por las recomendaciones desde el agente originario de ellas, hasta el destinatario de las mismas. En la Figura 5.13 podemos observar un ejemplo del camino seguido por una solución desde A_1 hasta A_0. Las flechas indican la dirección hacia la que se devuelven las soluciones. Las líneas continuas indican el camino que ha seguido la solución adoptada finalmente y las discontinuas casos que finalmente no han formado parte de la solución.

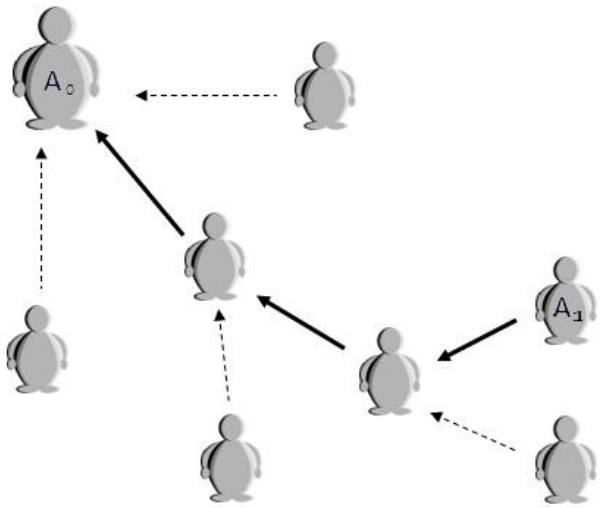

Figura 5.13: Camino de una solución en un sistema CBR distribuido

Es importante señalar que no es suficiente con las trazas ya presentes en las peticiones descritas en la Sección 5.1.3, puesto que éstas siguen un recorrido inverso al de las soluciones. Para un agente cualquiera de la red durante el funcionamiento del sistema CBR distribuido se tendrá información de:

- Camino que ha seguido la consulta desde el agente que la ha realizado hasta el agente actual (traza de la consulta).
- Camino que han seguido cada uno de los casos devueltos por el sistema CBR distribuido ante la consulta propagada por el agente actual (traza de las soluciones).

Por lo tanto, siguiendo esta especificación, tendremos que:

- En el agente que realiza la consulta, la traza de la petición únicamente incluirá a este agente, puesto que esta no ha seguido ningún camino hasta llegar a él, mientras que la traza de los casos presentará el camino completo seguido por estos hasta alcanzar a este agente.
- En los nodos que se encuentren al final del sistema CBR distribuido la traza de la consulta estará completa, mientras que la traza de las soluciones únicamente presentará el agente que la ha generado.

5.4.3. Modificación de los valores de la confianza entre agentes

En la sección anterior hemos expuesto cómo se identifica el camino seguido por una solución dentro de un sistema CBR distribuido. En esta sección pasaremos a establecer como se modifican los valores de la confianza entre agentes para

que el sistema pueda aprender el modelo de confianza asociado al sistema CBR distribuido.

Dado que en un sistema CBR distribuido las soluciones siguen un camino por la red antes de llegar al agente que ha creado la consulta, debemos establecer en qué medida son modificados los valores de las credibilidades de los agentes en función de su posición dentro del camino de la solución. Con esto se pretende que la confianza del agente que ha generado la solución se vea afectada en mayor medida que la confianza que uno de los agentes que simplemente la ha transmitido en la red. De esta forma, se producirá una modificación progresiva de los valores de la confianza en la ruta de la solución, que será más fuerte cuanto más cerca del agente que genera la solución estemos, y más débil cuanto más nos alejemos de él y más nos acerquemos al agente generador de la consulta.

La justificación de este tipo de modificación dentro de los sistemas distribuidos se debe a que las soluciones se realizan en función de la similitud de los perfiles de los agentes. Por ello, si un agente ha realizado una solución válida basándose en su perfil, podemos suponer que ante la llegada de otra consulta de otro perfil similar también va a aportar una buena solución. Los valores de la confianza van siendo modificados más débilmente según nos alejamos de este agente debido a que se van teniendo en cuenta más casos procedentes de otros agentes, por lo que no podemos asegurar con la misma fuerza como lo hacíamos anteriormente que vayamos a generar otra buena solución.

Para determinar si una solución es adecuada fijaremos lo que llamamos *rango de similitud*. Este rango determina si la valoración de la solución por parte del usuario se ajusta lo suficiente a la valoración de la solución. El *rango de similitud* se calcula en función de un parámetro σ propio de cada sistema, que fija la anchura del rango de similitud. La fórmula que calcula el rango de similitud es:

$$rangoSimilitud = \sigma - |V_{real} - V_{caso}| \quad \text{donde:}$$

- σ**:** Amplitud del rango de similitud.
- V_{real}**:** Valoración real del usuario de la solución.
- V_{caso}**:** Valoración de la solución (solución existente en el caso).

Para aquellas soluciones en las que el rango de similitud sea positivo, la confianza entre los agentes implicados en la solución se verá reforzada positivamente según la fórmula 5.3 mientras que si es negativo, los valores de la confianza se verán decrementados siguiendo la fórmula 5.4.

Si $rangoSimilitud >= 0$:

$$\Delta Confianza = \delta \cdot \left(\frac{l_n}{l_t \cdot n} \cdot \frac{rangoSimilitud}{\sigma} \right) \tag{5.3}$$

Si $rangoSimilitud < 0$:

$$\Delta Confianza = -\delta \cdot \left(\frac{l_n}{l_t \cdot n} \cdot \frac{rangoSimilitud}{\sigma - (V_{max} - V_{min})} \right) \quad \text{donde:} \tag{5.4}$$

- σ**:** Amplitud del rango de similitud.
- δ**:** Factor de aprendizaje de rango [0,1].
- l_n**:** Longitud de la traza de solución en el agente actual.
- l_t**:** Longitud total de la traza de solución.
- n**:** Número de soluciones proporcionadas por el sistema CBR distribuido.
- V_{max}**:** Valoración máxima posible.
- V_{min}**:** Valoración mínima posible.

5.4.4. Soluciones iguales por parte de varios agentes

Durante el proceso de solución de las consultas puede darse la situación de que un mismo caso sea propuesto como solución por parte de dos agentes distintos. Por ejemplo, esta situación se produce cuando durante el protocolo de argumentación de la Sección 5.2 se da un acuerdo entre los agentes acerca de la solución a una consulta.

Para dar solución a este problema, no bastará con una única traza para almacenar la procedencia de la solución, sino que necesitamos tantas trazas como agentes de la red hayan proporcionado la misma solución. De esta forma, podremos identificar cada uno de los nodos que haya aportado la solución.

Para esta situación, los valores de los incrementos de la confianza se calcularán siguiendo también las fórmulas 5.3 y 5.4, utilizando como traza la parte de la traza de la solución adecuada en cada caso.

5.5. Conclusiones

En este capítulo, hemos desarrollado cuatro aspectos fundamentales para la construcción de un sistema CBR distribuido bajo el marco práctico desarrollado en el Capítulo 4:

1. **Mecanismo de propagación de la consulta:** Hemos desarrollado un mecanismo de propagación de la consulta completo y que permite una iniciación múltiple.

2. **Protocolo de argumentación**: Hemos modificado el protocolo AMAL tradicional definido en [2] permitiendo su uso sobre todo tipo de casos con la única premisa de que sean valorados.

3. **Sistema de razonamiento difuso:** Hemos diseñado y construido un sistema de generación de argumentaciones difuso para el protocolo de argumentación AMAL modificado.

4. **Mecanismo de aprendizaje del modelo de confianza:** Hemos diseñado un sistema de aprendizaje capaz de inferir los valores del modelo de confianza del sistema a través de las valoraciones de los usuarios sobre las soluciones aportadas.

La caracterización del sistema instanciado, siguiente las características extraidas en el Capítulo 3, y resumidas en la Figura 3.3 es la siguiente:

- **Número de agentes:** Se trata de un sistema diseñado para su utilización en sistemas grandes. No obstante, su aplicación es válida también para sistemas medianos e incluso pequeños.

- **Tamaño de las bases de casos y tipos de casos:** Dependerá de la aplicación CBR distribuida concreta que se implemente con este sistema.

- **Solapamiento de las bases de casos:** El sistema está diseñado para bases de casos solapadas, pudiendo utilizarse también sobre bases de casos iguales o no solapadas.

- **Topología de red:** El sitema admite la utilización de cualquier topología de red, si bien al estar pensado para sistemas grandes lo más razonable sería utilizarlo sobre redes con topologías de árbol o de redes sociales (u otras tologías de red importadas de otro tipo de redes, sin necesidad de que sean sociales).

- **Construcción del modelo de confianza:** No afecta al sistema diseñado por lo que podrá utilizarse cualquiera de las dos posibilidades y dependerá de la aplicación concreta implementada.

- **Gestión del modelo de confianza:** El sistema de aprendizaje de los valores de confianza realiza una gestión descretalizada de ésta (como se verá en la Sección 5.4), por lo que asociadas a cada conexión de red entre dos agentes A y B existen dos valores de confianza, uno para la confianza de A en B y otro para la confianza entre B y A.

- **Composición de los resultados:** El proceso de composición de resultados escoge n casos de las soluciones de cada agente, como se verá en la Sección 5.2, tantos como recomendaciones solicite la consulta.

- **Protocolo de argumentación:** Se utiliza el protocolo de argumentación AMAL modificado, que describiremos en la Sección 5.2, en el que se hace uso de agentes heterogéneos.

- **Tipo de razonamiento CBR:** No está limitado por el sistema diseñado y dependerá de la aplicación concreta implementada con el sistema.

- **Iniciación de las consultas:** El sistema de propagación de las consultas, expuesto en la Sección 5.1, permite la iniciación múltiple de estas. De esta forma, también se podían construir sistemas con iniciación única.

- **Propagación de las consultas:** El sistema diseñado utiliza una propagración completa de las consultas a nodos los nodos enlazados a aquel que la propaga, como se ha detallado en la Sección 5.1.

Una vez desarrollados los marcos teóricos y prácticos de los Capítulos 3 y 4 respectivamente, así como la instanciación a un sistema distribuido concreto realizada en este capítulo, pasaremos en el Capítulo 6 a la exposición de los experimentos realizados sobre el sistema para probar su correcto funcionamiento así como el rendimiento alcanzado.

Capítulo 6

Caso de estudio: Implementación de un recomendador musical distribuido

Una vez completados los objetivos planteados al comienzo de este proyecto en la Sección 1.1 únicamente falta por comprobar el funcionamiento del marco práctico y la instanciación del marco teórico sobre un sistema real. Para ello, realizamos el siguiente caso de estudio, con objetivo de probar el rendimiento del sistema.

La organización del capítulo actual para la exposición del caso de estudio desarrollado es la siguiente: en primer lugar se realiza una introducción a fin de comprender las motivaciones que nos han llevado a elegir un recomendador musical como caso de estudio, describiendo el problema al que nos enfrentamos. Posteriormente, en la Sección 6.2, se mostrará la implementación realizada del recomendador musical sobre la base existente expuesta en el Capítulo 4. Una vez explicado esto, describiremos en la Sección 6.3 cómo ha tenido lugar el desarrollo del experimento acabando con la exposición de los resultados obtenidos durante las pruebas y las conclusiones obtenidas de ellas, en las secciones 6.4 y 6.5 respectivamente.

6.1. Introducción

En esta sección realizamos una introducción al caso de estudio realizado a fin de que el lector comprenda el marco en el que es desarrollado, las motivaciones para hacerlo y los resultados generados.

En la Sección 6.1.1 expondremos cuál es el objetivo de este caso de estudio, explicando en la Sección 6.1.2 cuál ha sido nuestra motivación a la hora de seleccionar un recomendador musical como experimento.

6.1.1. Objetivo

Nuestro objetivo a la hora de realizar este experimento es comprobar el correcto funcionamiento de un sistema CBR distribuido construido utilizando el marco práctico de sistemas CBR distribuidos para jCOLIBRI expuesto en el Capítulo 4 así como la instanciación del marco teórico expuesta dentro del Capítulo 5. Gracias a este experimento y los resultados generados esperamos poder establecer que el sistema cumple con las expectativas generadas, permitiendo el desarrollo de sistemas CBR distribuidos desde la base de jCOLIBRI distribuido desarrollada.

Una vez comprobado el correcto funcionamiento del sistema CBR distribuido, esperamos poder demostrar de forma empírica las mejoras introducidas por las distintas aportaciones realizadas dentro del Capítulo 6.1.1. En concreto, demostraremos que el sistema de razonamiento fuzzy diseñado y utilizado dentro de un sistema CBR distribuido supone un mayor rendimiento del sistema mejorando los resultados obtenidos de un sistema CBR distribuido sin un sistema de generación de argumentaciones.

Además, con la realización de este caso de estudio pretendemos mostrar la construcción de sistemas CBR distribuidos con la nueva plataforma jCOLIBRI distribuido. De esta forma, existirá un primer sistema CBR distribuido que parte de ella y servirá como guía para el desarrollo de posteriores sistemas CBR distribuidos.

6.1.2. ¿Por qué un recomendador musical?

En primer lugar, surgió la posibilidad del realizar este caso de estudio mediante un recomendador. Los recomendadores destacan entre otras cosas por la simplicidad de los casos manejados en los que no se requiere un proceso complejo de adaptación de los casos. Esto, unido con que los casos son valorados (las valoraciones de los casos son las valoraciones de los usuarios sobre el producto representado por el caso), el único requisito impuesto por el protocolo AMAL modificado, los hacía idóneos para su uso como caso práctico.

Otro de los factores que llevó a la elección de un sistema recomendador es la posibilidad de crear factores de correlación entre los usuario a través del coeficiente de Pearson [34] utilizando sus bases de datos. Gracias a este factor, podemos determinar cuánto de parecidos son dos usuarios y por lo tanto en qué medida son valoraciones sobre los casos se parecen. De esta forma, podemos crear enlaces entre los usuarios más parecidos del sistema, creando una estructura de red basada en este factor. La red construida será por tanto una red en la que los nodos conectados serán aquellos cuya similitud sea mayor, simulando por tanto una red social basada en el ámbito del recomendador.

También influyó a la hora de elegir este tipo de sistema CBR la gran expansión y proliferación de este tipo de sistemas, tanto académica como comercialmente. La gran variedad de sistemas existentes (descritos en la Sección 2.2) hace posible la elección de un número importante de variables de diseño que permiten que

diseñemos un sistema que se ajusta a las necesidades de este experimento para demostrar los objetivos expuestos en la sección anterior.

La elección de un recomendador musical frente a un recomendador de cualquier otro tipo de productos responde a la facilidad de acceso de información acerca de discos así como la existencia de bases de datos de estos que poder integrar dentro de nuestro sistema. También influyó el conocimiento general que cualquier usuario tiene del mundo de la música, que le permite conocer el campo sin necesidad de ser un experto, con lo que el uso del sistema y la evaluación de los resultados resulta más sencilla.

6.1.3. Descripción del problema

Una vez planteado el ámbito en el que surge el caso práctico de este proyecto, realizaremos una descripción detallada del problema.

Nuestra intención es simular un escenario lo más real posible. Cada usuario dispondrá de una base de datos formada por un subconjunto de 50 discos pertenecientes a un catálogo inicial de 262 discos. Cada uno de estos discos tendrá los campos: título, autor, año, nacionalidad, género y valoración del usuario. Existen 11 géneros diferentes para los discos: Hip Hop, Rock Clasico, Folk, Rhythm And Blues, Pop, Flamenco, Clasica, Techno, Chill Out, House, Nuevas Tendencias del Rock. Asociados a estos 11 géneros diferentes se crean 11 perfiles de usuario distintos, cada uno vinculado a un género distintos.

Los perfiles de los usuarios del sistema se crearán de forma aleatoria, escogiendo uno de los estilos de la lista anterior. Cada uno de los discos de la lista anterior será valorado en función del perfil asignado al usuario y de una distribución probabilista. De esta forma, un usuario con un perfil, por ejemplo, de Pop, valorará los discos de Pop con una valoración alta, variando un porcentaje de la nota de acuerdo con una determinada probabilidad. Todos aquellos discos de géneros relacionados con el Pop serán valorados también de forma alta, pero no tanto como los del género acorde con su perfil. Los discos de su base de casos de géneros que no tienen relación con el suyo obtendrán puntuaciones bajas. Además, para incluir un factor de incertidumbre dentro del sistema y darle un mayor realismo, un porcentaje de los discos se valoran erróneamente, de forma opuesta al perfil, a fin de simular las variaciones existentes en los usuarios reales.

Cada uno de los usuarios del sistema estará representado por un agente CBR que gestionará su base de casos y lo representará en el sistema CBR distribuido. Cuando un usuario realiza una consulta, es decir, está solicitando que se le recomiende algún disco, el agente asociado a él se encargará de distribuir su consulta por el sistema distribuido haciéndole llegar el resultado de la recomendación.

Nuestras premisas a la hora de realizar el experimento son:

- El sistema fuzzy de generación de argumentaciones supondrá una mejora de los resultados del sistema frente a aquel sistema que no lo utilice.

- Un recomendador distribuido con una topología de red social obtendrá mejores resultados que un sistema con una topología aleatoria. Esto se debe a que las confianzas calculadas entre los usuarios a través del coeficiente de Pearson en una red aleatoria serán mayores que las calculadas en una red aleatoria. Por tanto, dado que las confianzas entre agentes serán mayores, las argumentaciones de cada uno de los agentes del sistema serán más fuertes y por lo tanto serán más tenidas en cuenta durante los protocolos de argumentación, mejorando los resultados del sistema.

- Los resultado de los sistemas con mayor número de nodos serán mejores que los resultados obtenidos con sistemas iguales pero con un menor número de nodos, dado que habitualmente el aumento del número de individuos en el sistema va a suponer ampliar el número de casos totales del sistema y el número de agentes parecidos al agente que realiza las consultas.

Para probar nuestras premisas, crearemos seis redes distintas, dos de ellas con 10 nodos, otras dos con 20 nodos y las dos restantes con 50 nodos. De cada una de las redes que tiene el mismo número de nodos una de las dos será construida creando nodos de forma aleatoria y la otra será creada escogiendo los enlaces entre aquellos nodos cuyo coeficiente de Pearson sea mayor. Para cada una de estas seis redes probaremos una serie de consultas, primero utilizando agentes que realicen argumentaciones gracias al protocolo difuso de la Sección 5.3 y después utilizando agentes que no realicen argumentaciones, comparando después los resultados obtenidos. De esta forma podremos comprobar nuestras tres premisas:

- Comprobaremos si el sistema de argumentación obtiene resultados mejores que el sistema que no utiliza argumentaciones mediante la comparación de los resultados obtenidos para redes de 10, 20 y 50 nodos, tanto creadas de forma aleatoria como mediante el coeficiente de Pearson.

- Comprobaremos si las redes construidas mediante el coeficiente de Pearson simulando redes sociales obtienen mejores resultados que los obtenidos por redes aleatorias, tanto si utilizamos un protocolo de argumentación como si no lo usamos.

- Podremos comprobar si la variación en el número de nodos de sistemas iguales influye en el resultado obtenido por los sistemas CBR distribuidos.

Además, comprobaremos el efecto del *"ensemble efect"* expuesto en [21], viendo como los resultados generados por los nodos de forma individual son peores que los generados por el sistema distribuido.

Los modelos de confianza entre los nodos del sistema CBR distribuido se crearán en función del coeficiente de Pearson existente entre ellos, ya sea una red creada de forma aleatoria o creada a través de este mismo valor. De esta forma, aquellos nodos que tengan un alto coeficiente de Pearson tendrán un alto valor de la confianza entre ellos. Se deriva del mecanismo de construcción de las redes, que

aquellas que han sido construidas mediante este coeficiente tengan valores altos de la confianza entre los nodos, lo cual es consecuente con el hecho de que estamos intentando simular una red social gracias al coeficiente de Pearson y los valores de confianza en las redes sociales tienden a ser altos, debido a que los usuarios están unidos en cuanto a sus gustos. Por otro lado, aquellas redes que hayan sido construidas de forma aleatoria tenderán a tener valores de confianza entre sus nodos más bajos.

Con esto, queda planteado nuestro experimento, pasando ahora a la descripción de la implementación realizada para la construcción de sistema recomendador musical:

6.2. Implementación del recomendador

En el Capítulo 4 hemos realizado una instanciación del marco teórico del Capítulo 3 utilizando el marco práctico para la construcción de sistemas distribuidos sobre jCOLIBRI del Capítulo 4. Es importante destacar que esta instanciación es independiente del tipo de sistema CBR que luego se desee utilizar con el sistema distribuido diseñado, y por ello, permite la construcción de cualquier sistema CBR cuyos casos estén valorados, y en este caso, permite la construcción del sistema recomendador de nuestro experimento.

Para poder crear el sistema recomendador, gracias a todo el sistema general construido, únicamente debemos implementar la aplicación CBR concreta del sistema recomendador. Para ello, utilizamos la clase *StandardCBRApplication*, que como hemos visto en la Sección 4.1.2 es la aplicación CBR que manejan los agentes del sistema multiagente. Por lo tanto, únicamente debemos centrarnos en el desarrollo de esta aplicación para la construcción de un sistema CBR distribuido. Cabe destacar por ello, la facilidad con la que podemos implementar cualquier sistema CBR distribuido usando el sistema diseñado, sin más que diseñar este sistema CBR estándar, donde jCOLIBRI 2 nos facilita todas las herramientas que podamos utilizar. Hemos simplificado con se puede observar, la construcción de un sistema CBR distribuido a la simple construcción de un sistema CBR tradicional como estábamos haciendo hasta ahora.

6.3. Desarrollo del experimento

Una vez planteadas nuestras premisas expuestas en la Sección 6.1.3, es hora de demostrarlas empíricamente. Para ellos, debemos desarrollar un experimento que nos permita medir el rendimiento de cada uno de los sistemas planteados en esa misma sección.

En este punto surge el siguiente problema: ¿Cómo medimos el rendimiento de un sistema recomendador?

No es una pregunta de fácil respuesta, y menos para un sistema creado de manera artificial como el del experimento en el que el usuario no nos puede indicar su

grado de satisfacción con las recomendaciones realizadas. Debemos encontrar por tanto, un método que nos permita conocer la precisión de cada una de las recomendaciones realizadas sin la intervención humana.

En primer lugar podemos pensar en medir el rendimiento del sistema recomendador a través de las valoraciones de los discos que devuelve: cuanto mejor valorado esté un disco recomendado, mejor será la recomendación que se realiza. Como demostraremos en la Sección 6.4, esto no es cierto. Si así fuera, bastaría con elegir los discos mejor valorados que más se ajusten a la consulta de todas las bases de casos y devolverlos. Imaginemos un usuario (r) con un perfil completamente distinto al usuario (c) que realiza la consulta. Supongamos que el usuario r ha valorado todos los discos de su base de casos con la máxima valoración posible. ¿Significa eso que las mejores recomendaciones para el usuario c son las del usuario r? Evidentemente no. Dado que los perfiles son completamente distintos también lo son sus gustos, luego no le van a gustar las recomendaciones que se le realizaran.

En segundo lugar, como consecuencia de la anterior medida del rendimiento propuesta, podemos proponer como medida de rendimiento la diferencia entre la valoración de la recomendación realizada por el sistema CBR distribuido y la valoración del usuario sobre el caso recomendado. Como hemos comentado al comienzo de esta sección, no disponemos de un usuario real, por lo que debemos encontrar un método para conseguir estas valoraciones. Para ello, podemos utilizar el mismo sistema de valoración de los casos que hemos utilizado para construir las bases de casos de cada usuario expuesto en la Sección 6.1.3. Con esto, obtendremos una valoración del usuario sobre los casos que le han recomendado, con lo que podremos obtener las diferencias con respecto a las valoraciones que habían realizado los usuarios que han recomendado esos casos. ¿Basta por tanto con medir las diferencias entre las valoraciones realizadas por los usuarios que recomiendan los casos y las valoraciones del usuario que recibe estas recomendaciones?. La respuesta es no. De forma intuitiva, ¿sería mejor una recomendación con una valoración baja de r y que c valore como baja o una recomendación con una valoración muy alta de r que c valore como alta?. A pesar de que la diferencia entre las valoraciones bajas sería cero, resultaría más interesante recomendar el segundo de los casos a pesar de que existe una mayor diferencia entre la valoración de r y la valoración de c, puesto que estamos construyendo un recomendador y es preferible recomendar casos que le gusten al usuario.

Como conclusiones de estas dos posibles medidas del rendimiento obtenemos que debemos establecer un compromiso entre ambas. Debemos intentar maximizar la valoración de los casos que recomendamos al usuario, puesto que queremos que sean casos que sean de su agrado y debemos minimizar la diferencia entre la valoración del caso y la valoración real del usuario, puesto que deben ser casos que realmente le gusten. Por ello, utilizaremos como medida del rendimiento la

siguiente fórmula:

$$rendimiento = \frac{val_{rec}(d)}{1 + |val_{rec}(d) - val_{real}(d)|}$$

donde *d* es el caso recomendado que se está evaluando, *val_{rec}(d)* es la valoración del caso realizada por el usuario que realiza la recomendación (*r*) y *val_{real}(d)* es la valoración del caso que realizaría el usuario que recibe la recomendación (*c*).

Analizando la fórmula del rendimiento podemos observar que esta será mayor cuanto mayor sea el término *val_{rec}(d)*, es decir cuanto mayor sea la valoración del casos realizada por el usuario que realiza la recomendación, mientras que será menor cuanto mayor sea el término $1 + |val_{rec}(d) - val_{real}(d)|$, es decir, cuanto mayor sea la diferencia entre la valoración del caso y la valoración real del usuario. El factor $1 + |...|$ se introduce para evitar que el denominador pueda ser 0 si las valoraciones de ambos usuarios son iguales. Los valores del rendimiento están acotados entre:

$$\left[\frac{min}{1 + (max - min)}, max\right]$$

En concreto, en nuestro sistema recomendador las valoraciones oscilan entre 0 y 10, por lo que el rango del rendimiento será $[0, 10]$.

Una vez establecida la medida de rendimiento para el sistema recomendador, construimos un conjunto de consultas que serán utilizadas para probar el sistema. Este conjunto ha sido construido de forma aleatoria, tanto el número de campos que se definen como el contenido de estos campos, utilizando todo el rango de géneros, autores, años y nacionalidades posibles. El objetivo de la creación aleatoria de las consultas que utilizaremos en las pruebas del sistema es no probar únicamente aquellos autores y géneros que son del gusto de los usuarios que van a lanzar estas consultas, sino también probar consultas cuyas recomendaciones más similares no son del agrado del usuario para probar como responde el sistema. También nos servirá para probar el correcto funcionamiento del recomendador independientemente del tipo de consulta realizada.

Para realizar el experimento, lanzaremos cada una de consultas construidas (un total de 20) en un mismo nodo de cada una de las redes descritas en la Sección 6.1.3. Por cada consulta solicitaremos un total de 5 recomendaciones, para cada una de las cuáles calcularemos el rendimiento del sistema y posteriormente realizaremos su media, calculando por tanto el rendimiento medio de las recomendaciones para cada una de las consultas planteadas.

Además, calcularemos la similitud entre las recomendaciones obtenidas y las consultas realizadas, a fin de comprobar cuál es la similitud entre las consultas realizadas y las recomendaciones devueltas para cada uno de los sistemas que componen este experimento.

Para comprobar el *"ensemble efect"*, cada uno de los nodos de los seis diferentes tipos de redes resolverán de forma individual el conjunto de consultas, calculando posteriormente la media de sus rendimientos. De esta forma, obtendremos el rendimiento medio de los sistemas resolviendo de forma individual las consultas.

Una vez descrito como va a ser llevado a cabo el experimento, y tras su ejecución, comentamos los resultados obtenidos en la siguiente sección.

6.4. Resultados del experimento

Una vez desarrollado el experimento según lo expuesto en la Sección 6.3, nos ocuparemos en esta sección de la exposición de los resultados obtenidos por los diferentes sistemas.

A continuación se muestran seis Figuras con los resultados obtenidos por cada una de las redes probadas. Como hemos explicado en la Sección 6.1.3, hemos probado tres tamaños de red distintos, 10 nodos, 20 nodos y 50 nodos. Para cada una de estos tamaños de red hemos construido dos redes distintas: una red construida mediante la creación de enlaces aleatorios entre los usuarios y otra construida mediante la creación de enlaces de red basados en el coeficiente de Pearson. Cada una de las Figuras incluye en el pié a qué sistema corresponde.

Recordemos que, dentro de cada una de estas redes hemos probado tres sistemas:

- **Sistema Fuzzy:** Los agentes utilizan el protocolo de argumentación AMAL de la Sección 5.2 junto con el sistema de generación de argumentaciones fuzzy de la Sección 5.3. Este sistema estará representado en las Figuras por las líneas de color azul.

- **Sistema Estándar:** Los agentes utilizan el protocolo de argumentación AMAL de la Sección 5.2, pero no el sistema de generación de argumentaciones fuzzy de la Sección 5.3, por lo que no pueden presentar argumentaciones. Este sistema estará representado en cada una de las Figuras que se muestran a continuación por las líneas de color rojo.

- **Sistema Individual:** Los agentes resolverán las consultas utilizando su propia base de casos como si de un sistema CBR tradicional se tratara. Este sistema, estará representado por las líneas de color verde.

En cada una de las Figuras se incluyen tres gráficos de la misma red:

- **Primer gráfico:** En este gráfico se visualizan las valoraciones medias de las recomendaciones realizadas por el sistema (las líneas continuas en las gráficas) frente a las valoraciones medias reales realizadas por el usuario sobre los casos recomendados (las líneas punteadas) para cada una de las consultas realizadas en cada uno de los tres sistemas anteriores.

- **Segundo gráfico:** Visualizamos la similitud de las recomendaciones realizadas por el sistema frente a la consulta realizada para los tres sistemas anteriores.

- **Tercer gráfico:** Visualizamos el rendimiento de los sistemas anteriores según la fórmula expuesta en la Sección 6.3. El valor medio del rendimiento se visualizará como una línea discontinua en esta gráfica para poder ver qué sistema está por encima de los demás.

Una vez que disponemos de toda la información necesaria para interpretar los resultados, podemos pasar a su visualización.

6.4.1. Red aleatoria de 10 nodos

En la Figura 6.1 se muestran los resultados correspondientes a una red de 10 nodos generada de forma aleatoria.

De la primera de las gráficas podemos deducir que los casos recomendados por el sistema no se ajustan al perfil del usuario que realiza las consultas. La diferencia entre las valoraciones de los casos recomendados por los distintos sistemas (líneas continuas) y las valoraciones reales realizadas por el usuario sobre los casos que se le han recomendado es muy grande, lo cuál nos indica esta falta de precisión en el sistema. Esto se puede deber fundamentalmente a:

- El tamaño de la red es demasiado pequeño para el número de perfiles existentes. Dado que existen 11 perfiles posibles diferentes y la red únicamente posee 10 nodos, es muy posible que no exista en la red ningún usuario lo suficientemente parecido al usuario que recibe las recomendaciones como para realizar recomendaciones de calidad

- El número de casos del sistema no es lo suficientemente grande. Aunque existan usuarios con perfiles parecidos, si la base de casos no es lo suficientemente grande para contener un caso bien valorado que pueda ser válido para el usuario objeto de las recomendaciones y lo suficientemente similar a la consulta, el sistema no podrá devolver recomendaciones válidas

Observando la gráfica que representa los valores de la similitud entre los casos recomendados y las consultas, observamos que esta similitud es prácticamente idéntica para los tres sistemas probados sobre esta red.

De la misma forma, la gráfica del rendimiento nos muestra como apenas existen diferencias, no superando ninguno de los dos sistemas un rendimiento igual a 2.

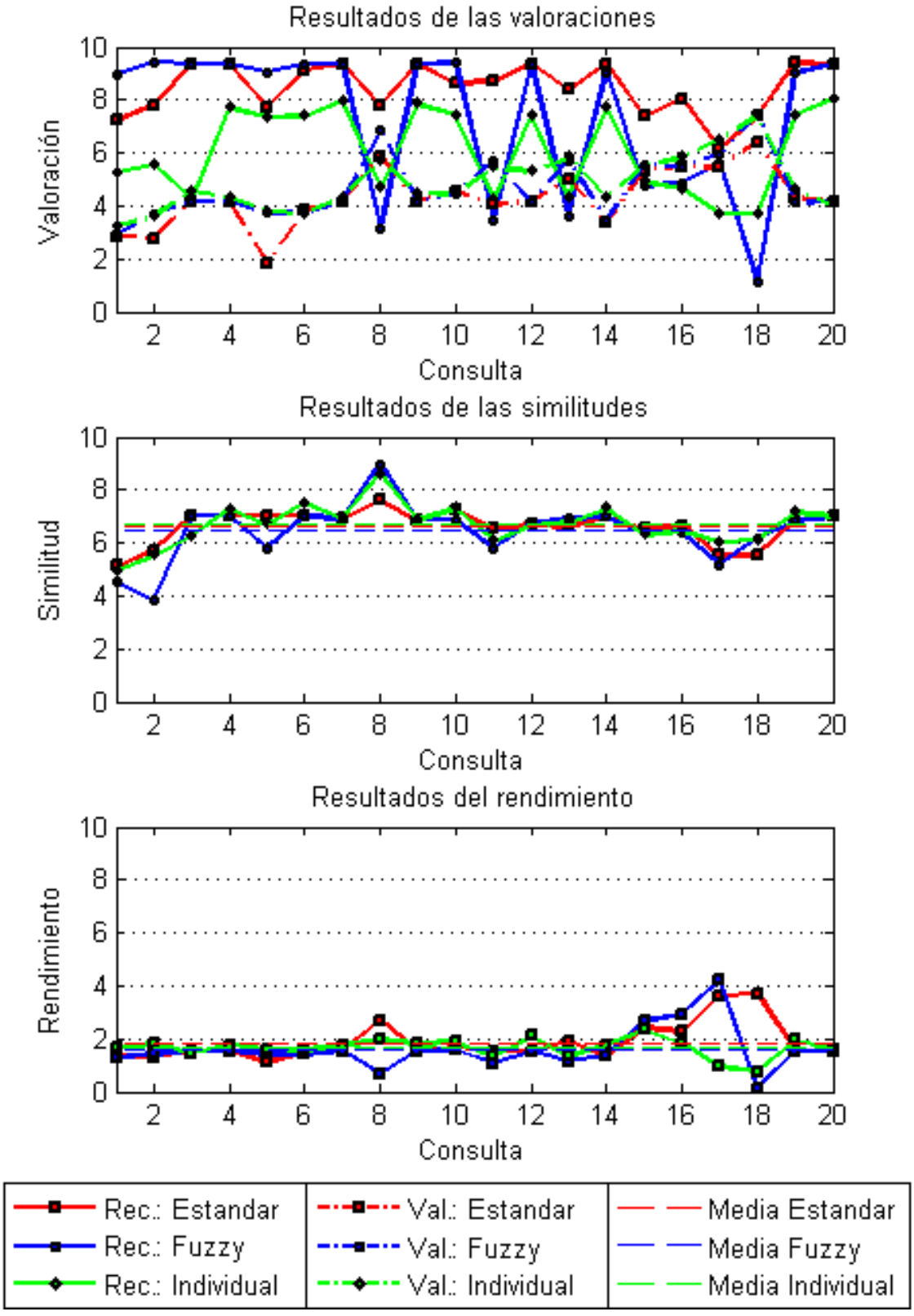

Figura 6.1: Resultados del experimento con una red aleatoria de 10 nodos

6.4.2. Red de Pearson de 10 nodos

La Figura 6.2 muestra los resultados obtenidos por una red de 10 nodos construida mediante el coeficiente de Pearson simulando una red social.

En la primera de las gráficas de esta Figura podemos observar como se ha producido un ligero decremento de la diferencia existente entre las valoraciones de los casos recomendados y las valoraciones reales en el sistema Fuzzy. Esto se debe a que los valores de confianza entre los nodos del sistema han aumentado y por ello aportan información al sistema de argumentación de tal forma que se pueden realizar argumentaciones mejores que las que se realizaban en el sistema aleatorio. Esto se debe a que únicamente podemos creer las argumentaciones de los nodos en los que confiemos, y en la red aleatoria no existen.

Observando la segunda gráfica vemos que la similitud entre las recomendaciones y las consultas sigue siendo igual entre los tres sistemas e igual a la obtenida por la red de 10 nodos construida de forma aleatoria.

Si bien las diferencias de rendimiento mostradas por la tercera de las gráficas siguen siendo muy pequeñas, podemos ver observando los valores medios de todas las consultas (líneas horizontales discontinuas), que las medias empiezan a distanciarse y que es el sistema Fuzzy el que mejores resultados está obteniendo, por delante del sistema Estándar, y por último, del sistema Individual.

Cabe destacar que los resultados del sistema Individual son los mismos para aquellas redes que tengan el mismo número de nodos, puesto que las conexiones no les afectan y los nodos siguen siendo los mismos.

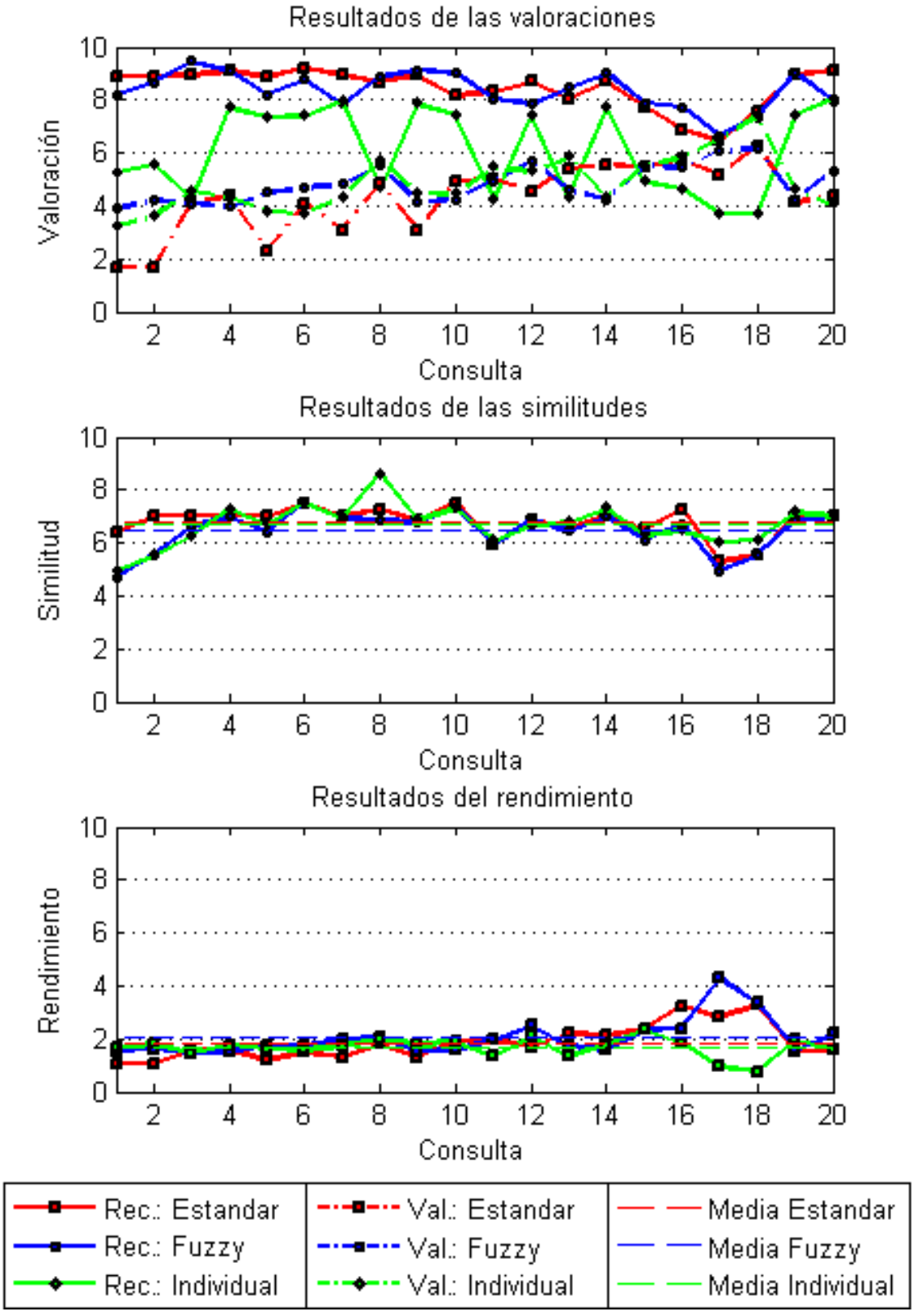

Figura 6.2: Resultados del experimento con una red de Pearson de 10 nodos

6.4.3. Red aleatoria de 20 nodos

En la Figura 6.3 podemos observar los resultados obtenidos con una red aleatoria de 20 nodos.

A pesar del aumento del número de nodos en el sistema, podemos observar en la primera de las gráficas como las diferencias entre las valoraciones de las recomendaciones realizadas y las valoraciones reales del usuario sigue siendo demasiado grande. Se debe de nuevo a los motivos expuestos en la Sección 6.4.1, la falta de perfiles similares o el tamaño de las bases de casos, o bien ambas.

De nuevo observamos como la similitud entre las consultas y los casos recomendados se mantiene prácticamente igual para los tres sistemas, y similar a la obtenida por las dos redes anteriores.

De la tercera gráfica podemos observar como esta vez, a diferencia de los resultados obtenidos en la red aleatoria de 10 nodos de la Sección 6.4.1, el rendimiento del sistema Fuzzy está por encima del rendimiento de los otros dos sistemas probados. De esto podemos extraer que, a pesar de que las diferencias entre las valoraciones de las recomendaciones realizadas y las valoraciones reales del usuario siguen siendo muy grandes, estas han disminuido, debido a la introducción de un mayor número de casos o de perfiles similares o parecidos. Por lo tanto, en cuanto existen los suficientes casos y perfiles el sistema Fuzzy diseñado es capaz de mejorar los resultados de los otros dos sistemas gracias a su capacidad de argumentación.

Sorprendentemente los resultados del sistema Estándar están por debajo de los resultados obtenidos por la media de los resultados individuales. Esto se debe fundamentalmente a dos factores:

- Al factor de incertidumbre introducido a la hora de crear las bases de datos de los nodos y que el sistema Estándar no es capaz de rechazar. Gracias al sistema de argumentación, estos casos son contraejemplificados en el sistema Fuzzy, sobretodo para la red de Pearson por unos valores de confianza entre agentes más altos.

- A la estructura de red aleatoria, que está haciendo que los resultados generados no sean válidos.

Es importante extraer como conclusiones tras analizar los resultados de esta red que se han mejorado los resultados obtenidos en la red aleatoria de 10 nodos de la Sección 6.4.3 siguiendo la tercera de las premisas expuestas en la Sección 6.1.3 al comienzo del proyecto. También extraemos que, a pesar de la aleatoriedad de la red, al aumentar los nodos el sistema Fuzzy es capaz de generar el mejor rendimiento de los tres sistemas.

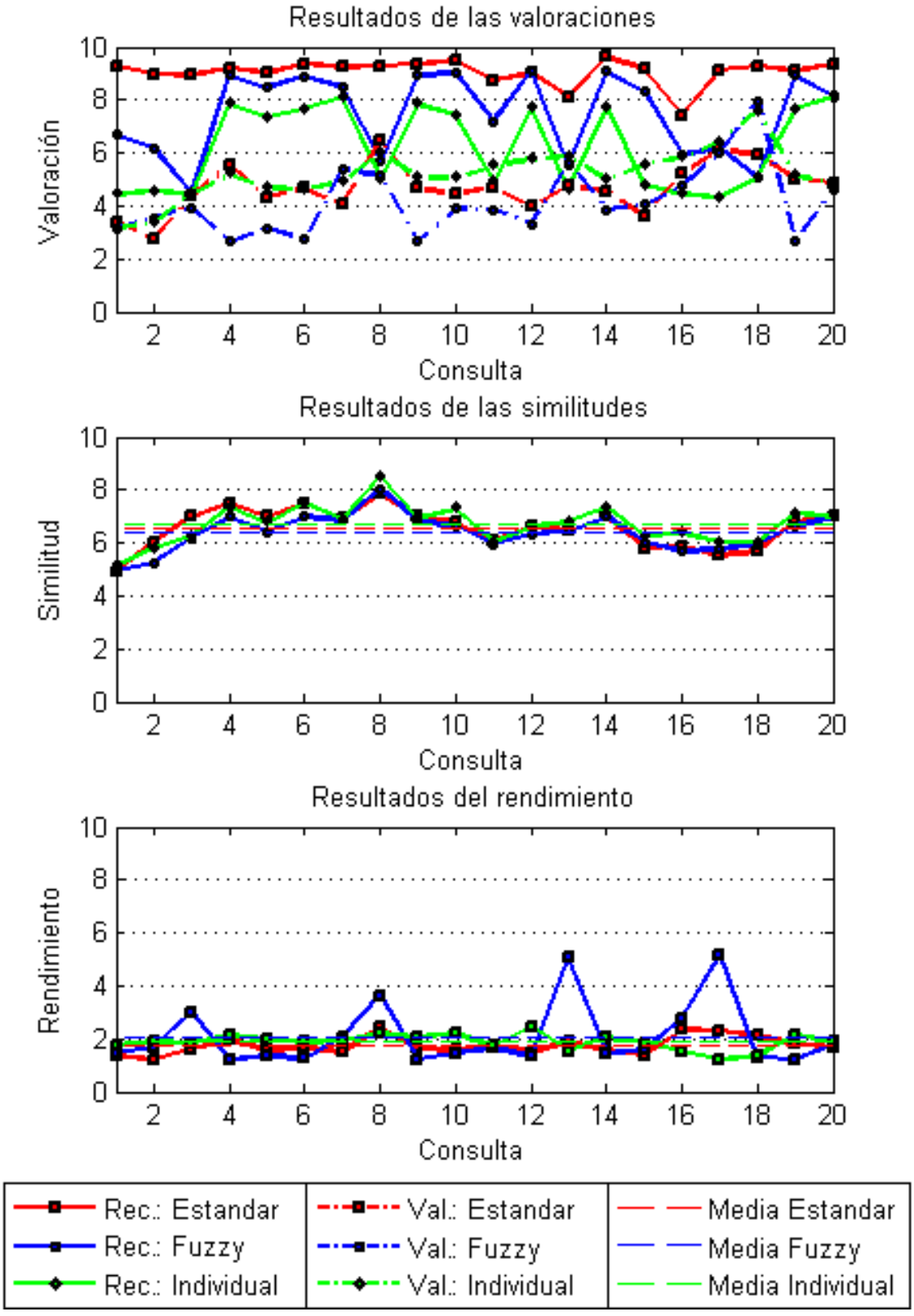

Figura 6.3: Resultados del experimento con una red aleatoria de 20 nodos

6.4.4. Red de Pearson de 20 nodos

Los resultados de la red de Pearson de 20 nodos se encuentran visualizados en la Figura 6.4.4.

Podemos observar en la primera gráfica como la diferencia entre la valoración de los casos recomendados y la valoración real del usuario de esos casos se reduce significativamente para el sistema Fuzzy. La reducción es significativa, tanto comparándolo con la red de 20 nodos aleatoria de la Sección 6.4.3 como con la red de Pearson de 10 nodos de la Sección 6.4.2. La mejora en el primer caso se debe a la utilización de una red social y a la mejora de las confianzas entre agentes derivado de esto, y en el segundo caso se debe a un aumento del número de usuarios y por lo tanto de usuarios parecidos al que realiza las consultas y al aumento del número de casos.

La gráfica de la similitud nos muestra que los tres sistemas siguen prácticamente igualados entre ellos e igualados a las similitudes del resto de redes probadas, si bien es cierto que el sistema Fuzzy es el que menor similitud tiene. Esto se debe a que sacrifica una parte de la similitud de las recomendaciones con la consulta en favor de la proximidad entre las valoraciones del caso y la realizada por el usuario.

Por último, en la gráfica del rendimiento de los sistemas podemos observar más claramente la mejora sufrida por el sistema Fuzzy. Su rendimiento se sitúa en torno a 3 puntos, frente al 2.2 aprox. del sistema Estándar. Por su parte, el sistema Individual no supera los 2 puntos de rendimiento.

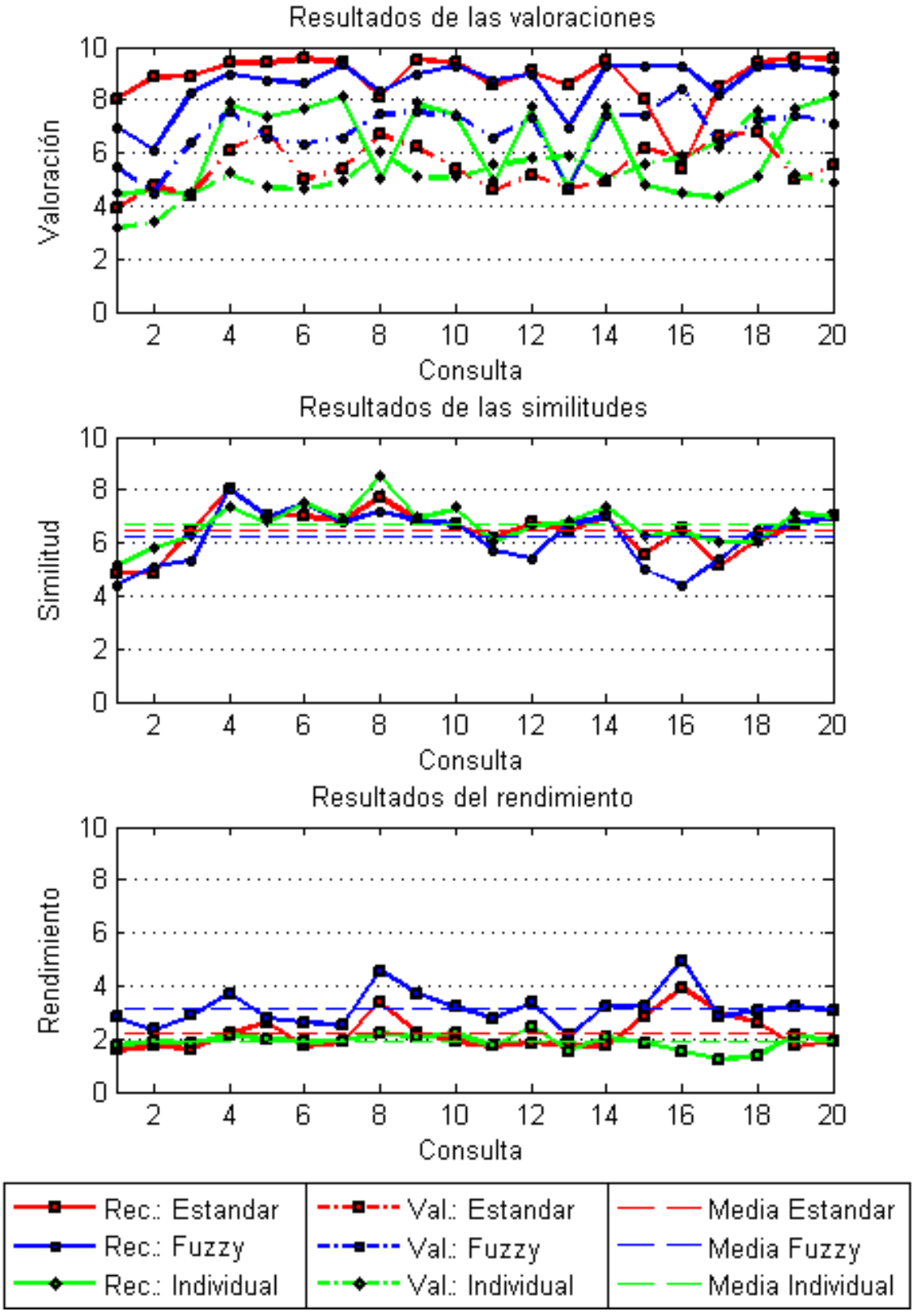

Figura 6.4: Resultados del experimento con una red de Pearson de 20 nodos

6.4.5. Red aleatoria de 50 nodos

La Figura 6.5 incluye los resultados de la red aleatoria de 50 nodos, la última de las redes aleatorias.

La primera de las gráficas de esta red nos muestra de nuevo la tendencia del sistema Fuzzy a ajustar las valoraciones de los casos recomendados a las valoraciones reales del usuario sobre esos casos. Es importante comprender que esto se consigue no mediante la modificación de las valoraciones de los casos recomendados, sino buscando casos con valoraciones más bajas pero más próximas a las valoraciones reales del usuario. Por su parte, los sistemas Estándar e Individual mantienen sus resultados de redes anteriores, haciendo recomendaciones con valoraciones muy altas muy lejos de las valoraciones reales en el caso del sistema Estándar y realizando recomendaciones muy bajas y no demasiado realistas en el caso del sistema Individual.

Se mantiene la tendencia de las redes anteriores en cuanto a los valores de similitud entre las consultas y los casos recomendados, existiendo muy poca diferencia entre los tres sistemas distintos.

Por último, en la gráfica del rendimiento podemos observar como los resultados obtenidos por el sistema Fuzzy están muy cerca de los 4 puntos, siguiendo la tendencia a crecer en los sistemas aleatorios según crece el número de nodos. Por su parte, el sistema Estándar sufre una ligera mejora con respecto a la red aleatoria de 20 nodos y se coloca por encima de los 2 puntos. El sistema Individual continúa sin variaciones.

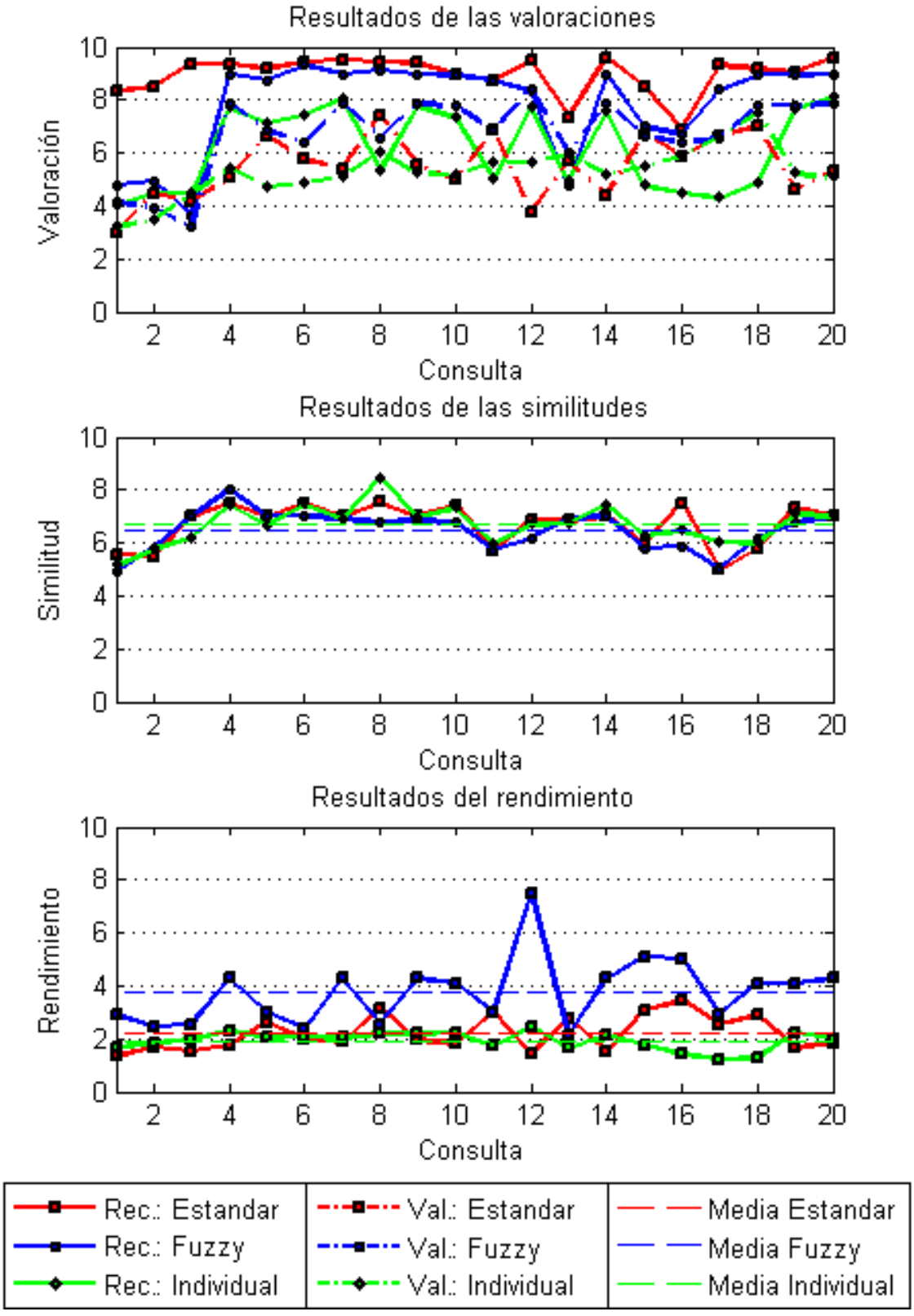

Figura 6.5: Resultados del experimento con una red aleatoria de 50 nodos

6.4.6. Red de Pearson de 50 nodos

Los resultados de la última de las redes, la red de Pearson de 50 nodos, se encuentran en la Figura 6.6.

La primera de las gráficas de esta última red nos muestra como los resultados del sistema Fuzzy han mejorado significativamente. Las valoraciones de los casos recomendados se acercan mucho a las valoraciones reales del usuario, llegando a solaparse en determinadas consultas (de la 6 a la 10). El sistema Estándar también consigue mejorar sus resultados, favorecido por una estructura de red más propicia.

La gráfica de los valores de similitud nos muestra al sistema Individual y al sistema Estándar con la misma media de similitud, por encima del sistema Fuzzy. De nuevo, como sucediera con la red 6.4.4 se debe a que sacrifica parte de la similitud de los casos recomendados con la consulta para poder realizar recomendaciones más cercanas a la realidad. A pesar de ello, se sitúa en torno a los 6 puntos.

En la última de las gráficas observamos de forma muy clara la significativa mejora experimentada por el sistema Fuzzy, situando su rendimiento por encima de los 6 puntos. Queda claro, por tanto, que el uso de una red de Pearson unido a una red con el suficiente número de nodos y de casos, supone una mejora muy importante de los resultados del sistema. También es destacable la subida más moderada experimentada por el rendimiento del sistema Estándar, situándose cerca de los 3 puntos.

En último sistema, cabe destacar el rendimiento alcanzado por el sistema Fuzzy, que dobla claramente el rendimiento alcanzado por el sistema Estándar y triplica el rendimiento del sistema Individual.

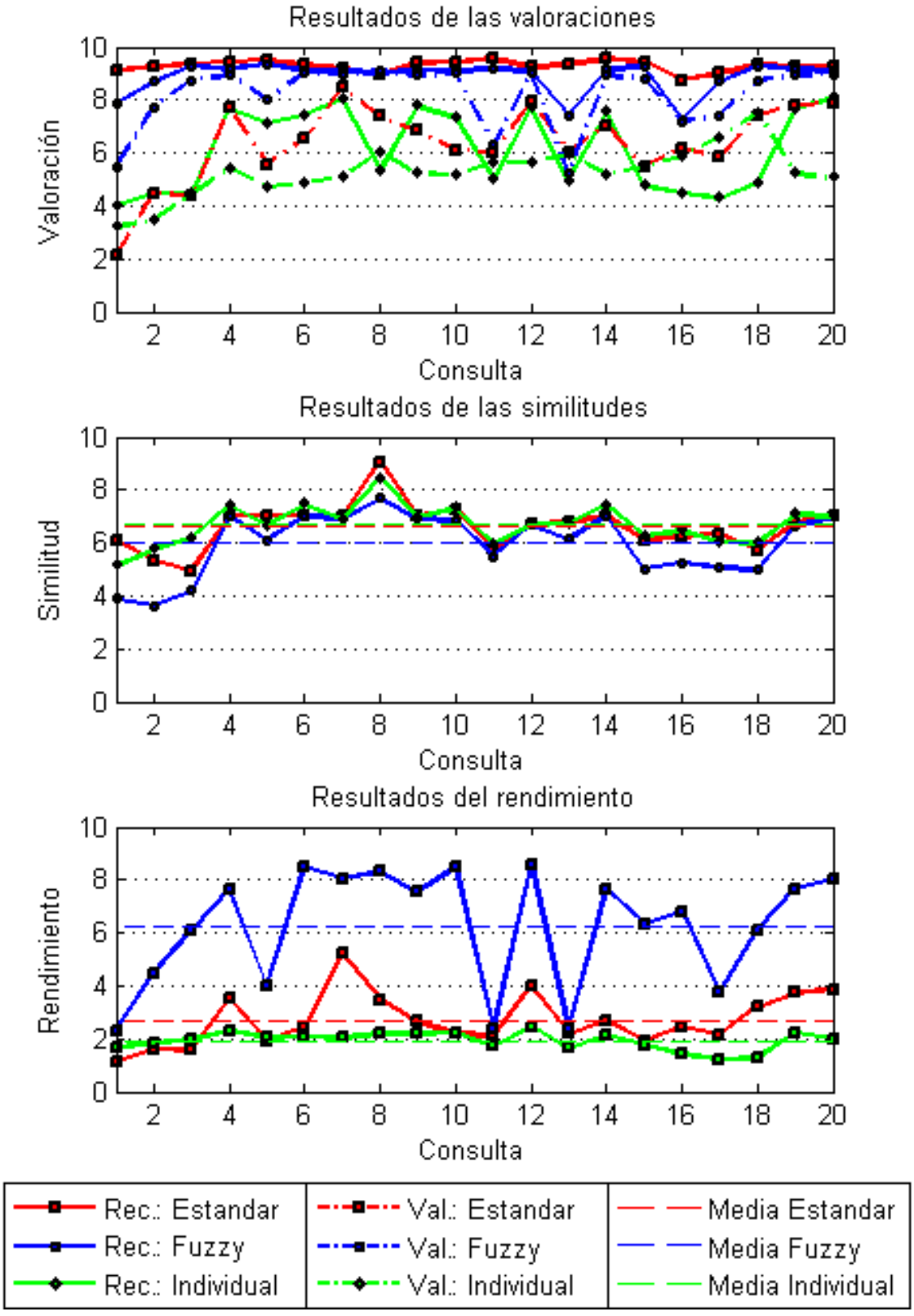

Figura 6.6: Resultados del experimento con una red de Pearson de 50 nodos

6.4.7. Resumen del experimento

A modo de resumen, aportamos la Figura 6.7 con los rendimientos de estos tres sistemas sobre las seis redes probadas.

Observamos en primer lugar el rendimiento significativo alcanzado por el sistema Fuzzy, muy superior a los otros dos sistemas. El uso de argumentaciones para descartar casos erróneos y no adecuados consigue que el rendimiento suba significativamente. Gracias a esto, demostramos el correcto funcionamiento del protocolo de argumentación AMAL modificado descrito en la Sección 5.2 y el sistema fuzzy de generación de argumentos de la Sección 5.3. Además, ratificamos nuestra primera hipótesis planteada en la Sección 6.1.3 en la que postulábamos que el sistema que utilizara el protocolo de argumentación junto con el sistema de generación de argumentaciones conseguiría un rendimiento mayor.

Observamos también como, para cualquier número de nodos, los resultados obtenidos por la red de Pearson son claramente mejores que los obtenidos por la red aleatoria, tanto para el sistema Fuzzy como para el sistema Estándar. Esto se debe a dos motivos. El primero de ellos es que la estructura de red de Pearson es capaz de representar mejor las preferencias de los usuarios, lo que ayuda a generar mejores recomendaciones. El segundo de los motivos es que los valores de confianza entre los agentes del sistema son mayores para la red de Pearson, lo que favorece la generación de argumentaciones, en el caso del sistema Fuzzy.

Otro aspecto destacable y claramente visible a simple vista en la figura 6.7 es como el rendimiento del sistema se incrementa con el número de nodos del sistema. Un mayor número de nodos supone un mayor número de usuarios que pueden ser parecidos al usuario que realiza las consultas y que por lo tanto realizarían buenas recomendaciones. Además, un mayor número de casos significa un mayor número de posibilidades a la hora de escoger una recomendación lo que facilitará la búsqueda de recomendaciones válidas. Es importante tener en cuenta que el rendimiento del sistema no crecerá de forma ilimitada con el número de nodos, sino que el crecimiento será asintótico con respecto al número de nodos del sistema. Llegará el momento en el que el número de usuarios y casos distintos sea muy grande y la inclusión de nuevos usuarios o casos no sea significativa, no variando por tanto el rendimiento del sistema.

Gracias a la inclusión del rendimiento del sistema Individual, hemos comprobado también el *"ensemble efect"* descrito en [21]. Efectivamente la cooperación y la deliberación entre nodos consiguen mejorar los resultados generados por un sistema trabajando de forma individual en cada uno de los nodos. Cada destacar que el rendimiento del sistema Individual permanece prácticamente constante en todas las redes probadas. Esto es debido a que por mucho que incluyamos nuevos perfiles y casos en el sistema, por mucho que sean similares al usuario que realiza las consultas, no sirve de nada si las soluciones no son compartidas ni evaluadas por el resto de los agentes del sistema.

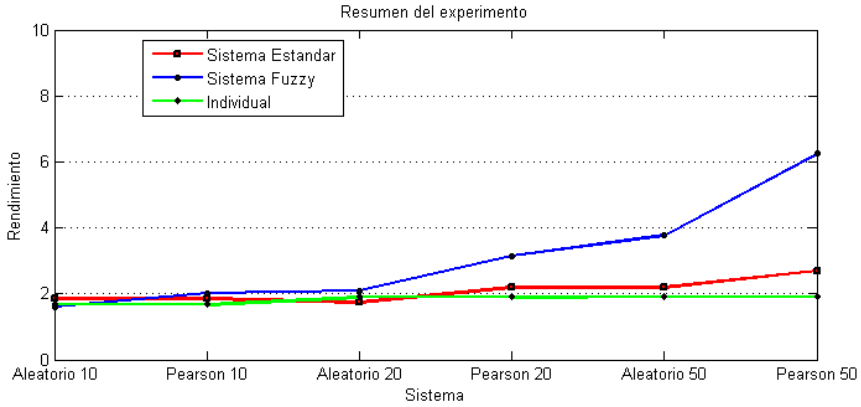

Figura 6.7: Resumen del rendimiento de los sistemas del experimento

6.5. Conclusiones

En este capítulo hemos expuesto de forma detenida el caso de uso realizado en este trabajo.

En primer lugar, hemos situado el experimento indicando porqué hemos elegido un sistema recomendador y qué nos permite probar, hemos descrito el caso de estudio estableciendo nuestras premisas y hemos expuesto la forma en la que íbamos a desarrollar nuestro experimento.

Tras ello, hemos desarrollado nuestro experimento, obteniendo los parámetros necesarios para realizar tras él un análisis de los resultados.

Una vez que hemos terminado el experimento, hemos expuesto de forma gráfica los resultados del mismo, comentando cada una de las conclusiones obtenidas.

Gracias al experimiento hemos podido probar nuestras premisas iniciales:

- El sistema fuzzy de generación de argumentaciones supondrá una mejora de los resultados del sistema frente a aquel sistema que no lo utilice.

- Un recomendador distribuido con una topología de red social obtendrá mejores resultados que un sistema con una topología aleatoria. Esto se debe a que las confianzas calculadas entre los usuarios a través del coeficiente de Pearson en una red aleatoria serán mayores que las calculadas en una red aleatoria. Por tanto, dado que las confianzas entre agentes serán mayores, las argumentaciones de cada uno de los agentes del sistema serán más fuertes y por lo tanto serán más tenidas en cuenta durante los protocolos de argumentación, mejorando los resultados del sistema.

- Los resultado de los sistemas con mayor número de nodos serán mejores que los resultados obtenidos con sistemas iguales pero con un menor número de nodos, dado que habitualmente el aumento del número de individuos en el sistema va a suponer ampliar el número de casos totales del sistema y el número de agentes parecidos al agente que realiza las consultas.

Una vez desarrollado el caso de estudio de este trabajo, y habiendo expuesto previamente todos los contenidos del mismo, finalizaremos exponiendo en el Capítulo 7 las conclusiones extraídas durante el desarrollo y el trabajo futuro tras la finalización.

Capítulo 7

Conclusiones y trabajo futuro

La última parte de este trabajo consiste de la exposición de las conclusiones alcanzadas durante el desarrollo del mismo, así como el trabajo futuro que surge tras este.

7.1. Conclusiones

En el Capítulo 3 hemos realizado un marco general para la caracterización de los sistemas CBR distribuidos y deliberativos a partir de los sistemas concretos de la literatura. Esta herramienta servirá de gran ayuda a la hora tanto de diseñar nuevos sistemas CBR distribuidos como a la hora de caracterizar los sistemas ya existentes, facilitando su comprensión y utilización.

Durante el Capítulo 4 hemos desarrollado un marco práctico o plataforma de diseño de sistemas CBR distribuidos para ampliar la funcionalidad de jCOLIBRI y permitir la construcción de cualquiera de los sistemas caracterizados por el marco teórico anterior.

Hemos instanciado el marco teórico general para diseñar un sistema CBR distribuido en el Capítulo 5. Nos hemos centrado en el desarrollo de aspectos fundamentales, táles como el protocolo de negociación, el funcionamiento distribuido, el sistema de generación fuzzy de argumentaciones y el modelo de confianza, describiéndolos en profundidad y analizando cada uno de sus aspectos. Además, hemos caracterizado el sistema instanciado mediante el marco teórico creado.

Para comprobar el funcionamiento del sistema implementado hemos realizado el caso de estudio del Capítulo 6. En él, hemos construido un recomendador musical sobre el sistema CBR distribuido instanciado anteriormente sobre el que hemos realizado una serie de experimentos que nos han permitido comprobar su correcto funcionamiento y rendimiento y nos ha permitido validar las siguientes premisas:

- El sistema fuzzy de generación de argumentaciones supone una mejora en el rendimiento de los resultados del sistema frente a aquel sistema que no lo utilice.

- Un sistema CBR distribuido con una topología de red social obtiene mejores resultados que un sistema con una topología aleatoria.
- Los resultado de los sistemas con mayor número de nodos son mejores que los resultados obtenidos con sistemas iguales pero con un menor número de nodos. Esta mejora no se mantiene de forma continua, sino que es asintótica con respecto al número de nodos.

Además hemos comprobado el *"ensemble efect"* expuesto en [21] observando como los resultados conseguidos por los sistemas CBR distribuidos superan los obtenidos por los sistemas CBR individuales.

Las principales aportaciones de este trabajo han sido:

- La creación de un marco teórico para la caracterización de sistemas CBR distribuidos. Permite conocer en detalle las características de los sistemas CBR distribuidos y la influencia de cada una de ellas en el diseño global del sistema.
- La extensión de jCOLIBRI para la creación de sistemas CBR distribuidos.
- La modificación del protocolo de argumentación AMAL para su aplicación sobre todo tipo de casos con la única restricción de que estén valorados.
- El diseño de un sistema de razonamiento fuzzy para la generación de argumentaciones dentro del protocolo AMAL.
- La creación de un sistema de aprendizaje del modelo de confianza para sistemas CBR distribuidos.
- La demostración de forma práctica los siguientes aspectos:
  - La topología de red influye de forma significativa en los resultados obtenidos por los sistemas CBR distribuidos
  - La redes sociales aportan una topología de red así como un modelo de confianza que mejoran los resultados obtenidos con otras topologías
  - La utilización de argumentaciones durante los procesos de razonamiento mejora los resultados obtenidos por los sistemas que la emplean.

7.2. Trabajo futuro

Como trabajo futuro tras la finalización de este trabajo, abrimos las siguientes vías de trabajo:

- Actualizar el marco teórico creado a medida que se creen nuevos sistemas distribuidos y con ello, el número de características que presenten.
- Extender la nueva funcionalidad de jCOLIBRI con paquetes de funcionalidades distribuidas, que permitan, por ejemplo, la comunicación de sistemas multiagente mediante varios mecanismos como *sockets*, *TCP/IP*, *JADE*, etc..
- Creación de plantillas en jCOLIBRI para la creación de sistemas CBR distribuidos a través de la definición de sus características.
- Investigar nuevos protocolos de razonamiento que no impongan restricciones sobre los casos.
- Comprobar de forma empírica hasta que número de nodos los sistemas CBR distribuidos pueden seguir incrementando su rendimiento y en qué punto se encuentra el punto crítico de crecimiento, momento a partir del cual la mejora en el rendimiento es despreciable frente al incremento en el número de nodos.
- Investigar ámbitos de aplicación de sistemas CBR distribuidos para comprobar su rendimiento en estos y su aplicabilidad.
- Comprobar el funcionamiento del sistema de aprendizaje del modelo de confianza mediante un experimento. Este puede consistir en comparar los resultados del sistema con unos valores por defecto de confianza entre los agentes frente al mismo sistema tras el aprendizaje. Para realizar el aprendizaje deberíamos iniciar en varios puntos de la red de forma aleatoria un conjunto de consultas, repitiendo el proceso varias veces, analizando cómo van variando los valores de la confianza.

Apéndice A

Conjunto de reglas del sistema de fuzzy

A continuación se muestran los conjuntos de reglas que se utilizan en cada uno de los sistemas de razonamiento utilizados en el sistema de decisión fuzzy

A.1. Sistema de reglas: Aceptación de un contraejemplo

```
if(Contraejemplo == Erroneo & Confianza ==
Nada) ->Aceptacion = Nada;
if(Contraejemplo == Erroneo & Confianza ==
Poca) ->Aceptacion = Nada;
if(Contraejemplo == Erroneo & Confianza ==
Normal) ->Aceptacion = Nada;
if(Contraejemplo == Erroneo & Confianza ==
Mucha) ->Aceptacion = Nada;
if(Contraejemplo == Erroneo & Confianza ==
Completa) ->Aceptacion = Nada;

if(Contraejemplo == Dudoso & Confianza ==
Nada) ->Aceptacion = Poco;
if(Contraejemplo == Dudoso & Confianza ==
Poca) ->Aceptacion = Nada;
if(Contraejemplo == Dudoso & Confianza ==
Normal) ->Aceptacion = Nada;
if(Contraejemplo == Dudoso & Confianza ==
Mucha) ->Aceptacion = Nada;
if(Contraejemplo == Dudoso & Confianza ==
Completa) ->Aceptacion = Nada;
```

```
if(Contraejemplo == Claro & Confianza == Nada)
->Aceptacion = Mucho;
if(Contraejemplo == Claro & Confianza == Poca)
->Aceptacion = Mucho;
if(Contraejemplo == Claro & Confianza ==
Normal) ->Aceptacion = Poco;
if(Contraejemplo == Claro & Confianza ==
Mucha) ->Aceptacion = Nada;
if(Contraejemplo == Claro & Confianza ==
Completa) ->Aceptacion = Nada;

if(Contraejemplo == Rotundo & Confianza ==
Nada) ->Aceptacion = Completa;
if(Contraejemplo == Rotundo & Confianza ==
Poca) ->Aceptacion = Completa;
if(Contraejemplo == Rotundo & Confianza ==
Normal) ->Aceptacion = Mucho;
if(Contraejemplo == Rotundo & Confianza ==
Mucha) ->Aceptacion = Poco;
if(Contraejemplo == Rotundo & Confianza ==
Completa) ->Aceptacion = Nada;
```

A.2. Sistema de reglas: Aceptación de una defensa

```
if(Confianza == Nada & Contraejemplo ==
Erroneo & Defensa == Erronea) ->Aceptacion =
Nada;
if(Confianza == Nada & Contraejemplo ==
Erroneo & Defensa == Dudosa) ->Aceptacion =
Poco;
if(Confianza == Nada & Contraejemplo ==
Erroneo & Defensa == Clara) ->Aceptacion =
Media;
if(Confianza == Nada & Contraejemplo ==
Erroneo & Defensa == Rotunda) ->Aceptacion =
Mucho;
```

```
if(Confianza == Nada & Contraejemplo == Dudoso
& Defensa == Erronea) ->Aceptacion = Nada;
if(Confianza == Nada & Contraejemplo == Dudoso
& Defensa == Dudosa) ->Aceptacion = Nada;
if(Confianza == Nada & Contraejemplo == Dudoso
& Defensa == Clara) ->Aceptacion = Poco;
if(Confianza == Nada & Contraejemplo == Dudoso
& Defensa == Rotunda) ->Aceptacion = Media;

if(Confianza == Nada & Contraejemplo == Claro
& Defensa == Erronea) ->Aceptacion = Nada;
if(Confianza == Nada & Contraejemplo == Claro
& Defensa == Dudosa) ->Aceptacion = Nada;
```

```
if(Confianza == Nada & Contraejemplo == Claro
& Defensa == Clara) ->Aceptacion = Nada;
if(Confianza == Nada & Contraejemplo == Claro
& Defensa == Rotunda) ->Aceptacion = Poco;

if(Confianza == Nada & Contraejemplo ==
Rotundo & Defensa == Erronea) ->Aceptacion =
Nada;
if(Confianza == Nada & Contraejemplo ==
Rotundo & Defensa == Dudosa) ->Aceptacion =
Nada;
if(Confianza == Nada & Contraejemplo ==
Rotundo & Defensa == Clara) ->Aceptacion =
Nada;
if(Confianza == Nada & Contraejemplo ==
Rotundo & Defensa == Rotunda) ->Aceptacion =
Nada;

if(Confianza == Poca & Contraejemplo ==
Erroneo & Defensa == Erronea) ->Aceptacion =
Poco;
if(Confianza == Poca & Contraejemplo ==
Erroneo & Defensa == Dudosa) ->Aceptacion =
Poco;
if(Confianza == Poca & Contraejemplo ==
Erroneo & Defensa == Clara) ->Aceptacion =
Media;
if(Confianza == Poca & Contraejemplo ==
Erroneo & Defensa == Rotunda) ->Aceptacion =
Mucho;

if(Confianza == Poca & Contraejemplo == Dudoso
& Defensa == Erronea) ->Aceptacion = Poco;
if(Confianza == Poca & Contraejemplo == Dudoso
& Defensa == Dudosa) ->Aceptacion = Poco;
if(Confianza == Poca & Contraejemplo == Dudoso
& Defensa == Clara) ->Aceptacion = Media;
if(Confianza == Poca & Contraejemplo == Dudoso
& Defensa == Rotunda) ->Aceptacion = Media;

if(Confianza == Poca & Contraejemplo == Claro
& Defensa == Erronea) ->Aceptacion = Nada;
if(Confianza == Poca & Contraejemplo == Claro
& Defensa == Dudosa) ->Aceptacion = Nada;
if(Confianza == Poca & Contraejemplo == Claro
& Defensa == Clara) ->Aceptacion = Poco;
if(Confianza == Poca & Contraejemplo == Claro
& Defensa == Rotunda) ->Aceptacion = Poco;

if(Confianza == Poca & Contraejemplo ==
Rotundo & Defensa == Erronea) ->Aceptacion =
Nada;
if(Confianza == Poca & Contraejemplo ==
Rotundo & Defensa == Dudosa) ->Aceptacion =
Nada;
if(Confianza == Poca & Contraejemplo ==
Rotundo & Defensa == Clara) ->Aceptacion =
Nada;
if(Confianza == Poca & Contraejemplo ==
Rotundo & Defensa == Rotunda) ->Aceptacion =
Poco;

if(Confianza == Normal & Contraejemplo ==
Erroneo & Defensa == Erronea) ->Aceptacion =
Media;
if(Confianza == Normal & Contraejemplo ==
Erroneo & Defensa == Dudosa) ->Aceptacion =
Media;
if(Confianza == Normal & Contraejemplo ==
Erroneo & Defensa == Clara) ->Aceptacion =
Mucho;
if(Confianza == Normal & Contraejemplo ==
Erroneo & Defensa == Rotunda) ->Aceptacion =
Completa;

if(Confianza == Normal & Contraejemplo ==
Dudoso & Defensa == Erronea) ->Aceptacion =
Media;
if(Confianza == Normal & Contraejemplo ==
Dudoso & Defensa == Dudosa) ->Aceptacion =
Media;
if(Confianza == Normal & Contraejemplo ==
Dudoso & Defensa == Clara) ->Aceptacion =
Mucho;
if(Confianza == Normal & Contraejemplo ==
Dudoso & Defensa == Rotunda) ->Aceptacion =
Mucho;

if(Confianza == Normal & Contraejemplo ==
Claro & Defensa == Erronea) ->Aceptacion =
Poco;
if(Confianza == Normal & Contraejemplo ==
Claro & Defensa == Dudosa) ->Aceptacion =
Poco;
if(Confianza == Normal & Contraejemplo ==
Claro & Defensa == Clara) ->Aceptacion =
Media;
if(Confianza == Normal & Contraejemplo ==
Claro & Defensa == Rotunda) ->Aceptacion =
Mucho;

if(Confianza == Normal & Contraejemplo ==
Rotundo & Defensa == Erronea) ->Aceptacion =
Nada;
if(Confianza == Normal & Contraejemplo ==
Rotundo & Defensa == Dudosa) ->Aceptacion =
Poco;
if(Confianza == Normal & Contraejemplo ==
Rotundo & Defensa == Clara) ->Aceptacion =
Poco;
if(Confianza == Normal & Contraejemplo ==
Rotundo & Defensa == Rotunda) ->Aceptacion =
Media;

if(Confianza == Mucha & Contraejemplo ==
Erroneo & Defensa == Erronea) ->Aceptacion =
Mucho;
if(Confianza == Mucha & Contraejemplo ==
Erroneo & Defensa == Dudosa) ->Aceptacion =
Mucho;
if(Confianza == Mucha & Contraejemplo ==
Erroneo & Defensa == Clara) ->Aceptacion =
Completa;
if(Confianza == Mucha & Contraejemplo ==
Erroneo & Defensa == Rotunda) ->Aceptacion =
Completa;

if(Confianza == Mucha & Contraejemplo ==
Dudoso & Defensa == Erronea) ->Aceptacion =
Media;
if(Confianza == Mucha & Contraejemplo ==
Dudoso & Defensa == Dudosa) ->Aceptacion =
Mucho;
if(Confianza == Mucha & Contraejemplo ==
Dudoso & Defensa == Clara) ->Aceptacion =
Mucho;
if(Confianza == Mucha & Contraejemplo ==
Dudoso & Defensa == Rotunda) ->Aceptacion =
Completa;

if(Confianza == Mucha & Contraejemplo == Claro
& Defensa == Erronea) ->Aceptacion = Poco;
if(Confianza == Mucha & Contraejemplo == Claro
& Defensa == Dudosa) ->Aceptacion = Media;
if(Confianza == Mucha & Contraejemplo == Claro
& Defensa == Clara) ->Aceptacion = Mucho;
if(Confianza == Mucha & Contraejemplo == Claro
& Defensa == Rotunda) ->Aceptacion = Completa;

if(Confianza == Mucha & Contraejemplo ==
Rotundo & Defensa == Erronea) ->Aceptacion =
Nada;
if(Confianza == Mucha & Contraejemplo ==
Rotundo & Defensa == Dudosa) ->Aceptacion =
Poco;
if(Confianza == Mucha & Contraejemplo ==
Rotundo & Defensa == Clara) ->Aceptacion =
Media;
if(Confianza == Mucha & Contraejemplo ==
Rotundo & Defensa == Rotunda) ->Aceptacion =
Mucho;

if(Confianza == Completa & Contraejemplo ==
Erroneo & Defensa == Erronea) ->Aceptacion =
Mucho;
if(Confianza == Completa & Contraejemplo ==
Erroneo & Defensa == Dudosa) ->Aceptacion =
Mucho;
```

```
if(Confianza == Completa & Contraejemplo ==
Erroneo & Defensa == Clara) ->Aceptacion =
Completa;
if(Confianza == Completa & Contraejemplo ==
Erroneo & Defensa == Rotunda) ->Aceptacion =
Completa;

if(Confianza == Completa & Contraejemplo ==
Dudoso & Defensa == Erronea) ->Aceptacion =
Mucho;
if(Confianza == Completa & Contraejemplo ==
Dudoso & Defensa == Dudosa) ->Aceptacion =
Mucho;
if(Confianza == Completa & Contraejemplo ==
Dudoso & Defensa == Clara) ->Aceptacion =
Mucho;
if(Confianza == Completa & Contraejemplo ==
Dudoso & Defensa == Rotunda) ->Aceptacion =
Completa;

if(Confianza == Completa & Contraejemplo ==
Claro & Defensa == Erronea) ->Aceptacion =
Media;
if(Confianza == Completa & Contraejemplo ==
Claro & Defensa == Dudosa) ->Aceptacion =
Mucho;
if(Confianza == Completa & Contraejemplo ==
Claro & Defensa == Clara) ->Aceptacion =
Mucho;
if(Confianza == Completa & Contraejemplo ==
Claro & Defensa == Rotunda) ->Aceptacion =
Mucho;

if(Confianza == Completa & Contraejemplo ==
Rotundo & Defensa == Erronea) ->Aceptacion =
Poco;
if(Confianza == Completa & Contraejemplo ==
Rotundo & Defensa == Dudosa) ->Aceptacion =
Media;
if(Confianza == Completa & Contraejemplo ==
Rotundo & Defensa == Clara) ->Aceptacion =
Mucho;
if(Confianza == Completa & Contraejemplo ==
Rotundo & Defensa == Rotunda) ->Aceptacion =
Completa;
```

A.3. Sistema de reglas: Compatibilizar

```
if(SimilitudCasos == Distintos & SimilitudPerfiles
== Distintos) ->Compatibilidad = Nada;
if(SimilitudCasos == Distintos & SimilitudPerfiles
== Poco) ->Compatibilidad = Nada;
if(SimilitudCasos == Distintos & SimilitudPerfiles
== Medio) ->Compatibilidad = Nada;
if(SimilitudCasos == Distintos & SimilitudPerfiles
== Mucho) ->Compatibilidad = Poco;
if(SimilitudCasos == Distintos & SimilitudPerfiles
== Iguales) ->Compatibilidad = Poco;

if(SimilitudCasos == Poco & SimilitudPerfiles
== Distintos) ->Compatibilidad = Nada;
if(SimilitudCasos == Poco & SimilitudPerfiles
== Poco) ->Compatibilidad = Poco;
if(SimilitudCasos == Poco & SimilitudPerfiles
== Medio) ->Compatibilidad = Poco;
if(SimilitudCasos == Poco & SimilitudPerfiles
== Mucho) ->Compatibilidad = Poco;
if(SimilitudCasos == Poco & SimilitudPerfiles
== Iguales) ->Compatibilidad = Poco;

if(SimilitudCasos == Medio & SimilitudPerfiles
== Distintos) ->Compatibilidad = Nada;
if(SimilitudCasos == Medio & SimilitudPerfiles
== Poco) ->Compatibilidad = Poco;
if(SimilitudCasos == Medio & SimilitudPerfiles
== Medio) ->Compatibilidad = Poco;
if(SimilitudCasos == Medio & SimilitudPerfiles
== Mucho) ->Compatibilidad = Poco;
if(SimilitudCasos == Medio & SimilitudPerfiles
== Iguales) ->Compatibilidad = Normal;

if(SimilitudCasos == Mucho & SimilitudPerfiles
== Distintos) ->Compatibilidad = Poco;
if(SimilitudCasos == Mucho & SimilitudPerfiles
== Poco) ->Compatibilidad = Poco;
if(SimilitudCasos == Mucho & SimilitudPerfiles
== Medio) ->Compatibilidad = Normal;
if(SimilitudCasos == Mucho & SimilitudPerfiles
== Mucho) ->Compatibilidad = Mucho;
if(SimilitudCasos == Mucho & SimilitudPerfiles
== Iguales) ->Compatibilidad = Mucho;

if(SimilitudCasos == Iguales & SimilitudPerfiles
== Distintos) ->Compatibilidad = Poco;
if(SimilitudCasos == Iguales & SimilitudPerfiles
== Poco) ->Compatibilidad = Normal;
if(SimilitudCasos == Iguales & SimilitudPerfiles
== Medio) ->Compatibilidad = Mucho;
if(SimilitudCasos == Iguales & SimilitudPerfiles
== Mucho) ->Compatibilidad = Mucho;
if(SimilitudCasos == Iguales & SimilitudPerfiles
== Iguales) ->Compatibilidad = Completa;
```

A.4. Sistema de reglas: Confiar

```
if(CredibilidadA == Nada & CredibilidadB ==
Nada) ->Confianza = Normal;
if(CredibilidadA == Nada & CredibilidadB ==
Levemente) ->Confianza = Normal;
if(CredibilidadA == Nada & CredibilidadB ==
Poco) ->Confianza = Poca;
if(CredibilidadA == Nada & CredibilidadB ==
Algo) ->Confianza = Poca;
if(CredibilidadA == Nada & CredibilidadB ==
Bastante) ->Confianza = Nada;
if(CredibilidadA == Nada & CredibilidadB ==
Mucho) ->Confianza = Nada;
if(CredibilidadA == Nada & CredibilidadB ==
Creible) ->Confianza = Nada;
if(CredibilidadA == Levemente & CredibilidadB
== Nada) ->Confianza = Normal;
if(CredibilidadA == Levemente & CredibilidadB
== Levemente) ->Confianza = Normal;
if(CredibilidadA == Levemente & CredibilidadB
== Poco) ->Confianza = Normal;
if(CredibilidadA == Levemente & CredibilidadB
== Algo) ->Confianza = Poca;
if(CredibilidadA == Levemente & CredibilidadB
== Bastante) ->Confianza = Poca;
if(CredibilidadA == Levemente & CredibilidadB
== Mucho) ->Confianza = Nada;
if(CredibilidadA == Levemente & CredibilidadB
== Creible) ->Confianza = Nada;
```

```
if(CredibilidadA == Poco & CredibilidadB ==
Nada) ->Confianza = Mucha;
if(CredibilidadA == Poco & CredibilidadB ==
Levemente) ->Confianza = Normal;
if(CredibilidadA == Poco & CredibilidadB ==
Poco) ->Confianza = Normal;
if(CredibilidadA == Poco & CredibilidadB ==
Algo) ->Confianza = Normal;
if(CredibilidadA == Poco & CredibilidadB ==
Bastante) ->Confianza = Poca;
if(CredibilidadA == Poco & CredibilidadB ==
Mucho) ->Confianza = Poca;
if(CredibilidadA == Poco & CredibilidadB ==
Creible) ->Confianza = Nada;

if(CredibilidadA == Algo & CredibilidadB ==
Nada) ->Confianza = Mucha;
if(CredibilidadA == Algo & CredibilidadB ==
Levemente) ->Confianza = Mucha;
if(CredibilidadA == Algo & CredibilidadB ==
Poco) ->Confianza = Normal;
if(CredibilidadA == Algo & CredibilidadB ==
Algo) ->Confianza = Normal;
if(CredibilidadA == Algo & CredibilidadB ==
Bastante) ->Confianza = Normal;
if(CredibilidadA == Algo & CredibilidadB ==
Mucho) ->Confianza = Poca;
if(CredibilidadA == Algo & CredibilidadB ==
Creible) ->Confianza = Poca;

if(CredibilidadA == Bastante & CredibilidadB
== Nada) ->Confianza = Completa;
if(CredibilidadA == Bastante & CredibilidadB
== Levemente) ->Confianza = Mucha;
if(CredibilidadA == Bastante & CredibilidadB
== Poco) ->Confianza = Mucha;
if(CredibilidadA == Bastante & CredibilidadB
== Algo) ->Confianza = Normal;
if(CredibilidadA == Bastante & CredibilidadB
== Bastante) ->Confianza = Normal;
if(CredibilidadA == Bastante & CredibilidadB
== Mucho) ->Confianza = Normal;
if(CredibilidadA == Bastante & CredibilidadB
== Creible) ->Confianza = Poca;

if(CredibilidadA == Mucho & CredibilidadB ==
Nada) ->Confianza = Completa;
if(CredibilidadA == Mucho & CredibilidadB ==
Levemente) ->Confianza = Completa;
if(CredibilidadA == Mucho & CredibilidadB ==
Poco) ->Confianza = Mucha;
if(CredibilidadA == Mucho & CredibilidadB ==
Algo) ->Confianza = Mucha;
if(CredibilidadA == Mucho & CredibilidadB ==
Bastante) ->Confianza = Normal;
if(CredibilidadA == Mucho & CredibilidadB ==
Mucho) ->Confianza = Normal;
if(CredibilidadA == Mucho & CredibilidadB ==
Creible) ->Confianza = Normal;

if(CredibilidadA == Creible & CredibilidadB ==
Nada) ->Confianza = Completa;
if(CredibilidadA == Creible & CredibilidadB ==
Levemente) ->Confianza = Completa;
if(CredibilidadA == Creible & CredibilidadB ==
Poco) ->Confianza = Completa;
if(CredibilidadA == Creible & CredibilidadB ==
Algo) ->Confianza = Mucha;
if(CredibilidadA == Creible & CredibilidadB ==
Bastante) ->Confianza = Mucha;
if(CredibilidadA == Creible & CredibilidadB ==
Mucho) ->Confianza = Normal;
if(CredibilidadA == Creible & CredibilidadB ==
Creible) ->Confianza = Normal;
```

A.5. Sistema de reglas: Contraejemplificar

```
if(compatibilidad == Nada & Valoracion ==
Nada) ->Contraejemplo = Erroneo;
if(compatibilidad == Nada & Valoracion ==
Poco) ->Contraejemplo = Erroneo;
if(compatibilidad == Nada & Valoracion ==
Normal) ->Contraejemplo = Erroneo;
if(compatibilidad == Nada & Valoracion ==
Mucho) ->Contraejemplo = Erroneo;
if(compatibilidad == Nada & Valoracion ==
Totalmente) ->Contraejemplo = Erroneo;

if(compatibilidad == Poco & Valoracion ==
Nada) ->Contraejemplo = Dudoso;
if(compatibilidad == Poco & Valoracion ==
Poco) ->Contraejemplo = Erroneo;
if(compatibilidad == Poco & Valoracion ==
Normal) ->Contraejemplo = Erroneo;
if(compatibilidad == Poco & Valoracion ==
Mucho) ->Contraejemplo = Erroneo;
if(compatibilidad == Poco & Valoracion ==
Totalmente) ->Contraejemplo = Erroneo;

if(compatibilidad == Normal & Valoracion ==
Nada) ->Contraejemplo = Dudoso;
if(compatibilidad == Normal & Valoracion ==
Poco) ->Contraejemplo = Dudoso;
if(compatibilidad == Normal & Valoracion ==
Normal) ->Contraejemplo = Erroneo;
if(compatibilidad == Normal & Valoracion ==
Mucho) ->Contraejemplo = Erroneo;
if(compatibilidad == Normal & Valoracion ==
Totalmente) ->Contraejemplo = Erroneo;

if(compatibilidad == Mucho & Valoracion ==
Nada) ->Contraejemplo = Claro;
if(compatibilidad == Mucho & Valoracion ==
Poco) ->Contraejemplo = Dudoso;
if(compatibilidad == Mucho & Valoracion ==
Normal) ->Contraejemplo = Dudoso;
if(compatibilidad == Mucho & Valoracion ==
Mucho) ->Contraejemplo = Erroneo;
if(compatibilidad == Mucho & Valoracion ==
Totalmente) ->Contraejemplo = Erroneo;

if(compatibilidad == Completa & Valoracion ==
Nada) ->Contraejemplo = Rotundo;
if(compatibilidad == Completa & Valoracion ==
Poco) ->Contraejemplo = Claro;
if(compatibilidad == Completa & Valoracion ==
Normal) ->Contraejemplo = Dudoso;
if(compatibilidad == Completa & Valoracion ==
Mucho) ->Contraejemplo = Erroneo;
if(compatibilidad == Completa & Valoracion ==
Totalmente) ->Contraejemplo = Erroneo;
```

A.6. Sistema de reglas: Defender

```
if(Similitud == Distintos & Valoracion ==
Nada) ->Defensa = Erronea;
if(Similitud == Distintos & Valoracion ==
Poco) ->Defensa = Erronea;
if(Similitud == Distintos & Valoracion ==
Normal) ->Defensa = Erronea;
if(Similitud == Distintos & Valoracion ==
Mucho) ->Defensa = Erronea;
if(Similitud == Distintos & Valoracion ==
Totalmente) ->Defensa = Erronea;

if(Similitud == Poco & Valoracion == Nada)
->Defensa = Erronea;
if(Similitud == Poco & Valoracion == Poco)
->Defensa = Erronea;
if(Similitud == Poco & Valoracion == Normal)
->Defensa = Erronea;
if(Similitud == Poco & Valoracion == Mucho)
->Defensa = Erronea;
if(Similitud == Poco & Valoracion == Totalmente)
->Defensa = Dudosa;

if(Similitud == Medio & Valoracion == Nada)
->Defensa = Erronea;
if(Similitud == Medio & Valoracion == Poco)
->Defensa = Erronea;
if(Similitud == Medio & Valoracion == Normal)
->Defensa = Erronea;
if(Similitud == Medio & Valoracion == Mucho)
->Defensa = Dudosa;
if(Similitud == Medio & Valoracion ==
Totalmente) ->Defensa = Dudosa;

if(Similitud == Mucho & Valoracion == Nada)
->Defensa = Erronea;
if(Similitud == Mucho & Valoracion == Poco)
->Defensa = Erronea;
if(Similitud == Mucho & Valoracion == Normal)
->Defensa = Dudosa;
if(Similitud == Mucho & Valoracion == Mucho)
->Defensa = Dudosa;
if(Similitud == Mucho & Valoracion ==
Totalmente) ->Defensa = Clara;

if(Similitud == Iguales & Valoracion == Nada)
->Defensa = Erronea;
if(Similitud == Iguales & Valoracion == Poco)
->Defensa = Dudosa;
if(Similitud == Iguales & Valoracion ==
Normal) ->Defensa = Clara;
if(Similitud == Iguales & Valoracion == Mucho)
->Defensa = Rotunda;
if(Similitud == Iguales & Valoracion ==
Totalmente) ->Defensa = Rotunda;
```

A.7. Sistema de reglas: Valorar

```
if(perfiles == Distintos & casos == Distintos
& Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Distintos
& Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Distintos
& Valoracion == Normal) ->Recomendacion =
Nada;
if(perfiles == Distintos & casos == Distintos
& Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Distintos
& Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Distintos & casos == Poco &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Poco &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Poco &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Poco &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Poco &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Distintos & casos == Medio &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Medio &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Medio &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Medio &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Medio &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Distintos & casos == Mucho &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Mucho &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Mucho &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Mucho &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Mucho &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Distintos & casos == Iguales &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Iguales &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Iguales &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Iguales &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Distintos & casos == Iguales
& Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Poco & casos == Distintos &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Distintos &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Distintos &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Distintos &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Distintos &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Poco & casos == Poco &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Poco &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Poco &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Poco &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Poco &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Poco & casos == Medio &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Medio &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Medio &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Medio &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Medio &
Valoracion == Totalmente) ->Recomendacion =
Nada;
```

```
if(perfiles == Poco & casos == Mucho &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Mucho &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Mucho &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Mucho &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Mucho &
Valoracion == Totalmente) ->Recomendacion =
Poco;

if(perfiles == Poco & casos == Iguales &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Iguales &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Iguales &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Poco & casos == Iguales &
Valoracion == Mucho) ->Recomendacion = Poco;
if(perfiles == Poco & casos == Iguales &
Valoracion == Totalmente) ->Recomendacion =
Poco;

if(perfiles == Medio & casos == Distintos &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Distintos &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Distintos &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Distintos &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Distintos &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Medio & casos == Poco &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Poco &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Poco &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Poco &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Poco &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Medio & casos == Medio &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Medio &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Medio &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Medio &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Medio &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Medio & casos == Mucho &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Mucho &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Mucho &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Mucho &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Mucho &
Valoracion == Totalmente) ->Recomendacion =
Poco;

if(perfiles == Medio & casos == Iguales &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Iguales &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Iguales &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Medio & casos == Iguales &
Valoracion == Mucho) ->Recomendacion = Poco;
if(perfiles == Medio & casos == Iguales &
Valoracion == Totalmente) ->Recomendacion =
Normal;

if(perfiles == Mucho & casos == Distintos &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Mucho & casos == Distintos &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Mucho & casos == Distintos &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Mucho & casos == Distintos &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Mucho & casos == Distintos &
Valoracion == Totalmente) ->Recomendacion =
Nada;

if(perfiles == Mucho & casos == Poco &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Mucho & casos == Poco &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Mucho & casos == Poco &
Valoracion == Normal) ->Recomendacion = Poco;
if(perfiles == Mucho & casos == Poco &
Valoracion == Mucho) ->Recomendacion = Poco;
if(perfiles == Mucho & casos == Poco &
Valoracion == Totalmente) ->Recomendacion =
Poco;

if(perfiles == Mucho & casos == Medio &
Valoracion == Nada) ->Recomendacion = Poco;
if(perfiles == Mucho & casos == Medio &
Valoracion == Poco) ->Recomendacion = Poco;
if(perfiles == Mucho & casos == Medio &
Valoracion == Normal) ->Recomendacion =
Normal;
if(perfiles == Mucho & casos == Medio &
Valoracion == Mucho) ->Recomendacion = Normal;
if(perfiles == Mucho & casos == Medio &
Valoracion == Totalmente) ->Recomendacion =
Mucho;

if(perfiles == Mucho & casos == Mucho &
Valoracion == Nada) ->Recomendacion = Poco;
if(perfiles == Mucho & casos == Mucho &
Valoracion == Poco) ->Recomendacion = Poco;
if(perfiles == Mucho & casos == Mucho &
Valoracion == Normal) ->Recomendacion =
Normal;
if(perfiles == Mucho & casos == Mucho &
Valoracion == Mucho) ->Recomendacion = Normal;
if(perfiles == Mucho & casos == Mucho &
Valoracion == Totalmente) ->Recomendacion =
Mucho;

if(perfiles == Mucho & casos == Iguales &
Valoracion == Nada) ->Recomendacion = Poco;
if(perfiles == Mucho & casos == Iguales &
Valoracion == Poco) ->Recomendacion = Normal;
if(perfiles == Mucho & casos == Iguales
& Valoracion == Normal) ->Recomendacion =
Normal;
if(perfiles == Mucho & casos == Iguales &
Valoracion == Mucho) ->Recomendacion = Mucho;
if(perfiles == Mucho & casos == Iguales &
Valoracion == Totalmente) ->Recomendacion =
Mucho;

if(perfiles == Iguales & casos == Distintos &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Iguales & casos == Distintos &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Iguales & casos == Distintos &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Iguales & casos == Distintos &
Valoracion == Mucho) ->Recomendacion = Nada;
if(perfiles == Iguales & casos == Distintos
& Valoracion == Totalmente) ->Recomendacion =
Poco;

if(perfiles == Iguales & casos == Poco &
Valoracion == Nada) ->Recomendacion = Nada;
if(perfiles == Iguales & casos == Poco &
Valoracion == Poco) ->Recomendacion = Nada;
if(perfiles == Iguales & casos == Poco &
Valoracion == Normal) ->Recomendacion = Nada;
if(perfiles == Iguales & casos == Poco &
Valoracion == Mucho) ->Recomendacion = Poco;
```

```
if(perfiles == Iguales & casos == Poco &
Valoracion == Totalmente) ->Recomendacion =
Poco;

if(perfiles == Iguales & casos == Medio &
Valoracion == Nada) ->Recomendacion = Poco;
if(perfiles == Iguales & casos == Medio &
Valoracion == Poco) ->Recomendacion = Normal;
if(perfiles == Iguales & casos == Medio
& Valoracion == Normal) ->Recomendacion =
Normal;
if(perfiles == Iguales & casos == Medio &
Valoracion == Mucho) ->Recomendacion = Mucho;
if(perfiles == Iguales & casos == Medio &
Valoracion == Totalmente) ->Recomendacion =
Mucho;

if(perfiles == Iguales & casos == Mucho &
Valoracion == Nada) ->Recomendacion = Poco;
if(perfiles == Iguales & casos == Mucho &
Valoracion == Poco) ->Recomendacion = Normal;
if(perfiles == Iguales & casos == Mucho &
Valoracion == Normal) ->Recomendacion = Mucho;
if(perfiles == Iguales & casos == Mucho &
Valoracion == Mucho) ->Recomendacion = Mucho;
if(perfiles == Iguales & casos == Mucho &
Valoracion == Totalmente) ->Recomendacion =
Totalmente;


if(perfiles == Iguales & casos == Iguales &
Valoracion == Nada) ->Recomendacion = Normal;
if(perfiles == Iguales & casos == Iguales &
Valoracion == Poco) ->Recomendacion = Normal;
if(perfiles == Iguales & casos == Iguales &
Valoracion == Normal) ->Recomendacion = Mucho;
if(perfiles == Iguales & casos == Iguales &
Valoracion == Mucho) ->Recomendacion = Mucho;
if(perfiles == Iguales & casos == Iguales &
Valoracion == Totalmente) ->Recomendacion =
Totalmente;
```

Apéndice B

Tipos de información del sistema difuso

A continuación se detallan los tipos de información utilizados por el sistema de decisión Fuzzy. Todos los tipos de datos utilizan un rango entre 0 y 10 con una cardinalidad de 100.

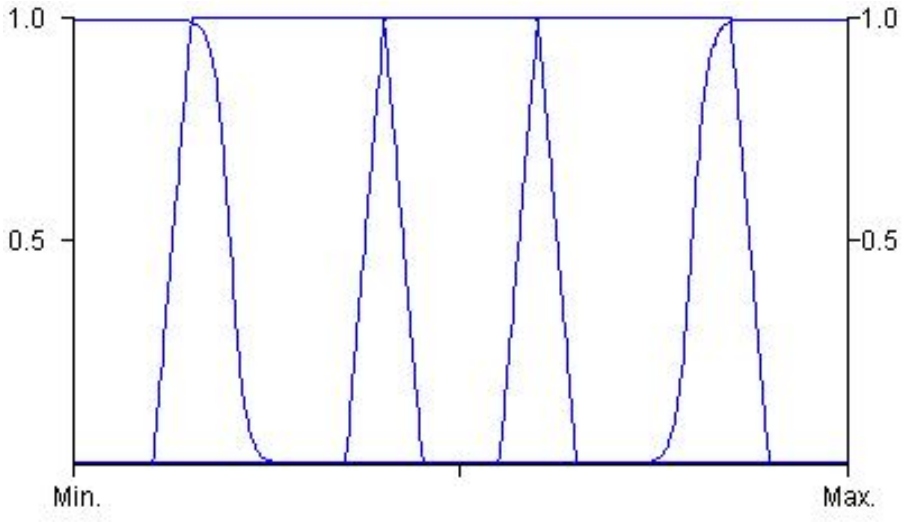

Figura B.1: Tipo de datos Compatibilidad

B.1. Tipo de datos: Compatibilidad

El modelado de los datos correspondientes al tipo Compatibilidad puede verse en la figura B.1. Utiliza como funciones de pertenencia:

- **Nada:** Modelado por una función sigmoidal de parámetros a = 2.0 y b = -0.1
- **Poco:** Modelado por una función trapezoidal de parámetros a = 1.0, b = 1.5, c = 4.0 y d = 4.5
- **Normal:** Modelado por una función trapezoidal de parámetros a = 3.5, b = 4.0, c = 6.0 y d = 6.5
- **Mucho:** Modelado por una función trapezoidal de parámetros a = 5.5, b = 6.0, c = 8.5 y d = 9.0
- **Completa:** Modelado por una función sigmoidal de parámetros a = 8.0 y b = 0.1

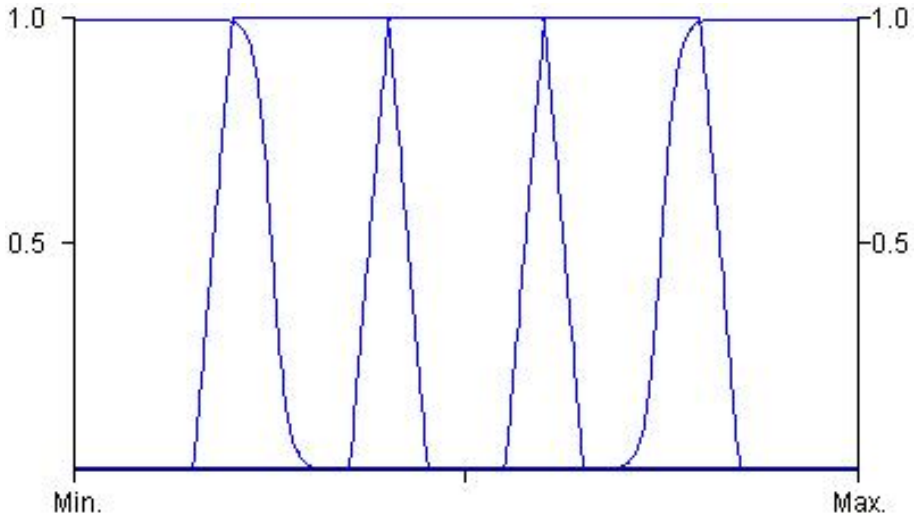

Figura B.2: Tipo de datos ConfianzaAgente

B.2. Tipo de datos: ConfianzaAgente

El tipo de datos ConfianzaAgente se encuentra representado en la figura B.2. Tiene las siguientes funciones de pertenencia:

- **Nada:** Modelado por una función sigmoidal de parámetros a = 2.5 y b = -0.1
- **Poca:** Modelado por una función trapezoidal de parámetros a = 1.5, b = 2.0, c = 4.0 y d = 4.5
- **Normal:** Modelado por una función trapezoidal de parámetros a = 3.5, b = 4.0, c = 6.0 y d = 6.5
- **Mucha:** Modelado por una función trapezoidal de parámetros a = 5.5, b = 6.0, c = 8.0 y d = 8.5
- **Completa:** Modelado por una función sigmoidal de parámetros a = 7.5 y b = 0.1

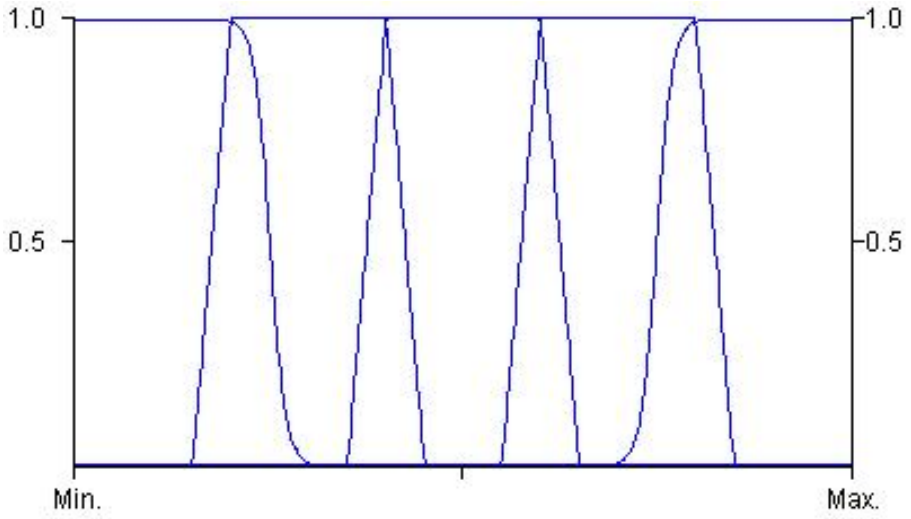

Figura B.3: Tipo de datos ConfianzaContraejemplo

B.3. Tipo de datos: ConfianzaContraejemplo

La figura B.3 representa el tipo de datos modelado por ConfianzaContraejemplo. Sus funciones de pertenencia son:

- **Nada:** Modelado por una función sigmoidal de parámetros a = 2.5 y b = -0.1
- **Poco:** Modelado por una función trapezoidal de parámetros a = 1.5, b = 2.0, c = 4.0 y d = 4.5
- **Media:** Modelado por una función trapezoidal de parámetros a = 3.5, b = 4.0, c = 6.0 y d = 6.5
- **Mucho:** Modelado por una función trapezoidal de parámetros a = 5.5, b = 6.0, c = 8.0 y d = 8.5
- **Completa:** Modelado por una función sigmoidal de parámetros a = 7.5 y b = 0.1

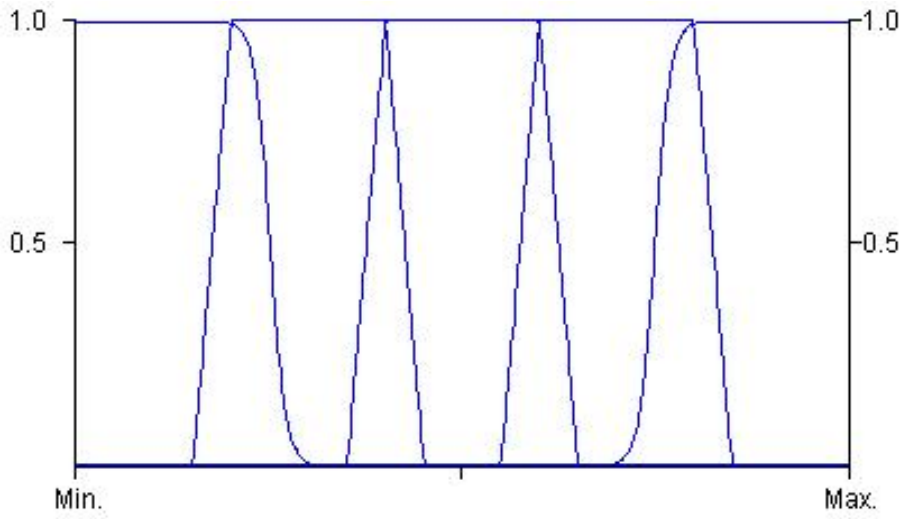

Figura B.4: Tipo de datos ConfianzaDefensa

B.4. Tipo de datos: ConfianzaDefensa

El modelado de los datos correspondientes al tipo ConfianzaDefensa puede verse en la figura B.4. Utiliza cinco funciones de pertenencia (Nada, Poco, Media, Mucho y Completa). Los parámetros concretos de cada una de ellas son:

- **Nada:** Modelado por una función sigmoidal de parámetros a = 2.5 y b = -0.1
- **Poco:** Modelado por una función trapezoidal de parámetros a = 1.5, b = 2.0, c = 4.0 y d = 4.5
- **Media:** Modelado por una función trapezoidal de parámetros a = 3.5, b = 4.0, c = 6.0 y d = 6.5
- **Mucho:** Modelado por una función trapezoidal de parámetros a = 5.5, b = 6.0, c = 8.0 y d = 8.5
- **Completa:** Modelado por una función sigmoidal de parámetros a = 7.5 y b = 0.1

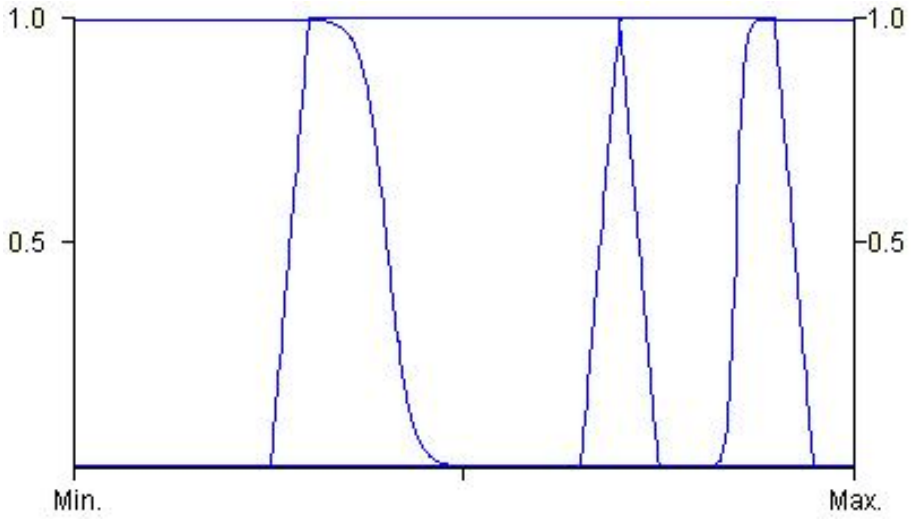

Figura B.5: Tipo de datos Contraejemplo

B.5. Tipo de datos: Contraejemplo

El tipo Contraejemplo está modelado por las funciones de pertenencia que pueden verse en la figura B.5. En concreto son cuatro: Erroneo, Dudoso, Claro y Rotundo. Los parámetros concretos de cada de sus funciones de pertenencia son:

- **Erroneo:** Modelado por una función sigmoidal de parámetros a = 4.0 y b = -0.15
- **Dudoso:** Modelado por una función trapezoidal de parámetros a = 2.5, b = 3.0, c = 7.0 y d = 7.5
- **Claro:** Modelado por una función trapezoidal de parámetros a = 6.5, b = 7.0, c = 9.0 y d = 9.5
- **Rotundo:** Modelado por una función sigmoidal de parámetros a = 8.5 y b = 0.05

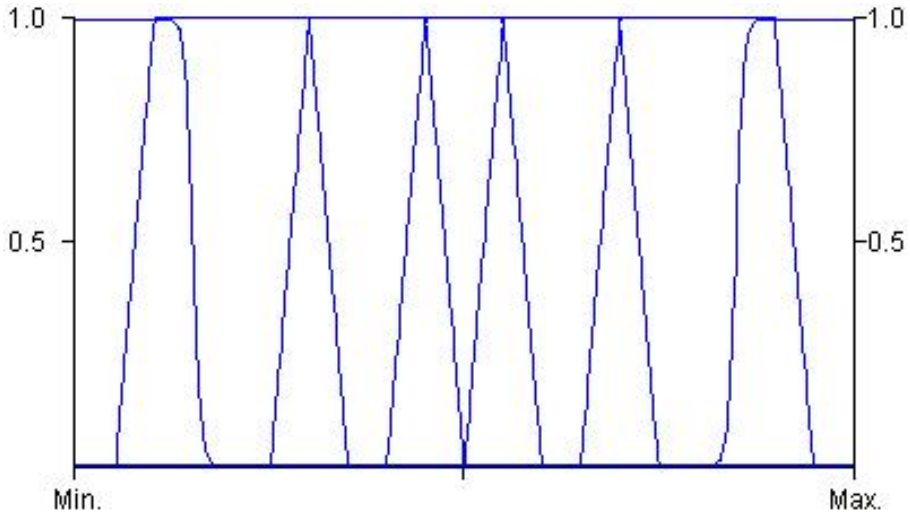

Figura B.6: Tipo de datos Credibilidad

B.6. Tipo de datos: Credibilidad

El tipo Credibilidad está modelado por siete funciones de pertenencia (Nada, Levemente, Poco, Algo, Bastante, Mucho y Creible), como puede verse en la figura B.6. Sus funciones de pertenencia están creadas con los parámetros:

- **Nada:** Modelado por una función sigmoidal de parámetros a = 1.5 y b = -0.05
- **Levemente:** Modelado por una función trapezoidal de parámetros a = 0.5, b = 1.0, c = 3.0 y d = 3.5
- **Poco:** Modelado por una función trapezoidal de parámetros a = 2.5, b = 3.0, c = 4.5 y d = 5.0
- **Algo:** Modelado por una función trapezoidal de parámetros a = 4.0, b = 4.5, c = 5.5 y d = 6.0
- **Bastante:** Modelado por una función trapezoidal de parámetros a = 5.0, b = 5.5, c = 7.0 y d = 7.5
- **Mucho:** Modelado por una función trapezoidal de parámetros a = 6.5, b = 7.0, c = 9.0 y d = 9.5
- **Creible:** Modelado por una función sigmoidal de parámetros a = 8.5 y b = 0.05

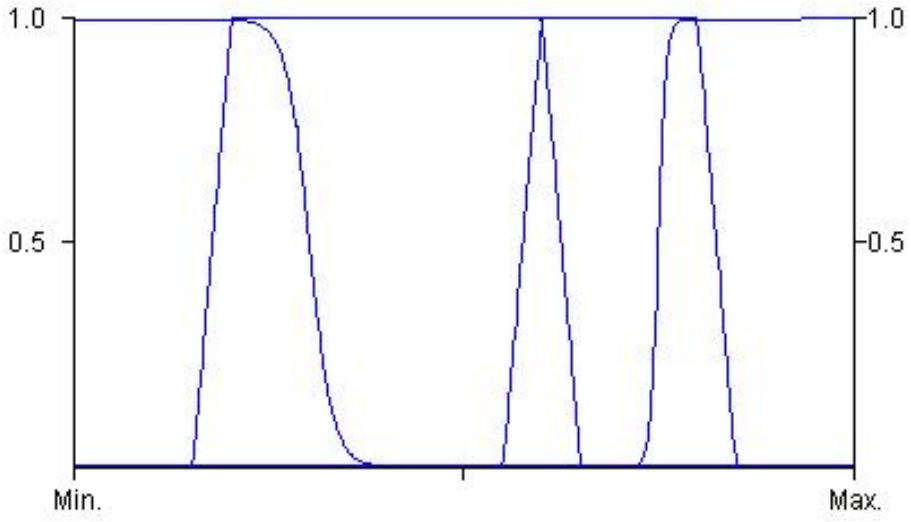

Figura B.7: Tipo de datos Defensa

B.7. Tipo de datos: Defensa

El modelado de los datos correspondientes al tipo Defensa es el de la figura B.7. Utiliza las cuatro funciones de pertenencia: Erronea, Dudosa, Clara y Rotunda. Los parámetros concretos de cada de ellas son:

- **Erronea:** Modelado por una función sigmoidal de parámetros a = 2.5 y b = -0.1
- **Dudosa:** Modelado por una función trapezoidal de parámetros a = 1.5, b = 2.0, c = 4.0 y d = 4.5
- **Clara:** Modelado por una función trapezoidal de parámetros a = 3.5, b = 4.0, c = 6.0 y d = 6.5
- **Rotunda:** Modelado por una función sigmoidal de parámetros a = 7.5 y b = 0.1

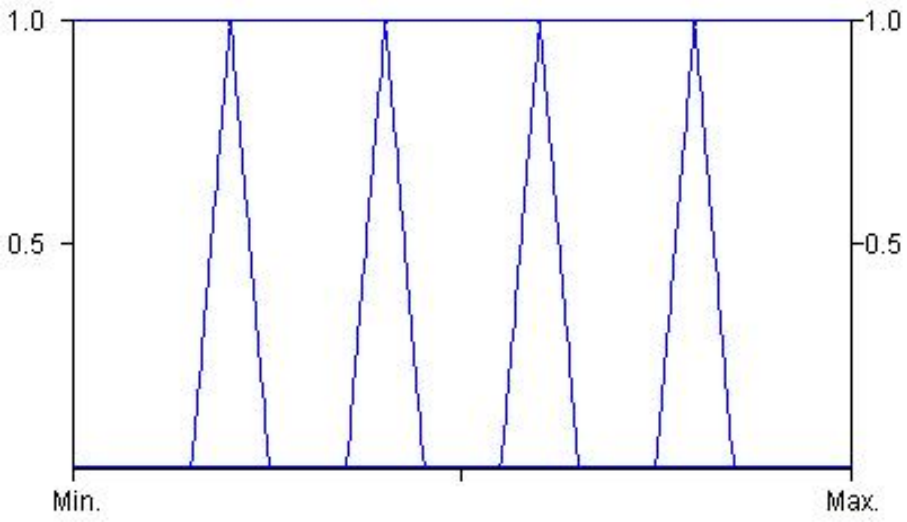

Figura B.8: Tipo de datos SimilitudCasos

B.8. Tipo de datos: SimilitudCasos

El tipo de datos SimilitudCasos en modelado mediante cuatro funciones de pertenencia (Distintos, Poco, Medio, Mucho e Iguales) como podemos observar en la figura B.8. Los parámetros concretos de cada de ellas son:

- **Distintos:** Modelado por una función rampa de parámetros a = 2.5 y m = -2.0
- **Poco:** Modelado por una función trapezoidal de parámetros a = 1.5, b = 2.0, c = 4.0 y d = 4.5
- **Medio:** Modelado por una función trapezoidal de parámetros a = 3.5, b = 4.0, c = 6.0 y d = 6.5
- **Mucho:** Modelado por una función trapezoidal de parámetros a = 5.5, b = 6.0, c = 8.0 y d = 8.5
- **Iguales:** Modelado por una función rampa de parámetros a = 7.5 y m = 2.0

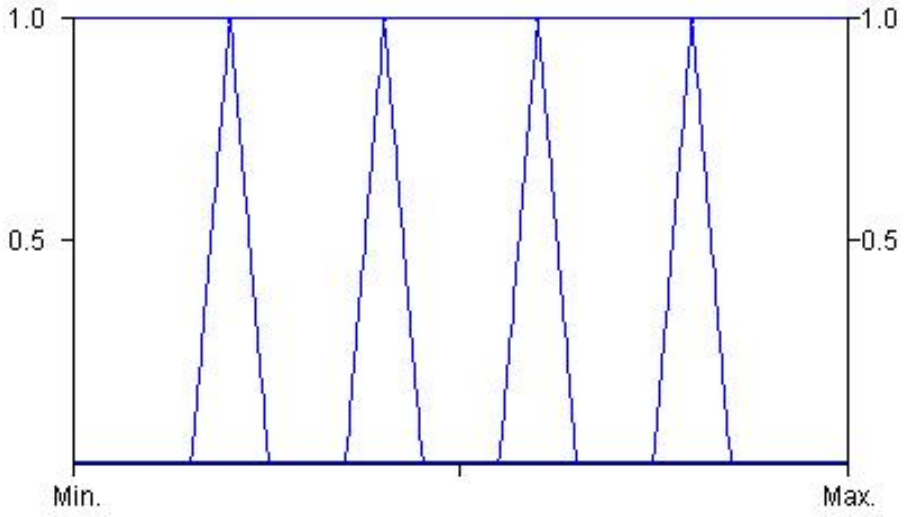

Figura B.9: Tipo de datos SimilitudPerfiles

B.9. Tipo de datos: SimilitudPerfiles

El modelado de los datos correspondientes al tipo SimlitudPerfiles está formado por cuatro funciones de pertenencia: Distintos, Poco, Medio, Mucho e Iguales. Este tipo de datos puede verse en la figura B.9. Los parámetros concretos de cada de sus funciones de pertenencia son:

- **Distintos:** Modelado por una función rampa de parámetros a = 2.5 y m = -2.0
- **Poco:** Modelado por una función trapezoidal de parámetros a = 1.5, b = 2.0, c = 4.0 y d = 4.5
- **Medio:** Modelado por una función trapezoidal de parámetros a = 3.5, b = 4.0, c = 6.0 y d = 6.5
- **Mucho:** Modelado por una función trapezoidal de parámetros a = 5.5, b = 6.0, c = 8.0 y d = 8.5
- **Iguales:** Modelado por una función rampa de parámetros a = 7.5 y m = 2.0

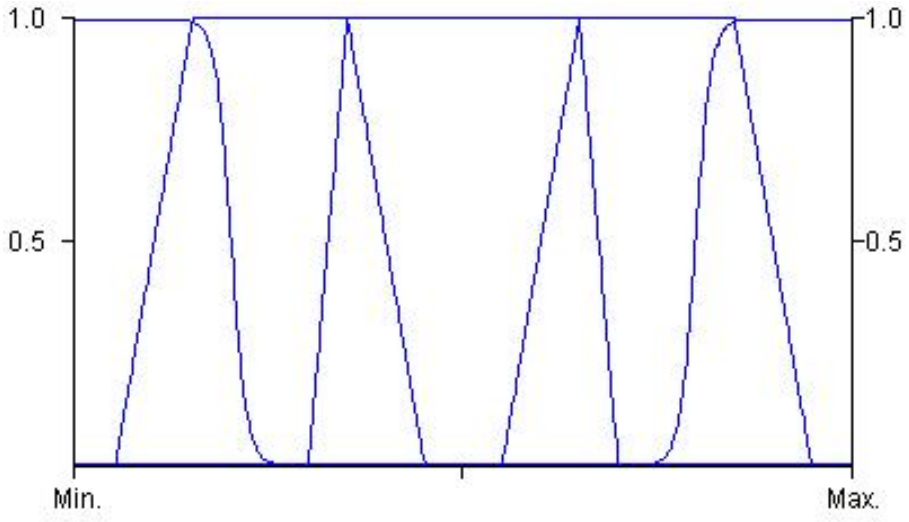

Figura B.10: Tipo de datos Valoracion

B.10. Tipo de datos: Valoracion

El último de los tipos de datos utilizados por el sistema fuzzy es el tipo Valoracion y puede verse en la figura B.10. Utiliza cinco funciones de pertenencia (Nada, Poco, Normal, Mucho y Totalmente). Los parámetros concretos de cada

- **Nada:** Modelado por una función sigmoidal de parámetros a = 2.0 y b = -0.1
- **Poco:** Modelado por una función trapezoidal de parámetros a = 0.5, b = 1.5, c = 3.5 y d = 4.5
- **Media:** Modelado por una función trapezoidal de parámetros a = 3.0, b = 3.5, c = 6.5 y d = 7.0
- **Mucho:** Modelado por una función trapezoidal de parámetros a = 5.5, b = 6.5, c = 8.5 y d = 9.5
- **Totalmente:** Modelado por una función sigmoidal de parámetros a = 8.0 y b = 0.1

Bibliografía

[1] Belén Díaz-Agudo, Pedro A. González-Calero, Juan A. Recio-García, and Antonio A. Sánchez-Ruiz-Granados. Building cbr systems with jcolibri. *Sci. Comput. Program.*, 69(1-3):68–75, 2007.

[2] Santiago Ontañón and Enric Plaza. An argumentation-based framework for deliberation in multi-agent systems. In Iyad Rahwan, Simon Parsons, and Chris Reed, editors, *ArgMAS*, volume 4946 of *Lecture Notes in Computer Science*, pages 178–196. Springer, 2007.

[3] editor. Leake, D. B. Case-based reasoning: Experiences, lessons, and future directions. In *Menlo Park, CA: AAAI Press/MIT Press, Menlo Park, CA*, 1996.

[4] Ian D. Watson and Dan Gardingen. A distributed case-based reasoning application for engineering sales support. In Thomas Dean, editor, *IJCAI*, pages 600–605. Morgan Kaufmann, 1999.

[5] Ivo Vollrath, Wolfgang Wilke, and Ralph Bergmann. Case-based reasoning support for online catalog sales. *IEEE Internet Computing*, 2(4):47–54, 1998.

[6] Hideo Shimazu. Expertclerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artif. Intell. Rev.*, 18(3-4):223–244, 2002.

[7] Rosalia Laza, A. Gómez, Reyes Pavón, and Juan M. Corchado. A case-based reasoning approach to the implementation of bdi agents. In *ECCBR Workshops*, pages 27–30, 2002.

[8] Lilian Noronha Nassif, José Marcos S. Nogueira, Mohamed Ahmed, Ahmed Karmouch, Roger Impey, and Flávio Vinícius de Andrade. Job completion prediction in grid using distributed case-based reasoning. In *WETICE*, pages 249–254. IEEE Computer Society, 2005.

[9] Oisín Boydell and Barry Smyth. Enhancing case-based, collaborative web search. [35], pages 329–343.

[10] Cindy Olivia, Chee-Fon Chang, Carlos F. Enguix, and Aditya K. Ghose. Case-based bdi agents: An effective approach for intelligent search on the world wide web. *AAAI*, 1999.

[11] Mecedes Gómez-Albarrán and Guillermo Jiménez-Díaz. Recommendation and students authoring in repositories of learning objects: A case-base reasoning approach. *SIIE*, pages 227–232, 2008.

[12] Barry Smyth and Paul Cotter. Personalized electronic program guides for digital tv. *AI Magazine*, 22(2):89–98, 2001.

[13] Jennifer Golbeck and James Hendler. Filmtrut: Movie recommendations using trust in web-based social networks. *CCNC*, 2006.

[14] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. 3986:93–104, 2006.

[15] Claudio Baccigalupo and Enric Plaza. A case-based song scheduler for group customised radio. [35], pages 433–448.

[16] R. C. Schank. *Dynamic Memory*. Cambridge Univ. Press, 1983.

[17] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.

[18] Derek G. Bridge, Mehmet H. Göker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. *Knowledge Eng. Review*, 20(3):315–320, 2005.

[19] Enric Plaza and Lorraine McGinty. Distributed case-based reasoning. *Knowledge Eng. Review*, 20(3):261–265, 2005.

[20] Lorraine McGinty and Barry Smyth. Collaborative case-based reasoning: Applications in personalised route planning. In *ICCBR*, pages 362–376, 2001.

[21] Santiago Ontañón and Enric Plaza. Arguments and counterexamples in case-based joint deliberation. In Nicolas Maudet, Simon Parsons, and Iyad Rahwan, editors, *ArgMAS*, volume 4766 of *Lecture Notes in Computer Science*, pages 36–53. Springer, 2006.

[22] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[23] Hamid Haidarian Shahri. Towards autonomous decision making in multi-agent environments using fuzzy logic. In Vladimír Marík, Jörg P. Müller, and Michal Pechoucek, editors, *CEEMAS*, volume 2691 of *Lecture Notes in Computer Science*, pages 247–257. Springer, 2003.

[24] Bijan Fazlollahi, Rustam M. Vahidov, and Rafik A. Aliev. Multi-agent distributed intelligent system based on fuzzy decision making. *Int. J. Intell. Syst.*, 15(9):849–858, 2000.

[25] Kerstin Bach, Meike Reichle, Alexander Reichle-Schmehl, and Klaus-Dieter Althoff. Implementing a coordination agent for modularised case bases. *13th UK Workshop on Case-Based Reasoning*, pages 1–12, 2008.

[26] Ali R. Montazemi. Case-based reasoning and multi-agent systems in support of tacit knowledge. *AAAI*, 1999.

[27] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. *Proceedings of the 1997 workshop on New security paradigms*, pages 48–60, 1998.

[28] Miquel Montaner, Beatriz López, and Josep Lluís de la Rosa. Developing trust in recommender agents. pages 304–305, 2002.

[29] Jennifer Golbeck and James A. Hendler. Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. 3257:116–131, 2004.

[30] John O'Donovan and Barry Smyth. Trust in recommender systems. pages 167–174, 2005.

[31] David W. McDonald. Recommending collaboration with social networks: a comparative evaluation. In Gilbert Cockton and Panu Korhonen, editors, *CHI*, pages 593–600. ACM, 2003.

[32] Jan Wendler, Pascal Gugenberger, and Mario Lenz. Cbr for dynamic situation assessment in an agent-oriented setting. *ECAI*, 1998.

[33] Hager Karoui, Rushed Kanawati, and Laure Petrucci. Cooperative cbr system for peer agent committee formation. *AP2PC*, 4461:51–62, 2008.

[34] Jerome Kelleher and Derek G. Bridge. An accurate and scalable collaborative recommender. *Artif. Intell. Rev.*, 21(3-4):193–213, 2004.

[35] Rosina Weber and Michael M. Richter, editors. *Case-Based Reasoning Research and Development, 7th International Conference on Case-Based Reasoning, ICCBR 2007, Belfast, Northern Ireland, UK, August 13-16, 2007, Proceedings*, volume 4626 of *Lecture Notes in Computer Science*. Springer, 2007.