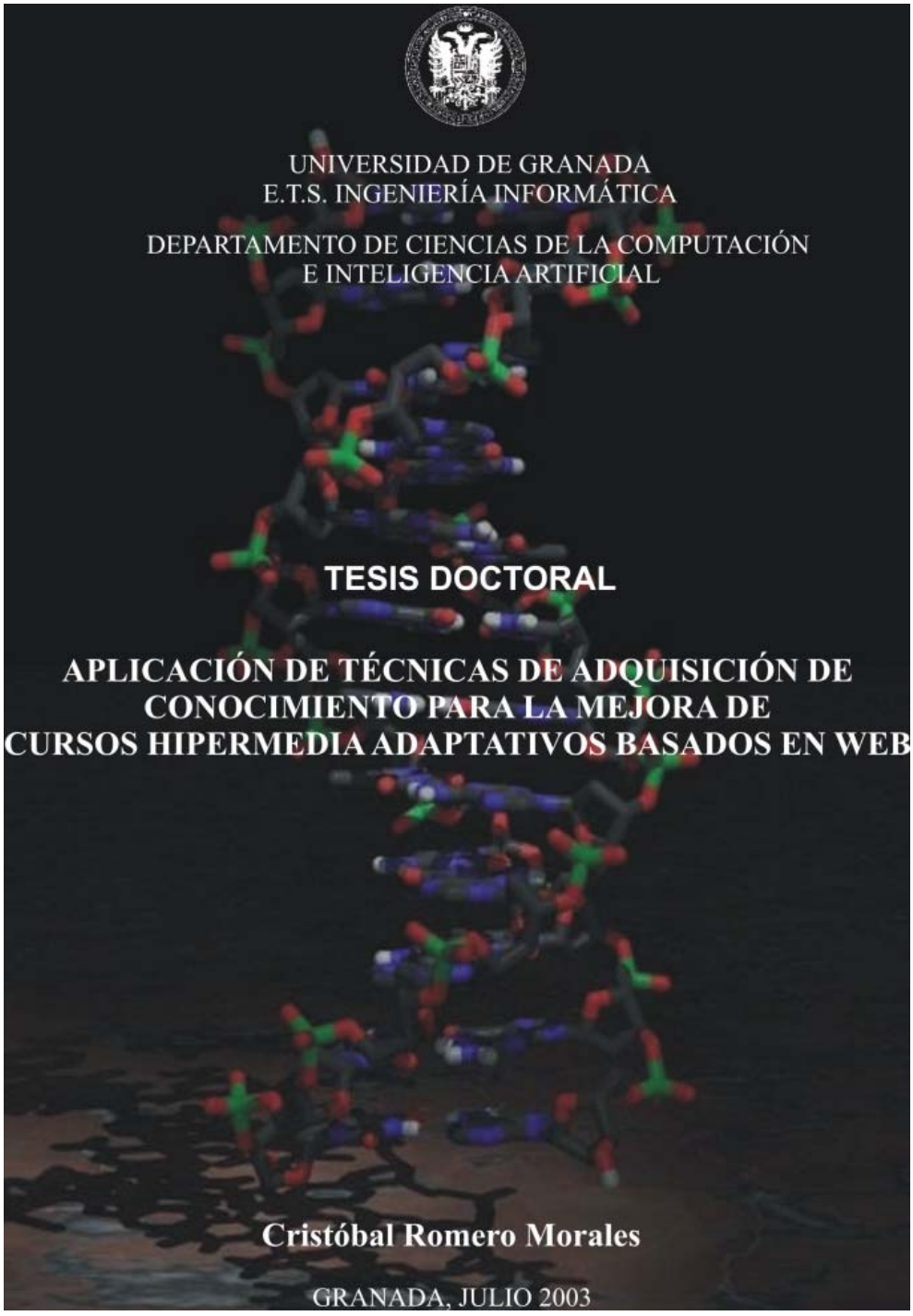

UNIVERSIDAD DE GRANADA
E.T.S. INGENIERÍA INFORMÁTICA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
E INTELIGENCIA ARTIFICIAL

TESIS DOCTORAL

APLICACIÓN DE TÉCNICAS DE ADQUISICIÓN DE CONOCIMIENTO PARA LA MEJORA DE CURSOS HIPERMEDIA ADAPTATIVOS BASADOS EN WEB

Cristóbal Romero Morales

GRANADA, JULIO 2003

Imagen de la Portada
Paul A. Thissen
www.ChemicalGraphics.com

UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍA INFORMÁTICA

Departamento de Ciencias de la Computación e Inteligencia Artificial

APLICACIÓN DE TÉCNICAS DE ADQUISICIÓN DE CONOCIMIENTO PARA LA MEJORA DE CURSOS HIPERMEDIA ADAPTATIVOS BASADOS EN WEB

TESIS DOCTORAL

Cristóbal Romero Morales

Granada, Julio 2003

APLICACIÓN DE TÉCNICAS DE ADQUISICIÓN DE CONOCIMIENTO PARA LA MEJORA DE CURSOS HIPERMEDIA ADAPTATIVOS BASADOS EN WEB

MEMORIA QUE PRESENTA

Cristóbal Romero Morales

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Julio 2003

DIRECTORES

Carlos de Castro Lozano.

Sebastián Ventura Soto.

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
E.T.S INGENIERÍA INFORMÁTICA, UNIVERSIDAD DE GRANADA

La memoria titulada **Aplicación de Técnicas de Adquisición de Conocimiento para la mejora de Cursos Hipermedia Adaptativos basados en Web**, que presenta D. Cristóbal Romero Morales para optar al grado de Doctor, ha sido realizada en el departamento de informática y análisis numérico de la Universidad de Córdoba bajo la dirección de los Doctores D. Carlos de Castro Lozano y D. Sebastián Ventura Soto, y bajo la tutela del Doctor D. Miguel Delgado Calvo-Flores del departamento de ciencias de la computación e inteligencia artificial de la Universidad de Granada

Granada, julio 2003.

El Doctorando:

Cristóbal Romero Morales

Los Directores:

Carlos de Castro Lozano Sebastián Ventura Soto

Tesis Doctoral Parcialmente Subvencionada por la Comisión Interministerial de Ciencia y Tecnología con el Proyecto TIC2002-04036-C05-02.

CICYT
TIC2002-04036-C05-02

Agradecimientos

A mis directores de tesis Carlos de Castro y Sebastián Ventura por toda la ayuda que me han prestado durante la realización de este trabajo y por haberse portado conmigo como unos amigos además de como directores.

Al profesor Paul de Bra por su ayuda y por su trato durante mi estancia en la Universidad Técnica de Eindhoven.

A mis compañeros de departamento Pedro González, José Antonio Herencia, Eloy Sanz, Rafael del Castillo, César Hervás, y ex-compañero y amigo Enrique López por la ayuda que me han ofrecido cuando los he necesitado.

A mis Padres, por el apoyo y comprensión que siempre han tenido conmigo.

A Ana, porque es lo mejor que me ha pasado durante la realización de esta memoria.

Gracias a Todos.

Índice General

Índice de Figuras

Índice de Algoritmos

Índice de Tablas

Capítulo 1
Introducción

En este primer capítulo se introducirá el trabajo que se describe a lo largo de esta memoria, titulada "Aplicación de Técnicas de Adquisición de Conocimiento para la mejora de Cursos Hipermedia Adaptativos basados en Web". Para ello, tras un planteamiento del problema y las áreas involucradas, se describen los objetivos que se pretenden conseguir. Después se enumeran las aportaciones tanto científicas como tecnológicas que se han pretendido obtener con el desarrollo de este trabajo. Por último, se comenta el contenido de cada uno de los capítulos que comprenden esta memoria.

CAPÍTULO 1: INTRODUCCIÓN

4.1 Planteamiento

La educación basada en el web (Web-based Education, WBE) es un área de investigación en expansión [Brusilovsky 2001], debido principalmente a los beneficios que presenta, entre los que cabe destacar la independencia de la localización física y de la plataforma utilizada. Un grave problema que presentan este tipo de cursos es que la mayoría de ellos no son más que una red de páginas hipertexto estáticas. Para solucionarlo [Brusilovsky 1999] ha aumentado su adaptabilidad e inteligencia, lo que ha dado lugar a los denominados Sistemas Adaptativos para Educación basada en Web (Adaptive Systems for Web-based Education, ASWE) [Brusilovsky 1999]. Dichos sistemas proceden de la evolución de los sistemas tutores inteligentes (Intelligent Tutoring Systems, ITS) y los sistemas hipermedia adaptativos (Adaptive Hypermedia Systems, AHS), y comparten las características más deseables de ambos: aumento de la interacción con los usuarios y adaptación de los contenidos a las necesidades de éstos.

El desarrollo de un ASWE es una actividad laboriosa [Hérin et al. 2002], tanto más compleja cuanto mayor es el número de posibilidades de adaptación que se desea ofrecer. Por ejemplo, se deben elegir los contenidos que serán mostrados y establecer la estructura del curso, determinar cuáles de estos contenidos son los más adecuados para cada usuario potencial y cuál es la mejor organización de estos contenidos, aprovechando las posibilidades de adaptación ofrecidas por el sistema. Sin embargo, un diseño cuidadoso no suele ser suficiente sino que, en la mayoría de los casos, suele ser necesario realizar una evaluación posterior basada en los resultados obtenidos por los usuarios del mismo. Sería además deseable que esta actividad de autoevaluación se realizara de forma continua mientras el curso se encuentre a disposición de los alumnos [Ortigosa & Carro 2002]. Para ello es necesario utilizar herramientas y metodologías capaces de observar el comportamiento de los estudiantes y de asistir al profesor en el proceso de mejora continua de los cursos adaptativos, detectando de forma semiautomática posibles errores, carencias o mejoras que puedan realizarse en los cursos ya generados.

Para abordar el problema planteado, se están comenzando a utilizar técnicas de descubrimiento de conocimiento o minería de datos [Zaïne 2001] que asisten al profesor en la validación de los cursos. Estas técnicas permiten descubrir nuevo conocimiento a partir de los datos de utilización del curso por parte de los alumnos. La idea se ha aplicado ya con éxito en los sistemas de comercio electrónico[1], donde ha llegado a ser muy popular [Spiliopoulou 2000]. Sin embargo, se ha hecho muy

[1] Por ejemplo, se han desarrollado herramientas para la comprensión del comportamiento de los clientes y ofrecerle productos relacionados con sus hábitos de compra para aumentar las ventas.

poco en este sentido para la comprensión del comportamiento de los alumnos en los entornos de aprendizaje a distancia basados en web.

La minería de datos (Data Mining, DM) es un área multidisciplinar, donde confluyen multitud de paradigmas de cómputo tales como: la construcción de árboles de decisión, la inducción de reglas, las redes neuronales artificiales, el aprendizaje basado en instancias, el aprendizaje bayesiano, la programación lógica y varios tipos de algoritmos estadísticos [Witten & Frank 2000]. El objetivo de todos estos paradigmas es descubrir conocimiento nuevo, útil e interesante. En este sentido, desde hace relativamente poco tiempo se ha planteado la posibilidad de utilizar Algoritmos Evolutivos para resolver esta tarea [Freitas 2001]. Las principales ventajas del uso de los Algoritmos Evolutivos en el descubrimiento concreto de reglas de predicción es su capacidad para realizar una búsqueda global y el tratamiento óptimo del problema de la interacción de los atributos que la mayoría de los algoritmos voraces de inducción de reglas que se han utilizado tradicionalmente.

En el contexto de descubrimiento de reglas de predicción utilizando Algoritmos Evolutivos, un individuo corresponderá a una regla o conjunto de reglas candidatas; la función de ajuste corresponderá a alguna medida de la calidad de la regla o conjunto de reglas; el procedimiento de selección utilizará los valores del ajuste de los individuos para seleccionar las mejores reglas o conjunto de reglas; los operadores genéticos transformarán una regla candidata en otra. Los algoritmos evolutivos realizarán una búsqueda en el espacio de reglas candidatas como haría un método de inducción de reglas. La principal diferencia entre los algoritmos evolutivos empleados para el descubrimiento de conocimiento y los algoritmos de inducción de reglas es la estrategia de búsqueda empleada. En efecto, los algoritmos clásicos de aprendizaje inductivo suelen utilizar una estrategia voraz de búsqueda local, mientras que los algoritmos evolutivos utilizan una estrategia de búsqueda global inspirada en la selección natural.

Siempre es deseable que el conocimiento descubierto sea interesante [Bayardo & Agrawal 1999], entendiendo como tal que sea interesante, novedoso y pueda ser aprovechado por el usuario. Para satisfacer estas exigencias, se deben extraer sólo un subconjunto de reglas interesantes de entre todas las posibles reglas descubiertas y, aunque los algoritmos de minería de datos normalmente descubren una gran cantidad de reglas exactas y comprensibles, no suelen ser reglas interesantes al ser éste un objetivo más ambicioso y difícil. Existen dos métodos para la selección de reglas interesantes [Liu et al. 2000], denominados métodos subjetivos y objetivos. Los primeros son dependientes del dominio y están dirigidos por el usuario, que suele especificar plantillas de reglas indicando qué combinación de atributos puede ocurrir en las reglas para que éstas sean consideradas interesantes. En cambio los métodos objetivos están dirigidos por datos y son independientes del dominio,

estando basados en la idea de comparar entre sí las reglas descubiertas a través del uso de alguna métrica. En este sentido existen un gran número de métricas descritas en la bibliografía [Tan & Kumar 2000], cada una de las cuáles se centra en un aspecto concreto de la calidad de las reglas descubiertas. Sin embargo, para la tarea que nos ocupa, no se ha encontrado una medida que conduzca a un rendimiento significativamente mejor que las demás. Por esta razón, se hace necesario considerar varias medidas simultáneamente. Una forma simple de hacer esto es utilizar una métrica combinada, que pondera mediante pesos algunas de las métricas primitivas [Noda et. al 1999]. Sin embargo, ésta no es una buena aproximación debido a que las medidas utilizadas pueden estar en conflicto y ser no conmensurables, en el sentido de que evalúan aspectos muy diferentes de la solución candidata. Este problema sugiere el uso de una aproximación multiobjetivo [Freitas 2002] para el descubrimiento de reglas, donde el valor de la función ajuste a optimizar no es un valor escalar único, sino un vector de valores, donde cada valor mide un aspecto diferente de la calidad de la regla. Aunque la bibliografía de Algoritmos Genéticos Multiobjetivo (MultiObjective Evolutionary Algorithms, MOEA) es amplia [Fonseca & Fleming 2000] [Deb 2001], la utilización de MOEA en el descubrimiento de reglas parece relativamente inexplorada.

4.2 Objetivos

El objetivo principal de este trabajo consiste en evaluar la capacidad de distintas técnicas de aprendizaje como herramientas en la adquisición de conocimiento útil para la mejora de cursos hipermedia adaptativos basados en web. Para poder alcanzar este objetivo general se han determinado varios subobjetivos que se van a describir a continuación.

- **Desarrollar una metodología para la construcción de cursos hipermedia adaptativos basados en web que permita la mejora de dichos cursos.** Este primer objetivo se ha materializado en la propuesta de una nueva metodología dinámica de construcción de los cursos hipermedia adaptativos basados en web [Romero et al. 2002c], en la que la información generada por los usuarios (los alumnos del curso) es utilizada para extraer conocimiento útil en la mejora del sistema.

- **Implementar una arquitectura de ASWE.** La motivación de este objetivo, además de la puesta en marcha de la metodología propuesta, es la de disponer de un sistema real sobre el que experimentar, obteniendo información real basada en el uso que hagan los alumnos del curso y extrayendo conocimiento a partir de estas conclusiones. Como veremos

posteriormente, se ha construido un curso basado en esta arquitectura sobre el Sistema Operativo Linux [Romero et al. 2002c] y se ha testado sobre 40 alumnos del I.E.S. Gran Capitán de Córdoba y 10 expertos de Linux, los cuales han generado la información que se ha utilizado para abordar los siguientes objetivos de este trabajo.

- **Utilizar técnicas de descubrimiento de conocimiento para extraer información que asista al educador en la evaluación de cursos.** La principal motivación de este objetivo es comprobar la validez de la propuesta metodológica realizada, que requiere la existencia de algoritmos que extraigan conocimiento útil para la mejora de los sistemas adaptativos implementados. El conocimiento descubierto se presentará al profesor en forma de reglas de predicción [Freitas 2002], ya que se trata de un formato intuitivo y comprensible para los responsables didácticos de la aplicación. Además, como se verá posteriormente, este tipo de reglas podría incorporarse directamente en el modelo del alumno.

Para la realización de esta tarea de descubrimiento de reglas de predicción se van a utilizar algoritmos clásicos (algoritmos de inducción de reglas, algoritmos de construcción de árboles de clasificación y algoritmos de descubrimiento de reglas de asociación) y algoritmos evolutivos (programación genética basada en gramática) [Romero et al. 2003a], y se compararán los resultados obtenidos en términos de número de reglas producidas y calidad de las mismas.

4.3 Contribuciones

El desarrollo de este trabajo ha aportado varias contribuciones que se pueden ver desde dos puntos de vista: tecnológico y científico.

Desde el punto de vista tecnológico se han desarrollado varias herramientas software:

- **Herramienta autor para la construcción de cursos adaptativos basados en web**. Se ha desarrollado una herramienta autor específica [Romero et al. 2003b] para facilitar la construcción de cursos adaptativos basados en web que utilicen el modelo de enseñanza y la metodología propuesta.

- **Herramienta visual para minería de datos de cursos adaptativos basados en web.** Se ha desarrollado una herramienta gráfica específica [Romero et al. 2003a] que permite realizar todos los pasos del proceso de

descubrimiento de conocimiento a partir de datos de utilización de los alumnos de los cursos.

- **Curso adaptativo basado en web de Linux.** Se ha desarrollo un curso del sistema operativo Linux [Romero et al. 2002c] que implementa la metodología y el modelo propuesto. Los datos de utilización obtenidos con este curso han servido para la realización de las pruebas con los algoritmos de descubrimiento de reglas.

Desde el punto de vista científico este trabajo tiene las siguientes aportaciones:

- **Metodología para el desarrollo de sistemas adaptativos para la enseñanza basada en web.** Se ha definido una nueva metodología dinámica [Romero et al. 2002a] para el desarrollo de cursos adaptativos basado en web y su mejora a partir de la propia información de utilización de los alumnos.

- **Arquitectura de sistemas adaptativos para la enseñanza basada en web.** Se ha desarrollado una arquitectura de ASWE [Romero et al. 2002c] que permite la recogida de información de utilización, para poder realizar posteriormente tareas de descubrimiento de conocimiento.

- **Análisis de algoritmos para la obtención de reglas de predicción interesantes.** Se ha realizado la comparación de distintos tipos de algoritmos de descubrimiento de reglas de predicción interesantes, tanto clásicos como evolutivos [Romero et al. 2003c].

4.4 Contenidos

A continuación se van a describir brevemente los distintos capítulos en los que se ha dividido esta memoria:

El capítulo *Antecedentes* pretende realizar una revisión del estado en que se encuentra la investigación sobre la mejora de los sistemas educativos basados en web mediante técnicas de minería de datos, y la utilización de los algoritmos evolutivos para el descubrimiento de reglas.

El capítulo *Diseño e Implementación del ASWE* describe la metodología propuesta para la mejora de los ASWEs utilizando la información de los usuarios y el diseño del sistema que se ha desarrollado para obtener dichos datos, así como los aspectos más relevantes relativos a su implementación.

El capítulo *Descripción de la Información Inicial* muestra los datos de utilización de un curso sobre el Sistema Operativo Linux y que posteriormente se han empleado para realizar las pruebas, así como el tratamiento que se ha realizado sobre dicha información antes de su utilización.

El capítulo *Algoritmos Evolutivos para el Descubrimiento de Reglas de Predicción* describe la utilización de la GBGP para resolver la tarea de descubrimiento de reglas de predicción interesantes. Se han realizado varias propuestas utilizando distintos tipos de MOEAs y se han comparado con una serie de algoritmos clásicos de descubrimiento de reglas que se han adaptado para resolver la tarea de descubrimiento de reglas de predicción interesantes.

El capítulo *Descripción de la Información Descubierta* presenta los distintos tipos de reglas que se pueden obtener y su utilización para la mejora de ASWEs mostrando algunos ejemplos de reglas concretas descubiertas.

El último capítulo *Comentarios Finales* muestra las conclusiones y las futuras líneas de investigación que se plantean tras este trabajo.

Finalmente se presentan una serie de apéndices donde se muestran los *Términos Utilizados* a lo largo del trabajo, una *Introducción a los Algoritmos Evolutivos*, las distintas *Métricas de para la Valoración de la Calidad de las Reglas* existentes, y las *Herramientas* desarrolladas específicamente para la *Construcción de Cursos* y para el *Descubrimiento de Conocimiento* en ASWEs.

Capítulo 2
Antecedentes

El objetivo de este capítulo es dar una imagen general sobre el estado en que se encuentra la investigación relacionada con el uso de técnicas de adquisición de conocimiento para la mejora de los sistemas de aprendizaje basados en Web. Tras presentar en qué consisten este tipo de sistemas educativos, y su evolución en los últimos años, se describirá es su arquitectura, a fin de comprender qué posibilidades ofrecen para la adquisición de información utilizable en el proceso de mejora propuesto en esta memoria. En segundo lugar, se expondrá el estado en que se encuentra la investigación relacionada con la extracción de conocimiento aplicada a los sistemas educativos basados en web, eje fundamental del trabajo realizado. Por último, se introducirán una serie de cuestiones relacionadas con el problema del descubrimiento de reglas.

CAPÍTULO 2: ANTECEDENTES

2.1 Sistemas Adaptativos para Educación basada en Web

Los Sistemas Adaptativos para Educación basada en Web (ASWE) son un nuevo tipo de sistemas educativos procedentes de la evolución de los Sistemas Tutores Inteligentes (Intelligent Tutoring Systems, ITS) y los Sistemas Hipermedia Adaptativos (Adaptive Hipermedia Systems, AHS), con los que comparten una serie de propiedades [Brusilovsky 1999], [Brusilovsky 2001]. Con respecto a los primeros, los ASWE también utilizan información del dominio, estudiante y estrategias de tutorización para permitir un aprendizaje individualizado y flexible. Con respecto a los AHS, los ASWE también establecen un modelo que utilizan para adaptar el contenido y enlaces de las páginas hipermedia al usuario.

Los Sistemas Hipermedia Adaptativos construyen un modelo de los objetivos, preferencias y conocimiento de cada usuario y lo utilizan durante la interacción con dicho usuario para adaptarse a sus necesidades. Aunque existen AHS con fines no educativos, la educación ha sido una de las primeras áreas de aplicación de estos sistemas, que son los más populares y mejor investigados en la actualidad. A continuación, se va a realizar una revisión de lo que ha sido la evolución de esta disciplina en los últimos años, a fin de centrar el tema objeto de este trabajo.

2.1.1 Evolución histórica

La aparición de los primeros ASWE se remonta al período comprendido entre los años 1990 y 1996. En esta época comienzan a aparecer sistemas cuyo origen se puede atribuir a dos áreas dentro de la multimedia: por un lado aparecieron sistemas en el área de los ITS, los cuales intentaban extender el modelado del estudiante tradicional y las aproximaciones de adaptación con los componentes hipermedia [Beaummont 1994], [Brusilovksy et al. 1993], [Gonsckorek & Herzog 1995], [Pérez et al. 1995]. Por otro lado, en el área de los AHS se desarrollaron sistemas en los que se intentaba realizar la adaptación individual a los estudiantes del curso [De Bra 1996] [de La Passadiere & Dufresne 1992] [Hohol et al. 1996], [Kay & kummerfeld 1994].

Sin embargo, es a partir del año 1996 cuando la comunidad investigadora empieza a interesarse por este tipo de sistemas, debido principalmente a dos factores: la acumulación y consolidación de experiencia investigadora en el campo y, lo que es más importante, el rápido crecimiento del uso de la World Wide Web, que demandaba sistemas con un grado de adaptación elevado debido a la naturaleza

tan variable que presenta la audiencia a la que van dirigidas este tipo de aplicaciones. De este modo, mientras que la gran mayoría de los sistemas adaptativos desarrollados antes de 1996 utilizaban hipertextos e hipermedia clásicos, los desarrollados a partir de esta fecha eran todos ASWEs. De principios de esta fecha son sistemas como ELM-ART [Brusilovsky et al. 1996], InterBook [Brusilovsky et al. 1998], PT [Kay, & Kummerfeld 1997], y 2L670 [De Bra 1996], sistemas pioneros que influyeron en la creación de sistemas posteriores, entre los que se encuentran: Medtech [Eliot et al. 1997], AST [Specht et al. 1997], ADI [Schöch et al. 1998], HysM [Kayama 1998], AHM [Pilar da Silva et al. 1998], MetaLinks [Murray et al. 1998], CHEOPS [Negro et al. 1998], RATH [Hockemeyer et al. 1998], TANGOW [Carro et al. 1999], Arthur [Gilbert & Han 1999], CAMELEON [Laroussi & Benahmed 1998], KBS-Hyperbook [Henze et al. 1999], AHA! [De Bra et al., 1998], SKILL [Neumann, & Zirvas 1998], Multibook [Steinackeret al. 1998], ACE [Specht, & Oppermann 1998] y ART-Web [Weber 1999].

En la actualidad, la tendencia más importante en el área de los ASWE es el desarrollo de entornos comprensivos para educación basada en el web [De Bra, et al., 1998], [Henze & Nejdl 2000], [Papanikolaou et al. 2000], [Specht et al., 1998], [Steinacker et al. 1998], [Stern & Woolf 2000], [Süß et al. 2000], [Weber 1999]. Estos entornos cada vez se aproximan más a herramientas comerciales para el desarrollo de cursos basados en el Web, tales como WebCT [WebCT 1999], CourseInfo [Blackboard 1999] o TopClass [WBT Systems 1999]. Los desarrolladores de plataformas para educación hipermedia adaptativa están muy interesados en hacer que sus sistemas sean apropiados para manejar cursos reales, a diferencia de las herramientas de mediados de los 90, que eran plataformas de carácter experimental.

2.1.2 Adaptación en los ASWE

La adaptación es la principal característica de los ASWE. Existen varias tecnologías de adaptación que se han adoptado de los ITS como son la secuenciación del currículo, análisis inteligente de las soluciones del alumno, soporte de solución de problemas interactivo y soporte de solución de problemas basados en ejemplos y otras tecnologías que proviene de los AHS como son la adaptación de la presentación y la adaptación de la navegación. A continuación, examinaremos brevemente en qué consiste cada una de estas técnicas de adaptación [Brusilovsky 1999], así como los sistemas en los que dichas técnicas se han implantado:

- **Secuenciación del currículum**. También denominada planificación de la enseñanza, consiste en ayudar al estudiante a encontrar el camino optimo a través del material de aprendizaje, proporcionando la secuencia planificada

de unidades de conocimiento y tareas a realizar. El ejemplo clásico es el sistema BIP [Barr et al. 1976] en el dominio de la enseñanza de la programación. Otros ejemplos más actuales son ITEM-IP [Brusilovsky 1992], SCENT-3 [Brecht et al. 1989], ELM-ART [Brusilovsky et al. 1997], CALAT [Nakabayashi et al. 1997], InterBook [Brusilovsky & Shwarz 1997], AST [Specht et al. 1997], MANIC [Stern et al. 1997] y DCG [Vassileva 1997]. Dentro de esta modalidad de adaptación, hay que distinguir entre dos técnicas diferentes, de alto y bajo nivel. En la primera se planifican contenidos, es decir, el sistema indica al alumno el siguiente concepto a enseñar, sin plantear cuál o cuáles son las siguientes tareas a realizar. La secuenciación de bajo nivel se realiza a nivel de tareas, planteando al alumno la siguiente tarea a realizar.

- **Análisis inteligente de las soluciones del alumno.** Analiza las respuestas de los estudiantes a los problemas y asesora al alumno en cuestiones como errores cometidos, causas de estos errores, carencias observadas, etc. Estos analizadores inteligentes proporcionan una retroalimentación del error al estudiante y actualizan el modelo del alumno. Un ejemplo clásico en el dominio de la enseñanza de la programación es PROUST [Johnson 1986], siendo ejemplos más actuales CAMUS-II [Vanneste 1994] y ELM-PE [Weber & Möllenberg 1995]. En la actualidad existen al menos dos ASWE que implementan análisis inteligente de las soluciones: ELM-ART [Brusilovsky 1996], un sistema para la enseñanza de la programación en lenguaje LISP y WITS [Okazaki et al. 1997], un sistema para la enseñanza del cálculo diferencial.

- **Resolución interactiva de problemas.** Proporciona al estudiante ayuda inteligente en cada paso de la solución de un problema. El sistema puede ver las acciones del estudiante y, a partir de éstas, proporcionar ayuda y actualizar el modelo del estudiante. El ejemplo típico para el dominio de la enseñanza de la programación es LISP-TUTOR [Anderson & Reiser 1985], siendo ejemplos más actuales ACT Programing Tutor [Corbertt & Anderson 1992] y GRACE [Mckendree et al. 1992]. El soporte de solución de problemas interactivo no es tan popular en aplicaciones Web como en aplicaciones locales, debido a la lentitud de la respuesta de los servidores web cuando se intenta controla cada paso. Sin embargo, el sistema PAT-Online [Brusilovsky et al. 1997] ha implementado este tipo de adaptación.

- **Solución de problemas basados en ejemplos.** El alumno resuelve problemas utilizando la ayuda de ejemplos de experiencias anteriores. El sistema ayuda al alumno mostrándole los ejemplos más relevantes. Un ejemplo del dominio de la enseñanza de programación es ELM-PE [Weber

& Möllenberg 1995]. ELM-ART [Brusilovsky 1996] es el único sistema basado en web que utiliza esta tecnología.

- **Adaptación de la presentación.** Adapta el contenido de las páginas hipermedia dependiendo de los objetivos de un usuario, su conocimiento y otra información almacenada en el sistema. En los sistemas que utilizan adaptación de la presentación las páginas no son estáticas, sino que son generadas de forma automática, adaptándolas a cada usuario. Un ejemplo en el dominio de la enseñanza de programación es ITEM/IP [Brusilovsky 1992]. Otros sistemas que implementan presentación adaptativa son: C-Book [Kay & Kummerfeld 1994] y AHA [De Bra & Calvi 1997].

- **Adaptación de la navegación.** Permite orientar al usuario en la navegación cambiando la apariencia de los enlaces visibles. El sistema puede adaptativamente ordenar, anotar u ocultar enlaces para facilitar la elección del siguiente elemento a visitar. La adaptación de la navegación se puede considerar como una extensión de la secuenciación del currículo en un contexto hipermedia, ya que comparte el mismo objetivo de ayudar al estudiante a encontrar el camino optimo a través del material de aprendizaje. Sin embargo, esta técnica es menos conductista que la secuenciación tradicional, ya que es el estudiante quien decidirá el siguiente elemento a visitar de entre unos cuantos posibles. Ejemplos de sistemas de ejecución local son ISIS-Tutor [Brusilovsky & Pesin 1994] y Hypadapter [Hohl et al. 1996]. La forma más popular de adaptación de la navegación en el web es la anotación, consistente en añadir comentarios o imágenes junto a los enlaces para recomendarlos o no. Algunos ejemplos de sistemas que utilizan esta técnica son: ELM-ART [Brusilovsky 1996], InterBook [Brusilovsky & Schwarz 1997] y AST [Specht et al. 1997]. Otra tecnología popular es la deshabilitación de enlaces, consistente en habilitar o deshabilitar los enlaces que se le muestran al usuario. Esta técnica es la que se implementa en los sistemas AHA [De Bra & Calvi 1997] y Albatros [Lai et al. 1995].

2.1.3 Arquitectura de los ASWE

La arquitectura genérica de un ASWE es muy parecida a la de un ITS, de forma que ambos están formados [De Bra et. al, 1999] por un modelo del alumno, un modelo del dominio, un modelo tutor y un interfaz gráfico (ver Figura 2.1).

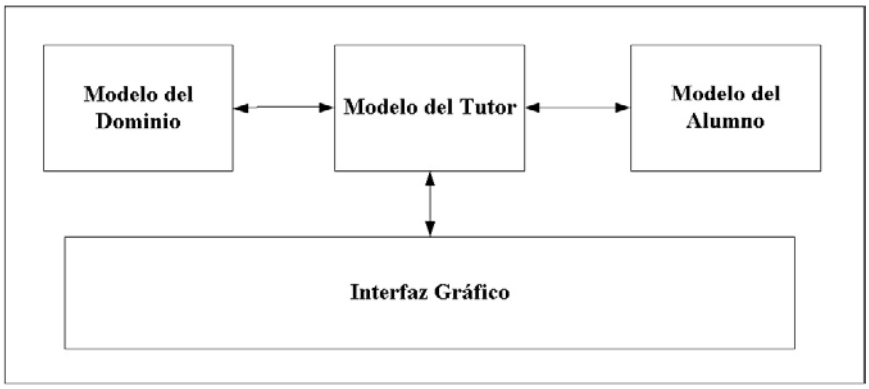

Figura 2.1: Arquitectura genérica de un Sistema Adaptativo para la Enseñanza basada en Web.

- **Modelo del dominio.** Contiene la información que se desea enseñar a los alumnos, almacenando tanto los elementos que se pueden presentar como la estructura o forma en que éstos se encuentran organizados, así como información de su disponibilidad dependiendo de determinadas situaciones. La información del dominio puede ser de muy diversos tipos: textos, imágenes, ejercicios, etc. además de información relativa a importancia, dificultad, relaciones, etc. El objetivo del sistema adaptativo o tutor inteligente es que el alumno aprenda todo o parte del conocimiento representado en el modelo del dominio, dependiendo de sus objetivos.

- **Modelo del alumno.** El modelo del alumno o del estudiante contiene la información sobre el alumno (a quién enseñar). Esta información incluye preferencias y objetivos del alumno, así como la información generada en la interacción con el sistema. Esta información es la que se puede utilizar en el proceso de adaptación del curso. Algunas de las entidades que comprenden este módulo son: conocimiento del alumno (nivel, velocidad de aprendizaje, tipo de aprendizaje, preferencias sobre los elementos multimedia utilizados, etc.), material didáctico utilizado (información que ya se ha presentado al alumno), historia del alumno (datos de interés sobre la interacción del alumno), etc.

- **Modelo del tutor.** El modelo del tutor, también denominado modelo de adaptación o modelo pedagógico es el motor del sistema o tutor en sí (el cómo enseñar), siendo responsable de adaptar la información al alumno en particular, en función de la información disponible sobre éste. Determina el plan de enseñanza y, para ello, suele realizar las siguientes funciones: selección del material didáctico que se va ofreciendo (en función del nivel de conocimiento del alumno en cada momento), establecimiento de la

accesibilidad del alumno a los contenidos (en función de los conocimientos adquiridos y del nivel que presente en cada uno de éstos) y evaluación (que se suele realizar en base a los resultados obtenidos en las distintas actividades realizadas).

- **Interfaz gráfico.** El interfaz es el entorno donde se produce la comunicación entre el alumno y el sistema. Este elemento puede ser estudiado desde dos puntos de vista diferentes: desde el punto de vista del usuario, el interfaz es el medio a través del cual va aprendiendo la información que se le presenta, interactuando con él. Desde el punto de vista del tutor, el interfaz es el medio a través del cual se enseña, mostrando en cada momento la información que se considera más adecuada.

2.1.4 Construcción de los ASWEs

La metodología clásica utilizada para la construcción de cursos [Hérin et al. 2002], proviene de observaciones de la vida real sobre la experiencia en enseñanza, donde un profesor construye un curso siguiendo una serie de pasos (ver Figura 2.2) que son los siguientes:

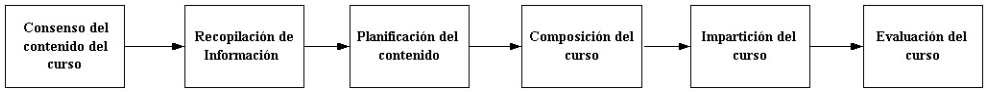

Figura 2.2: Metodología clásica para la construcción de ASWEs.

1. **Consenso del contenido del curso.** En esta etapa se realiza el planteamiento de los objetivos del curso, estableciendo o consensuando cuáles son los principales conceptos que se van a enseñar. Esta etapa es común para cualquier tipo de curso, ya sea presencial, basado en web, etc.

2. **Recopilación de información sobre el tema.** Este paso consiste en la recopilación de toda la información disponible: libros y artículos relacionados con la temática del curso y, en el caso de sistemas basados en web, otros elementos en formato digital: imágenes, videos, textos, sonidos, etc. Muchos de estos materiales pueden ser procedentes de otros cursos y/o contenidos web relacionados con el tema del curso que se va a construir. El curso consistirá por tanto en una composición de distintos elementos procedentes de fuentes muy heterogéneas. Será el autor del curso el responsable de darles una coherencia interna.

3. **Planificación de los contenidos.** Cada profesor tiene su propio método de enseñanza según el cual decide cuáles son los conceptos relevantes a enseñar en un curso y la secuencia de presentación de los mismos. Dicha secuencia va a definir el plan de estudios o curriculum del curso. Esta etapa, también común en el desarrollo de otro tipo de cursos, implica en el caso de cursos adaptativos basados en web, el diseño del modelo del dominio que, como ya se ha comentado, es la parte del sistema que almacena tanto la información que se va a presentar como la organización de ésta.

4. **Composición del curso.** Una vez que se tiene la planificación, se asocia cada concepto con las piezas de conocimiento que se han seleccionado de los diferentes libros de texto, cursos web, etc de la etapa anterior. Para el caso particular de cursos web esta etapa consistiría en la construcción o montaje del curso web, desarrollándose tanto el contenido del curso (un conjunto de páginas Web que contienen el material docente y representan los distintos conceptos - modelo del dominio) como su funcionamiento (determinado por la estructura de los conceptos según la planificación de los contenidos y la adaptación de la accesibilidad y contenidos a cada tipo de alumno - modelo del tutor).

5. **Impartición del curso.** Es la sesión de enseñanza donde el curso que se ha compuesto se le presenta a los estudiantes. En el caso particular de los cursos web esta fase consistiría en su publicación dentro de un servidor web, para que los alumnos puedan acceder a él mediante un navegador web y realizarlo de forma remota.

6. **Evaluación del curso**. En esta última etapa el profesor evalúa los resultados obtenidos por los alumnos en los diferentes exámenes y pruebas sobre los conceptos que componen el curso. Normalmente la evaluación se suele centrar en comprobar el correcto aprendizaje de los alumnos a lo largo del curso y no en evaluación del propio del curso. Esta fase también es común para todo tipo de cursos y se suele utilizar como ayuda para realizarla el análisis estadístico de los resultados.

Una vez evaluado el curso, se puede plantear una etapa adicional que provocaría el reinicio del ciclo de vida de éste: la modificación del curso tras su evaluación. En efecto, tras la construcción del curso, el profesor lo publica en un servidor web y, cuando dispone de una cantidad "suficiente" de información, comprobará los resultados obtenidos por los alumnos. A continuación, planteará modificaciones basadas en la evaluación de estos resultados y su experiencia. Dicha evaluación [Zaïane & Luo 2001] se ha realizado hasta la fecha de forma

manual, y off-line, mediante el uso de técnicas estadísticas que permiten realizar sumarios, detectar los datos más frecuentes, y descubrir por ejemplo cuáles son los errores más comunes cometidos por los alumnos. El problema que presenta este análisis es que, por lo general, la información que se genera en el sistema es casi imposible de manejar de un modo manual. Por esta razón, el análisis se suele limitar a resultados de la evaluación final, bastante más manejables.

Siguiendo esta idea de facilitar al profesor la labor de evaluación del curso, se puede avanzar un paso más con la utilización de las técnicas de minería de datos. La metodología que se va a proponer en el próximo capítulo [Romero et al. 2002c] va a hacer uso de estas técnicas de descubrimiento automático de conocimiento a partir de toda la información generada. Esta información, mucho más abundante, puede generar nuevo conocimiento que ayude al profesor a evaluar el curso, permitiéndole tomar decisiones sobre aspectos que hasta la fecha han sido imposibles de abordar.

2.1.5 Información disponible en los ASWE

Para realizar correctamente su trabajo de adaptación, los ASWE deben tener actualizado el modelo del alumno. Para ello, deben capturar una gran cantidad de información de interacción con el sistema, almacenándola en el historial del alumno, una zona que será accedida por el módulo de control del usuario.

El historial del alumno es una colección de datos de interés sobre la interacción del alumno, conteniendo entidades tales como el material didáctico utilizado (páginas visitadas), respuestas producidas en las distintas actividades, tiempos de acceso a cada página, etc.

Un aspecto muy importante dentro del historial del alumno es la información sobre la evaluación [Rios et al. 1998] del aprendizaje del alumno. Esta evaluación se suele realizar mediante pruebas de autoevaluación de tipo adaptativo o no adaptativo. La diferencia entre los test convencionales (ejercicios de opción múltiple, de respuesta breve, de tipo crucigrama, de tipo relaciona etc.) y los test adaptativos es que en los primeros los alumnos siguen una secuencia previamente establecida, mientras que en los segundos se adaptan a la situación particular de cada alumno, adaptando el nivel de dificultad de sus ítems o preguntas en función de las respuestas obtenidas.

Además del historial del alumno, el modelo de usuario almacena otras características del curso: conocimiento del dominio, tipo de alumno o categoría cognitiva, velocidad de aprendizaje, tipo de aprendizaje y preferencias sobre los elementos multimedia utilizados.

Aunque cada ASWE implementa una estructura propia para representar el modelo del alumno y almacena toda la información en su formato específico, al ser sistemas basados en web todos van a producir ficheros *log* dentro del servidor web donde se encuentran publicados. La información de estos ficheros se pueden ir recogiendo desde diferentes sitios [Srivastava et al. 2000]:

- **Desde el lado del servidor.** Utiliza los ficheros *log* de los servidores web que almacenan los comportamientos de navegación de los visitantes web. Estos ficheros logs pueden tener un formato típico o un formato extendido.

- **Desde la máquina cliente.** Se puede implementar utilizando un agente remoto (JavaScript o Applet Java) o modificando el navegador web para añadirle las capacidades de recogida de datos.

- **Desde servidores proxy intermedios.** Recoge los datos en un nivel intermedio entre los navegadores clientes y los servidores web.

- **Desde base de datos.** Base de datos de una organización que contiene los datos de la empresa o los datos web consolidados.

De esta forma mientras un usuario está interactuando con un sitio web, se está almacenando información sobre su comportamiento en estos ficheros históricos denominados ficheros *log* del servidor web (web server logs). Un fichero *log* [Cabral 2000] es un fichero texto en el que se almacena cada petición de una página que se le hace al servidor web. Para cada petición, el fichero log almacena la siguiente información:

- **Dirección IP.** Dirección IP del ordenador que hace la petición.

- **Identificación del usuario.** No se suele utilizar en la mayoría de los casos.

- **Fecha y hora.** Instante exacto de la petición.

- **Estado.** Si la petición se ha realizado de forma satisfactoria.

- **Tamaño.** Tamaño del fichero transferido.

- **URL de referencia.** URL de la página que contiene el enlace que ha generado la petición.

- **Navegador.** Nombre y versión del navegador web utilizado.

Se han realizado algunas propuestas para añadir información adicional a estos ficheros sobre las trazas detalladas de la interacción del alumno [Heift & Nicholson 2000]. Estas propuestas de mejora de los ficheros *log* se refieren a ampliar el tipo de información que pueden contener, de manera que se pueda almacenar todo el historial del alumno.

La información que contienen los ficheros *log* de los servidores web se puede utilizar para reconstruir las sesiones de navegación de los usuarios dentro de un sitio web. Actualmente existen gran cantidad de herramientas comerciales [Pal et al. 2002] para el análisis de ficheros *log*, aunque estas herramientas tienen sus capacidades de análisis limitadas produciendo sólo resultados tales como sumarios estadísticos y cuenta de páginas más frecuentemente visitadas. Las técnicas tradicionales que se han utilizados para analizar estos datos han sido técnicas estadísticas. En la actualidad se están aplicando con gran éxito las nuevas técnicas de minería de datos [Klösgen & Zytkow 2002] permitiendo complementar esta información y descubrir conocimiento oculto hasta entonces.

2.2 Minería de datos web

La necesidad de analizar la información generada en las distintas aplicaciones basadas en web, fundamentalmente de carácter comercial, ha propiciado la aparición de una disciplina cuyo objetivo es el descubrimiento y análisis de nuevo conocimiento útil a partir de dicha información. Dicha disciplina, denominada *minería web* o *minería de datos web* (Web Data Mining WDM) no es más que la aplicación de técnicas de data mining [Michalski et al. 1998], [Klösgen & Zytkow, 2002] sobre información procedente de este tipo de aplicaciones [Pal et al. 2002].

La minería de datos se encuentra enmarcada dentro del proceso de descubrimiento o extracción de conocimiento (Knowledge Discovery in Databases, KDD), entendiendo por tal a la extracción no trivial de información potencialmente útil, válida, novedosa y comprensible a partir de un gran volumen de datos [Klösgen & Zytkow 2002]. La minería de datos es uno de los pasos que comprenden dicho proceso de descubrimiento de conocimiento y que son:

- **Preprocesamiento.** Consiste en la recogida o extracción de los datos, limpieza de datos, discretización, selección de los atributos e integración de datos.

- **Minería de datos.** Consiste en la selección de los algoritmos de minería de datos a utilizar y la aplicación de dichos algoritmos sobre los datos.

- **Postprocesamiento**. Consiste en la interpretación, evaluación de los resultados obtenidos y la utilización del conocimiento descubierto.

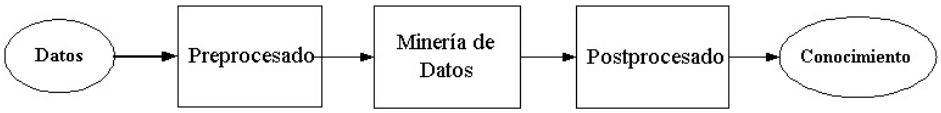

Figura 2.3: Proceso de descubrimiento de conocimiento en base de datos.

Dentro de la minería web se pueden distinguir tres categorías distintas [Srivastava et al. 2000]:

- **Minería de contenido web** (Web Content Mining, WCM). Trata de descubrir información útil a partir del contenido, datos, documentos y servicios web. El contenido web no es sólo texto sino que se extiende a audio, vídeo, representaciones, metadatos e hiperenlaces de datos.

- **Minería de estructura web** (Web Structure Mining, WSM). Corresponde a la minería de la estructura de los hiperenlaces dentro de la propia web, donde la estructura representa el grafo de enlaces de un sitio o sitios webs.

- **Minería de utilización web** (Web Usage Mining, WUM). Utiliza los datos generados por la interacción del usuario con el web. Entre estos datos se incluyen datos de acceso a servidores web (denominados ficheros *log* o históricos), ficheros *log* de servidores intermedios, ficheros *log* de navegadores, perfiles de usuarios, ficheros de registro, sesiones de usuarios o transacciones, consultas de usuarios, carpetas de direcciones web preferidas (bookmarks), pulsaciones de ratón, desplazamientos en línea (scroll) y cualquier otra información generada en la interacción del usuario con el sitio web.

Las principales áreas actuales de aplicación de la minería de utilización web son [Srivastava et al. 2000]:

- **Personalización.** Individualizando o personalizando un sitio web a cada usuario en particular.

- **Inteligencia comercial.** Informando de cómo los clientes utilizan un sitio web comercial.

- **Caracterización de la utilización.** Caracterizando tanto el uso, el contenido y la estructura de un sitio web.

- **Mejora de Sistemas.** Aumentando el rendimiento y otros atributos de calidad de servicios como seguridad, etc.

- **Modificación de sitios web.** Tanto del contenido como de la estructura de sitios web adaptativos.

2.2.1 Minería web en Educación

El uso de técnicas de minería de datos web está bastante extendida en sistemas de carácter comercial, habiéndose desarrollado sistemas para la comprensión del comportamiento de los clientes y poder aumentar las ventas [Spiliopoulou 2000]. Sin embargo, el uso de este tipo de técnicas en sistemas educativos está en su infancia aunque, en la actualidad, ya han aparecido sistemas en los que intentan explotarse las ventajas de la adquisición de conocimiento para mejorar los sistemas educativos en uno u otro sentido.

Antes de entrar a describir cuáles son los avances que se han producido hasta la fecha, debemos hacer una precisión sobre la diferencia de la aplicación de minería de datos en problemas del área de educación basada en web con respecto a la del comercio electrónico. Aunque las herramientas que se usan en ambos problemas de descubrimiento de conocimiento son muy parecidas, los objetivos de ambas aplicaciones son muy distintos. Así, mientras que en el comercio electrónico el objetivo es dirigir al visitante en la compra, en el aprendizaje a distancia el objetivo es dirigir al aprendiz en el mejor aprendizaje efectivo.

Actualmente los educadores utilizan entornos de aprendizaje y herramientas que permiten muy poca evaluación de las actividades de aprendizaje y discriminación de entre distintos comportamientos de aprendices. Las técnicas de minería de datos pueden potenciar estos entornos de aprendizaje, proporcionando al educador información que le permitirá evaluar el proceso de aprendizaje [Zaïne 2001].

Como ya se ha visto los ficheros *log* proporcionan información en bruto sobre la navegación y las actividades realizadas, por ejemplo, las páginas visitadas y los tiempos empleados en la visualización de cada una de ellas. Pero en el caso de los sistemas educativos se necesita disponer de unos ficheros *log* más especializados y completos que los tradicionales. Se necesita ampliar la información que ya anota el servidor web, incluyendo por ejemplo los resultados de las actividades educativas realizadas los tiempos empleados en la realización de dichas actividades o los niveles de conocimiento en los que el sistema va clasificando al alumno en cada momento. Una vez ampliados, ya sí es posible aplicar algoritmos de minería de

datos sobre estos ficheros y extraer patrones o reglas que el profesor pueda utilizar para mejorar el sistema de aprendizaje. Las fases para realizar el proceso de minería de utilización web aplicada a entornos educativos son [Zaïne & Luo 2001]:

- **Preprocesar los ficheros logs.** Consiste en la etapa de procesado de datos donde se realiza la limpieza de los ficheros *log* y la identificación de secuencias de actividades de cada usuario.

- **Aplicar técnicas de minería de datos.** Se utilizan las mismas técnicas que en minería de datos para buscar tendencias, patrones o reglas que puedan identificar los datos y sean interesantes para el profesor. Los métodos de minería de datos más utilizados son la extracción de reglas de asociación, clustering o agrupación, clasificación, modelado de dependencias y predicción.

- **Potenciar el entorno de aprendizaje basado en web.** Es la etapa más específica, donde se toman las decisiones necesarias para cambiar, mejorar y adaptar el contenido del curso, utilizando para ello la información que proporcionan los patrones y comportamientos descubiertos.

En la actualidad existen muy pocas investigaciones relacionadas con la aplicación de técnicas de minería web en educación. Sin embargo, esta situación va a cambiar en poco tiempo, debido a que están empezando a vislumbrarse los beneficios que ofrece la aplicación de técnicas de minería web a la educación a distancia [Zaïne & Luo 2001]. A continuación, se van a describir algunas de las principales investigaciones actuales relacionadas con la aplicación de técnicas de minería web en educación.

Uno de los pioneros en la aplicación de técnicas de minería de datos para la mejora de este tipo de sistemas es Omar Zaïne [Zaïne 2001], que ha aplicado minería de datos patrones útiles que puedan ayudar a los educadores a evaluar e interpretar las actividades de los cursos on-line, evaluando así el proceso de aprendizaje, comprobando las acciones típicas que realizan los estudiantes y midiendo la efectividad de la estructura del curso. Actualmente está desarrollando una herramienta para minería de datos para el propósito específico del aprendizaje on-line, basada en restricciones y visualización de patrones.

Una investigación muy relacionada con la anterior es la realizada por Pahl y Dave [Pahl & Dave 2002], que proponen la utilización de técnicas de minería de datos para analizar el proceso de aprendizaje del estudiante y evaluar la efectividad y la usabilidad de los cursos basados en web. El trabajo que reportan se basa en el análisis de las sesiones individuales de cada estudiante, definiendo como tal un periodo de tiempo de aprendizaje activo de un estudiante. Tras particionar los

ficheros *log* en sesiones individuales, formadas por las secuencias de peticiones de páginas del alumno dentro de un periodo de 20 minutos, aplican las siguientes técnicas:

- **Estadísticas de sesión**. Estadísticas básicas sobre sesiones como son la longitud de sesión media en tiempo o en número de peticiones.

- **Patrones de sesión**. Determinación del proceso de aprendizaje del estudiante extraído de la navegación y el comportamiento de petición. Utilizando técnicas de determinación de patrones de secuencias.

- **Series de tiempo de datos de sesión**. Análisis del desarrollo de estadísticas de sesión y patrones de sesión sobre un periodo de tiempo.

Otra investigación sobre la utilización de tecnología de minería de datos para analizar los ficheros logs en entornos de aprendizaje es la realizada por Wang [Wang 2002]. En ella el conocimiento extraído incluye clústeres de material asociativo (asociación entre páginas) y secuencias entre éstos (secuencias entre páginas). Este conocimiento permite al profesor estudiar la estructura dinámica de navegación e identificar algunos patrones de aprendizaje interesantes o inesperados. Esta información puede proporcionar líneas de decisión para organizar la estructura de enseñanza de una forma más eficiente. La idea es descubrir patrones de navegación de los estudiantes que puedan revelar la verdadera estructura requerida para ayudar a los alumnos. El modelo de navegación está formado por dos tipos de relaciones:

- **Relaciones de asociación**. Dos páginas están asociadas si son frecuentes en una sesión de aprendizaje y no hay una secuencia observable entre ellas.

- **Relaciones de secuencia**. Una página precede a otra si ambas son frecuentes en una sesión de aprendizaje y hay una secuencia observable entre ellas.

Una última investigación del campo de los tutores inteligentes es la realizada por Heift [Heift & Nicholson 2000] que propone un procedimiento iterativo para diseñar y testar un tutor inteligente utilizando los ficheros log de servidores web mejorados. Heift propone añadir información adicional en los ficheros log sobre trazas detalladas del programa analizadas en términos de eficiencia, exactitud e inteligencia del sistema. Para la realización de pruebas Heift ha desarrollado un tutor inteligente de Alemán, donde supone 3 tipos de aprendices: principiante, intermedio y experto. Actualmente está investigando sobre el tipo y el nivel de dificultad de los ejercicios y la correlación potencial entre ciertos tipos de errores y cómo algunos ejercicios particulares pueden mejorar las habilidades de lenguaje de

los alumnos. Para la realización de este tipo de análisis Heift propone la futura utilización de técnicas de minería de datos.

2.3 Descubrimiento de reglas

El uso de reglas es una de las formas más popular de representación del conocimiento debido, entre otras razones, a su sencillez, capacidad de expresión y escalabilidad. Dependiendo de la naturaleza del conocimiento que almacenan, se ha establecido una tipología informal para es tipo de estructuras. Así, se habla de reglas de decisión, asociación, clasificación, predicción, causalidad, optimización etc. En el ámbito de la extracción de conocimiento en bases de datos, las más estudiadas han sido las reglas de asociación, de clasificación y de predicción.

ReglaAsociación	::= “**SI”** (<antecedente>) **“ENTONCES”** <consecuente>
<antecedente>	::= <condición> | <condición> “**Y”** <antecedente>
<consecuente	::= <condición> | <condición> “**Y”** <consecuente>
<condición>	::= <atributo> <operador> <valor> | <atributo> <operador> <atributo>
<atributo>	::= *Cada uno de los posibles atributos del conjunto*
<valor>	::= Un valor del dominio del atributo correspondiente
<operador>	::= “=” | “<” | “>” | “≤”| “≥” | “≠”

Figura 2.4: Definición de las reglas de asociación en notación EBNF.

El propósito de las reglas de asociación es la buscar relaciones entre un tipos de atributos de la base de datos, que se colocan en el antecedente de la regla, y otros, que se colocan en el consecuente. Reglas de asociación típicas son las que informan sobre las preferencias de compra de un cliente de un establecimiento de comercio electrónico: **SI** (el usuario X adquiere el producto A) **ENTONCES** (adquiere el producto B). La Figura 1.4 ilustra el formato de estas reglas. Como puede comprobarse, el antecedente puede presentar una o más condiciones, en las cuáles se compara un atributo con un valor o dos atributos entre sí; el consecuente puede también presentar una o más condiciones.

Las reglas de clasificación tienen como objetivo almacenar conocimiento encaminado a la construcción de una clasificador preciso. En su antecedente contienen una serie de requisitos (en forma de condiciones) que debe cumplir un objeto determinado para que pueda considerarse que pertenece a la clase que identifica el consecuente de la regla. Desde el punto de vista sintáctico (ver Figura 1.5), la principal diferencia que presentan con las reglas de asociación es que presentan una sola condición en el consecuente, que además es de un identificador de clase.

ReglaClasificac	::= "**SI"** (<antecedente>) **"ENTONCES"** <consecuente>
<antecedente>	::= <condición> | <condición> "**Y"** <antecedente>
<consecuente	::= "**PERTENECE A**" *identificador de clase*
<condición>	::= <atributo> <operador> <valor> | <atributo> <operador> <atributo>
<atributo>	::= *Cada uno de los posibles atributos del conjunto*
<valor>	::= Un valor del dominio del atributo correspondiente
<operador>	::= "=" | "<" | ">" | "≤"| "≥" | "≠"

Figura 2.5: Definición de las reglas de clasificación en notación EBNF.

La Figura 2.6 muestra el formato genérico de las reglas de predicción en formato EBNF. Como puede observarse, comparte características sintácticas con las reglas de asociación y con las de clasificación. Al igual que sucede en el caso de las reglas de clasificación, el consecuente sólo presenta una condición. Sin embargo, el contenido de este consecuente no tiene que ser un identificador de categoría o clase, sino que puede tratarse de una condición de cualquier tipo (que es la que se pretende predecir, de ahí el nombre de reglas de predicción). El significado semántico de una regla de predicción es que si todas las condiciones especificadas por el antecedente de la regla son satisfechas por los atributos predictores de un ejemplo, la regla predice que el atributo objetivo (el que aparece en el consecuente) de esa instancia tendrá el valor especificado en el consecuente de la regla.

Regla	::= "**SI"** (<antecedente>) **"ENTONCES"** <consecuente>
<antecedente>	::= <condición> | <condición> "**Y"** <antecedente>
<consecuente	::= <condición>
<condición>	::= <atributo> <operador> <valor> | <atributo> <operador> <atributo>
<atributo>	::= *Cada uno de los posibles atributos del conjunto*
<valor>	::= Un valor del dominio del atributo correspondiente
<operador>	::= "=" | "<" | ">" | "≤"| "≥" | "≠"

Figura 2.6: Defición de las reglas de predicción en notación EBNF.

Además de las diferencias sintácticas existentes entre este tipo de reglas, es importante recordar la diferencia en su significado, la cual condiciona de algún modo la forma de llevar a cabo la minería de un tipo de reglas determinado. En este sentido, Alex Freitas [Freitas 2000] expone la naturaleza determinista que presenta la tarea de obtención de reglas de asociación, que establece relaciones entre atributos con unas garantías mínimas expresadas a través del soporte y confianza de la regla, frente a la naturaleza mal definida y no determinista que supone la tarea de extracción de reglas de clasificación que, en general, utiliza un conjunto de datos de entrenamiento para construir un clasificador que empleará sobre datos no pertenecientes a dicho conjunto.

2.3.1 Modelado de dependencias

La tarea de modelado de dependencias consiste en *la predicción de una serie de relaciones entre atributos, definidos o no por el usuario, de una base de datos*. Dicha tarea, que suele llevarse a cabo mediante la extracción de un conjunto de reglas de predicción, puede considerarse como una generalización de la tarea de extracción de reglas de clasificación en la que sólo existe un único objetivo a predecir. Sin embargo, la tarea del modelado de dependencias involucra una noción más amplia de dependencias que la clasificación y normalmente está asociado con un espacio de búsqueda mucho más amplio. Además, la tarea de clasificación es muy asimétrica con respecto a los atributos, ya que el atributo objetivo sólo puede ocurrir en el consecuente y los atributos predictores en el antecedente.

También hay que hacer notar la diferencia entre la tarea de modelado de dependencias y la de descubrimiento de reglas de asociación, cuyo objetivo es extraer todas las reglas que tienen unas medidas de soporte y confianza mayor o igual a un umbral especificado por el usuario. En efecto, aunque ambas tareas son simétricas con respecto a los atributos, se trata de tareas totalmente distintas. La tarea del descubrimiento de reglas de asociación es una tarea bien definida y determinista mientras que el modelado de dependencias, al igual que la clasificación es una tarea no determinista y mal definidas dado que, a partir de un conjunto de entrenamiento, van a intentar inducir una serie de relaciones entre atributos que, posteriormente, utilizarán sobre datos no pertenecientes a dicho conjunto.

A pesar de lo dicho, existe la posibilidad de realizar la tarea de modelado de dependencias utilizando algoritmos pertenecientes al contexto de la extracción de reglas de asociación y/o clasificación, sin más que realizar una conveniente adaptación de los mismos. En el Capítulo 5 se presentarán variantes de una serie de algoritmos clásicos que se han adaptado para este fin, y que se emplean en la comparación de los algoritmos evolutivos desarrollados en esta Memoria.

2.3.2 Algoritmos Clásicos para el descubrimiento de reglas

La tarea del descubrimiento de reglas ha sido abordada desde multitud de paradigmas: construcción de árboles de decisión, aprendizaje inductivo, aprendizaje basado en instancias y, más recientemente redes neuronales y algoritmos evolutivos [Witten & Frank 2000]. El tipo de búsqueda que realizan cada uno de estos algoritmos va a determinar dónde se encuentran localizados dentro del panorama de la minería de reglas [Dhar et al. 2000] y desde el punto de vista de la minuciosidad de la búsqueda (ver Figura 2.7).

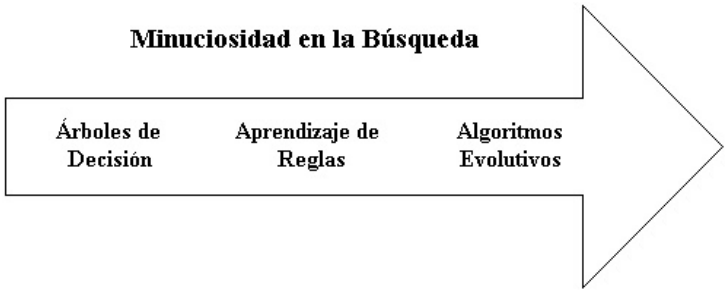

Figura 2.7: Minuciosidad de búsqueda de los algoritmos de descubrimiento de reglas.

En la Figura 2.7 se muestra el espectro de las técnicas de búsqueda en términos de la minuciosidad de la búsqueda que realizan. Por un lado del espectro están los algoritmos de inducción de reglas mediante árboles de decisión que utilizan heurísticas altamente voraces y realizan una búsqueda irrevocable. Los algoritmos de inducción de árboles son actualmente las técnicas más utilizadas en minería de datos. Son muy rápidos y sorprendentemente efectivos para encontrar clasificadores precisos, además de clasificar completamente los datos. Pero los algoritmos de inducción de reglas generalmente pierden parte de exactitud por su velocidad. La mayoría utilizan técnicas de particionado recursivo que van partiendo el conjunto de datos utilizando heurísticas voraces que pueden pasar por alto relaciones multivariables que no aparecen si se tratan las variables individuales aisladamente. Justo al lado del espectro (ver Figura 2.7) se encuentran los algoritmos de aprendizaje de reglas convencionales donde se consideran una amplia variedad de alternativas cuya característica común es la de ser más minuciosos que los anteriores. Y en el otro lado del espectro (ver Figura 2.7) se encuentran los algoritmos evolutivos que son capaces de conducir muchas búsquedas minuciosas y realizar un retroceso implícito en la búsqueda del espacio de reglas que va a permitir encontrar interacciones complejas que los otros tipos de algoritmos no son capaces de encontrar.

A lo largo de esta Sección se presentarán los algoritmos más populares dentro de la categoría de los que podríamos denominar “clásicos”. Dichos algoritmos son el algoritmo ID3 (un algoritmo de construcción de árboles de decisión), el algoritmo A priori y el algoritmo PRISM, procedente del ámbito del aprendizaje inductivo. En la siguiente Sección presentaremos la resolución de la tarea planteada mediante Algoritmos Evolutivos, un enfoque mucho más actual y que, como se verá posteriormente, presenta claras ventajas sobre la mayoría de los enfoques tradicionales.

2.3.2.1 Algoritmos de construcción de árboles de decisión

Una de las formas más simples de representar conocimiento es mediante árboles de decisión. Dicha estructura de representación del conocimiento está formada por una serie de nodos, donde:

- Cada nodo interno es etiquetado con el nombre de uno de los atributos predictores.
- Las ramas que salen de un nodo interno son etiquetadas con los valores del atributo de ese nodo.
- Cada nodo terminal, o nodo hoja es etiquetado con el valor del atributo objetivo.

Un ejemplo de árbol de decisión se muestra en la Figura 2.8 donde el atributo objetivo es el atributo nivel y los atributos predictores son el atributo tiempo y el atributo acierto.

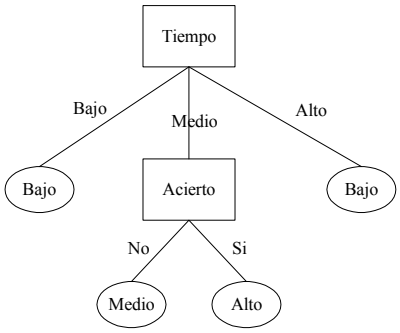

Figura 2.8: Ejemplo de árbol de decisión.

Una característica muy interesante de los árboles de decisión es que, a partir de ellos, es muy fácil derivar a partir de él un conjunto de reglas de producción del tipo Si-Entonces (If-Then) completamente equivalente al árbol original. El algoritmo que nos permite realizar este cambio de modelo de representación es casi trivial, y convierte cada camino que va desde el nodo raíz hasta una hoja en una regla de la siguiente forma:

- Los nodos internos y sus correspondientes ramas de salida son convertidos en las condiciones del antecedente de la regla.
- Los nodos hojas se convierten en el consecuente de la regla.

Al convertir el árbol de decisión en una colección de reglas se obtiene una regla por hoja del árbol. Dichas reglas serán, por tanto, mutuamente excluyentes. Por ejemplo, del árbol de decisión de la Figura 5.1 se obtienen las siguientes reglas:

Si Tiempo = Bajo Entonces Nivel = Bajo
Si Tiempo = Medio Y Acierto = No Entonces Nivel = Medio
Si Tiempo = Medio Y Acierto = Si Entonces Nivel = Alto
Si Tiempo = Alto Entonces Nivel = Bajo

El algoritmo de construcción de árboles de decisión más popular es el algoritmo ID3 [Quinlan 1987]. Este algoritmo es del tipo "divide y vencerás" (ver Algoritmo 2.1), y construye el árbol de decisión desde la raíz hacia las hojas, incrementando en cada paso la complejidad del árbol.

Inicializar T con el conjunto de todas las instancias
Si (todas las instancias en el conjunto T satisfacen el criterio de parada) **Entonces**
Crear un nodo hoja etiquetado con el nombre de la clase y parar.
Sino
Seleccionar un atributo para utilizar como atributo de particionado.
Crear un nodo etiquetado con el nombre del atributo y crear una rama por cada valor del atributo.
Particionar T en subconjuntos tales que contengan las instancias que cumplan los valores del atributo.
Aplicar este algoritmo recursivamente para cada subconjunto.
Fin si
Recorrer el árbol y formar las reglas.

Algoritmo 2.1: Algoritmo ID3.

Como se puede comprobarse, el algoritmo ID3 es un algoritmo recursivo, donde hace falta especificar varias cosas:

- Como determinar que atributo se utiliza para particionar. Para ello se utiliza una medida de la ganancia de información denominada entropía y que se obtiene como:
  $entropia(p_1, p_2, ..., p_n) = -p_1 \log p_1 - p_2 \log p_2 ... - p_n \log p_n$.
  Utilizando esta medida se puede seleccionar el atributo con el que se realiza el mejor particionado, aquel que tiene un mayor valor de entropía.

- El criterio de parada, que es cuando todos los ejemplos tienen la misma clasificación.

Las principales ventajas del algoritmo ID3 son:

- El algoritmo descubre conocimiento que tiende a ser intuitivamente simple y comprensible para el usuario, siempre que el árbol de decisión no sea

grande. Para conseguir que los árboles de decisión no sean grandes algunos autores han propuesto algoritmos de poda [Catlett 1991].

- Un árbol de decisión también proporciona información sobre la relevancia relativa de los atributos para propósitos de predicción. Mientras más cerca esta el atributo de la raíz del árbol más relevante es. Esta información no la suelen proporcionar los conjuntos de reglas.

- Es un algoritmo rápido debido a que utiliza una aproximación divide y vencerás. El algoritmo expande el árbol con nodos que contienen menos y menos instancias de los datos.

aunque sus principales desventajas son:

- El problema de la fragmentación [Friedman et al. 1996]. Debido a que cada vez el número de instancias es menos significa también que hay un menor soporte estadístico para la selección de un atributo.

- Realiza una búsqueda voraz o greedy con los atributos a poner en el árbol. Construye el árbol paso a paso, añadiendo el mejor atributo cada vez. Sin embargo, la secuencia de los mejores decisiones locales posibles no garantiza la mejor decisión global posible.

- Debido a la estructura del árbol de decisión, las ramas se deben asociar con todos los posibles valores de atributo particionador. Y puede ocurrir que sólo uno de esos valores sea relevante, dando lugar a pares atributo-valor irrelevantes que pueden incrementar considerablemente el tamaño del árbol haciéndolo innecesariamente complejo.

2.3.2.2 Algoritmos de minería de asociación

La tarea de minería de reglas asociación (association rule mining) fue introducida por Agrawal [Agrawal et al. 1993] que la define como la tarea de encontrar relaciones entre cualquier variable. Dicha tarea se ha aplicado tradicionalmente a bases de datos transaccionales, donde una transacción T está formada por un conjunto de artículos o ítems. A un conjunto de ítems se le suele denominar itemset, en general, o itemset de grado k, cuando se especifica el número k de ítems que incluye. Una regla de asociación es una implicación $X \rightarrow Y$ en la cual X e Y son itemsets de intersección vacía, es decir sin ítems en común. El significado de la regla de asociación es que las transacciones o tuplas que contienen a X también tienden a contener Y. Las reglas de asociación deben de superar además un valor para dos medidas denominadas confianza y soporte de la

regla. La confianza de la regla de asociación es la proporción de las transacciones que, conteniendo a *X*, también incluyen a *Y,* mientras que el soporte es la fracción de transacciones en la base de datos que contienen tanto a *X* como a *Y*. La extracción de reglas de asociación también puede realizarse en bases de datos relacionales como es nuestro caso, donde un item es un par atributo-valor.

El proceso habitualmente seguido por los algoritmos de descubrimiento de reglas de asociación consiste en extraer todas las reglas que tengan un soporte y una confianza por encima de unos umbrales establecidos por el usuario. Este problema se suele descomponer en los dos siguientes subproblemas utilizando la estrategia divide y vencerás.

1. Encontrar todos los ítems frecuentes, que son aquellos cuyo soporte es mayor que el umbral de soporte mínimo establecido.

2. Generar las reglas de asociación que se derivan de los ítems frecuentes.

El descubrimiento de itemsets frecuentes es la parte computacionalmente más costosa, mientras que la generación de reglas de asociación a partir de los itemsets frecuentes es casi inmediata. Por este motivo la mayor parte de los algoritmos de extracción de reglas de asociación se han centrado en la enumeración eficiente de todos lo itemsets.

El algoritmo más popular de descubrimiento de reglas de asociación es el algoritmo Apriori [Agrawal et al. 1993], que genera el conjunto de elementos que superan un soporte mínimo que posteriormente se transforman en reglas que superan una confianza mínima (ver Algoritmo 2.2). El algoritmo realiza varias pasadas sobre los datos para encontrar los conjuntos de elementos frecuentes. En la primera pasada simplemente cuenta la ocurrencia de los elementos de tamaño 1. En la pasada k, el algoritmo encuentra todos los elementos de tamaño k a partir de los elementos de tamaño k-1. En cada pasada, primero construye un conjunto de elementos candidatos potencialmente frecuentes y luego identifica cuales los son realmente.

L_1 = Conjuntos de Elementos Frecuentes de tamaño *1*.
Para $k = 2$ **mientras** $L_{k-1} \neq 0$ **incrementando** k **hacer**
  C_k = Nuevos candidatos de tamaño k generados a partir de L_{k-1}
  Para todas las transacciones **hacer**
    Contabilizar el soporte de todos los candidatos C_k
  Fin Para
  L_k = Añadir todos los candidatos en C_k que superen el soporte mínimo.
Fin Para
Unir todos los L_k para formar todos los conjuntos de elementos frecuentes.
Generar las reglas a partir de los conjuntos frecuentes que superen la confianza mínima.

Algoritmo 2.2: Algoritmo Apriori.

Para la generación de las reglas a partir de los conjuntos frecuentes se utiliza una regla que dice que todo subconjunto de un conjunto de elementos frecuente también es un subconjunto de elementos frecuente. Este algoritmo simple para generar las reglas se muestra en Algoritmo 2.3.

Para cada conjunto frecuente X y subconjunto B de X **hacer**
  Si ($A = X - B$ **Y** *Soporte(X) / Soporte(A) >= ConfianzaMínima*) **Entonces**
    Generar como regla válida $A \rightarrow B$
Fin Para

Algoritmo 2.3: Algoritmo para generación de reglas a partir de los conjuntos frecuentes.

Este algoritmo ha sido muy investigado y, en la actualidad, existen multitud de propuestas distintas de algoritmos que intentan mejorarlo [Hipp et al. 2000], obviando sus principales desventajas:

- El algoritmo Apriori realiza una pasada sobre el conjunto de datos para cada tamaño distinto del conjunto de elementos. Cuando la base de datos es muy grande este proceso se ralentiza notablemente, por lo que se debería de reducir el número de pasadas.

- La cantidad de computo necesario para generar las reglas de asociación depende críticamente del soporte mínimo especificado, de forma que mientras menor sea el soporte mayor será el número de conjuntos de elementos y mayor el tiempo de ejecución.

2.3.2.3 Algoritmos de aprendizaje inductivo

En este tipo de algoritmos, las reglas inducidas (descubiertas) cubren regiones de espacios en los datos que se pueden solapar, de forma que una instancia de los

datos puede estar cubierta por más de una regla. Existen una gran cantidad de algoritmos que inducen conjuntos de reglas de tipo Si-Entonces directamente a partir de un conjunto de datos. Algunos de estos algoritmos realizan una búsqueda dirigida en el espacio de posibles hipótesis. La búsqueda dirigida, una generalización del algoritmo "primero el mejor", se caracteriza por mantener una lista limitada de soluciones parciales durante el proceso de exploración del espacio de búsqueda. Algunos ejemplos de tipo de algoritmos que utilizan variantes de la búsqueda dirigida en el proceso de construcción de reglas son: la metodología STAR, las listas de decisión y los algoritmos evolutivos.

Un ejemplo típico de algoritmo de inducción de reglas es el algoritmo Prism [Cendrowska 1987] que es un algoritmo de tipo cobertura que en cada paso identifica una regla que cubre algunas de las instancias. Debido a su naturaleza la aproximación de cobertura se dirige hacia un conjunto de reglas en lugar de hacia un árbol de decisión. El algoritmo Prism (ver Algoritmo 2.4) a diferencia de los algoritmos de construcción de árboles de decisión que utilizan una estrategia "divide y vencerás", utiliza una estrategia "separa y vencerás". Dicha estrategia consiste en buscar una solución parcial al problema (una sola regla) y, una vez encontrada, reducir el problema eliminando todos los ejemplos cubiertos por la solución encontrada.

```
Para cada clase C
  Inicializar E con el conjunto de Instancias
Mientras E contenga instancias de la clase C Hacer
   Crear una regla R con antecedente vacío y con la clase C en su consecuente.
   Hasta que (Confianza(R) < 100%  o no haya más atributos que utilizar) hacer
      Para cada atributo A no utilizado en R y cada valor v.
         Considerar añadir la condición A=v en el antecedente de la regla.
          Seleccionar A y v para maximizar la exactitud
      Fin Para
     Añadir A=v a la regla.
    Fin Hasta
   Eliminar de E las instancias cubiertas por R.
Fin Mientras
Fin Para
```

Algoritmo 2.4: Algoritmo Prism.

Como se puede apreciar (Algoritmo 2.4) en el algoritmo Prism también es un algoritmo de tipo voraz, ya que va construyendo la regla paso a paso añadiendo la mejor condición cada vez. Sin embargo, el criterio que emplean para elegir el mejor atributo es distinto en este caso: el algoritmo ID3 escogía el atributo que maximiza la ganancia de información, mientras que estos algoritmos eligen el par valor-atributo que maximiza la probabilidad de la clasificación deseada. Además se

puede apreciar que el algoritmo Prism genera sólo reglas perfectas o correctas, de forma que cualquier regla con exactitud menor del 100% la considera incorrecta.

La principal ventajas del algoritmo Prism es que opera de una manera muy similar a la de un algoritmo divide y vencerás de tipo arriba-abajo o top-down, aunque sus principales desventajas son:

- Al ser un algoritmo de tipo voráz no tiene en cuenta la interacción de los atributos, ya que los va añadiendo uno a uno.

- Debido a la forma en que se construyen las reglas, parece que se tienen que interpretar en orden, es decir como una lista de decisión. Esto es debido a que las instancias cubiertas por una regla son eliminadas del conjunto de instancias, de forma que las reglas subsecuentes actúan sobre las instancias no cubiertas. Sin embargo, aunque las reglas parecen que se comprueban en orden realmente no lo hacen. Por lo que no importa el orden.

2.3.3 Algoritmos Evolutivos para descubrimiento de reglas

Los Algoritmos Evolutivos (Evolutionary Algorithms, EA) son algoritmos estocásticos de búsqueda basados en las ideas de la evolución darwiniana. Aunque existen distintos paradigmas de Computación Evolutiva (Evolutionary Computation, EC), todos tienen algunas características comunes [Bäck 2000]:

- Trabajan con una población de individuos (soluciones candidatas) al mismo tiempo, en lugar de con una única solución candidata cada vez.

- Utilizan un método de selección guiado por una medida de la calidad de cada solución candidata. De esta forma, mientras mejor capacitado está un individuo para resolver el problema planteado, más cantidad de veces es seleccionado y más partes de su material genético (partes de su solución candidata) se pasa a las generaciones posteriores de individuos.

- Generan nuevos individuos mediante un mecanismo de herencia de los individuos existentes, de forma que los individuos descendientes son generados al aplicar unos operadores estocásticos sobre los individuos existentes de al generación actual. Los dos tipos de operadores más utilizados son el cruce y la mutación. El cruce intercambia material genético entre dos o más individuos, mientras que la mutación cambia el valor de una pequeña parte del material genético a un nuevo valor aleatorio.

Las principales ventajas de la utilización de los algoritmos evolutivos para el descubrimiento de reglas con respecto a los algoritmos clásicos son, según [Dhar et al. 2000]:

- Los algoritmos de inducción de reglas utilizan un operador determinista, mientras que los algoritmos evolutivos utilizan operadores estocásticos.
- Los algoritmos de inducción de reglas construyen y evalúan una regla candidata de una forma incremental, mientras que los algoritmos evolutivos normalmente evalúan una regla candidata o conjunto de reglas como un conjunto.
- A diferencia de los algoritmos de inducción de reglas, los algoritmos evolutivos trabajan con una población de reglas candidatas o conjuntos de reglas en lugar de con una única regla o conjunto de reglas a la vez.
- Los algoritmos evolutivos hacen frente mejor al problema de la interacción de los atributos que los algoritmos de inducción de reglas que suelen estar basados en una búsqueda de tipo voraz o greedy, donde las reglas se construyen considerando un atributo cada vez y seleccionando el mejor atributo a incluir en la regla.
- Son un método de búsqueda global en el espacio de reglas.
- Han demostrado que son apropiados para el aprendizaje de patrones con forma de reglas.
- Realizan retroceso implícito en la búsqueda del espacio de reglas que permite encontrar interacciones complejas que otras búsquedas no encuentran.
- La búsqueda es conducida por una comparación de entre un conjunto de reglas candidatas que compiten entre sí. Las reglas compiten una frente a otra utilizando su función de ajuste.
- Tienen la habilidad de permitir la utilización de funciones de ajuste arbitrarias en la búsqueda.

aunque, según este autor, también presentan los siguientes inconvenientes con respecto a otros métodos de descubrimiento de reglas

- El tiempo de ejecución que requieren en grandes bases de datos tiende a ser muy superior a otros algoritmos de inducción de reglas de tipo voraz. Aunque se puede utilizar paralelismo y se puede optimizar la representación y evaluación para aumentar su velocidad de ejecución.
- Su aleatoriedad en la creación de la población inicial y en las exploraciones subsecuentes. Aunque se puede utilizar otras inicializaciones no aleatorias.
- Pueden ser miopes después de encontrar una buena solución. Suelen tener la tendencia de enfocarse muy cerca de una solución simple de alta calidad. Aunque se pueden utilizar métodos de reenfoque para solucionar este problema.

Por último, una de las principales contribuciones de los algoritmos evolutivos en el descubrimiento de reglas es que tienen en cuenta la interacción de los atributos, como consecuencia de su búsqueda global. En contra de la búsqueda voraz y local realizada por los muy utilizados algoritmos de inducción de reglas y árboles de decisión. La mayoría de los algoritmos de inducción de reglas generan una regla seleccionando una condición o atributo de la regla cada vez, por lo que no tienen en cuenta la posible interacción entre atributos. Por el contrario los algoritmos evolutivos normalmente evalúan una regla como un conjunto utilizando la función de ajuste en lugar de evaluar el impacto de añadir o eliminar una condición o atributo de la regla. Esto no quiere decir que los algoritmos evolutivos sean inherentemente superiores a los algoritmos de inducción de reglas, ya que no existe un algoritmo de descubrimiento superior en todos los casos.

Los paradigmas de Computación Evolutiva que se han aplicado para resolver el problema del descubrimiento de reglas son los Algoritmos Genéticos (Genetic Algorithms, GA) y la Programación Genética (Genetic Programming, GP). En lo que sigue, presentaremos las características más sobresalientes de cada paradigma cuando se aplica a la tarea que nos ocupa, examinando también las aplicaciones más sobresalientes descritas en la bibliografía hasta la fecha.

2.3.3.1 Algoritmos Genéticos

Existen dos posibles enfoques a para la obtención de reglas mediante algoritmos genéticos:

- **Aproximación Michigan.** La población está formada por reglas individuales de longitud constante que compiten entre sí, por lo que la solución final consiste la población completa. Este enfoque es especialmente apto para la obtención de reglas de clasificación.

- **Aproximación Pittsburgh.** En este caso, las soluciones son representadas mediante individuos que compiten entre ellos durante el proceso evolutivo. Los individuos son de longitud variable y están formados por un conjunto de reglas de longitud fija.

La elección de una de estas dos aproximaciones depende mucho del tipo de regla que deseamos descubrir. La aproximación de Pittsburg tiene en cuenta la interacción entre reglas cuando calcula la función de ajuste de un individuo. Sin embargo los individuos son más largos y el cálculo del ajuste implica un mayor coste computacional. Además esta aproximación suele requerir operadores genéticos adaptados a estos individuos. Por el contrario, en la aproximación de Michigan los individuos son más simples y sintácticamente más cortos,

reduciéndose el tiempo de computación en el cálculo de la función de aptitud y simplificándose el diseño de los operadores genéticos. Ahora bien, la función ajuste sólo evalúa la calidad de cada regla independientemente, sin considerar cuenta las interacciones entre atributos. A continuación se va a describir de forma detalla la codificación de reglas utilizando la aproximación tipo Michigan.

2.3.3.1.1 Codificación de los individuos

En principio las condiciones de una regla se pueden codificar utilizando una codificación binaria o de alto nivel.

- **Codificación binaria.** A cada atributo predictor se le asigna cierto número de bits, que depende del tipo de dato del atributo. En el caso de atributos categóricos (nominales) una posibilidad podría ser codificar cada valor de cada atributo con un único bit, donde si el bit se pone a 1 el atributo se incluye en la regla y si es 0 no se incluye. En el caso de atributos continuos (valores reales) la situación es más compleja y son posibles muchos esquemas alternativos de codificación, aunque la solución más simple consiste en discretizarlos y codificando los intervalos resultantes mediante atributos categóricos.

- **Codificación de alto nivel.** Codifica los valores de los atributos directamente en el genoma (genotipo o genoma). En el caso de atributos categóricos la parte del genoma correspondiente contiene uno de los valores de los atributos. En el caso de atributos continuos existen dos formas principalmente de codificación. La primera consiste en codificar un valor simple del atributo o valor umbral dentro del genotipo junto a un operador de comparación ($\leq$,$\geq$,<,>). La segunda forma consiste en codificar tanto un umbral máximo y un umbral mínimo dentro del individuo junto con dos operadores de comparación.

- **Codificación híbrida.** Utiliza tanto codificación binaria como de alto nivel [Kwedlo & Kretowski 1999] de forma conjunta para representar atributos categóricos y continuos.

Establecida la forma de codificar cada una de las condiciones, hay que abordar el problema de que, en principio, no sabemos el número de condiciones que forman el antecedente. Para tratar este problema existen dos aproximaciones:

- **Utilizar un genotipo de longitud fija, estableciendo una correspondencia entre éste y el antecedente de la regla (de longitud variable).** Para ello se asume una codificación posicional de los atributos en el

genotipo. Esta representación tiene la ventaja de simplificar la acción de los operadores genéticos como el cruce.

- **Utilizar un genotipo de longitud variable que es equivalente al antecedente de la regla de longitud variable.** Tiene el inconveniente de que hay que tener cuidado de que los operadores genéticos como el cruce no produzcan individuos inválidos (por ejemplo un individuo que contengan dos condiciones contrarias).

Para codificar el consecuente de la regla existen también diferentes aproximaciones que asocian el consecuente con un antecedente dado:

- **Evolucionando el consecuente.** Consiste en codificar el consecuente de la regla dentro del genotipo del individuo. De forma que tanto el antecedente como el consecuente evolucionan juntos.

- **Eligiendo el mejor consecuente.** A partir de un consecuente ya codificado en el genotipo se selecciona el mejor consecuente para forma la regla cada vez que se crea una nueva regla o se modifica por un operador genético.

- **Utilizando el mismo consecuente para toda la población.** Simplemente asocia a todos los individuos de la población el mismo consecuente, por lo que no se necesita codificar el consecuente dentro del genotipo de los individuos. Pero necesitaría ejecutar varias veces el GA, una para cada valor del consecuente que se desee obtener.

- **Utilizando el mismo consecuente para cada subpoblación.** Consiste en dividir la población en subpoblaciones y asociar a cada una un consecuente fijo.

2.3.3.1.2 Inicialización de la población

En la mayoría de los algoritmos genéticos la población inicial se genera de forma aleatoria. Sin embargo, en el contexto de descubrimiento de reglas es posible que las reglas generadas de forma aleatoria no cubran instancias de los datos, con lo que tendrían un valor de ajuste muy bajo. Este problema se incrementa con el tamaño de las reglas ya que, a mayor tamaño de regla, mayor probabilidad existe de que se contengan condiciones de bajo cubrimiento. Algunas soluciones a este problema son:

- **Inicializar aleatoriamente la población sólo con reglas generales.** Estas reglas son las que tienen un número pequeño de condiciones (generalmente 2 ó 3).

- **Influenciar la generación de reglas de menor tamaño.** Es una variante algo más flexible de la opción anterior. Se permite generar reglas largas (con más condiciones), pero se prefieren reglas cortas (con menor número de condiciones).

- **Inicializar la población sólo con reglas que garantizan que cubren al menos una instancia.** Esto se puede hacer mediante semillas, por ejemplo eligiendo aleatoriamente instancias de los datos y transformádolas en reglas especializadas. La elección de instancias semillas se suele hacer de forma aleatoria, incrementando el no determinismo del algoritmo genético.

2.3.3.1.3 Operadores genéticos

Además de los operadores clásicos de cruce y mutación, existen varios operadores específicos de Generalización y Especialización de los GA que se pueden utilizar cuando los individuos representan reglas:

- **Generalización y/o especialización mediante Cruce y Mutación.** Los operadores de cruce o mutación realizan una acción de generalización o especialización de una regla dada, dependiendo de si la regla actual es muy específica (cubre pocas instancias) o muy general (cubre muchas instancias). La idea de este tipo especial de cruce es generalizar o especializar una regla dada, dependiendo de si sobre ajusta o apenas ajusta a los datos, respectivamente. Una forma simple de implementarlo es utilizando los operadores lógicos AND y OR, aunque existen otras formas de hacerlo [Shaffer 1993]. La especialización o generalización de una regla se puede hacer de otras formas además de utilizando el operador cruce, como ya hemos visto. Por ejemplo se puede generalizar utilizando un tipo de mutación que genere valores aleatorios muy pequeños con los que se modifican los valores de las condiciones. Otra forma de generalizar es simplemente borrar una de las condiciones, este es el denominado operador condición de caída. Al contrario, para especializar se puede también utilizar el operador de mutación que añade pequeños valores aleatorios. Otra forma de especializar es añadir condiciones.

- **Inserción/Eliminación de condiciones.** Otra forma de generalizar o especializar una regla consiste en eliminar o insertar una condición respectivamente.

2.3.3.1.4 Funciones de Ajuste

La métrica de evaluación de GA se denomina función de ajuste y la manera en la que dicha función de ajuste afecta a la selección de los individuos para reproducirse se denomina como algoritmo de selección. La función de ajuste depende del problema específico que se desea resolver.

El conocimiento descubierto por un algoritmo de minería de datos debe satisfacer tres criterios principales [Freitas 2002]: exactitud, comprensibilidad e interés. Por lo tanto la función de ajuste tiene que incorporar estos criterios.

- **Exactitud de la regla.** Mide la exactitud o precisión de las reglas. Para el caso de reglas de clasificación se utiliza la matriz de confusión. Para el caso de reglas de predicción se utiliza una matriz similar a la de confusión que se denomina matriz de contingencia. En la bibliografía se ha descrito una gran cantidad de métricas para evaluar la exactitud de la regla a partir de esta matriz de contingencia [Lavrac et al. 1999].

- **Comprensibilidad de la regla.** La medida más utilizada de comprensibilidad es una medida sintáctica de la longitud de la regla. En general mientras menor es el número de reglas y el número de condiciones en una regla, más comprensible es ésta [Bojarczuk et al. 2000].

- **Interés de la regla.** Mide el interés de las reglas. Es la medida más difícil de calcular y existen dos aproximaciones para medir el interés de una regla: aproximación objetiva y aproximación subjetiva. La aproximación objetiva o dirigida por usuario se basa principalmente en tener en cuenta conocimiento o expectativas anteriores del usuario, por lo que es dependiente del dominio. En cambio la aproximación subjetiva o dirigida por datos sólo utiliza los propios datos, por lo que es independiente del dominio. Normalmente se pueden combinar ambos tipos de medidas en lugar de utilizarlos de forma mudamente exclusiva.

Una forma simple de tener en cuenta estos tres criterios en la función de ajuste consiste en utilizar una función de ajuste con pesos [Noda et. al 1999]. Sin embargo, estos tres criterios pueden estar en conflicto y ser no conmensurables, en el sentido de que evalúan aspectos muy diferentes de la solución candidata. Este problema sugiere el uso de una aproximación multiobjetivo para el descubrimiento de reglas. Aunque la bibliografía de Algoritmos Genéticos Multiobjetivo (MOEA) es amplia [Fonseca & Fleming 2000] [Deb 2001], la utilización de MOEA en el descubrimiento de reglas parece relativamente inexplorada.

2.3.3.1.5 Métodos de selección

Después de determinar la calidad de los individuos mediante el cálculo de la función de ajuste, hay que determinar sobre qué individuos se van a aplicar los operadores de cruce y mutación y cuáles se van a mantener en la población o se van a reemplazar. La selección es una consecuencia de la competición de los individuos en la población. Existen diferentes métodos de selección como la selección proporcional, la basada en ordenación o rangos, selección por torneo, etc.

La selección de reglas en la aproximación Pittsburg no es un gran problema, ya que se puede utilizar un método de selección convencional para seleccionar los individuos (conjuntos completos de reglas) a reproducir. La función de ajuste puede evaluar el conjunto de reglas como un conjunto, teniendo en cuenta las interacciones entre reglas. Pero esta situación cambia para la aproximación Michigan, donde cada individuo representa una regla y estamos interesados en descubrir un conjunto de reglas. Un método que resuelve este problema es utilizar un procedimiento de selección de reglas basados en la cobertura de los datos [Greene & Smith 1993]. La idea consiste en considerar los datos como un recurso utilizado por los individuos del algoritmo genético, de forma que cuando un recurso es consumido por un individuo ya no está disponible para otros individuos. Los individuos compiten para consumir recursos, de forma que diferentes individuos ocupan diferentes nichos que consisten en que cobren diferentes subconjuntos de instancias de datos. Otro método de selección de reglas para la aproximación Michigan es el denominado sufragio universal [Giordana et al. 1994] donde la idea es que los individuos sean elegidos por las instancias de los datos, de forma que cada instancia vota a una regla que la cubre de forma estocástica (probabilística) basada en la función de ajuste. Estos métodos implementan efectivamente una forma de nichos fomentando la evolución de diferentes reglas, donde cada una cubre un espacio de los datos distinto, es decir, que ayudan a evitar la convergencia de la población hacia una única regla.

2.3.3.1.6 Sistemas para Descubrimiento de Reglas basados en GA

El primer sistema que emplea algoritmos genéticos para el aprendizaje de reglas fue REGAL (RElational Genetic Algorithm based Learned) [Giordana & Saitta 1993]. Dicho sistema utiliza un modelo híbrido Michigan-Pittsburg en el que la población es un conjunto redundante de individuos de longitud fija, cada uno de los cuales es una descripción parcial de un concepto que evoluciona separadamente. El sistema trabaja directamente con cadenas binarias de longitud fija, siendo también es capaz de codificar fórmulas complejas en el modelo inicial, como por ejemplo con cuantificadores o negación. El sistema presentaba una serie de limitaciones desde el punto de vista del descubrimiento de conocimiento, siendo la más

importante que el usuario debía proporcionar la estructura de las reglas a aprender, lo que supone que el usuario tiene una idea del conocimiento que espera descubrir. Permite la formación de varias subpoblaciones para la búsqueda simultánea de varias reglas, de forma que el conjunto completo de reglas representan la población. Utiliza una aproximación basada en cobertura e introduce un nuevo operador de selección de sufragio universal para impulsar la cooperación entre miembros de la población.

El sistema GIL (Genetic-based Inductive Learning) [Janikow 1993] utiliza un algoritmo evolutivo en el que se diferencian las reglas como entidades independientes y el conjunto de reglas como entidad global. La idea es añadir a la evaluación global de los individuos una medida que evalúe las reglas dependiendo del conjunto al que pertenece. Introduce los conceptos de completitud[1] y consistencia de una regla y de un conjunto de reglas como medidas de la calidad de un individuo en base a un conjunto de datos. Los operadores genéticos utilizados se aplican a diferentes niveles, desde a nivel de conjunto de reglas, como de reglas, hasta nivel de condiciones.

El sistema GABIL (Genetic Algorithm Batch-Incremental) [De Jong et al. 1993] aborda el problema del aprendizaje de conceptos mediante un algoritmo en el que el conjunto de entrenamiento contiene ejemplos de los distintos conceptos que deben ser aprendidos. El sistema utiliza la forma normal disyuntiva modificada como lenguaje de descripción de las reglas usando una representación binaria de las mismas. Cada cromosoma tiene una longitud variable (formada por la concatenación de reglas simples) y representa un conjunto de reglas para clasificar los conceptos descritos en el conjunto de ejemplos. La representación de los individuos se realiza mediante codificación binaria, utilizando para cada atributo un bit que determina su presencia (1) o su ausencia (0). Con respecto a la función de evaluación, establece la bondad de un individuo como un porcentaje de acierto del conjunto de reglas sobre el conjunto de entrenamiento. Los operadores genéticos utilizados son el cruce bi-puntual y la mutación aleatoria en un punto. Además incluye dos nuevos operadores genéticos de mutación: operador de adición de alternativas y operador de eliminación de alternativas que permiten generalizar y especificar un conjunto de reglas.

Un algoritmo supervisado para la inducción de reglas es SIA (Supervised Inductive Algorithm) [Venturini 1994], que combina el algoritmo de cobertura AQ [Michalski et al. 1986] con un algoritmo genético. Está limitado a la representación basada en atributos, aunque es capaz de codificar atributos discretos y continuos.

[1] Estos conceptos (completitud y consistencia), así como otras métricas que se emplean para medir la calidad de las reglas producidas se explican con suficiente detalle en el Apéndice “Métricas para la valoración de la calidad de las reglas”.

Además pondera los atributos asignándoles un peso. Para valorar la calidad de las reglas utiliza el concepto de fortaleza, que es la función objetivo que pretende maximizar el algoritmo genético.

Un algoritmo para el aprendizaje de reglas en lógica de primer orden utilizando un algoritmo genético es SIAO1 [Augier et al. 1995]. Utiliza el principio de cobertura utilizado en AQ [Michalski et al. 1986] donde los ejemplos semilla son generalizados en reglas pero utilizando ahora una búsqueda genética como en el algoritmo SIA. Representa las reglas directamente en lógica de primer orden utilizando los predicados y sus argumentos como genes. El algoritmo intenta encontrar a partir de un ejemplo, la mejor regla según un criterio de evaluación. Entonces todos los ejemplos positivos cubiertos por la regla son eliminados y el sistema aplica el mismo proceso a otro ejemplo escogido del conjunto de ejemplos positivos no cubiertos que quedan. El algoritmo de cobertura utilizado en SIAO1 es similar al AQ. Los operadores genéticos se han adaptado al principio de cobertura del algoritmo. La mutación actúa como una generalización y el cruce es un típico cruce en un punto que intercambia información entre dos individuos. La función de evaluación da una evaluación numérica de un individuo teniendo en cuenta cuatro criterios: consistencia de la regla, completitud de la regla, generalidad sintáctica de la regla y las preferencias del usuario.

El sistema GA-MINER [Flockhart & Racliffe 1996] implementa un algoritmo genético paralelo para la búsqueda de reglas comprensibles. Se trata de una herramienta de búsqueda de patrones general que soporta varias formas de patrones y es capaz de funcionar con una variedad de niveles de supervisión de usuario: minería de datos pura o no dirigida, minería de datos dirigida y prueba de hipótesis y refinamiento. GA-MINER incluye una variedad de formas de patrones basadas en la descripción de subconjuntos (cláusulas que se usan para seleccionar subconjuntos de la base de datos, formados por disyunción o conjunción de restricciones atributo-valor o atributo rango). Los patrones se construyen a más alto nivel como un número de estos subconjuntos. Para evaluar los patrones se han utilizado un número de funciones que estiman su interés, principalmente medidas estadísticas como la ganancia, J-medida y la propuesta por Piatetsky-Shapiro. Además el algoritmo genético utiliza un método de selección de tipo torneo, un operador de cruce uniforme y de un solo punto, y un operador de mutación clásico. La heurística utilizada para actualizar el conjunto de reglas es reemplazar aquella regla con menor ajuste.

Un sistema basado en Algoritmos evolutivos para descubrimiento de reglas de decisión es EDRL (Evolutionary Decisión Rule Learner) [Kwedlo & Kretowski 1998], donde para cada clase etiquetada genera un conjunto disjunto de reglas de decisión en forma de proposiciones. El algoritmo genético busca una regla cada vez y elimina todos los ejemplos que cubre la regla del conjunto de aprendizaje,

repitiendo la búsqueda con los ejemplos restantes. Utiliza una representación tipo Michigan donde cada regla de decisión se representa por una cadena de tamaño fijo. La función de ajuste la define en función del número de ejemplos positivos (POS) y negativos (NEG) que cubre la regla. Además del cruce uniforme, utiliza dos operadores genéticos no estándares: uno denominado condición cambiante, que es muy similar al operador de mutación estándar y el operador de inserción, cuyo objetivo es modificar la regla para que cubra un ejemplo positivo elegido al azar y y que actualmente no cubre la regla.

Un algoritmo genético paralelo es GA-PVMINER [Alves 1999]. Utiliza PVM para descubrir reglas de predicción (no de clasificación), es decir, que puede haber distintos objetivos en el consecuente. La población global del algoritmo genético se divide en varias poblaciones que se asignan a distintos procesadores. Cada individuo representa a una regla simple. Para evaluar la regla se hace de forma independiente de otras reglas. La codificación del cromosoma tiene dos partes, una de k elementos para el antecedente y otra de 1 sólo elemento con consecuente. Después de cada evolución la mejor regla de cada subpoblación se le muestra al usuario. Como función de ajuste utiliza la medida J-measure propuesta por Smyth y Goodman, que mide el grado de interés de la regla. El tipo de cruce implementando cruza sólo un número de atributos de la parte del antecedente. La mutación muta un valor o un atributo. Además tiene un operador eliminación que elimina un atributo de la regla.

Un algoritmo genético para inducción de reglas generalizadas es GA-Nuggets [Freitas 1999]. Estas reglas generalizadas son reglas de predicción de alto nivel. El usuario especifica un conjunto pequeño de atributos objetivos en los que está interesado en predecir. Por lo que es distinto a la tarea de descubrir reglas de asociación, donde cualquier atributo puede estar en el consecuente. Devuelve al usuario un conjunto de reglas de calidad y no todas las reglas. Un cromosoma está formado por una cadena de tamaño fijo con los valores de todos los atributos. Si no hay un atributo se indica con –1. GA.Nuggets sólo maneja atributos categóricos. Los operadores genéticos utilizados son los de mutación, cruce estándar y operadores para añadir y eliminar atributos.

Una extensión mejora del algoritmo genético SIA es ESIA [Liu et. al. 2000] (Extended SIA). Utiliza divide y vencerás para la inducción de reglas. Obtiene reglas de clasificación y no de relación. El algoritmo ESIA aprende una disyunción cada vez y entonces todas las disyunciones juntas forman la descripción del concepto destino. Los operadores genéticos utilizados son de cruce, mutación, especialización y generalización. Además, las probabilidades de los operadores son adaptativas, de forma que se ajustan dinámicamente en base a la aptitud de los individuos. La función de ajuste tiene tres aspectos o componentes: consistencia que utiliza la Laplace para penalizar reglas con poca cobertura, completitud para

medir la exactitud de clasificación y generalidad que mide el número de condiciones de la regla.

El sistema GeSeCo [Weijters & Paredis 2000] (Genetic Sequential Covering Algorithm) es un algoritmo genético con cobertura secuencial para la búsqueda de reglas de clasificación binarias. Los algoritmos de cobertura secuencial consisten en que cada vez que se aprende una regla los ejemplos que cubre dicha regla se eliminan del conjunto de ejemplos antes de buscar una nueva regla. Las reglas se representan por cadenas binarias de longitud variable dependiendo del número de atributos y el número de posibles valores que pueden tomar los atributos. La función de ajuste mide la exactitud de clasificación de la regla. Utiliza un operador de cruce en dos puntos y una mutación que selecciona aleatoriamente el bit de la regla a invertir su valor.

GLOWER [Dhar et al. 2000] es un sistema para descubrimiento de patrones en dominios financieros que utiliza algoritmos genéticos. Incorpora algunas ideas de inducción de árboles y aprendizaje de reglas. El objetivo es encontrar modelos representados por un conjunto de reglas que predigan un número útil de ganancias sorprendentes con una alta exactitud. El algoritmo genético utiliza una representación donde cada patrón o cromosoma representa una regla específica. El algoritmo genético trabaja con múltiples reglas hipótesis que forman la población. Cada patrón es una expresión definida sobre variables del problema y constantes utilizando los operadores relacionales igual, mayor que y menor que y los operadores lógicos Y, O y NO. Para la determinación del ajuste, los cromosomas o patrones se evalúan basándose en un criterio como la entropía, soporte, confianza o cualquier combinación de estos. La población inicial de los patrones es creada aleatoriamente o sesgada de alguna manera. Utiliza operadores de cruce y mutación clásicos. Utiliza técnica de nichos secuenciales, que permite que la creación de nichos o grupos de subpoblaciones se formen de manera dinámica. Y utiliza una heurística de cobertura que va eliminando los ejemplos cubiertos por cada regla descubierta.

Una arquitectura de minería de datos es IMiN [Poon & Prasher 2001], que realiza una hibridación de Algoritmos Genéticos con el algoritmo Apriori maximizando lo mejor de ambos algoritmos. Además permite la interacción con el usuario que puede controlar el flujo del proceso de minería. El algoritmo A priori es un algoritmo ampliamente probado para el acercamiento y la construcción de reglas interesantes, mientras que los algoritmos genéticos permiten la búsqueda de regiones interesantes para investigaciones adicionales. En el primer paso el algoritmo genético busca asociaciones entre atributos del conjunto de datos, el usuario debe seleccionar un grupo de atributos como atributos objetivo de las reglas. Tras un número de generaciones el sistema muestra todas las reglas de la población que satisfacen los requisitos de nivel de soporte y confianza

especificados por el usuario. El usuario selecciona un subconjunto de reglas para continuar el proceso de minería y puede también ajustar los parámetros de minería.

daR [Au & Chan 2002] realiza descubrimiento de reglas de asociación interesantes utilizando un algoritmo genéticos en el que cada cromosoma codifica un conjunto de reglas. Cada alelo del cromosoma codifica una regla, que se representa por un valor simbólico no binario. El proceso evolutivo comienza con la generación de un conjunto inicial de reglas de primer orden (reglas con una condición) utilizando una técnica de inducción probabilística en lugar de hacerlo de forma aleatoria. Las reglas de mayor orden (dos o más condiciones) se obtienen iterativamente. Utiliza un esquema de reproducción de estado estacionario (steady-state) en el que sólo dos cromosomas son reemplazados cada vez. Para identificar reglas interesantes se utiliza una medida de interés. El ajuste de un cromosoma se define en términos de la probabilidad de que el valor de un atributo de una tupla se puede determinar correctamente utilizando la regla que codifica. En lugar de utilizar un umbral especificado por el usuario de soporte mínimo y confianza mínimo, daR utiliza un ajuste residual como una medida objetiva de interés para distinguir reglas de asociación interesantes. Los operadores genéticos utilizados son: Un esquema de selección de tipo ruleta, un cruce de dos puntos, una mutación tradicional y una actualización de la población de tipo estado estacionario reemplazando sólo los dos cromosomas con peor ajuste.

2.3.3.2 Programación genética

La Programación Genética se puede considerar como un paradigma de búsqueda más abierta que el de Algoritmos Genéticos. La búsqueda realizada por la GP puede ser muy útil para clasificación y otras tareas, ya que el sistema puede producir diferentes combinaciones de atributos, utilizando las funciones disponibles en un conjunto preestablecido por la codificación, que no se considerarían utilizando un algoritmo genético convencional. Algunos autores consideran a la GP como una variación de la GA. Sin embargo existen varias diferencias importantes entre estos dos paradigmas:

- **Representación de los individuos.** La principal diferencia radica en la forma de representar a los individuos y los operadores genéticos correspondientes para dichas representaciones. La mayoría de los GP utilizan una representación de los individuos en forma de árbol, donde los nodos internos son funciones y los nodos terminales son variables o constantes del problema a resolver.

- **Contenido de los individuos.** Otra distinción importante es que los individuos en GP pueden contener no sólo valores de las variables, es decir

datos, sino también funciones. Por contra los individuos en GA normalmente sólo contienen datos. De hecho en GP un individuo se suele denominar a menudo como programa.

En general la representación utilizada en GP tiene una mayor expresividad que la que presentan los GAs. El usuario puede incluir funciones arbitrariamente complejas dentro del conjunto de funciones, que se pueden combinar formando nuevas funciones en los individuos. Otra ventaja de GP es que tiene la potencia de producir reglas realmente interesantes y sorprendentes para el usuario y que, al mismo tiempo, tienen una buena exactitud de predicción [Gilbert et al. 1998]. Este hecho parece ser debido a la aplicación de diferentes funciones sobre los atributos originales. Por otro lado, la utilización de GP para el descubrimiento de reglas tiene algunas desventajas potenciales: baja capacidad de generalización en algunos casos y tendencia a producir grandes conjuntos de reglas, que afecta a su comprensibilidad.

2.3.3.2.1 Propiedad de Cierre

Uno de los problemas que presenta la codificación de los individuos en GP es el denominado problema del cierre: dado que la salida de un nodo en un árbol GP se va a utilizar como entrada del nodo padre de éste, hay que garantizar que cada nodo produzca resultados del tipo de datos correcto. En problemas de minería de datos, en los que se utilizan conjuntos de datos que mezclan valores continuos y categóricos, la propiedad de clausura no tendría por qué cumplirse y si se utilizara un sistema GP estándar podrían generarse árboles inválidos. Se han propuesto varias soluciones para afrontar este problema en el contexto del descubrimiento de reglas de predicción: convertir en valores lógicos todos los símbolos terminales, utilización de GP restringida por sintaxis o fuertemente tipada, y utilización de GP basada en gramática.

- **Convertir a valores lógicos todos los terminales.** La forma más general para satisfacer el requisito de clausura consiste en no utilizar múltiples tipos de datos. De esta forma, si el conjunto de los datos contiene atributos de diferentes tipos, hay que convertir todos los tipos a un tipo único e incluir en el conjunto de funciones sólo funciones cuyos argumentos y salida sean de ese tipo. Un ejemplo particular de este tipo de aproximación para la estandarización de tipos de datos consiste en convertir en valores lógicos todos los atributos [Bojarczuk et al. 2000]. Las reglas de predicción son esencialmente combinaciones lógicas de condiciones de atributo-valor, y la propiedad de cierre se satisface convirtiendo todos los terminales en valores lógicos y utilizando sólo funciones lógicas (Y lógico, O lógico, et.)

en el conjunto de funciones. Esta conversión se suele realizar en la etapa de preprocesado de los datos.

- **Restricción por Sintaxis y Fuertemente Tipada.** Otra aproximación distinta para cumplir la propiedad de clausura de GP es utilizar alguna forma de GP restringida por sintaxis o fuertemente tipada. La idea de GP restringida por sintaxis [Koza 1992] es que para cada función disponible en el conjunto de funciones, el usuario especifique qué terminales/funciones se pueden utilizar como nodos hijos de un nodo que contiene la función. La idea de GP fuertemente tipada básica (Strongly Typed GP, STGP) especifica qué hijos puede tener cada función nodo de una forma indirecta [Montana 1995]. Para cada función disponible en el conjunto de funciones, el usuario especifica el tipo de dato de sus argumentos y del resultado.

- **Basada en Gramática.** Una última aproximación para cumplir la propiedad de clausura de GP consiste en utilizar una gramática que imponga las restricciones sintácticas y semánticas [Wong & Leung 2000]. Esta aproximación se puede considerar como una variante de la restricción por sintaxis y fuertemente tipada, pero ahora las restricciones se especifican mediante una gramática.

2.3.3.2.2 Sistemas para Descubrimiento de Reglas basados en GP

El sistema Masson [Ryu & Eick 1996] realiza una tarea de descubrimiento de conocimiento, denominada la búsqueda de descripciones comunes o de parentesco, consistente en buscar consultas en una base de datos orientada a objetos que describan parentescos (valores de atributos comunes) de un conjunto de tuplas especificadas por el usuario. Desde un punto de vista de aprendizaje inductivo esta tarea es significativamente más simple que las tareas de clasificación o la de inducción de reglas generalizadas. Masson realiza la búsqueda de conocimiento en el contexto de bases de datos orientadas a objetos, y los parentescos entre objetos se especifican mediante consultas. El conjunto de funciones que utiliza son: {SELECT, RELATED, GET-RELATED, RESTRICTED},{UNION, INTERSECTION, DIFFERENCE}, {AND, OR, NOT}, {<,<=,=,>,>=} y el conjunto de terminales son: CLASS-SET, SLOT-SET, VALUE-SET. Inicialmente se generan de forma aleatoria un número predefinido de consultas sintácticamente legales formando la población inicial. Cada individuo se evalúa utilizando una función de ajuste predefinida que mide el parecido de la consulta. Para crear las siguientes poblaciones utiliza operadores de selección, cruce y mutación clásicos.

Alex Freitas [Freitas 1997] ha desarrollado un entorno de para la realización de dos tareas de minería de datos (clasificación e inducción) mediante GP. Este

entorno enfatiza la integración del algoritmo de programación genética con sistemas de base de datos relacionales. En particular, el ajuste de los individuos se calcula mediante consultas SQL en un servidor de base de datos paralelo, de forma que se mejora significativamente la eficiencia del algoritmo. Algunas ventajas de esta integración desde un punto de vista de minería de datos son la escalabilidad, control de la privacidad de los datos y la automatización del paralelismo. Para la tarea de clasificación el genotipo de un individuo consiste en la representación en forma de árbol de descriptor-conjunto-tupla. El conjunto de terminales del árbol que codifica un descriptor-conjunto-tupla consta de los nombres de los atributos predictores y el valor correspondiente de sus dominios. El conjunto de funciones están formados por los conectores lógicos (AND, OR, NOT) y los operadores de comparación (>,<,=, ≠). Utiliza un operador de cruce convencional de GP. Un ejemplo de descriptor-conjunto-tupla es: ((A1 > V1) AND (A2 = V2)) OR (A1 < A4). Para la tarea de inducción de reglas de generalización un individuo consta de dos partes: un árbol que codifica el descriptor-conjunto-tupla y un gen que codifica el atributo objetivo. El conjunto de funciones y terminales son similares a los especificados para clasificación. El gen extra que codifica el atributo objetivo introduce la oportunidad de utilizar el operador de mutación que sólo reemplaza el alelo del atributo objetivo. La función de ajuste la define como el producto de los indicadores de exactitud de predicción y simplicidad.

2.3.4 Descubrimiento de reglas interesantes

Un aspecto crucial en la minería de datos es que el conocimiento descubierto sea interesante para el usuario [Klösgen & Zytkow 2002], en el sentido de sea sorprendente, inesperado, útil (en el sentido de que lo pueda utilizar posteriormente para realizar algo) y novedoso (conocimiento que no tuviera ya el usuario).

La investigación sobre el interés del conocimiento se ha desarrollado principalmente en el área de la minería de reglas de asociación [Piatesky-Shapiro & Matheus 1994], debido a que el número de reglas encontradas por este tipo de algoritmos suele ser muy grande, y la mayoría de las reglas no son interesantes para el usuario. En este sentido, la minería de reglas de asociación interesantes incorpora restricciones especificadas por el usuario sobre el tipo de reglas deseadas y define métricas objetivas del interés de la regla. Dicho interés se puede medir utilizando dos tipos de factores, denominados por Liu [Liu et al. 2000] factores *objetivos* y *subjetivos*.

2.3.4.1 Factores Subjetivos

Los factores subjetivos están dirigidos por el usuario. En este sentido, [Silberschatz & Tuzhilin 1996] definen dos factores, inesperabilidad y procesabilidad, que relacionan directamente con el interés de las reglas. El primero de estos factores está relacionado con el hecho de que, si una regla resulta sorprendente para el usuario, es decir, si revela conocimiento que él desconoce o contradice algún conocimiento previo, será interesante para él. El segundo está relacionado con el uso que se pueda hacer de la regla obtenida. Estos dos factores no son mutuamente excluyentes, de forma que desde el punto de vista subjetivo el interés de una regla se puede clasificar en cuatro categorías:

- **Reglas inesperadas y aplicables.** Estas son las reglas más interesantes.
- **Reglas inesperadas y no aplicables.** Son reglas menos interesantes.
- **Reglas aplicables y no inesperadas**. Son reglas menos interesantes.
- **Reglas inesperadas y no aplicables**. No son reglas interesantes.

Además de otros posibles factores, existen distintas aproximaciones para la implementación de estos factores subjetivos del interés de las reglas. De entre ellas, la aproximación más ampliamente utilizada es la utilización de restricciones [Roberto et al.1997], que algunos autores también han denominado filtros, condiciones o plantillas [Klemettinen et al. 1994]. Otras aproximaciones distintas son la utilización de impresiones generales [Liu et al. 1997] y la búsqueda de excepciones o reglas inesperadas [Suzuki & Zytkow 2000]. A continuación se van a describir ambas aproximaciones.

2.3.4.2 Factores Objetivos

Los factores objetivos están dirigidos por los datos y solucionan el problema de la calidad de las reglas [Tan & Kumar 2000] utilizando medidas de interés que representan distintos aspectos de la regla. Este tipo de factor objetivo se ha utilizado ampliamente en otras áreas de investigación como estadística, aprendizaje de máquinas, etc. para realizar la tarea de identificación de conjuntos de datos o reglas interesantes.

La evaluación de las reglas de tipo *Si-Entonces* obtenidas de un proceso de descubrimiento de conocimiento tiene como objetivo la ordenación del conjunto de reglas descubiertas según una medida de evaluación, de cara a realizar un filtrado o selección de las reglas que superan un valor de umbral para dicha medida [Yao & Zhong 1999]. De esta forma se consigue reducir el número de reglas potenciales descubiertas sobre un conjunto de datos, pasando de un número muy alto de reglas a un número reducido y realmente útiles. Se han descrito multitud de medidas para

este fin, las cuáles se presentan en el Apéndice que hemos denominado "Métricas para la evaluación de la calidad de las reglas", orientadas a medir distintos aspectos de su calidad.

Por otro lado, Piatetsky-Shapiro [Piatetsky-Shapiro 1991] han propuesto tres principios para medir el interés de las reglas de asociación. Según estos autores, cualquier medida de precisión o interés debe satisfacer 3 propiedades básicas:

1. **RI = 0 si p(A,C) = p(A)*p(C).** Esto en principio nos dice que la medida RI es cero si el antecedente y el consecuente de la regla son estadísticamente independientes.

2. **RI se incrementa monótonamente con p(A,C) cuando los otros parámetros están fijos.** Que significa que para un valor fijo de p(A) y p(C) el RI se incrementa monótonamente con p(A,C).

3. **RI se decrementa monótonamente con p(A) o p(C) cuando los otros parámetros están fijos.** Que significa que para un valor fijo de p(A) y p(A,C) el RI se decrementa monótonamente con p(C) o que para un valor fijo de p(C) y p(A,C) el RI se decrementa monótonamente con p(A).

Estos principios pueden servir de guía para evaluar diferentes medidas objetivas, aunque en la práctica hay muchas buenas medidas que no satisfacen todos los principios. Por ejemplo, el primer principio es demasiado rígido debido a que especifica que el valor del interés absoluto debe de ser independiente de los elementos. Estos principios se han diseñado principalmente para considerar los factores de calidad de reglas ampliamente utilizados como, por ejemplo, los factores de cobertura, completitud y confianza.

Capítulo 3
Diseño e Implementación del ASWE

En este capítulo se van a describir una serie de cuestiones relacionadas con el diseño e implementación del ASWE desarrollado. Comenzaremos el capítulo presentando una propuesta metodológica para la mejora asistida de este tipo de sistemas mediante el uso de técnicas de descubrimiento de conocimiento, para continuar describiendo el diseño del ASWE desarrollado; dicha descripción se centrará en los aspectos relativos a la adaptación y la captura de información. El Capítulo finaliza con una serie de consideraciones relativas a la implementación realizada, haciendo también referencia al curso desarrollado sobre el sistema Operativo Linux.

CAPÍTULO 3:DISEÑO E IMPLEMENTACIÓN DEL ASWE

3.1 Una nueva Metodología para la mejora de ASWEs

En el capítulo anterior se hizo referencia a la metodología que se viene utilizando clásicamente en la construcción de ASWEs [Hérin et al. 2002], indicando que se trata de una metodología descendente, formada por una serie consecutiva de etapas sin ningún tipo de retroalimentación. En estos sistemas no existe claramente una etapa de mantenimiento del curso y, una vez finalizada su construcción y publicación, no suelen producirse modificaciones en éste. Además, el diseñador del curso sólo utiliza la información de evaluación de los estudiantes para comprobar su correcto aprendizaje.

Se propone añadir una *etapa de mantenimiento* que utiliza esta información como base para el descubrimiento de información hasta ahora no descubierta por las técnicas estadísticas clásicas o difícilmente extraída directamente por una persona humana debido a su gran volumen.

La mejora en la metodología [Romero et al. 2002a] que se propone para el desarrollo de los sistemas adaptativos para educación basada en Web (ver Figura 3.1) es una *metodología cíclica*, que consta de las siguientes etapas:

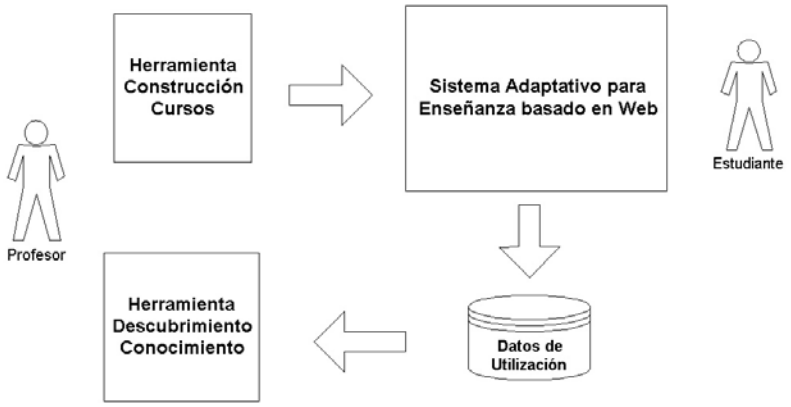

Figura 3.1: Metodología propuesta para la mejora de ASWEs.

- **Construcción del curso.** Esta primera etapa se corresponde con las fases 1, 2, 3 y 4 descritas anteriormente (consenso del contenido del curso, recopilación de información sobre el tema, planificación del contenido y composición del curso – ver Sección Construcción de los ASWE del capítulo *Antecedentes*). El diseñador del curso suele ser el

responsable de construir el curso proporcionando para ello la información del modelo del dominio (contenido), el modelo pedagógico (reglas) y el módulo interfaz (aspecto gráfico). Normalmente suele utilizar una herramienta autor para facilitar esta tarea, ya sea una herramienta genérica de tipo comercial como DreamWeaver, Toolbook, Director, etc. o una herramienta específica diseñada para un tipo de curso concreto (HamWeb). El resto de información, modelo del tutor (motor de ejecución del sistema adaptativo) y modelo del estudiante (información del alumno) suelen venir dada por el propio sistema adaptativo de soporte de enseñaza utilizado. Para poder aplicar nuestra metodología, es necesario que el sistema desarrollado almacene la información relativa a la interacción del usuario con el sistema. Al finalizar esta etapa el curso debe colocarse en un servidor web para que los alumnos puedan utilizarlo de forma remota.

- **Ejecución del curso.** Es la realización del curso por parte de los alumnos y corresponde con la etapa 5 (impartición del curso, ver Sección Construcción de los ASWE del capítulo *Antecedentes*). Los estudiantes, utilizando un navegador web, se conectarán al sitio que alberga la aplicación para realizar el curso. Durante esta etapa se recogerá la información de utilización de forma transparente para el alumno, almacenándose en el servidor web dentro de los distintos ficheros históricos o *logs*.

- **Aplicación de técnicas de minería de datos.** Consiste en la aplicación de técnicas de descubrimiento de conocimiento sobre los datos de utilización recogidos en la etapa anterior de ejecución del curso. Esta etapa se añadiría a la etapa 6 de evaluación del curso (ver Sección Construcción de los ASWE del capítulo de Antecedentes). El objetivo es procesar toda la información almacenada en el sistema, previo preprocesado y colocación en un sistema de gestión de bases de datos que garantice una manipulación más rápida de dicha información. Una vez transferidos los datos, el diseñador del curso puede aplicar los algoritmos de minería de datos y descubrir relaciones importantes entre éstos.

- **Mejora del curso.** En esta etapa de mantenimiento se realizarán modificaciones sobre el curso orientadas a solventar carencias, corregir problemas o mejorar determinados aspectos del mismo. Esta etapa de mantenimiento está asistida por las técnicas de adquisición de conocimiento, diferenciándose en eso de la revisión que podría llevarse a cabo en algunos sistemas tradicionales. El diseñador del curso,

> ayudado por la información que se le suministra en forma de relaciones importantes descubiertas entre los datos, realiza las modificaciones que crea más adecuadas para mejorar el rendimiento del curso. Estas modificaciones pueden afectar al contenido del curso, su estructura, el interfaz gráfico, etc.

Como puede comprobarse, la introducción de una etapa de mantenimiento basada en el uso de técnicas de aprendizaje automático mejoraría sensiblemente la calidad del sistema, dada la posibilidad de aprovechar la enorme cantidad de información que se genera como consecuencia de la interacción de los alumnos con el sistema. Además, y dado que este ciclo puede repetirse cuantas veces se desee con un coste relativamente bajo (la fase de extracción de conocimiento se realiza automáticamente), se confiere un carácter dinámico al ciclo de vida de la aplicación, que puede ir mejorando progresivamente a medida que se dispone de más información.

Es evidente que una condición para el éxito de nuestra propuesta de mejora de los ASWEs es la constatación de que los algoritmos de obtención de conocimiento van a aportar información relevante para la mejora del sistema, y ese es el objetivo fundamental que se persigue con el trabajo desarrollado en esta Tesis Doctoral. Es, por tanto, fundamental disponer de un ASWE que sirva como banco de pruebas para la obtención de información que se utilizará en la fase de extracción de conocimiento.

Desgraciadamente, no se ha encontrado ningún sistema de las características requeridas para realizar esta fase de recogida de información. Por esta causa, se decidió elaborar uno propio aprovechando la arquitectura del sistema AHA [De Bra et al. 1999], una herramienta de dominio público[1] que, si bien no está orientada a educación, presentaba mucha similitud con la herramienta que necesitábamos. Este sistema ya implementado sirvió como plataforma para el desarrollo de un curso sobre el Sistema Operativo Linux, que fue ejecutado por 40 alumnos del I.E.S. "Gran Capitán" de Córdoba y 10 expertos informáticos en dicho sistema operativo. La información generada por los alumnos del curso es la que se ha utilizado en la fase de extracción de conocimiento, que ha motivado el desarrollo de la investigación que se describe en los siguientes capítulos.

A continuación se describirá el diseño del sistema, y se mostrarán algunas características relacionadas con la implementación (tecnologías utilizadas, algoritmos para el modelo de usuario y tutor) etc. Dicha descripción no pretende ser exhaustiva sino que, se procura dar una visión sobre los requisitos que se

[1] El sistema AHA se distribuye con licencia GPL, por lo que podía accederse a su código fuente y modificarlo en base a nuestros requisitos.

imponen a un sistema de este tipo para que pueda ser utilizado eficazmente en el desarrollo de esta metodología.

3.2 Diseño del ASWE

Para el diseño del ASWE, nos hemos basado el modelo de referencia AHAM (Adaptive Hypermedia Application Model, [Wu et al. 1999]), que es una extensión del modelo Dexter [Halasz & Schwartz 1994]. Este modelo fue diseñado para capturar las estructuras y funcionalidades de los sistemas hipermedia adaptativos, presentando los componentes típicos de estos sistemas (ver Figura 3.2):

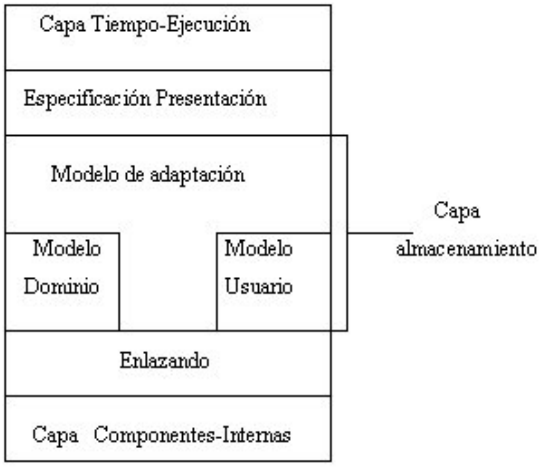

Figura 3.2: Modelo de referencia AHAM.

La división en un modelo de dominio (Domain Model, DM), modelo de usuario (User Model, UM) y modelo de adaptación (Adaptation Model, AM) proporciona una separación clara de intereses cuando se desarrolla una aplicación hipermedia adaptativa. Desgraciadamente, un problema en muchos de los sistemas adaptativos actuales es que esos 3 factores o componentes no están claramente separados. El modelo AHAM apoya la separación de estos componentes en los AHS que utilizan dicho modelo.

Para desarrollar una adaptación basada en DM y UM, un autor necesita especificar que la interacción del usuario con los AHS influye en la presentación de la información contenida en el DM. En AHAM, esto se hace a través de un AM consistente en reglas de adaptación. Un motor de adaptación (Adaptive Engine,

AE) usa estas reglas para manipular enlaces y generar todo lo que el modelo de Dexter [Halasz & Schwartz 1994] denomina *especificaciones de presentación*.

En el diseño específico del ASWE propuesto (ver Figura 3.3) se ha ampliado la funcionalidad del modelo AHAM para que permita almacenar toda la información de interacción del alumno dentro de ficheros *log* mejorados. Asimismo, se ha definido un modelo didáctico específico para nuestros cursos. Las principales características de este modelo didáctico son:

- La estructuración del dominio se basa en la estructuración típica de un curso, en la que el curso está formado por una serie de *temas*, que están dividido en distintas *lecciones*, y donde cada lección contiene una serie de *conceptos*. Un concepto esta compuesto por una serie de *escenarios* o páginas web.

- El control del aprendizaje o evaluación del alumno se realiza en cada tema mediante una test inicial al comienzo de cada tema que determina el nivel de conocimiento inicial del tema, después se le propone al alumno una actividad. compuesta por escenarios tipo exposición y ejercicios, por cada uno de los conceptos que componen el tema. La realización de esta actividad influirá en la determinación del nivel del alumno por cada concepto, y finalmente un test final de todo el tema que determina el nivel final del alumno en el tema.

- El modelo presenta la capacidad de trabajar con diferentes grados de dificultad, permitiendo adaptar la accesibilidad y los conceptos presentados a cada alumno en función del nivel de conocimiento que posee en el tema.

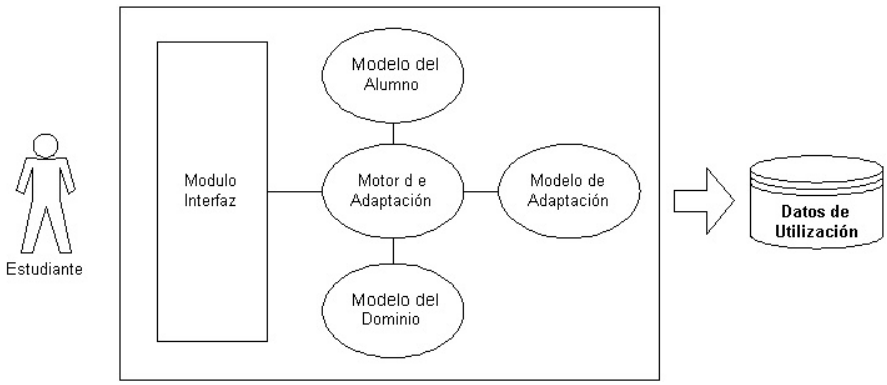

Figura 3.3: Arquitectura del ASWE propuesto.

En las siguientes secciones, se van a describir algunos aspectos importantes relacionados con cada uno de los componentes del ASWE propuesto (ver Figura 3.3).

3.2.1 Modelo del Dominio

El modelo de dominio va a estar formado básicamente por conceptos y relaciones entre éstos. Un concepto es un componente que representa un elemento abstracto de información que es la unidad principal del sistema. Un concepto puede ser de dos tipos:

- **Atómico.** Corresponde a una unidad básica de información que pertenece al nivel interno del sistema.

- **Compuesto.** Contiene una serie de subconceptos o conceptos hijos que pueden ser, a su vez, atributos atómicos o compuestos.

Existen distintos tipos de relaciones entre los conceptos que se van a presentar en el sistema, las cuales pueden ser de los siguientes tipos:

- **Relación parte de.** Corresponde a conceptos de tipo compuesto y a la relación entre padre e hijos, indicando que una serie de conceptos hijos forman parte de un concepto compuesto o padre. En terminología de Lenguaje de Modelado Unificado (Unified Modelling Language, UML), se trataría de una relación de composición. La jerarquía de un concepto compuesto debe seguir una estructura de grafo dirigido acíclico, como se muestra en la Figura 3.4.

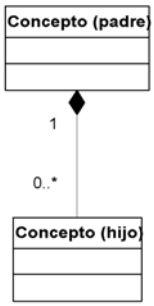

Figura 3.4: Relación Parte De entre conceptos.

- **Relaciones enlace.** Indica relaciones que puede haber entre conceptos (ver Figura 3.5). Es el tipo de relación más común y se corresponde con los enlaces (link) de la mayoría de los sistemas hipermedia.

- **Relaciones prerrequisito.** Es un tipo de relación de requisito que puede haber entre conceptos (ver Figura 3.5). Por ejemplo, cuando un concepto C1 es prerrequisito de otro concepto C2, significa que un usuario deberá leer C1 antes de leer C2. Pero no significa que deba haber un enlace de C1 a C2, sino que el sistema tendrá en cuenta que leer C2 no será deseable (o aconsejable) si no se tiene el suficiente conocimiento que se adquiere al leer C1. Todo prerrequisito debe tener al menos un elemento origen y un elemento destino.

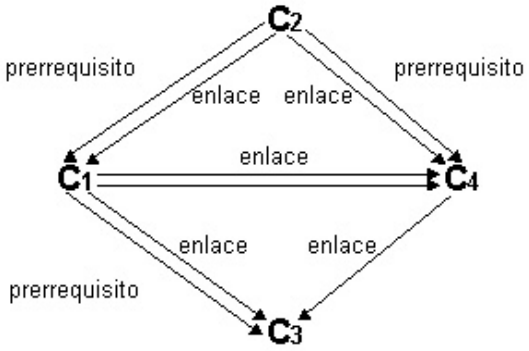

Figura 3.5: Ejemplo de relaciones prerrequisito y enlaces entre conceptos.

Además, nuestro modelo del dominio presenta una estructuración que organiza el material docente en cursos, temas, lecciones y conceptos. Cada uno de estos elementos está formado por elementos de un nivel inferior. De este modo, cada curso está formado por una serie de temas o capítulos, los cuales se dividen en lecciones, éstos en conceptos y, por último cada concepto presentará un conjunto de escenarios asociados (ver Figura 3.6).

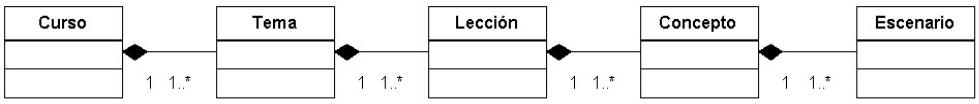

Figura 3.6: Estructuración de un curso en UML.

Los escenarios van a permitir enseñar y evaluar un concepto y pueden ser de los siguientes tipos:

- **Escenario tipo exposición.** Es una combinación de imágenes, textos, sonidos y vídeos que muestran los contenidos teóricos del concepto. Este tipo de escenario es el primero que se le muestra al alumno en cada concepto a modo de explicación del mismo.

- **Escenario tipo ejercicio.** Compuesto por pruebas que el alumno debe resolver. Permite reforzar los conocimientos adquiridos en la exposición y evaluarlos y es mucho más interactivo que el escenario de exposición. Existen distintos tipos de ejercicios, entre los que podemos destacar:

    - **Escenario tipo test.** Donde se presenta una pregunta con múltiples respuestas, algunas correctas y otras incorrectas sobre los contenidos teóricos del concepto. Tras finalizar el test se le muestra si la respuesta elegida ha sido la correcta o no. El test que se acaba de describir es un tipo de test básico formado por una sóla pregunta, pero también puede haber test con multiples preguntas, e incluso test de tipo adaptivo.

    - **Escenario tipo relaciona.** Donde se presentan dos columnas con elementos (textos o imágenes) que el alumno debe relacionar, pulsando sobre uno de los objetos de una columna y luego sobre el correspondiente de la columna contraria. Si la relación es correcta aparecerá una flecha relacionando ambos elementos.

    - **Escenario tipo vídeo interactivo.** Donde se muestran varias imágenes fijas (frames), que simulan una serie de pasos para la explicación de una determinada actividad. En cada paso se le propone al alumno que pulse sobre cualquier zona del escenario o que escriba una palabra o frase en un cuadro de texto. Si lo hace correctamente se mostrará la siguiente imagen y un mensaje de opción correcta, en caso contrario se mostrará un mensaje de ayuda.

Una de las características específicas que posee nuestro modelo del dominio es la clasificación o agrupación de las lecciones y conceptos en distintos *grados de dificultad*, estableciendo, el diseñador de curso. una jerarquía independiente de la definida a través de las relaciones detalladas anteriormente. De esta forma, dentro de cada tema, el diseñador del curso debe determinar qué conceptos o lecciones están asociados a cada uno de los distintos grado de dificultad. Posteriormente, el sistema, con los datos de evaluación del alumno al realizar una serie de ejercicios de una actividad o test inicial o final, determinará el nivel de conocimiento que posee. En función del nivel se realiza la adaptación del tema, de forma que sólo se

le mostrará al alumno, las lecciones que tienen un determinado grado de dificultad apropiado a dicho nivel de conocimiento.

3.2.2 Modelo del Alumno

El modelo de usuario está formado por una serie de atributos generales, relacionados con la identificación y una serie de atributos asociados a los progresos realizados por el usuario en el sistema. En cuanto a los primeros, los atributos que se almacenan son los siguientes:

- **Identificación.** Información personal sobre edad, sexo y estudios, además del nombre y clave para la entrada al curso a través de Internet.

- **Historial de navegación.** Información sobre la navegación realizada. Para cada página visitada se almacena el instante de tiempo en que se accedió.

- **Preferencias.** Estas preferencias, prefijadas por defecto y que pueden ser actualizadas por el usuario, están relacionadas con aspectos de la interfaz. Como por ejemplo el color de los hiperenlaces, color de fondo de las páginas web, etc.

- **Objetivos.** Información relacionada con objetivos e intereses manifestados por el alumno al comienzo del curso.

- **Experiencia.** Conocimiento anterior que posee el alumno. Se obtiene tras la realización de un test de preconceptos por parte del usuario.

Además de estos atributos generales, existe información relativa al aprendizaje y progreso del alumno en el curso, almacenándose para ello la siguiente información:

- **Marca de continuación**. Indica el punto exacto dónde se ha quedado el alumno en la última sesión, para poder continuar en una sesión futura cuando se vuelva a conectar al curso desde este punto donde lo dejó.

- **Marca de Utilizado**. Lista de escenarios que ya han sido utilizadas por un alumno, es decir, que ya se han mostrado en alguna sesión anterior o en la actual. Esto se hace con el objeto de no volver a mostrar la misma información y evitar el aburrimiento del alumno.

- **Acierto.** Indica el acierto o fallo cometido por el usuario en cada escenario de tipo ejercicio.

- **Nivel**. Es el grado de conocimiento sobre cada concepto y tema asignado al alumno por el sistema. Este campo se actualiza en función de los resultados obtenidos en las actividades de cada concepto y de los test iniciales y finales de cada tema.

Además, el sistema permite, al diseñador del curso, ampliar el número de atributos dándole a éste la posibilidad de ampliar el modelo de usuario.

3.2.3 Modelo de Adaptación

El modelo del adaptación está basado en una serie de reglas que van a permitir la adaptación del contenido de la información del sistema hipermedia y de su estructura de enlaces. Estas reglas establecerán la estructura de la relación que existe entre el modelo de dominio, el modelo de usuario y la presentación personalizada que se generará para cada usuario. Dichas reglas son establecidas por el diseñador del curso.

Las reglas que permiten al diseñador del curso especificar la adaptación del curso tienen el siguiente formato:

SI <Antecedente> ENTONCES <Consecuente>

Donde el antecedente de la regla se refiere al nivel de conocimiento obtenido por el alumno en los conceptos y temas, y el consecuente se refiere a acciones de adaptación que va a realizar el sistema, pudiendo establecer para ello el grados de dificultad del tema o mostrar mensajes de recomendación al alumno sobre qu debe realizar a continuación, como por ejemplo, pasar al siguiente tema o concepto, repetir el concepto o tema, realizar las actividades o test, etc.

Existen dos tipos de órdenes que controlan el acceso a los conceptos, relacionadas con las principales técnicas de adaptación existentes:

- **Inclusión condicional de fragmentos**: El usuario sólo podrá ver los fragmentos que le son planificados por el sistema, en función de sus objetivos, intereses, nivel obtenido y dependencia entre los distintos conceptos. El resto de fragmentos, que no son aconsejables, no se muestran al usuario.

- **Ocultación o anotación de enlaces**: El sistema modificará el aspecto de los enlaces en función del resultado obtenido. En este sentido hay que indicar que éstos se presentan como texto coloreado y no mediante

subrayado, como es habitual en otras aplicaciones web. Se suele utilizar una notación con tres tipos de enlaces:

- ***Enlaces recomendados**:* Se trata de los enlaces con páginas que son aconsejables y que no han sido aun visitadas por el alumno. Suelen ponerse en un color que resalta claramente del texto de la plantilla.

- ***Enlaces neutros**:* Se trata de enlaces cuyo contenido no está directamente relacionado con los objetivos que se persiguen. Suele ponerse en un color diferente del de los enlaces recomendados.

- ***Enlaces no aconsejables**:* Suelen ponerse en el mismo color del texto normal, de manera que no son perceptibles por el usuario.

3.2.4 Motor de Adaptación

El motor de adaptación es el elemento más importante del sistema, siendo el responsable del funcionamiento del mismo. Este módulo proporciona aspectos dependientes de la implementación. Sus principales funciones son las siguientes:

- **Realizar la adaptación**. Presentar y adaptar el modelo del dominio a cada usuario, dependiendo de su modelo de alumno y del modelo de adaptación.

- **Actualizar el modelo de usuario**. Actualizar el modelo del usuario conforme este interactúa con el curso.

El motor de adaptación va a ser el encargado de establecer los *niveles de accesibilidad o dificultad* de las lecciones de cada tema y los siguientes temas recomendados, en función del nivel de conocimiento obtenido por cada alumno en los test y actividades de evaluación de los conceptos y los temas.

La Figura 3.7 muestra como se realiza la adaptación del curso.

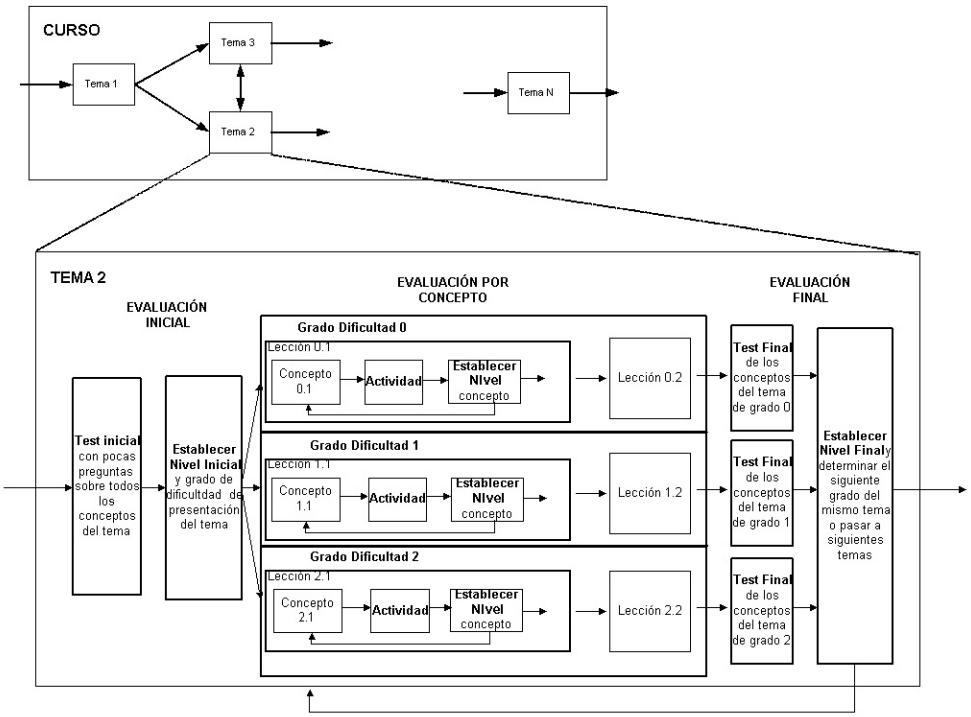

Figura 3.7: Realización de la adaptación en el ASWE propuesto.

Se puede apreciar que un curso es como un grafo dirigido donde los nodos son temas, de forma que varios alumnos pueden realizar varias rutas de ejecución del curso distintas dependiendo de que temas tiene disponibles en cada momento. El diseñador del curso del curso es de nuevo el encargado de establecer la accesibilidad de cada tema, es decir, a que temas se puede acceder tras finalizar el tema actual. También debe de establecer cual es el tema inicial o de comienzo del curso que es el primero que se le va a mostrar a todos los alumnos.

Una vez que un alumno accede a un tema, el funcionamiento del sistema va a ser siempre el mismo y es el que se describe a continuación.

Lo primero que se va a mostrar al alumno es un conjunto de preguntas que forman el test inicial que va a permitir evaluar el nivel de conocimiento inicial que posee el alumno. Dependiendo del nivel obtenido por el alumno, el sistema determina a que grado de dificultad se le va a mostrar del tema y por tanto, las lecciones a las que puede acceder. Para cada lección el alumno debe realizar las actividades de evaluación de todos los conceptos tras visualizar las exposiciones, de forma que se pueda determinar cual es el nivel de conocimiento que posee en cada concepto. Dependiendo del nivel obtenido se permitirá pasar al siguiente

concepto o repetir la exposición y evaluación del mismo concepto hasta que obtenga el nivel de conocimiento propuesto en la correspondiente regla por el diseñador del curso. Finalmente tras haber superado las actividades de los conceptos de todas las lecciones que componen el grado de dificultad establecido al comienzo del tema, se le realiza al alumno una evaluación final del mismo, mediante un test final con preguntas de los conceptos de ese grado. El nivel final obtenido por el alumno, junto con los niveles obtenidos en las actividades y el test inicial determinarán si el alumno debe volver a realizar alguna actividad, o repetir el tema en igual grado o distinto grado de dificultad, o si por el contrario puede pasar al siguiente tema.

Además, durante todo este proceso de interacción del alumno y adaptación del sistema se va almacenado toda la información generada en el historial del alumno, tanto la correspondiente los escenarios (tiempo, acierto) como a los niveles de conocimiento obtenidos por el alumno tras realizar los test y actividades.

Presentados los aspectos generales de funcionamiento del esquema de adaptación utilizado en nuestro sistema, pasaremos a explicar, con más detalle, el funcionamiento de los elementos que están más íntimamente relacionados con el algoritmo de adaptación: evaluación inicial y final de cada tema y actividades de control.

3.2.4.1 Evaluación Inicial

Esta evaluación previa establece el grado a mostrar del tema al alumno, se lleva a cabo mediante *tests iniciales* que contienen preguntas de opción múltiple sobre todos los conceptos que componen el tema al que se va a acceder. Estos test iniciales aunque implementan adaptación, no llegan a ser test de tipo adaptativo [Rios et al. 1998]. El uso de adaptación en los tests va a permitir reducir el número de preguntas necesarias para estimar el nivel de conocimiento inicial del estudiante permitiendo dirigir al estudiante al grado de dificultad del tema adecuado a su nivel.

El funcionamiento específico de estos test iniciales y su adaptación es el siguiente (ver Figura 3.8): se realizarán un número determinado de preguntas al usuario (por defecto cuatro). La primera pregunta se escogerá al azar de entre todas las preguntas disponibles de todos los conceptos con el grado inicial establecido (por defecto grado medio). Si el alumno acierta, la siguiente pregunta será escogida al azar de entre las preguntas de todos los conceptos de grado alto (de forma que el acierto incrementa el grado de dificultad de la siguiente pregunta); en cambio si el alumno falla, la siguiente pregunta será escogida al azar de entre las preguntas de conceptos de grado bajo (de forma que el fallo reduce el grado de dificultad de la

siguiente pregunta). Este proceso se repetirá, hasta que se formulen todas las preguntas establecidas y entonces se determinará el nivel inicial del alumno dependiendo del número total de preguntas acertadas (también establecido por el diseñador del curso).

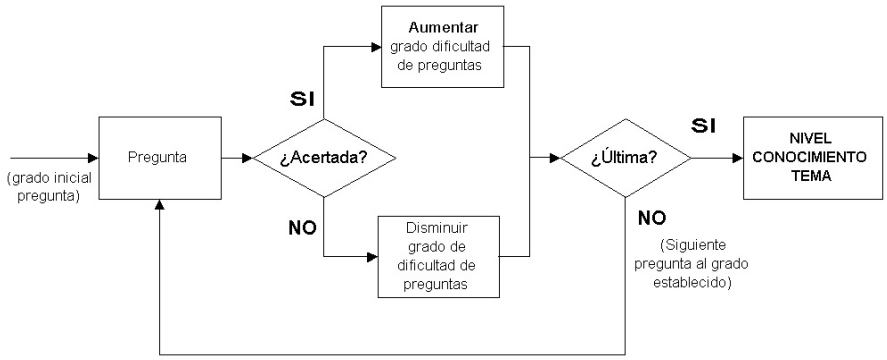

Figura 3.8: Funcionamiento de la evaluación inicial.

3.2.4.2 Actividades de control

Son actividades que evalúan el aprendizaje de cada concepto y se realizan al término de la visualización del contenido teórico o escenario tipo exposición de cada concepto. Suele tratarse de preguntas individuales de opción múltiple, aunque se puede utilizar otros escenarios tipo ejercicio.

El funcionamiento del modelo didáctico es el siguiente (ver Figura 3.9). Se le presenta al alumno un número de preguntas perteneciente a un primer grupo, elegidas de forma aleatoria, sobre el concepto que acaba de leer, si acierta más de un número mínimo establecido, se asigna un nivel dependiendo del número de aciertos en ese concepto y pasa al siguiente concepto. Si por el contrario, no consigue el número mínimo de aciertos establecido, se le presenta un segundo grupo de preguntas. Si acierta, en esta segunda oportunidad, más del número mínimo de preguntas establecidas, se asigna un nivel de conocimiento dependiendo del número de aciertos. Si, por el contrario, no consigue el número mínimo de aciertos establecido, se le volverá a mostrar el escenario tipo exposición del mismo concepto, tantas veces como sea necesario, hasta que logre superar la actividad.

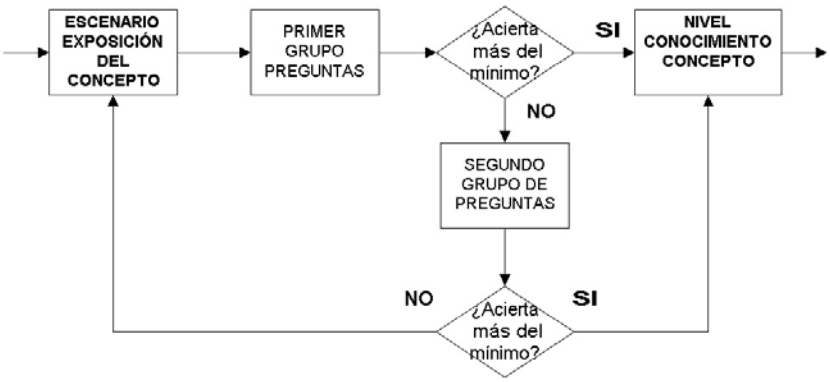

Figura 3.9: Funcionamiento del modelo de actividades de control.

3.2.4.3 Evaluación Final

Para poder acceder al test final de un tema es necesario tener realizadas y superadas previamente las actividades de todos los conceptos del tema a un grado de dificultad.

La evaluación final consiste en un *test de tipo clásico* formado por un grupo preguntas (especificadas por el diseñador del curso) sobre los distintos conceptos, que se muestran en forma de lista, con un grado de dificultad igual al obtenido al realizar las actividades del tema.

El funcionamiento de este test final es el siguiente (ver Figura 3.10), se le presenta al alumno dicho grupo de preguntas, si el número de aciertos es mayor que el mínimo establecido, se le asigna un nivel final del tema, dependiendo del número de preguntas acertadas. Si en cambio, el alumno no supera el número mínimo, se le obligará a repetir el mismo tema con igual grado de dificultad, hasta que supere el número mínimo de preguntas del test final.

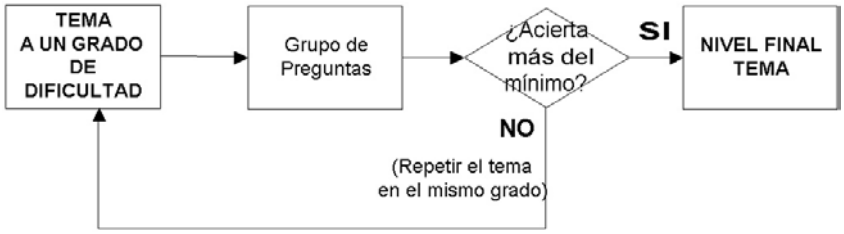

Figura 3.10: Funcionamiento de la evaluación final.

3.2.5 Módulo Interfaz

Desde el módulo interfaz, el alumno va a interactuar con el entorno del curso. Los componentes principales (ver Figura 3.11) que va a tener el módulo interfaz son:

- **Índice del tema actual.** Muestra las distintas lecciones y/o conceptos que componen el tema que actualmente se está presentando. El alumno podrá pulsar sobre cualquier elemento del índice y aparecerá el contenido del escenario exposición del concepto seleccionado.

- **Escenario actual.** Muestra el contenido de los escenarios de los conceptos, ya sean de tipo exposición o de tipo ejercicio. Dispone de enlaces para pasar de un tipo de escenario a otro.

- **Información y Opciones.** Conjunto de opciones que permiten acceder a la información sobre el estado actual tanto del curso, como del usuario y de su nivel de conocimiento en cada momento.

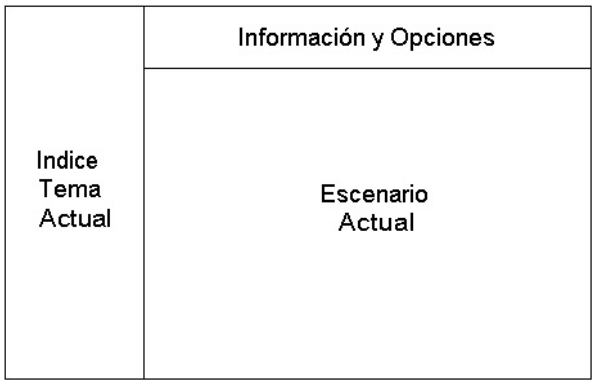

Figura 3.11: Componentes del Módulo Interfaz.

Además de este interfaz principal del curso, también van a existir otros interfaces específicos. Un ejemplo es el interfaz para la conexión al curso, donde se le pide al usuario que se identifique mediante su nombre y palabra clave. Otro ejemplo es el interfaz para la ayuda de utilización del curso, donde se explica cada una de las opciones del curso y la forma correcta de utilización.

3.3 Implementación del ASWE

La implementación del ASWE se corresponde con el diseño expuesto en el apartado anterior realizado utilizando como referencia el sistema AHA (Adaptive Hipermedia Architecture) [De Bra & Rutier, 2001] disponiéndose del código fuente para la realización de modificaciones o adaptaciones, por lo que es ideal como arquitectura base sobre el que desarrollar nuestro sistema.

A pesar de las buenas características del sistema, al estar orientado a aplicaciones de propósito general, no presenta muchos de los requisitos impuestos anteriormente y que derivan del modelo que hemos planteado en la Sección anterior. Por esta razón, ha habido que realizar una serie de modificaciones sobre la versión original de AHA, cuya explicación es el principal objetivo de esta sección.

A continuación se van a describir cada uno de los componentes implementados que forman parte del ASWE (ver Figura 3.12).

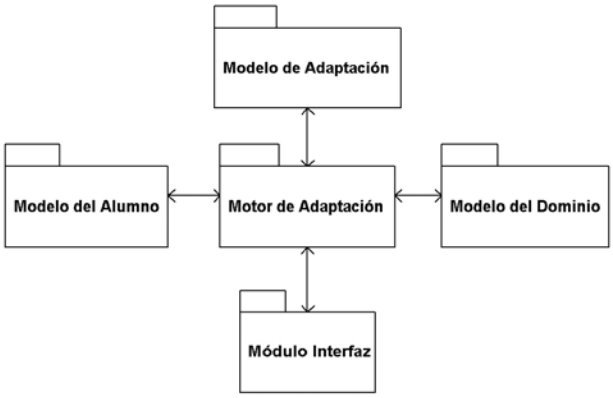

Figura 3.12: Diagrama de paquetes UML del ASWE implementado.

La implementación del ASWE propuesto se ha realizado utilizando principalmente dos tecnologías Web:

- **XML (**eXtensible Markup Language**).** Este lenguaje de marcas extensible, es un nuevo estándar que mejora al tradicional HTML (HyperText Modelling Language) que permite estructurar la información del curso de forma que se pueda adaptar más fácilmente a cada usuario. Este lenguaje de marcas se ha utilizado para implementar el modelo del dominio, el modelo de adaptación y gran parte del modelo del alumno.

- **Servlets.** Son aplicaciones Java que se ejecutan dentro de un servidor Web y que permiten controlar sesiones de páginas Web. Estas aplicaciones se han utilizado para realizar la personalización del curso a cada alumno y para implementar el motor de adaptación del ASWE.

3.3.1 Modelo del Dominio

En la implementación que se ha realizado del modelo del dominio, un concepto va a estar formado por una o varias páginas Web relacionadas. Cada página Web va a ser un fichero XML que contiene una serie fragmentos incluidos condicionalmente y enlaces de hipertexto.

El sistema AHA [De Bra & Rutier 2001] representa el modelo del dominio de la forma descrita, pero se le han realizado varias modificaciones, principalmente en el establecimiento de una normativa en los nombres de las páginas XML dependiendo del tipo de escenario que contienen, y el establecimiento de distintos niveles de dificultad o visibilidad para los conceptos que forman el dominio.

Con respecto a los grados de dificultad, en principio sólo se han implementado tres grados de dificultad, que se han denominados alto, medio y bajo, y que van a dar lugar a que cada tema pueda tener hasta tres grados de accesibilidad o dificultad distintos por cada concepto o lección. Para referenciar a cada uno de estos grados se le han añadido al final del nombre de los ficheros XML un numero que indica el grado al que pertenece, quedando de la siguiente forma: 0 indica un grado bajo, un 1 indica un grado medio y un 2 indica un grado alto.

Como se ha comentado anteriormente, el nombre de los ficheros va a depender del tipo de escenario que contiene, del tema al que pertenece y del nivel de visibilidad, quedando de la siguiente forma:

- **Concepto.** Todas las páginas de contenido teórico (escenario tipo exposición), de todos los conceptos deben tener la nomenclatura *Concepto_Tema-Grado.xml*. Donde *Tema* es el nombre del tema al que pertenece el concepto y *Grado* es un número entero que indica el grado de dificultad del concepto. Por ejemplo, una página perteneciente a la introducción del tema ficheros con un grado alto, podría llamarse *introducción_ficheros-2.xml*.

- **Test Inicial**. Contiene 3 ficheros distintos por tema, uno por cada grado de dificultad y su nomenclatura es *Testi_Tema-Grado.xml***.** Siendo *Tema* el nombre del tema y *Grado* el grado de dificultad que tiene asignado. Por

ejemplo el test inicial para el tema de programas del grado bajo, se denotaría por *testi_programas-0.xml*.

- **Actividad**. Los ficheros con las actividades de un concepto tienen la misma nomenclatura que la del concepto a la que se refiere: *Concepto_Tema-Grado.xml*. ¿??

- **Test Final**. Los ficheros con los tests finales tienen la nomenclatura *testf_Tema-Grado.xml.* Siendo *Tema* el nombre del tema y *Grado* el grado de dificultad asignado a ese test final. Por ejemplo el test final para el tema de utilidades de grado medio, se denotaría por *testf_utilizades-1.xml.*

- **Evaluación.** Debe de existir 3 ficheros de evaluación por cada tema, uno para cada grado de dificultad y la nomenclatura usada para identificarlos es: *evaluación_Tema*-Grado.*xml.* En este archivo, el diseñador del curso incluirá todas las reglas necesarias para adaptar el contenido del tema, utilizando para ello referencias sobre los niveles de conocimiento obtenidos en los test iniciales, test finales y actividades como se explica más adelante en la implementación del Modelo de Adaptación.

Finalmente, para desarrollar el dominio completo para un curso, primero se debe crear un directorio principal con el nombre del curso dentro del directorio *www* del servidor web que va a contener nuestro sistema ASWE, dentro del cual se crearán los siguientes subdirectorios (ver Figura 3.13):

- **xml.** Contendrá las páginas *xml* que formen parte de algún modo del curso.

- **test.** Contendrá los test del curso, tanto iniciales, finales como actividades.

- **images**. Contendrá todas las imágenes del curso.

- **sounds.** Contendrá todos los sonidos del curso.

- **movies.** Contendrá todos los vídeos del curso.

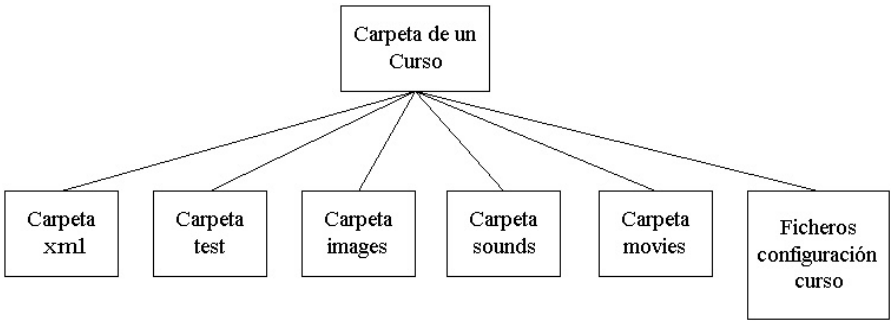

Figura 3.13: Contenido de la carpeta de un curso.

Además, deben crearse los siguientes ficheros de configuración del curso:

- **docs**. Contendrá una lista de todas las páginas *xml* que componen el curso.
- **properties**. Especifica cuales son los ficheros de configuración del curso.
- **header** y **footer**. Estos archivos contienen el texto que se introducirán en el encabezado y pie de página de todos los archivos de conceptos.
- **xmlreqlist**. Será el archivo que contendrá las relaciones que indican bajo qué condiciones qué páginas y conceptos serán aconsejables para el alumno.

3.3.2 Modelo del Alumno

El modelado del alumno se ha implementado utilizando ficheros XML y ficheros de texto plano tipo *log*s. El sistema AHA utiliza un fichero *log* que almacena todo el historial de navegación y un fichero *model* que almacena información del modelo del alumno.

A la implementación de AHA [De Bra & Calvi 1997] se le han modificado varias elementos para adaptarla a nuestro diseño. Por un lado se ha añadido un fichero plano denominado *test* que almacena los aciertos y fallos cometidos por los alumnos en cada una de las preguntas del curso tanto en actividades como en test iniciales y finales. Y por otro lado se ha añadido una serie de atributos nuevos al fichero *model*. Los atributos añadidos al sistema AHA son:

- **Marca de continuación**. Contiene el nombre del tema y el nivel de conocimiento que el alumno poseía en el momento de salir o desconectarse del curso.

- **Marca de Utilizado**. Contiene el nombre del escenario y un valor 0 ó 1, donde 0 indica que no ha sido utilizado por un alumno, y un 1 indica que ya se han mostrado en alguna sesión anterior o en la actual.

- **Acierto.** Contiene el nombre del escenario de tipo ejercicio y un valor que indica el acierto o fallo cometido por el alumno:

  Si. Indica que el alumno ha seleccionado la solución correcta del ejercicio.

  No. Indica que el alumno no ha seleccionado la solución correcta del ejercicio.

- **Nivel de conocimiento**. Variable cuya nomenclatura está compuesta por el Nombre de un concepto o tema, y por un valor entre 0 y 100 que indica el conocimiento que posee el alumno. Aunque en principio sólo puede tomar 4 valores que son experto, medio, principiante y conocimiento nulo. La correspondencia de valores será:

  0: significa que no conoce nada. Todavía no ha visitado nada.
  10: significa un nivel bajo de conocimiento o principiante
  50: significa un nivel medio de conocimiento.
  100: significa un nivel alto de conocimiento o experto.

- **Experiencia.** Nivel de estudios académicos que posee el alumno.

Esta información del modelo del alumno se va a almacenar en el sistema dentro de un directorio *logs* con el nombre de cada usuario y en los siguientes ficheros:

- **log.** Almacena el historial de navegación. Páginas visitadas y tiempo de visita.

- **test.** Almacena el acierto en los ejercicios. Nombre de la pregunta y acierto o fallo cometido.

- **model.** Almacena el resto de la información. Información del alumno, nivel de conocimiento, material utilizado, etc.

Toda esta la información almacenada para cada alumno puede ser de dos clases, dependiendo del tipo de información, por un lado están los referentes a elementos de tipo escenario (página web de contenido teórico o ejercicios) y por otro lado los elementos de tipo conocimiento (conceptos y temas).

- **Información sobre elementos tipo Escenario.** Los elementos tipo escenario son las distintas páginas Web que forman el curso y que pueden contener un tipo de escenario exposición o ejercicio. Por lo tanto, los posibles valores que se pueden almacenar en estos elementos son el tiempo de visualización y el acierto o fallo cometido en una pregunta. En concreto para el escenario tipo exposición sólo se almacena el tiempo y para el tipo ejercicio se almacena el tiempo y el acierto.

- **Información sobre elementos tipo Conocimiento.** Los elementos tipo conocimiento son las unidades de conocimiento del curso (conceptos como la unidad básica y temas). La información que se almacena en cada uno de ellos es el nivel de conocimiento asignado al alumno tras realizar una evaluación. En concreto para los conceptos, el nivel se asigna tras realizar una actividad (que está formada por un conjunto de ejercicios) y para los temas se asigna tras realizar un test inicial o un test final

3.3.3 Modelo de Adaptación

El modelo de adaptación se ha implementado mediante un conjunto de reglas que se consiguen partiendo del nivel de conocimiento obtenido por el alumno en los test y actividades, y estableciendo el grado de dificultad de los conceptos disponibles en cada momento de forma que, cuando el nivel de conocimiento del alumno es bajo, el grado de dificultad también deberá ser bajo; en cambio, si el nivel de conocimiento del alumno es alto, entonces el grado de dificultad asignado deberá ser alto.

El sistema AHA [De Bra et al. 2000] utiliza reglas de tipo Si-Entonces dentro de las propias páginas XML de contenido de los conceptos y además no trabaja con grados de dificultad o accesibilidad. Por ello se ha tenido que modificar dicho sistema para tener en cuenta nuestro diseño, creándose unos ficheros específicos donde se incluyen estas reglas, separando de forma clara el dominio del curso de la adaptación del curso.

Las reglas se han incluido dentro de ficheros XML, denominados ficheros de *evaluación* del tema. Estos ficheros se muestran al alumno cuando finalizan un tema y comunicándole el nivel de conocimiento obtenido en el test inicial, las distintas actividades de los conceptos del tema, test final y presentándole la

adaptación pertinente del sistema según los resultados obtenidos. También se puede acceder directamente a esta evaluación cuando se está realizando un tema en cualquier momento, para comprobar el estado del tema actual y el nivel de conocimiento. Estos ficheros de evaluación tienen formato XML, y debe de haber uno para cada grado de dificultad de todos los temas, por lo que se tienen tres ficheros de evaluación para cada tema. La nomenclatura usada para identificarlos es *evaluación_Tema-Grado.xml*. Dónde *Tema* contendrá el nombre del tema al que pertenece y *Grado* será un número correspondiente al grado de dificultad. Por ejemplo, la nomenclatura del fichero de evaluación de un tema denominado introducción para el grado bajo sería *evaluación_introducción-0.xml*. En este archivo de evaluación se incluirán las reglas necesarias para realizar la adaptación del tema. La estructura interna que tienen estos ficheros, es la siguiente:

- **Test Inicial**. Se establecen las reglas para determinar el grado de dificultad al que se le va a mostrar el tema al usuario dependiendo del nivel de conocimiento obtenido al realizar el test inicial. Algunos ejemplos son:

SI *nivel_test_inicial=10* ***ENTONCES*** *Grado Bajo para realización del tema.*
SI *nivel_test_inicial=50* ***ENTONCES*** *Grado Medio para realización del tema.*
SI *nivel_test_inicial=100* ***ENTONCES*** *Grado Alto para realización del tema.*
SI *nivel_test_inicial=0* ***ENTONCES*** *no has realizado el Test Inicial.*

- **Actividad**. Se establecen las reglas para cada uno de los conceptos que componen el tema, que asignan el nivel de conocimiento en cada uno de ellos dependiendo del valor obtenido en su realización. Algunos ejemplos son:

SI *concepto1=10* ***ENTONCES*** *Nivel Principiante en concepto1 y deberías repetir el concepto1.*
SI *concepto1=50* ***ENTONCES*** *Nivel Medio en concepto1 y podrías repetir el concepto1 o pasar al siguiente concepto.*
SI *concepto1=100* ***ENTONCES*** *Nivel Experto en concepto 1 y puedes pasar al siguiente concepto.*
SI *concepto1=0* ***ENTONCES*** *No has realizado la Actividad del concepto1 y debes realizarla.*

- **Test Final**. Se establecen las reglas para determinar el nivel de conocimiento final del usuario en un tema dependiendo del valor que se ha obtenido al realizar el test final. Algunos ejemplos son:

SI *valor_test_final=10* ***ENTONCES*** *Nivel Principiante final en el tema.*
SI *valor_test_final=50* ***ENTONCES*** *Nivel Medio final en el tema.*
SI *valor_test_final=100* ***ENTONCES*** *Nivel Experto final en el tema.*

SI *valor_test_final=0* ***ENTONCES*** *No has realizado el Test Final.*

- **Evaluación Final**. Este punto es quizá el más importante de toda la adaptación. Las reglas que se incluyen en este apartado serán diferentes dependiendo del grado de dificultad en el que nos encontremos. Estas reglas adaptarán los pasos siguientes a realizar por el usuario teniendo en cuenta todas las pruebas que ha realizado (tests iniciales, actividades y tests finales) y los resultados obtenidos. En función de todos ellos, se determina si se debe repetir el tema al mismo nivel, si se puede repetir a un nivel superior al que hemos realizado, o si se puede pasar al tema siguiente, etc. Por ejemplo el diseñador del curso puede determinar que solo si se supera el grado de dificultad alto de un tema se podrá determinar, de entre un rango de posibilidades, el tema siguiente. Algunos ejemplos son:

SI *valor_test_final>0 y concepto1>0 y concepto2>0 ...* ***ENTONCES***
COMIENZO
SI *grado_tema<2 Y valor_test_final>50* ***ENTONCES***
puedes pasar ir a un grado superior del mismo tema.
SI *grado_tema=2 Y valor_test_final>50* ***ENTONCES***
puedes pasar a otros temas.
SI *valor_test_final<50* ***ENTONCES***
debes repetir el tema actual al mismo grado.
FIN
SINO *Debes de realizar todas las actividades y el test final.*

3.3.4 Motor de Adaptación

El motor de adaptación se ha implementado mediante *servlets* que son aplicaciones Java que se invocan desde una página Web y que van a permitir personalizar las páginas que devuelve el servidor Web. El sistema AHA [De Bra et al. 2000] proporciona un conjunto de *servlet* que realizan un control de la adaptación general de cualquier tipo de aplicación. Pero para poder realizar la adaptación de la realización de actividades que se ha explicado en el diseño del ASWE, se ha tenido que modificar el sistema AHA añadiéndole un conjunto de nuevos *servlets* que permitan realizar la adaptación específica del ASWE propuesto.

En la Figura 3.14 se muestra el diagrama de clases UML de los nuevos *servlets* que se han desarrollado para realizar la adaptación de los cursos.

A continuación se muestra una breve descripción de cada uno de los *servlets*:

- **GenericServlet.** Modelo base de *servlet*, y del que todos los demás heredan. Se ha modificado para que permita la utilización de diferentes grados de visibilidad o dificultad. De manera que dependiendo del nivel de conocimiento que tenga asignado un alumno en un tema, sólo se le mostrarán las páginas de grado adecuadas para su nivel como indican las reglas de los ficheros de evaluación de cada tema.

- **DoInitialTest.** Es encargado de implementar el test inicial con adaptación descrito en el apartado de diseño del ASWE. De forma que es el encargado de ir seleccionando la siguiente pregunta a mostrar al usuario dependiendo del nivel actual y la respuesta a la anterior pregunta.

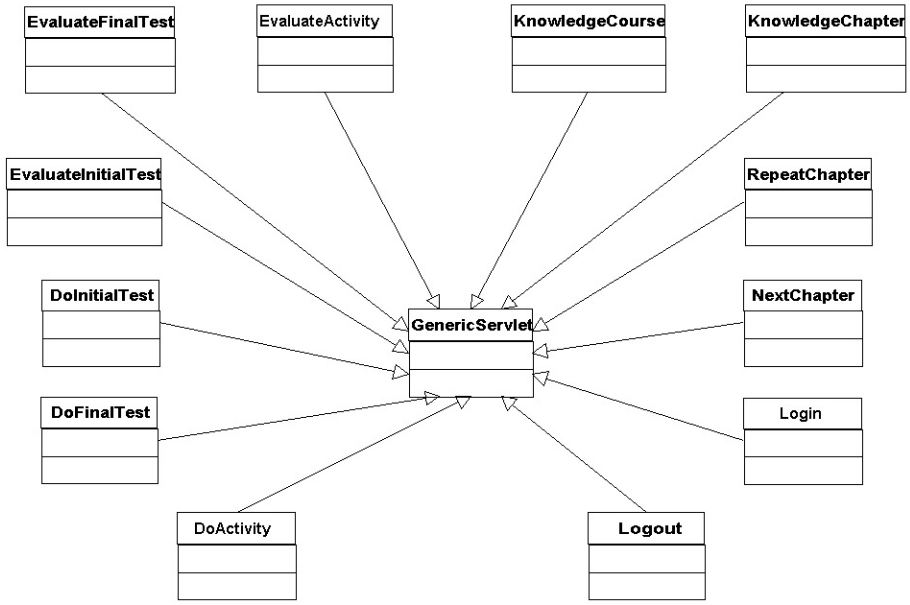

Figura 3.14: Diagrama de clases UML de los servlets del motor de adaptación.

- **EvaluateInitialTest.** Clase encargada de evaluar el test inicial y establecer el grado de accesibilidad del alumno de entrada en un tema dependiendo de las preguntas acertadas. El nivel de conocimiento inicial del alumno se determinará en función del número total de aciertos El diseñador del curso es el encargado de establecer este número para cada nivel, que por defecto, para cuatro preguntas, clasifica al alumno de la siguiente manera: nivel experto (las cuatro preguntas acertadas), nivel medio (tres o dos preguntas acertadas) y nivel principiante (una o ninguna pregunta acertada).

- **DoFinalTest.** Clase encargada de implementar el test final multipregunta de tipo clásico descrito en el apartado de diseño del ASWE. De forma que presenta al usuario un número determinado de preguntas con sus posibles respuestas para que seleccione las que crea correctas. El umbral mínimo de preguntas acertadas es de 3 para poder evaluar el test final.

- **EvaluateFinalTest.** Clase encargada de evaluar el test final y establecer el nivel final de conocimiento de un tema en función del número de preguntas acertadas en el test y recomienda al alumno repetir el tema o pasar a los siguientes. El número de preguntas que forman el test final es especificado por el diseñador del curso (por defecto seis preguntas), el cual también determina el número de aciertos para clasificar a un alumno, que por defecto es para nivel experto 6 preguntas acertadas, para nivel medio 5 preguntas acertadas y para nivel principiante menos de 5 preguntas acertadas.

- **DoActivity.** Clase encargada de implementar los ejercicios tipo actividad descrito en el apartado de diseño del ASWE. De forma que presenta una serie de preguntas de forma secuencial y controla si se puede pasar al siguiente o siguientes conceptos o se debe de reforzar el contenido del concepto actual. El umbral mínimo de preguntas acertadas es de dos para poder evaluar la actividad.

- **EvaluateActivity.** Clase encargada de evaluar los ejercicios de una actividad y establecer el nivel de conocimiento que posee el alumno en un concepto, dependiendo del número de aciertos en un grupo de preguntas. El número de preguntas de cada grupo que forman una actividad la establece el diseñador del curso, a partir de las cuales se establece el nivel del alumno. Por defecto el valor es 4, de forma que si el alumno acierta todas se le asigna un nivel experto, si acierta tres preguntas se le asigna un nivel medio y si acierta menos de tres preguntas se le asigna un nivel principiante. El umbral mínimo de preguntas acertadas es dos

- **KnowledgeCourse.** Clase encargada de visualizar el estado actual del curso. De forma que muestra al alumno un índice completo de todos los temas que compone el curso, indicando los temas que ya ha realizado, los que le falta por realizar y el nivel de conocimiento final que ha obtenido en los temas que ha finalizado de forma completa, es decir, en los que ha realizado el test final.

- **KnowledgeChapter.** Clase encargada de visualizar el estado actual de un tema. De forma que presenta el nivel de conocimiento que tiene un alumno en el tema actual en el que se encuentra. Para ello muestra una lista de los

test y actividades realizados del tema y los que faltan por realizar, y para los realizados el nivel de conocimiento obtenido. Además muestra sugerencias del tipo: "*debes de realizar la actividad o el test", "podrías volver a leer el concepto y hacer de nuevo la actividad para aumentar tu conocimiento", "puedes pasar al siguiente concepto", "puedes pasar a los siguientes temas", "debes de volver a realizar el tema para aumentar tu conocimiento"*, etc.

- **RepeatChapter.** Clase encargada de controlar la posibilidad de repetir el mismo tema a un mismo grado de dificultad o a otro distinto. Una posibilidad es que el alumno desee, tras finalizar un tema, repetir el mismo tema a un grado distinto de dificultad del que acaba de visualizar, por si desea repasar o reforzar información. Aunque lo normal es que sea el propio sistema el que obligue al alumno a repetir el tema con el mismo grado de dificultad si el nivel de conocimiento final obtenido ha sido bajo.

- **NextChapter.** Clase encargada de determinar a qué temas se pueden acceder desde un determinado tema. Dependiendo del nivel obtenido en el tema actual y en los temas anteriormente realizados, establece a que temas se puede acceder a continuación.

- **Login.** Clase encargada de identificar al alumno que entra en el curso y determinar el punto exacto de continuación o comienzo, es decir el tema, concepto y nivel, en el que lo dejó en la última sesión que realizó. En el caso de que sea la primera vez que accede al curso, por defecto, se le conduce directamente al test inicial del primer tema del curso.

- **Logout.** Clase encargada de desconectar al alumno del curso. De forma almacena el punto exacto, es decir el tema, concepto y nivel en el que se encontraba el alumno justo antes de salir del curso.

3.3.5 Módulo Interfaz

El módulo interfaz se genera de forma dinámica por el motor de adaptación, descrito en la sección anterior, y está formado por páginas Web de tipo XML y programas *servlets* de Java. De forma que, para cada usuario, se muestran las páginas XML personalizas, que contienen una serie de enlaces (link) que no son enlaces típicos sino que en realidad son *servlets* que van a controlar la interacción con el alumno y la adaptación.

El interfaz principal del curso está formado por una página Web que está dividida en tres partes (ver Figura 3.15). El color de fondo de todas las páginas

Web también indica el grado que tiene el tema en cada momento. Por ejemplo, los colores utilizados por defecto son el rosa, el azul y el amarillo, de forma que el color rosa indica un grado bajo, las de color azul indican un grado medio y las de color amarillo indican un grado alto.

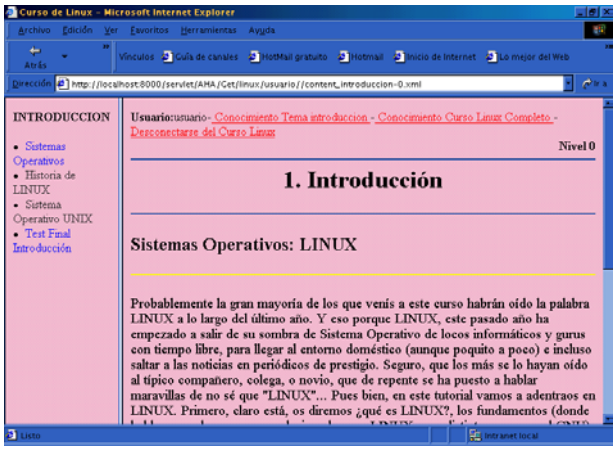

Figura 3.15: Página Web típica de un curso.

A continuación se va a describir más detalladamente cada una de las partes de esta página principal del curso (ver Figura 3.15) y sus significados:

- **Índice**. Muestra los conceptos y un enlace al test final del tema. Cada uno de estos conceptos son también enlaces, de manera que se puede navegar por el tema libremente y realizar la lectura de los conceptos en el orden que se desee establecer. Pero es obligatorio en cada tema realizar primero las actividades de todos los concepto y después el test final para poder pasar a otro nivel del mismo tema o a otro tema diferente.

- **Usuario:** Muestra el identificador del usuario que está realizando el curso y con el cuál se ha conectado.

- **Conocimiento Tema Actual**. Este enlace lleva a una página Web que muestra el conocimiento del tema que actualmente se está realizando. Por este motivo, en el ejemplo de la Figura 3.15, el enlace se muestra como *Conocimiento Tema introducción*. Si se pulsa sobre este enlace, se muestra el conocimiento que hasta ahora se tiene en función de las páginas leídas, las actividades realizadas y los resultados de los test (ver Figura 3.16).

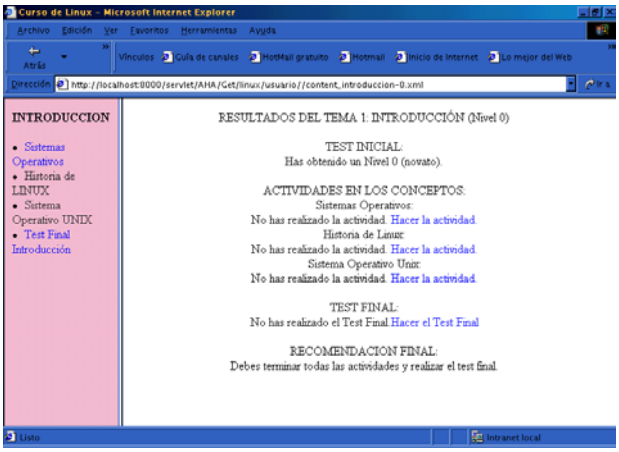

Figura 3.16: Página Web con la Información sobre el conocimiento del tema actual.

- **Conocimiento Curso Completo**. En este enlace lleva a una página Web con el índice completo de temas del curso que se está realizando, y los niveles que se han obtenido en cada uno de ellos.

- **Desconectarse del Curso**. Este enlace permite finalizar la sesión actual del alumno, de forma que lo desconecta el curso.

- **Nivel.** Indica, al igual que el color de fondo de las páginas Web, el grado de dificultad en el que se está mostrando actualmente el tema al alumno. Se indica mediante un número que significa, un valor 0 indica un grado bajo, un valor 1 indica un grado medio y un valor 2 indica un grado alto.

- **Contenido**. La parte central e inferior de la página lo ocupa los escenarios del concepto del tema en el que nos encontramos. Normalmente se muestra el escenario de tipo exposición y dentro de él aparecen distintos enlaces a los escenarios tipo ejercicio actividad o test final.

- **Actividades**. Dentro de la página Web de contenidos de exposición de cada concepto y normalmente al final, se encuentra un enlace a la actividad del concepto actual. Esta actividad es un escenario de tipo test actividad anteriormente descrito, y su interfaz es muy parecido al que se muestra en la Figura 3.17.

- **Test Final.** Es un escenario de tipo ejercicio test final que se muestra al pulsar sobre el enlace que aparece al final del índice del tema. La apariencia de un test final se muestra la Figura 3.17.

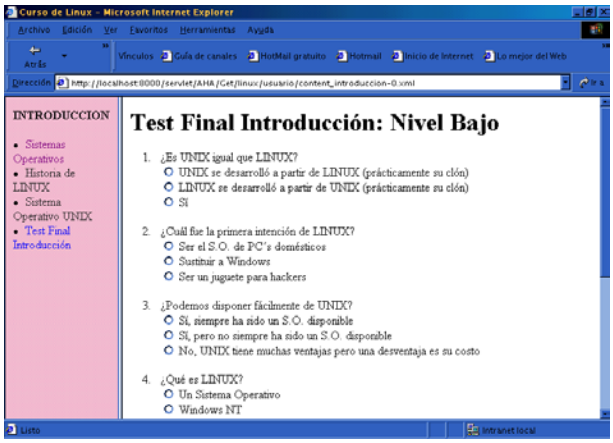

Figura 3.17: Interfaz de un ejercicio tipo test final.

- **Test Inicial.** Es un escenario de tipo test final que se nos muestra la primera vez que se entra a un tema y su interfaz es muy parecido al que se muestra en la Figura 3.17.

Capítulo 4
Descripción de la Información Inicial

En el tema anterior se han mostrado los aspectos relacionados con el diseño e implementación del sistema desarrollado. También se ha indicado que su implantación real como sistema para la enseñanza del sistema operativo Linux ha permitido obtener información de usuarios que se utilizará en la tarea de adquisición de conocimiento. Es momento ahora de describir el tipo de información que se ha generado como consecuencia de este uso e investigar sobre las distintas formas de extracción de conocimiento. Para finalizar se presentarán las tareas de preprocesado realizadas sobre la información disponible, antes de realizar dicha tarea de descubrimiento de conocimiento, que se describirá en el siguiente Tema.

CAPÍTULO 4: DESCRIPCIÓN DE LA INFORMACIÓN INICIAL

4.1 Introducción

El diseño y la construcción de un curso hipermedia adaptativo basado en web es una tarea ardua y laboriosa, debido a que hay tomar importantes decisiones sobre:

- Cuántos temas componen un curso y cuáles son los más apropiados.
- Cuál es el orden más adecuado entre los temas.
- Cuántos conceptos componen un tema y cuáles son los más apropiados.
- Cuál es el orden más adecuado entre conceptos.
- Qué contenidos teóricos van a presentarse para cada concepto.
- Cuántos grados de accesibilidad o dificultad debe tener un tema.
- Qué conceptos, dentro de un tema, se van a colocar en cada grado de accesibilidad o dificultad.
- Qué actividades se asocian a cada concepto.
- Qué preguntas se van a plantear para evaluar cada concepto.
- Cuáles de las anteriores preguntas para un concepto, se van a plantear en la preevaluación de los temas, o en la postevaluación de los temas, o en las propias actividades de evaluación del concepto.

Debido a la gran cantidad de decisiones que hay tomar cuando se construye un curso de tipo ASWE, es muy difícil establecer, a priori, la mejor estructura y el contenido más adecuado para un curso concreto. De hecho, es muy probable que la planificación que propongan dos autores para un mismo curso sea muy distinta, dependiendo de distintas consideraciones, muchas de ellas subjetivas. A esto hay que añadirle el problema de la subjetividad de tener que agrupar o clasificar todo el material del curso en grados de accesibilidad o dificultad para permitir una correcta adaptación de curso, siendo en este caso mayores aun las diferencias de opinión. Por todo ello, se deduce que no es nada fácil diseñar de forma correcta un curso de este tipo.

Partiendo de la idea de que un curso de tipo ASWE puede y debe mejorarse, y que la metodología propuesta pretende que los algoritmos de conocimiento proporcionen al autor del curso información orientada a su mejora, la tarea de extracción de conocimiento a realizar en este trabajo consistirá en el *descubrimiento de relaciones interesantes entre los distintos conceptos del curso, la secuenciación de dichos contenidos y la estructuración por niveles de los conceptos*. Dicha información se presentará en forma de reglas de predicción. Esta forma de representación es muy común en minería de datos debido a que representa el conocimiento descubierto con un alto nivel de abstracción, lo que las hace comprensibles para la mayoría de los usuarios. Además, este tipo de reglas se pueden utilizar directamente en un proceso de toma de decisiones.

Hay que indicar que la originalidad de este trabajo radica fundamentalmente en el tipo de conocimiento que se pretende descubrir, ya que, como se ha indicado en el Tema de Antecedentes, existen referencias sobre el uso de técnicas de minería de datos sobre los datos de utilización de cursos [Zaïne 2001], [Pahtl & Dave 2002], [Wang 2002], [Heift & Nicholson]. Si embargo, el objetivo que se han perseguido en estos trabajos ha sido descubrir asociaciones entre páginas visitadas, analizándose secuencias de páginas o agrupaciones de patrones de navegación, orientados a mejorar la estructura dinámica del curso. La propuesta presentada en esta Memoria pretende ser más ambiciosa, ya que se intentará descubrir relaciones de dependencia de los elementos (preguntas, contenidos teóricos, actividades y test) que contienen las distintas páginas Web (escenarios) de un curso: algunas pueden emplearse para modificar el comportamiento dinámico del curso pero, como ya se verá en el Tema *Información Descubierta*, también pueden influir sobre los propios contenidos que se han incluido en el curso, validando aspectos de éste como la calidad de las actividades planteadas, instante de presentación de un concepto, etc.

Antes de aplicar los algoritmos de minería de datos sobre la información disponible, es necesario llevar a cabo una recopilación de la información generada y un preprocesado de ésta, que la ponga en un formato apto para su utilización. El resto de este Tema se dedica a presentar dicha información, así como el preprocesado que se ha realizado sobre ella.

4.2 Descripción de la Información

Como ya se ha descrito en el tema anterior, se ha desarrollado un curso de Sistema Operativo Linux [Romero et al. 2002c] sobre nuestro sistema adaptativo para la educación basada en web. Para la construcción de este curso se ha utilizado la herramienta autor desarrollada específicamente para facilitar esta tarea al profesor y que se describen en el apéndice *Herramienta para la Construcción de Cursos*.

El curso sobre Linux está compuesto por 15 temas, cada uno de los cuáles presenta tres grados de dificultad, excepto los cinco últimos temas que sólo presentan un único grado. Debido al reducido número de conceptos que componen cada uno de los grados de dificultad de los temas, se ha optado por simplificar la estructura y no agruparlos en lecciones. El índice completo de los temas que forman el curso es el siguiente:

1. Introducción.
2. Instalación.
3. Inicio de una sesión Linux.
4. Trabajando con UNIX.
5. Programas para Linux.
6. El Sistema X-Windows.
7. Personalización del sistema.
8. Gestión de usuarios.
9. Gestión de ficheros.
10. Tareas administrativas básicas.
11. Telnet.
12. FTP.
13. Correo electrónico.
14. Redes TCP/IP.
15. Documentación.

Este curso ha sido ejecutado por 50 alumnos de distinta naturaleza, 40 de primer año de ciclo formativo de grado superior en Informática de Sistema, pertenecientes al I.E.S. "Gran Capitán" de Córdoba, que lo realizaron en horas de prácticas de la asignatura "Sistemas Operativos", y otros 10 pertenecientes a niveles de estudios superiores: alumnos de 2º curso de Ingeniería Informática de Sistemas de la Universidad de Córdoba y miembros del aula de Linux de esta universidad.

Para cada uno de estos alumnos, el ASWE ha generado un directorio con su *login* o identificador de acceso al curso, en el que va almacenando toda la información de utilización dentro de tres ficheros de texto denominados *logs*, *model* y *test*. En la siguiente sección se describirá qué información se ha almacenado en cada uno de los ficheros, así como el formato que presenta cada uno de ellos.

4.2.1 Fichero Log

El fichero *log* es un fichero texto que contiene información sobre los distintos escenarios visitados (que se corresponden con páginas XML del curso) y el instante de tiempo (en milisegundos) en el que el alumno ha accedido a ellos. La Figura 4.1 muestra un fragmento de uno de estos ficheros *log*. Como puede observarse, cada línea del fichero contiene una cadena compuesta de dos nombres separados por una barra inclinada (los cuáles se corresponden con el nombre del *servlet* que ha devuelto la página y el nombre de la página devuelta) y un valor numérico correspondiente con el instante de tiempo en que el sistema presentó la página al usuario. El nombre de *servlet* es necesario porque nos va a permitir

identificar la página como correspondiente a un escenario de exposición o a un escenario de tipo ejercicio.

```
DoInitialTest/testi_introduccion-1.xml 1012431922
EvaluateInitialTest/testi_introduccion-1.xml 1012431938
DoInitialTest/testi_introduccion-0.xml 1012432389
EvaluateInitialTest/testi_introduccion-0.xml 1012432393
DoInitialTest/testi_introduccion-1.xml 1012432428
EvaluateInitialTest/testi_introduccion-1.xml 1012432441
DoInitialTest/testi_introduccion-0.xml 1012432443
EvaluateInitialTest/testi_introduccion-0.xml 1012432457
...
...
DoActivity/caracte_introduccion-1.xml 1041503291
EvaluateActivity/caracte_introduccion-1.xml 1041503313
DoActivity/caracte_introduccion-1.xml 1041503318
EvaluateActivity/caracte_introduccion-1.xml 1041503374
Get/gnu_introduccion-1.xml 1041503382
DoActivity/gnu_introduccion-1.xml 1041503426
EvaluateActivity/gnu_introduccion-1.xml 1041503585
...
```

Figura 4.1. Fragmento de un fichero log correspondiente a un usuario del curso.

El tamaño de estos ficheros *log* es distinto para cada usuario, ya que depende del número de escenarios por los que haya tenido que pasar. Sin embargo, para los usuarios seleccionados el tamaño es de aproximadamente de unas 2000 líneas, correspondientes al número total de escenarios que suele examinar un usuario promedio.

4.2.2 Fichero Model

El fichero *model* es un fichero XML que contiene información general sobre el alumno y su nivel de conocimiento en los distintos conceptos que componen el curso. La Figura 4.2 muestra un fragmento típico de un fichero *model*. Se puede apreciar que la primera línea indica la versión de XML utilizada y la codificación de los caracteres. La segunda línea indica el fichero DTD (del inglés Document Type Definition) que define los componentes del fichero XML. Toda la información está dentro de un único componente denominado *usermodel,* formado por varios subcomponentes. El primer subcomponente denominado *person* contiene la información sobre el usuario entre las distintas etiquetas XML, donde el nombre de la etiqueta especifica la información que almacena la propia etiqueta (nombre, e-mail, identificador, clave, universidad, nombre del tema que actualmente está realizando, grado de accesibilidad actual del tema). El segundo componente denominado *knowledgelist* contiene información sobre los niveles de

conocimiento obtenidos por el usuario en los distintos test y actividades. Para cada página XML, que representa un concepto y para cada tema contiene un elemento *knowlegeitem* que contiene el nivel de conocimiento obtenido por el alumno en la actividad de evaluación del concepto o en los test de evaluación del tema. Este nivel está representado mediante un número entero comprendido entre 0 y 100.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE usermodel SYSTEM
  ../../../WWW/AHAstandard/usermodel.dtd'>
<usermodel>
  <person>
    <name>Cristobal Romero Morales</name>
    <email>cromero@uco.es</email>
    <id>cristobal</id>
    <passwd>rocio</passwd>
    <institute>Universidad de Córdoba</institute>
    <department>Licenciatura en Informática</department>
    <nivelinicial>medio</nivelinicial>
    <nombretemaactual>redes</nombretemaactual>
    <niveltemaactual>1</niveltemaactual>
  </person>
...
  <knowledgelist>
    <knowledgeitem>
      <knowledgename>desmon_unix-2</knowledgename>
      <knowledgeperc>100</knowledgeperc>
    </knowledgeitem>
    <knowledgeitem>
      <knowledgename>shell_unix-0</knowledgename>
      <knowledgeperc>100</knowledgeperc>
    </knowledgeitem>
...
```

Figura 4.2: Fragmento de un fichero model correspondiente a un usuario del curso.

El tamaño de estos ficheros *model* también es algo distinto para cada usuario, dependiendo del número de conceptos visitados. Sin embargo para los usuarios seleccionados el tamaño es de aproximadamente de unas 3000 líneas por usuario.

4.2.3 Fichero Test

El fichero *test* es un fichero texto que contiene información sobre el acierto o fallo cometido en las preguntas de los test iniciales y finales de cada tema, y en las actividades de cada concepto. La Figura 4.3 muestra parte de uno de estos ficheros. Se puede observar como cada pregunta se encuentra en una línea distinta, donde aparece primero el nombre de la página XML que contiene la pregunta (sin extensión .xml), separado por un espacio en blanco de su tipo (TestInicial,

Actividad o TestFinal), separado por otro espacio en blanco del número de la pregunta (identifica a la pregunta dentro del fichero que la contiene) y su acierto o fallo (S o N respectivamente).

```
testi_introduccion_1 TestInicial 2 N
testi_introduccion0 TestInicial 0 S
testi_introduccion1 TestInicial 0 N
testi_introduccion0 TestInicial 2 S
ssoo_introduccion-0 Actividad 0 S
historia_introduccion-0 Actividad 0 N
historia_introduccion-0 Actividad 1 S
unix_introduccion-0 Actividad 1 S
unix_introduccion-0 Actividad 0 S
...
...
testf_introduccion1 TestFinal 3 N
testf_introduccion1 TestFinal 1 S
testf_introduccion1 TestFinal 2 N
testf_introduccion1 TestFinal 0 N
testf_introduccion1 TestFinal 5 S
testf_introduccion1 TestFinal 4 N
...
```

Figura 4.3: Fragmento de un fichero .test correspondiente a un usuario del curso.

Al igual que en los casos anteriores, el tamaño de estos ficheros *test* también es algo distinto para cada usuario, dependiendo del número de escenarios de tipo ejercicio realizados. Sin embargo, para los usuarios seleccionados el tamaño es de aproximadamente de unas 1000 líneas, correspondientes al número total de preguntas que suele contestar un usuario promedio.

4.3 Preprocesado de la Información

El preprocesado de la información consiste en la realización de una serie de operaciones destinadas a adaptar los datos para una determinada labor [Kotásek 2000]. Esta es una de las etapas más importantes de la minería de datos ya que la tarea de descubrimiento de conocimiento va a depender de los datos de entrada utilizados.

Existen distintas tareas que pueden englobarse dentro de la categoría del preprocesamiento: *selección de atributos*, *limpieza de datos* para eliminar ruido y datos inconsistentes, *transformación de los datos* a una forma adecuada para la minería e *integración de los datos***.** A lo largo de esta sección se describirá cuál ha sido el preprocesado realizado sobre los datos obtenidos a partir de los alumnos del curso de Linux.

4.3.1 Selección de Atributos

El objetivo de esta tarea es seleccionar un subconjunto de atributos relevantes del conjunto de todos los disponibles [John et al. 1994], para su empleo en el algoritmo de minería de datos. Esta es una tarea muy importante, debido a que el espacio de búsqueda crece exponencialmente con el número de atributos y es crucial que los atributos predictores sean relevantes para predecir el atributo objetivo. Además, permite descubrir reglas con una mayor exactitud y simplicidad que las que se obtendrían utilizando el conjunto original de atributos.

La forma más común de seleccionar los atributos relevantes es la manual, basándose en el conocimiento del problema y en el significado de los atributos. Sin embargo, también existen métodos automáticos que permiten reducir la dimensionalidad de los datos seleccionando sólo un conjunto de atributos relevantes, como son los métodos de aproximación de filtro y de envoltura [Witten & Frank 2000].

En nuestro caso, se ha determinado de forma manual cuáles son los atributos principales que deben formar parte de las reglas de predicción y cuáles los que van a emplearse en el descubrimiento de dichas reglas. Los atributos seleccionados para formar parte de las reglas han sido:

- **Elemento.** Es el atributo principal, del que se pretenden buscar relaciones. Indica el nombre del escenario de tipo exposición o ejercicio, o el nombre de un concepto o tema del curso.

Debido a que existen varios tipos de elementos y cada elemento puede tener asociado tres valores importantes para nuestro problema (tiempo, acierto y nivel), se han seleccionado también como atributos principales aquellos que nos indican sus posibles valores.

- **Tiempo.** Indica el intervalo de tiempo empleado por un alumno en la visualización de una página de tipo exposición o ejercicio (actividad, test inicial o test final).

- **Acierto.** Indica el resultado (acierto o fallo) de un alumno al contestar una pregunta dentro de una actividad o test.

- **Nivel.** Indica el grado de conocimiento asignado en un concepto o tema y que ha sido obtenido por el alumno tras realizar un test inicial, final o actividad.

El resto de atributos seleccionados, aunque no van a aparecer directamente en la información descubierta en forma de reglas, permitirán restringir su búsqueda:

- **Alumno.** Es el nombre que identifica a cada alumno del curso en particular.

- **Repetición.** Es el número de repetición actual o número de veces que este alumno a visitado una misma página Web.

- **Tema.** Indica el nombre que identifica a cada tema del curso.

- **Tipo.** Indica de que tipo es una página Web. Se han distinguido cuatro tipos distintos: contenido teórico, actividad, test inicial y test final.

- **Dificultad.** Es el grado de comprensión que tiene asociado un concepto y los distintos escenarios que lo componen.

4.3.2 Limpieza de datos

La limpieza de los datos [Matelic & Marcus 1999] consiste en la búsqueda y eliminación de datos duplicados, erróneos, irrelevantes, etc. Dicho proceso está formado por tres fases: definir y determinar los tipos de errores, buscar e identificar las instancias que contienen errores y corregir los errores descubiertos.

Para realizar la búsqueda de datos erróneos se han utilizado una serie de estadísticos correspondientes a cada uno de los atributos anteriormente descritos, observando la distribución de cada variable, sus valores máximos, mínimos, etc. y se ha realizado una búsqueda de forma manual sobre los propios ficheros de datos, descubriéndose tres tipos de errores:

- **Tiempo elevado.** Se descubrió que el atributo tiempo podía tener valores demasiado altos, superiores incluso a 20 minutos. Esto podía ser debido a que el alumno había abandonado la aplicación, estaba haciendo otra cosa, habían ocurrido errores en la conexión, etc. Este tipo de anomalías se han corregido incorporando un algoritmo de transformación discreta de la variable tiempo, que considera como datos ruidosos los tiempos que exceden un valor máximo establecido y que asigna dicho valor máximo al dato considerado erróneo.

- **Duplicidad de datos.** Se descubrió que existía información duplicada sobre la realización del test inicial por parte de algunos usuarios. Los test iniciales son los únicos que no pueden ser repetidos por el usuario, por lo

que su duplicidad tenía que ser debida a algún error de ejecución del propio ASWE.[1] Para solucionar este tipo de problemas se eliminó toda la información de los test iniciales duplicados, dando por válida sólo la información de la última vez que se realizó el test.

- **Información incompleta.** Se descubrió que algunos alumnos no habían completado las actividades que comprenden el curso, ya que para esos usuarios sólo se disponía de información de utilización sobre algunos temas. En el caso en que fue posible, se contactó con los alumnos y se les solicitó que lo finalizaran, con el fin de poder utilizar su información. En los casos en que esto no fue posible, se desechó la información relativa a ese alumno.

Además de la detección de errores y duplicidad de datos, también se ha encontrado información irrelevante que se ha eliminado. Dentro del sistema, páginas que son necesarias para el funcionamiento de un curso pero irrelevantes desde el punto de vista de seguimiento o evaluación del curso en sí. Estas páginas son:

- **Páginas contenedoras.** Son páginas de tipo marco (frame) formadas por dos submarcos: uno en la parte derecha que contiene una página con el índice del tema y otro en la parte izquierda donde van apareciendo las páginas Web con los escenarios de los conceptos de cada tema.

- **Páginas de índice.** Son páginas que contienen el índice de cada tema.

- **Páginas de evaluación.** Son páginas que contienen la evaluación sobre el estado actual del tema.

- **Páginas de documentación.** Son páginas que contienen documentación de ayuda sobre el curso.

- **Páginas de desconexión.** Es la página de despedida que aparece cada vez que nos salimos del curso.

Dado que estas páginas no aportan ninguna información interesante que pueda utilizarse en la tarea que nos ocupa, se ha eliminado toda la información relacionada con ellas.

[1] Un escenario de error común podría ser: el servidor Web tarda en responder (o deja de responder) y el alumno abre una nueva conexión, entrando de nuevo al curso y volviendo a realizar el test inicial.

4.3.3 Construcción de Atributos

El objetivo de la construcción de atributos, también denominada construcción o inducción constructiva [Bloedorn & Michalski 1998], es la de generar nuevos atributos mediante la aplicación de operadores o funciones sobre los originales, de forma que los nuevos atributos faciliten el proceso de minería de datos. El mayor problema que presenta esta tarea es que el espacio de búsqueda tiende a ser muy grande, debido a la gran cantidad de atributos y operaciones candidatas posibles a aplicar.

Los métodos de construcción de atributos se pueden categorizar según varias dimensiones [Hu 1998]: La primera está relacionada con la distinción entre el *preprocesado* y el *interpolado*. En el preprocesado el método de construcción de atributos se utiliza como un preproceso del algoritmo de minería de datos. En el interpolado el método de construcción de atributos y el algoritmo de minería de datos se unen en un único algoritmo. El método de interpolado tiene la ventaja de que los atributos son optimizados para el algoritmo de minería de datos, pero tiene la desventaja de la pérdida de generalización. Otra dimensión concierne a la distinción entre métodos *dirigidos por hipótesis* y *dirigidos por datos*. Los primeros construyen nuevos atributos a partir de hipótesis generadas previamente, mientras que los segundos generan directamente los nuevos atributos aplicando varias funciones a los atributos originales y evaluando los nuevos atributos de acuerdo a una medida de su calidad.

No se ha utilizado ningún método específico de los anteriormente descritos para la construcción de atributos, sino que se han creado los nuevos atributos de forma manual a partir de otros atributos existentes. Durante este proceso también se han realizado modificaciones en algunos de los ya existentes. Los atributos que se han generado son:

- **Nombre.** Este atributo contiene información sobre el nombre del elemento. Contiene el nombre del concepto al que se refiere (si es una pregunta, actividad o contenido teórico) o una indicación de si es test final o inicial (si es un test final o inicial de un tema), el nombre del tema al que pertenece y el grado de dificultad que tiene asociado. Dicho atributo se ha obtenido a partir de la información existente en los ficheros *logs*, modificándole el nivel (transformándolo de un entero a una etiqueta) y eliminando si fuera necesario parte de información innecesaria contenida en la parte inicial de la cadena (como el nombre del *servlet* que la ha devuelto). Pero además es necesario añadirle más información para identificarla unívocamente cuando se trata de una pregunta, para ello se le añade entre paréntesis el número de la pregunta. Por lo finalmente tenemos dos formatos de elemento, dependiendo de si la página web se

refiere a una pregunta individual (ver Figura 4.4) o no (ver Figura 4.5). Como se observa además de variar en el formato, también van a variar en cuales de los atributos principales (tiempo, acierto y nivel) van a tener algún valor relacionado con dicho elemento. De forma que las páginas referidas a preguntas sólo van a tener relacionados valores para tiempo y acierto, mientras que las páginas referidas a contenidos teóricos y actividades sólo van a tener relacionados valores para tiempo y nivel respectivamente. Finalmente para los elementos de tipo test inicial y final tienen un formato distinto (ver Figura 4.6). Como es lógico este tipo de elementos sólo puede tener valores relacionados sobre niveles.

Concepto_Tema-Dificultad(Número)

Donde:

Concepto: Nombre del concepto al que se refiere la pregunta.
Tema: Nombre del tema al que pertenece el concepto.
Dificultad: Grado de accesibilidad que tiene asociado el concepto.
Número: Número entero que identifica a una pregunta dentro del conjunto de preguntas de un mismo concepto.

Figura 4.4: Formato de nombre de elemento referido a una pregunta.

Concepto_Tema-Dificultad

Donde:

Concepto: Nombre del concepto al que se refiere la actividad o la exposición.
Tema: Nombre del tema al que pertenece el concepto.
Dificultad: Grado de accesibilidad que tiene asociado el concepto.

Figura 4.5: Formato de nombre de elemento referido a un actividad o contenido teórico.

Test_Tema-Dificultad

Donde:

Test: Indica el tipo de test, es decir, inicial o final. En concreto se ha utilizado para diferenciarlos la cadena TESTI o TESTF respectivamente.
Tema: Nombre del tema al que pertenece el test.
Dificultad: Grado de accesibilidad que tiene asociado el test.

Figura 4.6: Formato de nombre de elemento referido a un test inicial o final.

- **Tema.** Este atributo indica el nombre del tema al que pertenece una página. Para obtenerlo se ha sustraído de la cadena del nombre de la página, que tiene el formato *Pagina_Tema–Dificultad*. Por lo que sólo hay que coger la cadena que hay entre los caracteres _ y -. Para el caso que nos ocupa, los posibles valores para este atributo son los identificadores correspondientes a los nombres de todos los temas del curso: INTRODUCCION, INSTALACION, INICIO, UNIX, PROGRAMAS, XWINDOWS, USUARIOS, GESTION, PERSONALIZACION, ADMINISTRACION, TELNET, FTP, CORREO y REDES.

- **Dificultad**. Este atributo indica el grado de dificultad que tiene asociada una página. Para obtenerla se ha sustraído de la cadena nombre de la página, de donde sólo hay que tomar el número que hay entre el carácter – y el fin de cadena. En el caso particular del curso de Linux puede tomar tres valores: 0, 1 y 2. Para clarificar su significado se ha transformado este número en una etiqueta, de forma que sus valores han quedado como:

    - **Baja.** Si tiene el valor 0.
    - **Media.** Si tiene el valor 1.
    - **Alta.** Si tiene el valor 2.

- **Tipo.** Este atributo nos va a indicar de que tipo es una página. A diferencia de los otros atributos, este atributo se ha creado totalmente nuevo, dependiendo del nombre de la página web y del *servlet* que la ha devuelto. Los valores que puede tomar y su significado son:

    - **Contenido.** Se refiere a una página web de contenido teórico o escenario tipo exposición.
    - **Actividad.** Se refiere a una pregunta tipo ejercicio de un concepto, o la actividad en conjunto.
    - **TestInicial.** Se refiere a una pregunta de un test inicial, o al test inicial en conjunto.
    - **TestFinal.** Se refiere a una pregunta de un test final, o al test final en conjunto.

4.3.4 Transformación de Variables Continuas a discretas

Aunque la mayoría de los algoritmos de búsqueda de reglas pueden manejar atributos numéricos, la transformación discreta de los datos [Hussain et al. 1999] tiene la ventaja de que mejora la comprensibilidad de las reglas descubiertas al transformar los datos de bajo nivel en datos de alto nivel y también reduce significativamente el tiempo de ejecución del algoritmo de búsqueda. Su principal

desventaja es que puede reducir la exactitud del conocimiento descubierto, debido a que puede causar la pérdida de alguna información.

La transformación a variable discreta se puede ver como una categorización de los atributos que toma un conjunto pequeño de valores. La idea básica [Hussain et al. 1999] consiste en particionar los valores de los atributos continuos dentro de una lista pequeña de intervalos. Cada intervalo resultante es una estimación de un valor discreto del atributo.

El proceso de transformación discreta típico consta de cuatro pasos:

1. Ordenación de los valores continuos. Utilizar un algoritmo de ordenación para ordenar los valores en orden ascendente o descendente.

2. Evaluación de un punto de corte para dividir o evaluar intervalos adyacentes para unir. Existen multitud de funciones de evaluación como la medida de entropía y medidas estadísticas.

3. Corte o unión de los intervalos del valor continuo.

4. Finalización del proceso según algún criterio de parada.

Existen diferentes métodos de transformación de variable continua a discreta, que se pueden agrupar según distintas aproximaciones [Dougherty et al. 1995]:

- **Métodos locales.** Realizan la transformación discreta en una región del espacio de las instancias, por ejemplo, utilizando un subconjunto de las instancias.

- **Métodos globales.** Utilizan el espacio entero de las instancias.

- **Métodos supervisados.** Utilizan la información de la clase (valor del atributo objetivo) de la instancia de los datos cuando discretizan un atributo.

- **Métodos no supervisados.** No requieren la información de la clase para poder pasar a valores discretos . Sólo utilizan la distribución de los valores del atributo continuo como fuente de información.

En la siguiente tabla (ver Tabla 4.1) se muestra una lista de métodos de transformación discreta clasificados según la aproximación que siguen:

	Globales	**Locales**
Supervisado	1RD Cuantificadores Adaptativos Margen de Chi D-2 MCC Máximo Valor Predictivo	Cuantificación de Vector Entropía de Máxima Jerarquía Fallad e Irani C4.5
No Supervisado	Intervalo de igual longitud Intervalo de igual frecuencia	Agrupamiento de k-medidas

Tabla 4.1: Métodos de transformación discreta.

En el caso de los datos de utilización del curso de Linux, no disponemos de información de la clase, ya que cualquier atributo puede ser objetivo. Por lo tanto se tiene que utilizar un método de transformación discreta supervisado. Además el único atributo que va a necesitar una transformación discreta es el atributo tiempo. De forma que se desea pasar el tiempo que es una variable continua a una variable discreta, en nuestro caso, a una etiqueta. Previo a la transformación discreta, se ha tenido que realizar un cálculo del propio valor, ya que el sistema lo que en realidad almacena para cada página visitada es el instante de tiempo exacto en el que el usuario accede a dicha página expresado como la diferencia entre el tiempo del reloj del ordenador y el tiempo universal coordinado (Universal Time Coordinated, UTC) que es el tiempo en milisegundos transcurridos desde el 1 de enero de 1970 a las 00:00:00. Por lo tanto, para conocer el tiempo o intervalo de visita, es decir, el tiempo que ha estado el alumno visualizando una página, habría que restar el valor del instante de tiempo en que accede a la siguiente página menos el valor en el que accedió a la actual. De esta forma obtenemos un valor de tiempo de visita en milisegundos que es el que queremos pasar a valores discretos.

Se van a utilizar sólo 3 posibles valores (ALTO, MEDIO y BAJO) en la etiqueta del atributo tiempo, para facilitar la comprensibilidad de las reglas resultantes y por similitud con las variables dificultad y nivel que también tienen 3 valores con un significado semántico muy parecido.

Se han implementado dos métodos de transformación discreta no supervisados, ya que cualquier atributo puede ser objetivo. En concreto, los métodos implementados son: el método de igual anchura, el método de igual frecuencia, y un método manual donde el usuario establecen manualmente los límites de las categorías.

- **Método de igual en anchura.** Donde se debe especificar el número de divisiones que se quiere realizar, en nuestro caso tres, y entonces el rango continuo de los valores de un atributo es dividido uniformemente en

intervalos que tienen igual anchura y cada intervalo representa una división.

- **Método de igual en frecuencia.** Donde se especifica el número de divisiones que se quiere realizar, en nuestro caso tres, y entonces el rango continuo de los valores de un atributo es dividido uniformemente en intervalos que tienen igual número de valores continuos y cada intervalo representa una división.

- **Método manual.** Donde el usuario específica directamente cuales son los puntos de corte para realizar las divisiones. En nuestro caso que tenemos tres divisiones, el usuario debe especificar dos puntos de corte.

A los métodos de transformación discreta de igual anchura y de igual frecuencia, se le ha añadido la posibilidad de eliminar valores ruidosos que superen un umbral máximo establecido. Estos métodos que comienzan estableciendo un valor mínimo, dato con menor valor normalmente 0 y un valor máximo, dato con mayor valor. La idea consiste en utilizar en lugar del valor máximo utilizar un nuevo valor umbral especificado, de manera que cualquier valor que lo sobrepase se considera anormal y se le asigna el valor umbral.

De entre los tres métodos anteriormente descritos, se ha utilizado el método manual para pasar a valores discretos los datos sobre los tiempos de visita de las páginas del curso de Linux [Romero et al. 2002c]. Esta elección se ha realizado debido a que este método es más preciso que los otros dos si se dispone de información sobre cuales son los mejores puntos de corte. En nuestro caso, tras analizar los datos de los tiempos de los alumnos y teniendo en cuenta algunos estudios sobre los tiempos estimados de lectura de una página web [Sullivan 1997], se ha establecido un valor mínimo de 1 minuto y un valor máximo de 3 minutos. Significando que los tiempos inferiores a 1 minuto serán etiquetados con un valor de tiempo BAJO, los tiempos entre 1 minuto y 3 minutos serán etiquetados con un valor MEDIO y los tiempos superiores a 3 minutos serán etiquetados con un valor ALTO.

Además del atributo tiempo, también se han utilizado otros atributos que aunque no necesitan de un método de transformación discreta, sí van a ser etiquetados, es decir, se le va a asignar una etiqueta a cada posible valor numérico discreto. Estos atributos son dificultad y nivel.

- **Dificultad.** Como ya se ha descrito, el grado de dificultad es el que tiene asignado cada concepto y todas las páginas que contienen sus escenarios. Es un valor entero que en nuestro ejemplo concreto del curso de Linux puede tomar los valores 0, 1 y 2. Se le han asignado una etiqueta para que

sea más descriptivo su significado, quedando de la siguiente forma: BAJA, MEDIA y ALTA respectivamente. Este grado de dificultad de cada concepto es asignado por el profesor o autor del curso en la fase de construcción del curso.

- **Nivel.** El sistema asigna automáticamente el valor del nivel de conocimiento obtenido por el alumno en un test o actividad en función del número de respuestas correctas con respecto al número total de preguntas que componen el test o actividad. Para ello asigna un valor entero entre 0 y 100, que en principio para el caso particular del curso de Linux sólo puede tomar los valores específicos 0, 10, 50 y 100. Además el valor 0 no se ha tenido en cuenta, ya que sólo se utiliza para inicializar todas las páginas a no visitadas, ya que después nunca aparece en los ficheros *logs* al haber realizado todos los alumnos el curso completo, por lo que todos los datos tienen un valor distinto a 0. Cada uno de estos valores tiene un significado que es justo el nombre de la etiqueta que le vamos a poner.

    - **Principiante.** Si tenía el valor 10.
    - **Medio.** Si tenía el valor 50.
    - **Experto.** Si tenía el valor 100.

4.3.5 Integración de los datos

Normalmente cuando se trabaja en un problema de minería de datos es necesario primero formar un único conjunto con todos los datos que provienen de distintas fuentes, en nuestro caso todos los ficheros de utilización de cada uno de los usuarios el curso. Esta idea en el ámbito de integración de bases de datos empresariales se conoce con el nombre de almacenes de datos o "datawarehouse" [Kimball 1996] y proporcionan un punto único y consistente de acceso a los datos corporativos sin importar la división departamental. De hecho, los almacenes de datos han sido precursores de la minería de datos. Además de utilizar bases de datos, la integración también se puede realizar en ficheros planos, existiendo distintos algoritmos de minería de datos para cada tipo de almacenamiento:

- **Almacenamiento en base de datos.** Son aquellos algoritmos que están basados en sentencias SQL de consulta y modificación de una base de datos.

- **Almacenamiento en ficheros planos.** Son aquellos algoritmos que emplean sofisticadas estructuras de datos en memoria y los datos son almacenados y recuperados de ficheros planos.

De las dos opciones anteriores, se ha elegido utilizar un almacenamiento en una base de datos relacional, habiéndose realizado una trasferencia de información desde los ficheros de utilización de cada alumno (*log*, *model* y *test*) hacia tablas de la base de datos. Esta elección está motiva por la necesidad de facilitar y aumentar la velocidad del algoritmo de descubrimiento de reglas [Trueblood & Lovett 2001], dada la gran cantidad de información disponible. En principio se ha realizado de forma que se pueda utilizar cualquier sistema de base de datos relacional que permita el lenguaje SQL92. En concreto para la realización de todas las pruebas se ha utilizado MySQL [Dubois 2002] debido a su portabilidad, velocidad y libre disposición. La Figura 4.7 muestra el diagrama conceptual de la base de datos utilizada. Para la representación de dicho modelo se ha utilizado el modelo Entidad-Interrelación Extendido (EE-R) con notación P. Chen.

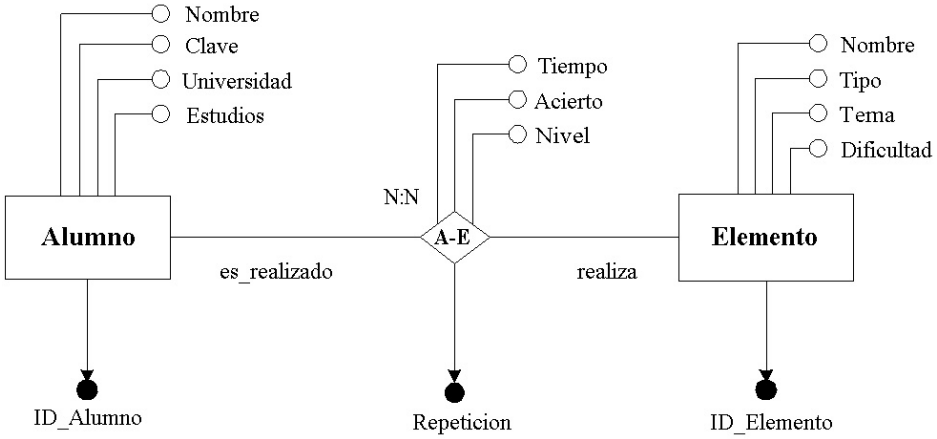

Figura 4.7: Diagrama Conceptual de la base de datos utilizada.

En el diagrama conceptual de la Figura 4.7 aparecen los siguientes tipos de entidades, que son los objetos que intervienen en el sistema desde el punto de vista de la información que se ha de tener en cuenta:

- **Tipo de Entidad Alumno.** Mediante este tipo de entidad representamos a las personas que han finalizado el curso adaptativo hipermedia, y han sido seleccionados por el usuario para analizar sus resultados en el desarrollo del curso. En este tipo de entidad se almacena toda la información genérica de relevancia que se puede tener sobre el alumno. Para este tipo de entidad se consideran los atributos: *ID_Alumno*, *Nombre*, *Clave*, *Universidad* y *Estudios*. El atributo *ID_Alumno* es el identificador principal de este tipo de entidad, ya que no pueden existir dos alumnos con el mismo identificador, y permite designar de forma unívoca a cada alumno. El atributo *Nombre* representa el nombre de usuario que el

alumno ha usado para identificarse en el curso adaptativo hipermedia, que junto al atributo *Clave* permiten el acceso a dicho curso. El atributo *Universidad* identifica el nombre de la universidad en la que están o han estudiado. El atributo *Estudios* determina los estudios que posee el alumno.

- **Tipo de Entidad Elemento.** Este tipo de entidad representa la información referente a los distintos elementos que componen el curso. Así, este tipo de entidad, almacena información sobre el nombre del elemento, el tipo de elemento, el tema al que pertenece, etc. Para este tipo de entidad se consideran los siguientes atributos: *ID_Elemento*, *Nombre*, *Tipo*, *Tema* y *Dificultad*. El atributo *ID_Elemento* identifica sin ambigüedad al elemento, por lo que es el identificador principal para este tipo de entidad. El atributo *Nombre* representa el nombre del elemento dentro del curso, el atributo *Tipo* representa el tipo de elemento (pregunta, actividad, contenido, test inicial y test final), el atributo *Tema* indica en que tema se encuentra y el atributo *Dificultad* indica el grado de dificultad que tiene asociada dicho elemento.

Los tipos de interrelaciones que existen entre los objetos que intervienen en el sistema desde el punto de la información que se ha de tener en cuenta, son los siguientes:

- **Tipo de Interrelación Alumno/Elemento (A-E).** Representa la relación entre los tipos de entidad Alumno y Elemento, el tipo de interrelación es del tipo N:N,. Se considera que un Alumno que ha finalizado el curso, ha realizado un elemento determinado una o más veces, ya que si no lo realiza no podrá finalizar el curso, por lo tanto el tipo de entidad Alumno participa con una cardinalidad de (1,n). De igual forma, un mismo elemento puede ser realizado por uno o más alumnos, ya que el sistema sólo captura información sobre aquellos elementos que han sido realizados por algún alumno, por lo tanto el tipo de entidad Elemento participa con una cardinalidad de (1,n). Este tipo de interrelación tiene cuatro atributos: *Repetición*, *Tiempo*, *Acierto* y *Nivel*. El atributo *Repetición* indica el número repetición del elemento por cada alumno, es decir, al alumno podría repetir la realización de los elementos, por lo que se tiene que guardar información de cada una de las repeticiones. Este atributo se ha designado como identificador principal de la interrelación ya que identifica de forma unívoca las diferentes repeticiones del alumno en cada elemento. El atributo *Tiempo* indica el tiempo tardado en realizar un elemento tipo contenido o pregunta. El atributo *Acierto* indica si se ha respondido bien o no a un elemento de tipo pregunta. El atributo *Nivel*

indican el nivel obtenido por el alumno tras realizar un elemento tipo test o actividad.

La descripción lógica de los datos se realiza mediante el proceso de traducción del esquema conceptual, que se acaba de describir, al esquema relacional. Para ello se utilizan una serie de reglas de transformación de un esquema conceptual a un esquema relacional. En la Figura 4.8 se muestra el esquema relacional de la base de datos utilizada.

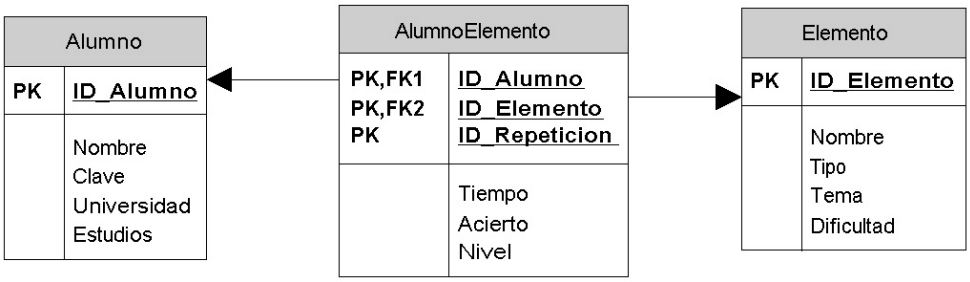

Figura 4.8: Esquema Relacional.

El esquema relacional (ver Figura 4.8) está formado por las siguientes tablas, donde las claves principales se indicarán subrayadas (clave principal) y las claves foráneas se indicarán en negrita (**clave foránea**):

Alumno(ID_Alumno, Nombre, Clave, Universidad, Estudios)

Elemento(ID_Elemento, Nombre, Tipo, Tema, Dificultad)

AlumnoElemento(**ID_Alumno**, **ID_Elemento**, Repetición, Tiempo, Acierto, Nivel)

Como se puede apreciar, todas las relaciones se encuentran normalizadas en FNBC, por lo que no es necesario realizar ninguna operación de normalización sobre las tablas definidas para el problema.

Finalmente se va a realizar la descripción física implica la definición de los atributos o propiedades que constituyen cada uno de los tipos de entidad, especificando el tipo de dato de que se trata y el dominio en el que se encuentra definido.

Tabla Alumno. Esta tabla se forma a partir del tipo de entidad Alumno y se ha creado a partir de los datos del fichero *model* de todos los alumnos del curso, seleccionados por el usuario. La tabla Alumno tiene los siguientes campos:

- **ID_Alumno**. Es el identificador principal de la tabla, y representa sin ambigüedad, mediante un número entero, a un determinado alumno de la tabla.

- **Alumno**. Es una cadena de caracteres que indica el nombre de usuario que utilizó el alumno durante la ejecución del curso.

- **Clave**. Es una cadena de caracteres que indica la clave del usuario que utilizó el alumno durante la ejecución del curso.

- **Universidad.** Es una cadena de caracteres que indica la universidad a la que pertenece el alumno. Sus posibles valores son: Universidad de Córdoba, Universidad Andaluza, Universidad Española, Universidad Europea, Universidad no Europea, No pertenece a ninguna universidad.

- **Estudios**. Es una cadena de caracteres que indica los estudios que ha desarrollado el alumno. Sus posibles valores son: 7º-8º EGB o 1º-2º ESO,1º-2º BUP o 3º-4º ESO, COU o 2º Bachiller, Ciclo Formativo Nivel Medio, Ciclo Formativo Nivel Superior, Plan Garantía Social, Diplomatura o Ing. Téc. en Informática, Diplomatura o Ing. Téc. en alguna Ingeniería, Diplomatura o Ing. Téc. en otras titulaciones, Licenciatura o Ing. en Informática, Licenciatura o Ing. en alguna Ingeniería, Licenciatura o Ing. en otras titulaciones, Otra titulación superior (doctor,etc.).

Tabla Elemento. Esta tabla se forma a partir del tipo de entidad Elemento y se crea a partir del fichero *logs* de todos los alumnos seleccionados por el usuario. Tiene los siguientes campos:

- **ID_Elemento**. Es el identificador de la tabla, y representa, mediante un número entero, sin ambigüedad a un elemento de la tabla.

- **Nombre.** Es una cadena de caracteres que indica el nombre del elemento.

- **Tipo**. Es una cadena de caracteres que indica el tipo del elemento. Se puede referir a un elemento de tipo contenido teórico, a una actividad o pregunta de una actividad, a un test inicial o pregunta de un test inicial y a un test final o pregunta de un test final. De

forma que los posibles valores pueden ser CONTENIDO, ACTIVIDAD, TESTINICIAL y TESTFINAL respectivamente.

- **Dificultad**. Es una cadena de caracteres que indica el grado de accesibilidad del elemento. Los posibles valores son: ALTA, MEDIA, BAJA.

- **Tema**. Es una cadena de caracteres que indica el nombre del tema al que pertenece el elemento. Los posibles valores son: INTRODUCCION, INSTALACION, INICIO, UNIX, PROGRAMAS, XWINDOWS, USUARIOS, GESTION, PERSONALIZACION, ADMINISTRACION, TELNET, FTP, CORREO y REDES.

Tabla AlumnoElemento. Esta tabla se forma a partir del tipo de interrelación (A-E) que existe entre los tipos de entidad Alumno y Elemento y se obtiene a partir de los ficheros *logs*, *model* y *test* de todos los alumnos seleccionados por el usuario. Tiene los siguientes campos:

- **ID_Alumno**. Es un número entero que sirve como identificador de la tabla, representando al alumno que ha realizado este elemento.

- **ID_Elemento**. Es un número entero que sirve como identificador de la tabla, representando el elemento que ha realizado el alumno.

- **Repetición**. Es un número entero que sirve como identificador de la tabla, representando el número de veces que ha repetido el alumno el elemento.

- **Tiempo**. Es una cadena de caracteres que indica el tiempo que el alumno ha estado realizando el elemento. Sus posibles valores son: ALTO, MEDIO, BAJO.

- **Acierto**. Es una cadena de caracteres que indica el acierto o fallo de un elemento tipo pregunta realizada por el alumno. Sus posibles valores son: SI y NO.

- **Nivel**. Es una cadena de caracteres que indica el nivel del alumno obtenido en un elemento de tipo test o actividad. Sus posibles valores son: PRINCIPIANTE, MEDIO, EXPERTO.

Capítulo 5

Algoritmos Evolutivos para el Descubrimiento de Reglas de Predicción

En este capítulo se analiza la capacidad de los Algoritmos Evolutivos de para resolver la tarea de descubrimiento de reglas de predicción interesantes planteada, comparando sus resultados con los que se obtienen mediante una serie de algoritmos clásicos que se han empleado clásicamente para esta tarea. En primer lugar se presentará y se justificará la elección del paradigma de computación evolutiva empleado. A continuación se describirá el esquema de codificación, así como los aspectos más relevantes que conciernen a la dinámica de la evolución simulada. Tras una descripción de las pruebas realizadas, se finalizará el Capítulo comentando los resultados obtenidos y presentando las conclusiones.

CAPÍTULO 5: ALGORITMOS EVOLUTIVOS PARA EL DESCUBRIMIENTO DE REGLAS DE PREDICCIÓN

5.1 Introducción

En el capítulo anterior se ha presentado la información que se genera como consecuencia de la interacción de los usuarios con el ASWE desarrollado, y se ha indicado que el conocimiento que se considera interesante de cara a una posible modificación del sistema (en concreto, del modelo del dominio) es el que establece relaciones entre los atributos de tiempo invertido en la realización de actividades, nivel asignado al alumno en cada concepto y grado de acierto/fallo en las actividades realizadas.

También se ha indicado que el problema se va a abordar desde el punto de vista del descubrimiento de reglas de predicción. Este es el enfoque que resulta más interesante, dada la alta comprensibilidad de las reglas resultantes y a que permite que el profesor aporte su experiencia e intereses al proceso de descubrimiento, indicando sobre qué atributos concretos desea que se realice la predicción.

Aunque esta tarea se ha abordado desde distintos paradigmas de cómputo, los resultados más prometedores que se han reportado en los últimos años corresponden a los métodos basados en el uso de Algoritmos Evolutivos [Freitas 2002]. Hay muchas razones que apoyan este hecho, siendo la más sobresaliente que tienen en cuenta la interacción entre atributos (lo cual es consecuencia de la búsqueda global que realizan). El propósito de este trabajo es comprobar la eficiencia de este tipo de algoritmos para resolver el problema planteado, y comparar los resultados que producen con respecto a los Algoritmos Clásicos.

5.2 Programación Genética Basada en Gramáticas

La Programación Genética basada en gramáticas (Grammar Based Genetic Programming, GBGP) [Whigham 1995] es un paradigma de programación genética en el que los individuos vienen representados como árboles de derivación de una gramática definida por el usuario para especificar el espacio de soluciones al problema. Se ha elegido este paradigma por la expresividad que presenta, que va a facilitar enormemente la interacción con el usuario. Como se verá posteriormente, se puede construir una gramática que representa todas las reglas extraíbles a partir del conjunto de datos y, tras interactuar con el usuario, restringir dicha gramática para que sólo se generen las reglas solicitadas.

5.2.1 Representación de los individuos

La Figura 5.1 representa la gramática empleada para representar las reglas que van a ser buscadas por el algoritmo evolutivo. Como puede comprobarse, se trata de reglas de predicción, en las que el antecedente puede presentar una o más condiciones y el consecuente presenta una única condición. Cada una de las posibles condiciones relacionan un atributo de la tabla con uno de los valores posibles que presenta el atributo. Por razones de presentación[1], no se han incluido los símbolos terminales que representan las posibles etiquetas para cada uno de los atributos de la tabla (atributos relacionados con aciertos, tiempo y nivel).

<regla>	::=	"**SI**" <antecedente> "**ENTONCES**" <consecuente>
<antecedente>	::=	<antecedente> "**Y**" <condición> | <condición>
<consecuente>	::=	<condición>
<condición>	::=	<atributonivel> "**=**" <valornivel> |
		<atributotiempo> "**=**" <valortiempo> |
		<atributoacierto> "**=**" <valoracierto>
<atributonivel>	::=	"**TIEMPO.**"Nombre de atributo nivel válido
<atributoacierto>	::=	"**ACIERTO.**"Nombre de atributo acierto válido
<atributotiempo>	::=	"**NIVEL.**"Nombre de atributo tiempo válido
<valornivel>	::=	"**PRINCIPIANTE**" | "**MEDIO**" | "**EXPERTO**"
<valoracierto>	::=	"**SI**" | "**NO**"
<valortiempo>	::=	"**ALTO**" | "**MEDIO**" | "**BAJO**"

Figura 5.1: Gramática de reglas de predicción en formato BNF.

La Figura 5.2 presenta el árbol de derivación correspondiente a la regla:

SI ((ACIERTO.CARACTE_INTRODUCCION-ALTA(2) = NO) **Y**
(TIEMPO.CARACTE_INTRODUCCION-ALTA(2) = ALTO))
ENTONCES
NIVEL.CARACTE_INTRODUCCION-ALTA = EXPERTO

que está indicando que los alumnos que han sido finalmente evaluados como expertos en el concepto "Características del sistema Linux", no aciertan la pregunta número dos dentro de la actividad de evaluación de dicho concepto (que presenta un grado de dificultad elevado dentro del tema Introducción) y que además, invierten un tiempo elevado en la realización de esta pregunta.

[1] Hay que tener en cuenta que, para el problema que se ha resuelto, existen 534 atributos tipo nivel, 1474 atributos tipo acierto y 1648 atributos de tipo tiempo.

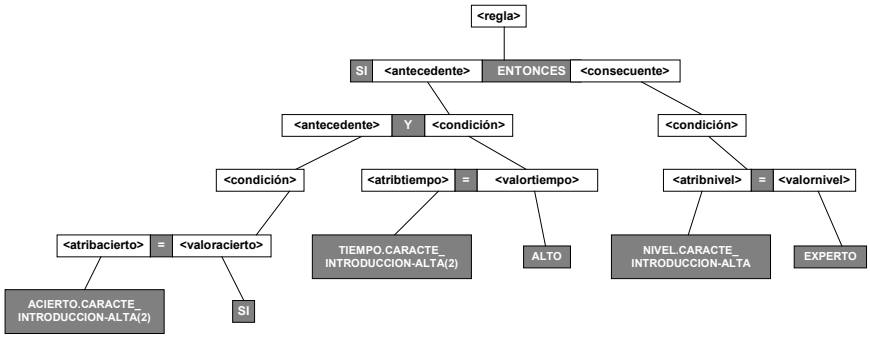

Figura 5.2: Árbol de derivación para una regla (ver explicación en el texto).

Como puede comprobarse, los nodos internos del árbol representan a símbolos no terminales de la gramática (en color blanco), mientras que los nodos hoja (en color gris) representan a los terminales. Para generar una regla a partir de este árbol, basta con recorrer el árbol y guardar los nodos hoja en la estructura que contenga la regla en cuestión.

5.2.2 Inicialización de individuos

La generación aleatoria de un individuo en GBGP es relativamente simple. La raíz del árbol coincide con el símbolo terminal de la gramática. El algoritmo de creación expande el árbol hasta que no quedan símbolos no terminales sin expandir. Para ello, se comienza eligiendo al azar una de las producciones del símbolo de inicio, con lo que la raíz del árbol queda expandida. Se siguen expandiendo todos los símbolos no terminales eligiendo también al azar una de las producciones que presentan en la gramática. En general, el sentido en el que se lleva a cabo la expansión es el mismo que el del recorrido "primero en profundidad" del árbol de derivación.

Las variantes que existen de cara a la inicialización de individuos se diferencian fundamentalmente en la elección aleatoria de la siguiente producción, dado que esta condiciona el resto de la operación. Así, en [Whigham 95] se propone emplear una ruleta definida por el usuario, que pondera unas producciones a favor de otras basándose en el conocimiento del problema y finalizar el proceso de creación en función de un número máximo de producciones a aplicar. En este caso, cabe la posibilidad de que se generen individuos incorrectos, dado que con el número máximo de producciones se pueden generar individuos que presentan símbolos no terminales en las hojas del árbol. En este caso, se procedería a repetir el proceso. Este método presenta el defecto de que, si

no se eligen adecuadamente las probabilidades, puede generar poblaciones de individuos con bastantes menos producciones del máximo permitido.

Por otra parte, A. Geyer-Schulz [Geyer 97] propone construir una ruleta basada en consideraciones de cardinalidad de cada una de las producciones de la gramática. La justificación de esta metodología, más compleja de implementar que la anterior, estriba en que no todas las producciones son capaces de generar el mismo número de individuos, sino que las que tienen un mayor número de símbolos no terminales van a producir una mayor variedad de individuos y, por esta razón, deben ponderarse por encima de las producciones que generan menos individuos. Esta metodología presenta la ventaja de producir poblaciones con una mayor diversidad, además de garantizar que los individuos son siempre correctos.

5.2.3 Operadores genéticos

Los operadores genéticos típicos en GBGP son los denominados por P. Whigham [Whighan 95] *cruce selectivo* y *mutación selectiva*, aunque se han reportado algunos otros operadores genéticos que producen resultados análogos a los anteriores en la resolución de problemas de regresión simbólica [Ventura et al. 2002].

5.2.3.1 Cruce selectivo

El operador de cruce selectivo es análogo al cruce de subárboles que se presenta en cualquiera de los paradigmas de programación genética. Sin embargo, por la estructura de los árboles de derivación el punto de cruce elige siempre a símbolos no terminales de la gramática. Para que la operación sea sintácticamente correcta, se obliga a que las raíces de los dos subárboles que se intercambian sean idénticas (Ver Figura **5.3**). Una de las características más importantes de este operador genético es que no presenta la naturaleza destructiva que presenta el cruce en la programación genética convencional dado que, si la gramática se construye adecuadamente, los subárboles que se van a intercambiar son realmente elementos intercambiables, es decir, se corresponden con subestructuras semánticamente compatibles.

En la operación de cruce selectivo se pueden añadir restricciones tales como que los hijos resultantes no excedan un determinado número de producciones o que la probabilidad de elegir unos símbolos no terminales sobre otros no sea uniforme.

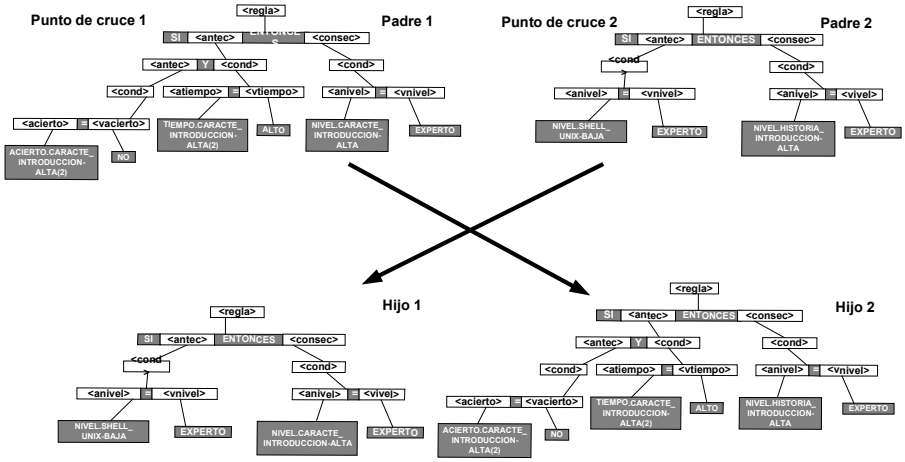

Figura 5.3: Ejemplo que ilustra el funcionamiento del cruce selectivo.

En el caso que nos ocupa, el cruce selectivo puede llevar a cabo cinco tipos de operaciones: Intercambio de los antecedentes de las reglas, intercambio de los consecuentes de las reglas, intercambio de una condición de una parte del árbol al otro, intercambio del atributo de comparación de una condición por otro o intercambio del valor de dicho atributo de comparación en una condición. Como se ilustrará posteriormente, se ha maximizado la probabilidad para el cruce de condiciones, atributos y valores y se ha minimizado la probabilidad de que se realice cruce de antecedentes y consecuentes completos, lo que supone una modificación mucho más drástica en la estructura de las reglas.

5.2.3.2 Mutación selectiva

La mutación selectiva reconstruye un subárbol tomando como raíz un nodo no terminal del mismo, sin más restricción que el número máximo de producciones que puede tener el árbol de derivación resultante. La **Figura 5.4** muestra cómo actúa la mutación selectiva sobre el árbol de derivación de la Figura 5.1, produciendo como regla resultante

SI NIVEL.COMODINES_UNIX-BAJA = EXPERTO
ENTONCES
NIVEL.CARACTE_INTRODUCCION-ALTA = EXPERTO

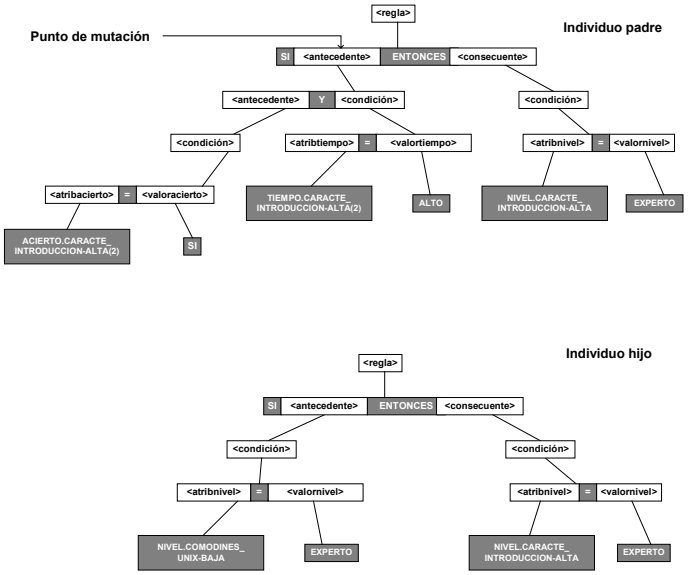

Figura 5.4: **Esquema que ilustra el funcionamiento de la mutación selectiva.**

En este operador genético, que pretende mantener la diversidad de la población, se pueden también ponderar la probabilidad con la que se puede elegir un símbolo terminal, aunque en nuestro caso todos los símbolos han sido equiprobables.

5.2.4 Reparación de individuos

En el problema del descubrimiento de reglas de predicción, utilizando la gramática que se muestra en la Figura 1.1 pueden presentarse individuos que, si bien son correctos sintácticamente, no lo son desde un punto de vista semántico. En efecto, desde este punto de vista, no tiene sentido que una regla de predicción presente la misma condición en el antecedente y en el consecuente, dado que no estaría prediciendo ninguna relación entre atributos. Por otra parte, tampoco tiene sentido permitir que la misma condición se repita en el antecedente. Este problema es consecuencia de las limitaciones inherentes a las gramáticas de contexto libre [Hopcroft 02], y sólo puede ser resuelto después de construir el árbol de derivación.

Existen dos formas de resolver el problema planteado anteriormente. La primera de ellas consiste en penalizar los individuos que no son correctos, introduciendo un término en la función de aptitud que tuviera en cuenta este

efecto. La segunda, que es la que se ha llevado a la práctica, consiste en reparar los individuos incorrectos mutando las condiciones causantes de la incorrección del individuo. Este algoritmo actúa antes de la evaluación de los individuos.

5.3 Algoritmo evolutivo

La extracción de reglas de predicción mediante de algoritmos evolutivos puede enfocarse desde dos puntos de vista, que se podrían denominar *enfoque restringido* y *enfoque no restringido*. En el primer caso, el problema se aborda de forma análoga al de la extracción de reglas de clasificación, es decir, el usuario prefija el atributo que se quiere predecir. El individuo sólo necesita contener el antecedente de la regla que se pretende descubrir[2], y una población de individuos contendrá un conjunto de antecedentes candidatos que compiten entre sí para resolver el problema de la predicción del atributo problema. El enfoque no restringido consiste en abordar el problema como uno de extracción de reglas de asociación: el individuo codifica una regla completa (antecedente y consecuente) y la población contiene reglas que predicen atributos distintos.

En este trabajo se ha abordado la tarea de descubrimiento de reglas mediante el enfoque que hemos denominado no restringido aunque, como se verá posteriormente, se han implementado mecanismos de filtrado en la gramática para limitar el número de elementos que integran el antecedente y el consecuente, reduciendo así el espacio de búsqueda y el tiempo de cómputo del algoritmo.

El Algoritmo 5.1 es el que se ha empleado en todas las pruebas desarrolladas. Como puede comprobarse, el algoritmo utiliza dos conjuntos de individuos. El primero de ellos se corresponde con la población, mientras que el segundo almacena una élite de individuos que son los que serán devueltos al usuario tras finalizar el algoritmo. Tras la inicialización de la población, se eligen los padres a partir de la población actual (cuya diversidad está garantizada, como ser verá) y del conjunto élite (para garantizar que haya padres de calidad). Una vez generados los hijos tras la aplicación de operadores genéticos, se actualiza la élite añadiendo los individuos con mejores propiedades no existentes aún y, posteriormente, se actualiza la población, garantizando la diversidad de ésta. La evolución finaliza cuando se alcanza un máximo de generaciones.

[2] En el caso de la GBGP, este objetivo puede conseguirse restringiendo la gramática presentada en la Figura 5.1, y evitando que el nodo del consecuente pueda experimentar modificaciones por cruce o mutación.

```
Inicio
  Crear la población inicial P y el almacén élite E
  Mientras (generación < MaxGeneraciones) hacer
    Seleccionar los futuros padres a partir de los individuos de P y E
    Aplicar operadores genéticos sobre los padres seleccionados
    Evaluar los hijos obtenidos
    Actualizar los conjuntos P y E
    generación ← generación + 1
  Fin Mientras
  Devolver el conjunto E
Fin
```

Algoritmo 5.1: Algoritmo Evolutivo empleado.

Para garantizar la diversidad de la población se ha utilizado una métrica basada en medir la distancia de cada individuo con la del resto de individuos con los que va a competir por la supervivencia. Dicha distancia se mide como el número de elementos distintos que existen en el antecedente y consecuente. Los individuos con mayor valor en dicha métrica serán los más distintos estructuralmente hablando y serán los más probablemente elegidos. En la nueva población se garantiza que no van a existir individuos repetidos, para evitar problemas de convergencia prematura.

En cuanto al conjunto élite, la elección se realiza de forma distinta dependiendo del algoritmo empleado. Así, en el caso de utilizar una única medida de aptitud, se establece un umbral por encima del cual los individuos pasan al conjunto élite. En el caso de enfoques basados en el concepto de frente de Pareto, se eligen siempre individuos no dominados[3].

A continuación, se describirán una serie de detalles relativos a la implementación concreta de cada una de las fases que comprenden el proceso de evolución.

5.3.1 Inicialización de la población

El método de inicialización de la población es el conocido como *inicialización escalonada o en rampas*, en el que se garantiza una distribución uniforme del número de individuos en función del número de producciones que presentan. En este caso hay que definir los números mínimo y máximo de producciones para los

[3] El concepto de frente de Pareto se explica en la página 122.

individuos lo que, en nuestro caso, representa limitar el número de condiciones en el antecedente de la regla.

Para realizar la inicialización de la población también se tienen en cuenta además las restricciones impuestas por el usuario, las cuáles están relacionadas con el conocimiento que éste desea extraer en una evolución concreta. En este sentido, el usuario puede especificar qué atributos desea que formen parte de las condiciones del antecedente y qué atributos desea que formen parte de las condiciones del consecuente, pudiendo también puede restringir los valores que aparecerán en las condiciones. De este modo, si un usuario desea analizar qué variables influyen en que un alumno obtenga una buena evaluación en el tema *Introducción*, indicaría al sistema que desea que las reglas contengan en el consecuente sólo elementos relativos a dicho tema. Con la implementación realizada, basada en GBGP esto se consigue de forma trivial. Se parte de la gramática inicial, que es la mostrada en la Figura 5.1, y se filtran los elementos terminales que no cumplen las expectativas del usuario, construyéndose una gramática restringida, que es la que se empleará en el algoritmo.

Por otro lado, el usuario también puede filtrar elementos iniciales en base al soporte de éstos, indicando a qué rango de soporte deben pertenecer los elementos que se incluyen en el conjunto de terminales. En función del número de elementos iniciales, el espacio de búsqueda será más amplio y, en consecuencia, el número de posibles reglas a encontrar será mayor.

5.3.2 Valoración de los individuos

El proceso de valoración de los individuos consiste en obtener una regla correcta a partir del árbol de derivación que contiene[4] y, posteriormente, aplicar una función que produzca una medida de la calidad de la regla. Se ha comentado anteriormente que existen multitud de métricas para valorar la calidad de las reglas, cada una de las cuáles se centran en algunos aspectos de las mismas. Sin embargo, no existe una medida que supere claramente a las demás en todos los dominios de aplicación, como muestran los estudios realizados a este respecto [Yao & Zhong 1999], [Lavrac et al. 1999], [Tan & Kumar 2000]. La razón de este hecho estriba precisamente en que cada medida capta mejor unos aspectos de la regla que otros y, si hay que elegir uno de ellos, es necesario realizar un análisis específico para cada dominio de aplicación.

[4] Lo cual puede suponer la reparación de la regla obtenida, en virtud del algoritmo de reparación expuesto anteriormente.

Las consideraciones anteriores sugieren el planteamiento de este problema como un problema de *optimización multiobjetivo* [Fonseca y Fleming 1994]. En este caso, no existiría una única función de aptitud asociada a una medida, sino que existen varias funciones que se desean optimizar simultáneamente. Existen varias formas de abordar el problema de la optimización multiobjetivo con algoritmos evolutivos: en el primer caso se hace uso de *funciones de agregación*, mientras que el segundo caso hace uso del concepto de *frente de Pareto*. Ambos enfoques serán tratados en las siguientes Secciones.

5.3.2.1 Funciones de agregación

En este caso, la función de aptitud es una combinación lineal de las distintas métricas que se quieren optimizar. Los pesos de ponderación de esta combinación lineal dan fe de la importancia relativa de cada una de las medidas en la función global. Existen varios ejemplos de métricas que combinan linealmente algunos de los objetivos perseguidos en la tarea de modelado de dependencias, entre los que se pueden citar las propuestas por Alves [Alves et al. 1999], Liu [Liu & Kwok 2000] o Freitas [Freitas 2002].

5.3.2.1.1 Función de ajuste de Alves

La función de ajuste propuesta por Alves [Alves et al. 1999] está basada en dos componentes: la primera utiliza una métrica denominada *J-medida* [5], que está relacionada con el interés de la regla, y la segunda utiliza el número de atributos potenciales del antecedente. Un atributo es potencial si al menos hay un ejemplo que cumple el atributo y el consecuente. La expresión de la función de Alves es:

$$AjusteA(A \rightarrow C) = \frac{w1 * J1 + w2 * \left(\frac{n_{pu}}{n_T} \right)}{w1 + w2}.$$

donde n_{pu} es el número de atributos potenciales en el antecedente y n_T es el número total de atributos en el antecedente. **w1** y **w2** son pesos puestos por el usuario y que el autor ha establecido en 0.6 y 0.4, respectivamente.

[5] En realidad, la función de ajuste de Alves utiliza una variante de la medida J denominada en medida J1.

5.3.2.1.2 Función de ajuste de Liu

La función de ajuste propuesta por Liu, tal y como se describe en el algoritmo SIA extendido [Liu & Kwok 2000] es la siguiente:

$$AjusteLK(A \rightarrow C) = w1 * Cons(A \rightarrow C) + w2 * Comp(A \rightarrow C) + w3 * Gen(A \rightarrow C)$$

Donde *Cons* es la consistencia de la regla, que se mide mediante la medida de *Laplace* [Bayardo & Agrawal 1999], *Comp* representa la completitud, que se calcula como

$$Comp(A \rightarrow C) = p(AC) / p(C)$$

y *Gen* es la generalidad, que se calcula mediante:

$$Gen(A \rightarrow C) = 1 - \frac{TamañoAntecedente}{TamañoMáximo}$$

Como puede comprobarse, las tres cualidades que estos autores consideran esenciales que una regla sea de calidad son que sea consistente, tenga un elevado cumplimiento y sea general. También indicar que los autores de esta métrica sugieren la utilización de los siguientes valores para los pesos *w1, w2 y w3*:

w1 = 0.5 + 0.25Cons(A→C).
w2 = 0.5-0.25Cons(A→C .
w3 = 1 – (w1 + w2).

5.3.2.1.3 Función de ajuste de Freitas

La función de ajuste propuesta por Freitas [Freitas 2002] tiene dos componentes, uno que mide la exactitud de la regla en predicción y el otro la comprensibilidad de la regla:

$$AjusteF(A \rightarrow C) = w1 * (CF * Comp) + w2 * Simp$$

donde:

$$CF = \frac{p(AC)}{p(AC) + p(A\neg C)}$$

$$Comp = \frac{p(AC)}{p(AC) + p(\neg AC)}$$

$$Simp = 1 - (\frac{TamañoAntecedente}{TamañoMáximo})$$

Los valores recomendados para w1 y w2 son de 0.8 y 0.2 respectivamente.

5.3.2.2 Otros enfoques multiobjetivo

El uso de métricas combinadas para la resolución del problema de descubrimiento de reglas no suele ser una buena aproximación debido fundamentalmente a que cada una de ellas está asociada a una de las cualidades deseables para las reglas y, con frecuencia dichas cualidades suelen estar en conflicto. Aunque, en general, se ha reportado que el uso de este tipo de métricas produce mejores resultados que de una sola métrica que contemple uno de los objetivos a optimizar, es mucho más correcto trabajar desde el punto de vista de los algoritmos basados en el concepto de *frente de Pareto*. En estos algoritmos, existe un vector de objetivos a optimizar por individuo, y el propósito de los algoritmos es hacer que se converja hacia el conjunto que está formado por las mejores soluciones (en términos de todos los objetivos individuales, no de cada uno por separado), denominado frente de Pareto [Freitas 2002]. Se han implementando dos algoritmos pertenecientes a esta familia, el algoritmo MOGA y el NSGA. A continuación, describiremos brevemente ambos algoritmos, finalizando la sección con una serie de consideraciones que han servido para tomar la decisión de qué métricas emplear en estos algoritmos.

```
Inicio
  Para cada individuo
    Calcular el número de individuos por los cuales es dominado
    Número de orden del individuo = Número individuos que lo dominan + 1
  Fin Para
  Ordenar la población según los rangos de los individuos
  Asignar el ajuste de los individuos interpolando desde el de rango 1 al de rango n≤N
Fin
```

Algoritmo 5.2: Algoritmo MOGA.

5.3.2.2.1 Algoritmo MOGA

El algoritmo MOGA (Multi-Objective Genetic Algorithm) [Fonseca & Fleming 1993] está basado en la idea de ordenar a los individuos dependiendo de su no dominación, de forma que el orden de cada individuo corresponde con el número de individuos por los cuales es dominado. El pseudocódigo del algoritmo es el que se muestra en Algoritmo 5.2. Como se puede observar, a todos los individuos no dominados se le asigna un valor de orden 1, mientras que el resto son penalizados según la cantidad de individuos por los cuales es dominado. Para comprobar si un individuo es dominado por otro individuo se ha utilizado una comparación Pareto de los dos individuos utilizando el Algoritmo 5.3, que compara todos los objetivos.

```
Inicio
  Para cada objetivo
    Si (objetivo[i]IndividuoA > objetivo[i]IndividuoB)
      IndividuoB domina al IndividuoA
    Sino Si objetivo[i]IndividuoA < objetivo[i]IndividuoB
      IndividuoA domina al IndividuoB
    Sino
      Ninguno es dominado por el otro
    Fin Si
  Fin Para
Fin
```

Algoritmo 5.3: Algoritmo comprobación de dominancia.

5.3.2.2.2 Algoritmo NSGA

El algoritmo NSGA (Non-dominated Sorting Genetic Algorithm) [Srinivas & Deb, 93] se basa en varias capas de clasificación de los individuos y también establece rangos en los individuos basándose en su no dominancia (ver Algoritmo 5.4). Como se puede observar, primero se ordena la población en función del concepto de no dominanción y a continuación se le asigna la aptitud a cada individuo en función de su rango dentro de la población e incluyendo un método de compartición de aptitud.

Para calcular el número de individuos por los cuales es dominado, se utiliza el método de comparación Pareto que se ha presentado anteriormente (ver Algoritmo 5.3). El valor del radio del nicho *delta-share* se suele obtener según la ecuación

$delta - share = 1/(2\sqrt[p]{q})$, donde p es el número de variables y q el número máximo de soluciones óptimas de Pareto que se desea obtener.

```
Inicio
  Para cada individuo
    Calcular el número de individuos por los cuales es dominado
    Rango del individuo = Número individuos por los cuales es dominado + 1
  Fin Para
  Asignar Aptitud Falsa a todos los individuos obtenida como
                                        Rango individuo / Tamaño población
  Para cada individuo
    Calcular su distancia euclídea con respecto a los otros individuos
    Si (distancia es menor que parámetro delta-share) entonces
      Funcion Compartición = Función Compartición +  ( distancia / delta-share)^2
    Fin si
    Calcular el ajuste compartido como aptitud falsa / Función Compartición
  Fin para
Fin
```

Algoritmo 5.4: Algoritmo NSGA.

5.3.2.2.3 Elección de la métrica más adecuada

A la hora de llevar a cabo un algoritmo evolutivo basado en el concepto de dominación Pareto, hay que elegir una serie de métricas de calidad de las reglas a optimizar. Según [Freitas 2002], el conocimiento descubierto por un algoritmo de minería de datos debe satisfacer tres aspectos principales: *exactitud*, *comprensibilidad* e *interés*. Por esta razón, la función de ajuste debe estar formada por un vector de tres valores donde cada uno mida uno de estos criterios. Las métricas que se han seleccionado como objetivos parciales son las denominadas *factor de certeza* [Shortliffe & Buchanan 1975], *interesabilidad* [Tan & Kumar 2000] y *simplicidad* [Liu & Kwok 2000].

La elección de dichas medidas se basa en el análisis de la bibliografía disponible y en que cada una de ellas está relacionada con cualidades distintas. En cuanto al factor de certeza, indicar que se trata de una medida de exactitud que, según [Delgado et al. 2001], ha mostrado mejores resultados que la confianza, medida empleada clásicamente para cuantificar esta cualidad. La medida denominada interesabilidad, que está relacionada con el interés de las reglas, produce, según [Silverstein et al. 1998], mejores resultado que los que arroja el empleo de la medida interés, de la cual deriva. La medida denominada *simplicidad* [Liu & Kwok 2000] mide la longitud sintáctica de la regla y está relacionada con

la compresibilidad de la regla ya que, según su autor, esta cualidad es inversamente proporcional a su tamaño, y se puede calcular como:

$$Simplicidad = 1 - (\frac{TamañoAntecedente}{TamañoMáximo})$$

Esta medida puede tomar valores entre 0 y 1, donde el valor máximo 1 indica la regla más simple y el valor mínimo de 0 indica la regla más grande posible.

5.4 Sección experimental

En esta sección se describirán todas las pruebas realizadas, así como la información relevante relacionada con el desarrollo de la experimentación relacionada con esta fase de extracción de conocimiento.

5.4.1 Descripción de las pruebas

Se han realizado tres tipos de pruebas orientadas a comparar los resultados que produce cada uno de los algoritmos implementados en la tarea de descubrimiento de conocimiento planteada. La primera está orientada a comparar el *tiempo de ejecución de los algoritmos*. Esta variable es de cierta importancia si se pretende implantar esta fase de extracción de conocimiento en línea con la aplicación. La segunda batería de pruebas está orientada a comparar el *número de reglas descubiertas en cada caso y la calidad de las mismas* en base a las métricas planteadas anteriormente.

Para la realización de las pruebas de velocidad, se ha ejecutado cada uno de los algoritmos 10 veces en un ordenador personal Pentium III a 800 MHz, empleando los parámetros que se muestran la **Tabla 5.1**. Para cada uno de los algoritmos ejecutados, mantuvieron los parámetros de configuración de éstos, excepto en los algoritmos evolutivos en los que se modificó la semilla del generador de números aleatorios empleado.

Para la realización de la segunda batería de pruebas, se han realizado también 10 repeticiones de cada algoritmo y se ha realizado la comparación a nivel de cuatro factores: número total de reglas, interés, exactitud y comprensibilidad de las reglas producidas.

En ambos casos se han utilizado tres conjuntos de elementos iniciales: el primero coincide con el conjunto de partida, es decir, está formado por todos los

valores posibles que existen en la base de datos, el segundo conjunto utiliza como conjunto inicial el formado por los elementos frecuentes, que son aquellos que superan un umbral de soporte de 0.5 y el tercer conjunto está formado por elementos cuyo soporte está contenido en el intervalo (0.2, 0.9). La razón de elegir el conjunto de todos los elementos y los frecuentes está en examinar el rendimiento de los algoritmos en condiciones de un gran número de elementos de partida y un número mucho más reducido. La razón de elegir el rango intermedio, se examinará en el siguiente Capítulo.

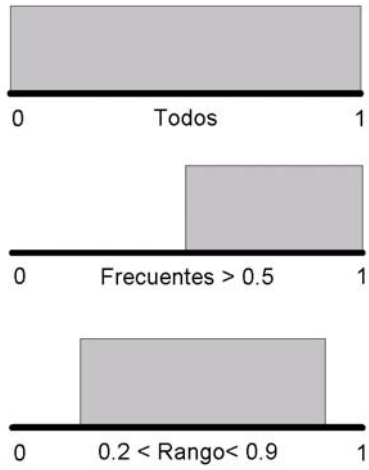

Figura 5.5: Métodos de inicialización empleados en las pruebas.

5.4.2 Algoritmos clásicos y extracción de reglas de predicción

En el Capítulo *Antecedentes* se plantearon las diferencias entre las reglas de asociación, clasificación y predicción; de la discusión allí planteada, puede sacarse como conclusión que las diferencias sintácticas que existen entre dichas tipologías de reglas son mínimas, siendo las principales diferencias entre ellas diferencias de tipo semántico. Por esta razón, no resulta difícil modificar un algoritmo de extracción de reglas de asociación para realizar la tarea de modelado de dependencias, del mismo modo que es posible modificar un algoritmo de extracción de reglas de clasificación para que realice la misma labor.

La conversión de un algoritmo de obtención de reglas de asociación en un algoritmo de obtención de reglas de predicción es trivial, dado que sólo hay que forzar a que el consecuente de la regla presente un único atributo. En el caso de los algoritmos de obtención de reglas de clasificación, la regla que se construye indica

la pertenencia o no a una clase, que no es más que la predicción de un atributo categórico. La adaptación al ámbito de reglas de predicción sería que dicho atributo no fuese categórico, sino que la condición que se predice fuera cualquiera de las que se pueden extraer de la base de datos. Sin embargo, un algoritmo de obtención de reglas de clasificación sólo genera una regla. Por tanto, si se desea extraer reglas de predicción para N atributos de la base de datos, será necesario ejecutar N veces el algoritmo de extracción de reglas de clasificación, empleando en cada caso un atributo como variable clasificador.

Con el objeto de comparar los resultados producidos por algoritmos clásicos con los producidos por los algoritmos basados en GBGP, se han realizado implementaciones para los algoritmos Apriori, PRISM e ID3 orientadas a la generación de reglas de predicción. La elección de estos algoritmos como referente frente a los desarrollados se debe a su popularidad en el ámbito de la minería de datos, que hace hayan sido utilizados en multitud de ocasiones en esta tarea de comparación (consultar, por ejemplo [Freitas 2002]).

5.4.3 Parámetros de los algoritmos

En la Tabla 5.1 se muestran los parámetros de configuración para los algoritmos que han integrado las pruebas. Para cada uno de los métodos de inicialización descritos anteriormente (frecuentes, rango y todos), se han utilizado tamaños de población de 50, 100 y 200 individuos, respectivamente. La inicialización se realiza mediante el método denominado "en rampa", y el número de producciones se ha ajustado para que se generen individuos con 1, 2 y 3 condiciones en el antecedente. Para garantizar la diversidad de la población, se ha evitado la repetición de individuos en dicha población inicial.

Con respecto a la dinámica del proceso de evolución, se ha realizado en todos los casos selección basada en orden (*rank-based selection*). En este mecanismo, la probabilidad de elegir a un individuo es proporcional a la posición que presenta en el conjunto después de que éste haya sido ordenado por aptitud según un orden creciente, es decir, el peor individuo presenta probabilidad 1, el segundo peor 2, ... y el mejor N (siendo N el tamaño del conjunto). Con respecto a los operadores de cruce aplicados, un 80% de los individuos que son seleccionados como padres son sometidos a cruce selectivo, y un 20% son sometidos a mutación selectiva. La Tabla 5.1 también muestra que, en el caso del cruce, se ha potenciado la operación de intercambio de condiciones frente a las demás, aunque se ha permitido que el cruce se realice a nivel de todos los elementos no terminales de la gramática.

Con respecto al algoritmo NSGA, se ha fijado también el valor del parámetro delta-share (ver Algoritmo 5.4) al valor de 2.

FASE DE INICIALIZACIÓN

- **Tamaño de población**: 50, 100 y 200 individuos
- **Método de inicialización**: En rampa

Número mínimo de producciones**:** 9
Número máximo de producciones: 18

FASE DE REPRODUCCIÓN

- **Selección** basada en orden
- Operador de cruce selectivo
  - Probabilidad de éxito: 0.8
  - Ruleta:
    - <antecedente> = 1
    - <consecuente> = 1
    - <condición> = 4
    - <atributo$_i$> = 2
    - <valor$_i$> = 2
- **Operador de mutación selectiva:**
  - Probabilidad de éxito: 0.2
  - Ruleta:
    - <antecedente> = 1
    - <consecuente> = 1
    - <condición> = 1
    - <atributo$_i$> = 1
    - <valor$_i$> = 1

CRITERIO DE PARADA

- **Número máximo de generaciones**: 50

Tabla 5.1: Parámetros del algoritmo evolutivo.

5.4.4 Implementación

Todos los algoritmos de GBGP se han implementado en Java utilizando la biblioteca de clases Java para Computación Evolutiva desarrollada por el grupo "Aprendizaje y Redes Neuronales Artificiales" de la Universidad de Córdoba [Ventura et al. 2001].

La implementación que presenta esta biblioteca de clases para el paradigma de la GBGP codifica los árboles sintácticos como vectores de enteros ordenados según el recorrido del árbol en preorden[6]. El valor almacenado en el vector codifica, en forma de campos de bits, el símbolo contenido en el nodo, toda la información necesaria para su manipulación y posterior conversión a una consulta SQL. Esta implementación presenta la ventaja de permitir la reutilización de los individuos generados, reduciendo sensiblemente los requisitos de memoria de la aplicación y, consiguientemente, aumentando su eficiencia. Además, la localización de nodos es muy eficiente, reduciéndose el tiempo de cómputo en las operaciones de cruce y mutación selectivos. La valoración de individuos consiste en la conversión de la cadena de enteros en una serie de consultas SQL mediante las cuáles se determinan los valores necesarios para el cálculo de la métrica o métricas empleadas como objetivos a optimizar.

Los algoritmos clásicos empleados en esta comparación (ID3, Prism y A priori adaptados a la extracción de reglas de predicción) se implementaron también lenguaje Java y están integrados en la herramienta EPRules [Romero et al. 2003a], una herramienta de extracción de reglas de predicción que ha sido desarrollada por el autor de la presente Memoria y que implementa todas las fases del proceso de extracción de conocimiento descritos en este Capítulo y el anterior.

5.5 Resultados y discusión

5.5.1 Tiempos de ejecución

La Tabla 5.2 muestra los valores medios para el tiempo de ejecución invertido en cada uno de los algoritmos considerados.

[6] El requisito que impone esta implementación es que se represente a expresiones en notación prefija. Por esta razón, se ha modificado la gramática que se ha mostrado en la Figura 5.1, adaptándola a este requisito.

Algoritmo	Tiempo de Ejecución (seg)		
	Todos	Rango	Frecuentes
ID3	4.532	983	214
Prism	8.886	1.052	157
Apriori	8.882	328	29
AE-GBGP	699	381	147

Tabla 5.2: Comparación de tiempos de ejecución para los algoritmos desarrollados.

Como puede comprobarse, los algoritmos evolutivos son, en general más rápidos que los algoritmos clásicos de descubrimiento de reglas cuando el número de elementos de partida es elevado. En este caso, la duración de una ejecución del algoritmo (alrededor de una hora y cuarto en el caso del algoritmo ID3 y alrededor de 3 horas en los algoritmos Prism y A priori) no justificaría su empleo en el sistema propuesto[7]. Estas diferencias se van reduciendo a medida que disminuye el número de elementos de partida, llegando a invertirse en el caso del algoritmo A priori, que es significativamente más rápido que el evolutivo cuando se parte de conjuntos reducidos (elementos frecuentes) y comparable a éste con conjuntos de tamaño intermedio. Este hecho revela la debilidad de este algoritmo, que tiene muy buenas propiedades cuando el conjunto de elementos de partida es reducido, pero que falla cuando trabaja con conjuntos de datos elevados. En el caso del algoritmo evolutivo, sin embargo, los tiempos son mucho más manejables (alrededor de 12 minutos en los casos más desfavorables y alrededor de 2 minutos en los más favorables), lo que los hace mucho más apropiados de cara a la incorporación on-line de la fase de extracción de conocimiento en el sistema.

La Tabla 5.3 realiza una comparación del tiempo de ejecución para todos los algoritmos evolutivos desarrollados. Como puede comprobarse, las diferencias son en este caso mucho menos acusadas que en el caso anterior (de hecho, estas diferencias radican en el coste computacional que representa el cálculo de la función de aptitud) aunque son significativas con un nivel de confianza del 95%. Como puede comprobarse, en general, las versiones que utilizan un fitness basado en el concepto de frente de Pareto son más rápidas que las que se basan en las métricas de Lui, Freitas o Alves.

[7] A no ser que la cantidad y calidad de las reglas obtenidas fuera muy superior a las que se obtienen en el caso de los algoritmos evolutivos, lo cual, como se verá, no es el caso.

Fitness	Tiempo de Ejecución (seg)		
	Todos	Rango	Frecuentes
Liu	732	354	160
Freitas	702	293	151
Alves	793	390	161
MOGA	642	323	138
NSGA	624	247	123

Tabla 5.3: Comparación de tiempos de ejecución para las distintas versiones del AE desarrollado.

5.5.2 Número total de reglas descubiertas

La Tabla 5.4 muestra el número total de reglas que se descubren tras la ejecución de los algoritmos clásicos y la media de todas las pruebas para el caso de las distintas versiones del algoritmo evolutivo desarrollado, y la Tabla 5.5 compara los resultados para las diferentes versiones del AE desarrollado.

Algoritmo	Número de reglas descubiertas		
	Todos	Rango	Frecuentes
ID3	474	131	89
Prism	657	172	62
Apriori	7960	491	70
AE-GBGP	198	162	51

Tabla 5.4: Número total de reglas descubiertas por los algoritmos desarrollados.

Fitness	Número de reglas descubiertas		
	Todos	Rango	Frecuentes
Liu	317	244	74
Freitas	148	159	37
Alves	350	272	85
MOGA	98	79	30
NSGA	75	57	28

Tabla 5.5: Número total de reglas descubiertas para las distintas versiones del algoritmo evolutivo desarrollado.

Como puede comprobarse, el número de reglas que devuelve un algoritmo clásico no es manejable cuando se utiliza como conjunto de partida el formado por todos los posibles atributos y valores, siendo extraordinariamente alto en el caso del algoritmo A priori. Este comportamiento se atenúa a medida que desciende el número de elementos de partida de forma que, cuando este número se reduce sensiblemente (el conjunto de elementos frecuentes es aproximadamente un 10% del tamaño del conjunto formado por todos los elementos), su comportamiento es análogo al de los demás algoritmos clásicos. En este caso se sigue comprobando que el punto fuerte del algoritmo Apriori es trabajar con un conjunto de partida reducido, y que en ningún caso es aplicable con conjuntos de un tamaño elevado, como es este caso.

Examinando la Tabla 5.5 puede comprobarse que existe una gran diferencia entre el número de reglas que se descubren en el caso de los AEs que emplean una función de agregación y los que utilizan el enfoque basado en frente de Pareto, siendo el algoritmo NSGA el que produce un menor número de reglas.

5.5.3 Calidad de las reglas descubiertas

Se ha evaluado la calidad de las reglas descubiertas en términos del porcentaje de reglas que cumplen unos determinados requisitos de exactitud, comprensibilidad e interés, criterios que, como ya se ha comentado, deben ser optimizados de cara a obtener una regla de calidad [Freitas 2002].

5.5.3.1 Exactitud de las reglas descubiertas

La Tabla 5.6 muestra el porcentaje de reglas exactas obtenidas. Para ello, se han filtrado las reglas que presentan un valor de factor de certeza mayor o igual a 0.7 y se ha calculado el porcentaje de este número respecto al total de reglas descubiertas.

Algoritmo	Porcentaje de reglas exactas		
	Todos	Rango	Frecuentes
ID3	46.0	51.9	60.3
Prism	71.9	53.7	91.9
Apriori	84.3	90.0	93.0
AE-GBGP	76.5	86.1	96.3

Tabla 5.6: Porcentaje de reglas exactas descubiertas por los algoritmos desarrollados.

Consultando la información puede comprobarse que, en general, el algoritmo Apriori produce reglas muy exactas, es decir, que presentan un elevado grado de cumplimiento (no en vano se basa en el concepto de soporte de la regla, que es también una medida de su exactitud). Sin embargo, los resultados que presentan los algoritmos basados en el concepto de frente de Pareto producen reglas exactas con una garantía muy cercana al 100% en todos los casos (ver la Tabla 5.7), no existiendo diferencias significativas entre los resultados producidos por las variantes MOGA y NSGA.

Fitness	Porcentaje de reglas exactas		
	Todos	Rango	Frecuentes
Liu	63.1	74.1	95.2
Freitas	61.6	79.1	95.4
Alves	62.3	82.9	94.4
MOGA	97.0	96.7	97.3
NSGA	98.5	97.6	99.3

Tabla 5.7: Porcentaje de reglas exactas para las distintas versiones del algoritmo evolutivo desarrollado.

Es muy importante resaltar que, en el caso de MOGA y NSGA este comportamiento puede considerarse independiente del tamaño del conjunto de partida (dado que no existen diferencias significativas entre los valores medios calculados), a diferencia del que exhiben los algoritmos clásicos y los basados en las métricas de Liu, Alves y Freitas.

5.5.3.2 Comprensibilidad de las reglas descubiertas

La Tabla 5.8 muestra el porcentaje de reglas descubiertas que presentan un valor de la medida de simplicidad superior a 0.5, es decir, que presentan un grado de comprensibilidad elevado, derivado de su sencillez. La Tabla 5.9 muestra la misma información para el caso de AEs en los que los individuos se evalúan usando las métricas de Liu, Freitas y Alves y los enfoques MOGA y NSGA.

Un examen de los resultados obtenidos muestra que, en general, los algoritmos evolutivos producen un porcentaje de reglas simples muy superior al que producen el resto de los algoritmos, siendo este porcentaje independiente del tamaño del conjunto inicial. En efecto, examinando la Tabla 5.9 se comprueba que, en cada uno de los casos, la proporción de reglas simples se mantiene constante,

representando aproximadamente un 25%, sin que existan diferencias significativas entre estos valores con un nivel de confianza de un 95%.

Algoritmo	Porcentaje de reglas comprensibles		
	Todos	Rango	Frecuentes
ID3	9.0	9.1	6.6
Prism	4.0	1.0	3.0
Apriori	13.7	7.7	12.2
AE-GBGP	20.0	19.2	19.5

Tabla 5.8: Porcentaje de reglas comprensibles descubiertas por los algoritmos desarrollados.

Fitness	Porcentaje de reglas comprensibles		
	Todos	Rango	Frecuentes
Liu	22.1	17.6	20.8
Freitas	23.8	24.5	24.4
Alves	3.0	3.9	4.7
MOGA	25.3	24.3	23.3
NSGA	25.9	25.8	24.4

Tabla 5.9: Porcentaje de reglas comprensibles para las distintas versiones del algoritmo evolutivo desarrollado.

5.5.3.3 Interés de las reglas descubiertas

Las Tabla 5.10 muestra el porcentaje de reglas producidas que presentan un interés elevado, expresado como un valor superior a 0.5 para la magnitud interesabilidad.

Algoritmo	Número de reglas descubiertas		
	Todos	Rango	Frecuentes
ID3	1.5	7.6	15.6
Prism	2.5	11.6	49.3
Apriori	3.6	7.9	53.1
AE-GBGP	21.9	60.4	76.6

Tabla 5.10: Porcentaje de reglas interesantes descubiertas por los algoritmos desarrollados.

Un examen de esta información revela que, en general, los algoritmos evolutivos producen reglas con un interés mucho más elevado que los algoritmos clásicos, siendo este porcentaje tanto mayor cuanto mayor es el soporte de los elementos del conjunto de partida. Este resultado queda también patente al consultar la Tabla 5.11, en la que se comparan las cinco versiones del AE. Como en los casos anteriores, las versiones MOGA y NSGA producen, en general, mejores resultados que las que están basadas en las métricas propuestas por Liu, Freitas y Alves. Es interesante reseñar también que la medida propuesta por Alves, que contempla una magnitud relacionada con el interés, presenta reglas mucho más interesantes que las que se obtienen con las métricas de Liu y Freitas. Recordemos que esta métrica es la que produce peores resultados en términos de exactitud y de comprensibilidad de las reglas.

Fitness	**Número de reglas interesantes**		
	Todos	**Rango**	**Frecuentes**
Liu	19.6	18.8	31.0
Freitas	21.0	23.8	24.3
Alves	49.1	59.4	72.9
MOGA	68.5	74.0	81.1
NSGA	83.4	85.4	87.1

Tabla 5.11: Porcentaje de reglas interesantes para las distintas versiones del algoritmo evolutivo desarrollado.

5.6 Conclusiones

Se ha comprobado la capacidad de los algoritmos de programación genética basada en gramáticas para resolver la tarea de extracción de reglas de predicción orientadas a la mejora de un ASWE. La elección de este tipo de codificación es muy apropiada en el sentido de que permite fácilmente adaptar el espacio de búsqueda a las necesidades del usuario, permitiendo a éste definir el tipo de información que desea extraer. La valoración de los individuos se ha empleado utilizando un enfoque multiobjetivo, en el que se intentan optimizar la exactitud de la regla, su interés y su comprensibilidad, y se han ensayado versiones del algoritmo que emplean funciones de aptitud que combinan estos objetivos (métricas de Liu, Freitas y Alves) y otras que enfocan el problema desde el punto de vista de la obtención del frente de Pareto: algoritmos MOGA y NSGA.

Los resultados obtenidos muestran que, en general, los algoritmos evolutivos son más rápidos que los algoritmos clásicos, siendo más aptos para su utilización

on-line como métodos de extracción de conocimiento en el sistema de enseñanza adaptativo.

Los algoritmos clásicos, y sobre todo el Apriori producen, en general, reglas bastante bastante exactas, pero fallan a la hora de generar reglas con interés elevado y, además, la longitud de las reglas que producen dificulta su comprensibilidad. Además, cuando el conjunto de partida es elevado (lo cual puede suceder cuando el usuario desea extraer información global acerca del sistema, sin aplicar ningún tipo de restricción sobre dicho conjunto), los generan un conjunto tan enorme de reglas que hace impide su aprovechamiento posterior.

Los resultados obtenidos para los algoritmos evolutivos muestran que, en general, producen un menor número de reglas que los algoritmos clásicos, siendo esta diferencia de un orden de magnitud en los casos más favorables. Además, la proporción de reglas comprensibles e interesantes es bastante superior. En cuanto a los enfoques basados en funciones de agregación, ha quedado patente que las funciones de ajuste de Liu y Freitas se centran fundamentalmente en optimizar la exactitud y la comprensibilidad de las reglas, siendo la métrica de Alves la que produce reglas con mayor valor de interés. El uso de algoritmos basados en el concepto de frente de Pareto (MOGA y NSGA) optimiza, sin embargo, los tres objetivos planteados, produciendo en todas las ejecuciones la mayor proporción de reglas exactas, comprensibles e interesantes.

Capítulo 6
Descripción de la Información Descubierta

En este capítulo se va a describir la información descubierta en forma de reglas de predicción y su posible utilización para la mejora de cursos hipermedia adaptativos basados en web, y en concreto la del curso de sistema operativo Linux. En primer lugar se explicarán los distintos tipos de reglas que se pueden descubrir y la posible utilización de cada una de ellas para la mejora del curso, finalmente se muestran algunos de ejemplos concretos de reglas descubiertas de cada tipo y su utilidad.

CAPÍTULO 6: DESCRIPCIÓN DE LA INFORMACIÓN DESCUBIERTA

6.1 Introducción

La información, descubierta en forma de reglas de predicción, se puede utilizar para ayudar al diseñador o autor de un curso hipermedia adaptativo basado en web, en la toma de decisiones sobre qué modificaciones se podrían hacer en el curso para mejorarlo. Para ello, tras descubrir un conjunto de reglas, se necesita realizar una etapa de postprocesamiento de la información descubierta [Klösgen & Zytkow 2002] que consiste en la interpretación, evaluación de los resultados obtenidos y la utilización del conocimiento descubierto. En este capítulo se va a tratar precisamente esta etapa de análisis de la información o reglas descubiertas para su utilización en la mejora de los cursos. En los capítulos anteriores de descubrimiento de reglas de predicción se han realizado diferentes pruebas experimentales con diferentes tipos de algoritmos. De manera que sólo se ha utilizado las reglas descubiertas para comparar la eficiencia de los algoritmos desde el punto de vista del número de reglas descubiertos, el tiempo de ejecución, y el porcentaje de las reglas descubiertas que cumplen una serie de aspectos como son la exactitud, interés y comprensibilidad. En este capítulo describirá la información descubierta, desde un punto de vista de su utilización específica, en el descubrimiento de posibles mejoras de cursos.

Se ha desarrollado una herramienta gráfica específica para facilitar la realización del proceso de descubrimiento de información sobre los datos de utilización de nuestro ASWE [Romero et al. 2003a]. Dicha herramienta está orientada a su utilización por un diseñador o autor del curso. El uso de esta herramienta simplifica y facilita todo el proceso de descubrimiento de información. La funcionalidad de la herramienta se describe en el apéndice *Herramienta para Descubrimiento de Conocimiento*.

6.2 Descripción de la Información Descubierta en forma de Reglas de Predicción

El objetivo que se persigue con el descubrimiento de las reglas de predicción, es obtener relaciones importantes entre los datos de utilización de los ASWE. Esta información descubierta se desea utilizar para la toma de decisiones y deducir qué modificaciones en la estructura de los cursos podrían mejorarlo. Tradicionalmente la información que se utiliza para la evaluación de un curso [Zaïane & Luo 2001] es la información obtenida al realizar un análisis estadístico de los datos de los alumnos, que sólo tienen en cuenta a los elementos o datos de forma individual, descubriendo información sobre los valores máximos, mínimos, medias, varianzas, etc. El diseñador del curso podría utilizar esta información para tomar decisiones sobre posibles modificaciones a realizar sólo en contenido de las distintas páginas web. Por ejemplo, puede decidir cambiar el contenido teórico de una determinada página debido al excesivo tiempo de visualización de dicha página, o cambiar el enunciado de las preguntas donde el porcentaje de aciertos o fallos sea muy elevado.

En cambio la propuesta que se hace en este trabajo consiste en utilizar la información referente a relaciones entre los elementos o datos de utilización, además de la anterior información estadística referente a elementos individualmente. Utilizando esta nueva información se podrían tomar decisiones para realizar modificaciones, no sólo sobre el contenido, sino también sobre la estructura del curso. Para ello, el diseñador, a partir de los distintos tipos de reglas de predicción descubiertas, debe obtener sus propias conclusiones sobre las relaciones descubiertas y determinar si se deben realizar modificaciones en el curso para eliminar o reforzar estas relaciones dependiendo de si son beneficiosas o no.

Pero, antes de pasar a describir detalladamente los distintos tipos de relaciones que se pueden descubrir, hay que matizar las restricciones que se deben realizar cuando se realiza un proceso de descubrimiento, para poder obtener un conjunto reducido de conocimiento interesante. La realización de un proceso de búsqueda de reglas sin especificar ningún tipo de restricción, produce normalmente un número enorme de reglas [Bayardo & Agrawal 1999], que dificulta su utilización y por consiguiente la obtención de conclusiones, ya que muchas de ellas pueden no ser útiles o simplemente muestren información obvia, información de simple interés informativo, información no relevante, información de relaciones aleatoria, etc. Por lo tanto se hace necesaria la utilización de restricciones tanto objetivas como subjetivas que sirvan de filtrado de las reglas descubiertas. Estas restricciones son especificadas por el propio diseñador o persona encargada de la construcción y mantenimiento del curso, para que pueda concretar el tipo de reglas que le interesa descubrir en cada momento.

6.2.1 Restringiendo la Información Descubierta

Como ya se ha descrito anteriormente se pueden distinguir dos tipos de restricciones, por un lado, las referentes a los datos de utilización y por otro las referentes a las propias reglas. A continuación se va a realizar un breve resumen de los distintos tipos de restricciones.

Restricciones en los datos. Se pueden especificar dos tipos de restricciones en los datos a utilizar:

- **Basadas en la frecuencia o soporte de los elementos.** Se puede seleccionar entre utilizar todos los posibles elementos, sólo los frecuentes, los infrecuentes o un rango.

- **Basadas en las características de los elementos.** Se puede elegir el tipo que deben tener los elementos (*testinicial*, *pagina*, *actividad* y *testfinal*), un tema determinado del curso al que deben pertenecer, un número de repetición de la visualización, y un valor de tiempo, de nivel o acierto que debe tener o no los elementos.

Restricciones en las reglas. Las restricciones en las reglas van a permitir limitar el tipo de reglas que se van a descubrir. Se pueden especificar dos tipos de restricciones en las reglas:

- **Restricciones subjetivas.** Se puede seleccionar el tamaño máximo de la regla, las condiciones que deben y no deben aparecer en el consecuente y/o antecedente de la regla.

- **Restricciones objetivas.** Se puede seleccionar la medida de evaluación de las reglas y el umbral mínimo que deben superar.

Además todas estas restricciones no son excluyentes, obteniéndose, al utilizarse de forma conjunta, un filtrado mucho mayor en el número de reglas finalmente descubiertas. De esta forma, el diseñador del curso, mientras más restricciones especifique antes de realizar la búsqueda, menor número de reglas obtendrá y estas serán más interesantes.

Las restricciones específicas que se han utilizado en las diferentes pruebas descritas en este capítulo, a partir de los datos de utilización del curso de Linux, han sido:

Se ha restringido el tamaño máximo de la regla a descubrir, habiéndose establecido a un valor de 2. Esto es debido a que se desea descubrir las reglas más

comprensibles, ya que la longitud de la regla es inversamente proporcional a la comprensibilidad [Freitas 2002].

De los tres tipos de inicialización de los datos utilizados en las pruebas del capítulo anterior, a) todos los datos, b) sólo los datos frecuentes y c) sólo un rango de los datos. Se ha elegido para la realización de las pruebas definitivas el tipo c) ya que la utilización de todos los datos produce demasiadas reglas por lo cual queda descartada, debido a que el profesor no sabría discernir cuales son las que realmente le son útiles. Además el tiempo de computo necesario es muy elevado. La utilización de sólo los datos frecuentes, produce un número muy reducido de reglas con altos valores de interés y exactitud. Y aunque parezca que mientras mayor es el soporte de los datos utilizados mejor es el conjunto de posibles reglas descubiertas, esto puede no ser siempre cierto, ya que puede ser fuente de engaños, debido a que los datos muy frecuentes los presentan la mayoría o la totalidad de los alumnos. Además el conjunto de datos frecuentes suele ser muy reducido con respecto al número total de datos, por lo que puede ser no representativo.

Teniendo en cuenta los problemas ocasionados con las opciones anteriores, se ha propuesto la utilización de un rango representativo de los datos, eliminando por un lado los que tienen una muy alta frecuencia (casi todos los alumnos) y por el otro los que tienen una baja frecuencia (muy pocos alumnos). El rango que se ha establecido entre un soporte mínimo de 0.2 y un soporte máximo de 0.9, afecta a un gran un número de datos por lo que resulta más representativo que la utilización de datos frecuentes. Además el número de reglas obtenidas es reducido y con unos valores de interés y exactitud elevados.

Por último, el algoritmo que se ha utilizado para descubrir las reglas de predicción ha sido el GBGP utilizando una aproximación NSGA debido a que produce, en general, mejores resultados que el resto de los algoritmos implementados, como se muestra en el capítulo anterior.

6.2.2 Tipos de Relaciones Importantes Descubiertas

La información descubierta en un proceso de descubrimiento de conocimiento va a depender de la información de partida que se utilice. En el problema concreto de datos de utilización de ASWE, estos datos iniciales pueden ser de tres tipos: información relativa a los tiempos de visualización de las páginas, aciertos y fallos obtenidos en las preguntas y niveles de conocimiento en las distintas actividades y test iniciales y finales (tiempo, acierto, nivel). Por lo tanto, la información descubierta va a hacer referencia a estos tres aspectos de los escenarios de un curso adaptativo basados en web:

- **Tiempo**. El empleado en la realización de las preguntas de las actividades y test iniciales, en la realización completa de los test finales y en la visualización de cada página web de contenido expositivo del curso.

- **Acierto.** Recogido en las distintas preguntas realizadas en el curso. Tanto los test iniciales y finales como las actividades están formadas por preguntas.

- **Nivel.** El obtenido tras realizar las distintas actividades y test que componen el curso. En el curso concreto de Linux se han utilizado 3 niveles distintos: *principiante*, *medio* y *experto*.

En principio, la información descubierta va a hacer referencia a distintos tipos de relaciones entre estos tres aspectos . Una regla podría contener elementos o condiciones de cualquiera de estos tres tipos en el antecedente o consecuente de la regla. Así pues, el formato genérico de los tipos de reglas que se van a descubrir es el siguiente:

SI Nivel|Tiempo|Acierto **Y** … **ENTONCES** Nivel|Tiempo|Acierto

Donde:
Nivel: Es una condición de tipo nivel.
Tiempo: Es una condición de tipo tiempo.
Acierto: Es una condición de tipo acierto.

Figura 6.1: Formato genérico de tipos de regla descubiertos.

Como se puede apreciar en la Figura 6.1, las reglas descubiertas contienen condiciones de tipo acierto, tiempo y nivel, mostrando todas las posibles relaciones que puede haber entre estas tres variables. Para facilitar la interpretación de todas la posibles relaciones, se pueden clasificar en tres tipos de relaciones distintas, basándose en el formato de las reglas de predicción donde sólo puede haber un elemento en la parte del consecuente de la regla.

En esta clasificación se puede, además, hacer una distinción dentro de cada tipo de escenario. Como ya se ha explicado en el capítulo *Diseño e Implementación del ASWE* y para el curso de Linux, existen varios tipos de escenarios.

- **Escenario tipo exposición.** Son páginas donde se realiza una presentación de la información, se caracteriza por no ser muy interactivo. A este tipo de página le llamaremos de contenido.

- **Escenario tipo ejercicio.** Son páginas que permiten evaluar el conocimiento adquirido y pueden ser de distintos tipos:

- Escenario tipo test inicial.
- Escenario tipo actividad.
- Escenario tipo test final.

Por lo tanto, se podría hacer otra clasificación de los tipos de reglas dependiendo del tipo de escenarios. En estas reglas, todas las condiciones tienen que referirse a un mismo tipo de escenario. De manera que habría subtipos de reglas dependiendo del tipo de elemento que componen la regla: contenido, actividad, test inicial y test final.

A continuación se van a detallar todos estos tipos de relaciones y comentar algunos ejemplos de reglas descubiertas a partir de los datos de utilización del curso hipermedia adaptativo basado en web de Linux, y su posible utilización como ayuda para la toma de decisiones sobre la mejora de dicho curso. La mayoría de las reglas de ejemplo que se van a mostrar van a tener un tamaño mínimo, es decir, un solo antecedente y un solo consecuente, para facilitar su compresión.

Una vez detectadas las reglas descubiertas que pudieran ser interesantes para la mejora de los cursos, fueron trasladadas a un grupo de 10 expertos que confirmaron, en su mayoría, las conclusiones obtenidas por dichas reglas.

6.2.2.1 Relaciones sobre Aciertos/Fallos

Este tipo de reglas van a mostrar las relaciones que existen entre el acierto en una determinada pregunta con respecto a otros aciertos, niveles o tiempos obtenidos. Este tipo de regla tiene la forma genérica siguiente:

SI Nivel|Tiempo|Acierto **Y** … **ENTONCES** Acierto

Donde:
Nivel: Es una condición de tipo nivel.
Acierto: Es una condición de tipo acierto.

Figura 6.2: Formato de regla de relación sobre aciertos.

Estas relaciones (ver Figura 6.1) muestran si el acierto o fallo de una determinada pregunta está relacionado con algún otro acierto, nivel o tiempo obtenido en otra actividad, página de contenido o test del curso.

Dentro de las posibles combinaciones de los atributos nivel, tiempo y acierto simultáneamente en el antecedente de este tipo de reglas, se han distinguido tres

subtipos de reglas, Acierto-Acierto, Nivel-Acierto y Tiempo-Acierto. Indicando en cada caso que en el antecedente todos los elementos son de un mismo tipo: Acierto, Nivel o Tiempo. Veamos cada uno de estos tres subtipos.

A. Con respecto a la relación **Acierto-Acierto**, en principio sólo nos va a interesar la relación de Acierto SI en ambos lados, o la de Acierto NO en ambos lados de la reglas.

SI AciertoPregunta=SI **Y** … **ENTONCES** AciertoPregunta=SI
ó
SI AciertoPregunta=NO **Y** … **ENTONCES** AciertoPregunta=NO

Donde:
AciertoPregunta: Es una pregunta de una actividad o test.

Figura 6.3: Formato de regla de relación sobre aciertos tipo Acierto-Acierto.

Esa relación muestra que si se ha acertado en una pregunta o preguntas entonces también se ha acertado en otra, igualmente si se ha fallado en una entonces también se ha fallado en otra. De aquí se puede concluir que al estar ambas preguntas muy relacionadas, puede ser que ambas tengan un enunciado parecido o se refieran al mismo concepto teórico. Si en el antecedente de la regla hubiera más de una pregunta, el resultado es el mismo, indicando que todas las preguntas se pueden estar refiriendo al mismo concepto. El diseñador ante este tipo de relación debería comprobar si son la misma pregunta, en cuyo caso debería de cambiar el enunciado de una de ellas. Si esto no ocurriera habría que comprobar si están clasificadas en el mismo grado de dificultad, pues lo lógico es que las dos tengan el mismo grado de dificultad.

Un ejemplo de regla descubierta de tipo SI-SI es:

Si *ACIERTO.UNIX_INTRODUCCION-ALTA(2) = SI* ***Entonces***
ACIERTO.HISTORIA_INTRODUCCION-BAJA(0) = SI
Factores (Interés = 0.71, Certeza = 0.77, Soporte = 0.59)

Esta regla muestra la relación que existe entre el acierto SI de la pregunta número 2 de la actividad del concepto UNIX que tiene un nivel ALTO en el tema de INTRODUCCION y la pregunta número 0 de la actividad del concepto HISTORIA que tiene un nivel BAJO en el mismo tema. Con respecto a las métricas de evaluación de calidad es una regla muy interesante (interesabilidad de 0.71), exacta (certeza de 0.77) y que afecta a casi el 60% de los alumnos (0.59 de soporte). En este caso particular se consideró que el ítem UNIX_INTRODUCCION (2) debía tener el mismo grado de dificultad que el ítem HISTORIA_INTRODUCCIÓN (0)

Un ejemplo de regla tipo NO-NO es:

Si *ACIERTO.TESTF_XWINDOWS-BAJA(5) = NO* ***Entonces***
ACIERTO.TESTF_XWINDOWS-BAJA(1) = NO
(Interés= 0.50, Factor Certeza= 0.87, Soporte = 0.29)

Esta regla muestra la relación que existe entre el acierto NO de la pregunta número 5 del test final del tema XWINDOWS en un nivel BAJO con el acierto NO de la pregunta número 1 del mismo test final. Con respecto a las métricas de evaluación de calidad es una regla muy interesante (0.50), exacta (0.87) y que afecta casi al 30% de los alumnos (0.29). En este caso particular se comprobó que los dos ítem relacionados eran de similares características, procediéndose a eliminar uno de ellos para evitar la redundancia en el test.

B. Con respecto a la relación **Nivel-Acierto**, en principio sólo nos va a interesar las dos siguientes relaciones:

- nivel EXPERTO y acierto NO en la pregunta.
- nivel PRINCIPIANTE y acierto SI en la pregunta.

SI Nivel=EXPERTO **Y** … **ENTONCES** AciertoPregunta=NO
ó
SI Nivel=PRINCIPIANTE **Y** … **ENTONCES** AciertoPregunta=SI

Donde:
Nivel: Es el nivel de una actividad o test.
AciertoPregunta: Es una pregunta de una actividad o test.

Figura 6.4: Formato de regla de relación sobre aciertos tipo Nivel-Acierto.

Esta regla muestra la relación que existe entre el nivel final obtenido por el alumno y el acierto a una determinada pregunta. Esta relación indica que una pregunta que no han acertado un número importante de alumnos de nivel EXPERTO puede estar mal planteada o no se entiende, pudiendo crear confusión en el alumno.

De la misma forma, si los factores interés, certeza y soporte son altos y se descubre una relación entre el nivel PRINCIPIANTE y una pregunta acertada podemos concluir que la pregunta es demasiado evidente o que las posible respuestas están redactadas con una gran dosis de inocencia.

Un ejemplo de regla descubierta de tipo EXPERTO-NO es el siguiente:

Si *NIVEL.EMULADORES_PROGRAMAS -ALTA = EXPERTO* ***Entonces***
ACIERTO.EMULADORES_PROGRAMAS-ALTA(1) = N
(Interés= 0.69, Factor Certeza= 0.73, Soporte = 0.44)

Esta regla muestra la relación que existe el nivel EXPERTO en el nivel final de evaluación del concepto EMULADORES que tiene dificultad ALTA dentro del tema PROGRAMAS y el acierto NO a la pregunta número 1 de la actividad de evaluación del mismo concepto. Con respecto a las métricas de evaluación esta regla es interesante, exacta y afecta a casi la mitad de los alumnos. En este caso particular se comprobó que el ítem EMULADORES_PROGRAMAS(1) era confuso en el planteamiento de la pregunta y se corrigió el problema.

Un ejemplo de tipo PRINCIPIANTE-SI es:

Si *NIVEL.TESTI_UNIX-ALTA =PRINCIPIANTE* ***Entonces***
ACIERTO.TESTI_UNIX-ALTA(2) = S
(Interés= 0.58, Factor Certeza= 0.74, Soporte = 0.37)

Esta regla muestra la relación que existe entre la obtención de un nivel PRINCIPIANTE en el test inicial del tema UNIX y el acierto en la pregunta número 2 de dicho test inicial. Con respecto a las métricas de calidad esta regla es interesante, exacta y afecta casi al 40% de los alumnos. En este caso particular se comprobó que el ítem TESTI_UNIX-ALTA(2) correspondiente al test inicial de UNIX de grado dificultad ALTA estaba mal clasificado ya que correspondía una dificultad BAJA.

C. Con respecto a la relación **Tiempo-Acierto**, en principio sólo nos va a interesar la relación de tiempo ALTO en una pregunta, y acierto NO en la misma pregunta.

SI TiempoPregunta = ALTO **ENTONCES** AciertoPregunta=NO

Donde:
TiempoPregunta: Es el tiempo de visualización de una pregunta.
AciertoPregunta: Es una pregunta de una actividad o test.

Figura 6.5: Formato de regla de relación sobre aciertos tipo Tiempo-Acierto.

Esta regla muestra la relación entre el tiempo empleado en leer una pregunta y el fallo en la contestación de dicha pregunta. Esta relación está confirmando que la pregunta no está bien formulada o tiene algún tipo de error, ya que no sólo se obtiene un tiempo alto en leerla, sino que además y de forma simultánea se responde incorrectamente. El diseñador ante este tipo de relaciones debe de

corregir estas preguntas, modificando el enunciado si se encuentra el posible error o cambiándola por otra pregunta distinta.

Un ejemplo de regla descubierta de este tipo es:

Si *TIEMPO.TESTF_ADMINISTRACION-ALTA(0) = ALTO* ***Entonces***
ACIERTO. TESTF_ADMINISTRACION-ALTA(0) = N
(Interés= 0.51, Factor Certeza= 0.79, Soporte = 0.27)

Esta regla muestra la relación entre la visualización con tiempo ALTO de la pregunta 0 del test final de grado ALTO del tema Administración y el acierto NO de dicha pregunta. Con respecto a las métricas de calidad de la regla es exacta, pero menos interesante y afecta a un menor número de alumnos que las reglas anteriores. En este caso particular se comprobó que el test correspondiente del concepto ADMINISTRACION era confuso y no estaba bien formulado, y se paso a cambiarlo por otra pregunta de similares características, mejor definida.

6.2.2.2 Relaciones sobre Niveles

Este tipo de reglas van a mostrar las relaciones existentes entre el nivel obtenido en un test o actividad, con respecto a aciertos, tiempos o niveles. Este tipo de regla tiene la forma genérica siguiente:

SI Nivel|Tiempo|Acierto **Y** Nivel|Tiempo|Acierto **Y** … **ENTONCES** Nivel

Donde:
Nivel: Es una condición de tipo nivel.
Tiempo: Es una condición de tipo tiempo.
Acierto: Es una condición de tipo acierto.

Figura 6.6: Formato de regla de relación sobre niveles.

Estas reglas (ver Figura 6.6) muestran las relaciones entre el nivel obtenido para un determinado test o actividad, acierto, nivel, o tiempo de ítem.

Dentro de las posibles combinaciones de los atributos nivel, tiempo y acierto que pueden ocurrir en el antecedente de este tipo de reglas, se han distinguido tres subtipos de reglas, en concreto los subtipos Acierto-Nivel, Nivel-Nivel y Tiempo-Nivel.

D. Con respecto a la relación **Nivel-Nivel**, en principio sólo nos va a interesar la relación de un nivel EXPERTO en ambos lados de la regla, o nivel PRINCIPIANTE en ambos lados de la regla

SI Nivel=EXPERTO **Y** … **ENTONCES** Nivel=EXPERTO
ó
SI Nivel=PRINCIPIANTE **Y** … **ENTONCES** Nivel=PRINCIPIANTE

Donde:
Nivel: Es el nivel de una actividad o test.

Figura 6.7: Formato de regla de relación sobre niveles tipo Nivel-Nivel.

Esta regla (ver Figura 6.7) muestra que los niveles obtenidos en los test o actividades han sido simultáneamente altos o bajos. Esto indica que los conceptos asociados están relacionados. En este caso, el diseñador del curso debería comprobar el contenido de ambos conceptos para ver a qué se debe la relación y optar por:

- unir ambos conceptos en un único concepto,
- colocar ambos conceptos en una misma lección
- asignarles el mismo grado de dificultad
- corregir las reglas de asignación de niveles.

Si los niveles se refieren a test, ya sea iniciales o finales en lugar de actividades, podemos concluir que los temas están relacionados. El diseñador del curso, en este caso, puede unir los temas, o ponerlos uno a continuación del otro. Un ejemplo de regla descubierta de tipo EXPERTO-EXPERTO para el caso de referirse a conceptos es el siguiente:

Si *NIVEL.INTERFAZ_REDES-ALTA = EXPERTO* ***Entonces***
NIVEL.TCPIP_TELNET-MEDIA = EXPERTO
(Interés= 0.57, Factor Certeza= 0.75, Soporte = 0.37)

Esta regla muestra la relación entre la obtención de un nivel EXPERTO en el concepto INTERFAZ_REDES que tiene asociado un grado de dificultad ALTO y la obtención de un nivel EXPERTO en el concepto TCPIP_TELNET que tiene un grado de dificultad MEDIO. En este caso particular se consideró que había que corregir las reglas de asignación de niveles.

Un ejemplo del tipo PRINCIPIANTE-PRINCIPIANTE es:

Si *NIVEL.CREAR_USUARIOS-MEDIA = PRINCIPIANTE* ***Entonces***
NIVEL.BORRAR_USUARIOS-ALTA = PRINCIPIANTE

(Interés= 0.71, Factor Certeza= 0.76, Soporte = 0.51)

Esta regla muestra la relación que existe entre la obtención de un nivel PRINCIPIANTE en el concepto CREAR_USUARIOS que tiene un grado de dificultad MEDIO y la obtención de un nivel PRINCIPIANTE en el concepto BORRAR_USUARIOS que tiene un grado de dificultad ALTO. En este caso particular se consideró que el ambos conceptos debían tener el mismo grado de dificultad y estar unidos en un mismo concepto.

6.2.2.3 Relaciones sobre Tiempos

Este tipo de reglas van a mostrar las relaciones existentes entre el tiempo de visualización de una página con respecto a otros niveles, aciertos o tiempos. Este tipo de regla tiene la forma genérica siguiente:

SI Nivel|Tiempo|Acierto **Y** Nivel|Tiempo|Acierto **Y** … **ENTONCES** Tiempo

Donde:
Nivel: Es una condición de tipo nivel.
Tiempo: Es una condición de tipo tiempo.
Acierto: Es una condición de tipo acierto.

Figura 6.8: Formato de regla de relación sobre tiempos.

Estas relaciones (ver Figura 6.6) van a indicar si el tiempo de visualización de una determinada pregunta o página de contenido, está relacionado con el nivel obtenido para un determinado test o actividad, o está relacionado con el acierto o fallo cometido en una pregunta, o con el tiempo de visualización de otra página o pregunta del curso.

Dentro de las posibles combinaciones de los atributos nivel, tiempo y acierto en el antecedente de este tipo de reglas, se han distinguido tres subtipos de reglas, los subtipos Acierto-Tiempo, Nivel-Tiempo y Tiempo-Tiempo. Indicando que en el antecedente todos los elementos sean de tipo Acierto, Nivel y Tiempo.

Con respecto a la relación **Nivel-Tiempo**, es exactamente la misma que la descrita para la relación Tiempo-Nivel. Esto es debido a que las relaciones descubiertas son bidireccionales, en el sentido de que el antecedente de la regla implica al consecuente y viceversa.

Exactamente igual ocurre con la relación Acierto-Tiempo, ya que es el mismo tipo de relación Tiempo-Acierto ya descrita.

Con respecto a la relación Tiempo-Tiempo, muestra las relaciones entre tiempos de visualización de páginas y preguntas.

SI Tiempo **Y** … **ENTONCES** Tiempo

Donde:
Tiempo: Es una condición de tipo tiempo.

Figura 6.9: Formato de regla de relación sobre tiempos tipo Tiempo-Tiempo.

Esta relación Tiempo-Tiempo, no es una relación que pueda resultar interesante, ya que, en la mayoría de los casos, es fruto del azar.

Capítulo 7
Comentarios Finales

El objetivo de este Capítulo es hacer un compendio del trabajo realizado y expuesto a lo largo de esta Memoria. En primer lugar, se presentará un resumen de todo el trabajo desarrollado. A continuación, se detallarán las conclusiones alcanzadas y se finalizará presentando cuáles son las futuras líneas de investigación que, a nuestro parecer, se suscitan a raíz del trabajo.

CAPÍTULO 7: COMENTARIOS FINALES

7.1 Resumen

En este trabajo se ha presentado una metodología para la mejora de sistemas hipermedia adaptativos educativos basados en web, apoyado en el uso de técnicas de aprendizaje evolutivo, para la extracción de información interesante que puede revertir en dicha mejora. A fin de contrastar la validez de esta propuesta, se ha desarrollado una aplicación de estas características y, sobre ella, se ha construido un curso realizado por una población de alumnos y expertos en la materia durante un cierto tiempo, a fin de obtener la información empleada en el resto de esta investigación.

Se ha propuesto la aplicación de Algoritmos Evolutivos para llevar a cabo la tarea de extracción de conocimiento, mediante descubrimiento de reglas de predicción. En concreto, se ha trabajado en el paradigma de la Programación Genética Basada en Gramáticas, representando cada regla mediante un árbol de derivación de una gramática de contexto libre. Un análisis de las distintas métricas existentes para valorar la calidad de las reglas producidas, revela la necesidad de la aplicación de algoritmos multiobjetivo. En concreto, se han utilizado las aproximaciones MOGA y NSGA. La calidad de los resultados, en función del número de reglas obtenidas, tiempo empleado en la ejecución del algoritmo, y el grado de interés, precisión y comprensibilidad de las reglas, son muy superiores en este caso en comparación con el resto de algoritmos propuestos, que utilizan una única medida o una composición de varias. También se ha comprobado cómo la utilización de técnicas de nichos secuenciales permite reducir notablemente el número de reglas descubiertas manteniendo, e incluso aumentando, la calidad de dichas reglas.

Con respecto a la utilidad práctica de las reglas descubiertas para la toma de decisiones sobre posibles modificaciones que se pueden realizar en ASWEs, se han descrito los distintos tipos de reglas, se han descrito las utilidades que pueden tener para la mejora del curso y se han mostrado ejemplos concretos de reglas descubiertas con el curso de Linux. Finalmente, para facilitar el proceso de descubrimiento de conocimiento se ha desarrollado una herramienta específica que permite realizar el preprocesado de los datos de utilización de los cursos web, el establecimiento de restricciones sobre el tipo de información que se desea descubrir, así como la aplicación de los algoritmos de minería de datos para extracción de reglas y la visualización de las mismas. Además para facilitar la realización de los cambios se ha desarrollado también una herramienta autor para la construcción y modificación de los cursos web.

7.2 Conclusiones

Las conclusiones obtenidas tras el desarrollo de este trabajo han sido las siguientes:

1. Se ha comprobado que el uso de técnicas de extracción de conocimiento, aplicadas a la información de uso que se genera en los ASWEs, permite obtener datos de utilidad para la mejora de los mismos. Dicho conocimiento se ha expresado en forma de reglas de predicción que es una forma de representación del conocimiento muy comprensible para el usuario y que se pueden utilizar directamente en procesos de toma de decisiones como en nuestro problema.

2. Se ha propuesto una metodología para la mejora de dichos sistemas, basada en el uso de la información descubierta mediante algoritmos de aprendizaje evolutivo, que realiza una labor de adquisición de conocimiento sobre los datos de utilización del curso. Esta metodología de mejora se podría aplicar de igual forma en otros tipos de sistemas basados en web con objetivos no educacionales, como comerciales, de ocio o entretenimiento, etc. La metodología propuesta es independiente del dominio de aplicación, y sólo cambiaría en cada caso el tipo de información de utilización del sistema en particular, condicionando por lo tanto el tipo de reglas que se pueden descubrir y las modificaciones de mejora que se le pueden realizar.

3. Se ha desarrollado un sistema ASWE que implementa mecanismos de registro de información, dando soporte a dicha metodología, así como una herramienta autor que asiste a profesores en la elaboración de cursos específicos para este sistema. Utilizando dicha herramienta se ha construido un curso sobre el Sistema Operativo Linux que se ha utilizado como banco de pruebas, capturando la información de utilización y permitiendo posteriormente la realización de modificaciones. Además, se ha desarrollado una herramienta de extracción de conocimiento que automatiza esta tarea, y que permite además la incorporación, casi inmediata, de nuevos algoritmos de descubrimiento de reglas. Debido a que esta herramienta es específica para ASWEs, el profesor o autor del curso puede facilitarle gran cantidad de información sobre el dominio y aumentar de esta forma el interés del conocimiento descubierto.

4. Se ha comprobado que el uso de algoritmos evolutivos, y en concreto los fundamentados en Programación Genética Basada en Gramáticas, constituyen una herramienta muy potente para la extracción de información en el sistema objeto de este estudio, mejorando sensiblemente los resultados obtenidos por otros algoritmos de descubrimiento de reglas

como el Apriori, ID3 y Prism, en términos del tiempo de cómputo, número de reglas total producidas y porcentaje de reglas interesantes, exactas y comprensibles. Uno de los parámetros críticos a tener en cuenta en la configuración de este tipo de algoritmos GBGP ha sido la elección de la función de aptitud, ya que debe contemplar los distintos requisitos de calidad exigibles a las reglas descubiertas. Además la flexibilidad que permite poder expresar los individuos o reglas en GBGP utilizando una gramática, que permitiría al propio profesor cambiar el formato de regla fácilmente y sin tener que cambiar la codificación de los individuos y todos los operadores evolutivos, hace que este enfoque sea más apropiado que otros que han sido utilizados para esta tarea, como los Algoritmos Genéticos.

5. Se ha comprobado que la aplicación de la aproximación multiobjetivo al problema planteado mejora sensiblemente los resultados obtenidos en términos del número total de reglas generadas y del interés, exactitud y comprensibilidad de dichas reglas. En concreto, de las distintas propuestas de algoritmos evolutivos multiobjetivo, la aproximación tipo NSGA es el que produce mejores resultados obteniendo un menor número de reglas con mayor exactitud e interés. Por lo tanto se ha demostrado la superioridad de la utilización de aproximaciones basadas en el frente de Pareto sobre las aproximaciones clásicas basadas en funciones de ajuste en nuestro problema de optimización de reglas utilizando tres criterios de calidad.

6. Se ha comprobado que las reglas generadas, partiendo de unas condiciones y restricciones iniciales concretas especificadas por el profesor o autor del curso, aplicadas sobre los datos de utilización del curso de Linux son reglas interesantes, coherentes en la mayoría de los casos y han servido para mejorar las distintas preguntas o conceptos referenciadas en las relaciones mostradas en las reglas con respecto a su grado de dificultad, su comprensión o el nivel en los que puede clasificar a los alumnos. Se han descrito los distintos tipos de reglas que se pueden obtener y la utilidad que pueden tener cada uno de ellos para la modificación de ASWEs mostrando ejemplos de reglas descubiertas a partir de los datos de utilización de un curso de Linux.

7.3 Líneas de Investigación Futuras

A continuación, se presentarán algunas de las líneas de trabajo que se han suscitado a raíz del trabajo realizado en la presente memoria.

1. **Análisis y desarrollo de nuevas métricas para evaluar el interés de las reglas generadas.** En el Capítulo 2 se ha indicado que existen multitud de métricas para evaluar la calidad de las reglas, algunas de las cuales se han empleado en los algoritmos de descubrimiento de conocimiento implementados. Sin embargo, como se indica en el Capítulo 5, muchos de ellos están relacionados, y resulta difícil encontrar un conjunto de ellos que reflejen fielmente la calidad de las reglas, sobre todo con aspectos subjetivos de las reglas, como el interés. Un estudio preliminar realizado sobre las 36 métricas descritas en el Apéndice C indica que, un análisis en componentes principales (PCA) de dichas medidas produce 3 componentes que almacenan más de un 85% de la varianza de los datos. Considerando que estas variables no están relacionadas, y que se corresponden con direcciones ortogonales del nuevo espacio generado, sería interesante estudiar el empleo de dichas componentes principales como objetivos a optimizar en el algoritmo multiobjetivo. En esta línea de estudio de nuevas métricas, consideramos también interesante la búsqueda de métricas relacionadas con el interés subjetivo que muestran los profesionales por las reglas generadas. En este sentido, existen referencias de AEs [Williams 1999] en los que no existe una función de aptitud, sino que los individuos son valorados por un experto en cada ciclo del algoritmo, aunque este enfoque podría ser inaplicable dado el número de reglas que pueden generarse en una iteración del algoritmo. Sin embargo, una primera aproximación interactiva, en la que el tamaño de población sea pequeño, podría arrojar información que se utilizase en el desarrollo de métricas que permitieran de forma efectiva modelar estas preferencias.

2. **Automatización del proceso de mejora de cursos hipermedia adaptativos basados en web.** Se puede automatizar de forma completa el proceso de mejora de los cursos, de manera que la información descubierta en forma de reglas se aplique directamente sobre éste. De esta forma, se elimina el proceso de toma de decisiones del profesor, siendo el propio sistema el que decide qué modificaciones deben de hacerse para mejorarse. El profesor ya no tendría que analizar las reglas descubiertas para ver qué cambios debe realizar, sino que su labor sería la de aceptar o rechazar los cambios que el propio sistema va realizando para mejorarse. Para poder llevar a cabo esta idea, habría que generar reglas de acción (relacionadas con posibles modificaciones del sistema) que se activarían con la aparición

de eventos predichos mediante las reglas de predicción descubiertas a partir de los datos de utilización. De esta forma, el sistema recomendaría al profesor una serie de reglas de acción junto con la de predicción que la ha provocado, explicando la acción que se quiere realizar, de manera que si las acepta se aplicaría directamente sobre el sistema modificándolo. Este trabajo requeriría investigar en la línea de los sistemas de recomendación aplicados en enseñanza [Zaïne 2002].

3. **Evaluación de la aplicabilidad de cada tipo de regla descubierta.** Debido a que se han encontrando distintos tipos de reglas que pueden ayudar en la toma de decisiones sobre la mejora de los cursos. Se podría hacer un estudio y evaluación sobre la aplicación práctica de cada uno de estos tipos. Para ello se necesitaría disponer de reglas obtenidas con distintos cursos y utilizados con diferentes poblaciones de alumnos. Posteriormente se realizarían los cambios indicados por las reglas sobre dichos cursos y se volverían a ejecutar por otras poblaciones de alumnos distintas. A partir de los nuevos datos de utilización se descubrirían nuevas reglas y se realizaría un estudio comparativo con respecto a las descubiertas en la primera versión de los cursos sin las modificaciones. Los resultados del estudio podrían determinar que tipo de reglas se vuelven a descubrir y cuales no. Además también habría que tener en cuenta los niveles de conocimiento obtenidos por los alumnos en ambas versiones, sin modificación y con modificaciones, de forma que se pueda comprobar el efecto de las modificaciones en la mejora del sistema con respecto al aprendizaje de los alumnos, y así poder determinar que tipo de reglas son las interesantes desde el punto de vista de la adaptación del curso.

Apéndice A
Términos Utilizados

En este apartado se van a describir los principales términos y siglas utilizadas a lo largo de este trabajo. Debido a que todos los términos provienen de palabras inglesas, para cada término se va a mostrar su traducción en español y tanto la palabra inglesa de la que procede como la sigla original que se suele utilizar para referenciar a dicho término.

AES (Adaptive Educative System). Sistemas Educacionales Adaptativos.

AE (Adaptive Engine). Motor de Adaptación.

ASWE (Adaptive Systems for Web-based Education). Sistemas Adaptativos para Educación Basada en Web.

AH (Adaptive Hypermedia). Hipermedia Adaptativa.

AHA (Adaptive Hypermedia Architecture**)**. Arquitectura Hipermedia Adaptativa.

AHAM (Adaptive Hypermedia Application Model**)**. Modelo de Aplicación Hipermedia Adaptativa.

AHS (Adaptive Hypermedia Systems**)**. Sistemas Hipermedia Adaptativos.

AI (Artificial Intelligence). Inteligencia Artificial.

AM (Adaptation Model). Modelo de Adaptación.

CS (Classifier Systems). Sistemas Clasificadores.

CAI (Computer Assisted Instruction). Enseñanza Asistida por Ordenador.

DM (Data Mining). Minería de Datos.

DM (Domain Model). Modelo del Dominio.

DTD (Document Type Definition). Definición de tipo de documento.

ITS (Intelligent Tutoring System**)**. Sistemas Tutores Inteligentes.

EA (Evolutionary Algorithms). Algoritmos Evolutivos.

EC (Evolutionary Computation). Computación Evolutiva.

EP (Evolutionary Programming). Programación Evolutiva.

ES (Evolutionary Strategies). Estrategias Evolutivas.

EMOO (Evolutionary Multi-Objective Optimization). Optimización Multiobjetivo Evolutiva.

GA (Genetic Algorithms). Algoritmos Genéticos.

GBGP (Grammar Based Genetic Programming, GBGP). Programación Genética Basada en Gramática.

GP (Genetic Programming). Programación Genética.

HTML (HyperText Modelling Language**)**. Lenguaje de modelado de Hipertexto.

KDD (Knowledge Discovery in Databases). Descubrimiento de Conocimiento en Bases de datos.

LS (Learning Systems). Sistemas de Aprendizaje.

ML (Machine Learning). Aprendizaje de Máquinas.

MO (Multiobjetive Optimization). Optimización Multiobjetivo.

MOEA (MultiObjective Evolutionary Algorithm**).** Algoritmo Evolutivo Multiobjetivo.

MOGA (MultiObjective Genetic Algorithm**).** Algoritmo Genético Multioobjetivo.

NSGA (Non-dominated Sorting Genetic Algorithm). Algoritmo Genético de Ordenación No dominada

PEA (Parallel Evolutionary Algorithms). Algoritmos Evolutivos Paralelos.

RI (Rule Interestingness). Interés de las reglas.

SN (Sequential Niching). Nichos secuenciales.

STGP (Strong Typed Genetic Programming) . Programación Genética fuertemente tipada.

SGA (Simple Genetic Algorithm). Algoritmo Genético Simple.

UM (User Modeling/User Model). Modelado de Usuario/Modelo de usuario.

UML (Unified Modelling Language). Lenguaje de Modelado Unificado.

VEGA (Vector Evaluated Genetic Algorithm). Algoritmo Genético Vector Evaluado.

WBE (Web-Based Education). Educación basada en el web.

WCM (Web Content Mining). Minería del contenido web.

WDM (Web Data Mining). Minería de datos web.

WSM (Web Structure Mining). Minería de la estructura web.

WUM (Web Usage Mining). Minería de la utilización web.

WWW (World Wide Web**)**. Red de ancho mundial.

XML (eXtensible Markup Language**)**. Lenguaje de marcas extensible.

Apéndice B
Algoritmos Evolutivos

En este apéndice se va a realizar una introducción a los algoritmos evolutivos. Primero se da una visión general de los algoritmos evolutivos, presentando la idea fundamental. Después se describen y distinguen los distintos tipos de algoritmos evolutivos que existen. A continuación se analizan los aspectos más importantes a tener en cuenta al diseñar un algoritmo evolutivo y se presentan algunos aspectos avanzados relativos a los algoritmos evolutivos. Finalmente se trata la optimización multiobjetivo en una sección aparte debido a su importancia en este trabajo.

APÉNDICE B: ALGORITMOS EVOLUTIVOS

B.1 Introducción

En general, cualquier tarea abstracta a realizar puede considerarse como la resolución de un problema que, a su vez, puede verse como un proceso de búsqueda a través de un espacio de soluciones potenciales, que es lo que se conoce como espacio de búsqueda. Cuando el objetivo consiste en encontrar la mejor solución posible, nos encontramos ante un problema de optimización. En otras ocasiones se desea encontrar una solución o un conjunto de soluciones.

Para problemas cuyos espacios de búsqueda son pequeños o simples pueden utilizarse métodos de optimización y búsqueda clásicos, como son los métodos de tipo directo o basados en el gradiente [Deb 1995, Reklaitis et al.1983]. Sin embargo, este tipo de técnicas presentan ciertas limitaciones, entre las que podemos mencionar como más comunes:

- La convergencia a una solución óptima depende de la solución inicial de la que parte la búsqueda.
- Muchos algoritmos son propensos a verse atrapados en un óptimo local.
- Falta de robustez, es decir, un algoritmo eficaz en la resolución de un determinado problema puede no ser adecuado para otro tipo diferente de problema.
- Algunos algoritmos no son capaces de tratar adecuadamente problemas con variables discretas.
- No son propicios a implementaciones en máquinas paralelas.

Debido a esto, para hacer frente a espacios de búsqueda grandes y/o complejos, son necesarias técnicas heurísticas avanzadas, dentro de las que se incluyen los *algoritmos evolutivos* (Evolutionary Algorithm, EA). El paradigma de los EA (también conocido como computación evolutiva) consiste en la utilización de algoritmos de búsqueda estocástica (probabilística) que se basan en la abstracción de ciertos procesos de la teoría de la evolución darwiniana. Bajo la denominación genérica de EA se agrupa un cierto número de técnicas diferenciadas, las cuales parten todas de la misma idea fundamental, pero que suelen distinguirse tanto por sus características propias como por motivos históricos. En la sección B.2 se describe cada una de estas técnicas.

Los EA solventan todos los inconvenientes mencionados más arriba respecto a las técnicas clásicas. Los EA realizan una búsqueda global, ya que trabajan con una población de soluciones candidatas, más que trabajar con una sola solución candidata. Esto, junto con su naturaleza estocástica, reduce la probabilidad de quedarse atascados en óptimos locales e incrementa la probabilidad de encontrar el óptimo global, que depende en menor medida de cuáles sean los puntos de partida

de la búsqueda. La utilización de una población de soluciones propicia las implementaciones paralelas. Son perfectamente capaces de manejar variables discretas. Los EA son de naturaleza robusta, lo que se ve demostrado por su aplicación exitosa a muchos y muy variados problemas [Klösgen & Zytkow 2002].

B.1.1 Inspiración en la naturaleza

En las últimas décadas han ido apareciendo algunas técnicas novedosas de optimización y búsqueda que tienen en común la característica de que han sido inspiradas por sistemas o fenómenos naturales. Podemos mencionar entre estas las redes neuronales [Haykin 1999], el enfriamiento simulado [Kirkpatrick et al. 1983], los sistemas de hormigas [Bonabeau et al. 1999] y los EA. De estos últimos nos ocupamos aquí. La mayoría de los biólogos están de acuerdo en que la principal fuerza conductora que hay tras la evolución natural es el principio propuesto por Darwin de la supervivencia del más fuerte [Dawkins 1976, Eldredge 1989]. En la mayoría de las situaciones, la naturaleza sigue dos sencillos principios:

1. Si por medio de un procesamiento genético se crea un individuo cuyo ajuste al medio está por encima de la media, su periodo de supervivencia será mayor a la media y por lo tanto tendrá más oportunidades que el individuo medio de producir descendencia que cuente con algunos de sus rasgos destacados.

2. Si por contra, nace un individuo cuyo ajuste al medio es inferior a la media, su probabilidad de supervivencia es menor a la media, por lo que acabará siendo eliminado de la población.

Basándose en estos principios, la mayoría de los EA presentan una serie de características básicas:

- Trabajan con una población de individuos (soluciones candidatas) a la vez, en lugar de con una única solución candidata.
- Utilizan un método de selección sesgado en base al ajuste del individuo, es decir, en una medida de calidad que indica cómo de buena es una solución candidata. Así, cuanto mejor es el ajuste de un individuo, más probable es que sea seleccionado en el proceso evolutivo, y por lo tanto más probable será que su material genético pase a las posteriores generaciones de individuos.
- Generan nuevos individuos por medio de un mecanismo de herencia de los individuos existentes en la generación actual. Los descendientes son generados aplicando operadores estocásticos a los individuos de la

generación actual. Los dos operadores fundamentales y más habitualmente usados son el cruce (recombinación) y la mutación. El cruce intercambia parte del material genético entre varios individuos (normalmente dos), mientras que la mutación cambia de manera aleatoria el valor de una pequeña parte del material genético de un individuo.

Estos elementos fundamentales de los EA serán analizados con más detalle en la sección B.3. Este planteamiento de lo que es un EA supone una modelización muy simple y parcial de los procesos evolutivos en la naturaleza; de hecho, la gran brecha que existe entre la evolución desde el punto de vista biológico y el de los EA, ha suscitado numerosas discusiones y propuestas novedosas dentro del campo de los EA [Back & Schwefel 1996, Banzhaf et al. 1998]. En cualquier caso, incluso con un planteamiento elemental como el descrito, los EA constituyen un método robusto y poderoso aplicable a numerosos problemas.

B.1.2 Algoritmo evolutivo genérico

Tal como se ha mencionado al principio de esta introducción, y tendremos ocasión de comprobar en la sección B.2, existen distintas técnicas que se agrupan bajo la denominación genérica de EA. No obstante, a pesar de sus diferencias, puede plantearse un algoritmo evolutivo genérico al que todas estas técnicas se ajustan en gran medida (ver Algoritmo B.1).

```
procedimiento EA
inicio
    t ← 0
    inicializar población P(t)
    evaluar P(t)
    mientras (no condición_terminación) hacer
    inicio
        t ← t + 1
        seleccionar individuos para tener descendencia
        aplicar operadores genéticos a individuos seleccionados
        evaluar descendencia
        actualizar P(t)
    fin
fin
```

Algoritmo B.1: Algoritmo Evolutivo genérico.

Veamos a continuación cada uno de los pasos de este algoritmo:

Al principio se crea una población inicial de individuos, que será la que corresponda a la primera generación (0) del proceso evolutivo. Normalmente, la población se inicializa de manera aleatoria, aunque es posible inicializarla basándose en conocimiento específico del dominio del problema, para tratar de dirigir desde el principio la búsqueda hacia las áreas más prometedoras del espacio de búsqueda.

La evaluación de una población consiste en asignar a cada individuo de la misma un valor de ajuste, que indica el grado de ajuste de ese individuo (solución potencial) con respecto al objetivo que persigue el problema a resolver. Para obtener el valor de ajuste de cada individuo se aplica una función de ajuste al mismo. La elección de la función de ajuste adecuada a cada problema es una de las cuestiones más importantes y decisivas para el éxito o fracaso en la aplicación de un determinado EA.

El proceso evolutivo es de naturaleza iterativa, por lo que se encuentra dentro de un bucle, que está controlado por una condición de terminación, que puede basarse en distintos tipos de criterios:

- Puede estar basada en un contador, fijándose de antemano un número máximo de generaciones o de evaluaciones de la función de ajuste.
- Puede basarse en el grado de evolución del mejor individuo de la población, por ejemplo en base a una medida del gradiente del valor de ajuste a lo largo de un cierto número de generaciones. La idea es tratar de medir cuándo el proceso evolutivo se estanca y no llegan a apreciarse cambios significativos a lo largo de una ventana de generaciones de un determinado tamaño.
- Es también posible fijarse en el grado de diversidad entre los individuos de la población. Si todos los individuos se parecen entre sí, es el momento de terminar el proceso evolutivo.
- También es posible utilizar condiciones de parada compuestas que combinen varios de estos criterios.

La selección consiste en elegir algunos individuos de la población actual para que tengan descendencia. Existen numerosos métodos de selección (ver apartado B.3.2), tanto determinísticos como estocásticos. Todos ellos están basados en el ajuste de los individuos. Dependiendo del esquema de selección utilizado, algunos individuos pueden ser seleccionados más de una vez. El proceso de selección da lugar al nacimiento de un determinado número de copias de los individuos seleccionados.

El siguiente paso consiste en aplicar los operadores evolutivos a los clones generados en por el proceso de selección. La elección de si aplicar determinado

operador o no a cada uno de los individuos seleccionados es estocástica, es decir, que cada operador se aplica con una determinada probabilidad. Los operadores más habituales son la mutación y el cruce (recombinación), que se tratan con más detalle en B.3.3 y B.3.4 respectivamente. El cruce consiste en generar un conjunto de nuevos individuos combinando partes provenientes de un conjunto de individuos padres. La mutación consiste en generar un nuevo individuo mutando aleatoriamente una pequeña parte de un individuo padre.

El último paso consiste en actualizar la población actual, es decir, elegir qué individuos formarán parte de la nueva generación. Esta elección puede hacerse sólo entre los descendientes generados o bien entre el conjunto de individuos formado tanto por los descendientes como por sus padres. En cualquier caso, puede aplicarse un procedimiento determinístico o uno estocástico.

Otro aspecto importante a tener en cuenta que debe mencionarse aquí es que cada solución potencial al problema a resolver deberá representarse mediante algún tipo de estructura de datos para su procesamiento computacional por el EA. Normalmente, cada individuo de la población consiste en una codificación de una solución potencial, es decir, en una representación o codificación de esa solución potencial adecuada para su tratamiento computacional; pero dicha representación no tiene por qué ser una representación fácilmente interpretable por el usuario o directamente utilizable. Para distinguir estos dos planos de representación se utilizan unos términos tomados de la genética. En 1909, Johansen [Johansen 1911] se dio cuenta de la importancia de distinguir entre la apariencia de un organismo y su constitución genética, acuñando los términos *genotipo* (o genoma), que se refiere al ADN de un organismo, y *fenotipo* (o fenoma), que es el conjunto de propiedades observables de un organismo. En EA, el fenotipo es una solución directamente utilizable por el usuario, mientras que el genotipo es la codificación de una solución que manipula el EA. Los operadores evolutivos se aplican a nivel de genotipo, mientras que el cálculo del valor de ajuste se realiza a nivel de fenotipo. Por último es necesario decir que hay ocasiones en que el espacio genotípico y el fenotípico coinciden, es decir, que no siempre es necesaria una codificación de las soluciones potenciales, siendo posible que el EA opere directamente con la misma representación que utiliza el usuario. En B.3.1 se tratan más detenidamente estas cuestiones.

B.2 Tipos de EA

Es posible rastrear el origen de los EA hasta los años 50 [Fraser 1957, Box 1957, Friedberg 1958, Friedberg et al. 1959], y ha dado lugar hasta nuestros días a diferentes técnicas, métodos y aproximaciones. En esta sección se describen brevemente las cuatro metodologías que más atención han recibido en las últimas décadas: programación evolutiva (Evolutnionary Algorithm, EA), estrategias de evolución (Evolutionary Strategies, ES), algoritmos genéticos (Genetic Algorithm, GA) y programación genética (Genetic Programming, GP); limitándonos únicamente a mencionar otras, como los algoritmos culturales [Reynolds 1994] y la optimización de enjambres de partículas [Kennedy et al. 2001]. Aunque son similares a un nivel alto de abstracción, las cuatro técnicas consideradas implementan un EA de una manera diferente. Las diferencias afectan a prácticamente todos los aspectos de los EA, incluyendo la elección de la representación de los individuos, los mecanismos de selección empleados, los operadores evolutivos y las medidas de ajuste. A continuación se analizan las similitudes y diferencias entre los cuatro enfoques. Es importante sin embargo resaltar que, auque se presentará una caracterización de cada una de las técnicas, hay que tener en cuenta que dentro de cada una de ellas existen numerosas variantes, que pueden llegar a ser notablemente distintas entre sí. Además, hoy en día se observa una tendencia a la unificación entre los EA, utilizándose ideas propias de un método en otro, siendo cada vez más borrosa la frontera entre estos enfoques. Por lo tanto, aquí se presentará una visión ceñida a los planteamientos más clásicos y estándar de estos métodos.

B.2.1 Programación evolutiva

La EP [Fogel et al. 1965, Fogel et al. 1966] aparece en los años 60. Inicialmente se utilizó para resolver problemas de predicción de secuencias, haciendo evolucionar una población de máquinas de estado finito. En los últimos años, la EP ha ampliado su ámbito de aplicación, utilizando representaciones específicas del problema en cuestión, así, en problemas de optimización con valores reales, los individuos son vectores de valores reales, y en el problema del viajante de comercio los individuos vienen dados por listas ordenadas. El Algoritmo B.2 corresponde a la EP.

```
procedimiento EP
inicio
    t ← 0
    inicializar población P(t)
    evaluar P(t)
    mientras (no condición_terminación) hacer
    inicio
        t ← t + 1
        seleccionar individuos para tener descendencia
        mutar
        evaluar descendencia
        actualizar P(t)
    fin
fin
```

Algoritmo B.2: Programación Evolutiva.

Tras la inicialización, los *N* individuos son seleccionados para ser padres, y entonces son mutados, produciendo *N* hijos. Estos hijos son evaluados y se eligen *N* supervivientes de entre los *2N* individuos, usando una función probabilística basada en el ajuste (los individuos con mejor ajuste tienen más probabilidades de sobrevivir). Los operadores de mutación que se utilizan en la EP presentan dos características destacadas. En primer lugar, el tipo de mutación depende de la representación usada, que como acabamos de ver, depende directamente del problema en cuestión. En segundo lugar, los operadores de mutación son de naturaleza autoadaptativa. A medida que el algoritmo se acerca al valor óptimo de ajuste, la probabilidad de mutación se va decrementando, es decir, que la probabilidad de mutación no es fija, sino que se adapta según el progreso del procesamiento a lo largo del espacio de búsqueda. Al comparar el algoritmo B.2 con el B.1, la diferencia que salta a la vista es que en la EP no se aplican operadores de cruce, ya que los operadores de mutación que se usan son muy flexibles, pudiendo producir efectos similares a los del cruce si se desea.

B.2.2 Estrategias de evolución

Las ES surgieron en los años 60 en Alemania, enfocados a problemas de optimización hidrodinámica. El trabajo seminal se debe a Rechenberg [Rechenberg 1973], que propuso las ES primigenias con selección, mutación y una población formada por un único individuo consistente en un vector de valores reales. Este individuo se somete a una mutación que consiste en la suma de un valor numérico dado por una distribución normal de media cero y una determinada varianza σ. Se elige entre el padre y el hijo aquel que tiene el mejor ajuste para formar la siguiente generación.

Posteriormente, Schwefel [Schwefel 1981] introdujo el cruce y poblaciones con más de un individuo (en general, μ individuos), que es el planteamiento al cual se corresponde el Algoritmo B.3.

```
procedimiento ES
inicio
    t ← 0
    inicializar población P(t)
    evaluar P(t)
    mientras (no condición_terminación) hacer
    inicio
        t ← t + 1
        seleccionar individuos para tener descendencia
        recombinar
        mutar
        evaluar descendencia
        actualizar P(t)
    fin
fin
```

Algoritmo B.3: Estrategia de Evolución.

Tras la inicialización y evaluación, se selecciona un conjunto de individuos de manera uniformemente aleatoria para ser padres. Dos padres generan mediante una operación de cruce (recombinación) un hijo, que puede verse posteriormente alterado debido a la mutación. En cada generación se generan λ hijos ($\mu < \lambda$). La actualización de la población es determinística y puede hacerse de dos maneras distintas. En la ES-$(\mu + \lambda)$ se eligen los μ mejores individuos de entre los $\mu + \lambda$ padres e hijos, por lo que se conoce como ES elitista. En la ES-(μ, λ) se seleccionan los μ mejores individuos de entre los λ hijos, por lo que se conoce como ES no elitista.

En las ES también es importante la autoadaptación, permitiendo un proceso evolutivo a nivel de los parámetros estratégicos (como la probabilidad de mutación o el tipo de cruce) simultáneo a la evolución de las propias soluciones al problema en cuestión. Aunque es normal utilizar operadores de cruce en las ES, la mutación sigue teniendo una importancia destacada y mayor que la de la recombinación.

B.2.3 Algoritmos genéticos

Los GA [Holland 1975] hacen su aparición en los años 70, siendo en la actualidad el paradigma más desarrollado y popular en el campo de los EA. Inicialmente, los GA se caracterizaron por una representación de los individuos independiente del dominio basada en cadenas de bits. Aunque hoy en día los GA contemplan el uso de diferentes representaciones según el problema a tratar, la influencia de la codificación binaria sigue siendo significativa. En el Algoritmo B.4 se representa el GA genérico.

```
procedimiento GA
inicio
    t ← 0
    inicializar población P(t)
    evaluar P(t)
    mientras (no condición_terminación) hacer
    inicio
        t ← t + 1
        seleccionar individuos para tener descendencia
        recombinar
        mutar
        evaluar descendencia
        actualizar P(t)
    fin
fin
```

Algoritmo B.4: Algoritmo Genético

Tras la inicialización y la evaluación, se procede a la selección de los individuos que serán padres. La selección es probabilística y basada en el grado de ajuste. *N* padres dan lugar a *N* hijos. Dos padres pueden recombinarse para dar lugar a dos hijos. A su vez, cada uno de los hijos puede ser posteriormente mutado. A los operadores de cruce y mutación se les asocian unas probabilidades de aplicación (probabilidad de cruce y probabilidad de mutación). Es interesante observar que en los GA el énfasis en los distintos operadores evolutivos es el opuesto al de las ES: en los GA la mutación suele considerarse un operador secundario, con una probabilidad de aplicación muy inferior a la del cruce.

B.2.4 Programación genética

La GP [Koza 1992] nace como una variante de los GA caracterizada por la utilización de árboles para representar a los individuos. De hecho, el algoritmo correspondiente a la GP es el mismo que el de los GA (ver algoritmo 8.4). Sin embargo, hay ciertas particularidades que han llevado a considerar a la GP como una rama distinguida de las demás dentro del campo de los EA. La representación en forma de árbol da lugar a operadores de cruce y mutación característicos de la GP. Pero la diferencia más importante es que en la GP los individuos representan programas de ordenador (aunque las interpretaciones de lo que es un programa de ordenador pueden ser muy variadas), por lo que los individuos consisten no sólo en estructuras de datos, sino también en operaciones (operadores y funciones). Así, en un individuo árbol, las hojas corresponden con los símbolos terminales (variables y constantes), mientras que los nodos internos se corresponden con los no terminales (operaciones). En general, el conjunto de no terminales debe cumplir dos propiedades: *suficiencia* y *clausura*. La suficiencia hace referencia al hecho de que el poder expresivo del conjunto de no terminales debe bastar para poder representar una solución para el problema en cuestión. La clausura se refiere a que una función u operador debería ser capaz de aceptar como entrada cualquier salida producida por cualquier función u operador del conjunto de no terminales. En la práctica, no es raro encontrarse con situaciones en las que los programas a evolucionar tengan que manejar variables de distinto tipo, lo cual hace que sea difícil de satisfacer la propiedad de clausura, lo cual presenta un problema que no es fácil de resolver.

Una diferencia entre la GP y el resto de EA vistos es que en la GP los individuos son de tamaño variable, es decir, que el tamaño de los árboles varía a medida que sufren cruces y mutaciones; mientras que los individuos de EP, ES y GP son normalmente de tamaño fijo (aunque no siempre). Eso sí, en la GP normalmente se utiliza un tamaño máximo que puede alcanzar cada árbol, que puede venir dado por un número máximo de nodos o una profundidad máxima.

Por último, hay que mencionar que en la GP la preponderancia del cruce sobre la mutación es aún mayor que en los GA, y que si bien la representación en forma de árbol es la más habitual, existen otras posibilidades como la representación lineal y la basada en grafos [Banzhaf et al. 1998].

B.3 Elementos fundamentales

En esta sección se tratan con algo más de profundidad algunos de los elementos más importantes a la hora de diseñar un EA para resolver un determinado problema. La elección de la función de ajuste, el método de selección de los

mejores individuos de la población, los tipos de operadores evolutivos aplicados sobre los mismos y la manera de representarlos resultan determinantes para el éxito o el fracaso de un EA en su búsqueda de la solución o soluciones a un problema dado.

B.3.1 Representación

Una de las decisiones más importantes a la hora de aplicar un EA a un determinado tipo de problema es la especificación del espacio de búsqueda, lo cual se logra definiendo una correspondencia entre los puntos del dominio del problema y los puntos del dominio de la representación interna.

Hay una gran diversidad de opiniones entre los distintos paradigmas de los EA acerca de las estrategias para seleccionar la representación más adecuada. Como se ha visto en la sección B.2, la EP inicialmente utilizó máquinas de estado finito, pasando luego a decantarse por una filosofía en la que la representación se toma del problema a resolver; las ES han utilizado tradicionalmente vectores de valores reales; los GA iniciaron su carrera con una representación basada en cadenas de bits, aunque en los últimos años se han abierto a otras formas de representación adaptadas al problema en cuestión; el tipo de representación más común en GP está basada en el uso de árboles, aunque también se utilizan representaciones lineales (cadenas de sentencias en un determinado lenguaje de programación) o representaciones en forma de grafo.

Cualquier operador evolutivo que pretenda utilizarse debe definirse con una determinada representación en mente. Seguramente, la cuestión más discutida en el campo de los EA acerca de la representación ha sido la referente a la alternativa a la representación binaria y representaciones específicas para los GA. Tradicionalmente, los GA emplearon cadenas de bits de longitud fija para representar los individuos. En teoría, esta representación hace al GA más independiente del problema, ya que pueden aplicarse los mismos operadores evolutivos a cualquier problema, una vez que se ha encontrado una manera de codificar las soluciones potenciales al problema en cuestión en forma de cadenas de bits. No obstante, hay que tener en cuenta que si de esta manera evitamos tener que definir operadores evolutivos específicos, ahora el trabajo recae precisamente en hallar una manera eficaz y eficiente de codificar las soluciones potenciales en forma de cadenas binarias, lo cual puede ser bastante complejo en muchos casos.

Otro de los motivos por los que la representación binaria tuvo tanto peso durante muchos años fue la dependencia de las bases teóricas de los GA, fundamentada en la teoría de los esquemas [Holland 1975], de dicha representación. No obstante, la generalización de Radcliffe de dicha teoría a otras

formas de representación [Radcliffe 1991] facilitó la adopción de otras representaciones en la comunidad de los GA. Hoy en día existe un consenso prácticamente unánime acerca de la conveniencia de utilizar representaciones específicas para cada tipo de problema (y por lo tanto operadores específicos) con los GA [Michalewicz 1996, Goldberg 1989].

Si bien todo este trabajo de investigación y debate ha tenido lugar en el área concreta de los GA, no cabe duda de que ha arrojado una valiosa luz acera de cuestiones relativas a la representación que son aplicables a cualquier tipo de EA.

Aunque se han desarrollado numerosos trabajos experimentales relativos a la representación, los resultados teóricos son escasos (ver 1.4.1). Pueden señalarse los ya mencionados trabajos de Holland [Holland 1975] y Radcliffe [Radcliffe 1991], y los estudios acerca de los paisajes de ajuste [Manderick et al. 1991, Jones 1995]. Sería deseable contar con una base teórica que nos ayude a saber qué constituye una buena representación.

B.3.2 Ajuste y selección

Un aspecto fundamental en el diseño de cualquier EA es la función de ajuste, ya que es la función que debe ser optimizada. Idealmente, debería medir la calidad de un individuo de la manera más precisa posible, teniendo en cuenta las restricciones en cuanto a capacidad de cómputo disponible, a conocimiento acerca del problema a resolver y a los requerimientos del usuario. La función de ajuste tiene una gran importancia en lo que respecta a la eficiencia computacional del EA, ya que normalmente debe evaluarse muchas veces (una vez por cada individuo en cada generación del proceso evolutivo).

Otro aspecto vital en todo EA, íntimamente ligado a la función de ajuste, es el método de selección, que se encarga de elegir los mejores individuos en base a su grado de ajuste. Aunque la idea de la selección es siempre la misma, suele aplicarse de distinta manera en cada tipo de EA. En EP y ES normalmente se trata de seleccionar los mejores individuos para formar la siguiente generación, una vez que se les han aplicado los operadores evolutivos, mientras que en GA y GP lo que normalmente se hace es seleccionar a los mejores individuos para aplicarles los operadores evolutivos; es decir, en EP y ES se hace la selección después de aplicar los operadores evolutivos, mientras que en GA y GP se hace antes.

En cualquier caso, la selección de individuos siempre se hace en base a sus valores de ajuste. Los métodos de ajuste pueden ser determinísticos o estocásticos (probabilísticos). Los ejemplos de selección determinística más conocidos son los de las ES, en los que los individuos que formen la siguiente generación serán los

mejores a elegir bien entre los hijos de la presente generación o bien entre los padres y los hijos de la presente generación (ver 1.2.2). En este apartado nos centraremos en los métodos de selección estocásticos, que se caracterizan porque cuanto mejor sea un individuo (mejor sea su valor de ajuste) más probable es que sea seleccionado. Se presentarán tres enfoques de selección estocástica: proporcional, por ordenación y por torneo.

En la *selección proporcional*, el individuo i-ésimo es seleccionado con una probabilidad directamente proporcional a su valor de ajuste (f_i). Normalmente el tamaño de la población se mantiene constante, y la probabilidad de selección acumulada para todos los individuos de la población debe valer 1. Por lo tanto, la probabilidad de seleccionar al individuo i-ésimo será $\frac{f_i}{\sum_{j=1}^{N} f_j}$ siendo *N* el tamaño de la población.

Una manera de implementar computacionalmente la selección proporcional es simulando el funcionamiento de una ruleta sesgada. A cada individuo de la población le corresponde una ranura de la ruleta, cuyo funcionamiento queda sesgado porque el tamaño de cada ranura es proporcional al ajuste del individuo correspondiente. Para seleccionar un individuo se hace girar la ruleta y se comprueba en qué ranura cae la bola, seleccionándose el individuo correspondiente a esa ranura. La ruleta se hace girar *N* veces en cada generación. Esto significa que en cada generación se espera obtener $f_i / \bar{f}$ copias del individuo i-ésimo, siendo $\bar{f}$ el valor de ajuste medio de la población.

La selección proporcional ha sido utilizada durante mucho tiempo, y hoy en día sigue siendo muy usada. No obstante, presenta ciertos inconvenientes:

1. Supone que todos los individuos tienen valores de ajuste no negativos y que la función de ajuste debe maximizarse, lo cual no tiene por qué ser cierto.
2. Requiere el cálculo de un valor global ($\sum_{j=1}^{N} f_j$), lo cual reduce el potencial del EA para ser implementado paralelamente.
3. Si un individuo tiene un valor de ajuste mucho mayor que el del resto de individuos se harán muchas copias de ese individuo (llamado superindividuo), por lo que llegará a dominar rápidamente toda la

población. Si ese individuo representa un óptimo local, el EA convergerá de manera prematura a una solución subóptima.

4. Si todos los individuos tienen un valor de ajuste similar, el proceso evolutivo degenerará en una búsqueda prácticamente aleatoria.

Estas desventajas han llevado al planteamiento de otros enfoques para la selección. La *selección por ordenación* consiste en dos pasos. Primero todos los individuos son ordenados en base a su valor de ajuste (esta ordenación puede ser ascendente o descendente, dependiendo de si se desea minimizar o maximizar la función de ajuste), evitando así el primer problema antes mencionado. Tras esto, se lleva a cabo un proceso selectivo similar al de la selección proporcional, pero basado en el que la probabilidad de un individuo de ser elegido es directamente proporcional a su índice de ordenación y no a su valor de ajuste. Así, la selección se basa en la bondad de cada individuo relativa al resto de individuos, sin tener en cuenta la magnitud de las diferencias entre los distintos valores de ajuste. Esto solventa los problemas tercero y cuarto. No obstante, la selección basada en ordenación sigue presentando el segundo problema, ya que requiere un cálculo global, la ordenación de todos los individuos.

En la *selección por torneo* se eligen aleatoriamente k individuos de la población (k es un parámetro llamado tamaño del torneo) y se les hace competir. Normalmente, sólo uno de entre los k individuos es elegido como ganador del torneo y por tanto seleccionado. El ganador del torneo puede ser seleccionado de manera determinística o probabilística. Cuando se hace determinísticamente, el ganador es el que tiene el mejor valor de ajuste. Cuando se hace probabilísticamente, cada uno de los k individuos puede ser elegido ganador con una probabilidad directamente proporcional a su valor de ajuste (lo que equivale a una mini-ruleta de tamaño k). Esta filosofía hace que este enfoque sí sea adecuado para implementaciones paralelas, al no requerir ningún cálculo centralizado.

Un concepto relacionado con la selección es la *presión selectiva*, que hace referencia al número de copias que se hacen de cada individuo en el proceso de selección. Cuanto mayor es la presión selectiva, más copias se hacen del mejor individuo de la población. En general, la presión selectiva es mayor en los métodos determinísticos que en los probabilísticos, y entre estos es mayor en la selección proporcional que en la selección por ordenación. En la ordenación por torneo puede ajustarse la presión con el tamaño del torneo (k); cuanto mayor sea, mayor será la presión. Cuanto mayor es la presión selectiva mayor es la *explotación* por parte del EA de la mejor solución encontrada hasta la fecha, es decir, más se concentra el proceso de búsqueda en las soluciones similares a la mejor encontrada hasta ese momento. Para evitar la convergencia prematura a un óptimo local es necesario equilibrar el grado de explotación con el de *exploración*, que se refiere a

la diversificación de la búsqueda para alcanzar diferentes regiones del espacio de búsqueda. La exploración se ve favorecida por los operadores evolutivos (ver 1.3.3 y 1.3.4), especialmente por la mutación.

B.3.3 Mutación

Como acabamos de ver, la selección sirve para enfocar la búsqueda en áreas de ajuste elevado. Si sólo se aplicara la selección, la población nunca tendría otros operadores que los introducidos en la población inicial, por lo que es necesaria la presencia de operadores evolutivos que alteren a los individuos, dando lugar a la exploración de áreas cercanas, como es el caso de los operadores de mutación. La mutación actúa en un único individuo, cambiando una parte del mismo. La mutación puede introducir en un individuo un valor que no estaba antes presente en la población, ayudando así a aumentar la diversidad genética de la población y favoreciendo la exploración de nuevas áreas del espacio de búsqueda.

Los distintos tipos de EA hacen un uso diferente de la mutación. Como vimos en B.2, hay EA que dan más importancia a la mutación que otros. Además, cada tipo de EA aplica distintas formas de mutación, adaptadas a la manera de funcionar del algoritmo. Veamos brevemente las maneras de utilizar la mutación en cada tipo de EA.

B.3.3.1 Programación evolutiva

En EP no se utiliza el cruce, por lo que la mutación tiene una importancia fundamental. La mutación consiste en un cambio aleatorio en la descripción de un individuo. Las mutaciones se aplican normalmente con una probabilidad uniforme, y el número de mutaciones en un individuo puede hacerse fijo o bien elegirse también de acuerdo a una distribución de probabilidad. Hoy en día, la EP suele utilizar operadores de mutación autoadaptativos, en los que la probabilidad de mutación va disminuyendo a medida que la búsqueda se aproxima al valor óptimo de ajuste.

B.3.3.2 Estrategias de evolución

En las ES, la mutación sigue siendo el operador predominante, incluso en los casos en los que también se utiliza el cruce. Originalmente, la mutación consistía en la suma de un valor numérico dado por una distribución normal de media cero y una determinada varianza σ. Para ajustar el valor de σ se utilizaba la llamada *regla*

de 1/5 [Rechenberg 1973], que indica que cuando más de 1/5 de las mutaciones tienen éxito (no tienen éxito), se debe incrementar (decrementar) σ. Posteriormente, las ES introdujeron formas de mutación autoadaptativa más potentes, en las que cada individuo incorpora en su representación el valor de σ, que también está sujeto a los operadores evolutivos [Schwefel 1981].

B.3.3.3 Algoritmos genéticos

Normalmente, en GA la mutación se aplica con una probabilidad fija a lo largo de todo el proceso evolutivo. Esta probabilidad suele ser mucho menor que la de cruce. Cuando se utiliza la representación basada en cadenas de bits de longitud fija, la mutación consiste en invertir el valor de un bit. Cuando la representación es más compleja es necesaria la utilización de algún operador de mutación diseñado específicamente para dicha representación [Michalewicz 1996].

B.3.3.4 Programación genética

Igual que en los GA, la mutación se considera un operador secundario y se aplica por tanto con una probabilidad muy baja, incluso no se ha utilizado en absoluto en distintos trabajos, como por ejemplo en muchos de los llevados a cabo por Koza, el que es considerado el padre de la GP [Koza 1992]. Son posibles muchas formas de mutación en GP, veamos algunas de las más frecuentes cuando la representación utilizada es en forma de árbol, que es la más común.

- *Mutación puntual*: se sustituye un nodo por otro del mismo tipo generado aleatoriamente (hay que recordar que se distinguen nodos hoja y nodos internos – ver B.2.4, B.3.1-).
- *Mutación por colapso*: se selecciona aleatoriamente un nodo interno y se sustituye por un nodo terminal generado aleatoriamente.
- *Mutación por expansión*: se selecciona aleatoriamente un nodo terminal y se sustituye por un subárbol generado aleatoriamente.
- *Mutación de subárbol*: se selecciona aleatoriamente un nodo interno y se sustituye el subárbol que cuelga de él por un nuevo subárbol generado aleatoriamente.

B.3.4 Cruce

El cruce consiste básicamente en el intercambio de material genético (información) entre dos individuos, dando lugar a un nuevo individuo o a otros dos nuevos individuos. El propósito del operador de cruce es doble. Por una parte sirve para

buscar a través del espacio de búsqueda. Por otra parte, esta búsqueda debe llevarse a cabo de manera que se preserve la información representada en los padres, ya que éstos representan buenas soluciones (si no no habrían sido seleccionados como padres). Como puede observarse, estos dos objetivos son opuestos, por lo que es necesario buscar un equilibrio adecuado entre ambos aspectos del operador de cruce para que el EA tenga éxito. Se trata de buscar el equilibrio entre explotación y exploración ya mencionada (ver B.3.2).

De manera similar a lo que ocurre con la mutación, cada tipo de EA aplica distintas formas de cruce, adaptadas a la manera de funcionar del algoritmo. Veamos brevemente las maneras de utilizar el cruce en cada tipo de EA, salvo en la EP, en la que no se utiliza en absoluto.

B.3.4.1 Estrategias de evolución

Aunque el operador fundamental en las ES es la mutación, hoy en día es habitual el uso del cruce también. Los dos tipos más comunes de cruce son el *cruce discreto*, consistente en intercambios aleatorios de variables entre padres y el *cruce intermediario*, en el que se hace la media de los distintos valores de una misma variable en cada uno de los padres (ver [Back & Schwefel 1995]). También se han propuesto operadores de cruce con más de dos padres [Beyer 1995a].

B.3.4.2 Algoritmos genéticos

Normalmente, en GA el cruce se aplica con una probabilidad fija a lo largo de todo el proceso evolutivo. Esta probabilidad suele ser mucho mayor que la de mutación. Veamos los tipos más comunes de cruce en GA:

- **Cruce monopunto:** si cada individuo tiene una longitud l, se toman dos padres y se elige de manera aleatoria un punto de cruce k entre 1 y l y los dos padres se intercambian sus porciones, de manera que dan lugar a dos hijos, uno de los cuales contiene los genes 1 a k de su primer padre y los genes k+1 a l de su segundo padre y el otro tiene los genes 1 a k de su segundo padre y los genes k+1 a l de su primer padre.

- **Cruce multipunto:** es una generalización del anterior, en el que se utilizan varios puntos de cruce en lugar de uno.

- **Cruce uniforme:** cada uno de los genes de los padres se intercambian entre sí con una determinada probabilidad fija y que es la misma para todas las posiciones.

Cada uno de estos tipos de cruce tiene un poder de búsqueda distinto. El menor poder de búsqueda corresponde al cruce monopunto; el del cruce multipunto depende del número de puntos de cruce que se utilicen, pudiendo llegar a ser tan grande como el del cruce uniforme, que es en general el que mayor poder de búsqueda tiene, siendo por lo tanto el más destructivo. Esto es algo a tener muy en cuenta a la hora de buscar el adecuado equilibrio entre explotación y exploración.

B.3.4.3 Programación genética

En GP el cruce es el operador fundamental, no sólo porque es el que más se aplica, sino que es también particularmente importante, ya que a causa de las particularidades de la GP, su utilización implica consideraciones y dificultades que no están presentes en el resto de EA (no en vano, en [Banzhaf et al. 1998] los autores se refieren a este operador como "el centro de la tormenta"). De nuevo nos centramos en la GP con árboles, por ser la más común. El cruce consiste básicamente en intercambiar un subárbol aleatoriamente elegido de un individuo con un subárbol aleatoriamente elegido de otro individuo. Este cruce es un operador no sensible al contexto, es decir, que cuando se mueve un subárbol *S* de un individuo i_1 a otro individuo i_2, se desconoce el contexto en que será colocado *S*. Es posible que *S* contribuya a mejorar la bondad (el grado de ajuste) de i_1, pero es muy probable que esta cualidad de *S* sea dependiente del contexto, es decir, que dependa de los otros nodos que hay en i_1. Cuando se mueve *S* a i_2, probablemente sea colocado en otro contexto, en el que es muy posible que *S* contribuya a empeorar el ajuste de i_2. El problema de que la calidad de un pequeño fragmento de material genético sea dependiente del contexto también puede ocurrir en los demás tipos de EA, pero parece que el problema resulta más serio en el caso de la GP [Angeline 1997].

Estas consideraciones han llevado a plantearse la naturaleza destructiva del cruce convencional. También se han desarrollado trabajos para tratar de comprobar si el cruce convencional puede ser considerado un operador de macromutación, es decir, que produce hijos que son sintácticamente muy diferentes de sus padres y que resulta tan destructivo como el operador de mutación. Los resultados de estos trabajos han confirmado en gran medida estas hipótesis, llevando así a numerosos esfuerzos por encontrar mejoras para el operador de cruce convencional [Banzhaf et al. 1998, Poli & Langdon 1998, Poli et al. 1999, D'haeseleer 1994].

B.4 Aspectos avanzados

En esta sección se presentan de forma somera algunas cuestiones adicionales relativas a los EA que han venido cobrando importancia a lo largo de los últimos años y que en la actualidad son objeto de diferentes investigaciones cuyo objetivo es lograr el avance y mejor comprensión de los EA.

B.4.1 Resultados teóricos

Los distintos tipos de EA se diferencian también por la consideración que han tenido acerca de los aspectos teóricos. De todos ellos, son las ES las que sin duda se llevan la palma en este tipo de consideraciones, ya que han considerado los estudios teóricos desde sus inicios. La naturaleza de las ES permiten estudios teóricos probabilísticos sobre su velocidad de convergencia. Ya desde el principio se propuso una regla genérica que permitía actualizar de manera óptima la probabilidad de mutación a medida que avanza el proceso evolutivo, la regla de 1/5 [Rechenberg 1973, Schwefel 1981] (ver 1.3.3.2). Posteriormente se presentaron análisis probabilísticos de convergencia [Back et al. 1993, Rudolph 1994b]. Beyer ha realizado numerosos trabajos teóricos acerca de la tasa óptima a la que debe progresar el proceso evolutivo, lo cual resulta muy importante para los aspectos de autoadaptación [Beyer 1993, Beyer 1994, Beyer 1995a, Beyer 1995b].

En el campo de los GA tradicionalmente se han destinado más esfuerzos a los estudios experimentales. Los estudios teóricos normalmente han seguido a los experimentales, y su desarrollo sigue considerándose hoy en día insuficiente y ha sido tradicionalmente fuente de polémica.

El concepto fundamental en el que se asientan los algoritmos genéticos es el de *esquema* [Holland 1975]. Un esquema es un patrón o plantilla que define un subconjunto de cadenas que son iguales en ciertas posiciones, es decir, que un esquema hace referencia a un subconjunto de cadenas que presentan cierta similitud. Los esquemas se definen sobre un determinado alfabeto compuesto por un conjunto finito de símbolos junto con el carácter especial '★' que indica que ese símbolo no está determinado, es decir, que puede ser cualquiera de los otros que forman el alfabeto (es como un símbolo comodín).

Originalmente se usó el alfabeto binario para derivar la teoría de esquemas, que es la base teórica de los GA. La teoría de esquemas se basa en que si queremos comprender mejor el dominio del problema, es esencial estudiar la similitud entre subconjuntos de cadenas así como su calidad como solución. De esta manera, pasamos de interesarnos por cadenas individuales a interesarnos por grupos de

cadenas de alta calidad para el problema en cuestión. Con un alfabeto binario un esquema sería una cadena de la forma

$\square\{0,1,\star\}$

Por ejemplo, el esquema H=(1★01★1), representa a las cadenas {(100101), (100111), (110101), (110111)}.

Es evidente que el tamaño del subconjunto de cadenas definido por un esquema aumenta a medida que aumenta el número de veces que aparece el símbolo '★' en dicho esquema. La idea subyacente es la de considerar la búsqueda no a lo largo de un espacio de cadenas individuales, sino a través de un conjunto de esquemas de los cuales las cadenas son instancias. Considerando una representación binaria de longitud L, cada cadena binaria puede ser la instanciación de $\square$ esquemas, por lo que al evaluar la bondad de una cadena estamos revelando de manera implícita mucha información sobre la bondad de sus correspondientes esquemas. Esta idea es una auténtica piedra angular dentro de la teoría de esquemas, y se conoce como *paralelismo implícito*. Cada población de N individuos nos permite estudiar la adaptación de entre $\square$ y $\square$ esquemas. No todos ellos son procesados, ya que dependiendo de la longitud y el orden, unos tendrán más probabilidades de sobrevivir a los operadores de cruce y mutación que otros. Se estima que aunque los GA sólo procesan N cromosomas en cada generación, procesan de manera implícita del orden de $\square$ esquemas [Holland 1975, Goldberg 1989].

Los esquemas que están por encima de la media se conocen como *bloques constructores*. A medida que se aplican los operadores evolutivos a los individuos de las sucesivas generaciones, un cierto número de estos bloques constructores se van afianzando. Además, estos bloques constructores van combinándose juntos debido a la acción de los operadores evolutivos, dando lugar así a otros bloques constructores mejores. Este proceso de construcción basado en bloques o piezas elementales va progresando hasta alcanzar la solución óptima o una cercana a ella. No existiendo ninguna prueba totalmente rigurosa de la convergencia de los GA, la idea expuesta, formalizada como *hipótesis de los bloques constructores* [Goldberg 1989] es la que se esgrime para justificar el éxito de los GA en numerosos problemas y experimentos. Paralelamente se vienen desarrollando diferentes trabajos destinados precisamente a encontrar una prueba matemática formal de la convergencia de los GA a un óptimo global [Eiben et al. 1991, Nix & Vose 1992, Rudolph 1994a, Davis & Principe 1991, Davis & Principe 1993, Cerf 1995, Vose 1991, Vose & Liepins 1991, Whitley 1992].

Aunque desarrollada inicialmente teniendo en cuenta una representación de los individuos en forma de cadenas de bits de longitud fija, la teoría de los esquemas ha sido ampliada a otras representaciones [Antonisse 1989, Radcliffe 1991].

Si bien la teoría de esquemas de los GA ha sido cuestionada acerca de su validez, sigue siendo el punto de partida para los análisis matemáticos de los GA. En el campo de la GP, se ha intentado adaptar la teoría de esquemas y la hipótesis de los bloques constructores desde el principio [Koza 1992], ya que la GP es realmente un tipo de GA que utiliza una representación compleja (normalmente árboles – ver B.2.4 y B.3.1). No obstante, el caso de la GP es mucho más complejo, ya que se utilizan individuos de tamaño variable y el operador de cruce mover material genético de una parte a otra del genoma (en el resto de EA, el cruce intercambia material genético entre individuos pero en la misma posición). Tras la propuesta inicial de Koza, distintos investigadores han analizado la teoría de los esquemas en el ámbito de la GP [Banzhaf et al. 1998].

El aspecto fundamental de la teoría de los esquemas en GP se centra en la medida en que el cruce tiende a perturbar o a preservar los buenos esquemas, es decir, si cumple la hipótesis de los bloques constructores. Todos los análisis empíricos y teóricos del operador de cruce dependen de un modo u otro de su balance entre perturbación y preservación de los esquemas, que es al fin y al cabo el problema del equilibrio entre exploración y explotación ya mencionado en este capítulo (ver B.3).

Por último, es necesario advertir que la mayoría de los análisis teóricos llevados a cabo en el campo de los EA se enfocan sobre modelos simples, que a veces son simplificaciones limitadas. Lo cual nos deja de momento sin teorías lo suficientemente generales y robustas como para ser aplicadas a cualquier EA y cualquier problema. Sin embargo, los resultados teóricos obtenidos hasta el momento constituyen una prometedora base para futuros desarrollos y proporcionan valiosos indicios a la hora de diseñar un EA para resolver un determinado tipo de problema.

B.4.2 Autoadaptación

Ya que los EA implementan la idea de evolución, y ya que la propia evolución ha debido evolucionar para alcanzar su actual estado de sofisticación, es natural pensar en utilizar la idea de adaptación no sólo para encontrar soluciones a un problema, sino también para ajustar el EA a ese determinado problema.

Los EA son famosos por la gran cantidad de parámetros que es necesario fijar para hacerlos funcionar. Estos parámetros están fuertemente interrelacionados entre sí, lo que hace difícil afinar un EA para obtener los mejores resultados posibles. No sólo es necesario elegir el tipo de EA, la representación y los operadores más adecuados para un problema, también se deben elegir los valores para distintos parámetros, como por ejemplo tamaño de la población, número máximo de

generaciones o probabilidades para los operadores, con el objetivo no sólo de que el algoritmo encuentre la solución óptima, sino de que lo haga eficientemente. Esta es una tarea ardua y que requiere mucho tiempo, por lo que se han destinado numerosos esfuerzos a tratar de automatizarla. Distintos investigadores han utilizado diferentes enfoques para encontrar los valores adecuados para los parámetros estratégicos de un EA. El enfoque tradicional ha sido experimentar con un determinado problema, ajustando los parámetros estratégicos a mano en base a esa experimentación, ejecutando el mismo EA para el mismo problema probando distintos valores para cada uno de los parámetros y registrando la calidad promediada de las soluciones obtenidas a lo largo de varias ejecuciones. Es evidente la gran cantidad de esfuerzo y tiempo que implica este enfoque, que además permite obtener conclusiones no extrapolables a otros tipos de EA o problemas.

Otros investigadores tratan de modificar los valores de los parámetros estratégicos durante la ejecución del algoritmo. Es posible hacer esto usando alguna regla (posiblemente heurística), utilizando un mecanismo de retroalimentación basado en el estado actual del proceso evolutivo o emplear algún mecanismo autoadaptativo.

Hay que tener en cuenta que estos cambios pueden efectuarse a distintos niveles de granularidad: pueden afectar a un componente de un individuo, a un individuo completo o a toda la población.

La autoadaptación, basada en la evolución de la evolución, fue introducida de manera pionera en las ES, donde ha tenido gran importancia prácticamente desde sus inicios, al igual que en la EP. En los GA y la GP, ha sido incorporada de manera algo más tardía y menos generalizada, ya que en estas dos últimas áreas, lo normal siguen siendo las representaciones, operadores y parámetros estáticos.
Las principales áreas en las que puede ser aplicada la autoadaptación son:

- Representación de los individuos. La representación utilizada para los individuos puede variar a lo largo del proceso evolutivo. Algunos trabajos desarrollados en esta línea son [Shaefer 1987, Schraudolph & Belew 1992, Goldberg et al. 1991, Whitley et al. 1991].
- Operadores. Distintos operadores pueden jugar papeles diferentes en las sucesivas etapas del proceso evolutivo [Schaffer & Morishima 1987, Spears 1995].
- Parámetros de control. La mayoría de los trabajos a este respecto se han centrado en la probabilidad de aplicación de los operadores [Davis 1989, Julstrom 1995, Srinivas & Patnaik 1994].

La necesidad de buscar los parámetros estratégicos óptimos a la hora de aplicar un EA a un determinado problema, ya sea mediante mecanismos autoadaptativos o de cualquier otra manera, es consecuencia de la falta de una base teórica sólida que nos permitiera determinar dichos valores óptimos a priori. No obstante, las técnicas de autoadaptación son también necesarias en la medida en que los EA se aplica en la actualidad a problemas cuyo espacio de búsqueda son complejos y varían con el tiempo.

B.4.3 Algoritmos evolutivos paralelos

La eficiencia en tiempo de computación es el talón de Aquiles de los EA, ya que debido a su filosofía es necesario manejar un conjunto de soluciones (la población), que además deben atravesar un proceso iterativo (la evolución a través de una serie de generaciones). Normalmente hay que llevar a cabo un cierto procesamiento por cada individuo de la población en cada una de las generaciones. Este problema puede ser mitigado en parte gracias a implementaciones paralelas o distribuidas, lo cual da lugar a los *algoritmos evolutivos paralelos* (Parallel Evolutionary Algorithms, PEA). Se distinguen dos aproximaciones a la paralelización de EA, la *paralelización estándar* y la *difusión*, además de un enfoque *híbrido* que combina a los dos anteriores.

La paralelización estándar o paralelización global consiste en implementar un EA secuencial en una máquina paralela. El EA en sí no sufre ningún cambio. Se mantiene una única población y la selección y/o la aplicación de los operadores evolutivos se lleva a cabo en paralelo. Una manera simple de lograr esto es paralelizar el bucle que crea una nueva población a partir de la anterior (ver algoritmo B.1). La mayoría de los pasos en este bucle pueden hacerse en paralelo (evaluación, cruce, mutación). La selección puede requerir algún cálculo global (ver B.3.2), lo cual supondrá un cuello de botella para la paralelización. Se utiliza una arquitectura maestro-esclavo en la que el procesador maestro supervisa la población y hace la selección, mientras que los procesadores esclavos aplican los operadores a los individuos seleccionados y los evalúan, para luego devolverlos al procesador maestro.

La paralelización por difusión se caracteriza porque la población está totalmente distribuida. Dentro de este enfoque se distinguen a su vez dos tipos principales: *PEA de grano grueso* y *PEA de grano fino*.

Los PEA de grano grueso son los más populares y que más atención han recibido tradicionalmente. La población se divide en varias subpoblaciones (llamadas *demes*), cada una de las cuales es asignada a un procesador. Cada procesador ejecuta un EA secuencial sobre su subpoblación. Cada subpoblación se

mantiene aislada la mayor parte del tiempo, aunque ocasionalmente intercambian individuos. A este intercambio de individuos se le llama migración, por lo que a este modelo de PEA se les llama también *modelo de migración* o *modelo de isla* (porque cada procesador con su subpoblación es como una isla). A veces también se les llama *EA distribuidos*, porque es habitual su implementación distribuida, más que paralela. Normalmente, se ejecuta el mismo EA en todas las islas, aunque no tiene por qué ser así necesariamente [Lin et al. 1994, Potts et al. 1994, Adamidis & Petridis 1996].

Hay que hacer notar que los PEA requieren algunos parámetros adicionales a los habituales en todo EA. En el caso de los PEA de grano grueso son necesarios el número de islas, el tamaño de cada subpoblación, la topología de interconexión entre islas, la frecuencia de migración (cada cuanto tiempo se intercambian individuos entre distintas islas), la tasa de migración (el número de individuos que migran de cada isla), y alguno más.

Los PEA de grano fino se caracterizan porque la población se divide en un gran número de subpoblaciones muy pequeñas. De hecho, el caso ideal es tener un solo individuo por cada subpoblación. Se define un vecindario para cada procesador, de manera que la comunicación entre procesadores se lleva a cabo siempre dentro de dichos vecindarios. La selección, el cruce y la mutación se aplican sólo a los individuos dentro de un vecindario. Eso sí, como los vecindarios se solapan, a lo largo de sucesivas generaciones, los mejores individuos pueden propagarse a través de toda la población. Aquí son parámetros fundamentales la forma y el tamaño del vecindario.

También existen enfoques híbridos de PEA, en los que se combinan dos o más métodos de paralelización de los vistos antes. Esto implica muchas veces un mayor grado de complejidad. Algunas posibilidades son tener un PEA de grano grueso a un primer nivel y un PEA de grano fino a un nivel inferior. Otra posibilidad es tener un PEA global (un esquema maestro – esclavo) en cada isla o al revés, un PEA maestro – esclavo con un PEA de grano fino a un nivel inferior.

B.4.4 Nichos

Existen problemas multimodales, que son aquellos que se caracterizan porque el espacio de búsqueda contiene varios picos (óptimos). Esta situación puede dar lugar a ciertos problemas. En primer lugar, aunque los EA son menos propensos a caer en óptimos locales que otros tipos de algoritmos como por ejemplo los voraces, esto es algo que puede ocurrir. Un EA puede converger a un óptimo local antes de tener tiempo de explorar otra zona del espacio de búsqueda que contenga el óptimo global.

El segundo problema deriva del hecho de que en ciertas ocasiones nos interesa encontrar varios óptimos, no sólo el óptimo global. En estos casos el objetivo no consiste ya en evitar la convergencia local, sino en evitar la convergencia a través del proceso evolutivo. Para ser más precisos, se trata de evitar una convergencia total, en la que todos los individuos convergen a un único óptimo. El objetivo es formar una convergencia "parcial" y proporcional al ajuste, en la que exista una subpoblación estable agrupada alrededor de cada óptimo, siendo el tamaño de cada subpoblación proporcional al valor de ajuste asociado al óptimo correspondiente [Goldberg & Richardson 1987].

La idea descrita se corresponde con la de un conjunto de individuos que se reparten entre varios *nichos*, cubriendo partes diferentes del espacio de búsqueda. La necesidad de mantener la diversidad de la población y de propiciar el descubrimiento de varios óptimos de alta calidad en el espacio de búsqueda ha llevado a proponer distintos métodos para nichos [Mahfoud 1995, Mahfoud 2000], de los cuales vamos a comentar uno de los más comunes, la *compartición de ajuste*.

La idea básica de este método es la *función de compartición* [Goldberg & Richardson 1987, Deb & Goldberg 1989], que proporciona una manera de determinar en qué medida debe degradarse el valor de ajuste de un individuo a causa de la presencia de un vecino a cierta distancia, medida usando una determinada métrica. Aquí se plantea también una analogía con la naturaleza, donde un nicho contiene una cantidad limitada de recursos, de manera que cuantos más individuos hay en el nicho, a cada uno le corresponderá una parte menor de esos recursos. Así, el cometido de la función de compartición es reducir el valor de ajuste de un individuo según el número de vecinos y su cercanía.

B.5 Optimización multiobjetivo

Muchos problemas del mundo real implican la utilización de varias medidas de rendimiento u objetivos, los cuales deben ser optimizados simultáneamente. Hay ocasiones en las que es posible optimizar por separado cada uno de los objetivos y así conocer qué es lo mejor que puede obtenerse para cada una de las dimensiones de rendimiento. Sin embargo, rara vez pueden encontrarse soluciones adecuadas al problema global de esta manera. El rendimiento óptimo con respecto a uno de los objetivos (si es que existe tal óptimo) a menudo implica un rendimiento inaceptablemente bajo para uno o más de los objetivos restantes. En estas situaciones es necesario buscar una solución de compromiso. Una solución aceptable para este tipo de problemas con objetivos contrapuestos debería ofrecer un rendimiento aceptable, aunque posiblemente subóptimo desde el punto de vista

de un único objetivo, en todas las dimensiones objetivo. La definición de aceptable depende del problema y está además sujeto a la subjetividad.

La optimización simultánea de múltiples y posiblemente contrapuestas funciones objetivo se diferencia de la optimización de una única función en que raramente admite una solución única y perfecta. En su lugar, los problemas de *optimización multiobjetivo* (multiobjective optimization, MO) tienden a caracterizarse por una familia de alternativas que deben considerarse equivalentes en ausencia de información relativa a la relevancia de cada objetivo con respecto a los demás. La multimodalidad (múltiples soluciones) puede surgir incluso en el más simple caso de dos objetivos contrapuestos. A medida que los distintos objetivos son más numerosos y más difíciles de caracterizar, el problema de encontrar una solución de compromiso se hace más y más complejo.

Las técnicas convencionales de optimización, como por ejemplo las basadas en el gradiente, así como ciertas técnicas heurísticas como el enfriamiento simulado son difíciles de aplicar a problemas MO, haciendo necesario reformular el problema MO a uno con un único objetivo, antes de aplicar la técnica en cuestión, lo cual no permite buscar de manera adecuada en el complejo espacio de búsqueda multiobjetivo.

Los EA se han aplicado a lo largo de los años típicamente a problemas de optimización, concretamente a problemas de optimización con un solo objetivo. No obstante, esto se ha debido más que nada a una cierta tradición en el uso que se ha venido dando a estas técnicas, más que a una limitación inherente a ellas. De hecho, la naturaleza de los EA los hace propicios para la resolución de problemas MO, ya que múltiples individuos pueden buscar múltiples soluciones de manera simultánea. Además, es bien conocida la robustez y capacidad de los EA para tratar espacios de búsqueda complejos, características que los hacen especialmente adecuados en escenarios MO.

En los siguientes apartados veremos primero una definición algo más formal de lo que constituye un problema MO y después distintas aproximaciones para la utilización de EA en este tipo de problemas.

B.5.1 Planteamiento del problema

La optimización multiobjetivo (MO) trata de optimizar los componentes de una función de coste cuyos valores están formados por vectores. A diferencia de la optimización de un solo objetivo, la solución a este tipo de problemas no está formada por un punto, sino por una familia de puntos conocida como *conjunto Pareto-óptimo*, *conjunto eficiente* o *conjunto admisible*.

Cada punto de la superficie definida por el conjunto Pareto-óptimo es óptimo en el sentido de que no puede conseguirse mejora alguna para uno de los componentes del vector que no implique una degradación para al menos otro de los componentes. Es decir, el conjunto Pareto-óptimo está formado por todos los elementos del espacio de búsqueda que se caracterizan porque sus componentes no pueden mejorarse de manera simultánea.

Consideremos, sin pérdida de generalidad, un problema de minimización de los n componentes $f_k, k = 1,\ldots,n$ de una función vectorial $\mathbf{f}$ sobre una variable vectorial $\mathbf{x}$ en un universo $\mathcal{U}$, donde $\mathbf{f}(\mathbf{x}) = (f_1(x),\ldots,f_n(x))$. Entonces, un vector de decisión $\mathbf{x_u} \in \mathcal{U}$ se dice Pareto-óptimo sii no existe $\mathbf{x_v} \in \mathcal{U}$ para el cual $\mathbf{v} = \mathbf{f}(\mathbf{x_v}) = (v_1,\ldots,v_n)$ domina a $\mathbf{u} = \mathbf{f}(\mathbf{x_u}) = (u_1,\ldots u_n)$, es decir, no existe $x_v \in \mathcal{U}$ tal que $\forall i \in \{1,\ldots,n\}, v_i \le u_i \quad \wedge \quad \exists i \in \{1,\ldots,n\} \mid v_i < u_i$

El *conjunto Pareto-óptimo* está formado por todos los vectores de decisión Pareto-óptimos, mientras que el correspondiente conjunto de vectores objetivo se llama *conjunto no dominado*. No obstante, no es raro utilizar ambos términos de manera indistinta para hacer referencia a ambos conceptos. La noción de Pareto-optimalidad es un punto de partida para la resolución de problemas multiobjetivo, los cuales implican habitualmente elegir una de las soluciones de compromiso del conjunto Pareto-óptimo de acuerdo a alguna información de preferencia. Al mantener una población de soluciones, los EA pueden buscar varios vectores no dominados en paralelo.

B.5.2 Uso de funciones de agregación

Los EA requieren una información escalar de ajuste sobre la que apoyarse en su funcionamiento, por lo que al utilizarlos siempre es necesaria una escalarización de los vectores objetivo. En la mayoría de los problemas no se puede obtener un criterio global a partir de la formulación del problema, por lo que es habitual combinar o agregar de manera artificial los objetivos en una función escalar en base a alguna información propia del problema en cuestión.

Optimizar una combinación de los objetivos tiene la ventaja de que se produce una única solución de compromiso, aplicando el EA tal cual, sin requerir ningún cambio. Pero si la solución óptima obtenida no es aceptable debido a que la función usada excluye aspectos del problema desconocidos al principio o a que los coeficientes de la función de agregación no eran los adecuados, serán necesarias

nuevas ejecuciones del optimizador. Son diversos los enfoques y trabajos que se han propuesto basados en el uso de funciones de agregación:

- **Suma ponderada:** varios investigadores [Syswerda & Palmucci 1991, Jakob et al. 1992, Jones et al. 1993] han utilizado este enfoque en el que se asigna un peso que indica su importancia a cada objetivo. El punto óptimo obtenido será una función de los coeficientes utilizados para combinar los objetivos. El problema con esta aproximación es precisamente cómo determinar los pesos cuando no se dispone de suficiente información sobre el problema.

- **Optimización de vector meta:** se trata de minimizar la distancia en el espacio de objetivos a un vector meta dado. Un ejemplo de este enfoque se encuentra en [Wienke et al. 1992].

- **Logro de metas:** esta técnica, relacionada con la anterior, busca minimizar la diferencia ponderada entre los valores objetivo y las correspondientes metas propuestas de antemano, y ha sido utilizada en [Wilson & MacLeod 1993].

- **Funciones de penalización:** la idea básica de esta aproximación es penalizar el valor de ajuste de un individuo cuando la solución producida viole alguna de las restricciones impuestas por el problema. Este enfoque ha sido utilizado en [Adeli & Cheng 94], sin embargo, el problema es que las funciones de penalización son generalmente dependientes del problema y por lo tanto difíciles de establecer.

B.5.3 Aproximaciones no Pareto basadas en la población

La idea básica que se expone en este apartado consiste en aprovechar el hecho de que los EA utilizan una población para buscar concurrentemente varias soluciones no dominadas en una única ejecución.

El primer trabajo en que se sigue este enfoque es [Schaffer 1985], en el que se propone un sistema llamado *VEGA* (Vector Evaluated Genetic Algorithm), en el que la única diferencia con respecto a un algoritmo genético convencional (ver 1.2.3) radica en la manera en que se lleva a cabo la selección: se genera un cierto número de subpoblaciones haciendo una selección proporcional (ver 1.3.2) de acuerdo a cada uno de los objetivos por turnos. Así, para un problema con q objetivos y una población de N individuos, se formarían q subpoblaciones de tamaño N/q cada una. Estas subpoblaciones son posteriormente mezcladas en una

sola población de tamaño N sobre la que se llevan a cabo los demás pasos del algoritmo genético de la manera habitual.

Sin embargo, tal como se indica en [Richardson et al. 1989], el hecho de mezclar los individuos de todas las subpoblaciones en una sola es equivalente a realizar una combinación lineal de los componentes del vector de ajuste dando lugar a una función de ajuste escalar, en la que los coeficientes dependen de la distribución de la población en cada generación. Esto implica, por una parte, que distintos individuos no dominados recibirían valores de ajuste diferentes, en contra de lo que cabría esperar en base a la definición de no dominancia. Por otra parte, es posible que la población sufra una tendencia a dividirse en diferentes especies, cada una de ellas especialmente ajustada a uno de los objetivos. Esta formación de especies no es deseable, al estar contrapuesta al objetivo de encontrar una solución de compromiso.

En [Fourman 1985] se propone otra manera de realizar la selección usando un algoritmo genético para obtener varias soluciones no dominadas. La selección se lleva a cabo comparando pares de individuos, cada par en base a uno de los objetivos. En una primera versión del algoritmo, el usuario asigna distintas prioridades a cada objetivo y los individuos se comparan con el objetivo de mayor prioridad. Si se produce un empate, se utiliza el segundo objetivo de mayor prioridad, y así sucesivamente. Esta idea se conoce como *ordenación lexicográfica*. Una segunda versión del algoritmo consiste en seleccionar aleatoriamente el objetivo a utilizar en cada comparación.

En [Kursawe 1991] se propone una versión multiobjetivo de las estrategias de evolución (ver 1.2.2). La selección utiliza tantos pasos como objetivos haya. En cada paso se selecciona aleatoriamente (con reemplazo) un objetivo según un vector de probabilidades, que se utiliza para determinar el borrado de una proporción de la población actual. Tras la selección, μ supervivientes se convierten en los padres de la siguiente generación.

Por último, en [Hajela & Lin 1992] se propone un método basado en la suma ponderada, pero incluyendo de manera explícita los pesos dentro del individuo y promoviendo la diversidad entre los individuos mediante la compartición del ajuste (ver 1.4.4 y [Goldberg & Richardson 1987, Deb & Goldberg 1989]). Así se logra que una familia de individuos evolucione concurrentemente para cada combinación de pesos.

B.5.4 Aproximaciones Pareto

En los trabajos comentados en el apartado anterior se trata de impulsar la generación de múltiples soluciones no dominadas, lo cual consiguen con un cierto éxito. Sin embargo, en ninguno de esos trabajos se hace uso directamente del concepto de Pareto-optimalidad (ver 1.5.1), que es precisamente lo que se hace en las propuestas que se revisan en este último apartado.

La asignación de ajuste basada en la Pareto-optimalidad fue propuesta por primera vez en [Goldberg 1989], como una manera de asignar igual probabilidad de reproducción para todos los individuos no dominados de la población. La idea consiste básicamente en realizar una selección por ordenación (ver 1.3.2) en la que se ordenan los individuos de la población asignando un índice 1 a todos los individuos no dominados, los cuales son eliminados de la contienda; entonces se busca un nuevo conjunto de individuos no dominados, a los que se les asigna un índice 2, y así sucesivamente.

En [Fonseca & Fleming 1993] se propone un esquema ligeramente distinto, por el cual el índice correspondiente a cada individuo en la ordenación corresponde al número de individuos en la población actual por los cuales es dominado. La población se ordena en base a los índices obtenidos de esta manera. El ajuste se asigna interpolando, por ejemplo linealmente, desde el mejor hasta el peor individuo, y luego promediándolo entre los individuos con el mismo índice. La selección se hace con un método propuesto en [Baker 1987] llamado *muestreo universal estocástico*.

En [Srinivas & Deb 1993] se emplea un método similar pero basado en la versión de Goldberg de Pareto-ordenación. En [Horn & Nafpliotis 1993] se propone un tipo de selección por torneo (ver 1.3.2) basado en el concepto de Pareto-optimalidad. Además de los dos individuos que compiten en cada torneo, se utiliza un cierto número de otros individuos para ayudar a determinar si los competidores son dominados o no. Finalmente, en [Cieniawski 1993, Ritzel et al. 1994] se propone una selección por torneo basada en el esquema de Pareto-ordenación de Goldberg. Los individuos son ordenados para decidir el ganador de cada torneo binario, lo cual se corresponde con una aproximación a la ordenación total de la población (que es lo que se hace en [Fonseca & Fleming 1993, Srinivas & Deb 1994]).

Hay que mencionar para terminar que, aunque la Pareto-ordenación asigna a todos los individuos no dominados el mismo valor de ajuste, no garantiza que el conjunto Pareto-óptimo sea muestreado de manera uniforme. Cuando en una población están presentes óptimos distintos pero equivalentes, ésta tiende a converger hacia uno de ellos, fenómeno que se conoce como *deriva genética*

[Goldberg & Richardson 1987]. Esto ha dado lugar a la utilización de alguna forma de compartición del ajuste (ver 1.4.4 y [Goldberg & Richardson 1987, Deb & Goldberg 1989]). La utilización de la compartición del ajuste para evitar la deriva genética y para favorecer el muestreo de todo el conjunto Pareto-óptimo por parte de la población fue propuesto en [Goldberg 1989], y adoptado en [Fonseca & Fleming 1993, Srinivas & Deb 1994, Horn & Nafpliotis 1993, Cieniawski 1993], así como en [Hajela & Lin 1992] entre las aproximaciones no Pareto, como hemos visto en 1.5.3.

Apéndice C

Métricas para la Valoración de la Calidad de las Reglas

En este apéndice se va a describir la base de datos que se ha utilizado en el proceso de descubrimiento de reglas. Esta base de datos contiene la información preprocesada de utilización del curso hipermedia adaptativo basado en web de Linux. El apéndice comienza con una breve introducción, después se va a describir el esquema conceptual y posteriormente se obtiene su modelo relacional.

APÉNDICE C:MÉTRICAS PARA LA VALORACIÓN DE LA CALIDAD DE LAS REGLAS

C.1 Introducción

En la actualidad existen un gran número de medidas para la evaluación de reglas [Lavrac et al. 1999]. Aunque estas medidas provienen de distintas áreas como aprendizaje de máquinas, minería de datos, estadística, clasificación, etc. casi todas se pueden calcular en base a la tabla de contingencia. La tabla de contingencia [Yao & Zhong 1999] es una generalización de la matriz de confusión que es la base estándar para las medidas de evaluación de reglas en problemas de clasificación binaria. La tabla de contingencia de un regla arbitraria con la forma $A \rightarrow C$, donde A representa el antecedente y C el consecuente, se muestra en la Tabla C.1.

	A	**¬A**	**Totales**
C	n(AC)	n(¬AC)	n(C)
¬C	n(A¬C)	n(¬A¬C)	n(¬C)
Totales	n(A)	n(¬A)	N

Tabla C.1: Tabla de contingencia de una regla.

Donde:

A: Conjunto de instancias que cumplen que el antecedente de la regla es cierto.
C: Conjunto de instancias que cumplen que el consecuente de la regla es cierto.
¬A: Conjunto de instancias que cumplen que el antecedente de la regla es falso.
¬C: Conjunto de instancias que cumplen que el consecuente de la regla es falso.
AC: Indica la intersección de los conjuntos A y C, es decir $A \cap C$.
n(X): Indica el cardinal del conjunto X, número de instancias.
N: Número total de instancias.

Además de los anteriores valores también se van a utilizar las siguientes probabilidades para el cálculo de las medidas:

p(A): Frecuencia relativa asociada a A, obtenida como $p(A)=n(A)/N$
p(C): Frecuencia relativa asociada a C, obtenida como $p(C)=n(C)/N$
p(AC) = $n(AC)/N$

p(A/C) = *p(AC) / p(C)*
p(C/A) = *p(AC) / p(A)*

A continuación se va a realizar una enumeración de las distintas medidas de evaluación de reglas que se han encontrado en la bibliografía agrupadas por área de investigación en la cual se han aplicado.

C.2 Medidas de Minería de Datos

Las técnicas de minería de datos se están aplicado mucho en la actualidad al descubrimiento de reglas. Pero debido al gran número de reglas que normalmente descubren, se han propuesto diferentes medidas objetivas para cuantificar el interés de estas reglas [Tan & Kumar 2000]. Las medidas más ampliamente utilizadas son el soporte y la confianza ambas basadas en el concepto de soporte de un conjunto de elementos, y que provienen de la tarea de descubrimiento reglas de asociación. Pero existen muchas más medidas que se han aplicado a esta y otras tareas de minería de datos.

- **Soporte (Support)**. El soporte [Agrawal et al. 1993] también denominado frecuencia o generalidad indica el porcentaje de instancias que contienen tanto C como A, y se define como:

  $$Sup(A \rightarrow C) = p(CA) = \frac{n(CA)}{N}.$$

  Indica el porcentaje de instancia a los que se puede aplicar la regla. Normalmente se ha creído que mientras mayor es el soporte, mejor es el conjunto de elementos, pero esto puede no ser siempre cierto, ya que un conjunto de elementos con alto soporte puede ser fuente de engaños debido a que aparecen en la mayoría de las transacciones.

- **Confianza (Confidence)**. La confianza [Agrawal et al. 1993] también denominada exactitud o precisión, indica el porcentaje de instancias que contienen C también contienen a A, y se define como:

  $$Conf(A \rightarrow C) = p(C / A) = \frac{p(CA)}{p(A)}.$$

  Es una medida de exactitud. Pero no satisface la primera propiedad de RI y sólo satisface la tercera para P(A). Por lo que la confianza no es capaz de detectar la independencia estadística ni la dependencia negativa entre los elementos debido a que no tiene en cuenta el soporte del consecuente.

- **Laplace.** La medida de Laplace [Bayardo & Agrawal 1999] es una medida parecida a la confianza. Normalmente utilizada para medir reglas para propósitos de clasificación y se define como:

$$Laplace(A \to C) = \frac{Sup(A \to C)+1}{Sup(A)+2}.$$

Esta medida tampoco satisface el primer principio de RI.

- **Convicción (Conviction)**. La convicción [Brin et al. 1997] es una función de la confianza que se define como:

$$Convicción(A \to C) = \frac{p(A)p(\neg C)}{p(A\neg C)}.$$

Donde el valor 1 significa independencia, los valores entre 0 y 1 significan dependencia negativa y el valor infinito significa exactitud total. No es fácil comparar reglas con esta medida y también es difícil definir un umbral. Además no verifica la tercera propiedad 3 de RI.

- **Interés (Interest).** El interés [Silverstein et al. 1998] también denominado lift , strenght o medida de independencia representa un test para medir la dependencia estadística. Es una medida muy popular del interés de las reglas que se define como:

$$I(A \to C) = \frac{p(CA)}{p(C)p(A)}.$$

El interés verifica las propiedades de RI, pero como la medida de convicción su rango no está limitado y tiene sus mismos inconvenientes. Además, el interés es simétrico, es decir que el interés de A→C es igual al de C→A, de forma que sólo nos mide el grado de dependencia y no la implicación en ambas direcciones.

- **Interesabilidad Tan&Kumar (Interestingness).** La interesabilidad [Tan & Kumar 2000] es otra propuesta de medida de interés que se deriva del interés con una correlación estadística en la región de bajo soporte y alto interés, y se define como:

$$IS(A \to C) = \sqrt{I(A \to C) * \frac{p(CA)}{N}}.$$

La Interesabilidad tiene muchas propiedades deseables a pesar de violar el primer principio de RI, al medir tanto el interés como la significancia de la regla.

- **Interesabilidad Piatetsky-Shapiro (Interestingness P-S).** Es una de las primeras medidas propuestas para medir el interés de las reglas [Piatetsky-Shapiro 1991], fue introducida por Piatetsky-Shapiro como un ejemplo de medida simple que satisface los tres principios fundamentales de RI y se define como:

  $$RI(A \rightarrow C) = p(CA) - p(C)p(A).$$

  La independencia estadística ocurre cuando el valor de esta medida es 0.

- **Fuerza del interés (Interes Strenght).** La fuerza del interés [Gray & Orlowska 1998] evalúa la fuerza de las asociaciones entre conjuntos de elementos en reglas de asociación. Contiene un componenete discriminador que da indicación de la independencia entre el antecedente y el consecuente y otro componente para el soporte de la regla. Se define como:

  $$I = \left(\left(\frac{p(AC)}{p(A)p(C)} \right)^{k} - 1 \right) * \left(p(CA) \right)^{m}.$$

  Donde k y m, son parámetros que dan peso a la importancia de los componentes de discriminación y el soporte respectivamente.

- **Klösgen.** La medida de Klösgen [Klösgen 1996] es otra medida cuantitativa de la regla A→C y se define como:

  $$K(A \rightarrow C) = p(A)^{\alpha} * (Conf(AC) - p(C)).$$

  Donde α es un parámetro introducido por el usuario.

- **Influencia (Leverage).** La influencia [Piatetsky-Shapiro 1991] es otra media introducida por Piatetsky-Shapiro, muy parecida a la anterior y que también cumple los tres principios.Se define como:

  $$Leverage(A \rightarrow C) = p(C/A) - (p(A) * p(C)).$$

  Es la porción de casos adicionales cubiertos por el antecedente y consecuente sobre aquellos esperados si el antecedente y el consecuente son independientes. Toma valores en el intervalo -1, 1. Si es <0 indica que hay una fuerte independencia entre el antecedente y el consecuente. Si es cercano a 1 indica que la regla es importante. Cuando es >0 A está positivamente correlada a C, y si es <0, entonces está negativamente correlada.

- **Quinlan.** La medida de Quinlan [Quinlan 1987] es una medida propuesta por Quinlan para podar árboles de decisión, pero que se utiliza para sustituir la confianza en minería de datos [Freitas 1999] y su valor es:

  $$Quinlan(A \rightarrow C) = \frac{n(CA) - 1/2}{n(A)}.$$

Permite eliminar el problema de la confianza que está muy sesgada hacia el descubrimiento de pequeñas disyunciones (por ejemplo, reglas que cubren poco número de elementos). Para ello resta ½ al numerador. Penalizando a las reglas que cubren muy pocos ejemplos.

C.3 Medidas de Estadística

Existen varias medidas estadísticas que se pueden utilizar en la evaluación de reglas, sobre todo para medir la inferencia de dependencias entre variables en datos, ya que es un área muy bien estudiada en estadística. Aunque la medida más utilizada suele ser la ganancia o entropía, existen otros estadísticos para medir las dependencias entre variables categóricas [Tan & Kumar 2000].

- **Chi-cuadrado (Chi-squared).** La medida Chi-cuadrado [Silverstein et al. 1998] o estadístico de Pearson, que se suele escribir como χ^2, es un procedimiento de test que mide la discrepancia entre el ejemplo observado de su distribución esperada bajo la hipótesis nula. Se define como:

  $$\chi^2(A \to C) = \frac{N(n(AC)n(\neg A\neg C) - n(A\neg C)n(\neg AC))^2}{n(A)n(\neg A)n(C)n(\neg C)}.$$

  Es una medida muy útil que mientras mayor es su valor más evidencia tenemos para rechazar la hipótesis de independencia. Pero no nos dice la fuerza de la correlación entre los elementos. Se utiliza para encontrar dependencias entre elementos. Sin embargo el valor estadístico no es útil para medir el grado de dependencia.

- **Coeficiente de Correlación (Correlation Coefficient).** El coeficiente de correlación [Tan & Kumar 2000] es una medida de asociación que mide el grado de linealidad entre un par de variables aleatorias. Teóricamente se define como la covarianza entre dos variables, dividida entre sus desviaciones estándar.

  $$\rho_{AB} = \frac{Cov(A,B)}{\sigma_A \sigma_B}.$$

  Pero lo podemos expresar en función de la matriz de contingencia como:

  $$\rho_{AB} = \frac{n(AC)n(\neg A\neg C) - n(\neg AC)n(A\neg C)}{\sqrt{n(A)n(C)n(\neg A)n(\neg C)}}.$$

  Su rango es entre -1 y 1. Si las dos variables son independientes su valor es 0. Este coeficiente está estrechamente relacionado con el estadístico chi-cuadrado. Tanto que a su cuadrado se le denomina en literatura estadística como el coeficiente de determinación.

- **Asociación Predictiva (Predictive Association).** La asociación predictiva [Tan & Kumar 2000] es una medida estadística de asociación que mide la implicación de la regla y se denota por

$$\lambda_C = \frac{P(\in_C) - P(\in_C / A)}{P(\in_C)}.$$

y que se puede reescribir como:

$$\lambda_A = \frac{\sum_j \max_k n(C_k, A_j) - \max_k n(C_k)}{N - \max_k n(C_k)}.$$

- **Entropia o incertidumbre (Uncertainty or Entropy).** La medida de entropía de Shanon [Tan & Kumar 2000] o medida de incertidumbre, es otra medida de asociación originada de la teoría de la información. La entropía de una variable es:

$$H(A) = -\sum_k p(A_k) \log_2 p(A_k).$$

Para dos variables se puede extender como:

$$H(A,C) = -\sum_j \sum_k p(A_k B_j) \log_2 p(A_k B_j).$$

La medida global de asociación entre dos variables se puede expresar en términos del ratio de información mutua máxima.

$$S(A,C) = \frac{H(A) + H(C) - H(A,C)}{\min[H(A), H(B)]}.$$

- **Factor de Certeza (Certainty Factor).** El factor de certeza [Shortliffe & Buchanan 1975] es una medida que se desarrollada para representar incertidumbre en reglas de sistemas expertos y actualmente se está aplicando en minería de datos.

$Si \quad Conf(A \to C) > Sup(C)$	$CF = \frac{Conf(A \to C) - Sup(C)}{1 - Sup(C)}$
$Si \quad Conf(A \to C) < Sup(C)$	$CF = \frac{Conf(A \to C) - Sup(C)}{Sup(C)}$
$Otro \quad caso$	0

El factor de certeza toma valor entre -1 y 1 y se interpreta como una medida de la variación de la probabilidad de que C esté en una transacción cuando se consideran las transacciones donde A está. De forma que un valor positivo de CF mide el decrecimiento de la probabilidad de que C no esté en la transacción dado A. Interpretación análoga se da para valores negativos de CF. CF además tiene en cuenta tanto la confianza de la regla como el soporte de C. Además

verifica las propiedades RI. Algunos autores [Delgado et al. 2001] la han utilizado en lugar de la confianza y han denominado como reglas fuertes ha aquellas reglas que su soporte y CF superan unos umbrales mínimos.

- **Gini.** El índice de Gini [Breiman et al. 1984] es otra medida del interés de las reglas de asociación. Su valor varía entre 0 (independencia completa) y 0.5 (correlación perfecta). Trata de igual forma la correlación positiva y negativa de las reglas.

$$Gini = p(A)(p(C/A)^2 + p(\neg C/A)^2) + p(\neg A)(p(C/\neg A)^2 + p(\neg C/\neg A)^2) - p(C)^2 - p(\neg C)^2$$

- **Función Ganancia (Gain Function)** La función ganancia [Fukuda et al. 1996] de Fukuda se define como:

$$gain(A \rightarrow C) = p(AC) - \theta * p(A).$$

Donde θ es una constante fraccionaria entre 0 y 1.

- **J-medida.** La J-medida [Smythe & Goodman 1992] mide el contenido de información medio de una regla de clasificación probabilística y se utiliza para encontrar las mejores reglas relacionadas con atributos discretos.

$$J(A \rightarrow C) = p(C)\left[p(A/C)\log_2\left(\frac{p(A/C)}{p(A)}\right) + (1 - p(A/C)\log_2\left(\frac{1-p(A/C)}{1-p(A)}\right)\right]$$

Donde valores altos de la Jmedida son deseables, pero no necesariamente están asociados con la mejor regla, ya que condiciones raras pueden estar asociadas a valores altos de Jmedida.

- **J1-medida.** La J1-medida [Alves et. al. 1999] es una modificación de la J-medida para solucionar el problema de la simetría, de forma que sólo utiliza la primera parte:

$$J(A \rightarrow C) = p(C)\left[p(A/C)\log_2\left(\frac{p(A/C)}{p(A)}\right)\right].$$

- **Divergencia (Divergence).** La medida de divergencia [Kullback & Leibler 1951] [Smyth & Goodman 1991] se ideó para calcular la cantidad de información que aporta una regla. Y se define como:

$$H(A \rightarrow C) = p(A) * \left[\frac{p(CA)}{p(A)} * \log_2\left(\frac{p(CA)/p(A)}{p(C)}\right) + \frac{p(\neg CA)}{p(A)} * \log_2\left(\frac{p(\neg CA/p(A))}{p(\neg C)}\right)\right]$$

C.4 Medidas de Aprendizaje de Máquinas

Dentro del área del aprendizaje de máquinas (Machine Learning, ML) se han propuesto multitud de medidas para la evaluación de las reglas descubiertas. Por ejemplo para el descubrimiento de conocimiento descriptivo se utilizan medidas como novedad y descubrimiento de subgrupos. Pero existen muchas más medidas como son [Lavrac et al. 1999]:

- **Fiabilidad Negativa. (Negative Reliability).** La fiabilidad negativa [Lavrac et al. 1999] mide la fiabilidad de las predicciones negativas en problemas de clasificación binaria. Se define como:
  $Neg\,\mathrm{Re}l(A \to C) = p(\neg C / \neg A)$.

- **Sensibilidad (Sensitivity).** La sensibilidad [Lavrac et al. 1999] es la probabilidad condicional de que antecedente sea cierto dado el consecuente cierto. Mide la fracción de casos positivos que son correctamente clasificados en el caso de problemas de clasificación binaria. En términos de lógica de programación mide el cumplimiento de la regla. Se define como:
  $Sens(A \to C) = p(A / C)$.

- **Especificidad (Specificity).** La especificad [Lavrac et al. 1999] es la probabilidad condicional de que el antecedente sea falso dado el consecuente falso. Se utiliza en problemas de clasificación binaria. Se define como:
  $Spec(A \to C) = p(\neg A / \neg C)$.

- **Cobertura (Coverage).** La cobertura [Lavrac et al. 1999] mide la fracción de instancias que cubren el antecedente de la regla. Es una medida de generalidad de la regla. Se define como:
  $Cov(A \to C) = p(A)$.

- **Novedad (Novelty).** La novedad [Lavrac et al. 1999] también denominada innovación, interesabilidad, inusuabilidad o medida de Bhandari. Se define como:
  $Nov(A \to C) = p(CA) - p(C) * p(A)$.

- **Satisfacción (Satisfaction).** La satisfacción [Lavrac et al. 1999] es muy útil para el descubrimiento de conocimiento. Se define como:
  $$Sat(A \to C) = \frac{p(\neg C) - p(\neg C / A)}{p(\neg C)}.$$

- **Informatividad (Informativity).** La informatividad [Lavrac et al. 1999] es una medida parecida a la confianza, pero calculando su logaritmo. Se define como:

$$Inf(A \rightarrow C) = -\log_2(p(C/A)) = -\log_2(\frac{p(CA)}{p(A)}).$$

- **Precisión relativa (Relative Accuracy).** La precisión relativa [Lavrac et al. 1999] también de denominada dependencia es la ganancia relativa a la regla true → C. Otra forma de verla es que mide la utilidad de la conexión entre el antecedente con un consecuente dado. Se define como:

$$RAcc(A \rightarrow C) = p(C/A) - p(C).$$

Al poder salir valores negativos, también se calcula su valor absoluto.

- **Precisión relativa ponderada (Weighted Relative Accuracy).** La precisión relativa ponderada [Lavrac et al. 1999] está relacionada con la generabilidad y exactitud de la regla. Se define como:

$$WRAcc(A \rightarrow C) = p(A) * (p(C/A) - p(C)).$$

Se puede utilizar de medida filtro. Toma el mismo valor que la medida de novedad o innovación. Es una de las medidas fundamentales en la evaluación de reglas.

- **Suficiencia (Sufficiency).** La suficiencia [Kamber et Shinghal 1996] es la medida lógica de suficiencia y se define como:

$$S(A \rightarrow C) = \frac{p(A/C)}{p(A/\neg C)}.$$

Se utiliza para calcular el interés de reglas A → ¬C y ¬A→ ¬C.

- **Necesidad (Necessity).** La necesidad [Kamber et Shinghal 1996] es la medida lógica de necesidad y se define como:

$$N(A \rightarrow C) = \frac{p(\neg A/C)}{p(\neg A/\neg C)}.$$

- **Interés característico (Characteristic Interestingness).** El interés característico [Kamber et Shinghal 1996] mide el interés de reglas de clasificación basándose en la necesidad y la suficiencia. El interés característico, o el interés de una regla característica A->C es:

$$IC = \begin{cases} (1 - N(A \rightarrow C) * p(C)) & 0 \leq N(A \rightarrow C) < 1 \\ 0 & OtroCaso \end{cases}.$$

Varía entre 0 y 1, donde representan el mínimo y el máximo posible respectivamente.

- **Significancia (Significance).** La significancia [Liu et al. 1998] es una medida cuantitativa para la significancia de una regla del tipo A→C. Una primera versión define como el producto de la confianza por el logaritmo del interés. Se define como:

  $$S1(A \rightarrow C) = Conf(A \rightarrow C) * \log_2(I(A \rightarrow C)).$$

  Una segunda versión define como el producto del soporte por el logaritmo del interés:

  $$S2(A \rightarrow C) = Sup(A \rightarrow C) * \log_2(I(A \rightarrow C)).$$

- **Riesgo relativo (Relative Risk)**. El riesgo relativo [Ali et al. 1997] se define como:

  $$R(A \rightarrow C) = \frac{p(C/A)}{p(C/\neg A)}$$

Apéndice D
Herramienta para la Construcción de Cursos

En este apéndice se va a describir la herramienta autor específica que se ha desarrollado para facilitar la construcción de los cursos hipermedia adaptativos basados en web que se van a utilizar posteriormente dentro de nuestro ASWE.

APÉNDICE D:HERRAMIENTA PARA LA CONSTRUCCIÓN DE CURSOS

D.1 Introducción

La creación de cursos adaptativos específicos para nuestro sistema ASWE puede resultar una tarea laboriosa y no muy fácil para una persona que no conoce el sistema previamente, puesto que en principio ha de hacerse manualmente. La persona, normalmente el profesor o grupo de personas que crearán un curso debe de tener conocimiento sobre páginas XML, además de saber la especificación completa para este tipo de cursos, tienen que saber la forma de introducir la información del curso y organizarla adecuadamente. También debe conocer su estructura y los ficheros necesarios para que un curso pueda visualizarse adecuadamente. Estos archivos pueden ser de contenido, tests y ficheros de evaluación. Toda esta información se debe estructurar en ficheros con formato XML, con lo que se debe conocer a fondo este lenguaje de marcas para la generación de un curso adaptativo.

Para facilitar la construcción de cursos específicos que se van a utilizar dentro del ASWE propuesto se ha desarrollado una herramienta autor [Romero et al. 2003b] completa y específica para nuestro formato de curso adaptativo, con la que el autor del curso pueda construir un curso hipermedia adaptativo desde un interfaz web de una forma fácil e intuitiva y sin la necesidad de escribir código *XML*. Para ello el profesor sólo tiene que ir seleccionando las distintas páginas *HTML* que forman parte del curso e indicar a qué concepto, tema y nivel se refieren (ver Figura D.1) y automáticamente se crearán las páginas *XML* y la estructura del curso. De esta forma el profesor sólo tiene que disponer de una gran cantidad de información en formato *HTML*, que puede provenir de otros cursos web o se pueden desarrollar nuevos con alguna herramienta de edición Web, y posteriormente ir indicando a qué concepto, tema y nivel pertenece cada una de estas páginas. Para la realización de test iniciales, finales y actividades sólo tiene que ir añadiendo para cada concepto el enunciado de las preguntas y las respuestas, indicando cuales son las correctas. Por último para la creación de los ficheros de evaluación se está desarrollando una herramienta con la que el profesor puede especificar las reglas de adaptación de cada tema sin necesidad de tener que escribir manualmente estas reglas. Con la utilización de esta herramienta autor, se consigue una considerable reducción de tiempo a la hora de la creación de un curso adaptativo, puesto que hacerlo manualmente puede llevar semanas, mientras que con la herramienta se puede hacer sólo en horas. También se consigue un curso adaptativo totalmente funcional, libre de errores, con el ahorro de tiempo que supone el no tener que depurar los ficheros XML en busca de errores.

D.2 Descripción de la Herramienta

Para crear un curso con la herramienta autor primero se debe de disponer de páginas HTML con el contenido que van a servir de escenario de tipo exposición para cada uno de los conceptos que van a componer el curso. Estas páginas pueden provenir de cursos HTML ya existentes o se pueden crear con herramientas específicas como FrontPage, DreamWeaver, Hotmetal, etc. El siguiente paso es crear el curso con la herramienta, indicando su nombre, su ubicación y algunos datos referentes a la persona que lo crea o autor. Entonces sólo hay que introducir el nombre de los temas, así como el número de lecciones del tema que se seleccione para introducir información acerca de él. A continuación se especifican cuales son los temas accesibles desde el actual, es decir, los temas que puede visitar un alumno cuando finaliza un tema. Es una forma de establecer la jerarquía del curso. El siguiente paso es introducir el nombre de las lecciones del tema seleccionado, para posteriormente seleccionar el fichero o los ficheros que componen cada concepto o lección. Para ello se pulsa el botón Examinar y se elige el archivo o los archivos que forman la lección uno a uno. Seguidamente se introduce el grado de dificultad de cada lección o concepto. Por ejemplo, si la lección tiene una dificultad baja, se introduce nivel 0, si tiene una dificultad media tendrá nivel 1, si su dificultad es alta, tendrá un nivel 2, etc. (ver Figura D.1).

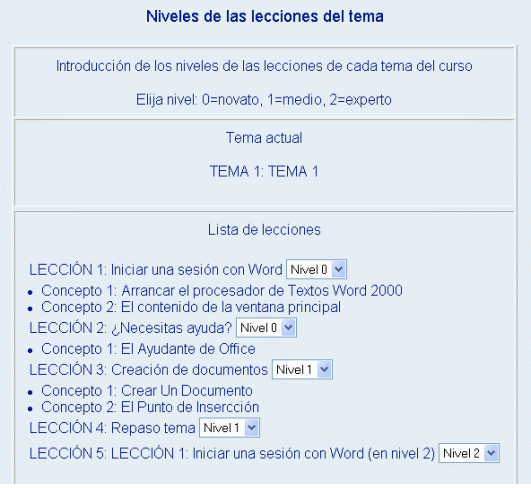

Figura D.1: Página de grados de dificultad de las lecciones de un tema.

El siguiente paso es introducir tanto los tests iniciales, como los tests finales y las actividades para cada grado de dificultad (ver Figura D.2) necesarios para que el curso sea adaptativo. Para ello se muestra una ventana en la que se selecciona el

test que se va a introducir, así como los que ya lo han sido (indicado como Modificados) y los que restan (indicado como Sin modificar).

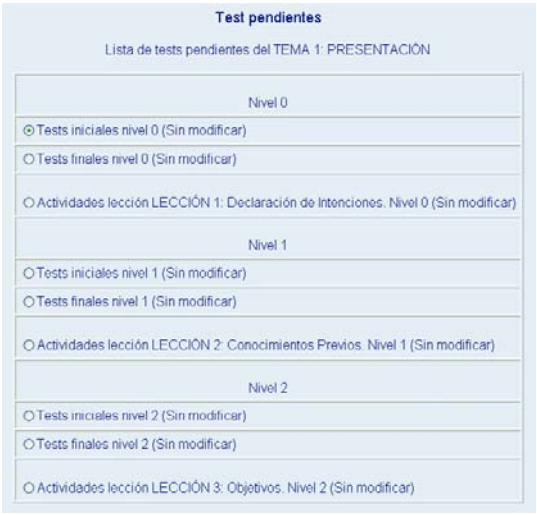

Figura D.2: Página de tests pendientes de un tema.

Para construir un test hay que introducir toda la información necesaria para generarlo, como es el número de preguntas, número de respuestas de cada pregunta, respuestas correctas y si se desea la presencia de comentario o no, al final, junto con la respuesta. En la implementación realizada para cada test es necesario suministrar un número mínimo de preguntas dependiendo del tipo de test, en principio se han establecido unos valores mínimos de 2 preguntas para una actividad, 4 para un test inicial, 6 para un test final. El número de respuestas que puede tener cada pregunta puede variar de 2 a 4, pudiendo existir varias respuestas correctas. En la Figura D.3 se muestra el formulario para la introducción del test.

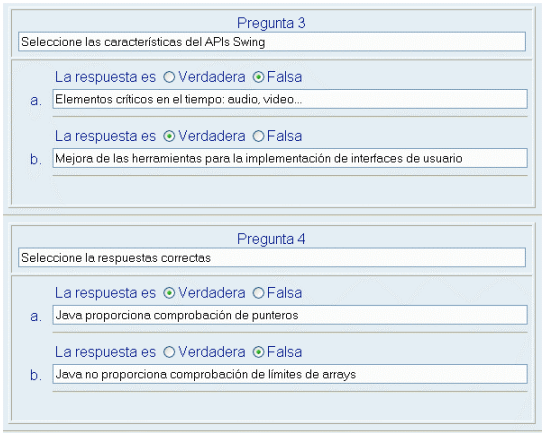

Figura D.3: Página para la creación de un test.

Cuando se hayan introducido todos los tests necesarios, ya se tiene toda la información para el tema. Hay que realizar estas operaciones para todos los temas y cuando se hayan completado todos los que componen el curso, éste estará listo para ser visualizado por nuestro sistema ASWE y ser publicado en la Web para que los alumnos puedan conectarse y realizarlo.

Apéndice E
Herramienta para Descubrimiento de Conocimiento

En este apéndice se va a describir la herramienta específica que se ha desarrollado para facilitar la realización del proceso de descubrimiento de información sobre los datos de utilización de nuestro ASWE.

APÉNDICE E: HERRAMIENTA PARA DESCUBRIMIENTO DE CONOCIMIENTO

E.1 Introducción

En la actualidad existen multitud de herramientas tanto comerciales como de libre distribución para la realización de diferentes tareas de minería de datos, entre ellas el descubrimiento de reglas. De entre todas ellas se pueden destacar DBMiner [Klösgen & Zytkow 2002] y Weka [Witten & Frank 2000] por ser sistemas de dominio público muy populares, tener un entorno gráfico integrado y permitir realizar casi todas las tareas de minería de datos. El principal inconveniente que presentan este tipo de herramientas es que son complejas de manejar para una persona no experta en minería de datos, además de que al ser de propósito general no se le puede realizar un tratamiento específico del conocimiento de los ASWEs.

Debido a estos problemas, se ha desarrollado una herramienta específica [Romero et al. 2003a] que se ha denominado EPRules (Education Prediction Rules) con el objetivo de facilitar el proceso de descubrimiento de reglas de predicción en sistemas educativos basados en web. Se ha implementado en el lenguaje de programación Java y está orientada para ser utilizada por el profesor o autor del curso. La principal característica de esta herramienta es su especialización en educación, utilizando atributos concretos, filtros y restricciones específicas para datos de utilización de los ASWE. Otras características de esta herramienta son:

- Facilidad para añadirle nuevos algoritmos de descubrimiento de reglas programados, con sólo modificar los ficheros de configuración de la herramienta.

- Facilidad de añadir nuevas medidas de evaluación de reglas, con sólo modificar los ficheros de configuración de la herramienta.

- Posibilidad de cambiar de idioma en tiempo real, pudiendo cambiar el lenguaje utilizado en toda la herramienta de español a ingles y viceversa.

- Funciona en cualquier plataforma (Windows, Unix, Mac, etc.) y con cualquier base de datos (MySQL, Oracle, SQLServer, etc.).

A continuación, se va a describir el proceso de descubrimiento de información que implementa la herramienta EPRules y posteriormente se describe la utilización de dicha herramienta.

E.2 Proceso de Descubrimiento de Conocimiento implementado

Utilizando esta herramienta el profesor o autor del curso puede realizar todo el proceso de descubrimiento de conocimiento, desde seleccionar y preprocesar los datos de utilización de los cursos, hasta visualizar las reglas descubiertas al aplicar los algoritmos de minería de datos. El proceso típico de descubrimiento de conocimiento en ASWE que realiza un profesor, a partir de la información de utilización de un curso concreto ya preprocesada y almacenada en una base de datos, es la que se muestra a continuación (Figura E.1):

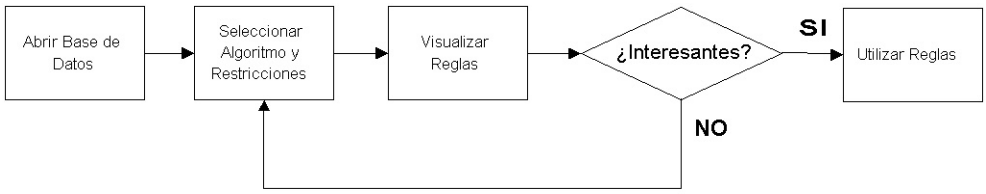

Figura E.1: Proceso de descubrimiento implementado en la herramienta EPRules.

Las etapas de este proceso de descubrimiento de conocimiento son:

- **Abrir Base de Datos.** Consiste en seleccionar la base de datos donde se encuentran almacenados los datos de utilización ya preprocesados del curso que se desea utilizar. Si los datos no se encuentran preprocesados, se debe previamente preprocesar los ficheros *logs* capturados por el curso y almacenarlos en una base de datos.

- **Seleccionar Algoritmo y Restricciones.** Consiste en seleccionar un algoritmo de descubrimiento de conocimiento y sus parámetros específicos, además de las restricciones tanto objetivas como subjetivas que se desean que cumplan las reglas descubiertas.

- **Visualizar las Reglas.** Consiste en visualizar el conjunto de reglas descubiertas, los elementos que forman el antecedente y consecuente de la regla, así como las medidas de evaluación de cada regla.

- **Determinar si las reglas son Interesantes.** Consiste en determinar si el conjunto de reglas descubiertas o un subconjunto son interesantes, tanto por su número, como por su calidad respecto a las diferentes medidas de evaluación, como por su significado semántico.

- **Utilizar las Reglas o volver a aplicar Algoritmo.** Consiste en, o bien utilizar el conjunto de reglas o un subconjunto de las reglas descubiertas, si se consideran suficientemente interesantes para tomar decisiones sobre posibles modificaciones a realizar en el curso, o bien se vuelve a aplicar el algoritmo con distintos parámetros o restricciones para descubrir un conjunto de reglas más interesante que el actual.

Este proceso se realiza de una forma directa desde la herramienta gráfica EPRules desarrollada específicamente para nuestro problema.

E.3 Descripción de la Herramienta

Al ser una herramienta gráfica desarrollada en Java, se necesita tener instalado una máquina virtual de Java para poder ejecutarla. Una vez ejecutada la aplicación aparece su interfaz gráfico (ver Figura E.2) que se compone de cuatro partes principales:

- **Datos de Entrada.** Permite abrir una base de datos ya existente con datos de utilización de un curso o bien crear una nueva, y añadirle nuevos alumnos (ver Figura E.2). Para crearla o añadir datos se deben seleccionar los ficheros de utilización del curso (*ficheros log, model* y *test*) a preprocesar, que pueden ser tanto la información de un solo alumno (opción Seleccionar Ficheros de la Figura E.2) como la de un grupo de alumnos (opción Seleccionar Directorio de la Figura E.2). También se pueden seleccionar y configurar los parámetros del algoritmo de discretización de la variable tiempo si se va a preprocesar los datos.

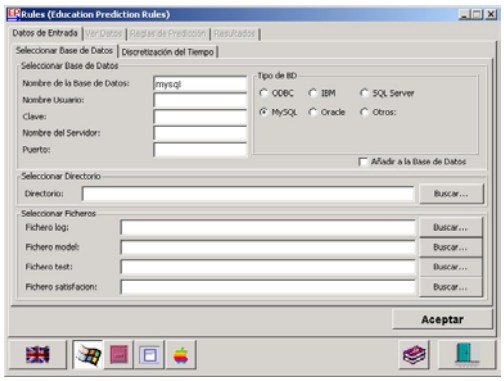

Figura E.2: Herramienta Gráfica EPRules para el Descubrimiento de Reglas de Predicción.

- **Ver Datos.** Permite visualizar todos los datos de utilización de los alumnos del curso y realizar algunas estadísticas básicas (máximos, mínimos, medias, etc.) sobre ellos. Estos datos son sobre los tiempos, aciertos o niveles obtenidos por los alumnos en las distintas páginas Web que componen el curso. Se pueden seleccionar desde visualizar los datos o realizar las estadísticas de todos los alumnos, hasta las de un alumno en concreto, o sólo para un tema determinado del curso, o sobre un concepto determinado, o de un nivel de visibilidad o dificultad de un tema determinado, para un número de repetición determinado, y sólo de un tipo determinado (tiempo, acierto o nivel).

- **Reglas de predicción.** Permite aplicar los distintos algoritmos de descubrimiento de reglas disponibles (ID3, Apriori, Prism y las diferentes versiones de GP), pudiendo seleccionar tanto el algoritmo que se desea utilizar y sus parámetros de ejecución específicos (ver Figura E.3), como las restricciones subjetivas que deben de cumplir las reglas (ver Figura E.4), como las medidas de evaluación objetivas a utilizar como filtrado final de las reglas descubiertas (ver Figura E.5), de manera que las reglas que finalmente se le muestran al usuario le sean realmente de utilidad.

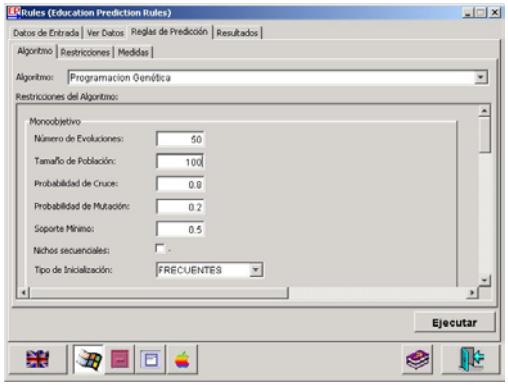

Figura E.3: Pantalla de selección del algoritmo de descubrimiento de reglas y parámetros.

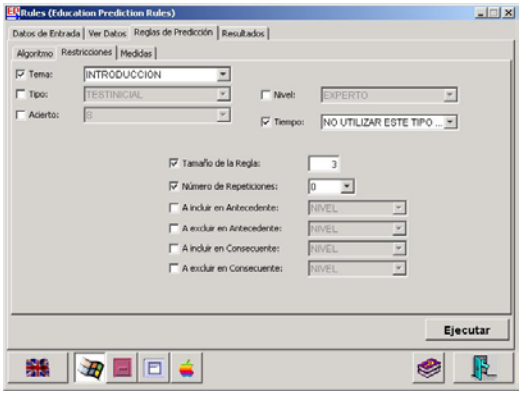

Figura E.4: Pantalla de selección de las restricciones que deben de cumplir las reglas de predicción.

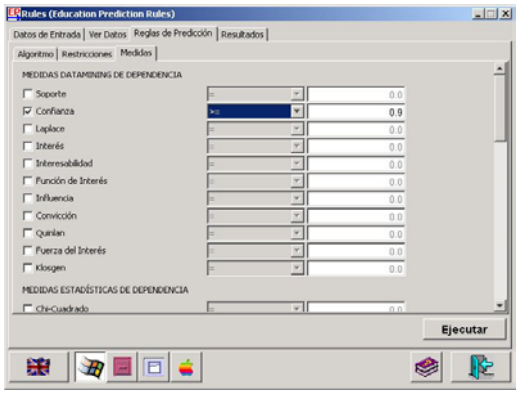

Figura E.5: Pantalla de selección de las medidas de evaluación de las reglas de predicción.

- **Resultados.** Permite visualizar tanto los datos de utilización de los cursos, como los resultados de las estadísticas, como las reglas de predicción descubiertas (ver Figura E.6). En concreto, en el último caso, para cada regla de predicción descubierta se muestran las condiciones que componen el antecedente y el consecuente de la regla, así como todos los valores para cada una de las medidas de evaluación de reglas disponibles. Aunque en principio las reglas se muestran ordenadas por el orden en el que se han descubierto, se pueden ordenar alfabéticamente por una condición del antecedente o por la del consecuente, o numéricamente por el valor de cualquiera de las medidas.

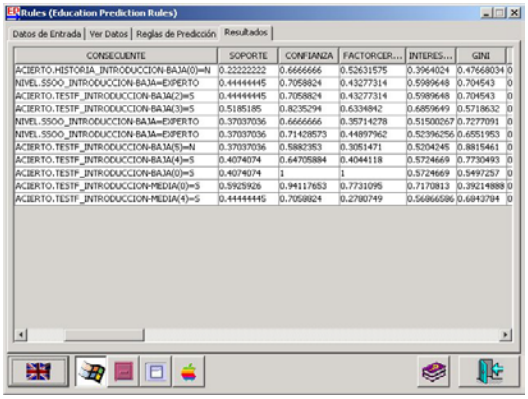

Figura E.6: Pantalla de Resultados con ejemplo de reglas descubiertas.

El tipo de información que se muestra en la herramienta EPRules como información de Resultados (ver Figura E.6) es de dos tipos muy distintos, por un lado la información de datos de utilización y estadística, y por otro la información descubierta en forma de reglas de predicción.

- **Información de datos de utilización y Estadística.** Esta información son los propios datos de utilización de los alumnos o estadísticas básicas de dichos datos. Estas estadísticas hacen referencia a elementos o condiciones individuales y es relativa a la ocurrencia de dichos elementos de forma individual en la base de datos por cada alumno, es decir número de alumnos que cumplen esa condición. Algunos ejemplos son los valores máximos, valores mínimos, el valor medio, etc. para los distintos tipos de atributos, proporcionando una información muy útil para el profesor o educador. De forma que el profesor puede conocer por ejemplo cuál es la página web de tipo exposición o ejercicio con un porcentaje de tiempo de visualización más alto, cuál es la pregunta con un porcentaje de aciertos o fallos más elevado, cuál es el test inicial o final con un porcentaje más alto de obtención de nivel experto o principiante, etc. para cada tema del curso o para el curso en general.

- **Información de reglas de predicción.** Esta información hace referencia a relaciones entre elementos y es relativa a la ocurrencia conjunta de varios elementos o condiciones en la base de datos para cada alumno, es decir el número de alumnos que cumplen varias condiciones específicas simultáneamente. Para descubrir este tipo de información se ha utilizado la tarea de modelado de dependencias [Klösgen & Zytkow 2002] para buscar relaciones entre atributos en forma de reglas de predicción. Con este tipo de información el profesor puede descubrir si el tiempo, acierto o nivel para una página web está relacionado u ocurre de forma conjunta con otro

tiempo, acierto o nivel de otra página web distinta. Utilizando como referencia estas relaciones descubiertas el profesor puede realizar los cambios que crea oportunos en el contenido, estructura o control de la adaptación del curso. Cambiando preguntas, actividades, páginas, reestructurando las relaciones de dependencia o requisitos entre conceptos, cambiando de grado o de tema un concepto, etc.

Referencias Bibliográficas

[Adamidis & Petridis 1996] P. Adamidis, V. Petridis. Co-operating populations with different evolution behaviours. Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, ICEC'96, pp. 188-191. 1996.

[Adeli & Cheng 1994] H. Adeli, N.T. Cheng. Augmented lagragian genetic algorithms for structural optimization. Journal of Aerospace Engeneering, vol. 7, pp. 104-108. 1994.

[Agrawal et al. 1993] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. ACM SIGMOD International Conference on Management of Data. Washington. 1993.

[Agrawal & Srikant 1994] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules. Proceedings. of 20 th. VLDB Conf. Santiago de Chile. 1994.

[Ali et al. 1997] K. Ali, S. Manganaris, R. Srikant. Partial classification using association rules. Proceedings of KDD-97, pp. 115-118. 1997.

[Anderson & Reiser 1985] J. R. Anderson, B. Reiser. The LISP tutor. Byte 10 (4), pp. 159-175. 1985.

[Angeline 1997] P. J. Angeline. Subtree crossover: building block engine or macromutation?. Proceedings of the 2nd Annual Genetic Programming Conference, pp. 9-17. 1997.

[Antonisse 1989] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. Proceedings of the Third International Conference on Genetic Algorithms, pp. 86-91. 1989.

[Asnicar & Tasso 1997] F. A. Asnicar, C. Tasso. ifWeb: A prototype of user model-based intelligent agent for document filtering and navigation in the World Wide Web. 6th International Conference on User Modeling, UM97. 1997.

[Back et al. 1993] T. Back, G. Rudolph, H. P. Schwefel. Evolutionary programming and evolution strategies: similarities and differences. Proceedings of the 2nd Annual Conference on Evolutionary Programming, pp. 11-22. 1993.

[Back & Schwefel 1995] T. Back, H. P. Schwefel. Evolution strategies I: variants and their computational implementation. Genetic Algorithms in Engineering and Computer Science, pp. 111-126. 1995.

[Back & Schwefel 1996] T. Back, H. P. Schwefel. Evolutionary computation: an overview. Proceedings of 1996 IEEE International Conference on Evolutionary Computation. ICEC '96, pp. 20-29. 1996.

[Back 2000] T. Back. Introduction to Evolutionary Algorithms. Evolutionary Computation I: Basic Algorithms and Operators. Institute of Physics Publishing, pp. 59-63. 2000.

[Baker 1987] J. E. Baker. Reducing bias and inefficiency in the selection algortihm. Proceedings of the 2nd International Conference on Genetic Algorithms, pp. 14-21. 1987.

[Banzhaf et al. 1998] W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone. Genetic Programming - an Introduction / On the automatic evolution of computer programs and its applications. Morgan Kaufmann. 1998.

[Barr et al. 1976] A. Barr, M. Beard, R. C. Atkinson. The computer as tutorial laboratory: the Stanford BIP project. International Journal on the Man-Machine Studies 8 (5), pp 567-596. 1976.

[Bayardo & Agrawal 1999] R. J. Bayardo, R. Agrawal. Mining the most interesting rules. Fifth conference ACM on Knowledge Discovery and Data Mining. SIGKDD. 1999.

[Beaumont 1994] I. Beaumont. User modeling in the interactive anatomy tutoring system ANATOM-TUTOR. User Modeling and User-Adapted Interaction 4,1, pp. 21-46. 1994.

[Beyer 1993] H. Beyer. Toward a theory of evolution strategies: some asymptotical results for the (1, +λ) theory. Evolutionary Computation, vol. 1, n. 2, pp. 165-188. 1993.

[Beyer 1994] H. Beyer. Toward a theory of evolution strategies: the (μ, λ) theory. Evolutionary Computation, vol. 2, n. 4, pp. 381-407. 1994.

[Beyer 1995a] H. Beyer. Toward a theory of evolution strategies: on the benefit of sex - the (μ/μ, λ) theory. Evolutionary Computation vol. 3, n. 1, pp. 81-111. 1995.

[Beyer 1995b] H. Beyer. Toward a theory of evolution strategies: self-adaptation. Evolutionary Computation vol. 3, n. 3, pp. 311-347. 1995.

[Bhattacharyya et al. 1998] S. Bhattacharyya, O. Pictet, G. Zumbach. Representational semantics for genetic programming based learning in high-frequency financial data. Genetic Programming 1998: Proc. 3rd Annual Conf., 11-16. Morgan Kaufmann, 1998.

[Blackboard 1999] CourseInfo. Blackboard Inc.

[Blickle 2000] T. Blickle. Tournament selection. Evolutionary Computation: Basic Algorithms and Operators. Institute of Physics Publishing, pp. 181-186. 2000.

[Bloedorn & Michalski 1998] E. Bloedorn, R.S. Michalski. Data Driven constructive induction: methodology and applications. Feature extraction. Constructive and Selection: a data mining perspective. Kluwer, pp. 51-68. 1998.

[Bojarczuk et al. 2000] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas. Genetic programming for knowledge discovery in chest pain diagnosis. IEEE Engineering in Medicine and Biology Magazine 19(4), pp. 38-44. 2000.

[Bonabeau et al. 1999] E. Bonabeau, M. Dorigo, T. Theraulaz. From Natural to Artificial Swarm Intelligence. Oxford University Press. 1999.

[Box 1957] G. E. P. Box. Evolutionary operation: a method of increasing industrial productivity. Applied Statistics, vol. 6, pp. 81-101. 1957.

[Brecht et al. 1989] B. J. Brecht, G. I. McCalla, J. E. Greer. Planning the content of instruction. Proceedings of 4-th International Conference on AI and Education. Amsterdam, IOS, pp. 32-41. 1989.

[Breiman et al. 1984] L. Breiman, J. Friedman, R. Olshen, C. Stone. Classification and Regression Trees. Chapman & Hall. New York. 1984.

[Brin et al. 1997] S. Brin, R. Motwani, J.D. Ullman, S. Tsur. Dynamic itemset counting and implication rules for market basket data. SIGMOD Record, 26(2), pp. 255-264. 1997.

[Brusilovsky 1992] P. Brusilovsky. Intelligent Tutor, Environment and Manual for Introductory Programming. Educational and Training Technology International. 29 (1), pp. 26-34. 1992.

[Brusilovsky et al. 1993] P. Brusilovsky, L. Pesin, and M. Zyryanov. Towards an adaptive hypermedia component for an intelligent learning environment. Human-Computer Interaction. Lecture Notes in Computer Science, Vol. 753. Springer-Verlag, Berlin . pp 348-358. 1993.

[Brusilovsky & Pesin 1994] P. Brusilovsky, L. Pesin. ISIS-Tutor: An adaptive hypertext learning environment. JCKBSE'94, Japanese-CIS Symposium on knowledge-based software engineering, Pereslavl- Zalesski, Russia, pp. 83-87. 1994.

[Brusilovsky 1996] P. Brusilovsky. Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction. Spec Iss. On Adaptive Hypertext and Hypermedia. 6 (2-3), pp. 87-129. 1996.

[Brusilovsky et al. 1996] P. Brusilovsky, E. Schwarz, and G. Weber. ELM-ART: An intelligent tutoring system on World Wide Web. Third International Conference on Intelligent Tutoring Systems. ITS-96. pp.261-269.1996.

[Brusilovsky & Shwarz 1997] P. Brusilovsky, E. Schwarz. User as student: Towards an adaptive interface for advanced Web-based applications. Proceedings of 6th International Conference on User Modeling. Sardinia, Italy, June 2-5, pp.177-188. 1997.

[Brusilovsky et al. 1997] P. Brusilovsky, J. Eklund, E. Schwarz. Adaptive Navigation Support in Educational Hypermedia on the World Wide Web. Proceedings of INTERACT97. Sydney, Australia, pp. 278-285. 1997.

[Brusilovsky et al. 1998] P. Brusilovsky, J. Eklund, ans E. Schwarz. Web-based education for all: A tool for developing adaptive courseware. Computer Networks and ISDN Systems , 30 (1-7) pp. 291-300.1998.

[Brusilovsky 1999] P. Brusilovsky. Adaptative Educational Systems on the World-Wide-Web: A Review of Available Technologies. 1999.

[Brusilovsky & Miller 1999] P. Brusilovsky, P.Miller. Web-based Testing for Distance Education. World Conference of the WWW and Internet, WebNet'99. Honolulu. pp. 149-154. 1999.

[Brusilovsky 2001] P. Brusilovsky. Adaptive Educational Hipermedia. Proceeding of Tenth Internacional PEG Conference. Finland. 2001.

[Burns & Capps 1988] H.L. Burns, C.G. Capps. Foundations of intelligent tutoring systems: An introduction. Foundations of intelligent tutoring systems. Hillsdale: Lawrence Erlbaum Associates, pp. 1-19. 1988.

[Cabral 2000] J.L. Cabral. A Data Mining Model to Capture User Web Navigation Patterns. PHD. London. Julio. 2000.

[Carro et al. 1999] R. M. Carro, E. Pulido, and P. Rodrígues .: TANGOW: Task-based Adaptive learNer Guidance on the WWW. Computer Science Report , Eindhoven University of Technology, pp. 49-57. 1999.

[Catlett 1991] J. Catlett. Overpruning large decision trees. Proc. Of the 12th International Joint Conference AI. IJCAI'91. Sidney. 1991.

[Cendrowska 1987] J. Cendrowska. PRISM: an algorithm for inducing modular rules. International Journal of Man-Machine Studies, 27: pp. 349-370. 1987.

[Cerf 1995] R. Cerf. An asymptotic theory of genetic algorithms. Artificial Evolution, Lecture Notes in Computer Science, 1063, pp. 37-52. Springer. 1995.

[Cieniawski 1993] S. E. Cieniawski. An investigation of the ability of genetic algorithms to generate the tradeoff curve of a multi-objective groundwater monitoring problem. Master's thesis. University of Illinois at Urbana-Champaign. 1993.

[Corbertt & Anderson 1992] A. Corbett, J. Anderson. Knowledge tracing in the ACT programming tutor. In: Proceedings of 14-th Annual Conference of the Cognitive Science Society. 1992.

[Davis 1989] L. Davis. Adapting operator probabilities in genetic algorithms. Proceedings of the Third International Conference on Genetic Algorithms, pp. 61-69. 1989.

[Davis & Principe 1991] T. E. Davis, J. C. Principe. A simulated annealing like convergence theory for simple genetic algorithm. Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 174-181. 1991.

[Davis & Principe 1993] T. E. Davis, J. C. Principe. A Markov chain framework for the simple genetic algorithm. Evolutionary Computation, vol. 1, n. 3, pp. 269-292. 1993.

[Dawkins 1976] R. Dawkins. The Selfish Gene. Oxford University Press. 1976.

[De Bra 1996] P. De Bra. Teaching Hypertext and Hypermedia through the Web. Journal of universal computer science **,**2(12), pp. 797-804. 1996.

[De Bra & Calvi 1997] P. De Bra, L. Calvi. Using dynamic hypertext to create multi-purpose textbooks. Proceedings of ED-MEDIA/ED-TELECOM'97. Calgary, Canada. 1997.

[De Bra et al. 1999] P. De Bra, P. Brusilovsky, G. Houben. Adaptative Hypermedia: From System to Framework. ACM Computer Surveys. 31 (4). 1999.

[De Bra 2000] P. De Bra. Pros and Cons of Adaptive Hypermedia in Web-Based Education. Journal on CyberPsychology and Behavior, Vol. 3, nr.1, pp. 71-77. 2000.

[De Bra et al. 2000] P. De Bra, H. Wu, A. Aerst, G. Houben. Adaptation Control in Adaptive Hypermedia Systems. International Conference on Adaptive Hypermedia. 2000.

[De Bra & Rutier 2001] P. De Bra, J. Ruiter, AHA! Adaptive Hipermedia for All. Proceedings of the WebNet Conference, pp. 262-268. 2001.

[De La Passardiere & Dufresne 1992] B. De La Passardiere, A. Dufresne . Adaptive navigational tools for educational hypermedia. ICCAL´92, 4-th International Conference on Computers and Learning. Berlin. Springer-Verlag. pp 555-567. 1992.

[Deb & Goldberg 1989] K. Deb, D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. Proceedings of the Third International Conference on Genetic Algorithms, pp. 42-50. 1989.

[Deb 1995] K. Deb. Optimization for Engineering Design: Algorithms and Examples. Prentice – Hall. 1995.

[Deb 2001] K. Deb. Multi-Objetive Optimization Using Evolutionary Algorithms. Wiley. 2001.

[Delgado et al. 2001] M. Delgado, D. Sánchez, M.J. Martín-Bautista, M.A. Vila. Mining Association rules with improved semantics in medical databases. Artificial Intelligence in Medicine. 21. 2001

[D'haeseleer 1994] P. D'haeseleer. Context preserving crossover in genetic programming. Proceedings of the 1994 IEEE World Congress on Computational Intelligence, pp. 256-261. IEEE Press. 1994.

[Dougherty et al. 1995] J. Dougherty, R. Kohavi, M. Sahami. Supervised and unsupervised discretization of continuos features. Proc. of the 12th International Conference Machine Learning, pp. 194-202. 1995.

[Dubois 2002] P. Dubois. Edición Especial MySQL. Prentice Hall. 2002.

[Eiben et al. 1991] A. E. Eiben, E. H. L. Aarts, K. M. Van Hee. Global convergence of genetic algorithms: a Markov chain analysis. Proceedings of the 1st Parallel Problem Solving from Nature, pp. 4-12. 1991.

[Eiben et al. 1994] A. E. Eiben, P. E. Raue, Z. Ruttkay. Genetic algorithms with multi-parent recombination. Proceedings of the Third International Conference on Parallel Problem Solving from Nature, pp. 78-87. 1994.

[Eldredge 1989] N. Eldredge. Macro-evolutionary Dynamics: species, niches, and adaptive peaks. McGraw – Hill. 1989.

[Eliot et al. 1997] C.Eliot, D. Neiman, and M. Lamar.Medtec: a Web-based intelligent tutor for basic anatomy. Proceedings of WebNet´97, World Conference of the WWW, Internet and Intranet, Toronto, Canada, November 1-5, AACE, pp. 161-165.1997.

[Fink et al. 1997] J. Fink, A. Kobsa, J. Schreck. Personalized hypermedia information provision through adaptive and adaptable system features. 6th International Conference on User Modeling, UM97. Sardinia, Italia. 1997.

[Fogel et al. 1965] L. Fogel, A. Owens, M. Walsh. Artificial intelligence through a simulation of evolution. Biophysics and Cybernetic Systems. Proceedings of the 2nd Cybernetic Sciences Symposium , pp. 131-155. 1965.

[Fogel et al. 1966] L. Fogel, A. Owens, M. Walsh. Artificial Intelligence through Simulated Evolution. John Wiley & Sons. 1966.

[Fonseca & Fleming 1993] C.M. Fonseca, P.J. Fleming. Genetic algorithms for multiobjetive optimization: formulation, discussion and generalization. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 416-423. 1993.

[Fonseca & Fleming 1994] C.M. Fonseca, P.J. Fleming. An overview of evolutionary algorithms in multiobjetive optimization. Tech. Rep. Department of Automatic Control and Systems Engineering. Universisty of Sheffield. U.K. 1994.

[Fourman 1985] M. P. Fourman. Compaction of symbolic layout using genetic algorithms. Proceedings of the International Conference of Genetic Algorithms and their Applications, pp. 141-153. 1985.

[Fraser 1957] A. S. Fraser. Simulation of genetic systems by automatic digital computers. Australian Journal of Biological Science, 10, pp. 484-491. 1957.

[Freitas 1997] A. Freitas. A Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction. Genetic Programming 97: Proceedings of the Second Annual Conference. 1997.

[Freitas 1998] A. Freitas. On objective measures of rule surprisingness. Proc. 2nd European Symp., PKDD-98. Lecture Notes in Artificial Intelligence 1510, pp. 1-9. 1998.

[Freitas 1999] A. Freitas. On Rule Interestingness Measures. Knowledge-Based Systems Journal. 1999.

[Freitas 2000] A. Freitas. Understanding the crucial differences between classification and discovery of association rules. ACM SIGKDD Explorations. 2(1) pp. 65-69. 2000.

[Freitas 2002] A. Freitas. Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag. 2002.

[Friedberg 1958] R. Friedberg. A learning machine, part I. IBM Journal of Research and Development, 2, pp. 2-13. 1958.

[Friedberg et al. 1959] R. Friedberg, B. Dunham, J. North. A learning machine, part II. IBM Journal of Research and Development, 3, pp. 282-287.

[Friedman et al. 1996] J. H. Friedman, R. Kohavi, Y. Yun. Lazy decisión trees. Proced. Of the 14th National Conference of the American Association for Artificial Intelligence, AAAI'96. 1996.

[Fukuda et al. 1996] T. Fukuda, Y. Morimoto, S. Morishita, T. Tokuyama. Data Mining usign two-dimensional optimized association rules: scheme, algotithms, and Vislualization. Proc. Of ACM-SIGMOD Int. Conf on Management of Data. 1996.

[Gilbert & Han 1999] J.E. Gilbert, C.Y. Han. Adapting instruction in search of a significant difference. Journal of Network and Computer Applications, 22. 1999.

[Gilbert et al. 1998] R.G. Gilbert, R. Goodacre, B. Shann, D.B. Kell, J. Taylor, J.J. Rowland. Genetic Programming-based variable selection for high-dimensional data. Proc. 3rd Conf. Genetic Programming, pp. 109-115. 1998.

[Giordana & Neri 1996] A. Giordana, F. Neri. Search-intensive concept induction. Evolutionary computation. 3(4): pp. 375-416. MIT. 1996.

[Giordana et al. 1994] A. Giordana, L. Saitta, F. Zini. Learning disjuntive concepts by jeans of genetic algorithms. Proc. 10th Internatinal Conf. Machine Learning (ML'94). pp. 96-104. 1994.

[Goethals et al. 2000] B. Goethals, J. V. Bussche. On Supporting Interactive Association Rule Mining. 2000.

[Goldberg & Richardson 1987] D. E. Goldberg, J. Richardson. Genetic algorithms with sharing for multimodal function optimization. Proceedings of the International Conference on Genetic Algorithms and their Applications (ICGA '87), pp. 41-49. 1987.

[Goldberg 1989] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison – Wesley. 1989.

[Goldberg et al. 1991] D. E. Goldberg, K. Deb, B. Korb. Don't worry, be messy. Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 24-30. 1991.

[Gonschorek & Herzog 1995] M. Gonschorek, and C. Herzog. Using hypertex for an adaptive helpsystem in an intelligent tutoring system. 7 th World Conference on Artificial Intelligence in Education, Washington, DC, AACE, pp 274-281. 1995.

[Gray & Orlowska 1998] B. Gray, M.E. Orlowska. Clustering categorical attributes into interesting association rules. Proc. Of Second Pacific-asia Conf. on Knowledge Discovery and Data Mining. Melburne, Australia. 1998.

[Grefenstette 1994] J.J. Grefenstette. Genetic Algorithms for Machine Learning. Kluwer Academic. 1994.

[Hajela & Lin 1992] P. Hajela, C.Y. Lin. Genetic search strategies in multicriterion optimal design. Structural Optimization, vol. 4, pp. 99-107, 1992.

[Halasz & Schwartz, 94] F. Halasz, M. Schwartz. The Dexter Hypertext Referente Model. Communications of the ACM. 37:2, pp. 30-39. 1994.

[Han et al. 1999] J. Han, L. Lakshamanan, R.T. Ng. Constraint-Based, Multidimensional Data Mining. IEEE Computer. 1999.

[Haykin 1999] S. Haykin. Neural Networks: a comprenhensive foundation. 2ª ed. Prentice Hall. 1999.

[Heift & Nicholson 2000] T. Heift, D. Nicholson. Enhanced Server Logs for Intelligent, Adaptive Web-based Systems. 2000.

[Henze & Nejdl 2000] N. Henze, W. Nejdl. Extendible adaptive hypemedia courseware: Integrating different courses and Web material. Adaptive Hypermedia and Adaptive Web-based Systems, AH2000. Springer-Verlag. pp 109-120.2000.

[Henze et al. 1999] N. Henze, K. Naceur, W, Nejdl, and M. Wolpers. Adaptive hyperbooks for constructivist teaching. Künstliche Intelligenz ,(4), pp.26-31.1999.

[Hérin et al. 2002] D. Herín, M. Sala, P. Pompidor. Evaluating and Revising Courses from Web Resources Educational. ITS 2002, LNCS 2363, pp. 208-218. 2002.

[Hilliard et al. 1989] M.R. Hilliard, G.E. Liepins, M. Palmer, G. Rangarajen. The computer as a partner in algorithmc design : Automated discovery of parameters for a multobjetivive scheduling heuristic. In Impacts of Recent Computer Advances on Operations Research. New York. 1989.

[Hipp et al. 2000] J. Hipp, U. Güntzer, G. Nakhaeizaeh. Algorithms for Association Rule Mining – A General Survey and Comparison. SIGKDD Explorations. 2.000.

[Hockemeyer et al. 1998] C. Hockemeyer, T. Held, and D Albert.: RATH-A relational adaptive tutoring hypertext WWW-environment based on knowledge space theory. Proc. of CALISCE´98, 4th International conference on Computer Aided Learning and Instruction in science and Engineering, Göteborg, sweden, pp.417-423.1998.

[Hohl et al. 1996] H. Hohl, D. Böcker, R. Gunzenhäuser. Hypadapter: An adaptive hypertext system for exploratory learning and programming. User Models and User Adapted Interaction 6 (this issue). 1996.

[Holland 1975] J. H. Holland. Adaptation in Natural and Artificial Systems. The University of Michigan Press. 1975.

[Höök 1997] K. Höök. Evaluating the utility and usability o an adaptive hypermedia system. Proceedings of 1997 International Conference on Intelligent User Interfaces. Orlando, Florida. 1997.

[Horn & Nafpliotis 1993] J. Horn, N. Nafpliotis. Multiobjective Optimization usign the Niched Pareto Genetic Algorithm. Tech. Rep. IlliGAI Report 93005. Universtity of Illinois. 1993.

[Hsu 1999] W.H. Hsu, W.M. Pottenger, M. Welge, J. Wu, T. Yand. Genetic algorithms for selection and partitioning of attributes in large-scale data mining problems. Proc. AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions. 1999.

[Hu 1998a] Y. Hu. A genetic programming approach to constructive induction. Genetic Programming 98: Proc. 3rd Annual Conf., 146-151. 1998.

[Hu 1998b] Y.J. Hu. Constructive induction: covering attribute spectrum. Feature extraction. Constructive and Selection: a data mining perspective. 51-68. Kluwer. 1998.

[Hussain et al. 1999] F. Hussain, H. Liu, C.L. Tan, M. Dash. Discretization : an enabling technique. Technical Report TRC6/99. The National University of Singapore. Junio. 1999.

[Jakob et al. 1992] W. Jakob, M. Gorges-Schleuter, C. Blume. Application of genetic algorithms to task planning and learning. Parallel Problem Solving from Nature 2, pp. 291-300. 1992.

[Johansen 1911] W. Johansen. The genotype conception of heredity. The American Naturalist, 45, pp. 129-159. 1911.

[John et al. 1994] G.H. John, R. Kohavi, K. Pfleger. Irrelevant features and the subset selection problem. Proced. Of the 11th International Conf on Machine Learning. pp. 121-129. 1994.

[Johnson 1986] W.L. Johnson. Intention-based diagnosis of novice programming errors. Pitman: Morgan Kaufmann. 1986.

[Jones 1995] T. Jones. Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis. The University of New Mexico, Albuquerque, New Mexico. 1995.

[Jones et al. 1993] G. Jones, R. D. Brown, D. E. Clark, P. Willett, R. C. Glen. Searching databases of two-dimensional and three-dimensional chemical structures using genetic algorithms. Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 597-602. 1993.

[Julstrom 1995] B. A. Julstrom. What are you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 81-87. 1995.

[Kamber et Shinghal 1996] M. Kamber, R. Shinghal. Evaluating the interestingness of characteristic rules. Proc. Of Second Int. Conf. on Knowledge Discovery and Data Mining, KDD96. Portland, Oregon. 1996.

[Kay & Kummerfeld 1994] J. Kay and R.J. Kummerfeld. An individualised course for the C programming language. In: Proceedings of Second International WWW conference , Chicago, IL, pp. 17-20. 1994.

[Kay & Kummerfeld 1997] J. Kay, and B. Kummerfeld. User models for customized hypertext. Intelligent hypertext: Advanced techniques for the World wide Web. Lecture Notes in Computer Science, Vol. 1326. springer-verlag, Berlin.1997.

[Kayama & Okamoto 1998] M. Kayama, and T. Okamoto.: A mechanism for knowledge-navigation in hyperspace with neural networks to support exploring activities. Proc. Of 4th World Congress on Expert systems. Mexico pp 41-48.1998.

[Kéller et al. 1996] R. M. Séller, S.R. Wolfe, J.R. Chen, J.L. Rabinowitz, N. Mathe. A bookmarking service for organizing and sharing URLs. Proceeding of Sixth International World Wide Web Conference. Santa Clara, CA. 1997.

[Kelly & Davies 1991] J.D. Kelly, L.A. Davies. A hybrid genetic algorithm for classification. Proc. 12th Int. Joint Conf. on Artificial Intelligence, IJCAI-91, Sidney, Australia, pp. 645-650. 1991.

[Kennedy et al. 2001] J. Kennedy, R. C. Eberhart, Y. Shi. Swarm Intelligence. Morgan Kaufmann. 2001.

[Kimbal 1996] R. Kimbal. The data warehouse toolkit. New York: Joh Wiley. 1996.

[Kirkpatrick et al. 1983] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. Optimization by simmulated annealing. Science, vol. 220, n. 4598, pp. 671-680. 1983.

[Klemettinen et al. 1994] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, A. Inkeri. Finding Interesting Rules from Large Sets of Discovered Association Rules. Int. Conf. on Information and Knowledge Management. pp 401-407. 1994.

[Klösgen 1996] W. Klösgen. Explora: a multipattern and multistrategy discovery assistant. Advances in Knowledge Discovery and Data Mining. AAAI/Press/MIT Press, pp. 249-271. 1996.

[Klösgen & Zytkow 2002] W. Klösgen, J.M. Zytkow. Handbook of Data Mining and Knowledge Discovery. Oxford University Press. 2002.

[Kotásek 2000] P. Kotásek. DMSL: The Data Mining Specification Language. Doctoral Thesis. Brno University. 2000.

[Koza 1989] J. R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. Procedings of the Eleventh International Joint Conference on Artificial Inteligence IJCAI-89, volume 1, pp. 768-774. 1989.

[Koza 1992] J. R. Koza. Genetic Programming: on the programming of computers by means of natural selection. MIT Press. 1992.

[Kursawe 1991] F. Kursawe. A variant of evolution strategies for vector optimization. Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, pp. 193-197. 1991.

[Lai et al. 1995] M.C. Lai, B.H. Chen, S.M. Yuan. Toward a new educational environment. Proceedings of 4th International World Wide Web Conference. Boston, USA. 1995.

[Laroussi & Benahmed 1998] M. Laroussi, M. Benahmed. Providing an adaptive learning through the Web case of CAMELEON. Proceedings of CALISCE´98 4 TH International conference on Computer Aided and Instruction in Science and Engineering, June 15-17. Göteborg, Sweden. Pp. 411-416.1998.

[Lavrac et al. 1999] N. Lavrac, P. Flach, B. Zupan. Rule Evaluation Measures: A Unifying View. ILP-99, LNAI 1634. Springer-Verlag Berlin Heidelberg. 1999.

[Lin et al. 1994] S. C. Lin, W. Punch, E. Goodman. Coarse-grain parallel genetic algorithms: categorization and new approach. Sixth IEEE Symposium on Parallel and Distributed Processing, pp. 28-37. 1994.

[Liu et al. 1997] B. Liu, W. Hsu, S. Chen. Using general impressions to analyze discovered classification rules. Proc. 3rd Int. Conf. Knowledge Discovery & Data Mining, KDD-97, pp. 31-36. 1997.

[Liu et al. 1998] B. Liu, W. Hsu, Y. Ma. Integrating clasification and association rule mining. KDD-98. New York. pp 27-31. 1998.

[Mahfoud 1995] S. W. Mahfoud. Niching Methods for Genetic Algorithms. PhD thesis, University of Illinois at Urbana-Champaign. 1995.

[Mahfoud 2000] S. W. Mahfoud. Niching methods. Evolutionary Computation II: Advanced Algorithms and Operators. Institute of Physics Publishing, pp. 87-92. 2000.

[Manderick et al. 1991] B. Manderick, M. de Weger, P. Spiessens. The genetic algorithm and the structure of the fitness landscape. Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 143-149. 1991.

[Martin-Bautista & Vila 1999] M.J. Martin-Bautista, M.A. Vila. A survey of genetic feature selection in mining issues. Proc. Congress on Evolutionary Computation, CEC-99. Washington D.C., USA. 1999.

[Matelic & Marcus 2000] J.I. Matelic, A. Marcus. Data Cleansing : Beyond Integrity Analisys. Proc. Conference on Information Quality, IQ2000, pp. 200-209. 2000.

[Matheus 1993] C. Matheus, P. Chan. G. Piatetsky-Shaprio. Systems for Knowledge Discovery in Databases. IEEE Transactions on Knowledge and Data Engineering. V. 5 no. 6. 1993.

[Mckendree et al. 1992] J. McKendree, B. Radlinski, M.E. Atwood. The Grace Tutor: a qualified success. Proceedings of Second International Conference, ITS'92, pp. 677-684. 1992.

[Michalewicz 1996] Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. 3ª ed. Springer. 1996.

[Michalski et al. 1986] R.S. Michalski, I. Mozetic, J. Hong, N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. Proced. Of fitth national conf. on Artificial Inteligence. 1986.

[Michalski et al. 1998] R. S. Michalski, I. Bratko, M. Kubat. Machine Learning and Data Mining Methods and Applications. John Wiley & Sons Ltd. 1998.

[Montana 1995] D.J. Montana. Strongly Typed Genetic Programming. Evolutionary Computation. 3(2), pp. 199-230. 1995.

[Murray et al. 1998] T. Murray, C. Condit, and E. Haugsjaa.: MetaLinks: A preliminary framework for concept-based adaptive hypermedia. Proc. of Workshop "WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems, san Antonio, TX. 1998.

[Nakabayashi et al. 1997] K. Nakabayashi, M. Maruyama, Y. Kato, H. Touhei, Y. Fukuhara. Architecture of an intelligent tutoring system on the WWW. Proceedings of AI-ED'97, World Conference on Artificial Intelligence in Education, Kobe, Japan, pp. 39-46. 1997.

[Negro et al. 1998] A. Negro, V. Scarano, and R Simari: User adaptivity on WWW Through CHEOPS. Computing science Reports, Eindhoven University of Technology. pp. 57-62.1998.

[Neumann & Zirvas 1998] G. Neumann, J. Zirvas. SKILL: A Scalable Internet-Based Teaching and Learning System. Webnet 98 World Conference of the WWW, Internet & Intranet. Pp. 688-693. Orlando, Florida. 1998.

[Newell & Simon 1972] A. Newell, H. Simon. Human problem solving. Prentice Hall. 1972.

[Ng & Yau 2002] W. Ng and J Yau. Adapting Web Interfaces by WHAT. Adaptive Hypermedia and Adaptive Web-based Systems. Malaga. 2002.

[Niimi & Tazaki 2000] A. Niimi, E. Tazaki. Rule Discovery Technique Using Genetic Programming Combined with Apriori Algorithm. Discovery Science. 273-277. 2000.

[Niimi & Tazaki 2001] A. Niimi, E. Tazaki. Extended Genetic Programing Using A priori Algorithm for Rule Discovery. JSAI 2001 Workshops, LNAI 2253, pp. 525-532. 2001.

[Nix & Vose 1992] A. E. Nix, M. D. Vose. Modeling genetic algorithms with Markov chains. Annals of Mathematics and Artificial Intelligence, vol. 5, n. 1, pp. 79-88. 1992.

[Noda et al. 1999] E. Noda, A. Freitas, H.S. Lopes. Discovering interesting prediction rules with a genetic algorithm. To appear in Proc. Congress on Evolutionary Computation, CEC-99. Washington D.C., USA, July 1999. IEEE, Piscataway, NJ. 1999.

[Okazaki et al. 1997] Y. Okazaki, K. Watanabe, H. Kondo. An Implementation of an intelligent tutoring system (ITS) on the World-Wide Web (WWW). Educational Technology Research 19 (1), pp. 35-44. 1996.

[Ortigosa & Carro 2002] A. Ortigosa, R.M. Carro. Asistiendo el Proceso de Mejora Continua de Cursos Adaptativos. III Congreso Internacional de Interacción Persona-Ordenador. Pp. 246-250. 2002.

[Osyczka 1985] A. Osyczka. Multicriteria optimization for engineering design. In Design Optimization, pp. 193-227. 1985.

[Pal et al. 2002] S.K. Pal, V. Talwar, P. Mitra. Web Mining in Soft Computing Framework : Relevance, State of the Art and Future Directions. IEEE Transactions on Neural Networks. 2002.

[Papanikolaou et all. 2000] K.A. Papanikolau, G.D. Magoulas, M. Grigoriadou. A connctionist approach for supporting personalized learning. Asaptive hypermedia and Adaptive Web-based Systems, AH2000, pp. 189-201. 2000.

[Pareto 1896] V. Pareto. Tours D'Economie Politique, volume I and II. F. Rouge. Lausanne. 1896.

[Pavillon 1996] G. Pavillon. ARC II : a system for inducing and simplifying dependence and causal dependence relationships. Thirteenth European Meeting on Cybernetics and Systems Research. 1996.

[Pérez et al. 2001] T.A. Pérez, J. Gutiérrez, R. López, A. González, J.A. Vadillo. Hipermedia, adaptación, constructivismo e instructivismo. Revista Iberoamericana de Inteligencia Artificial. No 12, pp 29-38. 2001.

[Piatetsky-Shapiro 1991] G. Piatetsky-Shapiro. Discovery, análisis and presentation of strong rules. Knowledge Discovery in Databases, pp. 229-248. 1991.

[Piatesky-Shapiro & Matheus 1994] G. Piatesky-Shapiro, J. Matheus. The interestingness of deviations. Proceedings of the AAAI-94 Workshop on KDD, pp. 25-26. 1994.

[Pilar da Silva et al. 1998] D. Pilar da Silva, R. V. Durm, E. Duval, and H. Olivié. Concepts and documents for adaptive educational hypermedia: a model and a prototype. Computing Science Reports, Eindhoven University of technology, Eindhoven. Pp.35-43. 1998.

[Poli & Langdon 1998] R. Poli, W. B. Langdon. On the search properties of different crossover operators in genetic programming. Genetic Programming 1998: Proceedings of the 3rd Annual Conference, GP ’98, pp. 293-301. 1998.

[Poli et al. 1999] R. Poli, J. Page, W. B. Langdon. Smooth uniform crossover, sub-machine code GP and demes: a recipe for solving high-order boolean parity problems. Proceedings of the Genetic and Evolutionary Computation Conference GECCO ’99, pp. 1162-1169. 1999.

[Potts et al. 1994] J. C. Potts, T. D. Giddens, S. B. Yadav. The development and evaluation of an improved genetic algorithm based on migration and artificial selection. IEEE Transactions on Systems, Man, and Cybernetics, vol. 24, n. 1, pp. 73-86. 1994.

[Pressman 2001] R.S. Pressman. Ingeniería del Software: Un enfoque prático. McGraw-Hill. 2001.

[Quinlan 1987] J.R. Quinlan. Generating Production rules from decision trees. Proc. IJCAI-87, pp.. 304-307. 1987.

[Radcliffe 1991] N. J. Radcliffe. Forma analysis and random respectful recombination. Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 222-229. 1991.

[Raymer et al. 1996] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn. Genetic programming for improved data mining – application to the biochemistry of protein interactions. Genetic Programming 1996. pp. 375-380. 1996.

[Rechenberg 1973] I. Rechenberg. Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog. 1973

[Rechenberg 1994] I. Rechenberg. Evolutionsstrategic '93. Frommann Verlag. Stuttfart, Germany. 1994.

[Reklaitis et al.1983] G. V. Reklaitis, A. Ravindran, K. M. Ragsdell. Engineering Optimization Methods and Applications. John Wiley & Sons. 1983.

[Reynolds 1994] R. G. Reynolds. An introduction to cultural algorithms. Proceedings of the Third Annual Conference on Evolutionary Programming, pp. 131-139. 1994.

[Richardson et al. 1989] J. T. Richardson, M. R. Palmer, G. E. Liepins, M. R. Hilliard. Some guidelines for genetic algorithms with penalty functions. Proceedings of the Third International Conference on Genetic Algorithms, pp. 191-197. 1989.

[Rios et al. 1998] A. Rios, J.L. Pérez de la Cruz, R. Conejo. SIETTE: Intelligent evaluation system using tests for TeleEducation. Workshop "WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems. 1998.

[Ritzel et al. 1994] B.J. Ritzel, J.W. Eheart, S. Ranjithan. Using genetic algotithms to solve a multiple objective groundwater pollution containment problem. Water Resources Research, vol 30. pp. 1589-1603. 1994.

[Roberto et al. 1997] J. Roberto, Rakesh, Dimitrios. Constraint-Based Rule Mining in Large, Dense Databases. 15 th Internacional Conference on Artificial Intelligence. Nagoya, Japan. 1997.

[Romero et al. 2002a] C. Romero, S. Ventura, C. De Castro, W. Hall, M. Hong. Using Genetic Algorithms for Data Mining in Web-based Educational Hypermedia Systems. Adaptive Hypermedia 2002. AH'2002 Workshop on Adaptive Systems for Web-based Education. Pp. 137-142. 2002.

[Romero et al. 2002b] C. Romero, P. de Bra, S. Ventura, C. de Castro. Using Knowledge Levels with AHA! For Discovering Interesting Relationship. E-Learn 02. World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education. Pp. 2721-2722. 2002.

[Romero et al. 2002c] C. Romero, S. Ventura, C. de Castro. Construcción de cursos hipermedia adaptativos basados en Web utilizando AHA. RIED. Revista IberoAmericana de Educación a Distancia. Número 2 Volumen 5, pp. 99-116. 2002.

[Romero et al. 2002d] C. Romero, C. de Castro, E. Espinosa. Curso básico de Java 2. Publicaciones de la Universidad de Córdoba y la Obra Social y Cultural de Cajasur. 2002.

[Romero et al. 2003a] C. Romero, S. Ventura, C. de Castro, P. De Bra. Discovering Prediction Rules in AHA! Courses. LNCS User Modeling'03. 2003.

[Romero et al. 2003b] C. Romero, C. de Castro, S. Ventura. Herramienta Autor para la Construcción de Cursos Hipermedia Adaptativos utilizando AHA. Congreso Interacción'03. Vigo. 2003.

[Romero et al. 2003c] C. Romero, C. de Castro, S. Ventura. Algoritmos Evolutivos para el Descubrimiento de Reglas de Predicción en la Mejora de Sistemas Educativos Adaptativos basados en Web. Revista ADIE. Asociación para el Desarrollo de la Informática Educativa. (número por salir) 2003.

[Rosenberg 1967] R. S. Rosenberg. Simulation of genetic populations with biochemical properties. PhD thesis. University of Michigan. Ann Harbor. Michigan. 1967.

[Rudolph 1994a] G. Rudolph. Convergence analysis of canonical genetic algorithm. IEEE Transactions on Neural Networks, vol. 5, n. 1, pp. 96-101. 1994.

[Rudolph 1994b] G. Rudolph. Convergence of non-elitist strategies. Proceedings of the 1st IEEE International Conference on Evolutionary Computation, pp. 63-66. 1994.

[Ryan & Rayward 1998] M.D. Ryan, V.J. Rayward-Smith. The evolution of decision trees. Genetic Programming 1998: Proc. 3rd Annual Conf., 350-358. Morgan Kaufmann, 1998.

[Ryu & Eick 1996] T.W. Ryu, C.F. Eick. MASSON: Discovering Commonalities of a Set of Objects Using Genetic Programming. Proc. of Genetic Programming Conference. 1996.

[Schaffer & Morishima 1987] J. D. Schaffer, A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. Proceedings of the 2nd

International Conference on Genetic Algorithms and their Applications, pp. 36-40. 1987.

[Schaffer 1985] J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. Proceedings of the First Internatinal Conference on Genetic Algorithms, pp. 93-100. 1985.

[Schraudolph & Belew 1992] N. N. Schraudolph, R. K. Belew. Dynamic paameter encoding for genetic algorithms. Machine Learning, vol. 9, n. 1, pp. 9-22. 1992.

[Schwefel 1981] H. P. Schwefel. Numerical Optimization of Computer Models. John Wiley & Sons. 1981.

[Schwefel 1995] H.P. Schwefel. Evolution and Optimum Secking. Sixth-Generation Computer Technology Series. John Wiley & Sons, New York. 1995.

[Shaefer 1987] C. G. Shaefer. The ARGOT strategy: adaptive representation genetic opyimizer technique. Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications, pp. 50-58. 1987.

[Schöch et all 1998] V. Schöch, M. Specht, and G. Weber : "ADI" – an empirical evaluation of a tutorial agent. World Conference on Educational Multimedia and Hypermedia and World Conference on Educational Telecommunications, Freiburg, Germany, AACE pp. 1242-1247. 1998.

[Shortliffe & Buchanan 1975] E. Shortliffe, B. Buchanan. A model of inexact reasoning in medicine. Mathematical Biosciences, 23:pp.351-379. 1975.

[Silberschatz & Tuzhilin 1996] A. Silberschatz, A. Tuzhilin. What makes pattterns interesting in Knoledge discovery systems. IEEE Trans. on Knowledge and Data Engineering. 8(6): pp.970-974. 1996.

[Silverstein et al. 1998] A. Silverstein, S. Brin, R. Motwani. Beyond market baskets : Generalizing association rules to dependence rules. Data Mining and Knowledge Discoverty. 2:pp.29-68, 1998.

[Smyth & Goodman 1991] P. Smyth, R.M. Goodman. Rule Induction using information theory. Knowledge Discovery in Databases. AAAI/MIT Press. Pp.159-176. 1991.

[Smythe & Goodman 1992] P. Smythe, R.M. Goodman. An Information Theoretic Approach to Rule Induction from Databases. IEEE Transactions on Knowledge and Data Engineering. 4(4): pp.301-316. 1992.

[Spears 1995] W. M. Spears. Adapting crossover in evolutionary algorithms. Proceedings of the Fourth Annual Conference on Evolutionary Programming, pp. 367-384. 1995.

[Specht et al. 1997] M. Specht,, G. Weber, S. Heitmeyer, V. Schöch. AST: Adaptive WWW-Courseware for Statistics. Proceedings of 6th International Conference on User Modeling, UM97, Chia Laguna, Sardinia, Italy, pp. 91-95. 1997.

[Specht & Oppermann 1998] M. Specht, R. Oppermann. ACE-Adaptive Courseware Environment. The new Review of Hypermedia and Multimedia, 4, pp. 141-161.1998.

[Spiliopoulou 2000] M. Spiliopoulou. Web Usage Mining for Web Site Evaluation. Communicacion of the ACM. 2.000.

[Srinivas & Deb 1993] N. Srinivas, K. Deb. Multiobjetive optimization using nondominated sorting in genetic algorithms. Tech. Rep. Department of Mechanical Engineering. Indian Institute of Technology. 1993.

[Srinivas & Patnaik 1994] M. Srinivas, L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, vol. 24, n. 4, pp. 656-667. 1994.

[Srikant et al. 1997] R. Srikant, Q. Vu, R. Agrawal. Mining Association Rules with Item Constrains. Proc 3rd. Int. Conf. Knowledge Discovery and Data Mining. Pp. 67-73. 1997.

[Srivastava et al. 2000] J. Srivastava, R. Cooley, M. Deshpnade, P. Tan. Web Usage Mining: Discovery and Aplications of Usage Pattern from Web Data. SIGKDD Explorations. ACM SIGKDD. Jan 2000.

[Steinacker et al. 1998] A. Steinacker, C. rechenberger, S. Fischer, R. Steinmetz. Dynamically generated tables of contents as guided tours in adaptive hypermedia systems. Educational Multimedia/Hypermedia and Telecomunications. 1998.

[Stern et al. 1997] M. Stern, B.P. Woolf, J. Kuroso. Intelligence on the Web? Proceedings of AI-ED'97, 8th World Conference on Artificial Intelligence in Education, Kobe, Japan, 18-22 August. Amsterdam: IOS, pp. 490-497. 1997.

[Stern & Woolf 2000] M.K. Stern, B.P. Woolf. Adaptive content in an online lecture system. Adaptive Hypermedia and Adaptive Web-based systems. Pp. 225-238. 2000.

[Sullivan 1997] T. Sullivan. Reading reader reaction: A proposal for inferential analisis of web server log files. Proceedings of Human factors and the Web 3 Conferences. 1997.

[Suzuki & Zytkow 2000] E. Suzuki, J. Zytkow. Unified Algorithm for Undirected Discovery of Exception Rules. Principles and Practice of Data Mining and Knowledge Discovery. pp 169-180. 2000.

[Syswerda & Palmucci 1991] G. Syswerda, J. Palmucci. The application of genetic algorithms to resource scheduling. Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 502-508. 1991.

[Tan & Kumar 2000] P. Tan, V. Kumar. Interesting Measures for Association Patterns: A Perspectiva. Technical Report TR00-036. Department of Computer Science. University of Minnnesota. 2000.

[Trueblood & Lovett 2001] R.P. Trueblood, J.N. Lovett. Data Mining and Statistical Analysis Using SQL. Springer-Verlag. 2001.

[Turney 1995] P. D. Turney. Cost-sensitive classification: empirical evaluation of a hybrid genetic decisión tree induction algortihm. Journal of Artificial Intelligent Research, 2, Mar. 1995.

[Vanneste 1994] P. Vanneste. The use of reverse engineering in novice program analysis (PhD Thesis). Katholieke Universiteit Leuven. 1994.

[Vassileva 1997] J. Vassileva. Dynamic Course Generation on the WWW. Proceedings of AI-ED'97, 8th World Conference on Artificial Intelligence in Education, Kobe, Japan, 18-22 August. Amsterdam: IOS, pp. 498-505. 1997.

[Velehuizen & Lamont 2000] D.A.V. Veldhuizen, G.B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. Evolutionary Computation 8(2): pp. 125-147. 2000.

[Ventura et al. 2001] S. Ventura, D. Ortiz, C. Herváz. Jclec: Una biblioteca de clases java para computación evolutiva. In I Congreso Español de Algoritmos Evolutivos y Bioinspirados. pp 23-30. 2001.

[Ventura et al. 2002] S. Ventura, A.J. Cruz, C. Herváz, F.Herrera. Análisis de operadores en Programación Genética basada en Gramáticas. In II Congreso Español de Algoritmos Evolutivos y Bioinspirados. pp 445-451. 2002.

[Vose 1991] M. D. Vose. Generalizing the notion of schema in genetic algorithms. Artificial Intelligence, vol. 50, n. 3, pp. 385-396. 1991.

[Vose & Liepins 1991] M. D. Vose, G. E. Liepins. Punctuated equilibria in genetic search. Complex Systems, n. 5. pp. 31-44. 1991.

[Wang 2002] F. Wang. On Using Data-Mining Technology for Browsing Log File Análisis in Asynchronous Learning Environment. World Conference on Educational Multimedia, Hypermedia and Telecomunications. 2002.

[WBT Systems 1999] TopClass. Dubñin, Ireland: WBT Systems. 1999.

[WebCT 1999] World Wide Web Course Tools. Vancouver, Canada: WebCT Educational Technologies. 1999.

[Weber 1999] G. Weber. ART-WEB .Trier: University of Trier. 1999.

[Weber & Möllenberg 1995] G. Weber, A. Möllenberg. ELM-Programming-Environment: A Tutoring System for LISP Beginners. Cognition and Computer Programming. Norwood, NJ: Ablex, pp. 373-408. 1995.

[Whigham 1995] P.A. Whigham. Gramatically-based Genetic Programing. Proceeding of the Workshop on Genetic Programming. pp. 33-41. 1995.

[Whitley 1989] D. Whitley. The GENITOR algorithm and selective pressure: why rank-based allocation of reproductive trials is best. Proc. Of 2nd Int. Conf. Genetic Algorithms. 116-121. 1989.

[Whitley 1992] D. Whitley. An executable model of a simple genetic algorithm. Foundations of Genetic Algorithms II, pp. 45-62. 1992.

[Whitley et al. 1991] D. Whitley, K. Mathias, P. Fitzhorn. Delta coding: an iterative search strategy for genetic algorithms. Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 77-84. Morgan 1991.

[Wienke et al. 1992] P. B. Wienke, C. Lucasius, and G. Kateman. Multicriteria target optimization of analytical procedures using a genetic algorithm. Analytica Chimica Acta, vol. 265, n. 2, pp. 211-225, 1992.

[Wilson & MacLeod 1993] P. Wilson, M. Macleod. Low implementation cost IIR digital filter design using genetic algorithms. IEE/IEEE workshop on Natural Algorithms in Signal Processing, pp. 1-8. 1993.

[Witten & Frank 2000] I.H. Witten, E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann. 1999.

[Wong & Leung 2000] M. L. Wong, K. S. Leung. Data Mining Using Grammar Based Genetic Programming And Applications. Kluwer Academic Pubrishers. 2000.

[Wu et al. 1999] H. Wu, G.J. Houben, P. De Bra. AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147-156, Darmstadt, Germany, 1999.

[Wu et al. 2001] H. Wu, E. De Kort, P. De Bra. Design Issues for General-Purpose Adaptive Hypermedia Systems. Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 141-150, Aarhus, Denmark, August 2001.

[Yan et al. 1996] T. Yan, M. Jacobsen, H. García-Molina, U. Dayal. From user access patterns to dynamic hypertext linking. In Fith Internacional World Wide Web Conference. Paris, France. 1996.

[Yao & Zhong 1999] Y.Y. Yao, N. Zhong. An Analysis of Quantitative Measures Associated with Rules. PAKDD. 479-488. 1999.

[Zaïane et al. 1997] O.R. Zaïane, M. Xin, J. Han, Discovering Web Access Pattern and Trends by Applying OLAP and Data Mining Technology on Web Logs. Proceeding from the ADL'98. Advances in Digital Libraries. Santa Barbara, 1998.

[Zaïne et al. 2000] O.R. Zaïne, J. Han. Implementation of a Web Usage Mining Framework for Web Activity Evaluation. TeleLearning Conference 2000.

[Zaïne 2001] O.R. Zaïene. Web Usage Mining for a Better Web-Based Learning Environment. Technical Report. 2001.

[Zaïane & Antonie 2001] O.R. Zaïane, M. Antonie. Automatic Text Categorization using Association Rule Mining. Journal of Intelligent Information Systems. Special Issue on Automated Text Categorization. 2001.

[Zaïne 2002] O.R. Zaïne. Building a Recommender Agent for e-Learning Systems. Proceeding of the Int. conf. Computers in Education. pp. 55-59. 2002.

[Zaïane & Luo 2001] O.R. Zaïane, J. Luo. Towards Evaluating Learners' Behaviour in a Web-Based Distance Learning Environment. Proc. IEEE International Conference on Advanced Learning Technologies. 2001.