CÁLCULO DE RUTAS EN SISTEMAS DE E-LEARNING UTILIZANDO UN ALGORITMO DE OPTIMIZACIÓN POR COLONIAS DE HORMIGAS

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

Memoria de Tesis Doctoral para optar al grado de

Doctor en Informática por la Universidad de Sevilla

presentada por

José Manuel Márquez Vázquez

Directores:

Dr. Juan Antonio Ortega Ramírez

Dr. Luis González Abril

Sevilla, marzo de 2012

Cálculo de Rutas en Sistemas de e-Learning Utilizando Algoritmos de Optimización por Colonias de Hormigas

José Manuel Márquez Vázquez

Memoria de Tesis

Índice

Índice de Figuras

Índice de Listados

Índice de Tablas

Índice de Acrónimos y Siglas

Siglas	Descripción
ACO	Ant Colony Optimization.
ACS	Ant Colony System
AJAX	Asynchronous Javascript And XML
AICC	Aviation Industry CBT Committee
ANSI	American National Standard Institute
AS	Ant System
ASALI	Ant System for Adaptive Learning Itineraries
ASCII	American Standard Code for Information Interchange
BPEL	Business Process Execution Language
CBT	Computer Based Training
CCSI	CopperCore Services Integration
CHAEA	Cuestionario Honey-Alonso de Estilos de Aprendizaje
CMI	Computer Managed Instruction
DEA	Diploma de Estudios Avanzados
DRI	Digital Repositories Interoperatibility
e-learning	Electronic Learning
ESB	Enterprise Service Bus
GCP	Graph Coloring Problem
GIA	Grafo de Itinerarios de Aprendizaje
GXL	Graph eXchange Language
GML	Graph Markup Language
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IDE	Integrated Development Environment
IMS CP	IMS Content Packaging
IMS LD	IMS Learning Design
IMS TI	IMS Tools Interoperability
ITEA	Information Technology for European Advancement
JSP	Job-shop Scheduling Problem
KLSI	Kolb's Learning Style Inventory
LCMS	Learning Content Management System
LMS	Learning Management System
LOM	Learning Object Metadata
LSD	Learning State Diagram
LTSA	Learning Technology Systems Architecture
LTSC	Learning Technology Standards Committee
m-learning	Mobile Learning
NP	No Polinomial
ODC	Osiris Domain Connector
OSGi	Open Services Gateway Initiative
PAP	Push Access Protocol
PC	Personal Computer
PDA	Personal Digital Assistant
QAP	Quadratic Assignment Problem

RFC	Requests For Comments
RSS	Really Simple Syndication
SCO	Shareable Content Object
SCORM	Shareable Content Object Reusable Model
SCSP	Shortest Common Supersequence Problem
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOP	Sequential Ordering Problem
SQL	Structured Query Language
TSP	Travelling Salesman Problem
UML	Unified Modeling Language
VPN	Virtual Private Network
VRP	Vehicle Routing Problem
WADL	Web Application Description Language
WS-CDL	Web Services Choreography Description Language
WSDL	Web Service Description Language
WWW	World Wide Web
XML	Extensible Markup Language
XPDL	XML Process Definition Language
XSD	XML Schema Definition

Índice de símbolos más utilizados

Símbolo	Descripción
α	Parámetro que controla la capacidad de un algoritmo ACO para explorar nuevos caminos. Es un potenciador del efecto de las feromonas.
β	Parámetro que controla la capacidad de un algoritmo ACO para explotar los caminos encontrados en base a la función heurística del problema. Es un potenciador del valor heurístico.
ε	Calificación obtenida en la evaluación final de un curso.
η	Función heurística de un problema.
ρ	Coeficiente de evaporación de las feromonas.
φ	Valor unitario de una feromona depositada por un alumno en un arco.
τ	Feromonas depositadas por el algoritmo ACO durante el cálculo de la solución óptima.
ω	Variable de calibración.
Φ	Cantidad de feromonas φ depositadas en un arco en un instante determinado. Es dependiente de la calificación obtenida.
$A_i[]$	Tabla de decisión del nodo *i*
$a_{ij}(t)$	Elemento de la tabla de decisión $A_i[]$ que indica la probabilidad de ser elegido el arco que va del nodo *i* al nodo *j*.
F	Función de ajuste. Es utilizada como función heurística que determina la *distancia* entre cursos.
N_i	Conjunto de nodos vecinos al nodo *i.*

Agradecimientos

En primer lugar, quiero agradecer la insistencia de mi madre y mi tía, que aunque rozando el agobio, han conseguido mantener en mí el espíritu de superación y sacrificio necesario para seguir adelante con esta dura tarea. Gracias.

Otro tanto por ciento significativo de mis agradecimientos son sin duda para Juan Antonio, Luis y Paco. Ellos son los que en los momentos de desánimo, cuando he estado a punto de tirarlo todo por la borda, han sabido darme los ánimos necesarios para seguir adelante. Gracias.

Y por último, y no menos importante, a mi esposa, Isabel, que ha sufrido como nadie mi falta de dedicación, mis momentos de ansiedad y mi ausencia en todos estos años. Mil gracias.

Prólogo

Esta tesis doctoral se enmarca dentro del campo del razonamiento automático y es fruto de la evolución de la investigación, a partir de un proyecto fin de carrera y su ampliación durante el período investigador del doctorado en su participación en proyectos de investigación europeos del VI Programa Marco. Durante este período se propuso una arquitectura orientada a servicios [1], basada en la especificación OSGi (inicialmente siglas de Open Services Gateway Initiative, http://www.osgi.org), para los sistemas de gestión del aprendizaje o LMS (del inglés *Learning Management System*) y se implementó un reproductor de cursos que incorporaba capacidades de adaptación y recomendación de contenidos según el perfil del usuario (alumno). Los perfiles de usuario eran bastante simples y se reducían a "usuario con necesidades especiales de visión", "usuario con necesidades especiales de audición" y "usuario estándar", además de un complemento adicional de personalización relativo a "gustos musicales" y "gustos de fondo de escritorio".

A raíz de este trabajo previo, surgió la hipótesis de que los perfiles de usuario no estuvieran limitados a hándicaps o a gustos, sino que también pudiera tenerse en cuenta el ritmo de aprendizaje o una clasificación en función del aprovechamiento de cursos anteriores. Para ello era necesaria la existencia de una secuencia de cursos, y así nació el escenario del itinerario formativo. Este escenario además parecía en un principio adecuado para resolver un problema detectado en nuestra primera aproximación a la adaptación de contenidos educativos. En nuestro trabajo de investigación anterior, la adaptación se realizaba por el reproductor del curso, estando los contenidos alternativos incluidos dentro del propio paquete de contenidos, creímos que este enfoque no era escalable si el número de perfiles de usuario y sus características aumentaban considerablemente. La posibilidad de describir itinerarios formativos con caminos alternativos y poder procesarlos para calcular el camino adecuado para cada alumno en función de sus necesidades de aprendizaje (ritmo, nivel de aprovechamiento, etc.) y luego aplicar las técnicas de adaptación de contenidos ya utilizadas, dio lugar al trabajo expuesto en esta tesis.

Capítulo 1. Introducción

Como indica el profesor Castells [2], la aparición de la *sociedad informacional* ha tenido un impacto de una magnitud similar al que tuvo en su momento la Revolución Industrial. Para Castells, la información, en su sentido más amplio, es decir, como comunicación del conocimiento, ha sido fundamental en todas las sociedades, por eso no es correcto llamar a la sociedad actual, sociedad de la información. En contraste, Castells establece un paralelismos con la distinción entre industria e industrial, para definir el atributo informacional cómo la forma específica de organización social en la que generación, el procesamiento y la transmisión de la información se convierten en las fuentes fundamentales de la productividad y el poder, debido a las nuevas condiciones tecnológicas que surgen en este nuevo período histórico. En efecto, la aparición de Internet permitió por primera vez la comunicación de muchos a muchos en un tiempo acordado y a una escala global. Su impacto en la sociedad ha sido tal que actualmente las principales actividades sociales, políticas, económicas y culturales del mundo están presentes y se organizan en Internet. Incluso podemos decir que la economía de los países desarrollados es dependiente de Internet, puesto que la desaparición de esta red ocasionaría una crisis económica inimaginable.

Con la misma velocidad que Internet ha penetrado en la sociedad desarrollada han ido apareciendo un gran número de sistemas de información destinados al aprendizaje. Su impacto en el plano laboral es igualmente considerable, al contar la mayoría de las grandes empresas con un departamento de formación que ofrece cursos *online* a sus empleados a través de un sistema de aprendizaje online, favoreciendo así la movilidad de éstos y la personalización de la formación a las necesidades de cada empleado. En algunos casos, los usuarios reciben la formación solicitada bajo demanda, cuando lo necesitan.

En los últimos años, la explosión global de la telefonía móvil ha abierto un nuevo horizonte para la educación a distancia a través de medios electrónicos, más conocida como *e-learning.* Con la llegada de dispositivos móviles cada vez más versátiles, redes de comunicaciones con mayor ancho de banda y el acceso a Internet desde un dispositivo móvil, la diferenciación que durante la última época se hacía entre las apliaciones de *e-learning* y de *m-learning* ha quedado carente de sentido. Hasta ahora se denominaba *m-learning* (del inglés *mobile learning*) a la metodología de

enseñanza y aprendizaje que se vale del uso de pequeños dispositivos móviles como teléfonos, agendas electrónicas y en general todo dispositivo de mano que tenga alguna forma de conectividad inalámbrica, ya sea autónoma o dependiente de un punto de acceso auxiliar. Se establecía una diferenciación con las metodologías *e-learning* por las limitaciones tecnológicas introducidas por los propios dispositivos o las redes de comunicaciones, que hacían inviables ciertos tipos de contenidos. Pero ahora con la aparición de los teléfonos inteligentes y tabletas (*tablet pc*) la frontera entre *e-learning* y *m-learning* ha desaparecido claramente, especialmente desde la fuerte irrupción de los *tablet pc*, si bien aún la tecnología soportada por todos los modelos difiere bastante como para aún necesitar un desarrollo personalizado para cada dispositivo.

Desarrollar un mismo curso y reproductores de contenidos para diferentes dispositivos encarece el producto de software, debido al mayor coste del desarrollo, de la gestión del proyecto y del mantenimiento asociado a las diferentes versiones. La necesidad de contar con sistemas de *e-learning* capaces de ofrecer cursos para cualquier dispositivo móvil es una prioridad para las empresas de hoy. La integración con un sistema de información móvil empresarial permitiría no sólo el acceso bajo demanda de los empleados a cursos de formación especialmente adaptados para su dispositivo móvil, sino que en algunos casos (dependiendo del tipo de dispositivo) podría facilitar la distribución de cursos formativos simultáneamente a un amplio número de empleados cuando la empresa así lo necesite, por ejemplo mediante el protocolo PAP[1]. Estas características aportan un alto valor añadido a los sistemas de información corporativos actuales. Sin embargo, en las empresas actuales es muy habitual que convivan aplicaciones y sistemas de información adquiridos o desarrollados a medida, según las necesidades de la empresa en cada momento, dando lugar a ecosistemas difíciles de gestionar y dificultando la integración de nuevos sistemas [3]. Si tenemos en cuenta que los gastos de integración superan en una proporción de entre cinco y veinte a los de desarrollo de nueva funcionalidad [4] no es despreciable tener en cuenta un enfoque de arquitectura orientada a servicios (SOA – Service Oriented Architecture) que facilite la integración de un LMS en el ecosistema tecnológico de la empresa.

Hasta ahora, la mayoría de los LMS existentes han adoptado SCORM (Shareable Content Object Reference Model) como estándar de facto para el intercambio de sus contenidos educativos [5]. SCORM combina varias especificaciones (de IMS, IEEE y AICC principalmente) y las particulariza para un caso concreto como

[1] Push Access Protocol. Wireless Application Protocol Forum, WAP-247-PAP-20010429-a, version 29 de Abril de 2001. http://www.openmobilealliance.org/tech/affiliates/wap/wap-247-pap-20010429-a.pdf

es el aprendizaje sobre la Web. SCORM define como secuenciar los contenidos dentro de un curso, como empaquetarlos, el marco de ejecución e incluso el protocolo de comunicación entre los contenidos de un curso con el LMS, lo que facilita que cualquier curso SCORM pueda utilizarse en cualquier LMS compatible. Sin embargo, dada la explosión exponencial del número de cursos de libre disposición en Internet y la creación de almacenes de contenidos didácticos, la problemática acerca de cómo organizar itinerarios compuestos por un conjunto de cursos, o como recomendar el curso más apropiado para cumplir con un cierto itinerario de formación, ha recabado el interés de la comunidad científica en la última década.

1.1 Planteamiento

De acuerdo con Brusilovsky [98], la adaptabilidad es de gran importancia en los procesos de enseñanza-aprendizaje online, debido a la potencial gran diversidad de la audiencia en cuanto a objetivos, estilos de aprendizaje, andamiaje cognitivo y necesidades especiales. Frente al clásico enfoque de un mismo curso o mismo itinerario para todos los alumnos, surge así la aproximación que se ha denominado Aprendizaje Adaptativo [6][8].

En los últimos años el interés se ha centrado tanto en la adaptación y recomendación de los contenidos internos de los cursos [9]-[11] como en la organización del itinerario completo [12], [13]. El entorno de aprendizaje, como elemento de comunicación adquiere un papel relevante, convirtiéndose en instrumento de ayuda a los alumnos mediante adaptaciones de presentación de los contenidos y navegación para crear rutas de aprendizaje adaptadas a cada individuo.

Ejemplos de adaptaciones de presentación son: proporcionar enlaces a explicaciones adicionales o ampliaciones que el alumno puede consultar a voluntad, ofrecer explicaciones comparativas que enfaticen las similitudes de los conceptos explicados con otros conocidos por el alumno, y ofrecer los conceptos ordenados por orden de relevancia [99], [100].

Las adaptaciones de presentación de los contenidos pueden afrontarse principalmente desde dos puntos de vista, siempre intentando ajustarse al estándar de facto de la industria, SCORM: desde fuera del paquete de contenidos, generando el LMS el paquete de contenidos ya adaptado a las necesidades del alumno [10], [13], o desde dentro del propio paquete de contenidos, incluyendo las reglas de adaptación necesarias [14]. Ambas soluciones concretan las necesidades en grupos que son

etiquetados con un perfil de usuario determinado de forma que sólo aquellos contenidos que coincidan con el perfil del alumno serán accesibles por el mismo. Los contenidos pueden etiquetarse con varios perfiles y un alumno puede a su vez asumir varios perfiles o roles, por ejemplo: perfiles relativos a su capacidad visual, a su nivel de aprendizaje, a los idiomas en que puede recibir un curso, etc. Estas situaciones en las que más de un contenido puede ser susceptible de ser recibido por el alumno justifican la necesidad de soluciones de recomendación de contenidos, que permitan decidir cuál de los posibles contenidos es el más adecuado [7],[16]. Esta decisión podría ofrecerse al alumno o tomarse automáticamente por el sistema, bien por el LMS teniendo en cuenta estadísticas de los resultados obtenidos por otros alumnos que anteriormente escogieron cualquiera de las posibilidades, o bien por la lógica contenida en el propio curso teniendo en cuenta una serie reglas.

La preocupación por dotar a los LMS de una inteligencia y capacidad de aprendizaje para decidir el itinerario formativo que mejor se adapte a las características del alumnado es uno de los problemas más complejos que puede abordarse en esta temática. Si representamos como un grafo los distintos caminos que pueden formar parte de este itinerario, puede plantearse como un problema No Polinomial (NP) de optimización de rutas, similar al problema del viajante o TSP (Travelling Salesman Problem), para el que los algoritmos probabilísticos pueden obtener una solución aproximada suficientemente buena. Destacan en la literatura las variantes del Algoritmo de Optimización por Colonia de Hormigas (ACO) de Marco Dorigo [15] y las diferentes soluciones basadas en algoritmos genéticos [16].

En cuanto a las adaptaciones de la navegación, Henze [101] define varios métodos para dar soporte a la navegación adaptativa: ruta guiada (ruta de contenidos estricta y conocida desde el inicio), ordenación adaptativa (los contenidos se ordenan según la relevancia para el alumno, teniendo en cuenta sus conocimientos previos), ocultación selectiva (se ocultan los contenidos didácticos no relevantes para el alumno), inclusión de mapas de navegación conceptuales que muestren gráficamente la relación entre los conceptos. Este último, constituye una primera aproximación a los grafos de itinerarios de aprendizaje, si bien aún no especifica cómo definir la distancia cognitiva entre dos conceptos interconectados.

A diferencia del cálculo del camino óptimo en el TSP, uno de los principales obstáculos para la aplicación de ACO en el cálculo de itinerarios formativos radica en la propia definición del concepto de distancia. Definir las distancias como valores estáticos facilita la simulación (realmente convierte el problema en una variante más del TSP)

pero lo hace independiente de la acción de los alumnos: el algoritmo terminará por converger hacia una solución óptima que permanecerá invariable independientemente de los resultados obtenidos por los alumnos en la evaluación de cada curso. Definir las distancias como probabilidad de éxito, dificulta la simulación con ACO pero puede facilitar la construcción de un sistema que pueda evolucionar en función de los resultados (calificación) obtenidos en la evaluación de cada curso por los alumnos, adaptando el camino óptimo al camino en el que mejor rendimiento (mayores calificaciones) se obtenga. Para la representación del problema suelen emplearse grafos dirigidos estocásticos.

Mediante grafos dirigidos pueden definirse las transiciones posibles entre los diferentes cursos que un alumno debe completar para adquirir unas competencias específicas, asociando los nodos a los cursos y asignando a cada arista una distancia. Para conseguir un modelo que pueda evolucionar en función de los resultados de los alumnos, estas distancias deberían aumentar o disminuir en función del rendimiento observado en cada tramo. Para ello se propone para su definición el uso de grafos dirigidos estocásticos en el que el concepto de distancia asociado a cada arista es una probabilidad. Estas probabilidades ayudan a determinar el siguiente curso que será entregado o recomendado al alumno, de forma que las aristas con probabilidades más altas serán elegidas más a menudo, y sus correspondientes unidades de aprendizaje serán habitualmente las entregadas. Este es el planteamiento inicial planteado en [12] y continuado en la tesis de S. Gutiérrez [18] aplicado a la ordenación de la secuencia de actividades dentro de un curso, lo que a nuestro juicio rompe la conformidad con el estándar de facto SCORM, que utiliza para este fin la especificación IMS Simple Sequencing (http://www.imsglobal.org/simplesequencing/).

Nuestro trabajo de investigación está centrado en la definición de un mecanismo que permita a un LMS calcular el itinerario formativo más adecuado para cada alumno, teniendo en cuenta los resultados obtenidos por el propio alumno como por el resto de semejantes, de forma que el camino óptimo pueda cambiar con el tiempo en función de los resultados que vayan obteniendo los propios alumnos.

1.2 Objetivos

La generación Z [19], [87] es la primera generación cuyos miembros han crecido con Internet y teléfonos móviles desde el primer día de sus vidas. Están acostumbrados a las tecnologías de la información pero éstas los hacen a su vez diferentes de sus antecesores. Los miembros de la generación Z, manejan mucha más

información, tienen avidez por aprender y han perdido la timidez para preguntar, para intervenir, para colaborar. En muchos casos, son ellos mismos quienes llevan sus preocupaciones a la clase, cambiando el rol del profesor desde un transmisor del conocimiento a un mediador de contenidos. Hoy en día, los miembros de más edad de esta generación están terminando sus estudios universitarios e incorporándose a sus primeros trabajos, y una nueva forma de trabajar está apareciendo. Por primera vez en la historia hay cuatro generaciones de adultos viviendo, trabajando y aprendiendo en la misma sociedad [19]. Esta situación se traduce en diferentes ritmos de aprendizaje y comportamientos frente a los sistemas de *e-learning*, ya sea por diferente velocidad de adaptación a estos sistemas de los miembros de cada generación o por las propias características personales de cada individuo, algunas de ellas muy relacionadas con la edad, como pueden ser deficiencias visuales o auditivas.

Es por eso que creemos que, en un sistema de *e-learning*, la definición de un contexto de aprendizaje que contemple diferentes caminos alternativos podría ayudar a mejorar la eficacia en el aprendizaje. Esta mejora es difícil de comprobar, por lo que nuestro objetivo principal se centra en estudiar la posibilidad de aplicar técnicas de optimización para el cálculo de itinerarios formativos adaptados a las necesidades de los alumnos, de forma que el camino evolucione en el tiempo al igual que cambian las necesidades, para adaptarse a los cambios que se produzcan, pues creemos que esto puede ayudar a la mejora en el aprendizaje. Para ello describiremos la arquitectura adecuada del sistema, cómo representar mediante grafos los diferentes caminos formativos y las relaciones entre cursos, cómo procesar este grafo y cómo hacer uso de algoritmos probabilísticos para dotar al LMS de la *"inteligencia"* necesaria para recomendar o asignar el itinerario formativo idóneo para cada alumno.

Para ello, el primer problema que nos encontramos es que los LMS actuales no trabajan con itinerarios formativos compuestos por varios cursos sino con cursos empaquetados según la especificación *IMS Content Packaging* (IMS CP). Todo el material didáctico relativo a un curso: textos, contenidos multimedia, gráficos, cuestionarios de evaluación… está incluido dentro del propio curso, siendo el LMS un mero distribuidor de estos cursos que además realiza el seguimiento de los cursos entregados a cada alumno, el porcentaje de curso "visualizado" por éste y en caso de haber realizado el test evaluación la calificación obtenida. Este enfoque nace de una primitiva concepción del e-learning en la que se concibe al alumno solitario frente a la pantalla del ordenador leyendo un texto con los contenidos del curso. Este enfoque conductista ha dado paso a una nueva concepción constructivista en la que la colaboración con semejantes y profesores es muy importante porque sirve de andamiaje en el proceso cognitivo del alumno. De ahí que todos los LMS han ido incorporando

paulatinamente herramientas que faciliten la colaboración: Wikis, blogs, foros, chats, videoconferencias, etc.

Frente al diseño centrado en el curso, queremos proponer un diseño centrado en el itinerario formativo, que facilite la federación de repositorios de contenidos [20] y la definición de itinerarios formativos que ofrezcan la posibilidad de escoger caminos alternativos en función de las necesidades de cada alumno y de su ritmo de aprendizaje. Con este fin definiremos una forma de representación basada en grafos dirigidos estocásticos de los itinerarios formativos y una definición formal de los mismos. Una vez representado el problema describiremos como integrarlo en un LMS como un módulo adicional, intentando que afecte lo menos posible a su funcionalidad.

Finalmente, nuestro principal objetivo es comprobar si es posible aplicar ACO a este escenario teniendo en cuenta una definición probabilística de la distancia, dependiente del rendimiento medio observado en los alumnos. Igualmente mostramos otras vías de aplicación de ACO a e-learning.

1.3 Período de Investigación

Durante el período investigador para la obtención del Diploma de Estudios Avanzados (DEA) se analizaron varios LMS[2] de código abierto realizándose una comparativa de la funcionalidad ofrecida por cada uno de ellos. A la hora de establecer una comparativa de LMS, es necesario tener en cuenta qué puede considerarse un LMS y qué no. Para ello, se tuvo en cuenta la recomendación ISO/IEC 9126[3] que proporciona seis medidas de calidad de cara a la evaluación del software y a la "definición funcional mínima de un LMS". En la comparativa se analizan los estándares de la industria soportados por varios LMS, así como el principal lenguaje de programación empleado para su implementación y las fuentes de almacenamiento de datos soportadas [5]. Algo que merece destacar es el apartado de características funcionales extras que ofrece cada plataforma, aquellas características que sin duda cualquier usuario incluiría en su

[2] .LRN (http://dotlrn.org), ATutor (www.atutor.ca), Claroline (www.clarolone.net), CopperCore (http://coppercore.sourceforge.net), Dokeos (www.dokeos.com), Eledge (http://eledge.sourceforge.net/), Ilias (www.ilias.de), Moodle (www.moodle.org), OLAT (www.olat.org) y Sakai (http://www.sakaiproject.org/).

[3] ISO/IEC 9126: The Standard of Reference. (1991)
http://www.cse.dcu.ie/essiscope/sm2/9126ref.html

descripción de la plataforma. Es fácil observar como herramientas de chat, foros, agendas y calendarios o cuestionarios de autoevaluación son funcionalidades añadidas en la mayoría de plataformas de e-learning analizadas. Son herramientas extras que no pertenecen exclusivamente a los LMS sino que pueden encontrarse en cualquier otra aplicación como el correo electrónico (por ejemplo Lotus Notes 8 incorpora Sametime como herramienta de chat), gestores de contenidos, etc. Incluso podrían tener entidad por sí solas. Es fácil pensar que estas aplicaciones extras podrían ya existir en el ecosistema de aplicaciones de una empresa y que sería más adecuado poder integrarlas con el LMS. Igualmente durante el periodo investigador previo a la obtención del DEA buscamos facilitar la integración de nuestra propuesta en una solución SOA, planteando la distribución de los cursos SCORM como *bundles* OSGi.

OSGi se ha convertido en la especificación más ampliamente aceptada para construir herramientas con una doble perspectiva: orientación a componentes (el bundle) y orientación a servcios. El número y variedad de aplicaciones que se ejecutan sobre un framework OSGi aumenta día a día: la suite Lotus Notes, los IDE Eclipse y NetBeans, algunos ESB como Service Mix o Fuse, servidores de aplicaciones como Glassfish, JBoss, JOnAS, WebLogic o WebSphere. Desde nuestro punto de vista, una plataforma orientada a servicios facilitará la utilización tanto on-line como off-line de la misma permitiendo que el usuario pueda seguir los contenidos de un curso sin depender de la conexión a Internet. Este aspecto es fundamental para implementar reproductores de cursos para dispositivos móviles que frecuentemente sufren cortes de la conexión debido a la infraestructura o a la propia movilidad, al pasar por zonas de sombra (túneles, metro…). Se comprobó la compatibilidad del formato de empaquetado de IMS CP con OSGi y la posibilidad de distribución de cursos IMS CP como bundles OSGi. En efecto, el descriptor de IMS CP describe el contenido de un curso, pero a pesar de ser léxica, sintáctica y semánticamente diferente al descriptor de OSGi, el tener nombres de fichero diferentes y el hecho de que el fichero que empaqueta los contenidos utilice el mismo formato de compresión (ZIP) posibilita la coexistencia de ambos en un mismo paquete de contenidos [5], lo que permite distribuir cursos SCORM indistintamente como bundles OSGi y a su vez incluir lógica Java en el propio paquete de contenidos, tal y como resume la Figura 1.

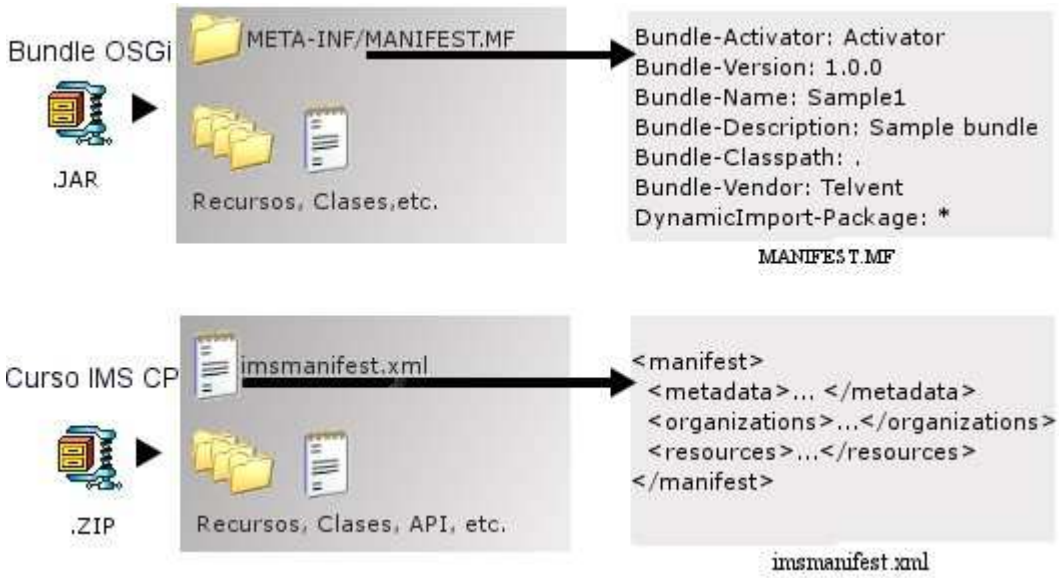


Figura 1: Estudio de compatibilidad IMS CP y OSGi

En el marco del proyecto ITEA OSIRIS[4] se desarrolló Osiris Domain Connector (ODC), un bus de servicios especialmente indicado para facilitar la integración de los componentes de un LMS como bundles y servicios OSGi, con la capacidad de funcionar como sistema distribuido en tantos nodos de una red local (o VPN) como fuera necesario [24], lo que supuso el primer paso para el desarrollo de un LMS distribuido. Una visión más amplia de la arquitectura y la integración segura de servicios en tiempo real, fue publicada también en [25].

Como ya hemos comentado anteriormente, uno de los problemas es que los LMS no se han diseñado para el seguimiento de itinerarios de aprendizaje, sino para la distribución y mínimo seguimiento de los cursos y sin tener en mente la necesidad de contar con cursos independientes (aunque sea por razones de ahorro de espacio) especialmente adaptados a las necesidades especiales de cada alumno (hándicap visual o auditivo, etc.). En este sentido, el diseño del proceso de enseñanza-aprendizaje se asemeja mucho a un diagrama de flujo o *workflow*. De forma general, el proceso de enseñanza-aprendizaje, como cualquier proceso, puede definirse como una secuencia de actividades en las que diferentes entidades (personas, máquinas, etc.) colaboran para conseguir un determinado objetivo. Un ejemplo de proceso, conocido como *proceso de negocio*, sería aquel que describe las actividades de una organización para satisfacer las necesidades de sus clientes.

[4] Consultar información sobre proyectos de I+D+i en los que ha participado el autor en la pág. 154.

Desde que a finales de los 80, M. Hammer propusiera el concepto de *reingeniería de procesos* para mejorar los procesos de negocio de las empresas [5], se han propuesto numerosas técnicas para modelarlos, facilitando así la comprensión de los mismos y, en su caso, ayudar a la mejora de éstos para alcanzar sus objetivos de manera más eficaz y eficiente. El consorcio *Workflow Management Coalition* (http://www.wfmc.org/) ha establecido diferentes normas para poder compartir información entre sistemas de este tipo. Actualmente, el uso de XML (*eXtensible Markup Language*, http://www.w3.org/XML/) en su especificación XPDL (XML *Process Definition Language*. http://www.wfmc.org/xpdl.html), es el formato preferido para representar de forma textual los modelos de los procesos de negocio, y algunos trabajos recientes ya plantean escenarios para *e-learning* [21]. Aunque existen muchas técnicas de representación gráfica de los procesos, la notación de UML (*Unified Modeling Language*. http://www.uml.org) es la más utilizada.

Para salvar las limitaciones que SCORM conlleva nació la especificación IMS LD (IMS *Learning Design*, http://www.imsglobal.org/learningdesign/). IMS LD define cómo describir y codificar las metodologías de aprendizaje y cómo incorporarlas en una solución *e-learning*. Soporta el uso de un amplio rango de pedagogías para aprendizaje on-line y permite definir nuevas metodologías pedagógicas haciendo uso de un lenguaje genérico y flexible diseñado para permitir la definición de muchas pedagogías diferentes. La Versión 1 es la única hasta la fecha, y fue publicada en Enero de 2003. IMS LD fue muy bien recibida por los educadores, especialmente pedagogos, pues su objetivo radica más en el diseño de pautas metodológicas que en la mera distribución de los contenidos. Algunos de los LMS ya la soportan pero no acaba de imponerse a SCORM por su mayor coste de producción de los cursos, falta de herramientas, y mayor complejidad técnica. IMS LD define su meta-modelo utilizando notación UML y proporciona la definición formal para el modelado de procesos de enseñanza-actividades en un esquema XSD (XML *Schema Definition*. http://www.w3.org/XML/).

Ante la carencia de herramientas de autor (modelado de procesos de enseñanza-aprendizaje, edición itinerarios formativos, creación y edición de contenidos…) conformes con IMS LD y la abundancia de herramientas de gestión de procesos de negocio, la idea es clara: cómo transformar procesos IMS LD en XPDL [22]. Esta es una de las líneas de investigación abiertas recientemente en relación con nuestro trabajo.

Otra línea de investigación relacionada con el problema de definición del itinerario formativo, está relacionada con la incorporación de características a cada nodo del grafo que aporten información de cara a la toma de decisiones ante caminos alternativos. En este sentido se colaboró con la Universidad Politécnica de Valencia para estudiar la viabilidad de usar modelado de características y el software MosKitt (*Modeling Software Kit*. http://www.moskitt.org/) para conseguir nuestro objetivo, publicándose el resultado en [23].

Finalmente hemos comenzado a investigar la aplicación de ACO para el cálculo de caminos óptimos en grafos estocásticos que definen itinerarios formativos con caminos alternativos [26] estudio que estamos ampliando para su publicación en revistas internacionales relevantes.

El siguiente capítulo resume el estado del arte sobre la optimización basada en algoritmos de colonias de hormigas.

Capítulo 2. Optimización por Colonias de Hormigas

2.1 Introducción

Inspirados en los estudios sobre el comportamiento social de las hormigas de los entomólogos Pierre-Paul Grassé [36], Bert Hölldobler y Edward Osborne Wilson [126], los algoritmos de optimización por colonias de hormigas o ACO (del inglés Ant Colony Optimization) fueron propuestos por Marco Dorigo [15] como para la optimización de problemas NP-completos como el problema del Viajante (TSP) [27] o el de asignación cuadrática (QAP) [28]. Desde entonces ha existido un gran interés en la comunidad científica por aplicar estos algoritmos a muy diversos problemas de optimización: planificación de tareas (JSP) [29]; enrutamiento de redes de Comunicaciones [30]; coloreado de Grafos [31]; y en el área del aprendizaje automático, especialmente en el diseño de algoritmos de aprendizaje para estructuras de representación del conocimiento: reglas clásicas [32], lógica difusa [33] y redes bayesianas [34].

Los algoritmos ACO están inspirados en el comportamiento colaborativo de las hormigas. Las hormigas son insectos sociales, que viven en colonias y para los que la preservación de la misma está por encima de la supervivencia de cualquier individuo. Uno de los comportamientos más interesantes de las colonias de hormigas, a parte de su alta jerarquización, es el relacionado con la búsqueda de alimento, en especial como encuentran siempre el camino más corto entre el nido y la fuente de alimento.

El método empleado por las hormigas se basa en el seguimiento del rastro de una sustancia química que ellas mismas secretan por dónde pasan: las feromonas. Ante varios caminos posibles, una hormiga huele los diferentes rastros dejados por sus predecesores, y tiende a elegir, con mayor probabilidad, los de mayor concentración de feromonas. El rastro de feromona ayuda a las hormigas a identificar el camino hacia la fuente de alimento descubierta por sus compañeras de colonia y también el camino de vuelta al nido.

Experimentos previos con hormigas reales [35] demostraron que en una bifurcación con dos ramas simétricas de idéntica longitud (Figura 2), tras un intervalo de tiempo en el que las hormigas escogían indistintamente una u otra rama, todas las hormigas tendieron a seguir la misma bifurcación, lo que hizo pensar que la probabilidad de escoger una u otra rama dependía del número de hormigas que hubieran escogido cada rama previamente.

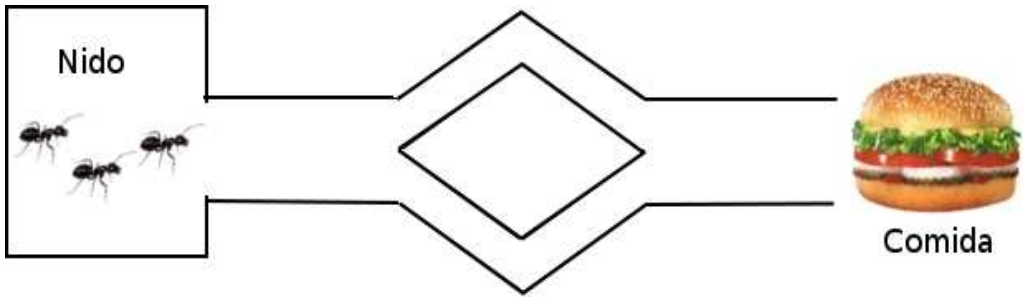


Figura 2: Dibujo que representa el experimento del *puente binario* de Deneubourg.

Así, en este experimento la probabilidad de selección de una rama u otra es del 50% cada vez que se realice el experimento, dependiendo la selección de una u otra de la distribución de individuos que elijan una u otra. Puesto que cada hormiga deposita feromonas al desplazarse, cuantas más hormigas escojan una rama, mayor probabilidad habrá de ser esa rama la elegida por el resto en detrimento de la otra, al concentrarse en ella mayor cantidad de feromonas. Asumiendo que el nivel de feromonas en un camino es proporcional al número de hormigas que han pasado por él, el modelo probabilístico que describe la probabilidad, $P_u(m)$, de que la hormiga (m+1)-ésima de la colonia elija el camino superior es:

$$P_u(m) = \frac{(U_m + k)^h}{(U_m + k)^h + (L_m + k)^h} \quad (1)$$

dónde U_m y L_m son el número de hormigas que han escogido el camino superior y el inferior respectivamente, después de haber tomado la decisión m hormigas. Las variables h y k sirven para ajustar el modelo a las observaciones de hormigas reales. A través de simulaciones utilizando el método Monte Carlo, Deneubourg y sus colegas [35] encontraron que para valores de $k=20$ y $h=2$, el modelo se ajustaba a los datos experimentales.

Al modificar el experimento, de forma que los caminos no tuvieran igual longitud (Figura 3), se observó que en todos los casos, las hormigas acababan escogiendo el camino más corto.

En este caso, las primeras hormigas en llegar a la comida suelen ser aquellas que toman los caminos más cortos, y al iniciar el camino de vuelta, el nivel de feromona en estos caminos también es mayor, por lo que además suelen volver por el mismo camino, aumentando aún más el nivel de feromona existente y estimulando a las demás hormigas. En este experimento, el tiempo de fluctuaciones aleatorias se reduce considerablemente respecto al caso de caminos de igual longitud.

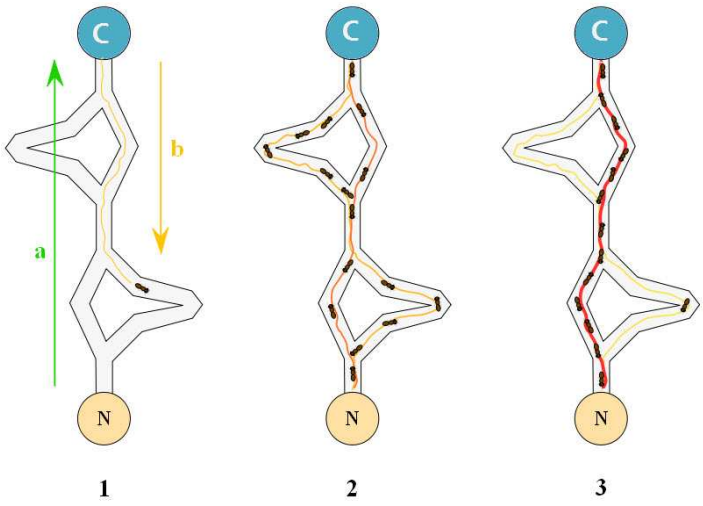


Figura 3: Representación del experimento con caminos alternativos de diferente longitud. 1. La primera hormiga sigue un camino elegido aleatoriamente. 2. El resto de la colonia se distribuye entre los caminos posibles. 3. Finalmente la mayor concentración de feromonas en los caminos más cortos hace que prácticamente todas sigan el mismo camino.

Los experimentos descritos revelan una especie de mecanismo colaborativo de optimización en el que cada individuo aporta una pequeña contribución. Lo interesante es que, aunque un individuo podría encontrar por sí solo la solución, la búsqueda del camino más corto sólo es posible con la colaboración de toda la colonia. El biólogo francés Pierre-Paul Grassé, tras observar el comportamiento de las termitas, denominó a este comportamiento *estigmergia*[5], definiéndolo como "*estimulación de los trabajadores a través del rendimiento obtenido*" [36]. En cierto modo, esta estimulación y la respuesta a los estímulos, puede entenderse como una forma de comunicación indirecta, caracterizada por la naturaleza física de la información liberada (las

[5] Del griego *estigma*: marca, señal; y *ergon:* trabajo, acción.

feromonas), que significa una modificación física del entorno; y la naturaleza local de la información, que requiere que otros visiten el mismo entorno para tener acceso a dicha información.

Otra importante característica destacada por Marco Dorigo es la *autocatálisis* o *realimentación positiva* [37] y la evaluación implícita de soluciones. Esto es: implícitamente se busca siempre el camino más corto como solución, puesto que cuanto más corto sea, más probable es que se complete el camino con mayor rapidez, y por tanto, mayor refuerzo de feromonas recibirá, animando con más fuerza a un número mayor de hormigas a optar por él. Este comportamiento contribuye rápidamente a la convergencia hacia una solución óptima.

2.2 Características Principales

La Optimización por Colonias de Hormigas es una meta-heurística que utiliza un conjunto de agentes que simulan el comportamiento cooperativo de las colonias de hormigas para encontrar buenas soluciones en problemas de optimización de elevada complejidad combinatoria. Se inspira en el comportamiento social de las hormigas, del que toma sus ideas principales:

- **Cooperación**: La cooperación es un concepto clave en el diseño de algoritmos ACO. La solución se alcanza mediante las pequeñas contribuciones de cada individuo. Aunque un individuo por sí sólo puede encontrar una solución suficientemente buena, no es sino la suma de las contribuciones de todos lo que determinará la solución final que terminará adoptando la colonia.
- **Búsqueda de caminos cortos y movimientos locales**: Tanto las hormigas reales como las artificiales de los algoritmos ACO tienen el mismo objetivo: encontrar el camino más corto entre el nido (origen) y la fuente de alimento (objetivo). Y ambas lo hacen del mismo modo: moviéndose paso a paso, de un estado a cualquiera de sus estados adyacentes. Ninguna de las dos salta ni vuela, y aunque existen hormigas aladas, estas no participan de la tarea de búsqueda de alimentos.
- **Rastro de feromonas**: El rastro de feromonas es la forma de comunicación indirecta que utilizan las hormigas. No existe una comunicación directa entre los individuos sino que la única vía de comunicación entre ellos se realiza mediante la modificación del entorno que supone el depósito de feromonas. En los algoritmos ACO esta información suele modelarse mediante un valor

numérico que es percibido por cada hormiga artificial como el valor de una función que representa el tráfico histórico de toda la colonia por ese punto hasta el momento. La evaporación de las feromonas (su disminución con el tiempo si no hay aportes continuos) ayuda a que poco a poco se olvide el pasado y facilite, gracias a procesos de decisión estocásticos, la búsqueda de nuevas soluciones.

- **Decisión probabilística y miope**: Tanto las hormigas reales como las artificiales de ACO no ven más allá de los posibles estados adyacentes y la decisión del siguiente estado se realiza aplicando una política de decisión probabilística. No existen otros mecanismos más avanzados de decisión que analicen las consecuencias de su posible decisión, como por ejemplo se aplica en la inteligencia artificial aplicada a los juegos de ajedrez.

En cambio las hormigas artificiales sí pueden enriquecerse con algunas habilidades que no se encuentran en su estado natural en las colonias reales para hacerlas más eficientes. Algunas de las características de estos agentes son:

- A diferencia de las hormigas reales, las hormigas artificiales empleadas en los algoritmos ACO viven en un mundo discreto, y sus movimientos están limitados a una transición entre conjuntos discretos de estados.
- Las hormigas artificiales poseen memoria, de forma que conocen el estado actual en el que se encuentran y sus acciones y estados anteriores.
- Se puede definir una función que determine la cantidad de feromona a depositar en función del porcentaje de éxito alcanzado o de la calidad de la solución encontrada. En realidad, esto no es una diferencia, puesto que algunas especies de hormigas reales también presentan este comportamiento, variando la cantidad e intensidad de las feromonas depositadas en función del alimento encontrado. Estudios recientes [38] han corroborado que la hormiga argentina (*Linepithema humile*) cambia la composición de su dieta según la estación del año, prefiriendo en invierno alimentos ricos en carbohidratos, como el azúcar de caña o la miel, frente a alimentos ricos en proteínas como el huevo.
- Se puede determinar el momento en el que realizar depósito de feromonas. En algunos casos podría decidirse que el depósito de feromona sólo se lleve a cabo una vez encontrada una solución y no con cada movimiento, como ocurre en las hormigas reales.
- Para mejorar la eficiencia general del sistema, los algoritmos ACO pueden enriquecerse con otras técnicas como optimización local o *backtracking* que no están presentes en las hormigas reales.

2.3 La meta-heurística ACO

En los algoritmos ACO, una o más colonias, compuestas de un número determinado de hormigas, buscan una buena solución a problemas de optimización. Cada hormiga busca una solución o un componente de ésta, a partir de un estado inicial que depende del problema. Cada hormiga, mientras construye su solución, obtiene información del entorno, de las características del problema y de su propio rendimiento, y usa toda esta información para modificar la representación del problema e influir así en la visión del mismo que tienen el resto de hormigas. Las hormigas pueden actuar concurrentemente y de forma independiente, pero no intercambian información directamente, siguiendo así el paradigma de la *estigmergia* que gobierna la comunicación entre las hormigas.

La construcción de la solución sigue un proceso incremental, expresándose como la ruta de menor coste a través de los estados que componen el problema, de acuerdo con las restricciones del mismo. El problema se representa como un grafo $G = (C, L)$, en el que cada $c_i \in C$ es una componente posible de la solución (un nodo del grafo) y cada $l_{ij} \in L$ una transición posible de la componente i a la j (arco del grafo). Tanto los nodos del grafo como los arcos pueden tener asociado un rastro de feromona, τ_i si nos referimos al rastro acumulado en el nodo c_i, o τ_{ij} si se trata del rastro acumulado en el arco l_{ij}. De la misma forma, ambos pueden tener asociados valores heurísticos, η_i y η_{ij} respectivamente, que representan la visibilidad de cada componente o trayecto, generalmente en función del coste o de una estimación del coste asociado de elegir el siguiente estado.

Cada hormiga construye una solución a través de una secuencia finita de movimientos entre estados vecinos. Cuando existan varias posibilidades de movimientos, se elegirá el estado siguiente aplicando una política de búsqueda local estocástica, en la que la probabilidad de visitar cada estado vendrá determinada por la propia memoria de la hormiga y por la información del entorno (feromonas). La cantidad de feromonas a liberar por una hormiga, así como el momento de su liberación dependen de las características y restricciones del problema, así como del diseño de la implementación. Las feromonas pueden ser liberadas con cada movimiento (liberación paso a paso) o una vez que la solución has sido construida (liberación retardada). La cantidad de feromona también puede variarse en función de la calidad de la solución construida. Además, cada hormiga construye una tabla de decisión utilizando una función que tiene en cuenta el nivel de feromona existente para cada movimiento como un conjunto de valores heurísticos. De esta forma se determina el siguiente estado para cada hormiga.

El grado de aleatoriedad en la decisión de los movimientos y la rapidez de evaporación de las feromonas determinan el balance entre la exploración de nuevos caminos y la explotación del conocimiento acumulado.

Hay numerosas implementaciones de esta meta-heurística aplicadas cada una de ellas a diferentes problemas de optimización combinatoria. El Listado 1 muestra el pseudocódigo de la meta-heurística ACO definida por Marco Dorigo.

```
procedimiento ACO()
 mientras (! criterio_finalizacion)
  planificar
  gestionar_hormigas();
  evaporar_feromonas();
  ejecutar_acciones_globales(); // opcional
  fplanificar
 fmientras
fprocedimiento
```

Listado 1: Meta-heurística ACO

La meta-heurística define las tres actividades básicas que deben realizarse: gestionar las hormigas encargadas de buscar soluciones locales, evaporar feromonas y realizar acciones globales, pero no impone la forma de llevarlas a cabo. De hecho, la sentencia `planificar` da libertad para decidir si debe existir algún tipo de paralelismo entre ellas o deben sincronizarse de alguna manera. La función `ejecutar_acciones_globales` representa a un conjunto de acciones opcionales que podrían ser realizadas por un proceso con conocimiento global del entorno, una especie de observador externo. Por ejemplo, a partir del comportamiento de todas las hormigas este observador podría reunir información útil y decidir depositar feromonas adicionales, condicionando de esta forma la búsqueda local de cada hormiga desde una perspectiva diferente.

La planificación de las actividades definidas por la meta-heurística ACO depende del problema y da lugar a diferentes algoritmos. Mientras algunos problemas tienen un único estado de inicio, otros cuentan con múltiples estados posibles de inicio.

Por ejemplo en el experimento de la búsqueda de alimento, el nido es siempre un mismo punto de partida, mientras que en el TSP existe la posibilidad de imponer como condición una ciudad de partida o buscar el camino más corto que recorra todo el conjunto de ciudades, sin importar la ciudad de inicio. El TSP fue el primer problema al que se aplicó ACO, siendo el algoritmo *Ant System* (Figura 4) la primera implementación de ACO [39] y la base para muchos algoritmos similares.

2.4 Algoritmos ACO

Los algoritmos ACO pueden aplicarse para resolver problemas de optimización estáticos y dinámicos. Los problemas estáticos son aquellos en los que las características del problema permanecen invariables mientras se busca una solución, mientras que en los dinámicos las características que determinan el problema a resolver pueden variar. Por ejemplo, el TSP es un clásico problema estático en el que tanto las ciudades como las distancias entre ellas son parte de la definición del problema y no varían. En cambio, si tenemos en cuenta el tráfico existente en cada trayecto entre ciudades y entendemos el concepto de distancia como el tiempo requerido para llegar de una a otra estaremos ante el paradigma de problema combinatorio dinámico, puesto que la distancia será función de múltiples variables como el día de la semana, la hora del día, las condiciones meteorológicas o de contingencias imprevistas como accidentes, manifestaciones, obras, etc. Las modificaciones topológicas (por ejemplo, en el TSP, la construcción o eliminación de una nueva carretera entre dos ciudades), pueden considerarse transiciones pertenecientes a un mismo tipo de problema. Gracias a su naturaleza adaptativa, los algoritmos ACO son especialmente adecuados para problemas dinámicos.

En función de la clase de problema combinatorio que se pretende resolver (estático o dinámico) existen en la literatura numerosas implementaciones de algoritmos ACO. Las Tabla 1 muestra los principales algoritmos ACO para la resolución de problemas estáticos. En relación con la resolución de problemas dinámicos, los principales algoritmos ACO existentes en la literatura se citan en la Tabla 2.

Más recientemente se ha comenzado a aplicar estos algoritmos a problemas con múltiples objetivos (Tabla 3).

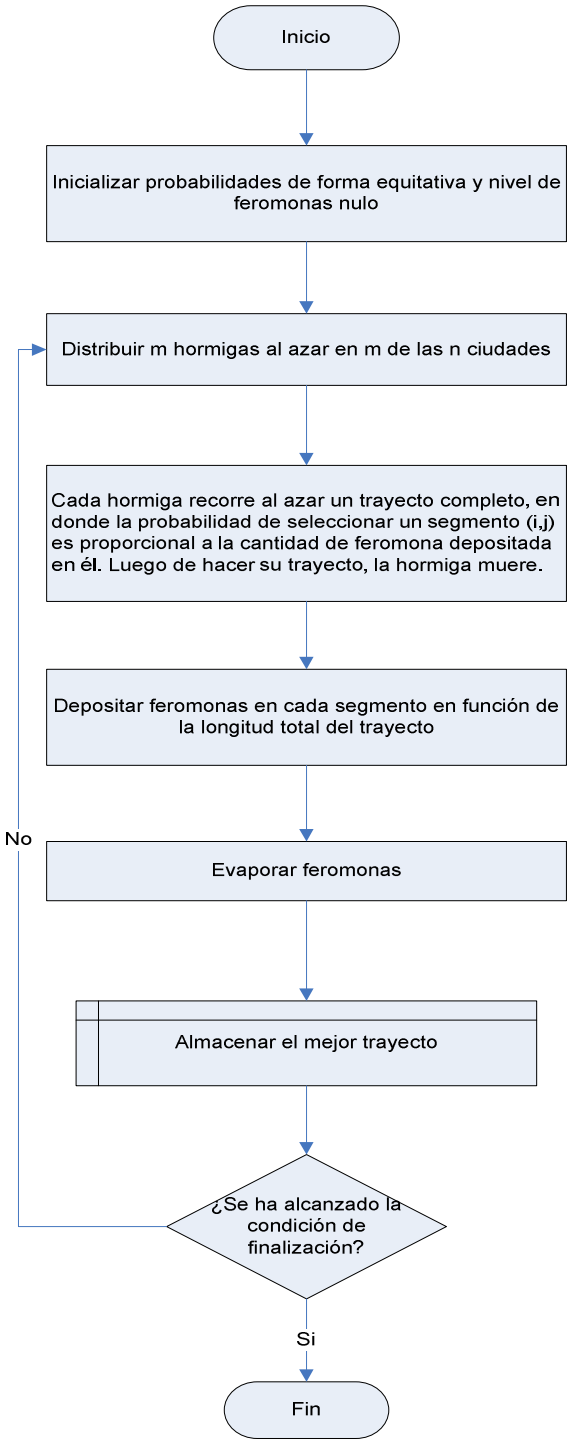


Figura 4: Algoritmo AS para el TSP. (M. Dorigo et al., 1996)

Tabla 1: Algoritmos ACO para resolución de problemas estáticos de optimización combinatoria

Problema	Nombre	Autores	Año	Referencia
TSP	AS	Dorigo et al.	1991	[37]
	Ant-Q	Gambardella & Dorigo	1995	[40]
	ACS, ACS-3	Dorigo & Gambardella	1997	[27]
	MMAS	Stützle & Hoos	1997	[41]
	AS_{rank}	Bullnheimer et al.	1997	[42]
QAP	AS-QAP	Maniezzo & Colorni	1998	[43]
	MMAS-QAP	Stützle & Hoos	1998	[44]
	HAS-QAP	Gambardella et al.	1999	[45]
JSP	AS-JSP	Colorni et al.	1994	[46]
VRP	AS-VRP	Bullnheimer et al.	1998	[47]
	HAS-VRP	Gambardella et al.	1999	[48]
SOP	HAS-SOP	Gambardella & Dorigo	1997	[49]
GCP	ANTCOL	Costa & Hertz	1997	[31]
SCSP	AS-SCS	Michel & Middendorf	1998	[50]

TSP: Travelling Salesman Problem; QAP: Quadratic Assignment Problem; JSP: Job-shop Scheduling Problem; VRP: Vehicle Routing Problem; SOP: Sequential Ordering Problem; GCP: Graph Coloring Problem; SCSP: Shortest Common Supersequence Problem.

Tabla 2: Algoritmos ACO para resolución de problemas dinámicos de optimización combinatoria

Problema	Nombre	Autores	Año	Referencia
Enrutado orientado a la conexión	ABC	Schoonderwoerd et al.	1996	[30]
	ASGA	White et al.	1998	[51]
	AntNet-FS	Di Caro & Dorigo	1998	[52]
	ABC-smart	Bonabeau et al.	1998	[53]
Enrutado no orientado a la conexión	AntNet & AntNet-FA	Di Caro & Dorigo	1997	[54]
	Regular ants	Subramanian et al.	1997	[55]
	CAF	Heusse et al.	1998	[56]
	ABC-backward	van der Put & Rothkrantz	1999	[57]

Tabla 3: Algoritmos ACO para problemas multi-objetivo

Nombre	Autores	Año	Referencia
MOAQ	Mariano & Morales	1999	[58]
BiAnt, BiMC	Iredi et al.	2001	[59]
PACO	Doerner et al.	2002	[60]
MOACS	Barán & Schaerer	2003	[61]
COMPETants	Doerner et al.	2003	[63]
M3AS	Pinto et al.	2005	[62]
MOA	Gardel et al.	2005	[64]
MAS	Paciello et al.	2006	[65]

2.5 Aplicación a problemas estáticos de optimización

Para problemas estáticos, la implementación de la heurística ACO es relativamente directa una vez se ha modelado el problema y definidas las relaciones de vecindad entre los nodos y la función para la transición de estados que será usada localmente.

2.5.1 El Problema del Viajante

El Problema del Viajante (TSP) es uno de los problemas más famosos en el campo de la optimización combinatoria computacional. Al ser un problema de búsqueda de caminos cortos, objetivo principal presente en la búsqueda de alimento de las hormigas reales, fue el primer problema NP-completo al que se aplicaron algoritmos de optimización por colonias de hormigas.

Este problema se define como sigue: Sean *n* ciudades de un territorio, el objetivo es encontrar una ruta que, comenzando y terminando en una ciudad concreta, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida por el viajante. Por ejemplo, el conjunto de ciudades de la Tabla 4 podría ser un escenario simple del problema.

Tabla 4: Matriz de adyacencias para un escenario simple del TSP

	Madrid	Barcelona	Valencia	Sevilla	Bilbao	Murcia	Vigo	Cadiz
Madrid	0	619	358	533	398	411	591	655
Barcelona	619	0	365	1015	610	633	1164	1135
Valencia	358	365	0	659	612	259	948	779
Sevilla	533	1015	659	0	862	526	884	123
Bilbao	398	610	612	862	0	813	700	982
Murcia	411	633	259	526	813	0	996	611
Vigo	591	1164	948	884	700	996	0	1006
Cadiz	655	1135	779	123	982	611	1006	0

El TSP es un clásico problema combinatorio de variables discretas, en el que la solución óptima sería aquella permutación de n ciudades R = $\{c_0, c_1, \ldots, c_n\}$ tal que la distancia total (2) del camino resultante para dicha permutación sea mínima.

$$d_R = \sum_{i=0}^{n-1} d[c_i, c_{(i+1) mod\ (n)}] \qquad (2)$$

El problema puede representarse como un grafo conexo y ponderado G=(C, L) en el que C es el conjunto de nodos, cada uno de ellos representando una ciudad, y L el

conjunto de lados o aristas del grafo, cada una de ellas representando un camino para hacer el trayecto entre dos ciudades y cuyo peso reflejaría la distancia entre ellas en el sentido indicado[6]. El objetivo sería encontrar el ciclo hamiltoniano de menor distancia. Un camino hamiltoniano en un grafo es un recorrido que transcurre por una sucesión de aristas adyacentes, que visita todos los vértices del grafo una sola vez. Si además el último vértice visitado es adyacente al primero, el camino es un ciclo hamiltoniano.

Dependiendo del problema la representación del grafo como una simple matriz de adyacencias (Tabla 4) puede ser factible, pero no es la manera más eficiente de representación si no existen conexiones entre muchos nodos, al ser una matriz cuadrada. Una matriz de incidencias permite representar por su parte caminos alternativos (lados paralelos) de diferente distancia entre dos nodos y permite un manejo mucho más eficiente de la memoria. El uso de tipos abstractos como listas doblemente enlazadas es lo más recomendable desde el punto de vista de la eficiencia, aunque complica su implementación.

Veamos a continuación como aplicar la optimización basada en el comportamiento de las hormigas al TSP.

Sistemas de Hormigas

El algoritmo de Sistemas de Hormigas o Ant System (AS) fue el primer algoritmo ACO [37] y se ha convertido en el prototipo para otros algoritmos posteriores. El algoritmo ejecuta *t* iteraciones. En cada iteración *m* hormigas construyen un camino en una serie de pasos. Cuando una hormiga decide moverse (aplicando una regla probabilística) de la ciudad *i* a la *j*, añade el arco l_{ij} a su camino en construcción.

Puesto que el mecanismo de depósito retardado demostró ser el más eficiente [39] en este algoritmo, las feromonas se depositan una vez que una hormiga ha completado un camino candidato a la solución. La cantidad de feromonas depositadas, $\tau_{ij}(t)$ en un arco l_{ij} representa la preferencia de selección de escoger la transición desde la ciudad *i* a la *j*, y la cantidad de feromonas depositadas varía en función de la calidad de la solución, siendo mayor cuanto más corta es la distancia del total del camino

[6] Las distancias no tienen por qué ser iguales en la ida desde la ciudad i a la j que en la vuelta. Cuando $d_{ij} \neq d_{ji}$ se suele llamar ATSP (Asymmetric TSP).

encontrado. La memoria de la hormiga contiene la lista de ciudades visitadas y se usa para decidir qué ciudades restan aún por visitar mientras la hormiga está construyendo su camino y también para, una vez terminado su camino, depositar las feromonas en los trayectos recorridos.

La tabla de decisión de un hormiga en el nodo *i*, $A_i[a_{ij}(t)]$, se construye mediante la composición de los rastros locales de feromonas con los valores heurísticos:

$$a_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \ \forall j \in N_i \qquad (3)$$

donde $\tau_{ij}(t)$ es la cantidad de feromonas existente en el arco l_{ij} en el instante *t*, η_{ij} es el valor heurístico que contribuye a la selección de un nodo, y que en este caso es la inversa de la distancia: $\eta_{ij} = \frac{1}{d_{ij}}$. Por último, N_i representa el conjunto de nodos vecinos del nodo *i*. Las constantes α y β son valores que ayudan a calibrar el sistema para controlar el peso relativo de las feromonas con respecto al valor heurístico.

La probabilidad de que la hormiga *k* seleccione el trayecto l_{ij} en el proceso de construcción de su camino en la iteración *t*-esima del algoritmo es:

$$P_{ij}^{k}(t) = \frac{a_{ij}(t)}{\sum_{l \in N_i^k} a_{il}(t)} \qquad (4)$$

donde $N_i^k \subseteq N_i$ representa los vecinos del nodo *i* para la hormiga *k* que aún no han sido visitados por ésta y son seleccionados del conjunto de vecinos N_i y posteriormente se comprueba que no se encuentren en la memoria interna de la hormiga (visitados).

La interpretación de las constantes de calibración α y β es la siguiente. Si α=0, las feromonas no influyen y el algoritmo se convierte en el clásico algoritmo voraz que seleccionará como siguiente ciudad a aquella más cercana en ese momento. En este caso sólo el valor heurístico tiene influencia sobre la probabilidad de selección de cada nodo. Como es sabido, los algoritmos voraces no son especialmente eficaces en problemas NP-completos. Así, por ejemplo, un algoritmo voraz cuya heurística fuera escoger la ciudad más cercana cada vez, para las ciudades de la Tabla 4, y empezando y terminando en Madrid, obtendría el camino:

R_{voraz} = {Madrid, Valencia, Murcia, Sevilla, Cádiz, Bilbao, Barcelona, Vigo y Madrid}, con una distancia total d_{voraz}=4613 Km.

En cambio, para este conjunto es fácil hallar a simple vista una ruta más corta: R' = {Madrid, Vigo, Bilbao, Barcelona, Valencia, Murcia, Sevilla, Cádiz, Madrid}, con una distancia total d'=3829 Km.

Por el contrario, cuando β=0 el valor heurístico deja de tener influencia, influyendo únicamente en la decisión las feromonas depositadas previamente, lo que conduciría rápidamente a una situación de bloqueo en el camino que aleatoriamente se hubiera seleccionado primero, lo que generalmente lleva a soluciones no óptimas [39].

Cuando todas las hormigas han completado un camino, siguiendo la meta-heurística ACO, se produce la evaporación de las feromonas. Puesto que se utiliza la deposición retardada de feromonas, la deposición coincide con el proceso de evaporación, depositándose en todos los arcos del grafo la cantidad de feromonas dada por la siguiente expresión:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{5}$$

donde ρ es el denominado *coeficiente de evaporación* y es un valor positivo menor o igual que 1. $\Delta\tau_{ij}(t)$ es la variación total de feromonas correspondiente a la suma de los depósitos de feromonas de todas las hormigas, que responde a la ecuación:

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{6}$$

siendo la cantidad de feromonas depositadas en cada trayecto (arco del nodo) l_{ij} por cada hormiga k, $\Delta\tau_{ij}(t)$, inversamente proporcional a la distancia total del camino encontrado si el lado pertenece al recorrido R^k de la hormiga k en el instante o iteración t. Si el lado l_{ij} no pertenece al recorrido no se depositan feromonas, penalizando así al trayecto que va de la ciudad i a la j, es decir,

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{1}{d^k(t)} & si\ l_{ij} \in R^k(t) \\ 0 & en\ otro\ caso \end{cases} \tag{7}$$

Dorigo estableció a través de la experimentación en su tesis doctoral [15] los valores óptimos siguientes para la calibración del sistema: m = n (número de hormigas igual al número de nodos), α=1, β=5 y $\rho = 0{,}5$. Además, en el proceso de inicialización, se deposita una cantidad de feromona $\tau_0 > 0$, pero muy próxima a cero.

Variantes del algoritmo AS

Una variante del algoritmo AS es el Max-Min AS (MMAS) de Stützle y Hoos [41]. En este algoritmo, el depósito de feromonas no es llevado a cabo por cada hormiga, sino por un agente controlador externo, encargado de realizar acciones globales. A diferencia de AS, el algoritmo MMAS es un algoritmo de optimización *elitista,* puesto que sólo tiene en cuenta el resultado de la hormiga que ha obtenido la mejor solución en cada iteración para realizar el depósito de feromonas, que se corresponde con la siguiente ecuación:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{8}$$

donde $\Delta\tau_{ij}^{best} = \frac{1}{f(s^{best})}$ y $f(s^{best})$ denota el coste de la solución ,que puede corresponderse con el de la mejor iteración (s^{ib}) o con el de todas las iteraciones hasta el momento (s^{gb}). A diferencia de otros algoritmos como ACS (Ant Colony System) que usan s^{gb}, MMAS se decanta por s^{ib}. Esta decisión se debe a que Stützle y Hoos [44] demostraron que el hecho de usar s^{gb}, refuerza en exceso la explotación de dicha solución limitando la exploración de nuevas soluciones, con el consiguiente riesgo de quedar el proceso de optimización atrapado en una solución de pobre calidad. En cambio, eligiendo s^{ib} se minimiza este riesgo gracias a que las soluciones de cada iteración pueden diferir considerablemente entre iteraciones sucesivas, de forma que un gran número de componentes de posibles soluciones puede recibir un refuerzo ocasional en forma de feromonas, potenciando la exploración de nuevos caminos frente a la explotación de los ya encontrados, con un balance entre exploración y explotación más equilibrado. En contraposición, la elección de s^{ib} tiene un impacto negativo en la eficiencia del algoritmo, al requerir numerosas iteraciones para obtener soluciones de calidad. De hecho, tanto para el TSP como para el QAP, los autores proponen una estrategia dinámica mixta, que aumente la frecuencia de uso de s^{gb} conforme aumente el número de iteraciones, obteniendo muy buenos resultados en la comparativa respecto a otros algoritmos.

Para evitar el bloqueo frecuente de los algoritmos elitistas, MMAS impone que sólo los arcos cuyo nivel de feromona esté comprendido en un rango determinado $[\tau_{min}, \tau_{max}]$ sean elegibles. Para garantizar la correcta inicialización, el nivel de feromonas en todos los arcos es inicializado a τ_{max}. Aun así, se producen situaciones de bloqueo que disminuyen la exploración de nuevos trayectos cuando algunos arcos tienen un nivel de feromonas cercano al máximo y otros al mínimo, por lo que sus autores idearon un mecanismo diferente de actualización de feromonas, que denominaron "*suavizado del rastro de feromonas*", y se expresa como sigue:

$$\tau_{ij}^{*}(t) = \tau_{ij}(t) + \delta \cdot \left(\tau_{max}(t) - \tau_{ij}(t)\right) \; con \; 0 < \delta < 1 \qquad (9)$$

El suavizado del rastro consiste en facilitar la exploración mediante el aumento de la probabilidad de seleccionar componentes con bajo nivel de feromonas. En (9), $\tau_{ij}(t)$ y $\tau_{ij}^{*}(t)$ son, respectivamente, la cantidad de feromonas del arco l_{ij} en el instante t antes y después del suavizado. Al ser 0 < δ < 1, el segundo sumando representa una proporción de la distancia relativa del nivel de feromonas actual con respecto al máximo nivel de feromonas definido. Esto contribuye a incrementar de una forma más rápida el nivel de feromonas en aquellos arcos con un nivel bajo de feromonas, pero también a disminuir a aquellos arcos cuyo nivel de feromonas actual es mayor al máximo definido, contribuyendo a que puedan volver al rango deseado. Con el suavizado del rastro de feromonas, el algoritmo MMAS obtuvo resultados significativamente mejores que el AS [41].

Otra variante del AS es el AS_{Rank} [42], que también opta, al igual que el MMAS, por una deposición retardada de feromonas que se delega en una tarea externa. Esta tarea externa se encarga de confeccionar un ranking con las σ-1 mejores hormigas en función de la longitud del recorrido de cada una ella, recibiendo los arcos de cada recorrido un número proporcional de feromonas en función de la posición del ranking que ocupe cada hormiga que lo usó. Además, los arcos del recorrido de la mejor hormiga (aquella que ocupa la primera posición del ranking) reciben una cantidad adicional de feromonas, de forma similar a las hormigas elitistas del algoritmo Ant System. La ecuación que determina la deposición de feromonas en un arco en el algoritmo AS_{Rank} es la siguiente:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \sigma\Delta\tau_{ij}^{+}(t) + \Delta\tau_{ij}^{r}(t) \qquad (10)$$

donde $\Delta\tau_{ij}^{+}(t)$ representa el refuerzo adicional de feromonas para la mejor hormiga y es un factor proporcional a la inversa del camino encontrado en la iteración t, en este caso $1/L^{+}(t)$. Como se observa en la ecuación, en este caso el factor de proporción es una unidad más que el número de hormigas que componen el ranking, esto es σ. Además cada hormiga *r*, recibe al final de cada iteración *t* una cantidad de feromonas dependiente de la longitud de los caminos encontrados por todas las hormigas en esa iteración: $\Delta\tau_{ij}^{r}(t) = \sum_{\mu=1}^{\sigma-1}\Delta\tau_{ij}^{\mu}(t)$, con $\Delta\tau_{ij}^{r}(t) = (\sigma-\mu)/L^{\mu}(t)$ si la hormiga de la posición μ del ranking usó el arco l_{ij} en su recorrido o $\Delta\tau_{ij}^{r}(t) = 0$ si no lo hizo, siendo $L^{\mu}(t)$ la longitud del recorrido de la hormiga que ocupa la posición μ en el instante *t*.

Sistemas de Colonias de Hormigas

Dorigo y Gambardella introdujeron el algoritmo Ant Colony System (ACS) en 1997 [27] para mejorar el rendimiento del AS, que no podía encontrar buenas soluciones en un tiempo razonable cuando el problema era de un tamaño moderadamente grande. Las principales diferencias con respecto a AS son:

- ACS utiliza también deposición retardada de feromonas que es llevada a cabo por una tarea externa, pero en lugar de depositar las feromonas correspondientes a cada hormiga, ACS aplica una estrategia elitista, haciendo que sólo la hormiga que ha obtenido el camino de mejor coste sea la que deposite las feromonas en los arcos que componen su solución. En el caso del algoritmo ACS-3-opt además, esta tarea externa activa previamente un procedimiento de búsqueda local basado en una variante del procedimiento 3-opt [66] para mejorar las soluciones obtenidas antes de proceder al depósito de feromonas. La regla para la deposición de feromonas es:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \rho\Delta\tau_{ij}(t) \qquad (11)$$

  donde $\rho \in (0,1]$ es el parámetro que controla la evaporación de feromonas y $\Delta\tau_{ij}(t) = 1/L^{+}$ es la variación correspondientes al mejor camino, de longitud (o coste) L^{+}. La ecuación sólo se aplica en este caso a los nodos que pertenecen al mejor camino en cada iteración.

- La segunda diferencia radica en el mecanismo de selección probabilística de los arcos. En ACS se emplea una variable aleatoria Q, uniformemente distribuida en [0,1] y $Q_0 \in [0,1]$ un valor configurable. La regla pseudo-aleatoria que se emplea para definir la probabilidad de selección de un arco depende del valor obtenido para Q con respecto a Q_0. Así teniendo en cuenta la tabla de decisión (3) para cada hormiga $a_{ij}(t)$:
  - Si $Q \leq Q_0$:

$$P_{ij}^{k}(t) = \begin{cases} 1 & si\ j\ es\ el\ nodo\ con\ mayor\ a_{ij} \\ 0 & en\ otro\ caso \end{cases} \qquad (12)$$

  - Si $Q > Q_0$ se aplica la ecuación (4) para el cálculo de la probabilidad de selección de cada nodo.

  De esta forma, cuando $Q \leq Q_0$ se explota el conocimiento disponible del problema, es decir, el conocimiento heurístico acerca de las distancias entre los nodos y el conocimiento aprendido, memorizado en forma de rastro de feromonas. En cambio cuando $Q > Q_0$ se aplica la misma exploración sesgada

de AS de la ecuación (4). La calibración de Q_0 permite modular el grado de exploración frente a explotación, permitiendo elegir cuando concentrar la actividad del sistema en la explotación de buenas soluciones (por ejemplo al obtener un valor mínimo deseado del coste).

- La tercera diferencia es la deposición paso a paso de las hormigas en el algoritmo ACS, que se produce cada vez que una hormiga se mueve de un nodo a otro, aplicando la ecuación siguiente:

$$\tau_{ij}(t) = (1 - \varphi)\tau_{ij}(t-1) + \varphi\tau_0 \qquad (13)$$

con $0 < \varphi < 1$ y τ_0 el valor inicial de feromonas de cada arco. En ACS no sólo se produce una deposición retardada teniendo en cuenta el mejor camino de la colonia en esa iteración, sino que cada hormiga también deposita paso a paso sus feromonas. La ecuación (13) permite calibrar la reducción de feromonas de un arco cada vez que una hormiga lo escoge con el fin de poder evitar que todas las hormigas de la colonia sigan el camino de una anterior cuyo camino coincide parcialmente. Por ejemplo, supongamos que la hormiga k_1 ha comenzado su recorrido en el nodo 2, continuando por el 3, luego 4, etc. Si la hormiga k_2 ha comenzado simultáneamente en el nodo 5 y elije el arco l_{52} para llegar al nodo 2, encontrará en el nodo 2 que el arco de salida l_{23} tiene un nivel de feromonas mayor, por lo que tenderá a seguir el camino marcado por las feromonas de la hormiga k_1, con un paso de retraso, evitando la exploración de otros caminos que posiblemente ofrezcan soluciones mejores. Con la utilización de la ecuación (13) para la liberación de feromonas paso a paso por cada hormiga, se disminuye el nivel de feromonas de los arcos visitados, lo que los hace menos atractivos para las hormigas que lleguen a él en un paso posterior, evitando que todas las hormigas de una misma colonia sigan los mismos pasos y fomentando el descubrimiento de caminos alternativos más rápidamente que únicamente con deposición retardada por una tarea externa. Como inconveniente, esta medida hace que, en consecuencia, las hormigas tiendan a no converger a una ruta común.

- Por último, el algoritmo ACS utiliza listas de candidatos con información heurística adicional que proporciona los nodos preferidos para visitar desde un nodo dado. En el algoritmo ACS cuando una hormiga está en un nodo examina la lista de nodos candidatos antes de desplazarse a otro, lo que ahorra las comprobaciones del total de nodos del problema, mejorando el rendimiento del algoritmo. Tan sólo si no hay nodos sin visitar en la lista de candidatos se pasa a buscar otros nodos.

ACS fue el sucesor de Ant-Q [222], un algoritmo que intentaba combinar AS y Q-learning. En realidad, únicamente difieren en la cantidad de feromonas iniciales, para la que Ant-Q definía un valor variable que se correspondía con la máxima concentración de feromonas de un arco perteneciente al recorrido. Al tener un comportamiento similar, fue abandonado dada la mayor simplicidad del algoritmo ACS que evitaba tener que calcular en cada deposición dicho valor (el valor inicial de las feromonas en ACS es una constante que se calcula al inicio y que generalmente depende inversamente del número de arcos del problema).

Según el estudio de Dorigo y Gambardella, ACS es el algoritmo con mejor rendimiento de todos los algoritmos combinatorios descritos anteriormente [27], obteniendo un rendimiento superior a algoritmos genéticos para problemas TSP de pequeño y medio tamaño (50, 75 y 100 ciudades).

Uno de los problemas de ACS es determinar el número adecuado de hormigas que componen la colonia, puesto que a colonias más pobladas, el rendimiento suele ser peor debido a que un mayor número de hormigas se ven influenciadas por el rastro de feromonas dejado por otras, llegando a la misma solución y penalizando el tiempo total del algoritmo. El número adecuado de hormigas depende de las características del problema y la mayoría de las veces ha de calcularse experimentalmente. Debido a la convergencia de los algoritmos ACO cuando el número de iteraciones t es suficientemente grande [76], lo recomendable es seleccionar un número de hormigas m tal que multiplicado por el número de iteraciones que ejecute cada una resulte mayor al número de iteraciones t a partir de la cual los resultados convergen a una misma solución. Puesto que la rapidez de convergencia del algoritmo depende tanto de las características del problema como de la calibración del sistema (constantes α, β y otras), el número adecuado de hormigas sólo puede definirse mediante la experimentación.

2.5.2 El Problema de la Asignación Cuadrática

El problema de la asignación cuadrática, que se denota por sus siglas en inglés QAP, *Quadratic Assignment Problem*, fue planteado por Koopmans y Beckman [67] en 1957. El problema de asignación cuadrática consiste en asignar n elementos a una cantidad n de ubicaciones con un coste asociado al desplazamiento entre diferentes ubicaciones y al asociado al cambio de asignación de un elemento por otro. Matemáticamente puede describirse como sigue.

Dadas dos matrices cuadradas de orden n, A=(a_{ij}) y B=(b_{ij}), encontrar una permutación Π^* que minimice la función:

$$f(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot b_{\pi_i \pi_j} \tag{14}$$

donde *P(n)*, es el conjunto de permutaciones de n elementos, siendo $\pi \in P(n)$ una permutación de las posibles. Se distinguen dos clases de QAP: aleatorios y estructurados, siendo los estructurados aquellos problemas tomados del mundo real. Por ejemplo un típico ejemplo de QAP estructurado es la planificación del reparto de diferentes mercancías por un mismo transportista por carretera, puesto que es necesario elegir la ruta a seguir para así decidir cómo cargar el camión. El coste dependerá de las distancias y entre las ubicaciones, además de un coste adicional por entregar cada elemento (la descarga) en cada ubicación específica. De este modo se buscará que este coste, en función de la distancia y flujo de la mercancía, sea mínimo. En cierto modo, puede considerarse QAP como una generalización del TSP. Maniezzo y Colorni, aplicaron el algoritmo AS a este problema usando la heurística Min-Max, denominándolo AS-QAP [43].

La única diferencia con el algoritmo AS del AS-QAP es la función objetivo que determina la cantidad de feromonas a depositar en los arcos del mejor recorrido en cada iteración. Los resultados obtenidos fueron de la misma calidad a los obtenidos con otras aproximaciones como programación evolutiva o algoritmos genéticos.

Similares resultados obtuvieron Stützle y Hoos con la aplicación directa del algoritmo MMAS al QAP [44]. Por último Gambardella, Taillard y Dorigo implementan un algoritmo híbrido entre sistema de hormigas y búsqueda local, denominado HAS-QAP [45]. En el algoritmo HAS-QAP, durante cada iteración, existe el problema de la elección de la solución de partida asociada a cada hormiga, para lo que se definen dos mecanismos interesantes: *intensificación* y *diversificación*. La intensificación se usa para explorar los vecinos de buenas soluciones, haciendo que la hormiga regrese hacia la solución que tenía al principio de la iteración si la solución era mejor que la solución que ha encontrado al final de la iteración. La diversificación implementa un reinicio parcial del algoritmo cuando las soluciones parecen no poder mejorarse nunca más, y consiste en la reiniciación tanto de la matriz de feromonas como de las soluciones asociadas a cada hormiga. Lo más interesante del algoritmo HAS-QAP es el mecanismo de búsqueda local que aplica para mejorar la solución. Este procedimiento de búsqueda local, examina sistemáticamente todos los posibles intercambios de permutaciones de los elementos, procediendo al intercambio cuando

encuentra alguna que mejora la solución actual. Se ha probado que el algoritmo HAS-QAP se comporta extraordinariamente bien con QAP estructurados.

2.5.3 El Problema de la Planificación de la Producción

El problema de la planificación de la producción, más conocido por sus siglas en inglés JSP, *Job-shop Scheduling Problem*, es un problema de optimización en el que una serie de trabajos deben ser asignados a unos determinados recursos en un tiempo determinado. La versión más simple puede describirse de la siguiente forma:

Dado un conjunto *J* de trabajos, un conjunto *M* de máquinas y un conjunto *O* de operaciones con *N* integrantes. Para cada operación $i \in O$ tenemos relacionado un trabajo $j \in J$ y una máquina $m \in M$ en la que debe realizarse, consumiendo un tiempo $t \in \mathbb{R}^+$. Además, es dada una relación de precedencia binaria $\prec$ que descompone *O* en cadenas, una para cada trabajo. Encontrar un tiempo de comienzo s_i para cada operación tal que se minimice el máximo tiempo de finalización de todas las operaciones sin que se procesen dos trabajos simultáneamente en la misma máquina. El problema tiene las siguientes restricciones:

a) Trabajos en tiempo futuro: $s_i \geq 0, \forall i \in O$

b) Precedencia de trabajos: $s_j \geq s_i + t_i\ , \forall i, j \in O\ \wedge i \prec j$

c) Máquinas dedicadas: $s_j \geq s_i + t_i\ \vee s_i \geq s_j + t_j\ , \forall i, j \in O\ \wedge m_i = m_j$

Colorni et al. [46] aplicaron el algoritmo AS directamente a este problema, con el único cambio del valor heurístico η que fue calculado usando la heurística LRT (*Longest Remaining Time*) para elegir el trabajo que necesite el mayor tiempo de proceso de entre los trabajos restantes por planificar. El algoritmo resultante, AS-JSP, fue probado con problemas de hasta 15 máquinas y 15 trabajos, encontrando soluciones óptimas aunque no excepcionales (dentro de un margen del 10% con la mejor solución), lo que sugiere que aún hay bastante margen de mejora en este problema.

2.5.4 El Problema del Enrutamiento de Vehículos

El problema del enrutamiento de vehículos, VRP (*Vehicle Routing Problem*) es en realidad un conjunto de variantes del mismo problema. En general, en todos ellos se trata de averiguar la mejor ruta de una flota de transporte para dar servicio a una serie de clientes. El problema se estudió por primera vez en la distribución de gasolina para estaciones de carburante [68].

La función objetivo depende de la tipología y características del problema. Lo más habitual es intentar: minimizar el coste total de operación, minimizar el tiempo total de transporte, minimizar la distancia total recorrida, minimizar el tiempo de espera, maximizar el beneficio, maximizar el servicio al cliente, minimizar la utilización de vehículos, equilibrar la utilización de los recursos, etc. Los principales elementos que constituyen este conjunto de problemas son:

- La red de transporte
- La flota de vehículos
- Los clientes y/o proveedores
- El depósito central (o depósitos)
- Los servicios a atender
- Las rutas que componen la solución

El modelado del problema depende de las restricciones impuestas (uno o varios proveedores, uno o varios almacenes, máxima distancia de una ruta, máximo tiempo de reparto para cada vehículo, etc.)

Bullnheimer et al. [47] modelan el problema VRP mediante un grafo dirigido ponderado completo, en el que $N=\{n_0,n_1...n_m\}$ es el conjunto de nodos del grafo y $A=\{(i,j): i\neq j\}$ es el conjunto de arcos que representan las vías de comunicación que unen dos nodos entre sí, teniendo cada uno un peso d_{ij} que representa la distancia del nodo n_i al n_j.. El problema modelado en [47] sólo cuenta con un depósito central, representado por el nodo n_0, en el que inicialmente está ubicada la flota de vehículos $M=\{m_0, m_1... m_l\}$ del problema, cada uno de ellos con una capacidad D. El resto de nodos son nodos de clientes. Cada nodo n_i tiene asociada una demanda $d_i\geq 0$ y un tiempo de servicio $\delta_i\geq 0$ siendo obviamente la demanda y tiempo de servicio del almacén (n_0) cero.

El objetivo del problema es encontrar las rutas de menor coste para cada cliente tales que:

a) Cada cliente sea visitado exactamente una vez por un único vehículo.

b) Para cada vehículo la demanda total no supere a su capacidad, D.

c) La longitud total del recorrido no exceda a un máximo establecido, L.

d) Cada vehículo comienza y termina su recorrido en el almacén.

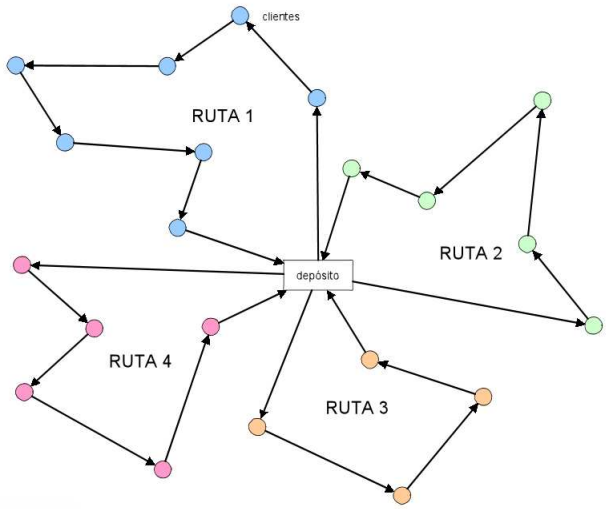


Figura 5: Representación de una variante del VRP con depósito único central

El algoritmo AS-VRP [47] es una ampliación del AS_{rank} en el que se tienen en cuenta las restricciones anteriores en el proceso de construcción de la tabla de memoria (nodos visitados) de cada hormiga. Los autores añadieron un proceso de optimización simple, basado en la heurística 2-opt. La comparación con otros métodos resultó bastante positiva, mejorando los resultados obtenidos con redes neuronales, por ejemplo.

El algoritmo HAS-VRP [48] se basa en ACS: cada hormiga construye un recorrido completo sin violar las restricciones de capacidad de los vehículos. Un recorrido completo se compone de muchos sub-recorridos que conectan almacenes, y cada sub-recorrido se corresponde con el recorrido asociado a uno de los vehículos. La actualización del rastro de feromonas es retardada, como en ACS. Además, HAS-VRP incorpora un procedimiento de intercambio de arcos que es aplicado al final de cada iteración por una tarea externa. Los resultados obtenidos con esta aproximación son competitivos incluso con los mejores algoritmos conocidos llevando a establecer nuevos límites para algunos ejemplos muy estudiados.

Los mismos autores estudiaron también el VRP con ventanas temporales, VRPTW, en el algoritmo MACS-VRPTW [69]. El VRPTW introduce un intervalo de tiempo para cada cliente, durante el cual ha de ser servido. Los vehículos que lleguen antes del inicio de la ventana de tiempo tendrán que esperar. Esta suele ser una típica restricción en los repartos a domicilio de muchas grandes superficies, que solicitan al

cliente las franjas horarias en las que podrían entregarle la mercancía. En la literatura siempre se considera al VRPTW como un problema multi-objetivo, puesto que interesan soluciones de bajo coste pero que garanticen el reparto en las ventanas de tiempo establecidas, prefiriéndose soluciones con un menor número de vehículos en uso (mayores trayectos) que otras con un mayor número de vehículos y trayectos más cortos.

Para optimizar las dos funciones simultáneamente, se emplea ACS con dos colonias de hormigas. La primera trata de minimizar el número de vehículos mientras que las segunda usa el número de vehículos obtenido por la primera para minimizar con ellos el tiempo del recorrido. Las dos colonias utilizan rastros diferentes de feromonas, pero la mejor hormiga puede actualizar (modificar) el rastro de feromonas de la otra colonia. Este es el primer claro ejemplo de empleo de ACS en problemas con múltiples objetivos.

2.5.5 El Problema del Ordenamiento Secuencial

L. Escudero denominó a la variante asíncrona y con relaciones del TSP *Problema del Ordenamiento Secuencial* [70], SOP (del inglés, *Sequential Ordering Problem*). El problema consiste en encontrar el camino hamiltoniano de menor coste en un grafo dirigido con arcos y nodos ponderados, respetando las restricciones de precedencia entre los nodos.

Partiendo del grafo *G=(V,A)* que modela el TSP y teniendo en cuenta la posibilidad de caminos asimétricos, el problema del ordenamiento secuencial se modela añadiendo un segundo grafo dirigido P = (N, Π) con $N \subseteq V$, que define las relaciones de precedencia, de tal forma que un camino hamiltoniano H factible debe satisfacer las relaciones de precedencia dadas por el conjunto de arcos Π, de forma que si $(i, j) \in \Pi$, entonces $i \prec j$ en todo camino hamiltoniano factible en G.

Tras múltiples aproximaciones [71], [72], casi todas basadas en extensiones de las soluciones publicadas para el TSP, la aproximación de Gambardella y Dorigo, HAS-SOP [49] obtuvo resultados excelentes[7]. HAS-SOP está basado en una primera

[7] Se realizaron pruebas con todos los problemas disponibles en TSPLIB. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html.

aproximación a la que denominaron ACS-SOP, basada en ACS. HAS-SOP tan sólo difiere de su predecesor ACS-SOP en el cálculo del conjunto de nodos posibles para el que se tienen en cuenta en cada iteración las reglas de precedencia; y en el proceso de optimización local, *SOP-3-exchange*, basado en el algoritmo de optimización *3-opt* consistente en el intercambio de tres arcos de un recorrido por otros tres y estudiar si ha habido una mejora.

2.5.6 El Problema de Coloreado de Grafos

El problema de coloreado de grafos, GCP (*Graph Coloring Problem*), consiste en minimizar el número de colores necesarios para colorear un grafo teniendo en cuenta que un nodo no puede tener el mismo color que ningún otro nodo vecino. El algoritmo ANTCOL [31] utiliza heurísticas bien conocidas para el coloreado de grafos como RLF (*Recursive Large First*) [73] y DSATUR[8][74]. Lo más novedoso del algoritmo propuesto es la utilización de dos rastros de feromonas para su uso en dos selecciones: primero la del siguiente nodo del recorrido y después, la elección del color a asignar a este nodo. Estas dos elecciones se llevan a cabo mediante dos reglas probabilísticas en función de dos rastros de feromonas distintos y dos heurísticas apropiadas. Los tests realizados revelaron unos resultados comparables a las mejores heurísticas conocidas y animaron a investigar en la utilidad de dos o más rastros de feromonas diferentes en los algoritmos ACO.

2.5.7 El Problema de la Supersecuencia Común más Corta

El problema de encontrar la supersecuencia común más corta, SCSP (*Shortest Common Supersequence Problem*) es un problema con grandes aplicaciones en Bioinformática. Consiste en encontrar la cadena de menor longitud que contenga a todas las cadenas dadas de un lenguaje, formadas de un mismo alfabeto. Es decir, dadas dos secuencias $X=<x_1,...,x_m>$ e $Y=<y_1,...,y_n>$, una secuencia $U=<u_1,...,u_k>$ es una supersecuencia común de X e Y si U es una supersecuencia de ambas X e Y. El objetivo en este problema es averiguar la supersecuencia de menor longitud dadas una serie de cadenas de entrada. Para el ejemplo con dos secuencias de entrada, *X=abcbdab* e *Y=bdcaba,* la supersecuencia común más corta es U=*abdcabdab*. Parece sencillo, pero incluso para un alfabeto de dos elementos, cuando el número de cadenas de entrada se incrementa y la longitud de dichas cadenas de entrada crece el problema se convierte en un problema combinatorio NP-completo.

[8] Algoritmo de coloreado de grafos basado en el grado de saturación de un vértice.

ACS-SCS, la solución propuesta por Michel y Middendorf [50], presenta como novedad la incorporación de una función de búsqueda adelantada que tiene en cuenta la influencia de la elección del siguiente símbolo a añadir en la siguiente iteración. El valor devuelto por esta función hace las veces de valor heurístico en la ecuación de la probabilidad de decisión (4). Los autores mejoraron su algoritmo notablemente mediante el uso de un modelo isla de computación, esto es, diferentes colonias de hormigas trabajaban en el mismo problema concurrentemente usando cada colonia su propio rastro de feromonas y pasado un número determinado de iteraciones, intercambiando entre sí la mejor solución encontrada.

El algoritmo AS-SCS-LM, que incorpora la heurística LM (*Longest Matching*) [75] fue comparado en [50] con un algoritmo genético contemporáneo especialmente diseñado para el SCSP, resultando que el AS-SCS-LM proporciona un mejor rendimiento en la mayoría de los tests.

2.6 Aplicación a problemas dinámicos de optimización

La principal contribución de los algoritmos ACO a problemas dinámicos ha sido su aplicación en problemas de enrutamiento de información en redes informáticas, puesto que en las redes de datos, con un amplio número de dispositivos de red (nodos) intermedios y una topología compleja de interconexión, toma especial importancia conocer la congestión y disponibilidad de cada segmento de red, para encaminar la información por el camino de menor coste. De forma similar a como las hormigas viajan del nido a la fuente de alimento, los paquetes de datos viajan del emisor al receptor y deben hacerlo por el camino de menor coste posible.

La utilización de algoritmos inspirados en el comportamiento de las hormigas para el enrutamiento es diferente tanto a las soluciones basadas en algoritmos por vector-distancia como a las basadas en el estado de los enlaces, puesto que la información no proviene ni de los nodos vecinos (vector distancia) ni de mensajes periódicos del resto de nodos (estado de enlaces), aunque sí necesitan conocer el grafo de la red, lo que no siempre es fácil.

El hecho de que las hormigas construyan sus caminos de una manera probabilística permite la exploración de rutas múltiples, lo que facilita la adaptación del

algoritmo a los cambios de la red. Otra característica que lo hace interesante es la posibilidad del reenvío aleatorio de los paquetes de datos con la información de las feromonas, que permite gestionar múltiples rutas. Si las feromonas se mantienen actualizadas y son función de la carga (congestión) de cada enlace, la carga se balanceará siguiendo automáticamente los cambios en la red.

Existen dos variables del problema de encaminamiento de mensajes: *orientado a la conexión*, en el que primero se haya el camino idóneo para que todos los paquetes de datos que componen el mensaje lo utilicen, y *no orientado a la conexión* en el que cada paquete de datos sigue el camino de menor coste en cada momento para adaptarse a condiciones cambiantes del entorno de red.

2.6.1 Aplicación al encaminamiento orientado a la conexión

El primer algoritmo ACO para el problema de encaminamiento orientado a la conexión, *Ant Based Control* (ABC) [30], se aplicó a un modelo de red de comunicaciones de British Telecom. La red se modeló como un grafo *G=(N, A)* en el que cada nodo representa un *switch* de capacidad limitada y en el que cada arco representa los enlaces, de capacidad ilimitada, entre *switches*. Cada nodo *i* se caracteriza por su capacidad total, C_i, y su capacidad libre S_i. C_i representa el número máximo de conexiones que el nodo *i* puede establecer, y S_i representa el porcentaje de capacidad que aún queda libre para nuevas conexiones. Cada arco *(i,j)* tiene asociado un rastro de feromonas $\tau_{i,j,d}$ para cada nodo *d* de la red que representa lo deseable de escoger dicho arco cuando el nodo destino es *d* (con $d \neq i$). El nivel de feromonas dado por la función $\tau_{i,j,d}(t)$ representa la probabilidad de que una hormiga que desea llegar al nodo *d*, sea encaminada en la iteración *t* desde el nodo *i* al nodo *j*.

A diferencia de algoritmos estáticos, no se establece ninguna heurística local (como por ejemplo en el TSP se establece como valor heurístico el valor inverso de la distancia entre dos nodos). En este problema, los valores de las tablas de decisión están basados únicamente en el valor de feromonas depositado en cada arco: $a_{ijd}(t) = \tau_{ijd}(t)$. Se añade un mecanismo de exploración, para evitar el bloqueo que puedan producir las feromonas, consistente en proporcionar a cada hormiga la posibilidad de elegir aleatoriamente el siguiente nodo vecino. Esta posibilidad se ofrece con una baja probabilidad. Además las hormigas no tienen memoria y no almacenan los nodos visitados.

Las tablas de encaminamiento se obtienen usando las tablas de decisión de las hormigas de forma determinista: durante la configuración del camino, al inicio, la ruta desde el nodo *s* hasta el *d*, se construye eligiendo secuencialmente el vecino con mayor probabilidad. Una vez que se ha establecido el camino (la conexión), la capacidad libre S_i de cada nodo de camino se disminuye una cantidad fija. Si durante la configuración del camino alguno de los nodos seleccionados no dispone de la capacidad libre suficiente, la conexión es rechazada por congestión de la red. Por otro lado, cuando la conexión ha terminado (todos los paquetes han sido transmitidos o la llamada, en el caso de llamadas telefónicas, ha concluido) la correspondiente capacidad que se ha reservado anteriormente para cada nodo (disminuyendo su capacidad libre) es devuelta de nuevo a cada nodo. Puesto que las condiciones de la red son variables, las hormigas son lanzadas periódicamente para recalcular los mejores caminos cada cierto tiempo, depositando las hormigas sus feromonas paso a paso, en cada arco que visitan.

El funcionamiento es el siguiente: una hormiga *k* que ha comenzado la construcción de su camino en el nodo *s* y ha llegado en el instante *t* al nodo *j* desde el nodo *i*, añade un incremento de feromonas $\Delta\tau^k(t)$ a la cantidad de feromonas existente $\tau_{jis}(t)$ en el arco l_{ji}. El nuevo valor de feromonas actualizado para el arco l_{ji} refleja lo deseable que es ir desde el nodo *j* al nodo s pasando por el nodo *i*. Es decir, las hormigas tratan de recorrer el camino inverso para obtener información que permita recomendar el mejor elemento de red para llegar a un nodo destino (el de partida de la hormiga). Una vez actualizado el nivel de feromonas se produce la evaporación para el resto de entradas que tienen al nodo s como destino. En este caso el factor de evaporación dado por los autores es $1/(1 + \Delta\tau^k(t))$ de forma que los valores de feromonas pueden seguir siendo usados como probabilidades:

$$\tau_{ins}(t) \leftarrow \frac{\tau_{ins}(t)}{1 + \Delta\tau^k(t)}, \forall n \in N_i \qquad (15)$$

Obviamente en este problema se supone que el camino de ida entre dos nodos tiene el mismo coste en términos de rendimiento, tiempo, etc. que el camino de ida.

El valor de feromonas a depositar por cada hormiga $\Delta\tau^k(t)$ se hace dependiente de la capacidad libre de cada nodo en el momento de ser recorrido por la hormiga, de forma que dicho valor es una función de la edad de la hormiga. Las hormigas son más viejas conforme más nodos recorren (cada uno un instante más) y algunas pueden quedar retrasadas en nodos que no tienen capacidad libre disponible, de forma que la

cantidad de feromonas depositada por cada hormiga es inversamente proporcional al grado de congestión en el camino recorrido.

En el algoritmo ABC no se tienen en cuenta tareas externas a realizar al final de cada iteración. Cada intento de conexión (o llamada) es aceptado o rechazado según el proceso de configuración que escoge el mejor camino de forma determinista según las tablas de encaminamiento. Si dicho camino tiene la capacidad suficiente se establece la conexión rechazándose en caso contrario.

El algoritmo ABC fue probado en una red isomorfa de una red real de telefonía de British Telecom (BT), obteniéndose un ratio de llamadas aceptadas frente a llamadas rechazadas mejor que los obtenidos por los investigadores de BT con otro algoritmo competidor.

Otro algoritmo muy interesante es ASGA (*Ant System plus Genetic Algorithm*) [51] que añade al AS un algoritmo genético para modificar en cada iteración los valores de las constantes de calibración α y β que controlan el carácter explorador o explotador de soluciones del algoritmo. En ASGA se utilizan las mismas tablas de decisión descritas para AS, siendo el valor heurístico localmente estimado en función del coste de cada arco, es decir, se escoge el nodo que está unido por el arco de menor coste. Al principio de cada iteración de una hormiga k se inicializa el coste del recorrido a cero, y se irá incrementando con el coste de cada arco seleccionado hasta el nodo destino, de forma que al final se tiene el coste total del recorrido. Una vez llegado al nodo destino, la hormiga emprende el camino de vuelta siguiendo el mismo camino por el que llegó, depositando en cada arco una cantidad de feromonas proporcional al coste total del camino en esa iteración. Cuando todas las hormigas han vuelto una tarea externa selecciona la ruta que fue seguida por un porcentaje mayor de hormigas. Durante la conexión un algoritmo genético envía y coordina nuevas hormigas, para cambiar la ruta en caso de fallo de conexión o congestión de la red, modificando los parámetros de α y β para intensificar la exploración de nuevos caminos o la explotación de los obtenidos con nuevas variantes.

2.6.2 Aplicación al encaminamiento no orientado a la conexión

Las aplicaciones de ACO al problema del encaminamiento no orientado a la conexión están basadas en su mayoría en el algoritmo AS y a partir de éste en el ya comentado ABC, a excepción de las distintas variantes del algoritmo AntNet [52].

Las principales diferencias de AntNet son el uso de modelos estadísticos locales que tienen en cuenta el tiempo empleado por las hormigas en recorrer un determinado camino, la deposición retardada de feromonas (una vez construido el recorrido) frente a la deposición paso a paso empleada en el ABC y la aplicación de heurísticas que tienen en cuenta el estado actual del tráfico para determinar la bondad de un camino respecto a otro. En AntNet el tiempo empleado por una hormiga en la construcción de un camino es una medida del retardo de la red en ese camino, pero es necesaria la evaluación de los caminos en relación con el estado actual de la red puesto que un tiempo *T* no es de por sí alto o bajo si no se tiene en cuenta el estado de congestión de la red. Esto es, el mismo tiempo *T* puede ser de baja calidad (excesivamente alto) en condiciones de baja congestión pero podría ser muy bueno si se da en condiciones de tráfico intenso. La cantidad de feromonas depositadas es proporcional a la bondad del camino construido.

En AntNet la tabla de decisión $A_i = [a_{ind}(t)]_{|N_i|,|N|-1}$ del nodo *i* se construye como composición de los rastros de feromonas con los valores heurísticos de la forma siguiente:

$$a_{ind}(t) = \frac{\omega\tau_{ind}(t) + (1-\omega)\eta_n}{\omega + (1-\omega)(|N_i| - 1)} \tag{16}$$

donde N_i es el conjunto de nodos vecinos del nodo *i*, *n* es un nodo perteneciente a dicho conjunto, *d* es el nodo destino, η_n es el valor heurístico normalizado al rango [0,1] y ω es una variable de calibración también en el rango [0,1]. En los elementos de la tabla de decisión (16), el denominador es un término de normalización.

En AntNet las hormigas son lanzadas de cada nodo de la red en busca de recorridos hacia los nodos destino, por lo que al menos son necesarias *N·(N-1)* hormigas. El algoritmo AntNet diferencia entre hormigas hacia adelante (*forward ants*), que van desde el nodo fuente al destino y hormigas hacia atrás (*backward ants*) que utilizan los caminos hallados por las primeras para volver al nido y son las encargadas del depósito de feromonas en cada arco (una vez calculado el coste del camino hallado por las *forward ants*). Todas las feromonas en un recorrido se actualizan respecto a todos los nodos sucesores y no sólo con respecto al nodo origen, tal y como aplicaron Bonebeau et al. en [53]. Además, AntNet elimina los ciclos que pueden producirse en el recorrido de las hormigas evitando que desde un nodo se vuelva a uno ya visitado.

Aunque AntNet demostró ser muy eficiente, sus autores introdujeron nuevas variantes como AntNet-FA [54] en el que introducen el concepto de "*hormigas*

voladoras" con una variación de las forward ants con la capacidad de movimiento hacia el siguiente nodo mucho más rápida (se descarga de tareas a la hormigas hacia adelante, por ejemplo evitándoles tener que almacenar el tiempo empleado en cada cambio de nodo). AntNet-FA es una versión mejorada de AntNet, en el que las hormigas se mueven más rápidamente sobre las colas con mayor prioridad. El rendimiento de AntNet-FA mejora con el incremento del tamaño de la red y su eficiencia es mejor que la de AntNet.

2.7 Aplicación a la resolución de problemas multi-objetivos

2.7.1 Introducción a problemas multi-objetivos

La optimización multi-objetivo puede entenderse como el problema de encontrar un vector de variables de decisión que satisfacen restricciones y optimizan un vector cuyos elementos representan las funciones objetivo. Generalmente, en la vida real, muchos de los problemas presentados en las secciones anteriores tienen una variante multi-objetivo que es la que más utilidad (o valor) ofrece para los interesados. Ejemplos de problemas multi-objetivo habituales son la compra de un automóvil (minimizar precio, maximizar potencia, confort, financiación, etc.), la ubicación de torres de repetición para telefonía móvil (minimizar costes de adquisición del suelo, de materiales, maximizar la cobertura…) o el problema del viajante en la que se pretenda no sólo minimizar la distancia (para ahorrar gastos de combustible), sino también el tiempo empleado en el recorrido por el viajante, y en muchas ocasiones, el camino más corto no siempre es el más rápido.

Los problemas multi-objetivo pueden tener más de una solución posible o vector $\vec{x} = [x_1, x_2 \dots x_n]^T$ que satisface un conjunto de restricciones sobre los valores de este vector y optimiza el vector de funciones:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}) \dots f_k(\vec{x})]^T \in \mathbb{R}^k. \quad (17)$$

El conjunto de todas las soluciones que satisfacen las restricciones se denomina conjunto de soluciones factibles y se representa como Ω, con $\Omega \subset \mathbb{R}^n$. Su imagen Ω_0 es:

$$\Omega_0 = \vec{f}(\vec{x}) \in \mathbb{R}^k \vee \vec{x} \in \Omega \quad (18)$$

En la optimización de un solo objetivo el conjunto de variables de decisión factibles está ordenado mediante una función objetivo *f,* siendo fácil discernir si para dos soluciones *a* y *b*, se cumple que *f(a) > f(b) o f(a) < f(b) o f(a) = f(b)*. En cambio en problemas con múltiples objetivos, el orden que se da suele ser parcial y no puede considerarse siempre que *f(a)* sea mejor que *f*(*b*) o al contrario. Estos casos son comunes en problemas en los que mejorar un objetivo suele empeorar otro, por ejemplo minimizar el coste de producción y maximizar la calidad en la fabricación de un producto. Para ilustrar este concepto, se presenta en la Figura 6 la relación entre inversión en recursos (humanos y materiales) y el número de incidencias que superan el acuerdo de nivel de servicio (o SLA del inglés *Service Level Agreement*) en una empresa de servicios. Se quieren minimizar ambos.

Los puntos de la curva de la Figura 6 representan soluciones Pareto-óptimas. Ninguna de ellas se puede definir como "mejor" que las demás, a menos que se incluya alguna otra información (restricciones, ponderaciones, etc.) que determine cuál de los objetivos es más importante.

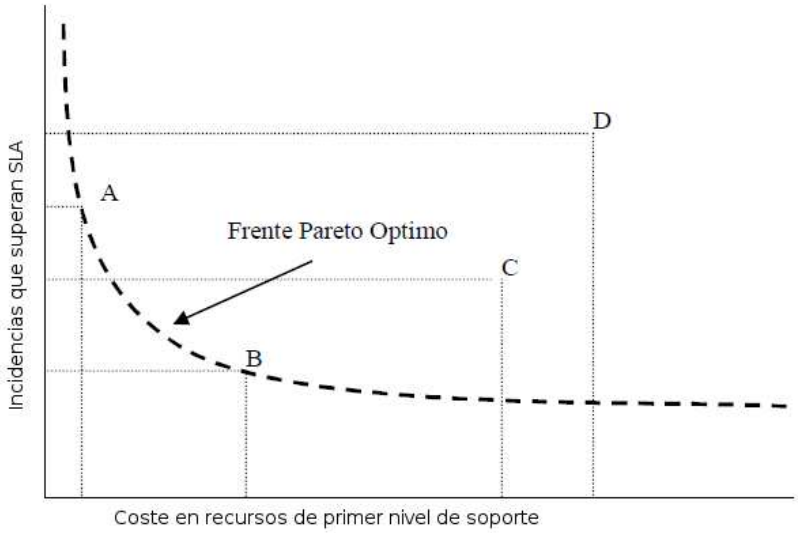


Figura 6: Incidencias que superan SLA frente al coste en técnicos de soporte

La solución representada por el punto *B* es mejor que la representada por el punto *C*, puesto que optimiza los dos objetivos. Sin embargo en la comparación entre *C* y *A*, se obtiene que, disminuyendo mucho los costes de los recursos, el número de incidencias en *A* es ligeramente mayor al de *C* pero no podemos decir que una solución sea mejor que otra puesto que no son comparables entre ellos si consideramos todos los

objetivos. Tampoco podemos afirmar al comparar *A* con *B* que alguna de las dos sea mejor si se considera que ambos objetivos son igualmente importantes y no se introduce alguna restricción adicional. Sin embargo, *B* es claramente superior a *C* en ambos objetivos. ¿Cuándo podemos decir que un vector de soluciones es igual a otro, mayor o menor? Para dar respuesta a esta pregunta se utiliza ampliamente en economía y en ingeniería el concepto de eficiencia de Pareto; lo cual se define como sigue.

Según Pareto, una situación *A* es superior o preferible a una situación *B* cuando el paso de *B* a *A* supone una mejora para todos los miembros de la sociedad, o bien una mejora para algunos, sin que los demás resulten perjudicados. Aplicando este concepto, dados dos vectores $u \in X$, v $\in Y$ se tiene que:

$$f(u) = f(v)\ si\ y\ sólo\ si\ \forall i \in \{1,2 \dots k\}: f_i(u) = f_i(v) \tag{19}$$

$$f(u) \geq f(v)\ si\ y\ sólo\ si\ \forall i \in \{1,2 \dots k\}: f_i(u) \geq f_i(v) \tag{20}$$

$$f(u) > f(v)\ si\ y\ sólo\ si\ f(u) \geq f(v)\ y\ f(u) \neq f(v) \tag{21}$$

$$f(u) \leq f(v)\ si\ y\ sólo\ si\ \forall i \in \{1,2 \dots k\}: f_i(u) \leq f_i(v) \tag{22}$$

$$f(u) < f(v)\ si\ y\ sólo\ si\ f(u) \leq f(v)\ y\ f(u) \neq f(v) \tag{23}$$

A partir de estas relaciones se define el concepto de *dominancia de Pareto*:

Se dice que, en un contexto de minimización (como el del ejemplo de la Figura 6) para dos soluciones $u, v \in \Omega$:

- u domina a v (y se denota como $u \prec v$) si y sólo si *u* es menor o igual que v en cada uno de los objetivos y estrictamente menor en al menos un objetivo:

$$f_i(u) \leq f_i(v) \forall i \in [1, 2 \dots b] \ \wedge\ \exists j \in [1, 2 \dots k] \mid f_j(u) < f_j(v) \tag{24}$$

- *u* y *v* no son comparables si y sólo si $u \not\prec v \ \wedge v \not\prec u$. Es decir si ninguna de las dos domina a la otra. Se denota como: $u \sim v$

En un contexto de maximización, el concepto de dominancia se define de forma similar a los dos puntos anteriores pero cambiando la relación < por > y la relación ≤ por ≥ en (24). En un contexto de maximización, la dominancia de *u* sobre *v* se denota como $u \succ v$.

A partir del concepto de dominancia se define el concepto de *optimalidad de Pareto*:

Dado un vector de decisión $\vec{x} \in X_f$, y su correspondiente vector objetivo $y = f(\vec{x}) \in Y_f$, se dice que *x* es *no dominado* respecto a un conjunto $A \subseteq X_f$ si y sólo si $\forall a \in A, x \prec a \ \vee x \sim a$.

En caso de que *x* sea *no dominado* respecto de todo el conjunto X_f se dice que *x* es una solución Pareto óptima (también llamado óptimo paretiano) y se representa como *x**, formando parte su correspondiente vector objetivo *y* del frente Pareto óptimo, denotado por Y_{true}. De esta forma, al conjunto de vectores de decisión no dominados con respecto a todo X_f, X_{true}, se le denomina *Conjunto de Pareto* y se representan como *x** mientras que el conjunto correspondiente de vectores objetivo $Y_{true} = f(X_{true})$ constituye el *Frente de Pareto*.

En otras palabras, la solución *x** es un óptimo de Pareto si no existe un vector que haga mejorar alguno de los objetivos -respecto a los valores obtenidos para *x**- sin que empeore de forma simultánea alguno de los otros. En general, la solución en el sentido de Pareto al problema de optimización multi-objetivo no será única, sino que estará formada por el conjunto de todos los vectores no dominados del Frente de Pareto.

2.7.2 Adaptación de ACO a problemas multi-objetivos

Pinto y Barán [62] proponen, como adaptación del ACS a problemas con *r* objetivos (con *r>1*), tener en cuenta en la ecuación de actualización retardada de feromonas (5) la suma de las variaciones de las mismas para cada objetivo, de forma que la ecuación (7) aplicada a un problema con *r* objetivos se expresaría como:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{1}{\sum_{n=1}^{r} d_n^k(t)} & si\ l_{ij} \in R^k(t) \\ 0 & en\ otro\ caso \end{cases} \qquad (25)$$

En la ecuación (25) $d_n^k(t)$ representa la distancia del recorrido obtenido por la hormiga *k* en el instante *t*, según la métrica de distancia dada para el objetivo *n*. Por motivos de normalización, los valores de dicha función de distancia son divididos por un valor máximo definido a priori y que normalmente, es el valor máximo posible para cada objetivo.

La hormiga que completó una solución debe actualizar el conjunto de Pareto (*CP)* si la solución encontrada es *no dominada* con respecto a las existentes en *CP* y luego debe eliminar las soluciones dominadas por la misma. Este comportamiento modifica la meta-heurística ACO para adaptarla a problemas multi-objetivos en lo que algunos autores [58], [61][62][64] han denominado MOACO (*Multio-Objective ACO*) aunque, como muestra el Listado 2, no es sino una particularización de la meta-heurística ACO (Listado 1), en la que la función de gestión de las hormigas debe hacerse cargo de esta casuística.

```
funcion gestionar_hormigas()
 para ant=1 to m // m = tamaño de la colonia
  solucion = {∅}
  mientras hay_estados_no_visitados()
   siguiente=seleccionar_siguiente_estado()
   solucion = solucion ∪ {siguiente}
   marcar_como_visitado(siguiente)
   si(actualizacion_paso_a_paso)
    actualizar_feromonas_paso_a_paso() // según (13)
   fsi
  fmientras
  evaluar_solucion(solucion)
  actualizar_conjunto_pareto()
 fpara
ffuncion
```

Listado 2: Adaptación de la gestión de las hormigas a problemas multi-objetivos

Las aproximaciones de adaptación a problemas multi-objetivo son variadas, basándose en alguno de los diferentes algoritmos de un sólo objetivo expuestos con anterioridad. Para que realmente sea un problema multi-objetivo, todos han de contemplar las diferentes visibilidades independientemente, puesto que el definir una visibilidad en función de alguna función que tenga en cuenta los diferentes objetivos (ponderados en función de su relevancia) convierte al problema en un problema de un único objetivo. Todos los algoritmos que se presentan a continuación consideran una visibilidad para cada objetivo pero difieren en el número de matrices de feromonas o de colonias de hormigas usadas.

Algoritmos con múltiples colonias y una matriz de feromonas

El algoritmo Multi-objective Ant Q (MOAQ) [58] es una adaptación en el algoritmo Ant-Q en la que utiliza una colonia por cada objetivo. El algoritmo gestiona

una única matriz de feromonas, actualizando cada colonia la misma matriz de feromonas en función de su objetivo particular.

Algoritmos con una colonia y varias matrices de feromonas

El algoritmo Bicreterion Ant (BiAnt) [59] se diseñó para dar solución a problemas con dos objetivos. Hace uso de una única colonia pero mantiene dos matrices de feromonas, τ y τ', una para cada objetivo. La visibilidad depende de cada objetivo, por lo que en el caso del BiAnt se contemplan dos visibilidades η y η'. La ecuación de la probabilidad de selección del siguiente nodo se complica:

$$P_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\lambda\alpha}\ \tau'^{(1-\lambda)\alpha}_{ij}\ \eta_{ij}^{\lambda\beta}\ \eta'^{(1-\lambda)\beta}_{ij}}{\sum_{x\in N_i}\tau_{ix}^{\lambda\alpha}\ \tau'^{(1-\lambda)\alpha}_{ix}\ \eta_{ix}^{\lambda\beta}\ \eta'^{(1-\lambda)\beta}_{ix}} & si\ j \in N_i \\ 0 & en\ otro\ caso \end{cases} \quad (26)$$

donde el valor de λ se calcula para la hormiga $k \in \{1, 2 \dots m\}$ como:

$$\lambda_k = \frac{k-1}{m-1} \quad (27)$$

Con la introducción del modificador λ_k se asegura que la colonia de hormigas realice búsquedas en distintas regiones del frente de Pareto.

El BiAnt ofrece además la posibilidad de controlar el carácter explorador o explotador de resultados de las hormigas con variables α y β diferentes para cada objetivo (26).

El algoritmo Pareto Ant Colony Optimization (PACO) [60] también hace uso de matrices de feromonas independientes, una para cada uno de los *r* objetivos. Cada vez que una hormiga avanza a otro estado, se realiza una actualización local paso a paso de las *r* matrices de feromonas según la ecuación (13) y considerando un valor constante para el incremento de feromonas, que se corresponde con la cantidad inicial ($\Delta\tau = \tau_0$). La función para calcular la transición hacia el siguiente nodo se basa en la probabilidad de selección del algoritmo ACS, comentado anteriormente en la página 31, utilizando un vector con *r* pesos uniformemente aleatorios que se emplean para seleccionar el siguiente nodo *j* aplicando la ecuación

$$j = \begin{cases} max_{j \in N_i} \left(\left(\sum_{k=1}^{b} w_k \, \tau_{ij}^k \right)^{\alpha} \eta_{ij}^{\beta} \right) & si \; Q \leq Q_0 \\ \hat{j} & en \; otro \; caso \end{cases} \tag{28}$$

calculándose la variable aleatoria $\hat{j}$ de acuerdo con la siguiente probabilidad:

$$P_{ij} = \begin{cases} \frac{\left[\sum_{n=1}^{b} (w_k \, \tau_{ij}^n)\right]^{\alpha} [\eta_{ij}^n]^{\beta}}{\sum_{x \in N_i} \left([\sum_{n=1}^{b} (w_k \, \tau_{ix}^n)]^{\alpha} [\eta_{ix}^n]^{\beta} \right)} & si \; j \in N_i \\ 0 & en \; otro \; caso \end{cases} \tag{29}$$

Para cada posible nodo siguiente *j* se calcula su probabilidad de ser elegido y de entre todos los posibles se elige uno teniendo en cuenta las probabilidades de cada uno. Una forma sencilla de implementar un algoritmo de selección en base a la probabilidad de cada uno es añadir $E(10^k \cdot p_j)$ copias de cada nodo candidato *j* a un vector de *N* elementos, con $N = k*100$, siendo *k* la precisión de los decimales que se quiera tener en cuenta para la probabilidad p_j de cada nodo. $E(n)$ representa la parte entera de *n*. Este algoritmo es sencillo de implementar pero introduce un error inversamente proporcional a *k*. Por otro lado a mayor precisión *k* (y menor error por tanto) mayor lentitud de cálculo.

Algoritmos con una colonia y una matriz de feromonas

Multiobjective Ant Colony System (MOACS), implementado para dos objetivos, utiliza una matriz de feromonas y dos visibilidades, η^0 y η^1, una para cada objetivo a optimizar. La regla de transición entre estados es similar a la del algoritmo PACO:

$$j = \begin{cases} max_{j \in N_i} \left(\tau_{ij} [\eta_{ij}^0]^{\lambda\beta} [\eta_{ij}^1]^{(1-\lambda)\beta} \right) & si \; Q < Q_0 \\ \hat{j} & en \; otro \; caso \end{cases} \tag{30}$$

calculándose la variable aleatoria $\hat{j}$, que representa el siguiente nodo a seleccionar, de la siguiente forma:

$$P_j = \begin{cases} \frac{\tau_{ij} [\eta_{ij}^0]^{\lambda\beta} [\eta_{ij}^1]^{(1-\lambda)\beta}}{\sum_{x \in N_i} \left(\tau_{ix} [\eta_{ix}^0]^{\lambda\beta} [\eta_{ix}^1]^{(1-\lambda)\beta} \right)} & si \; j \in N_i \\ 0 & en \; otro \; caso \end{cases} \tag{31}$$

Al igual que en PACO, cada hormiga al llegar al nodo *j*, deposita feromonas en el arco l_{ij} según la ecuación (11), teniendo en cuenta una variación de feromonas fija, mayor que cero e igual a la cantidad de feromonas inicial ($\Delta\tau = \tau_0$).

Al final de cada iteración, si la solución encontrada es no dominada, se actualiza el Conjunto de Pareto y se reinicia la matriz de feromonas con todos sus elementos iguales a τ_0. Si la solución encontrada es dominada se realiza la actualización de feromonas según la ecuación (11) teniendo en cuenta, ahora sí, la variación de feromonas correspondiente al mejor camino para $\Delta\tau$.

Un enfoque parecido sigue la ampliación del algoritmo Max-Min propuesta por Pinto et al. en [62] con el objetivo de resolver problemas de cuatro objetivos. Dicha ampliación, denominada Multiple Max-Min Ant System (M3AS), mantiene una única matriz de feromonas que se actualiza conjuntamente en función del refuerzo para cada objetivo. Las soluciones no dominadas actualizan la matriz según la ecuación (11) aplicando las cotas máximas y mínimas que se definen en el algoritmo Max-Min a los niveles de feromonas. Esto significa que:

$$\text{Si } \tau_{ij} > \tau_{max} \Rightarrow \tau_{ij} \leftarrow \tau_{max}$$

$$\text{Si } \tau_{ij} < \tau_{min} \Rightarrow \tau_{ij} \leftarrow \tau_{min} \tag{32}$$

El valor máximo permitido para el nivel de feromonas en una colonia de *m* hormigas es $\tau_{max} = \frac{\Delta\tau^k}{(1-\rho)}$, y el valor mínimo $\tau_{min} = \frac{\Delta\tau^k}{2m\,(1-\rho)}$, dónde *k* representa a la solución de la hormiga *k*-ésima y su variación de feromonas, $\Delta\tau^k$, se calcula según la ecuación (23).

Otro algoritmo de enfoque similar al M3AS es el Multiobjective Omicron [64] ACO (MOA), que utiliza una única tabla de feromonas y dos visibilidades, una para cada uno de los objetivos para los que fue diseñado. El algoritmo almacena una población de soluciones no dominadas durante un número *k* iteraciones, antes de actualizar con ellas la tabla de feromonas. La cantidad de feromonas que cada hormiga deposita es fija, y recibe el nombre de *ómicron*, siendo la regla de actualización de feromonas $\tau_{ij} = \tau_{ij} + \theta/h$, dónde θ es el *ómicron* y h el número de soluciones no dominadas.

Algoritmos con varias colonias y varias matrices de feromonas

El Bicriterion Multi Colony (BiMC) [59] es una ampliación del BiAnt dada por los mismos autores. En BiMC se consideran tantas colonias como objetivos, cada una con su matriz de feromonas asociada y su propia visibilidad (valor heurístico) y se fuerza explícitamente a cada colonia a buscar cada una en una región diferente del frente de Pareto.

Otro algoritmo que opta por la solución basada en más de una colonia de hormigas y varias matrices de feromonas es el COMPETants [63] (cuyo significado es Competing ants). Definido para un problema con dos objetivos a optimizar, usa dos matrices de feromonas, dos visibilidades y dos colonias de hormigas. El número de hormigas para cada colonia no es fijo, sino que varía dinámicamente durante la ejecución del algoritmo, recibiendo la colonia con mejor solución más hormigas para la siguiente iteración. Cada colonia se centra en la optimización de un objetivo, utilizando sus miembros únicamente las feromonas depositadas por otras hormigas pertenecientes a la misma colonia.

Una de las novedades del algoritmo COMPETants es la definición del concepto de hormiga espía, como aquella que usa, además, las feromonas depositadas por las hormigas de la otra colonia para la creación de su camino. Las hormigas espías se crean tras la primera iteración, una vez concluido el procedimiento de adaptación del número de miembros de cada colonia, que aumenta la población de la colonia que ha obtenido el mejor coste medio y disminuye la de su colonia competidora. El número de hormigas espías depende de la calidad del mejor recorrido de cada colonia, necesitando la colonia cuyo mejor recorrido sea de mayor coste (peor en un contexto de minimización) un mayor número de espías que su colonia oponente.

Las hormigas normales eligen el siguiente nodo con la probabilidad indicada en la ecuación siguiente:

$$P_j = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{x \in N_i}([\tau_{ix}]^{\alpha}[\eta_{ix}]^{\beta})} & si\ j \in N_i \\ 0 & en\ otro\ caso \end{cases} \tag{33}$$

Las hormigas espías, por su parte, utilizan feromonas tanto de la colonia a la que pertenecen como del resto para construir su recorrido, seleccionando en cada paso el siguiente nodo en función de la probabilidad de selección dada por la ecuación siguiente:

$$P_{ij}(t) = \begin{cases} z & si\ j \in N_i \\ 0 & en\ otro\ caso \end{cases} \tag{34}$$

donde *z* es:

$$z = \frac{\left[\left(\frac{1}{2}\ \tau_{ij}^{own}\right) + \left(\frac{1}{2}\ \tau_{ij}^{foreign}\right)\right]^{\alpha} [\eta_{ij}]^{\beta}}{\sum_{h \in N_i}\left(\left[\left(\frac{1}{2}\ \tau_{ih}^{own}\right) + \left(\frac{1}{2}\ \tau_{ih}^{foreign}\right)\right]^{\alpha} [\eta_{ih}]^{\beta}\right)} \tag{35}$$

siendo τ_{ij}^{own} y $\tau_{ij}^{foreign}$ las feromonas de la propia colonia y de la ajena, respectivamente. La actualización de feromonas en el algoritmo de colonias competitivas COMPETants, se realiza de forma retardada, utilizando el enfoque del algoritmo AS_{rank} de acuerdo con la regla:

$$\tau_{ij} = \rho\ \tau_{ij} + \sum_{\lambda=1}^{\delta} \Delta\tau_{ij}^{\lambda}\ , \forall i, j \in J \tag{36}$$

donde ρ es el coeficiente de evaporación (con $0 \leq \rho \leq 1$), *J* es el conjunto de nodos origen y donde tan sólo las δ mejores hormigas (que hayan obtenidos los recorridos de mejor coste en la última iteración, según la métrica de calidad usada para confeccionar el ranking) contribuirán al depósito de feromonas con una cantidad $\Delta\tau_{ij}^{\lambda}$.

$$\Delta\tau_{ij}^{\lambda} = \begin{cases} 1 - \dfrac{\lambda - 1}{\delta} & si\ 1 \leq \lambda \leq \delta \\ 0 & en\ otro\ caso \end{cases} \tag{37}$$

Capítulo 3. Aplicación al Problema de las Rutas de Aprendizaje.

El Problema de las Rutas de Aprendizaje consiste en averiguar la mejor secuencia de cursos que necesita realizar un alumno para adquirir un conjunto de competencias de forma que se maximice el conocimiento adquirido y se minimice el tiempo empleado.

Imagine que se cuenta con una base de datos de competencias y sus relaciones con una serie de recursos educativos que pueden facilitarnos la adquisición de dichas competencias. Este es el fin de los estándares IEEE Reusable Competency Definitions [77] y IEEE Simple Reusable Competency Map [78] encargados de la definición de competencias y de un mapa que las relacione entre sí, permitiendo jerarquías de competencias y competencias afines.

Partiendo de esta base de datos que nos relaciona competencias con recursos educativos podrían definirse itinerarios formativos que posibiliten la adquisición de un determinado número de competencias, facilitando la elaboración de itinerarios formativos orientados a competencias en lugar de a recursos educativos. Sin embargo, aunque las competencias adquiridas con varios recursos educativos sean las mismas, el grado de conocimientos y el tiempo que es necesario emplear para asimilarlos depende en gran medida de factores como la calidad de estos recursos, del orden en que se realizan, etc. Y además, también depende del momento, puesto que un determinado orden o conjunto de recursos educativos puede mostrarse eficaz durante un determinado periodo de tiempo pero perder valor en el futuro o aparecer nuevos recursos que mejoren el rendimiento de los participantes, por lo que es necesario un sistema que pueda evolucionar y seleccionar la mejor ruta en cada momento.

3.1 Escenario

Sea una empresa que desea preparar un itinerario formativo para un conjunto de trabajadores de cara a ocupar puestos de mayor responsabilidad. Estos puestos requieren de un conjunto de habilidades y conocimientos (competencias) que los

aspirantes han de adquirir para optar al mismo, y para ello es necesario superar con éxito una serie de cursos de formación específicos. La empresa desea que este itinerario formativo pueda repetirse con distintas promociones[9] y pueda adaptarse dinámicamente a las necesidades cambiantes que pueden darse en cada promoción, contribuyendo a la mejora de la formación de los alumnos participantes.

La adquisición de una determinada competencia requiere la superación de un determinado conjunto de cursos, a menudo en un determinado orden. Igualmente, las competencias necesarias para el puesto pueden ser interdependientes y requerir cierto orden. No obstante las restricciones de ordenación son opcionales y podría suceder que las competencias necesarias fueran independientes entre sí y no importase el orden en el que fueran adquiridas. Partiendo de la hipótesis de que un mismo itinerario de aprendizaje no siempre es el más adecuado para poblaciones de alumnos diferentes, la empresa desea diseñar un itinerario formativo que permita diferentes rutas alternativas, de forma que existan itinerarios con diferentes ritmos de aprendizaje, permitiendo al sistema gestor del aprendizaje escoger la ruta más conveniente en cada momento con el objetivo de maximizar la calificación media final de los alumnos.

3.2 Estudios previos

El sistema Parschool descrito por Valigiani et al. [81] hace uso de un algoritmo ACO modificado que utiliza dos tipos de feromonas, uno para el refuerzo del éxito y otro para el fracaso, que se utilizan de forma conjunta en una función de ajuste para seleccionar los mejores caminos que serán recomendados a los alumnos en cada nodo de decisión.

En Gutiérrez et al. [118] se registra la frecuencia y rendimiento (en términos de éxito o fracaso) de varios itinerarios que otros alumnos han seguido, y utilizando un algoritmo ACO muestran al alumno las diferentes opciones en cada nodo de decisión y su valoración (a partir de las feromonas en cada itinerario) para que el alumno escoja. Esta aproximación utiliza ACO como mecanismo de recomendación de itinerarios pero

[9] Una promoción es el conjunto de alumnos que participan en la formación y que disponen de un plazo de tiempo determinado para finalizar su itinerario, con fecha de inicio y fin concretas y común para todos ellos.

deja la elección del siguiente nodo al alumno, lo que dificulta su estudio debido a los complejos mecanismos de toma de decisión en los humanos.

A diferencia de los anteriores, Van den Berg et al. [119] desarrollan un algoritmo ACO simplificado que sólo tiene en cuenta los resultados obtenidos por aquellos alumnos que han finalizado el curso completo. Este algoritmo utiliza una matriz que relaciona los caminos posibles en cada nodo con las veces que cada uno ha sido seleccionado, como generador de probabilidades para el método Monte Carlo que se usará con el objetivo de recomendar al siguiente alumno el camino a seguir. En un trabajo posterior [120], analizaron las actividades de aprendizaje on-line de más de 800 alumnos, comparando los resultados de aquellos que usaron la recomendación con los que no lo hicieron. El estudio concluyó que era efectivo para mejorar la calificación media del grupo a través de la recomendación pero no era eficiente en términos del tiempo total invertido por los alumnos para completar el curso.

Uno de los últimos trabajos que enfrentan un problema similar al del escenario de esta Tesis, Style-based Ant Colony Systems (SACS) [121], utiliza el algoritmo ACS para adaptar la recomendación de itinerarios de aprendizaje según el estilo de aprendizaje de cada alumno. SACS categoriza a los alumnos en cuatro grupos según su estilo de aprendizaje: visual, auditivo, textual o kinestésico[10]. A diferencia de otras aproximaciones, las feromonas depositadas no dependen del rendimiento de los alumnos sino simplemente del número de alumnos de cada categoría que han realizado cada itinerario.

3.3 Propuesta de solución

La solución propuesta incorpora varias de las características de los de los estudios previos como son la utilización de dos tipos de feromonas, el uso de una función de ajuste que define el concepto de distancia entre los cursos y tener en cuenta el rendimiento de los alumnos (la calificación obtenida en cada curso). Como principales diferencias con respecto a los estudios previos, no se trata de un mecanismo para la recomendación de contenidos o unidades didácticas internas a un mismo curso, sino a cursos independientes. Además, nuestra aproximación no se centra en la

[10] El alumno kinestésico es aquel que aprende "haciendo", tiene especiales habilidades para todas las actividades que impliquen movimiento y la manipulación de objetos. Son aquellos alumnos que les cuesta por lo general estar sentados por mucho tiempo, suelen ser cuestionadores y piden "ver para creer".

recomendación del siguiente curso sino en la asignación de cada curso, hasta completar de forma automática el itinerario para cada alumno, paso a paso, con el objetivo de estudiar la viabilidad de la aplicación de algoritmos ACO en la evolución del itinerario preferente según el rendimiento observado en el conjunto de alumnos.

Para facilitar la consecución de los objetivos planteados en el escenario anterior, se propone el uso de un grafo con restricciones de navegación para el diseño del itinerario de aprendizaje por el equipo pedagógico (Figura 7), en el que se representen los diferentes caminos de aprendizaje alternativos.

3.3.1 Grafo de Itinerarios de Aprendizaje

El Grafo de Itinerarios de Aprendizaje (GIA) es la herramienta propuesta para el diseño de itinerarios de aprendizaje adaptativos. En el grafo de la Figura 7 se distinguen tres tipos de entidades: nodos, arcos y restricciones de navegación.

Los nodos representan cursos y las aristas o arcos dirigidos que los unen representan las posibles transiciones entre un curso y el siguiente. Cada arista es ponderada con un peso que designa la probabilidad de ser el curso destino de la arista el que sea entregado al alumno. Así, las aristas con una probabilidad mayor serán elegidas con mayor frecuencia.

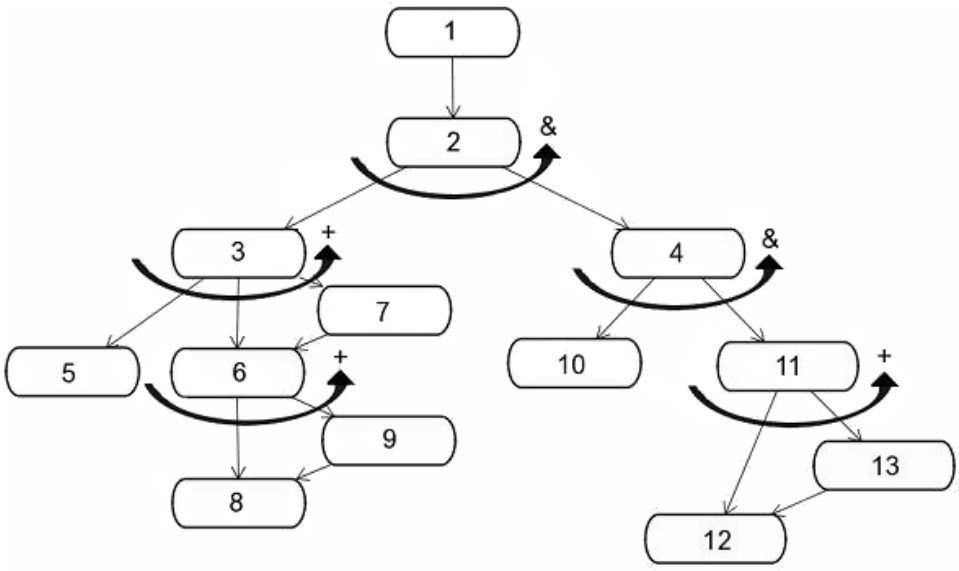


Figura 7: Grafo de Itinerarios de Aprendizaje (GIA)

Las restricciones de navegación se representan con una flecha curva etiquetada con un símbolo & (restricción AND) o + (restricción OR). Las restricciones posibles se

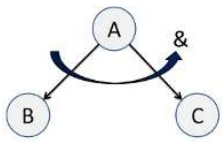
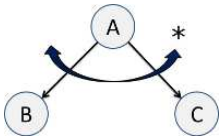
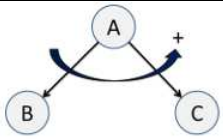
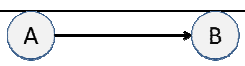
muestran en la Tabla 5. Todas las restricciones afectan a todos los arcos de salida del nodo para el que se definen.

La restricción AND indica que las transiciones han de realizarse en el orden establecido, por lo que para esta restricción el orden en el que se indican los nodos destinos es especialmente importante, a diferencia de la restricción OR en la que el orden en que se indiquen no tiene importancia.

La restricción de navegación AND es útil para delimitar secuencias obligatorias, no sólo de cursos, sino de bloques de cursos, por lo que puede ser usada por el equipo pedagógico para separar los conjuntos de cursos que forman al alumno para la adquisición de una determinada competencia.

La restricción de navegación OR sirve para definir caminos alternativos en función de las necesidades identificadas por el equipo de formación (caminos de refuerzo, caminos especialmente indicados para alumnos con ritmos de aprendizaje diferente, caminos de profundización, etc.)

Tabla 5: Restricciones de navegación

Tipo	Representación	Descripción	Sintaxis	Secuencias
AND		Restricción de Navegación obligatoria con orden específico. Visitar todos los nodos incluidos en la restricción en el orden indicado.	A & (B,C)	A→B→C
AND		Restricción de Navegación obligatoria sin importar el orden. Deben visitarse todos los nodos incluidos en la restricción sin importar el orden en que se visiten.	A*(B,C)	A→B→C A→C→B
OR		Restricción de Navegación opcional. Visitar sólo un nodo de entre los posibles nodos destinos implicados en la restricción.	A|(B,C)	A→B A→C
LINK		Enlace directo. Sólo es posible si el grado de salida del nodo origen es 1.	A,B	A→B

El itinerario propuesto en la Figura 7 podría representar tres grandes competencias A, B y C, determinadas por la primera relación AND que relaciona el nodo 2 con los nodos destino 3 y 4. De la misma forma la relación 4&(10,11) puede identificar dos bloques dentro de la competencia C (Figura 8). En este caso se pretende definir en el grafo itinerarios alternativos al habitual formado por la secuencia ordenada de nodos 1, 2, 3, 5, 4, 10, 11, 12. De esta forma según criterio del equipo pedagógico pueden definirse itinerarios de refuerzo a diferentes niveles, mediante relaciones OR, por ejemplo en los nodos 3, 6 y 11.

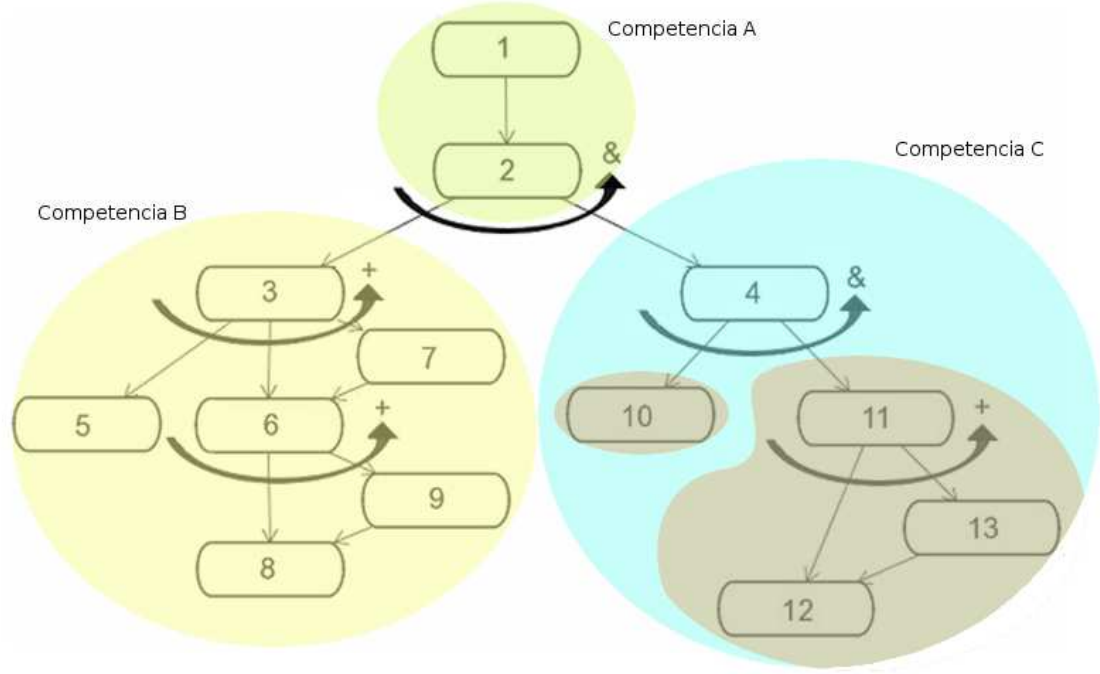


Figura 8: Identificación de competencias en el grafo de itinerarios de aprendizaje

El GIA permite diferentes grafos inconexos en caso de que no exista relación entre los cursos de uno y otro, y el orden en el que se completen sea irrelevante (Figura 9).

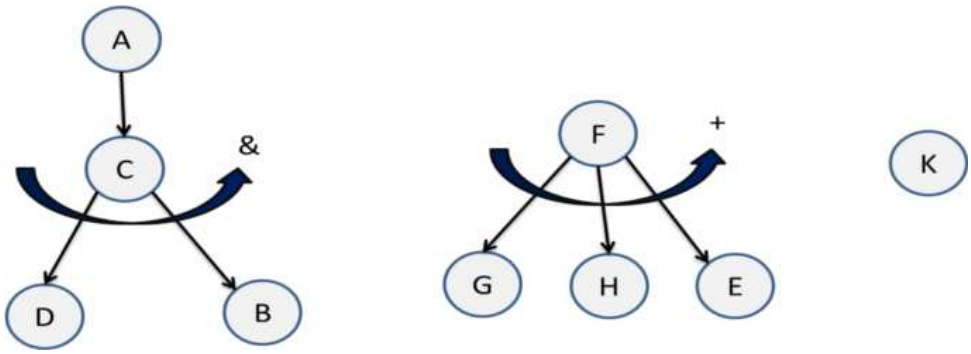


Figura 9: Ejemplo de grafo para itinerarios de aprendizaje independientes

El objetivo es encontrar el mejor itinerario para la mayoría de alumnos, pero haciendo posible la recomendación de caminos alternativos en función de las necesidades específicas de cada uno de ellos. La personalización del itinerario consiste un camino alternativo entre dos nodos. Existen ciertas condiciones que deben cumplir todos y cada uno de los grafos que componen el GIA:

a) Han de ser grafos dirigidos.

b) Han de contar con al menos un nodo con grado de entrada 0, a los que llamaremos *nodos raíz*.

c) Han de tener al menos un nodo con grado de salida 0, a los que llamaremos *nodos hoja*.

d) No pueden tener ciclos.

e) Los arcos de salida de un nodo pertenecen a alguno de los tres tipos de restricciones posibles: AND, OR o LINK.

f) Una restricción de navegación agrupa a todos los arcos de salida de un nodo.

g) Una misma restricción de navegación no puede afectar a arcos cuyos orígenes sean nodos diferentes.

h) Los nodos participantes en una restricción de navegación de tipo AND no están conectados entre sí ni directa, ni indirectamente. Es decir, no existe un camino que los conecte.

i) Sea el grafo G=(N,A) y $p, q \in N$ dos nodos unidos con una restricción de navegación de tipo AND, y $r \in N$ otro nodo del grafo, entonces si existe un camino entre q y r, no existe camino entre p y r.

Dada la posibilidad de contar con multigrafos, para evitar un mayor nivel de complejidad en el procesado del mismo, es necesario transformar el Grafo de Itinerarios de Aprendizaje en un grafo conexo siguiendo los siguientes pasos que se describen a continuación:

1. Eliminar relaciones AND de cada grafo del GIA.
2. Crear un único nodo Inicial y un único nodo Final.
3. Conectar cada nodo inicial con todos los nodos raíz del GIA.
4. Conectar todos los nodos hoja con el nodo final.

3.3.2 Representación formal del grafo

Son varios los formatos existentes para la representación de grafos, entre los que podemos destacar Graph eXchange Language (GXL[11]), Graph Markup Language (GML[12]), eXtensible Graph Markup and Modeling Language (XGMML[13]) y GraphML[14].

Se ha elegido GraphML principalmente por su sencillez y extensibilidad. GraphML es un lenguaje basado en XML que permite definir la estructura del grafo, y además ofrece un sencillo mecanismo de extensión para dar soporte a datos específicos de cada problema. GraphML soporta tanto grafos dirigidos como no dirigidos. La sintaxis de un grafo en formato GraphML viene definida por su esquema[15] XML. El Listado 3 muestra la representación gráfica de un grafo dirigido sencillo.

El formato de un documento GraphML consta un elemento `graphml` y varios subelementos: `graph`, `node` y `edge` que definen los diferentes grafos, nodos y arcos que lo componen. A continuación se describe brevemente la estructura de un documento en formato GraphML.

Cabecera

La cabecera de un documento GraphML tiene el formato que se muestra en el Listado 4.

La primera línea del documento indica que es un document conforme con el estándar XML 1.0 y que el juego de caracteres utilizado es UTF-8, la codificación estándar para documentos XML. Por supuesto otras codificaciones podrían tenerse en cuenta para documentos GraphML.

[11] http://www.grupo.de/GXL/
[12] http://www.infosun.fim.uni-passau.de/Graphlet/GML/
[13] http://www.cs.rpi.edu/~puninj/XGMML/
[14] http://graphml.graphdrawing.org/
[15] http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns/graphml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns/graphml">

 <key id="y" for="node">
  <desc>Y coordinate</desc>
  <default>0</default>
 </key>
 <key id="x" for="node">
  <desc>X coordinate</desc>
  <default>0</default>
 </key>
 <key id="id" for="edge">
  <desc>ID of the link</desc>
 </key>
 <key id="weight" for="edge">
  <desc>Weight of the arc</desc>
  <default>0</default>
 </key>

 <graph edgedefault="directed">
 <node id="Node1">
  <data key="y">226.0</data>
  <data key="x">271.0</data>
 </node>
 <node id="Node0">
  <data key="y">54.0</data>
  <data key="x">376.0</data>
 </node>

 <edge source="Node0" target="Node1">
  <data key="id">0</data>
  <data key="weight">5.0</data>
 </edge>

 </graph>
</graphml>
```

Listado 3: Ejemplo de grafo serializado en formato GrahML

En el Listado 4, la segunda línea contiene el elemento raíz: `graphml`. El elemento `graphml`, al igual que otros elementos de GraphML, se definen en el espacio de nombres del lenguaje y por este motivo se define este espacio de nombres como el espacio de nombres por defecto, por lo que no es necesario prefijar los elementos del lenguaje con ningún prefijo distintivo.

```
 <?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
   http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
```

Listado 4: Cabecera de un documento GraphML

Los otros dos atributos XML son necesarios para especificar el esquema XML para este documento. En el ejemplo se usa el esquema estándar de GraphML ubicado en el servidor `graphdrawing.org`. EL primer atributo, define el prefijo `xsi` para los elementos del espacio de nombres de XML.

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

El segundo atributo, indica la ubicación para todos los elementos del espacio de nombres de GraphML:

```
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd"
```

Aunque es útil para poder validar que el documento esté correctamente formado, las referencias al esquema XML no es obligatoria y la primera parte del documento puede quedar mucho más sencilla (ver Listado 5).

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
 ...
</graphml>
```

Listado 5: Cabecera reducida de un documento GraphML

Grafo

El grafo se declara con el elemento `graph`. Como elementos internos del grafo, se definen los elementos `node` y `edge` para los nodos y arcos del grafo respectivamente. Aunque en el Listado 6 aparecen primero los nodos y luego los arcos, no existe un orden definido para ello y podrían aparecer mezclados.

```
<graph edgedefault="directed">
 <node id="Node1">
  <data key="y">226.0</data>
  <data key="x">271.0</data>
 </node>
 <node id="Node0">
  <data key="y">54.0</data>
  <data key="x">376.0</data>
 </node>

 <edge source="Node0" target="Node1">
  <data key="id">0</data>
  <data key="weight">5.0</data>
 </edge>

</graph>
```

Listado 6: Ejemplo de grafo en GraphML

Declaración de un grafo

El lenguaje permite mezclar arcos dirigidos y no dirigidos, por lo que si no se especifica ninguna dirección en la declaración de un arco, se aplicará el tipo definido de arco definido por defecto en el atributo `edgedefault` del elemento `graph` (Listado 7). Puesto que el lenguaje permite multigrafos, opcionalmente puede asociarse un identificador a cada grafo, a través del atributo `id`, para poder referenciarlo.

```
<graph id=”G1” edgedefault="directed">
 ...
 </graph>
```

Listado 7: Declaración de un grafo en GraphML

Declaración de un nodo

Un nodo se declara mediante el elemento `node` dentro de un elemento `graph`. Cada nodo ha de tener un identificador único en todo el documento. Es decir, no puede existir otro nodo con el mismo identificador, ni dentro del mismo grafo ni en ningún otro grafo.

Declaración de un arco

Los arcos del grafo se declaran con el elemento `edge`. Cada arco debe definir sus extremos mediante los atributos `source` y `target` que denotan el nodo origen y el nodo destino respectivamente.

```
...
<edge id="e1" directed="true" source="n0" target="n2"/>
...
```

Listado 8: Declaración de un arco en GraphML

Los arcos con destino el propio nodo origen, también denominados nodos reflexivos, se declaran con el mismo identificador de nodo para los atributos `source` y `target.`

Se puede indicar si un arco es dirigido o no con el atributo `directed`. Si no se especifica este atributo se tendrá en cuenta el definido por defecto para el grafo. Opcionalmente puede diferenciarse el arco del resto mediante el atributo `id` que, en ese caso, ha de ser único.

Declaración de atributos adicionales

Como puede verse en el Listado 3, la sintaxis de GraphML define una etiqueta `key` que permite identificar datos específicos que pueden asociarse a los elementos estructurales del grafo (nodos y arcos), así como al propio grafo. La etiqueta `key` puede incorporar atributos para especificar el identificador de dicho atributo y, opcionalmente, un nombre y tipo para el mismo. Los atributos nombre y tipo, `attr.name` y `attr.type` respectivamente no se usan en el documento y su funcionalidad es poder ser de utilidad a las aplicaciones que procesan el grafo y crean las instancias en algún lenguaje de programación. El tipo está especialmente pensado para un mapeo directo con los tipos de Java, pudiendo tomar los valores `boolean`, `int`, `long`, `float`, `double`, o `string`.

El atributo `for` de un elemento `key` (Listado 9) puede hacer referencia, como se ha indicado, tanto a los nodos como a los arcos, pero también al propio grafo y a todos los elementos, pudiendo tomar los valores `graph`, `node`, `edge` y `all`.

Las propiedades que pueden definirse para cada atributo se limitan a la descripción del atributo y a su valor por defecto, que se definen respectivamente con los elementos `description` y `default`, incluidos dentro del elemento `key`, tal y como muestra el Listado 9.

```
...
<key id="weight" for="edge" attr.name="weight" attr.type="double">
  <desc>Weight of the arc</desc>
  <default>0.0</default>
</key>
...
```

Listado 9: Declaración de la descripción de un atributo y su valor por defecto

Este mecanismo de declaración de atributos adicionales es el empleado para la relación de cursos SCORM con nodos del grafo (Listado 10). Se utiliza una cadena de texto que representa la URL desde la que acceder al paquete SCORM que contiene el curso. Esta URL ha de ser codificada en Base64[16] para evitar conflictos con el lenguaje XML empleado. Base64 es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII. Esto ha propiciado su uso para codificación de correos electrónicos y otras aplicaciones. Todas las variantes famosas que se conocen con el nombre de Base64 usan el rango de caracteres A-Z, a-z y 0-9.

El uso de un codificador de URL sobre Base64 estándar, sin embargo, no resulta adecuado ya que traducirá los caracteres '+' y '/' en las secuencias especiales '%2B' y '%2F' respectivamente. Si se usa para almacenamiento en base de datos o entre sistemas heterogéneos, producirán un conflicto en el carácter '%' generado por el codificador de URL debido a que este carácter es usado en ANSI SQL como comodín. Este inconveniente puede solucionarse fácilmente modificando el algoritmo de codificación en base 64, para sustituir directamente los caracteres '+' y '/' por '*' y '-' respectivamente, de manera que ya no se necesita usar codificadores de URL (Listado 11).

[16] RFC 2045: http://www.ietf.org/rfc/rfc2045.txt

```
...
<key id="course" for="node" attr.name="course" attr.type="string">
  <desc>URL to the SCORM Course</desc>
  <default></default>
</key>
...
```

Listado 10: Declaración de la URL de un curso SCORM como atributo de un nodo

En el código del ejemplo del Listado 11 se ha utilizado codificado en Base64 la URL http://repository.us.es/scorm/lsi1234, que representa al recurso lsi1234 obtenido como un recurso REST. Para la codificación se utiliza el algoritmo de codificación/decodificación en base 64 de Stephen Ostermiller[17].

```
...
<node id="Node1">
  <data key="y">226.0</data>
  <data key="x">271.0</data>
  <data key="course">
aHR0cDovL3JlcG9zaXRvcnl1c2VzL3Njb3JtL2xzaTEyMzQ=</data>
</node>
...
```

Listado 11: Ejemplo de nodo con atributo para la URL del curso (http://repository.us.es/scorm/lsi1234)

Declaración de restricciones de navegación

El formato de definición de atributos asociados a los nodos de GraphML facilita la definición de las restricciones del problema. En el escenario propuesto, las restricciones pueden declararse como atributos asociados al nodo origen de la restricción (Listado 12).

```
...
<key id="rel" for="node" attr.name="constraint" attr.type="string">
<desc>Constraint</desc>
<default>LINK</default>
</key>
...
```

Listado 12: Declaración de restricciones como atributos de los nodos

[17] Base64.java Source Code. © 2001-2010 Stephen Ostermiller. Licencia GPL v2. http://ostermiller.org/utils/Base64.html

Y en la declaración de cada nodo podría utilizarse el atributo restricción para indicar el tipo de restricción aplicable a los arcos de salida. Como puede apreciarse en el Listado 13, se utiliza un lenguaje basado en tokens para diferenciar las diferentes restricciones de navegación.

```
...
<node id="Node0">
<data key="y">209.0</data>
<data key="x">37.0</data>
<data key="rel">&Node0::Node1::Node2</data>
</node>
...
```

Listado 13: Restricción de navegación en la declaración de un nodo

El principal problema asociado a este mecanismo de definición de restricciones de navegación a través del uso de atributos asociados a los nodos, es que a pesar de cumplir con la especificación del lenguaje, carece de información semántica para poder validar una de las restricciones indicadas para los GIA: no es posible garantizar que las restricciones de navegación "LINK" no sean asociadas a nodos con grados de salida mayor que 1. Desde el punto de vista sintáctico el XML es correcto, pero desde el punto de vista semántico, tal y como se ha definido el concepto de GIA, no. Por tanto, la herramienta que se implemente para dar soporte al diseño de GIA a equipos pedagógicos debe realizar esta validación al cargar un fichero GraphML y también en tiempo de edición de un grafo.

Para evitar este inconveniente se ha investigado [23], [79] la definición de restricciones mediante el modelado del grafo basado en características, usando el framework de modelado de características MosKitt desarrollado por la Universidad Politécnica de Valencia, que nos permite introducir la información semántica necesaria para validar el documento, evitando el problema anterior. Para integrarlo en el documento GraphML es necesario importar el esquema de MosKitt en la cabecera del documento y definir las restricciones de navegación como características.

En [79] además, las restricciones de navegación se consideraron relaciones, definiéndose además las relaciones que indican el carácter opcional u obligatorio de un nodo, así como la inclusión o exclusión de un nodo para indicar relaciones de dependencia entre nodos. Esta última está más relacionada con la implementación de la

plataforma sobre OSGi, sin efectuar diferenciación alguna entre nodos de cursos y nodos de librerías para la reproducción de un curso que pueden definirse en el descriptor del curso, liberando al modelo pedagógico de esta carga.

No obstante, por la complejidad que introduce la integración de dos modelos diferentes, es preferible delegar la validación en la capa de aplicación, lo que nos permite contar con un documento menos exhaustivo, más permisivo, pero mucho más simple y legible.

3.3.3 Transformación del Grafo

Eliminación de Restricciones

Para simplificar el procesado del GIA es necesario eliminar las restricciones AND, que si bien facilitan la compresión del grafo a los humanos, dificulta el procesado automático. Aplicando el algoritmo del Listado 14 se consigue transformar un GIA (Figura 10) en un grafo dirigido sin restricciones (Figura 11).

La función recursiva `procesarRama` es la encargada de crear un grafo que será fusionado con la función `fusionar` que recibe como parámetro la lista de nodos de un grafo que han de ser fusionados con un nodo de un segundo grafo, creando un arco que une cada uno de los nodos de la lista con el nodo destino.

Una vez eliminadas las restricciones, el itinerario queda reflejado por un grafo dirigido, en el que el itinerario de aprendizaje va desde los nodos raíz a los nodos hoja.

Creación de nodos Inicial y Final

Puesto que el GIA puede estar compuesto de varios grafos y además, como se observa en la Figura 10, un grafo puede tener más de un nodo raíz, se hace necesario añadir un único punto de inicio que nos permita unir todos los grafos independientes resultantes de la transformación anterior (Listado 15). De la misma forma, para facilitar la evaluación de todos los participantes que terminan el itinerario, se introduce un nodo final al que todos llegarán (Listado 16).

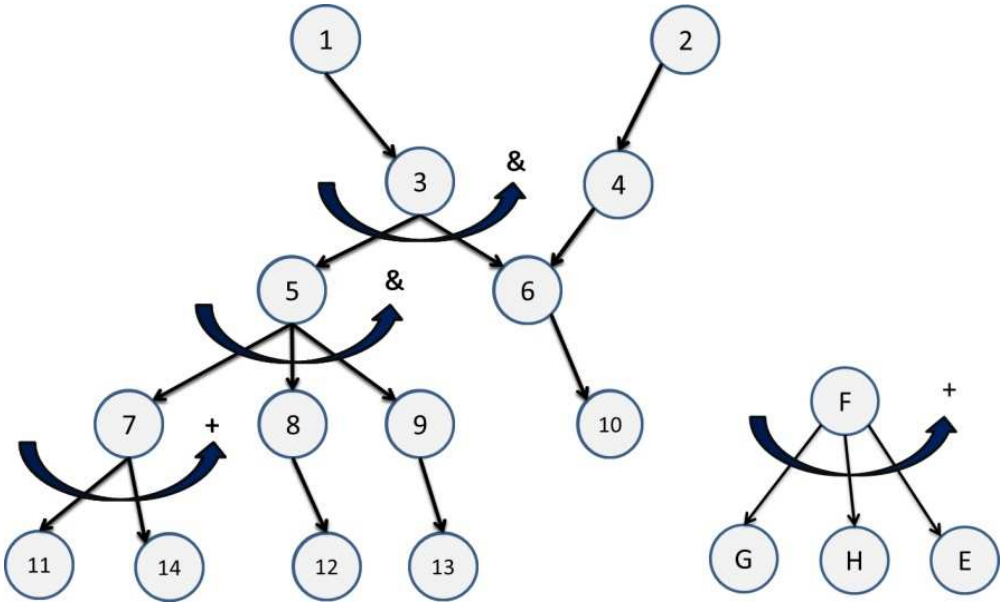


Figura 10: Ejemplo de Grafo de Itinerarios de Aprendizaje

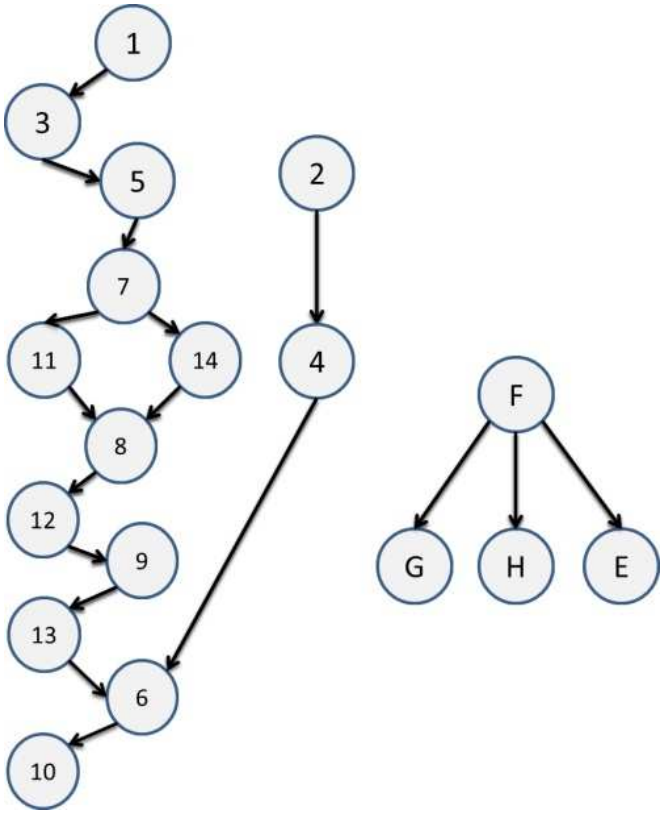


Figura 11: Grafo equivalente sin restricciones

```
Procedimiento transformarGIA
 G' ← nuevoGrafo (∅,∅)
 Para todo G(N,A) ∈ MIA hacer:
R ← raíces(G)
Para todo r ∈ R
procesarRama(r,G')
finPara
 finPara
finProcedimiento

función procesarRama(Nodo n, Grafo G)
 añadirNodo(n,G)
 res ← obtenerRestriccion(n)
 si res = null
    G' ← nuevoGrafo (n,∅)
 si res ≠ null ^ res no ha sido procesada ya
si tipo(res) = LINK entonces
    m ← destino(res)
          G' ← procesarRama(m,G(∅,∅))
       fusionar ({n},G,m,G')
 si tipo(res) = OR entonces
       Para todo nodo m ∈ destinos(res)
            G' ← procesarRama(m, G(∅,∅))
          fusionar({n},G,m,G')
       finPara
    si tipo(res) = ORDERED_AND entonces
       H ← hojas(G)
       Para todo nodo m ∈ destinos(res)
          G' ← procesarRama(m, G(∅,∅))
          fusionar(H,G,m,G')
          H← hojas(G)
       finPara
    si tipo(res) = UNORDERED_AND entonces
       P ← permutaciones(destinos(res))
       H ← hojas(G)
       Para toda permutación p ∈ P
          Para todo nodo m ∈ p
             G' ← procesarRama(m, G(∅,∅))
             fusionar(H,G,m,G')
          finPara
       finPara
    finSi
    marcar res como procesada
 finSi
devolver G'
finFuncion
```

Listado 14: Pseudocódigo para la transformación del GIA en un grafo dirigido eliminando las restricciones

```
i = Nodo inicial

P ← Escoger una permutación de los grafos resultantes
g = Extraer primer grafo de P
Conectar i con cada nodo raíz de g
Mientras quede grafo g' en P
  Conectar cada nodo hoja de g con cada nodo ráiz de g'
  g ← g'
  g' ← siguiente grafo de P
finMientras
```

Listado 15: Creación de nodo inicial

Una vez unidos todos los grafos independientes mediante un nodo inicial, se procede a unir las hojas existentes con el nodo final.

```
Hojas = Lista de nodos hoja
f = Nodo final
Para todo n ∈ Hojas
   Crear arco L_nf
finPara
```

Listado 16: Creación de nodo final

Una vez añadidos los nodos Inicial y Final, el grafo resultante de la eliminación de las restricciones queda finalmente como un único grafo conexo dirigido, con un

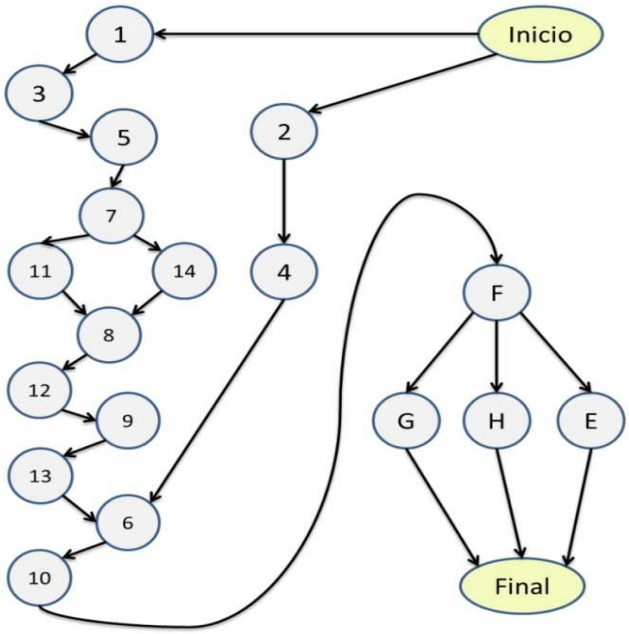


Figura 12: Diagrama de estados del itinerario de aprendizaje

único nodo raíz y un único nodo final (Figura 12). Denominamos a este grafo Diagrama de Estados del Aprendizaje o LSD (*Learning State Diagram*).

3.3.4 Adaptación del mejor camino

El problema de encontrar el camino más corto consiste en encontrar un camino entre dos nodos de tal manera que la suma de los pesos de las aristas que lo constituyen sea mínima.

El algoritmo de Dijkstra [80] determina el camino más corto desde un vértice origen al resto de vértices en un grafo ponderado. El mecanismo de este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen al resto de vértices que componen el grafo, el algoritmo se detiene.

En nuestro problema, los vértices representan los cursos, pero en cambio, el peso de las aristas depende del concepto de distancia que se desee aplicar. Las aristas expresan la posibilidad de transición de un curso *A* a otro *B*, pero el peso de la misma podría asociarse a diversos conceptos relacionados con el nodo destino:

- La duración media del curso (tiempo medio empleado por los alumnos en realizar el curso destino).
- Peso pedagógico del curso.
- Nivel de dificultad del curso.
- Calificación media obtenida por los participantes en el curso.

Para cualquiera de estos conceptos de distancia, el algoritmo de Dijkstra obtiene el camino de menor distancia de forma determinista. En cambio en el problema de las rutas de aprendizaje, esta distancia no es un valor determinado, sino que depende de los resultados obtenidos por los alumnos participantes en una edición así como de valores históricos a partir de los cuales inferir conocimiento.

Partiendo de la hipótesis que el perfil de los asistentes a la formación es similar (han sido seleccionados por la empresa para recibir una formación de cara a ser promocionados, por lo que reúnen una serie de requisitos comunes), no es descabellado pensar que los cursos que resulten adecuados para un pequeño grupo lo será también para el resto. No nos interesa tanto el camino más corto en este sentido sino el que

mejor se ajuste a la mayoría de alumnos, y puesto que esta idea de "mejor ajuste a las necesidades de la mayoría" es susceptible de variar para cada grupo de alumnos, se justifica en este hecho la utilización de técnicas probabilísticas como ACO en lugar de algoritmos deterministas como el algoritmo de Dijkstra.

Valigiani et al. [81] comenzaron a aplicar técnicas ACO a entornos de e-learning, gracias a que el material educativo disponible en internet puede ser organizado en forma de grafos que relacionan las diferentes unidades didácticas. Al igual que en [81], nuestra propuesta de solución también se basa en técnicas ACO que permitan al sistema recomendar la mejor opción en función de los resultados obtenidos por los alumnos. En nuestra aproximación, cada estudiante representa el papel de una de las hormigas virtuales empleadas por las técnicas ACO, moviéndose en el grafo, desde el nodo inicial, siguiendo la dirección marcada por cada arista y liberando feromonas en los arcos recorridos hasta llegar al nodo final. A diferencia de las feromonas positivas y negativas de valor fijo, propuestas en [81], en nuestra solución utilizaremos una cantidad de feromonas a depositar variable, que será función de la calificación obtenida por el alumno en el curso o cursos anteriores.

Así, cuando un alumno realiza la transición del curso *A* al *B*, una mayor calificación obtenida en el curso *B* implicará un mayor refuerzo en el arco L_{ab}, que tendrá cierta influencia en la selección del siguiente curso (en caso de ser *A* un nodo con grado de salida > 1) al curso *A*. En consecuencia, las feromonas representan un índice del nivel de éxito alcanzado en el nodo destino de un arco que une dos nodos.

Sea [0, M] con $M > 0$, el rango posible para la calificación que puedan obtener los alumnos en la evaluación de cada curso, cada curso puede tener asociado un mínimo requerido m, con $0 < m < M$, para considerar la calificación como aceptable. Este mínimo requerido servirá para decidir si el impacto de las feromonas ha de ser mayor o menor en función de la calificación obtenida.

Así, la cantidad de feromonas a depositar por el alumno *k* en el arco L_{ij} en función de la calificación obtenida, ε_j, trasladada a la escala [0,1], en el curso del nodo *j*, vendrá determinada por la siguiente expresión:

$$\Phi_{ij}^{k}(\varepsilon) = \begin{cases} -\left(1-\varepsilon_j^k\right)\varphi - \omega_3\left(\frac{m_i}{M_i} - \varepsilon_j^k\right) & si\ \varepsilon_j^k < \frac{m_i}{M_i} \\ \varepsilon_j^k\varphi + \omega_3\left(\varepsilon_j^k - \frac{m_i}{M_i}\right) & si\ \varepsilon_j^k \geq \frac{m_i}{M_i} \end{cases} \quad (38)$$

donde φ representa una constante con el valor unitario de una feromona y ω_3 es una constante de calibración del sistema. La cantidad de feromonas a depositar depende por tanto de la calificación obtenida por el alumno, así como del mínimo deseado, m_i, y de la máxima calificación posible, M_i, que pueden ser ambos definidos para cada curso.

El procedimiento utilizado se basa en la utilización de las acciones de los alumnos aplicando la meta-heurística ACO, de forma que los alumnos actúan como las hormigas utilizadas en los algoritmos ACO modificando la probabilidad a_{ij} de la ecuación (3) en cada arco a través del depósito paso a paso de Φ feromonas adicionales en cada arco recorrido. De esta forma se aporta un pequeño refuerzo a los arcos que han llevado al alumno a una calificación superior al mínimo requerido y se penaliza a aquellos arcos que no. En realidad, aunque las hemos llamado feromonas para facilitar la comprensión de su objetivo, la cantidad de refuerzo positivo o negativo añadido por la expresión (38) no son feromonas tal y como se entiende en las técnicas ACO, ni son utilizadas por una colonia de agentes (hormigas), sino que simplemente son un modificador de la función de visibilidad en la tabla de decisión de las hormigas. La ecuación para el cálculo de Φ es el nexo de unión entre el mundo real (acciones de los alumnos) y el mundo virtual (hormigas), lo que facilitará la adaptación en tiempo de ejecución y la evolución del sistema.

Cada vez que un alumno es evaluado al finalizar un curso, obtiene una calificación ε y se depositan las feromonas correspondientes según la ecuación (38). Si la tabla de probabilidad de cada nodo se basa únicamente en las feromonas depositadas en los arcos que lo tienen como destino, el algoritmo pronto quedará bloqueado en un mínimo local formado por la secuencia de nodos que garantizan una mayor calificación media, pero esto no siempre equivale a un mayor aprendizaje, sino que más bien suele corresponder con la secuencia de cursos que exige un menor esfuerzo o tiene unos test de evaluación de menor dificultad en cada momento. Para minimizar la probabilidad de un rápido bloqueo en el camino de menor esfuerzo es necesario tener en cuenta algún factor relacionado con la complejidad de un curso o con la tasa de éxito. Al igual que en [18] nuestra aproximación [82] también estudia la utilización de pesos, definidos por el equipo pedagógico que diseña el itinerario. Estos pesos se asignan a cada nodo del itinerario (Figura 13).

Además del peso aportado por el equipo pedagógico nuestra aproximación tiene en cuenta la idoneidad asociada a cada arco en las restricciones OR:

$$f_{ij} = \omega_1 W_j P_{ij} + \omega_2 \Phi_{ij} \quad (39)$$

donde ω_1 y ω_2 son constantes de calibración que permiten ajustar el sistema durante la experimentación; W_j es el peso asignado por el equipo pedagógico, P_{ij} es el factor de idoneidad del arco L_{ij} y Φ_{ij} es la suma de feromonas depositadas en él:

$$\Phi_{ij} = \sum_{k=1}^{n} \phi_{ij}^{k}(\varepsilon) \tag{40}$$

Mientras que el peso pedagógico, W_j, es definido en tiempo de diseño del itinerario, para el factor de idoneidad, P_{ij}, se propone definir el factor de idoneidad en función de la probabilidad de éxito de cada arco.

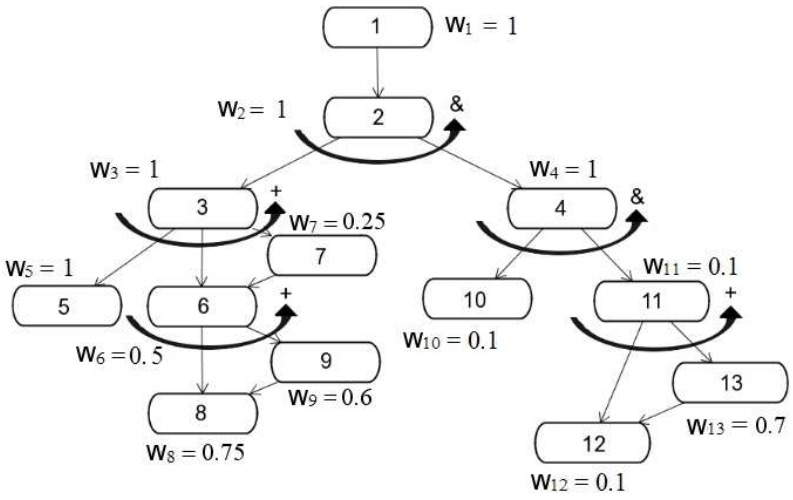


Figura 13: Nodos del GIA ponderados con pesos pedagógicos

Factor de idoneidad basado en la probabilidad de éxito

Para calcular el factor de idoneidad P_{ij} según la probabilidad de éxito se utiliza una red bayesiana. Conociendo la probabilidad de éxito de cada rama se puede recomendar una u otra rama. El teorema de Bayes puede aplicarse en el cálculo del factor de idoneidad de la siguiente forma: Sea el nodo de decisión planteado en la Figura 14, parte de un itinerario de aprendizaje del que se conoce, de ediciones anteriores, que del total de alumnos que finalizan el curso F, el 30% continúa con el curso G, el 50% lo hace por el curso H y el 20% restante por el E. También se conoce, de los datos de ediciones anteriores, la probabilidad de que un alumno que ha realizado un determinado curso supere un mínimo deseado al final del itinerario.

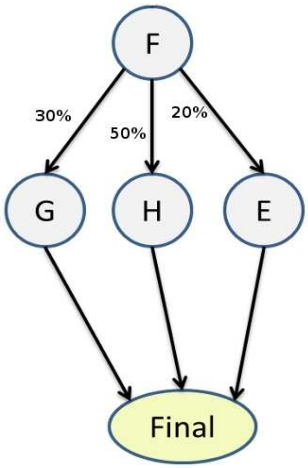


Figura 14: Ejemplo de restricción OR

Tabla 6: Probabilidad de obtener una calificación óptima en el nodo Final

Nodo	$P(\varepsilon_{final} \geq \frac{m}{M})$
G	0,8
H	0,6
E	0,5

Con estos datos podemos calcular:

a) la probabilidad, *P(Q)* de que un alumno obtenga una calificación superior al mínimo deseado,

b) la probabilidad de que un alumno que ha obtenido una calificación superior al mínimo deseado, haya elegido el curso *G* en la restricción OR de la Figura 14.

Teniendo en cuenta las probabilidades conjuntas, la probabilidad de que un alumno obtenga una calificación superior al mínimo, *P(Q)*, habiendo realizado el curso *G*, es $P(G \cap Q)=P(G)\cdot P(Q|G)$. Esta probabilidad, y de la misma forma la del resto de nodos de la restricción pueden calcularse con la creación del árbol de probabilidades conjuntas.

La probabilidad de obtener una calificación superior al mínimo en el nodo Final de la Figura 14 es (Teorema de la Probabilidad Total):

$$P(Q) = \sum_{i=1}^{3} P(Q|H_i)\, P(H_i) \tag{41}$$

que para el ejemplo resulta,

$$P(Q) = P(G \cap Q) + P(H \cap Q) + P(E \cap Q) = 0{,}24 + 0{,}30 + 0{,}10 = 0{,}64$$

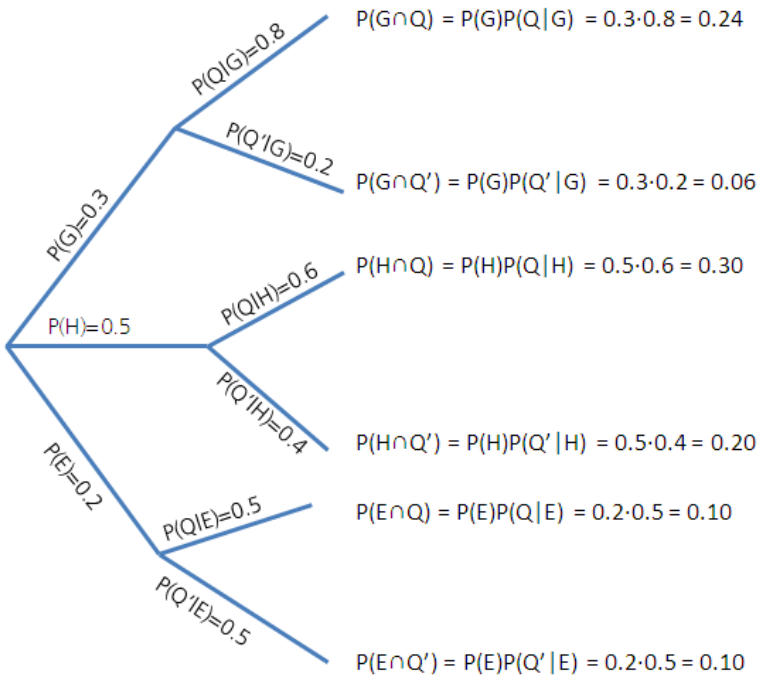


Figura 15: Árbol de probabilidades conjuntas

Para la segunda cuestión aplicamos el teorema de Bayes:

$$P(G|Q) = \frac{P(G \cap Q)}{P(Q)} = \frac{0{,}24}{0{,}64} = 0{,}375$$

que indica que un alumno que haya obtenido una calificación superior al mínimo deseado tiene un 37,5% de probabilidad de haber realizado el curso G, siguiendo el arco L_{FG}. De la misma forma, siguiendo el árbol de probabilidades conjuntas (Figura 15) puede calcularse para el resto de nodos de la restricción.

Si suponemos que la calificación que se obtenga será superior al mínimo deseado, pueden calcularse las probabilidades para cada nodo de una restricción y usarse para recomendar el mejor camino, haciendo que esta probabilidad basada en los datos históricos pueda influir en la función de ajuste de cada arco (39). De esta forma se utiliza P(G|Q) en la ecuación (39) como P_{FG}, P(H|Q) como P_{FH} y P(E|Q) como P_{FE}. Además ofrece una prestación interesante, pues las probabilidades pueden ser calculadas on-line por el LMS cada vez que un alumno finaliza el itinerario, lo que ayudaría al sistema a evolucionar por sí sólo adaptándose a las nuevas poblaciones de alumnos.

Este mecanismo es similar al empleado en los filtros anti-spam, y se emplea también para el diagnóstico de fallos estocásticos en e-learning [83] y para la elección de un curso que refuerce ciertos conocimientos que carecen de un andamiaje lo suficientemente robusto. Una aproximación similar a la nuestra se describe en [84], dónde se usa una red bayesiana para combinar la estructura de contenidos con el perfil del usuario y su estilo de aprendizaje, de forma que el sistema pueda recomendar direcciones alternativas. Otros autores [85] han aplicado también las redes bayesianas a la educación para modelar las necesidades de los alumnos, su estado y estilo de aprendizaje con el objetivo de personalizar el contenido educativo que se distribuirá a cada alumno.

En una primera aproximación [82] la probabilidad de obtener una determinada calificación se clasificó en tres intervalos, High ($0.7\text{M} \leq \varepsilon\text{M} \leq \text{M}$), Medium ($0.5\text{M} \leq \varepsilon\text{M} < 0.7M$) y Low ($0 \leq \varepsilon\text{M} < 0.5M$), definiendo la función de idoneidad de cada arco L_{ij} como:

$$P_{ij} = \begin{cases} P(J|Q_{HIGH}) + P(J|Q_{MEDIUM}) & si\ 0.7M \leq \varepsilon_i \leq 1 \\ 1/N_i & en\ otro\ caso \end{cases} \quad (42)$$

La ecuación (42) asigna una probabilidad a cada arco en función de la calificación obtenida en el nodo inicial. Si esta calificación se encuentra en un determinado intervalo, se supone que la probabilidad de superar el mínimo deseado en el nodo siguiente será lo suficientemente alta como para anticipar esa evidencia. A partir de ahí, aplicando el Teorema de Bayes, se asigna al arco L_{ij} la suma de las probabilidades de obtener una calificación en el intervalo High o en el Medium, o lo que es lo mismo, la probabilidad de obtener una calificación superior a la cota inferior del intervalo Medium. En caso de no haber obtenido el alumno una calificación adecuada en el nodo inicio de la restricción OR, se asignará a todos los arcos salientes de *i* la misma probabilidad, $1/N_i$, dónde N_i es el grado de salida del nodo *i*.

Esta función de idoneidad modifica el impacto introducido por el peso pedagógico definido para cada nodo, lo que hace que la influencia dependa no sólo de la complejidad definida a priori por el equipo pedagógico sino también del rendimiento que los alumnos han obtenido de este curso. Esta combinación es adecuada puesto que el empleo de únicamente el peso pedagógico haría la función de ajuste tendente rápidamente a seleccionar los cursos de mayor peso al principio, y únicamente se produciría un lento cambio en la ruta de aprendizaje tras una gran cantidad de bajas calificaciones.

El factor de idoneidad se va modificando durante el curso a medida que los alumnos van navegando a través del grafo. Para ello es necesario tener en cuenta algunas consideraciones en la definición del modelo de datos que represente los nodos y arcos del grafo, y que se representan gráficamente en la Figura 16:

- Al iniciarse un curso por primera vez, el factor de idoneidad de cada camino L_{ij} será $1/N_i$, dónde N_i representa el número de vecinos del nodo i.
- Cada nodo incluirá un contador que reflejará el número de alumnos que lo han recibido y que permitirá calcular la calificación media en dicho nodo.
- Cada nodo almacenará en una estructura de tipo clave-valor (ejemplo una tabla Hash) la nota media obtenida en el curso para los alumnos procedentes por cada uno de los arcos de entrada al nodo. Así, para un nodo que tiene tres arcos de entrada, esta estructura contendrá tres entradas cada una asociada a un arco, y contabilizará la nota media en el curso para los alumnos que hayan llegado por cada arco. Este punto obliga a mantener el arco seguido por el alumno durante la realización del curso de ese nodo.
- También se almacenará en la estructura anterior, por cada arco de entrada, el número de alumnos que han llegado al curso por dicho arco y el número de ellos que han obtenido una calificación superior a la mínima exigida por el nodo de dicho curso.

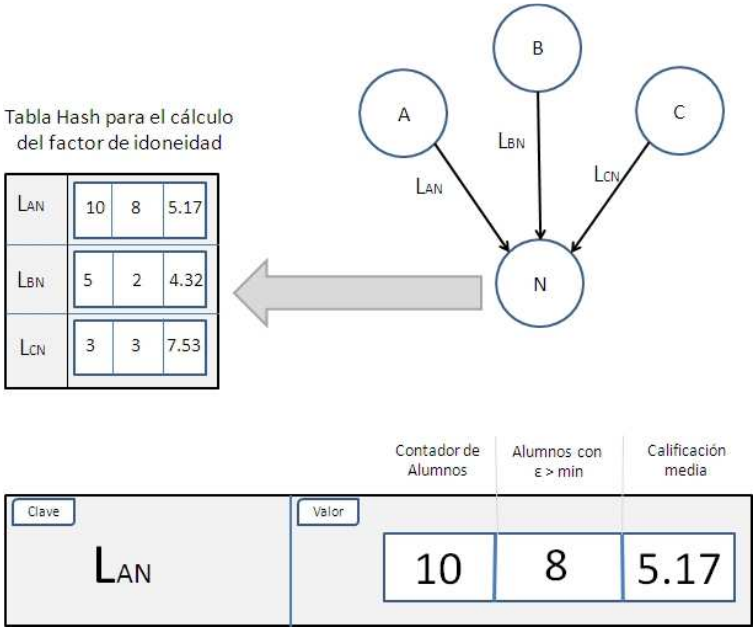


Figura 16: Ejemplo de tabla hash con información a incluir en el nodo destino para calcular el factor de idoneidad de cada arco de entrada

Con estas consideraciones se facilita el cálculo de las probabilidades de cada nodo durante la ejecución del algoritmo.

Como se vio en el Capítulo 2, la tabla de decisión de un hormiga, $A_i[a_{ij}(t)]$, en el nodo *i* se construye mediante la composición de los rastros locales de feromonas con los valores heurísticos (3). En nuestra aproximación la tabla de decisión de una hormiga se construye con los valores de la función de ajuste, normalizados al intervalo [0,1]:

$$a_{ij}(t) = \frac{[\tau_{ij}]^{\alpha}[f_{ij}(t)]^{\beta}}{\sum_{l \in N_i}[\tau_{il}]^{\alpha}[f_{il}(t)]^{\beta}} \quad \forall j \in N_i \tag{43}$$

La función de ajuste f_{ij} representa la visibilidad de cada nodo, siendo directamente proporcional a la idoneidad de selección del nodo. La función de ajuste es la que define el concepto de distancia para el problema de las rutas de aprendizaje adaptativas en sistemas e-learning. Las variables α y β son las variables de calibración descritas en el capítulo 2, que sirven para controlar la tendencia del sistema a la exploración de nuevos caminos o a la explotación de los óptimos locales. Esta definición de tabla de decisión nos permite utilizar dos tipos de hormigas diferentes, que depositan dos clases de feromonas diferentes. Las feromonas de la ecuación (38), Φ_{ij}, son depositadas en función de las acciones de los alumnos tras finalizar la evaluación en cada nodo. En otras palabras, podemos decir que son los propios alumnos los que toman el rol de las hormigas de la meta-heurística ACO. Estas feromonas sirven para influir en la distancia entre un curso y el siguiente, modificando la función de ajuste que toma el papel de la visibilidad del arco, η_{ij}, del algoritmo AS. En cambio, las feromonas τ_{ij} son depositadas al finalizar el itinerario completo en los arcos que pertenecen al itinerario, reforzándose además en aquellos arcos que pertenecen a la mejor solución. Estas feromonas servirán de refuerzo a las próximas ediciones de la misma promoción (una nueva convocatoria para realizar los cursos que doten de las competencias necesarias).

La probabilidad con la que se selecciona cada nodo se define en la ecuación (4). El siguiente nodo (cursos) en la secuencia de cursos que conforman el itinerario se selecciona aleatoriamente teniendo en cuenta la probabilidad de cada arco. Para implementar este mecanismo de selección se utiliza el algoritmo del Listado 17.

Para evitar a largo plazo un excesivo impacto de las feromonas en la función de ajuste f_{ij}, las feromonas sufren el proceso de evaporación igual que en la vida real. En

una primera aproximación [82] se estudió la evaporación paso a paso de las feromonas, pero al considerar a los alumnos como hormigas en la técnica ACO propuesta se dificulta la experimentación y se pierde una de las principales características de ACO: la exploración de caminos por las hormigas de una colonia. Con alumnos reales no es posible esta opción, pues no es aceptable guiar a determinados alumnos por caminos

```
L ← Lista de elementos vacía
tam = 0;
Para cada arco Lij de una restricción OR
  n ← aij*100;
  Desde k=0 hasta n
   añadir Lij a L;
   tam = tam + 1;
  finDesde
finPara
r ← generar número aleatorio entre 1 y tam
seleccionar arco en la posición r-ésima de L
```

Listado 17: Algoritmo de selección aleatoria con probabilidad determinada de un arco

excesivamente fáciles (en términos de nivel de exigencia requerido) o tediosos (en términos de números de cursos y/o horas necesarias de dedicación).

Aunque en un primer experimento los resultados fueron interesantes [82], no son significativos, por lo que se ha planteado una política de evaporación ligeramente diferente, definida por la siguiente ecuación:

$$\Phi_{ij}(t+1) = (1-\rho)\Phi_{ij}(t) + \theta_{final}(\varepsilon) \tag{44}$$

donde $\Phi_{ij}(t+1)$ representa la cantidad de feromonas existentes en el arco l_{ij} en el instante *t+1* y ρ es la tasa de evaporación. Mientras que la deposición de las feromonas por el recorrido de los alumnos se realiza paso a paso, la evaporación de las feromonas depositadas por éstos se lleva a cabo una vez que el grupo de alumnos ha completado el itinerario. El término $\theta_{final}(\varepsilon)$ representa la variación de feromonas dependientes de la calificación final obtenida por el alumno *k*, y se deposita sólo en los arcos del recorrido seguido, una vez completado el recorrido.

$$\theta_{ij}(\varepsilon) = \begin{cases} \Phi_{final}(\varepsilon) & si\ l_{ij} \in R^k \\ 0 & en\ otro\ caso \end{cases} \tag{45}$$

donde $\Phi_{final}(\varepsilon)$ se calcula como sigue:

$$\Phi_{final}(\varepsilon) = \begin{cases} -(1-\varepsilon_{final})\varphi - \omega_3\left(\frac{m_{final}}{M} - \varepsilon_{final}\right) & , si\ \varepsilon_{final} < \frac{m_{final}}{M} \\ \varepsilon_{final}\varphi + \omega_3\left(\varepsilon_{final} - \frac{m_{final}}{M}\right) & , si\ \varepsilon_{final} \geq \frac{m_{final}}{M} \end{cases} \tag{46}$$

La actualización y evaporación de las feromonas τ_{ij}, depositadas por las hormigas artificiales, se rigen por las ecuaciones del algoritmo AS (5), (6) y (7), teniendo en cuenta que en la ecuación (7), la distancia entre dos nodos es el valor de la función de ajuste para el arco que los une, y por tanto la distancia total del recorrido seguido por el alumno es la suma de los valores de la función de ajuste para los arcos del recorrido. El refuerzo se divide entre el número de nodos o vértices que tiene el recorrido para potenciar también los recorridos más cortos.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{f_{ij}^{k}(t)}{num\_vertices(R^{k}(t))} & si\ l_{ij} \in R^{k}(t) \\ 0 & si\ l_{ij} \notin R^{k}(t) \end{cases} \quad (47)$$

Esta distancia, al estar sujeta a actualizaciones paso a paso del resto de alumnos no es constante, sino que evoluciona y varía con el uso del sistema, por lo que se calcula cada vez que se ejecuta el proceso de evaporación de estas feromonas. Nótese que en la ecuación (39), la cantidad de feromonas, Φ_{ij}, existentes en el arco l_{ij} en el momento de calcularse la función de ajuste, es debida a otros alumnos que han escogido el mismo arco y han depositado feromonas en el nodo *j* pero no al propio alumno cuya decisión del siguiente curso se esté evaluando. Por tanto, el valor en ese momento de Φ_{ij} debe existir en cada arco, por lo que además de las feromonas τ_{ij} debe existir también este otro tipo de feromonas Φ_{ij} como atributos de los elementos de un grafo. Esto también es un proceso de influencia *estigmérgica*, en el que un conjunto de miembros de un colectivo está influenciando a otro igual en una decisión, modificando el entorno (la función de ajuste en este caso). Esto es lo que se pretendía desde un principio, encontrar una forma de influir de un grupo de individuos en otro, obteniendo algún beneficio para todo el grupo.

Las feromonas Φ_{ij} son depositadas por las acciones llevadas a cabo por los alumnos reales (humanos) mientras que las tradicionales feromonas τ_{ij} son las calculadas mediante un algoritmo ACO para la búsqueda de una solución óptima. Las primeras sirven para modificar la función de ajuste de cada arco, de forma que el valor heurístico "distancia" entre dos cursos varíe en función de los resultados obtenidos por los alumnos. Las segundas son creadas por el sistema gestor de aprendizaje en el proceso de búsqueda del mejor camino.

3.3.5 Implementación

El objetivo de esta sección no es documentar el proyecto software sino simplemente las decisiones más importantes en cuanto a diseño del framework que posteriormente se ha utilizado para la experimentación.

El lenguaje utilizado ha sido Java en su versión J2SE 1.6. Tras analizar varias librerías para el manejo de grafos (JGraphT[18], JGraph[19] y JUNG[20]) nos decidimos por la librería de código abierto JUNG por su madurez, por tener soporte para múltiples tipos de grafos (multigrafos, grafos dirigidos, anidados), gestores de presentación (para el color, disposición en pantalla, formas, etc), ser capaz de almacenar los grafos en diferentes formatos (entre ellos GraphML que se adapta a nuestros requisitos), la buena documentación que ofrece y los extras que la diferencian de otras librerías: análisis estadístico, algoritmos para el cálculo de distancias, clustering, etc.

Una vez decidida la librería gráfica a utilizar, el siguiente paso es definir los objetos necesarios en el dominio de nuestro problema, que en nuestro caso se resumen en la definición del nodo, arco, hormiga, colonia de hormigas, feromonas, rastro de feromonas, condiciones de parada y el planificador o demonio de ejecución de los algoritmos ACO.

Elementos de un grafo: nodos y arcos

La librería JUNG define la jerarquía de interfaces según la Figura 17. La base de toda la jerarquía de grafos de esta librería es la interfaz `HyperGraph`, de la que hereda la interfaz `Graph` (que define la funcionalidad básica de un grafo) y de ésta a su vez, las interfaces `DirectedGraph` (que define la funcionalidad de un grafo dirigido) y `UndirectedGraph` (que define la funcionalidad de un grafo no dirigido). Lo primero que observamos en esta librería es que la interfaz `Graph` está parametrizada con dos tipos genéricos, `V` y `E`, correspondientes a los vértices y arcos del grafo respectivamente, por lo que nos exige definir en nuestro modelo estos tipos. Para facilitar la separación de la funcionalidad necesaria para la visualización se define ésta en la interfaz `GraphMember` que representa a cualquier elemento miembro de un grafo dibujable en un plano (nombre del elemento y coordenadas `x` e `y` que determinen su

[18] http://www.jgrapht.org/
[19] http://www.jgraph.com
[20] The Java Universal Network/Graph Framework. http://jung.sourceforge.net/

posición). La clase `GraphElement` es la clase base que implementa la interfaz `GraphMember` y de ella heredan nuestros modelos de vértice y nodo, definidos respectivamente en las clases `Vertex` y `Edge` (Figura 18).

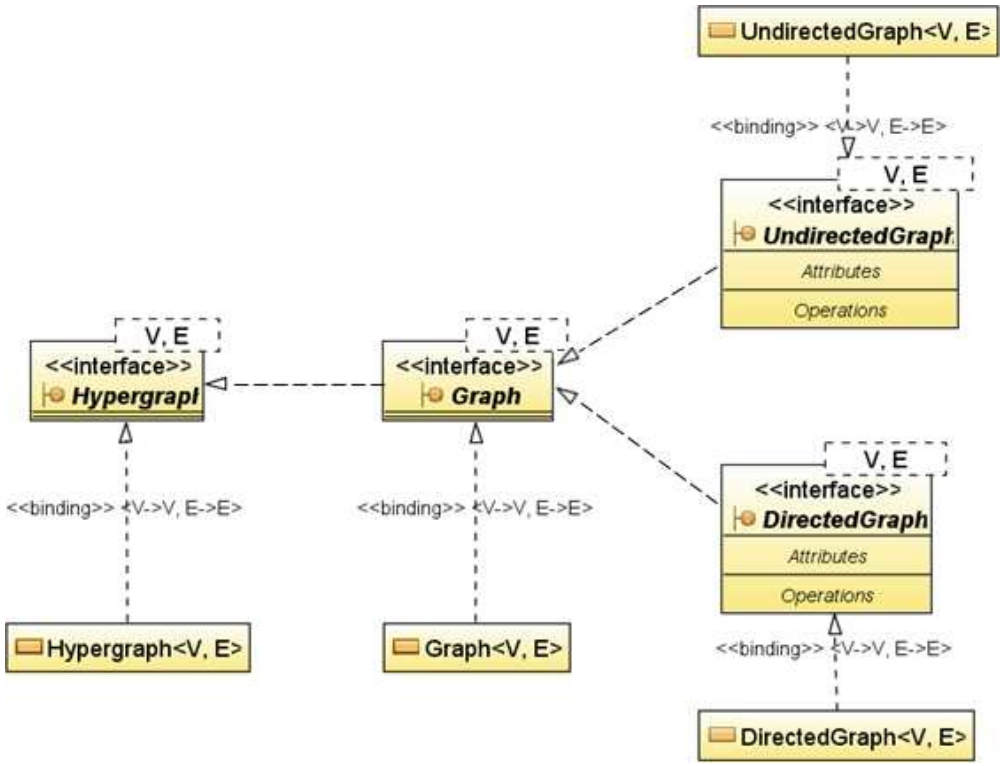


Figura 17: Jerarquía de Interfaces para el manejo de grafos de JUNG

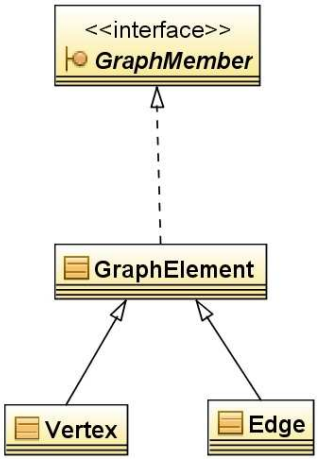


Figura 18: Jerarquía de nuestro modelo para la definición de vértices y arcos

La clase `Vertex`, que representa un nodo del grafo, ha de almacenar entre sus atributos la ubicación del curso (URL en la que se encuentra accesible), el valor de la

calificación mínima *m* requerida para el curso y el peso pedagógico *W* asignado por el equipo de expertos que diseñó el itinerario formativo. El nombre es heredado de la clase `GraphElement`, clase que también proporciona un atributo que indica el número de feromonas depositadas sobre el elemento del grafo (de forma que se pueda aplicar ACO con depósito de feromonas tanto sobre los nodos como sobre los arcos). Otros atributos de utilidad para el nodo de nuestra solución son: número de alumnos que han finalizado el curso de un nodo, calificación media obtenida por los alumnos que han realizado la evaluación final de un curso, y una serie de contadores que serán de utilidad a la hora de calcular las probabilidades de obtener una calificación en un rango determinado (número de alumnos con una calificación baja, media o alta, según los intervalos definidos) dividiendo el valor de cada contador entre el total de alumnos que finalizaron el curso.

En cuanto a la clase `Edge` que modela un arco del grafo, el atributo más importante que debe almacenar es la distancia existente entre los dos nodos que el arco conecta, que siguiendo el concepto de distancia establecido se corresponde con la función de ajuste (39).

Framework ACO

Destacaremos del framework ACO desarrollado para la experimentación las clases que modelan una hormiga, una colonia de hormiga, el entorno, las feromonas, los recorridos de cada hormiga, las condiciones de parada y el planificador de la ejecución del algoritmo ACO (Figura 19). El framework utilizado es una evolución del desarrollado por Fernando Otero [122], al que se ha adaptado para trabajar con colonias compuestas de hormigas con diferente comportamiento y para soportar la generación de rastros de feromona tanto en arcos como en vértices.

La clase `Scheduler` es el planificador encargado de la ejecución de la meta-heurística java (Listado 1) actuando sobre un entorno determinado (`Environment`). La Figura 20 muestra cómo el planificador necesita que le sea proporcionada una función objetivo (definida por la interfaz `ObjectiveFunction`) para evaluar los rastros de feromonas; una función encargada de realizar la búsqueda local (interfaz `LocalSearch`); una entidad encarga de realizar la actualización de las feromonas (interfaz `PheromoneUdapter`), puesto que en cada aproximación de ACO puede hacerse de una forma diferente; otra entidad que implemente la condición de parada para la ejecución del planificador (interfaz `StopCondition`); y por último, es necesaria una entidad responsable de la ejecución de las tareas a realizar por el

planificador como último paso de la meta-heurística ACO (implementación de la interfaz `DaemonAction`).

En el modelado de la hormiga virtual, implementada por la clase `Ant` de nuestro framework, tiene especial relevancia la necesidad de contar con memoria, de forma que pueda seleccionarse el mejor recorrido (el de menor coste). Para ello se utiliza como atributo una tabla con los recorridos realizados por la hormiga en cada iteración (Figura 20). Dicha tabla se inicializa con el número de iteraciones a realizar por cada hormiga, dato que debe ser proporcionado por el algoritmo planificador. La inserción de cada nuevo recorrido (implementado por la clase `Trail`) se hace de forma ordenada en la memoria, de forma que el primer elemento de la tabla siempre es el de menor coste.

En nuestro framework, la implementación de un recorrido se realiza como lista de elementos de un grafo (`GraphElement`) lo que nos permite no sólo tener recorridos expresados cómo lista de vértices o cómo lista de arcos, sino también como lista de vértices y arcos mezclados. Será la implementación del mecanismo de construcción de recorridos el que deberá encargarse de la opción que más se ajuste al algoritmo ACO a usar.

En la implementación se ha optado por el modelo de deposición retardada de feromonas, a cargo del planificador y al final de cada iteración. La implementación de esta opción es mucho más sencilla que la deposición paso a paso, y puede delegarse a cada colonia la construcción del recorrido (cada colonia tendría su propio mecanismo de deposición). Utilizando una factoría[21] se facilita la construcción de un recorrido (Listado 18) en la clase `Ant`.

```
public Trail walk() {
     setTrail(colony.getTrailFactory().createTrail(type));
     return trail;
}
```

Listado 18: Creación de un recorrido de una hormiga en la clase Ant

[21] Factory pattern, del libro Design Patterns, de Erich Gamma et al. [86]

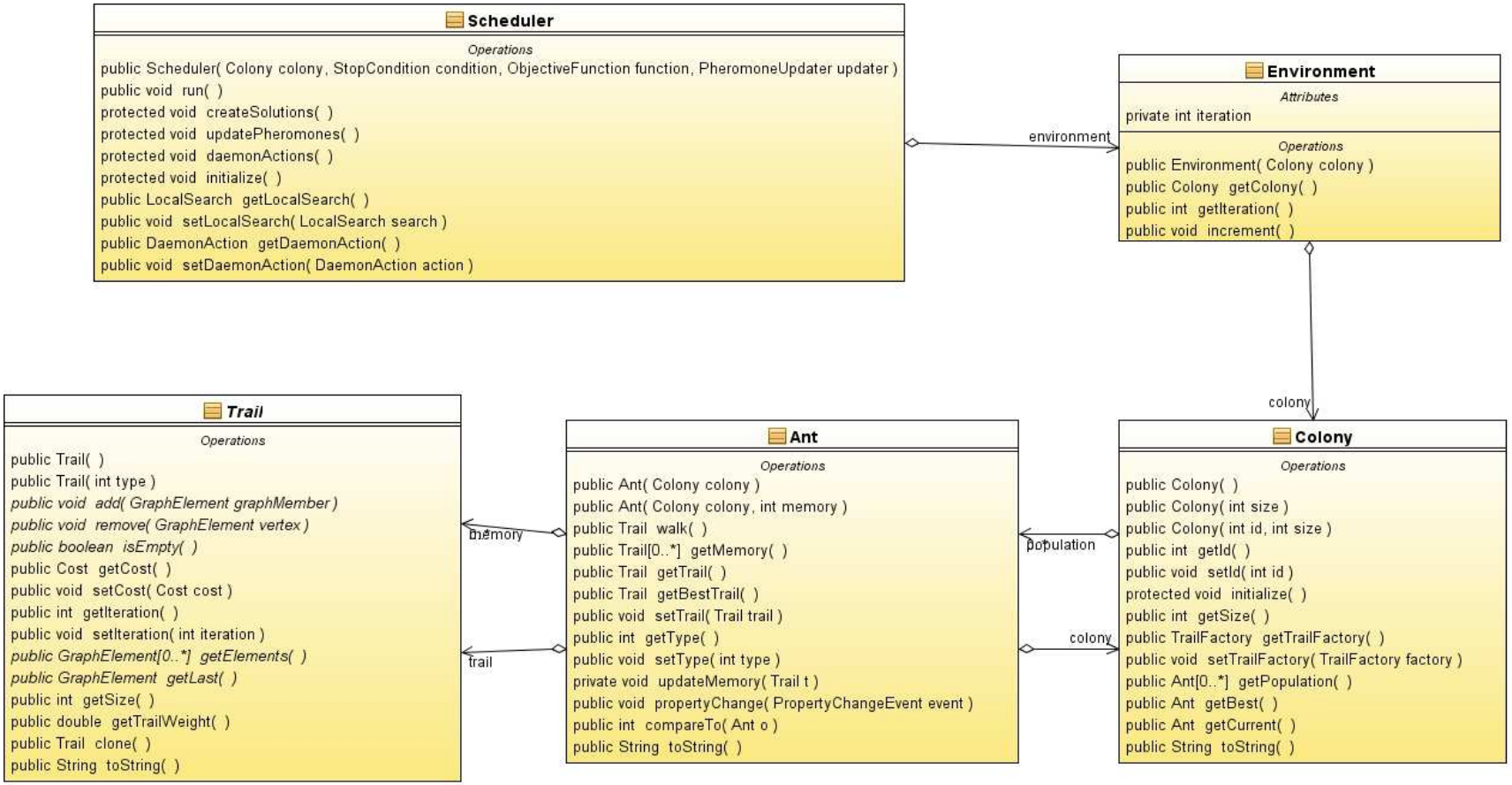


Figura 19: Principales entidades del framework ACO

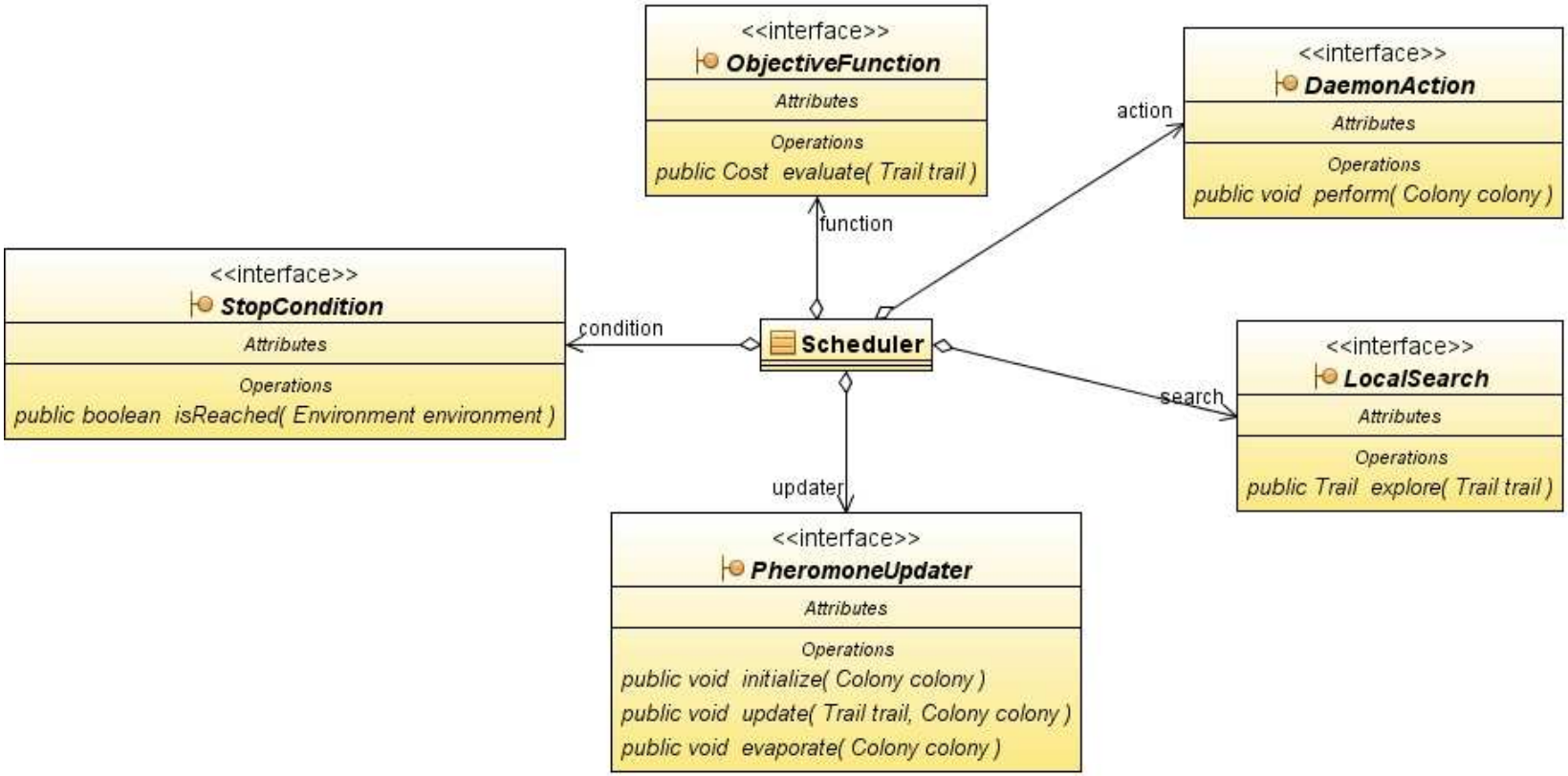


Figura 20: Relaciones del planificador con otras interfaces del framework

Para la creación del recorrido se proporciona un argumento `type`, que informa del tipo de hormiga, para así poder utilizar aproximaciones que tengan en cuenta diferentes tipos de hormigas dentro de una misma colonia (hormigas exploradoras, voladoras…), cómo se emplea por ejemplo en el algoritmo AntNet-FA, para que así la factoría pueda crear el recorrido aplicando una estrategia u otra. Cobra especial importancia en la implementación de cada algoritmo ACO, la implementación que se proporcione a la colonia de hormigas de la fábrica de creación de recorridos.

Para la implementación de la colonia de hormigas, en nuestro caso, la clase `Colony` sólo necesita conocer el número de hormigas que componen la colonia, la factoría encargada de la creación de recorridos para cada hormiga y un método que devuelva la mejor hormiga de la colonia, que será aquella que haya obtenido el recorrido de mejor coste, para lo que se implementa un comparador a tal efecto, que permita comparar objetos `Ant` entre sí (Listado 19).

```
public Ant getBest(){
Ant best = null;

     for (Ant ant : getPopulation()) {
          if ((best == null) || (ant.compareTo(best) == 1)) {
               best = ant;
          }
     }
     return best;
}
```

Listado 19: Método para obtener la mejor hormiga de una colonia.

El entorno (clase `Environment`) de la Figura 19 sólo contempla el uso de una colonia de hormigas, que es suficiente para nuestra aproximación. Para aproximaciones con más de una colonia habría que modificar la clase para poder gestionar más de una colonia y las operaciones derivadas de ella.

Las feromonas son una especie de peso que puede añadirse tanto a un arco como a un nodo del grafo con un valor $p \in \mathbb{R}$. Para permitir elementos de un grafo que no tienen feromonas porque no se han recorrido nunca (nótese la diferencia con aquellos

elementos del grafo que tengan un nivel de feromonas *p=0*) y evitar un error al leer o escribir el grafo (*null* = NaN[22]) se implementa la clase `Weight`, como un envoltorio[23] de los números reales apropiado para cualquier tipo de peso del grafo, y de la que hereda la clase `Pheromone.`

La condición de parada para el planificador se especifica a través de la interfaz `StopCondition` y ha de ser proporcionada por cada algoritmo ACO que use nuestro framework, si bien se proporcionan en el framework una sencilla implementación basada en el número de iteraciones finalizadas, que hará que el planificador detenga la ejecución una vez alcanzado el número máximo de iteraciones establecida.

Aplicación de ACO a la construcción de itinerarios de aprendizaje adaptativos

Nuestra aproximación se basa en la aplicación de técnicas ACO en e-learning con el objetivo de conseguir un sistema capaz de evolucionar los itinerarios de aprendizaje según las necesidades de cada momento. El procedimiento consiste en simular el comportamiento de un conjunto de alumnos participantes en un itinerario formativo para intentar proponer el mejor camino. En este trabajo tan sólo se tiene en cuenta la calificación obtenida en los cursos como criterio de la bondad de un camino, aunque ésta puede depender de múltiples factores, que quedan fuera del alcance de este trabajo, como la mejor adaptación de un arco frente al resto (en un nodo con grado de salida mayor que 1) debido al tipo de aprendizaje más adecuado en cada alumno (personalización del proceso de aprendizaje: deductivo, inductivo, etc.).

Debido a que la distancia entre dos cursos (nodos en el grafo de itinerarios de aprendizaje) depende, entre otros factores, de la calificación que los alumnos obtengan en el nodo destino, la ejecución de un algoritmo ACO clásico en el que una colonia de hormigas busca la solución óptima a lo largo de un número determinado de iteraciones no es aplicable, puesto que aunque podríamos simular las calificaciones mediante generación de números aleatorios, esto no nos garantiza que se correspondan con las calificaciones de los alumnos reales que participan de la formación. La aplicación de ACO a este problema, requiere que los propios alumnos asuman el rol de hormigas del algoritmo ACO empleado, depositando las feromonas correspondientes en función de la

[22] Not a Number.
[23] Patrón de diseño Wrapper, del libro Design Patterns, de Erich Gamma et al. [86]

calificación obtenida en el arco del grafo seguido. Por tanto no existen hormigas virtuales, sino que este rol es desempeñado por los alumnos reales. De la misma forma, el número de iteraciones que conduce a una colonia de hormigas a encontrar una solución lo suficientemente buena, pasa a convertirse en nuestro problema en el número de promociones que participarán en la formación, pudiendo este número de promociones ser infinito, puesto que lo que realmente interesa es la capacidad de adaptación (de variación) a los cambios por parte del algoritmo y no elegir una mejor solución tras un número predeterminado de iteraciones.

Esta adaptación, se medirá mediante la observación de los resultados obtenidos mediante simulación, analizando la sensibilidad del algoritmo para reforzar los mejores caminos de promociones anteriores, en mayor o menor grado, así como midiendo la distribución del reparto (en términos porcentuales) de alumnos por cada itinerario durante las sucesivas promociones.

En nuestra aproximación, al desempeñar los alumnos el rol de hormigas virtuales, no existe una colonia de hormigas como tal, puesto que, los miembros de la colonia no repiten promociones, sino que se renuevan en cada promoción.

Para la implementación, se tendrán en cuenta los siguientes puntos para la experimentación:

1. El punto de partida es único en el itinerario formativo diseñado por un equipo pedagógico, teniendo en cuenta itinerarios alternativos de aprendizaje (cursos de refuerzo, de ampliación, etc.). Si se cuenta con datos históricos para el mismo grafo de itinerarios de aprendizaje (no es la primera edición del itinerario formativo) se utilizarán para el cálculo del factor de idoneidad P_{ij} aplicando la ecuación (42). Puesto que en un instante inicial no se dispone de datos históricos para poder calcular el factor de idoneidad P_{ij} (42) de la función de ajuste, se tendrá en cuenta que:
    a) En los nodos con grado de salida 1, el factor de idoneidad de su arco de salida es 1.
    b) En los nodos con grado de salida n>1, el factor de idoneidad de cada arco es 1/n.
2. Cada vez que un alumno es evaluado al finalizar el curso de un nodo *j*, se producirá una realimentación Φ_{ij} que modificará la visibilidad del arco l_{ij} seguido para llegar a *j*, de acuerdo con la ecuación (38).

3. En los nodos con grado de salida n>1, la elección del siguiente curso se llevará a cabo probabilísticamente en función de las feromonas τ_{ij} depositadas en los *n* arcos de salida.
4. Nuestra aproximación permite la adición de nodos en cualquier momento. Cuando se añada un nuevo nodo, su factor de idoneidad P_{ij} será 1 y Φ_0=0, por lo que la visibilidad inicial de un nodo será un factor proporcional de su complejidad (indicada por su peso pedagógico): $\omega_2 \cdot W_j$
5. El número de feromonas en el instante inicial es τ_{ij}=0.1 para cada arco, para evitar una indeterminación en los valores a_{ij} de la tabla de decisión.
6. El modelo empleado para representar los itinerarios de aprendizaje permitirá contabilizar la cantidad de feromonas depositadas en los arcos y nodos del grafo en cualquier instante.

Algoritmo ASALI

A continuación, mostramos el algoritmo ASALI (Ant System for Adaptive Learning Itineraries) para hacer posible la adaptación del itinerario formativo de los alumnos según las necesidades del colectivo, basándonos en el procedimiento descrito en este trabajo. Se han eliminado las instrucciones relacionadas con la obtención de datos de la interfaz gráfica y tratamiento de estructuras de datos, que no son de gran interés para el estudio de aplicación de ACO en el contexto descrito.

La ejecución del algoritmo se inicia con la ejecución de una instancia del hilo (`Thread`) `ASALIOptimizer`, cuyo método `run` se resume en el Listado 20. Este método consiste en comprobar que existe un grafo para el que se puede calcular el mejor camino, iniciando en este caso, mediante una técnica de inmersión, el procedimiento de inicialización con los parámetros para el algoritmo y el inicio del planificador encargado del cálculo del mejor camino. Todo este proceso se realiza en la llamada al método `evolve()`que devuelve el mejor camino encontrado a través del objeto de retorno `Trail` (Listado 21).

```
public void run() {
     if (graph == null || graph.getVertexCount() == 0) {
          frame.noGraphWarning(); // WARNING: No hay grafo
     }else {
          // Obtener el mejor camino
          this.trail = evolve();
          // Calcular el coste del camino
          Collection<GraphElement> elements = trail.getElements();
          Iterator<GraphElement> iterator = elements.iterator();
          double cost = 0.0;
          while (iterator.hasNext()) {
               cost = cost + iterator.next().getWeight();
          }
          // Luego este coste puede mostrarse por pantalla.
     }
}
```

Listado 20: Inicio del algoritmo

La clase `Scheduler` es la que implementa la meta-heurística ACO. Esta clase pertenece al framework desarrollado para trabajar con algoritmos ACO y es la misma para cualquier implementación de ACO. A continuación se describe como volver a traspasar la responsabilidad a las clases concretas de cada algoritmo.

En la construcción del planificador, además de establecer como atributos los parámetros que se pasan en el constructor como la colonia de hormigas, la condición de finalización, la función objetivo y el actualizador de feromonas (Listado 22), se crea el entorno en el que la colonia actuará. La ejecución del algoritmo es llevada a cabo por el planificador en un hilo de ejecución independiente, tras la invocación al método `run` del planificador (Listado 23).

El proceso de inicialización consiste en el establecimiento de un valor inicial para las feromonas de cada arco del grafo. En nuestro caso lo que se hace es establecer como valor inicial la inversa del número de vértices que tiene el grafo, al igual que en al algoritmo ACS [27]. Posteriormente, mientras no se haya alcanzado la condición de parada, que en nuestro caso se corresponde con un número de iteraciones determinado, se procede secuencialmente a la creación de soluciones, la actualización de feromonas y la ejecución de acciones por parte del planificador

```
public Trail evolve() {

 // Crear factoria
 AsaliTrailFactory factory = new AsaliTrailFactory(this.graph);
 // Añadimos los valores de ω, α y β desde la interfaz gráfica (no se muestra)

 // Crear colonia de hormigas con niveles de rendimiento h m y l
 LearnersColony colony = new LearnersColony(this.csize, h, m, l);
 colony.setTrailFactory(factory);

 // La condición de parada compuesta por una o más condiciones
 CompoundStopCondition condition = new CompoundStopCondition();
 // Una de las condiciones es el número de iteraciones
 IterationTest itest = new IterationTest(this.iterations);
 condition.add(itest);

 // Función objetivo
 DefaultObjectiveFunction function = new DefaultObjectiveFunction();

 // Acción a ejecutar por el demonio
 AsaliDaemonAction daemonAction = new AsaliDaemonAction("asali");
 // Actualizador de feromonas
 DefaultPheromoneUpdater updater = new DefaultPheromoneUpdater();

 // Creamos el planificador (demonio)
 Scheduler sc = new Scheduler(colony, condition, function, updater);

 // No establecemos ningún mecanismo de búsqueda local
 sc.setLocalSearch(new NoLocalSearch());
 sc.setDaemonAction(daemonAction);

 // Iniciamos la ejecución
 sc.run();

 // Devolvemos el mejor camino
 return colony.getBest().getBestTrail();
}
```

Listado 21: Procedimiento de inicio

.

```
public Scheduler(Colony colony, StopCondition condition,
               ObjectiveFunction function, PheromoneUpdater updater)
{
     this.condition = condition;
     this.function = function;
     this.updater = updater;

     environment = new Environment(colony);

}
```

Listado 22: Constructor del planificador

```
public void run(){
     initialize();

     while (!condition.isReached(environment))  {
          environment.increment();
          createSolutions();
          updatePheromones();
          daemonActions();
     }
}
```

Listado 23: Ejecución del planificador. Método Scheduler::run.

Creación de Soluciones

La creación de soluciones es tarea de la factoría encargada de crear caminos, pues una solución no es sino un camino en el grafo. Para ello el planificador delega en cada hormiga, para que cree su camino y éstas a su vez en la factoría de la colonia (Listado 25).

```
protected void createSolutions() {
 for (Ant ant : environment.getColony().getPopulation())
 {
  Trail trail = ant.walk();           // La hormiga camina
  trail = search.explore(trail);        // mejorar con búsqueda local
  ant.setTrail(trail);             // fijar mejor camino
  trail.setCost(function.evaluate(trail));   // fijar coste
  trail.setIteration(environment.getIteration()); // fijar iteración
 }
}
```

Listado 24: Método de creación de soluciones del planificador

A la hora de crear un camino, la hormiga indica a la factoría qué tipo de hormiga es a través del parámetro `type` (Listado 25).

```
public Trail walk(){
     setTrail(colony.getTrailFactory().createTrail(type));
     return trail;
}
```

Listado 25: La hormiga crea su camino apoyándose en la factoría de la colonia

El algoritmo de creación de un camino por la factoría se describe a continuación (Listado 26).

```
public Trail createTrail(int type){

 double score = 0.0;
 double coste = 0.0;
 double peso = 0.0;
 double phi = 0.0;
 Edge previousEdge = null;

 Trail trail = new DefaultTrail(type);

 LearnerProfile p = new LearnerProfile(type);  // Perfil de la hormiga

 Set<Vertex> visited = new HashSet<Vertex>();
 Vertex v = getRootVertex();
 double drops = 0.0;
 while (graph.outDegree(v) > 0) {
  if (!visited.contains(v)) { // Simular la evaluacion del alumno en v
   score = takeExam(v, p);
   // Incrementar feromonas en función de la nota obtenida
   if (previousEdge != null) {
    phi = calcularPhi(score, v.getMinimum(), v.getMaximum(), PHI, OMEGA3);
    drops = previousEdge.getRecentDrops().getValue();
    previousEdge.setRecentDrops(new Pheromone(drops + phi));
    v.addScore(score, ""+previousEdge.getId());
   }else {
     v.addScore(score);
   }
   v.setLastScore(score);
   trail.add(v); // Añadir al recorrido
   peso = peso + v.getPeso();// Sumar al peso total
   coste = coste + v.getAverageScore();
  }
  Vertex next = nextVertexOnTrail(trail, v, prob, score);
   // Obtenemos el arco a partir del siguiente vértice
  previousEdge = (Edge) graph.findEdge(v, next);
  v = next;
 }
 // Último nodo
 score = takeExam(v, p);
 if (previousEdge != null) {
  phi = calcularPhi(score, v.getMinimum(), v.getMaximum(), PHI, OMEGA3);
  drops = previousEdge.getRecentDrops().getValue();
  previousEdge.setRecentDrops(new Pheromone(drops + phi));
  v.addScore(score, ""+previousEdge.getId());
 }else {
  v.setLastScore(score);
 }
 v.setLastScore(score);
 trail.add(v);       // Añadir la hoja al recorrido
 peso = peso + v.getPeso();
 coste = (coste + v.getAverageScore())/peso;
 trail.setCost(new NumericCost(coste));

 return trail;
}
```

Listado 26: Creación de un camino por la factoría

El rendimiento de cada alumno puede simularse por las hormigas ACO, que según su perfil, obtendrán una mayor o menor calificación en el proceso de simulación del examen de cada nodo. Cada vez que una hormiga calcula un nuevo camino lo almacena en su memoria. Tras crear el perfil como alumno para la hormiga actual, comienza un recorrido por el grafo, siempre desde el nodo raíz (con grado de entrada 0) hasta el nodo final (con grado de salida 0). Durante el recorrido, se va simulando la evaluación del alumno (representado por la hormiga virtual) mediante el método `takeExam(v, p)` que devuelve la calificación para el alumno en el nodo *v*, de acuerdo a las probabilidades dadas por su perfil *p*. La generación de la calificación se calcula mediante una función que devuelve con una probabilidad *pr* un número aleatorio dentro del rango [*min, max*] determinado por el perfil de rendimiento del alumno, con *MIN_SCORE* < *min* < *max* < *MAX_SCORE* siendo el intervalo [*MIN_SCORE*, *MAX_SCORE*] el rango de calificaciones posibles.

```
/**
 * Genera con probabilidad p un valor aleatorio comprendido en el intervalo
 * R = [min,max] incluido en el intervalo [rangeMin,rangeMax].
 *
 * Precondiciones: 0<E(probability)<100 & RANGE_MINIMUM<min<max<RANGE_MAXIMUM
 * Postcondiciones: n in [min,max] con probabilidad = probability
 * @param probability Probabilidad de obtener un entero entre el intervalo
     dado dentro del rango [RANGE_MINIMUN,RANGE_MAXIMUM]
 * @param min Número mínimo del intervalo
 * @param max Número máximo del intervalo
 * @param rangeMin mínimo valor del intervalo de búsqueda
 * @param rangeMax máximo valor del intervalo de búsqueda
 * @return Devuele un númoer aleatorio dentro del rango [min,max] con una
      probabilidad dada.
 * @throws IllegalArgumentException Si no se cumplen las precondiciones
 */
public int generateRandomIn(float probability, int min, int max, int
rangeMin, int rangeMax )throws IllegalArgumentException {
 int result;
 int p;
 if (min > max ||
     min < rangeMin ||
     max > rangeMax ||
     probability < 0f ||
     probability > 100f) {
  throw new IllegalArgumentException("Invalid range");
 }

 p = Math.round((rangeMax - rangeMin)*(probability/100));
 // En el constructor se construye: Random random = new Random();
 int r = random.nextInt(rangeMax);
 if (r < p) {
     result = generateRandomIn(min,max,rangeMin,rangeMax);
 }else {
     result = generateRandomOut(min,max,rangeMin,rangeMax);
 }

 return result;
}
```

Listado 27: Generación aleatoria de un numero con una probabilidad dada de estar contenido en un rango determinado

Para la generación de un número aleatorio dentro de un rango determinado utilizamos la función matemática `nextInt` de la clase `Random` del paquete `java.util` de Java. La función `nextInt(int n)` función genera números pseudoaleatorios con una distribución aproximadamente uniforme en el rango [0, *n*]. Para conseguir un número aleatorio en el rango [min,max] utilizamos la expresión:

```
random.nextInt(max - min) + min
```

Puesto que esta clase sólo ofrece la posibilidad de generar un número aleatorio dentro de un rango determinado, la forma de actuar para obtener un número aleatorio fuera de un determinado rango es rellenar un vector con números aleatorios que se encuentren en los diferentes intervalos fuera del rango deseado, tal y como se describe en el ejemplo de la Figura 21:

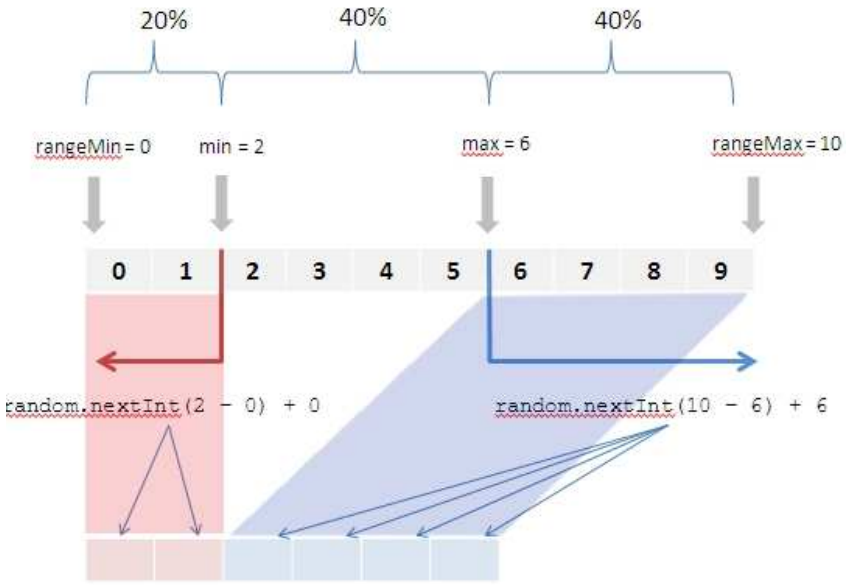


Figura 21: Esquema para la generación de números aleatorios fuera de un rango dado

EL algoritmo del Listado 26 construye caminos desde el nodo raíz al nodo final. En cada iteración, se obtiene un nuevo nodo mediante el método `nextVertexOnTrail` que se muestra en el Listado 28.

```
private Vertex nextVertexOnTrail(Trail trail, Vertex v, float prob, double
score) {
 Vertex destino = null;
 // Obtenemos los arcos de salida
 Collection<Edge> arcos = graph.getOutEdges(v);
 // Obtenemos los vecinos del nodo
 Collection<Vertex> vecinos = graph.getNeighbors(v);
 // Para cada arco saliente calcular su tabla de decisión
 Hashtable<Edge,Double> decisionTable = createDecissionTable(arcos, score);
 // Seleccionar uno de ellos en base a dicha tabla de decision
 Edge edge = chooseEdge(decisionTable);
 // Y se añade al recorrido si no lo estaba
 Collection<Vertex> incidents = graph.getIncidentVertices(edge);
 if (incidents.size() > 1) { // El vértice no está conectado consigo mismo
     destino = (Vertex) graph.getDest(edge);
 }

 return destino;

}
```

Listado 28: Método que calcula el siguiente arco en la construcción de un posible itinerario solución

Cada vez que el algoritmo busca el siguiente vértice en el camino que está construyendo, vuelve a construir la tabla de decisión para cada arco. En lugar de crear la tabla de decisión al inicio del proceso y actualizarla durante la ejecución, se ha optado por esta opción para permitir cambios estructurales en el grafo de itinerarios de aprendizaje en tiempos de ejecución. Esto es, la eliminación o adicción de nodos y sus correspondientes arcos, lo que podría dar lugar a la desaparición o aparición de caminos respectivamente. Es por esta razón que se ha optado por una tabla Hash como estructura de datos para la tabla de decisión en lugar de utilizar un *array* de tamaño fijo.

El Listado 29 muestra el algoritmo encargado de la creación de la tabla de decisión. Este algoritmo recorre el conjunto de arcos del grafo y calcula para cada uno de ellos el valor correspondiente para la tabla de decisión, según la ecuación (3).

La creación de cada elemento de la tabla de decisión se divide en dos tareas: calcular el numerador y calcular el denominador de la expresión (3). En el cálculo del numerador (Listado 30) se introduce la principal novedad de este trabajo, mediante el cálculo de la función de ajuste que actuará como valor heurístico, η, del problema.

```
private Hashtable createDecissionTable(Collection<Edge> edges, double
score) {
 double num = 0d;
 double denom = 1d;
 double probDecision = 0d;
 Hashtable<Edge,Double> h = new Hashtable<Edge,Double>();
 Iterator<Edge> iterator = edges.iterator();

 while (iterator.hasNext()) {
     Edge edge = iterator.next();
     num = calculaNumerador(edge, score);
     denom = calculaDenominador(edge, edges, num, score);
     probDecision = num/denom;
     if (probDecision <0)
          probDecision = 0;
     h.put(edge, new Double(probDecision));
 }
 return h;
}
```

Listado 29: Método de creación de la tabla de decisión

```
private double calculaNumerador(Edge e, double score) {
 double feromonas = e.getPheromone().getValue();
 double fitness = calculaFitnessFunction(e, score);

 e.setWeight(fitness); // Y lo fijamos como peso
 double result = Math.pow(feromonas, ALFA) * Math.pow(fitness,BETA);

 return result;
}

private double calculaFitnessFunction(Edge edge, double score) {
 double result=0;
 Vertex dest = (Vertex) graph.getDest(edge);
 double weight = dest.getWeight();
 double pij = calculaFactorIdoneidad(edge, score);

 result = this.OMEGA1*weight*pij +
      this.OMEGA2*edge.getRecentDrops().getValue();

 return result;
}
```

Listado 30: Cálculo del numerador en la expresión de probabilidad de un arco de la tabla de decisión

El método encargado de calcular el valor resultante de la función de ajuste, `calculaFitnessFunction`, devuelve el valor correspondiente a la ecuación (39), por lo que necesita calcular el factor de idoneidad P_{ij} y hacer uso del nuevo tipo de feromonas de deposición *on-line*, que se almacenan en cada arco en un atributo específico al que hemos denominado *recentDrops*.

El cálculo del factor de idoneidad, P_{ij}, implementa el procedimiento comentado en el capítulo 3.3.4, para lo que es necesario incluir en los vértices un atributo que registre el número de hormigas que han obtenido una calificación superior al mínimo establecido para el curso de dicho vértice. Por su mayor extensión y complejidad no se expone el código fuente del método `calculaFactorIdoneidad` en esta Tesis pero puede consultarse en el código fuente en el archivo o a la misma.

Para el cálculo del denominador en cada elemento de la tabla de decisión, se realiza un recorrido por los arcos de la colección que se pasa como parámetro (Listado 31). Además se proporcionan como parámetro el arco actual y el valor del numerador (cuya expresión representa cada uno de los sumandos del denominador) para facilitar su cálculo.

```
private double calculaDenominador(Edge e, Collection<Edge> edges,
double numerador, double score) {
 double denom = 1;
 double aux = 0D;

 Iterator<Edge> iterator = edges.iterator();
 while (iterator.hasNext()) {
     Edge edge = iterator.next();
     if (!edge.equals(e)) {
          aux = aux + calculaNumerador(edge, score);
     }else {
// si es el propio nodo se le pasa su valor ya previamente calculado
          aux = aux + numerador;           }
     }

     if (aux != 0) {
          denom = aux;
     }

     return denom;
}
```

Listado 31: Cálculo del denominador como sumatorio de la expresión que forma el numerador

La probabilidad que tiene un arco de ser seleccionado viene dada por la ecuación (4). Esta probabilidad se expresa como el valor de la tabla de decisión para un

arco frente a la suma de los valores de cada uno de los arcos con destino en un nodo vecino (método `calculaProbVecinos` del Listado 33).

El procedimiento de elección del siguiente nodo también hace uso de la función `random` de la clase `java.util`. En esta ocasión se obtiene un número aleatorio *r*, con $0 < r < 100$, y se recorre tabla de decisión restando en cada iteración el valor correspondiente a la probabilidad de cada elemento (expresada como un porcentaje entre 0 y 100). Este mecanismo, cuya implementación se muestra en el Listado 32, constituye una especie de torneo estocástico que otorga más probabilidad de salir elegido a aquellos arcos cuya probabilidad (valor de la tabla de decisión) sea mayor, pero que no garantiza de forma determinista su elección.

```
private Edge chooseEdge(Hashtable<Edge,Double> tabla) {
 int prob=0;
 Set<Edge> edgeSet = tabla.keySet();
 Iterator<Edge> iterator = edgeSet.iterator();

 int total = sg.generateRandomIn(0, 100, 0, 100);
 Edge edge = iterator.next();

 while (iterator.hasNext() && total>0) {
     if (trail.getElements().contains(graph.getDest(edge))) {
          prob = 0;
     }else {
          prob = (int) Math.floor(
     ((tabla.get(edge).doubleValue()/calculaProbVecinos(trail,tabla,
edge))*100);
     }

     total = total - prob;
     if (total > 0) {
          edge = (Edge) iterator.next();
     }
 }

 return edge;
}
```

Listado 32: Método de elección de un arco en la tabla de decisión

Actualización de feromonas

La actualización y evaporación de feromonas es un proceso secuencial iniciado desde el planificador y que se ejecuta al final de cada iteración. Se produce la evaporación tanto de las feromonas habituales de ACO, τ, como de las feromonas utilizadas en la función de ajuste, Φ. Primero se realiza la evaporación de las feromonas existentes en el arco y luego se produce el depósito de feromonas en aquellos arcos que

forman parte de la solución. El incremento para las feromonas τ es determinado por la ecuación (47), que proporciona un refuerzo equivalente a la suma de los valores de ajuste de cada arco perteneciente al camino. Por su parte para el refuerzo en las feromonas Φ, que a su vez influyen en la función de ajuste, se aplica la fórmula (46), que utiliza como el refuerzo adicional la calificación obtenida al final del recorrido. La evaporación de las feromonas se realiza en el método `evaporate` de la clase `AsaliPheromoneUpdater`, encargada de la funcionalidad de actualización y evaporación de las feromonas al final de cada iteración.

```
private double calculaProbVecinos(Trail trail, Hashtable table, Edge
edge) {
 double result = 0d;
 Vertex v = null;
 Edge neighborEdge = null;

 Vertex vsource = (Vertex) graph.getSource(edge);
 Collection neighbours = graph.getNeighbors(vsource);
 Iterator<Vertex> iterator = neighbours.iterator();
 Double aux = null;

 while (iterator.hasNext()) {
     v = iterator.next();
     neighborEdge = (Edge) graph.findEdge(vsource, v);
     if (neighborEdge != null) {
          aux = (Double) table.get(neighborEdge);
          if (aux != null) {
               result = result + aux.doubleValue();
          }
     }
 }
 return result;
}
```

Listado 33: Método que calcula el denominador en la expresión de probabilidad de elección de un arco

El método `calculateTau` (Listado 34) es el encargado de calcular el valor de la expresión $\Delta\tau_{ij}^{k}(t)$ de la ecuación (47), mediante un recorrido por todos los arcos del grafo.

Por su parte, el método `calculateTheta` (Listado 35) calcula el refuerzo para las feromonas Φ a partir de la calificación obtenida en el último curso del itinerario, aplicando la ecuación (46).

```
private double calculateTau(Colony colony) {

 AsaliTrailFactory factory = (AsaliTrailFactory)
   colony.getTrailFactory();
 Graph<Vertex,Edge> graph = factory.getGraph();
 Collection<Edge> edges = graph.getEdges();
 Trail trail = colony.getCurrent().getTrail();
 Collection<GraphElement> tvertices = trail.getElements();

 double tau = 0d;
 Vertex source = null;
 Vertex dest = null;

 for (Edge edge : edges)    {
     source = graph.getSource(edge);
     dest = graph.getDest(edge);
     if (dest != null && tvertices.contains(dest)) {
          if ( ((source != null) && tvertices.contains(source)) ||
          (source == null)) {
               tau = tau + factory.calculaFitnessFunction(edge,
                  dest.getAverageScore());
          }else {
               tau = 0;
          }
     }else {
          tau = 0;
     }

 }

 return tau;
}
```

Listado 34: Método encargado de calcular el incremento de feromonas τ

```
private double calcularTheta(double score, Vertex v) {
 double result = 0.0;
 double min = v.getMinimum();
 double max = v.getMaximum();
 double fi = AsaliTrailFactory.PHI;
 double w3 = AsaliTrailFactory.OMEGA3;

 if (score/max < min/max) {
     result = - (1 - score/max)*fi - w3*(min/max -score/max);
 }else {
     result = (score/max)*fi + w3*(score/max - min/max);
 }

 return result;
}
```

Listado 35: Método que calcula el refuerzo para las feromonas Φ

Los valores de las variables tau y theta se corresponden con las expresiones $\Delta\tau$ y θ de las ecuaciones (47) y (45) respectivamente.

Este refuerzo se lleva a cabo tras producirse la evaporación de las feromonas existentes y siguiendo una estrategia elitista, que sólo reforzará los arcos pertenecientes a la mejor solución.

La implementación de este mecanismo de refuerzo es responsabilidad del método `reinforceEdge` de la clase `AsaliPheromoneUpdater`, cuya implementación se muestra en el Listado 36.

```
private void reinforceEdge(Graph<Vertex, Edge> graph, Edge edge,
Collection<GraphElement> tvertices, double tau, double theta) {
 Vertex dest = graph.getDest(edge);
 Vertex source = graph.getSource(edge);
 double drops = edge.getRecentDrops().getValue();
 double pheromones = edge.getPheromone().getValue();

 if (dest != null && tvertices.contains(dest)) {

  Vertex previous = getPreviousVertex(tvertices, dest);
     if ((source != null) && tvertices.contains(source)
          && previous != null && previous.equals(source)) {
      drops = drops + theta;
      pheromones = pheromones + tau/((double) tvertices.size());
      edge.setRecentDrops(new Pheromone(drops));
      edge.setPheromone(new Pheromone(pheromones));
     }
  }
}
```

Listado 36: Método reinforceEdge encargado del refuerzo del arco

El método auxiliar `getPreviousVertex,` Listado 37, obtiene el vértice previo al de destino en la lista de vértices que se pasa como parámetro .

Ejecución de tareas adicionales por el planificador

En nuestro caso las tareas adicionales llevadas a cabo al final de cada iteración por el planificador, consisten en la grabación de datos que permitan estudiar el comportamiento del algoritmo y la evolución de las feromonas de cada arco. Para ello se implementa en la clase `AsaliDaemonAction`, el método `perform(Colony colony)` de la interfaz `DaemonAction`. Este método es el encargado de grabar en

ficheros de texto con extensión `.csv` los datos necesarios para el estudio del algoritmo. En concreto graba dos ficheros, uno con los itinerarios seleccionados por cada hormiga en cada iteración y otro con la evolución de los pesos y feromonas de cada arco del grafo, según el comportamiento (calificaciones obtenidas) de la colonia.

```
private Vertex getPreviousVertex(Collection<GraphElement> tvertices,
Vertex dest) {
 Vertex pv = null;
 Vertex aux = null;
 boolean enc = false;

 Iterator<GraphElement> iterator = tvertices.iterator();
 while (!enc && iterator.hasNext()) {
     aux = (Vertex) iterator.next();
   if (aux.equals(dest)) {
       enc = true;
     }else {
       pv = aux;
     }
 }

 if (!enc) {
     pv = null;
 }

 return pv;
}
```

Listado 37: Método que obtiene el vértice anterior a uno dado

Consideraciones

Mientras que en la vida real, las feromonas se evaporan con el tiempo, medido éste en segundos, en nuestra solución las feromonas se evaporan al final de una iteración. El final de una iteración coincide en nuestro caso con el final de una promoción de alumnos que participa en la formación, y sucede cuando o bien todos los alumnos han finalizado la formación o bien cuando se ha llegado a la fecha límite para su finalización.

A diferencia de lo que sucede en la vida real, nuestro algoritmo lanza a las hormigas en busca de solución secuencialmente y no en paralelo. En la formación on-line en la vida real, es muy frecuente que los alumnos de la misma promoción comiencen sus primeros cursos en momentos diferentes, y vayan completando su itinerario a velocidades diferentes. El tipo de hormiga que se lanza cada vez (High, Medium o Low) es aleatorio, esto es, no se lanzan todas las de un tipo y seguidamente las de otro.

Capítulo 4. Resultados experimentales

4.1 Entorno de pruebas

Las pruebas del comportamiento de los algoritmos se han realizado con un ordenador HP 6730b con procesador Intel Centrino® 2, con 2Gb de memoria RAM. El programa de pruebas se ha implementado en Java™ y se ejecuta sobre el JRE 1.6.

4.2 Descripción de la experimentación

El objetivo de la experimentación es estudiar la capacidad del algoritmo ASALI para, utilizando las calificaciones obtenidas por los alumnos en cada curso, poder adaptar el itinerario más recomendable para la mayoría de alumnos en las sucesivas promociones e incluso, en menor medida, en la misma promoción.

El método usado consiste en la simulación del comportamiento de los alumnos por las hormigas del algoritmo ACO, de forma que aplicando el método de Monte Carlo pueda generarse un estado inicial y a partir de él comprobarse la capacidad de la solución propuesta para adaptar de forma autónoma el itinerario en función de las calificaciones medias obtenidas en cada nodo. El comportamiento de los alumnos queda limitado en este experimento a la simulación de la realización del examen que proporcionará al alumno la calificación en cada curso, para lo que se usará un generador de calificaciones pseudoaleatorias. Una vez obtenida la calificación el alumno continuará el itinerario por el siguiente curso que asigne el algoritmo. En este caso no se ofrece al alumno varias opciones a elegir, puesto que esto dificultaría el estudio de adaptabilidad del algoritmo, al estar sometido entonces el proceso a la libre e impredecible elección del alumno.

Se ha optado por simular el comportamiento del alumno ante la dificultad para llevar a cabo experimentos con alumnos reales, lo que aumentaría el tiempo y coste del experimento, y dificultaría la extracción de conclusiones relativas al comportamiento del algoritmo al estar presente una componente humana sobre la que no se tiene control. En este experimento nos interesa estudiar la variación en refuerzo del mejor camino (el de mayor calificación media final) en diferentes muestras de alumnos, cada una de ellas

con diferentes proporciones de alumnos de cada perfil (que en nuestro experimento simplificaremos en tres perfiles, según su rendimiento). Puesto que el objetivo es estudiar el comportamiento del algoritmo, y no realmente el beneficio que puedan obtener los alumnos reales de la aplicación de éste en un entorno de e-learning real (cuyo estudio supondría un trabajo futuro de continuación de esta tesis) partimos de la hipótesis de que existen estas promociones de alumnos, cuyas calificaciones simularemos mediante un generador aleatorio de calificaciones, dentro de un rango adecuado a cada perfil.

Para la experimentación se utiliza el algoritmo ASALI en sucesivas iteraciones, guardando el estado al final de cada iteración en una hoja de cálculo, para comprobar la evolución del itinerario en cada iteración.

4.2.1 Consideraciones previas

Frente a la evaluación sistemática como mecanismo de selección de los mejores individuos este trabajo persigue la concepción de la evaluación como método de identificación de necesidades de refuerzo, y en consecuencia un mecanismo causal para poder plantear caminos alternativos. Estos caminos podrían ser detectados de entre los cursos alternativos, que a pesar de tener un menor peso pedagógico se ajustaran mejor a las características de los usuarios. Esta detección podría ser identificada a través de la función de ajuste planteada en este trabajo (39), que vería reforzado su valor bien por el factor de idoneidad del curso (42) o por el refuerzo proporcionado por otros alumnos que lo hubieran realizado a través de las feromonas on-line depositadas (44). Siguiendo un enfoque optimista, si al inicio del itinerario los alumnos están distribuidos en forma normal respecto a su aptitud para una determinada materia, y durante el proceso se les proporciona una instrucción de acuerdo a las características y necesidades de cada alumno, el rendimiento de ellos al término del proceso deberá indicar que casi todos logran el dominio del aprendizaje, superando una determinada calificación mínima deseada (Figura 22).

Figura 22: Distribución teórica de calificaciones antes y después de la formación

Para simplificar la complejidad de la experimentación y poder simular la diversidad de alumnos en cuanto a diferente ritmo de aprendizaje y diferentes resultados (calificaciones) se discretiza la población de alumnos en tres perfiles: `Low`, `Medium` y `High`, correspondiéndose respectivamente con alumnos que obtienen calificaciones bajas (es un caso muy común en la formación on-line en empresas, con profesionales que tienen durante la formación un pico de trabajo alto que les resta tiempo de dedicación a la formación), medias (que sería el caso más frecuente) y alumnos que obtienen calificaciones excelentes (no hay correspondencia directa con los alumnos más brillantes, sino que, en muchas ocasiones, suele ser indicativo de utilización de algún método de respuesta fraudulento en los tests de evaluación, y generalmente se corresponde con un tiempo de dedicación a la realización del test inferior a la media). Así, en lugar de suponer una probabilidad uniforme para la variable que representa la calificación, se aplican los perfiles indicados tal y como se muestra a continuación.

Esta clasificación no supone que los alumnos pertenezcan a uno u otro grupo antes de comenzar el curso, sino que es una previsión de lo que queremos obtener una vez finalizado el curso por todos los alumnos, pues las calificaciones serán simuladas mediante algoritmos de generación de números aleatorios con una probabilidad dada.

Para simular esta variedad en la población de alumnos, se modifica la colonia de hormigas del algoritmo ACO propuesto para que incluya hormigas de los tres tipos diferentes de perfil. Para ello se añade al framework la clase `LearnersColony` que amplía la funcionalidad de una colonia genérica de hormigas (clase `Colony`) para dar cabida a varios tipos de hormigas diferentes (Figura 23), cada uno tendente a obtener una calificación dentro del rango más probable que le corresponde.

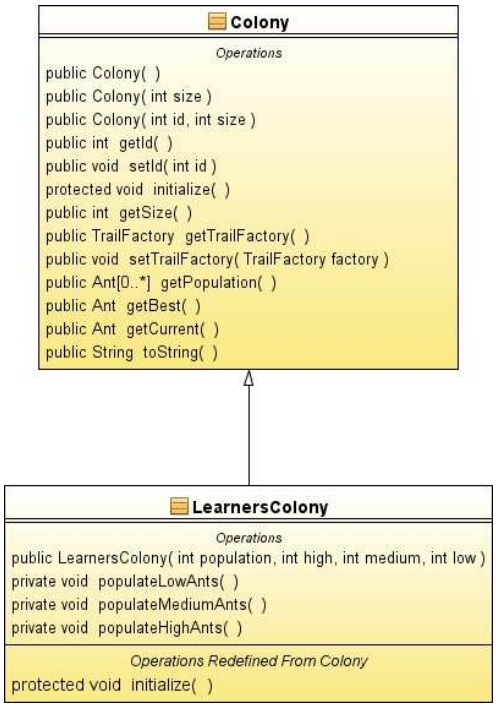


Figura 23: Colonia de alumnos como ampliación de una Colonia genérica

Consecuentemente hay que adaptar la interfaz gráfica de la aplicación de prueba con el objeto de poder especificar la proporción de la población de la colonia que pertenece a un perfil u otro, de forma que pueda crearse la colonia con estas proporciones. La Figura 24 muestra una captura de la interfaz gráfica del programa creado para la experimentación y que utiliza el *framework* descrito en el capítulo anterior. Mediante unos controles deslizantes se permite modificar la proporción (en porcentaje) de un perfil u otro de la población de la colonia, lo que permitirá simular una audiencia de alumnos compuesta por alumnos de estos perfiles en igual proporción. Además de las proporciones de cada perfil de alumno, el panel de configuración permite modificar las variables de calibración del sistema (ω_1, ω_2 y ω_3), las variables que controlan la tendencia a explorar o explotar soluciones (α y β), la composición de la colonia (número de colonias y número de hormigas por colonia) y el número de iteraciones que cada hormiga realizará.

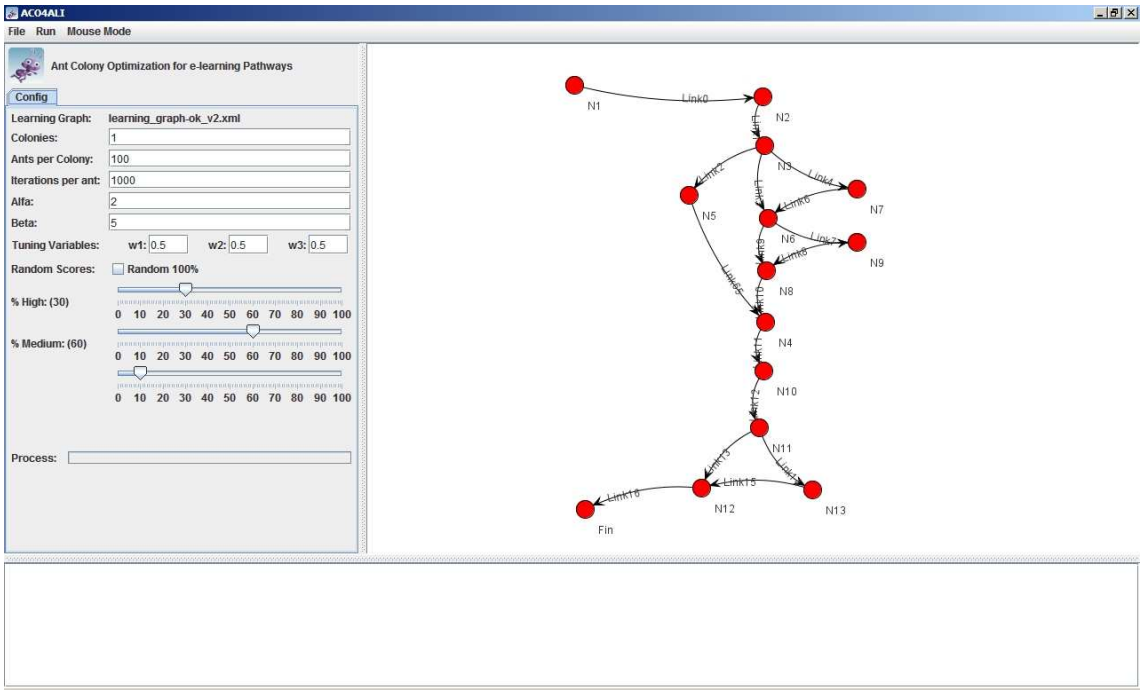


Figura 24: Aplicación ACO4ALI desarrollada para la experimentación

Para simular los diferentes perfiles de alumnos utilizaremos un generador de números pseudoaleatorios, que ha sido modificado para generar números de forma aleatoria dentro de un rango dado con una probabilidad determinada. Se utiliza una distribución de probabilidad normal. Esto no implica que la distribución de las calificaciones pudiese corresponder a un modelo probabilístico tras un período de enseñanza-aprendizaje, puesto que, supuestamente, el azar no tendría cabida cuando ha existido un esfuerzo sistemático por mejorar los niveles del conocimiento de los alumnos. Para nuestro experimento vamos a suponer que el resultado de la evaluación de un curso es independiente de los obtenidos en cualquier otro anterior,

El perfil Low clasifica los alumnos participantes en el itinerario formativo cuyas calificaciones en los test de evaluación de curso se encuentran en un 95% entre un 3 y un 5 (en una escala de 0 a 10). Es el perfil menos habitual, y cuando aparece suele ser indicativo de falta de dedicación al curso o de mala comprensión del mismo. La gráfica de la Figura 25 muestra la función de distribución de probabilidad de las calificaciones obtenidas por alumnos de este perfil.

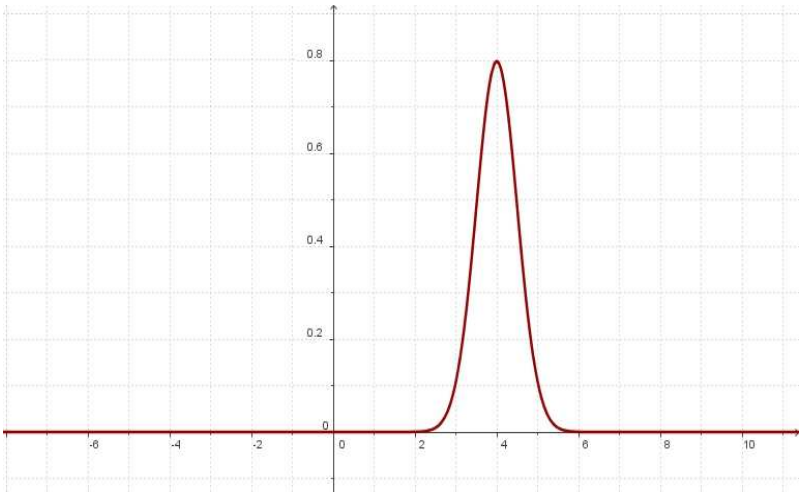


Figura 25: Función de densidad de probabilidad de un modelo normal N(4,0.5)

Los alumnos del perfil Medium obtienen el 95% de sus calificaciones entre un 5 y un 7 (en una escala de 0 a 10). Los alumnos de este perfil superan el mínimo exigido (un 5) pero no adquieren un dominio notable de las capacidades y conocimientos para los que fueron formados. Es el perfil medio y su gráfica de distribución de probabilidad para sus calificaciones es la mostrada en la Figura 26.

Por último, los alumnos etiquetados como perfil High son los alumnos que obtienen el 95% de sus resultados de evaluaciones en una calificación comprendida entre el 7 y el 9 en una escala de 0 a 10. Los alumnos de este perfil evidencian un dominio excelente de las capacidades para las que han sido formados. Es el perfil más deseado y en entornos de formación online suele ser el más habitual. La función de distribución de probabilidad usada para la generación de las calificaciones pseudoaleatorias para este perfil es la indicada en la Figura 27.

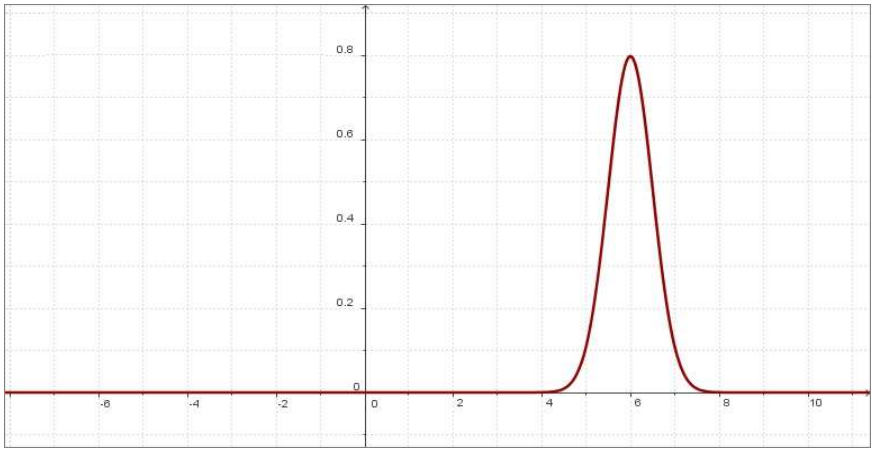


Figura 26: Función de densidad de probabilidad de un modelo normal N (6.0.5)

El programa nos permitirá simular la realización del curso por un conjunto de alumnos, determinando a priori la proporción de cada nivel de alumnos que se desea para el estudio de la variación del itinerario (a través de los controles de selección del panel de configuración), en función de las características del alumnado.

Para los alumnos que no pertenezcan a ningún perfil, se utilizará una función que genera las calificaciones correspondientes a las evaluaciones de cada curso de forma aleatoria y con una probabilidad uniforme. Esta última opción se habilita activando la casilla Random Scores del panel de configuración.

Además de considerar diferentes perfiles para las hormigas del algoritmo, que simularán el comportamiento de los alumnos reales, es necesario considerar la estrategia de ponderación del grafo así como la calibración del mismo, estudiando el comportamiento del algoritmo con la variación de los parámetros del algoritmo: α, β, ω_1, ω_2, ω_3 y ρ.

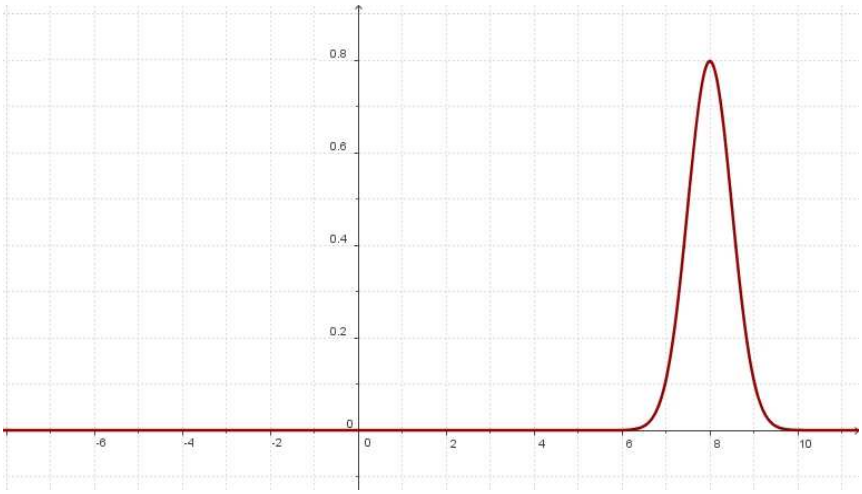


Figura 27: Función de densidad de probabilidad de un modelo normal N (8,05)

El objetivo final del algoritmo es maximizar el resultado de la función beneficio B^k que nos devuelve la calificación media ponderada con el peso total del recorrido realizado por la hormiga *k*. El beneficio obtenido por la hormiga *k* en su recorrido es:

$$B^k = \frac{\sum_i^n \bar{\varepsilon}_i}{\sum_i^n W_i} \tag{48}$$

donde $i \in R^k$ y su grado de entrada, $g_{in}(i)$, es mayor que cero. Se obtiene mediante la división de la suma de las calificaciones medias de cada nodo perteneciente al recorrido (excepto el nodo inicial) entre la suma de los pesos de los nodos de este recorrido (a excepción del peso del primer nodo).

Puesto que el denominador de la ecuación (48) permanece constante para cada itinerario, el beneficio será mayor en un itinerario cuanto mayor sea la suma de las notas medias de ese itinerario, o lo que es lo mismo, cuando aumente la nota media del itinerario. Los pesos asignados a cada nodo pueden influir en la probabilidad de selección de un arco, y en consecuencia en el beneficio obtenido, puesto que al actuar como factor multiplicador en el primer sumando de la función de ajuste (39), podrían influir en la construcción del camino, por ejemplo, seleccionando los nodos cuyos cursos sean de menor dificultad. A continuación estudiamos las estrategias de ponderación del grafo.

Estrategias de Ponderación del Grafo de Itinerarios de Aprendizaje

Equiponderar

Si asignamos el mismo peso a todos los cursos, entonces el beneficio será mayor cuanto mayor sea la calificación media del itinerario. Así si un recorrido *k* está formado por *m* cursos, si todos tienen el mismo peso, la función beneficio del recorrido *k* viene dada por la ecuación:

$$B^k = \frac{\sum_{i=1}^{m} \overline{\varepsilon}_i}{\sum_{i=1}^{m} W_i} = \frac{\overline{\varepsilon}_i + \overline{\varepsilon}_{i+1} + \ldots + \overline{\varepsilon}_n}{m \cdot W_0} = \frac{\overline{\varepsilon}_{R^k}}{W_0}$$

A priori, esta medida puede potenciar en exceso los caminos más fáciles. El razonamiento es sencillo: al tener los caminos posibles una función de ajuste similar en un primer momento, todos los caminos tendrán unas probabilidades similares de ser elegidos. Aquellos caminos que tengan un menor nivel de dificultad, obtendrían un mayor depósito de feromonas que llevaría rápidamente al algoritmo al bloqueo en un mínimo local, encaminando al resto de hormigas (alumnos en nuestro caso) por el itinerario de menor dificultad. En cambio, si el nivel de dificultad de los cursos es homogéneo, el resultado será impredecible, dependiendo la elección de los nodos de las calificaciones obtenidas en cada curso.

Para comprobar el comportamiento del algoritmo, utilizaremos el grafo de la Figura 28, en el que todos sus nodos tienen mismo peso, mínimos deseados y máximos.

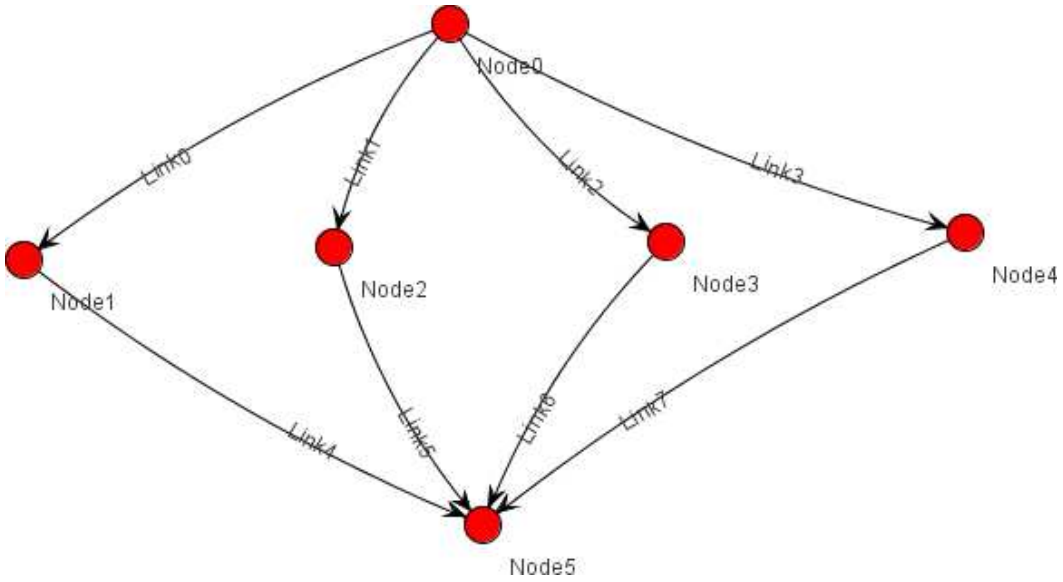


Figura 28: Grafo utilizado para probar la estrategia de asignación de pesos iguales a todos los nodos.

Los valores para cada nodo *i* del grafo son los indicados a continuación:

- Peso Pedagógico (W_i): 1.0
- Mínima calificación deseada (m): 5.0
- Máxima calificación posible (M): 10.0
- Calificación media (: 0.0

Todos los arcos del grafo tienen también los mismos atributos inicialmente:

- Peso: 192.0 (el valor no importa, siempre que todos los nodos tengan el mismo peso)
- Feromonas (τ): 0.0

Se realizan una serie de ejecuciones del algoritmo ASALI para simular promoción de alumnos en los siguientes casos:

Población de alumnos homogénea: se cuenta con un grupo de alumnos que obtiene todos el mismo rango de calificaciones (todos los alumnos de perfil High, todos los alumnos de perfil Medium o todos los alumnos de perfil Low). Se ejecuta el

algoritmo para una población de 1000 hormigas, que simularán la realización del curso por parte de 1000 alumnos (realizando una iteración únicamente). Las variables de calibración del sistema elegidas son: α=2, β=5, ω_1=0.5, ω_2=0.5 y ω_3=0.5. Se repite el experimento un número suficiente de veces para comprobar que el algoritmo no muestra sesgo hacia ningún camino en especial. Los resultados obtenidos demuestran que:

- Población homogénea de nivel High: en este experimento ninguno de los cuatro caminos posibles tiene preferencia sobre los otros tres. A partir de la hormiga vigésimo quinta, el algoritmo se queda bloqueado en una solución. La desviación típica aumenta conforme aumenta el número de hormigas que ha finalizado el curso (Tabla 7), lo que significa que conforme más hormigas van finalizando su camino, la influencia de las feromonas Φ es mayor, y el número de hormigas que escoge un camino se va distanciando más de la media teórica para cada camino que, en un sistema ideal que mantiene la misma probabilidad para cada camino, sería muy próxima a K/N, dónde K es el tamaño de la colonia de hormigas (población) y N es el número de caminos posibles.

Tabla 7: Desviación Típica para el experimento de ponderación con alumnos perfil High

Población	20	50	100	200	500	1000
Media	5,00	12,50	25,00	50,00	125,00	250,00
Desviación Típica media	3,30	16,36	40,92	90,37	237,55	485,21
Coeficiente de Variación	0,66	1,31	1,64	1,81	1,90	1,94

Si observamos la Figura 29, en la que se muestra el número acumulado de hormigas que han seleccionado cada uno de los cuatro posibles caminos en una de las ejecuciones del experimento, podemos observar como uno de los itinerarios pasa a ser el único seleccionado por el algoritmo cuando el número de hormigas que han finalizado su camino es mayor a 20. Se produce, por tanto, un bloqueo en un óptimo local, que en este caso se corresponde con el camino con mayor valor para su función de ajuste, y cuyo crecimiento se debe exclusivamente al depósito de feromonas Φ.

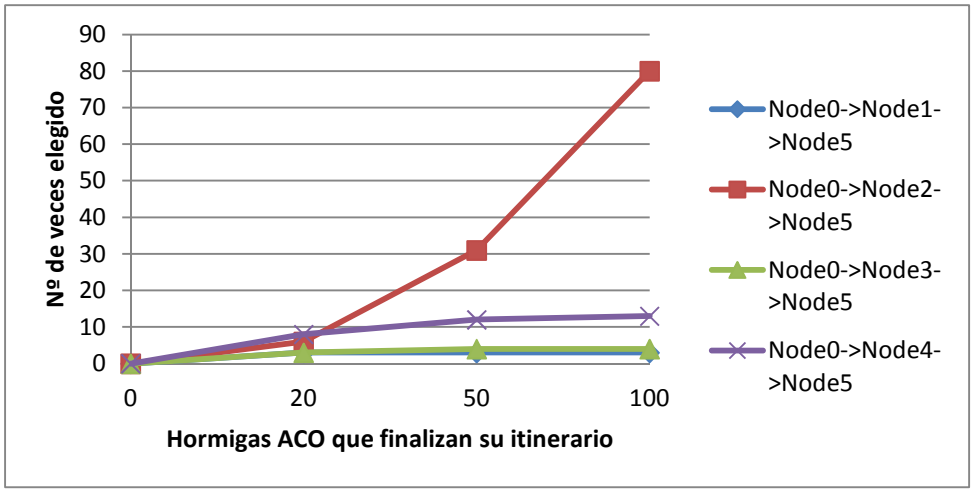


Figura 29: Evolución de la elecciones de caminos en una de las pruebas del experimento con alumnos de perfil High

- Población homogénea de nivel Medium: Los resultados son similares al caso anterior, aunque se observa un crecimiento más lento de la desviación típica (Tabla 8) lo que nos indica que la dispersión de los datos (número de hormigas que seleccionan un itinerario de entre los cuatro posibles) respecto a la media es ligeramente menor al caso anterior. La explicación está en que al simular estas hormigas a alumnos con un rendimiento académico más bajo (calificaciones menores) el refuerzo en feromonas Φ es menor y consecuentemente, el bloqueo de un camino es ligeramente más tardío.

Tabla 8: Dispersión para el experimento de ponderación con alumnos perfil Medium

Población	20	50	100	200	500	1000
Media	5,00	12,50	25,00	50,00	125,00	250,00
Desviación Típica media	2,18	6,04	19,83	55,42	166,30	351,57
Coeficiente de variación	0,44	0,48	0,79	1,11	1,33	1,41

En este segundo experimento, el "*torneo estocástico*" entre los caminos se mantiene igualado hasta la quincuagésima hormiga, quedando bloqueado en alguno de ellos a partir de ahí. La Figura 30 muestra la evolución de las elecciones para un caso concreto del experimento. El resto de ejecuciones de este caso son similares, variando el camino que acaba recibiendo el mayor número de feromonas pero produciendo el bloqueo siempre tras la hormiga número 50.

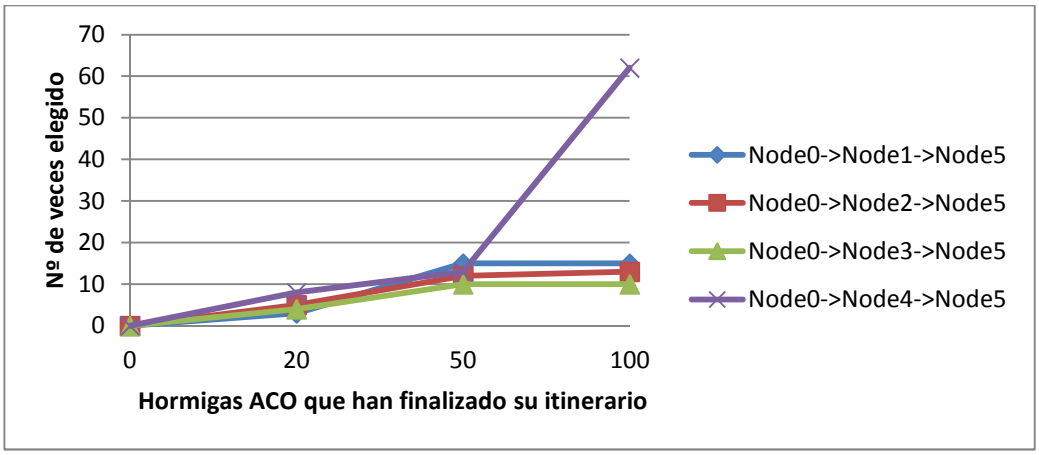


Figura 30: Evolución de las elecciones de caminos en una de las pruebas del experimento con alumnos de perfil Medium

- Población homogénea de nivel Low: Siguiendo el comportamiento descrito para los dos casos anteriores, en el experimento para alumnos de perfil Low, el bloqueo por un camino predominante tarda más en producirse, retrasándose en todas las pruebas realizadas para este experimento hasta pasada la hormiga número 200 (Figura 31).

Tabla 9: Dispersión para el experimento de ponderación con alumnos perfil Low

Población	20	50	100	200	500	1000
Media	5,00	12,50	25,00	50,00	125,00	250,00
Desviación Típica media	2,44	11,00	23,00	51,50	194,00	443,50
Coeficiente de variación	0,49	0,88	0,92	1,03	1,55	1,77

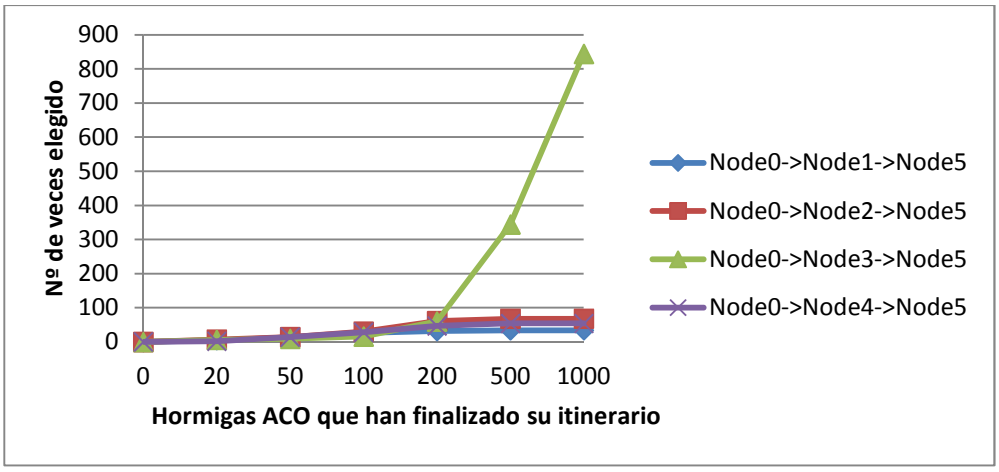
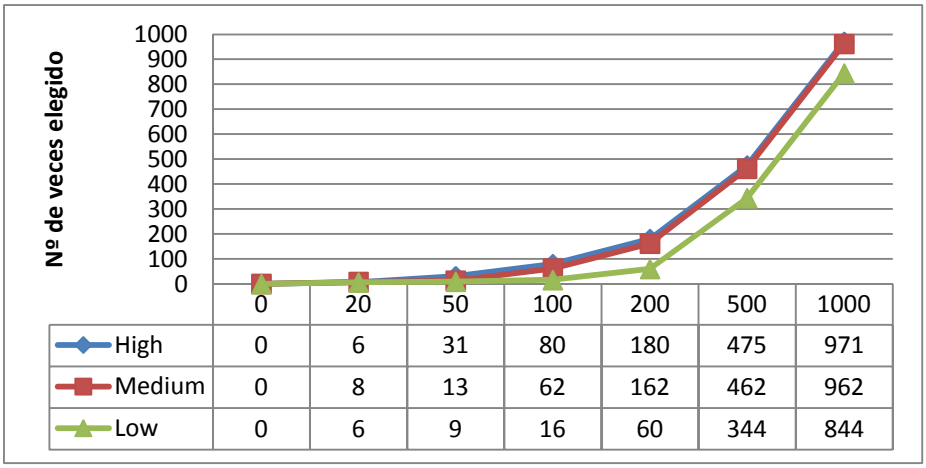


Figura 31: Evolución de las elecciones de caminos en una de las pruebas del experimento con alumnos de perfil Low

La Figura 32 muestra una comparativa de la evolución de elecciones acumuladas para los caminos que producen el bloqueo en los tres casos anteriores.



	0	20	50	100	200	500	1000
High	0	6	31	80	180	475	971
Medium	0	8	13	62	162	462	962
Low	0	6	9	16	60	344	844

Figura 32: Comparativa de la evolución de hormigas que seleccionaron el camino solución en los tres experimentos anteriores

El bloqueo para el perfil Low necesita un mayor número de hormigas, debido a la menor cantidad de feromonas que se depositan, lo que retrasa la aparición de una solución clara y favorece la realización de caminos diferentes.

A continuación vamos a comprobar qué pasa si la población de hormigas tiene diferentes perfiles.

- **Población de alumnos heterogénea.** Consideremos una población de alumnos en la que el 10% obtiene brillantes calificaciones (perfil High), el 60% calificaciones consideradas normales (perfil Medium) y un 30% tiene calificaciones inferiores (perfil Low). A priori, puesto que la cantidad de feromonas a depositar, y en consecuencia el refuerzo sobre cada arco, depende de la calificación obtenida, el comportamiento debe ser una mezcla de los casos anteriores, apareciendo un camino predominante (bloqueo) antes de llegar a la hormiga 200 pero después de haber pasado la 20. Para este caso se configura el algoritmo para que pueble la colonia con un 30% de hormigas de perfil High, un 60% de hormigas con perfil Medium y un 10% de hormigas de perfil Low. Las hormigas se lanzan en busca de caminos de

forma secuencial pero no ordenadas por tipo de perfil, siendo la elección del tipo de perfil aleatoria. La experimentación revela un comportamiento muy similar al primer experimento con hormigas de perfil High, pero en el que el bloqueo se produce, dependiendo de las calificaciones obtenidas entre la hormiga 20 y la 75.

Tabla 10: Dispersión para el experimento con una colonia de hormigas con diferentes perfiles (30% High, 60% Medium, 10% Low)

Población	20	50	100	200	500	1000
Media	5,00	12,50	25,00	50,00	125,00	250,00
Desviación Típica media	2,83	9,79	32,67	81,61	229,71	475,54
Coeficiente de variación	0,56	0,78	1,31	1,63	1,84	1,90

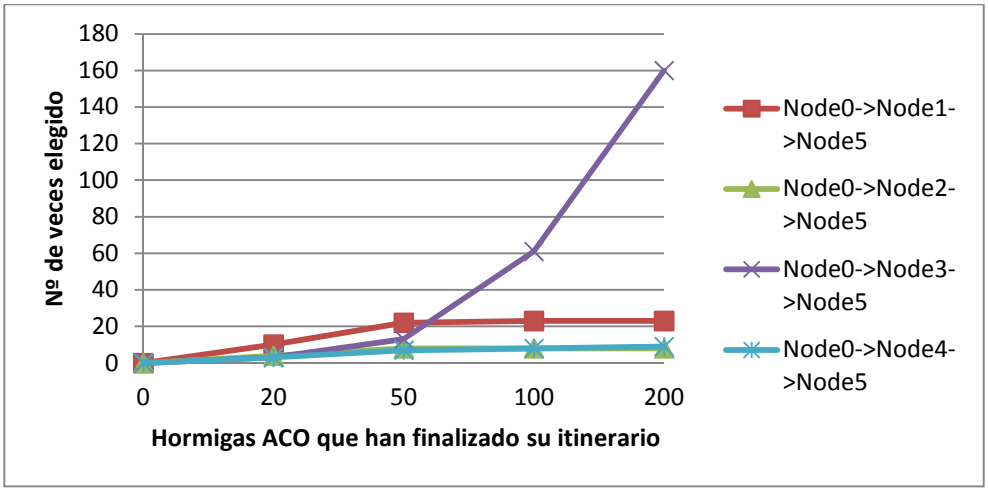


Figura 33: Evolución de elecciones de caminos en una de las ejecuciones para la prueba de colonia heterogénea con proporciones 30-60-10

No obstante a diferencia de las pruebas realizadas con colonias homogéneas, en este caso sí se observa que no es un bloqueo total, pues las últimas hormigas de la colonia, aunque en un número muy bajo, sí seleccionan caminos diferentes al que mayor número de feromonas acumula. Esto es debido a que las diferencias de depósitos de feromonas en los cuatro caminos opcionales no es tan grande como en los casos anteriores, y permite al algoritmo seleccionar un camino diferente en función de la calificación obtenida.

Ponderar en función del nivel de dificultad

La segunda opción a la hora de ponderar el grafo consiste en la asignación de un peso, $W_i > 0$, que refleje el nivel de dificultad del curso asociado al nodo *i*. Por ejemplo:

- Si el nivel de dificultad del nodo es el adecuado según los criterios del equipo pedagógico, se asignará como peso el valor $W_i = 1$.
- Si el nivel de dificultad del nodo es inferior al adecuado según el criterio del equipo pedagógico (por ejemplo se trata de un curso de refuerzo), se asignará como peso un valor W_i, tal que $0 < W_i < 1$. Esto minimizará el valor de la función de ajuste, si bien al ser de un nivel de dificultad menor es previsible que las calificaciones en los cursos de estos nodos sean mayores y por tanto el refuerzo gracias a las feromonas depositadas mayor.
- Siguiendo el mismo criterio para la asignación de pesos, puede establecerse un mínimo deseado para cada curso inversamente proporcional al nivel de dificultad, lo que también influirá en el cálculo de feromonas Φ a depositar.

Para comparar el comportamiento del algoritmo con la estrategia anterior, usamos el mismo grafo (Figura 28) ponderado según la Tabla 11.

Tabla 11: Ponderación basada en el nivel de dificultad para el grafo de la Figura 28

Nodo	W_i	m	M	$\bar{\varepsilon}$	Grado Salida	Arcos de Salida
N0	1	5.0	10.0	-	4	Link0, Link1, Link2, Link3
N1	1	5.0	10.0	-	1	Link4
N2	0.85	6.0	10.0	-	1	Link5
N3	0.70	7.0	10.0	-	1	Link6
N4	0.55	8.0	10.0	-	1	Link7
N5	0.50	8.0	10.0	-	0	-

Se realizan las mismas pruebas que para el estudio de la influencia de la estrategia basada en equiponderar el grafo, ejecutando el algoritmo para una población de 1000 hormigas que construirán una solución cada una. Las variables de calibración del sistema elegidas son las mismas: α=2, β=5, ω_1=0.5, ω_2=0.5 y ω_3=0.5. Se repite el experimento un número suficiente de veces.

Tras la experimentación se observa que el bloqueo en una solución aparece, con respecto al número de hormigas que terminan de construir su solución, ligeramente antes al observado en la estrategia anterior (equiponderación) con hormigas de perfil Medium. Para perfiles High y Low el comportamiento es similar, bloqueándose al completar su solución aproximadamente el mismo número de hormigas.

A diferencia de lo observado con la estrategia basada en la asignar los mismos pesos a todos los nodos, con esta estrategia basada en el nivel de dificultad si se observa una característica interesante en los experimentos realizados: el camino que pasa por el nodo de peso 1 (aquel con un nivel de dificultad adecuado) acaba siendo el que causa el bloqueo en el 90% de las veces, mientras que asignando el mismo peso a todos los nodos, los cuatro caminos tendían a ser los bloqueantes en el mismo número de veces. Por tanto esta estrategia, con los valores asignados a las variables de calibración y que posteriormente estudiaremos, cuando el número de alumnos es suficientemente alto, favorecerá el bloqueo de la solución en alguno de los caminos diseñado como base del itinerario por el equipo pedagógico (el camino con nivel de dificultad adecuada).

La Figura 34 compara las desviaciones típicas para una colonia de hormigas con diferente perfil, en proporciones de 30%, 60% y 10% de perfiles High, Medium y Low respectivamente. Mientras que a partir de una colonia de 200 hormigas, apenas hay diferencia en la desviación típica, entre 20 y 200 sí se observan diferencias sensibles, mostrando una desviación típica menor en la segunda estrategia de ponderación, lo que significan una menor diferencia entre el número de elecciones de cada camino con respecto a la media (calculada de forma equitativa) y por tanto, se traduce en un bloqueo más tardío, puesto que las elecciones del siguiente nodo se reparten entre los posibles, evitando que exista un excesivo depósito de feromonas en alguno de ellos.

En la mayoría de los casos, el bloqueo se produjo en el camino Node0→Node1→Node5, que es aquél que elige como primer arco a aquél que tiene como nodo destino el de mayor peso (peso 1). Por tanto como primera conclusión podemos deducir que el algoritmo en las primeras iteraciones (a las primeras hormigas de cada colonia) puede asignar cualquier camino de entre los posibles y gracias al depósito de feromonas (que no logran evaporarse totalmente) acaba bloqueándose una vez han finalizado su itinerario un número de hormigas. Este número de hormigas, que oscila entre 20 y 100 para los parámetros del algoritmo utilizados, dependerá de las calificaciones obtenidas por éstas y no puede determinarse *a priori*.

La muestra la evolución de las selecciones de camino en el nodo 0 para la estrategia de ponderación en función del nivel de dificultad.

Para el grafo de la Figura 28, la estrategia consistente en la ponderación según el nivel de dificultad, se ajusta bien con los parámetros utilizados para el algoritmo, mostrándose cambiante para una colonia inferior a las 100 hormigas. Esto es, podría utilizarse en promociones con un número inferior a 100 alumnos, puesto que para grupos más numerosos, el efecto bloqueante de las feromonas aumentaría proporcionalmente el número de malos resultados necesarios para cambiar el itinerario de los alumnos.

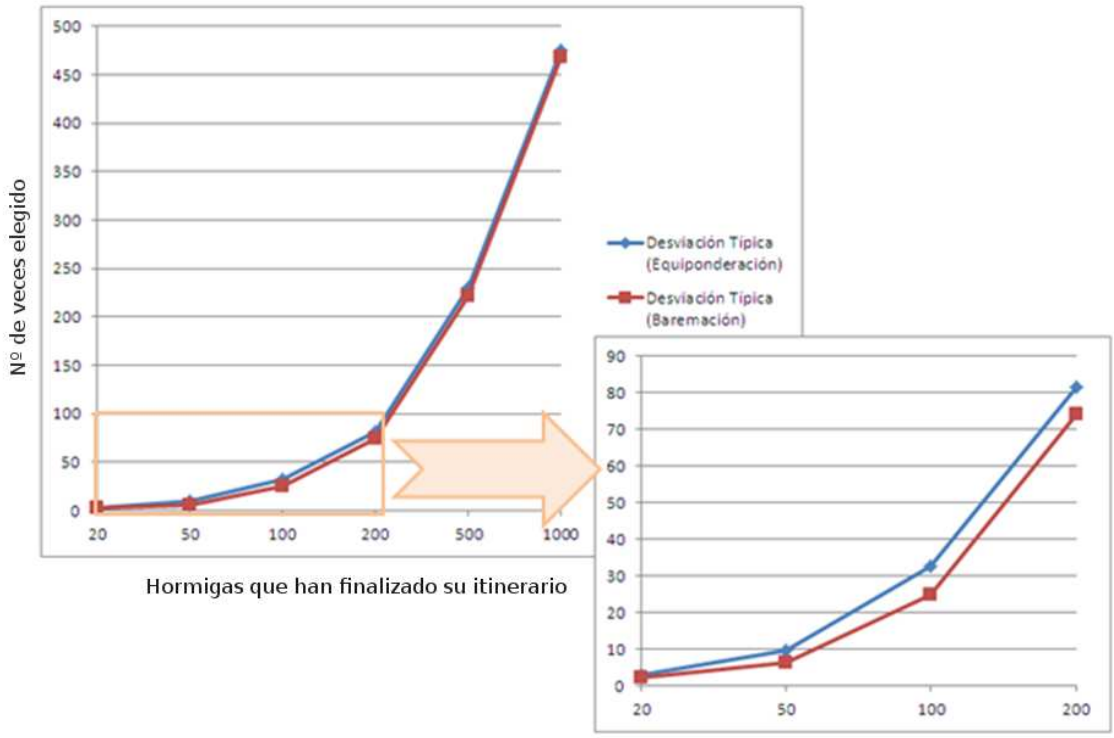


Figura 34: Comparativa entre desviaciones típicas para las dos estrategias de ponderación en colonias con perfiles heterogéneos.

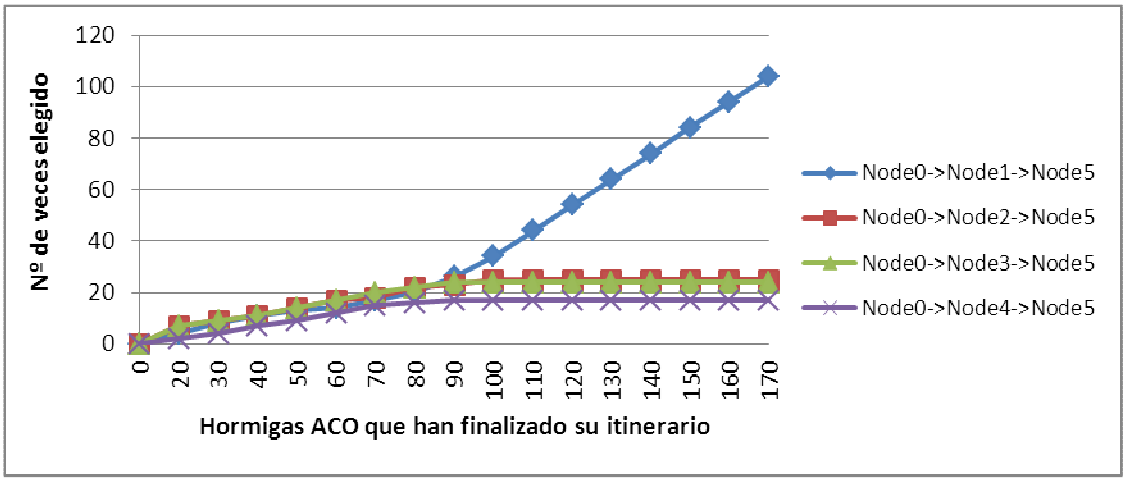


Figura 35: Evolución de una de las ejecuciones de la experimentación para la estrategia de ponderación en función del nivel de dificultad en una colonia con hormigas de diferentes perfiles

Para investigar cómo afecta el número de opciones en un nodo a la rapidez con que se llega a una situación de bloqueo, modificamos el grafo de la Figura 28, aumentando el número de nodos intermedios posibles entre el Nodo0 y el Nodo5, tal y como muestra la Figura 36. La ponderación del grafo se realiza de forma simétrica, de forma que tanto peso como calificación mínima de los nodos añadidos se corresponda con las de los nodos previamente existentes de la siguiente forma:

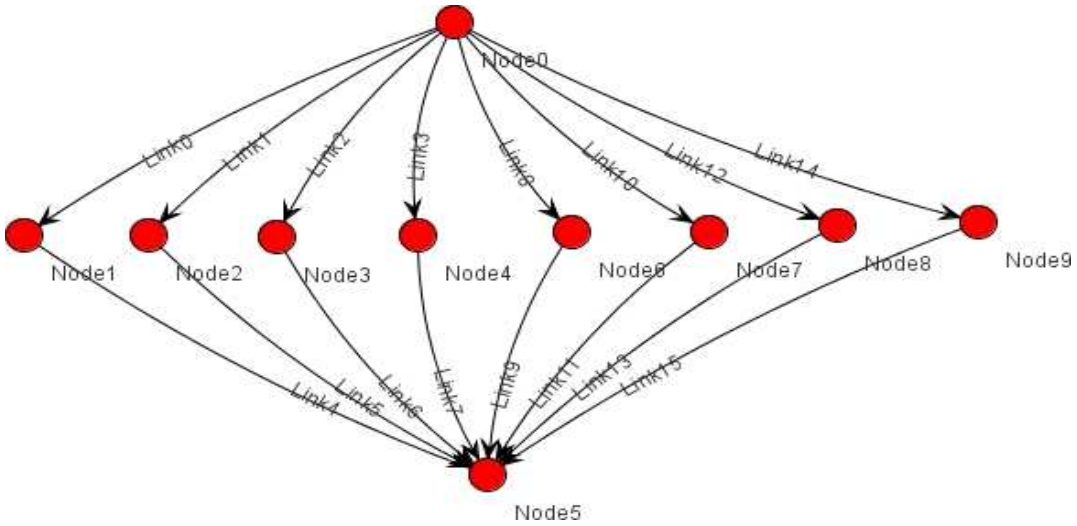


Figura 36: Grafo ponderado con 8 posibles itinerarios

Tabla 12: Ponderación para los nodos añadidos al grafo de la Figura 28

Nodo	W_i	m	M	$\bar{\varepsilon}$	Grado Salida	Arcos de Salida
N9	1	5.0	10.0	-	1	Link15
N8	0.85	6.0	10.0	-	1	Link13
N7	0.70	7.0	10.0	-	1	Link11
N6	0.55	8.0	10.0	-	1	Link9

El comportamiento del algoritmo no varía en exceso con respecto a un menor número de opciones, tal y como puede apreciarse en algunas de las evoluciones de las elecciones de itinerarios de la Figura 37. En esta figura, lo relevante no es identificar cada itinerario en cada gráfica, sino como en todas ellas existe un itinerario que acaba siendo el predominante. En algunas de ellas, se observa como el bloqueo aparece más tarde, respecto al grafo de la Figura 28. En cambio, en otras pruebas el número de hormigas que terminan sus recorridos antes de que se produzca el bloqueo es prácticamente el mismo, por lo que no parece existir una dependencia única entre el número de opciones o itinerarios alternativos y el número de hormigas que necesitan terminar su recorrido para favorecer la aparición de un bloqueo. Es lógico que un mayor número de opciones pueda retrasar la aparición de un bloqueo, al contar la tabla de decisión con más entradas y minimizar la probabilidad de repetición de una misma opción en un corto intervalo de tiempo. No obstante, no existe una implicación directa, pues al depender la cantidad de feromonas a depositar de la calificación obtenida, valor éste último que es impredecible, no se puede asegurar.

Tras este primer estudio podemos concluir que el algoritmo ASALI, al igual que en la vida real sucede con las hormigas, tiende a encontrar una solución que hace que el resto de hormigas virtuales se decanten por ella. El Itinerario es elegido aleatoriamente teniendo en cuenta el éxito obtenido (medido en feromonas) en cada una de las opciones posibles, por lo que los itinerarios de mayor éxito, tendrán más probabilidades de ser elegidos. Para no favorecer que el itinerario elegido coincida con el más fácil, esta función de ajuste tiene en cuenta el peso asignado por el equipo pedagógico que, según resultados experimentales favorece la elección de los cursos de mayor peso.

Para estudiar las propiedades del algoritmo ASALI y como poder ajustar su comportamiento es necesario estudiar la calibración del mismo.

4.2.2 Calibración del Sistema

El comportamiento del algoritmo puede ser modificado mediante las variables α y β que controlan la tendencia del algoritmo a tener un comportamiento más explorador de nuevos caminos o explotador de los ya encontrados. Durante la experimentación, hemos encontrado que esta variación de comportamiento también influye en la proporción de hormigas que sigue los caminos de más éxito. Para calibrar el sistema estudiaremos la variación de la selección de itinerarios para un caso concreto de perfiles de alumnos. En nuestro caso 30% perfil High, 60% perfil Medium y 10% perfil Low.

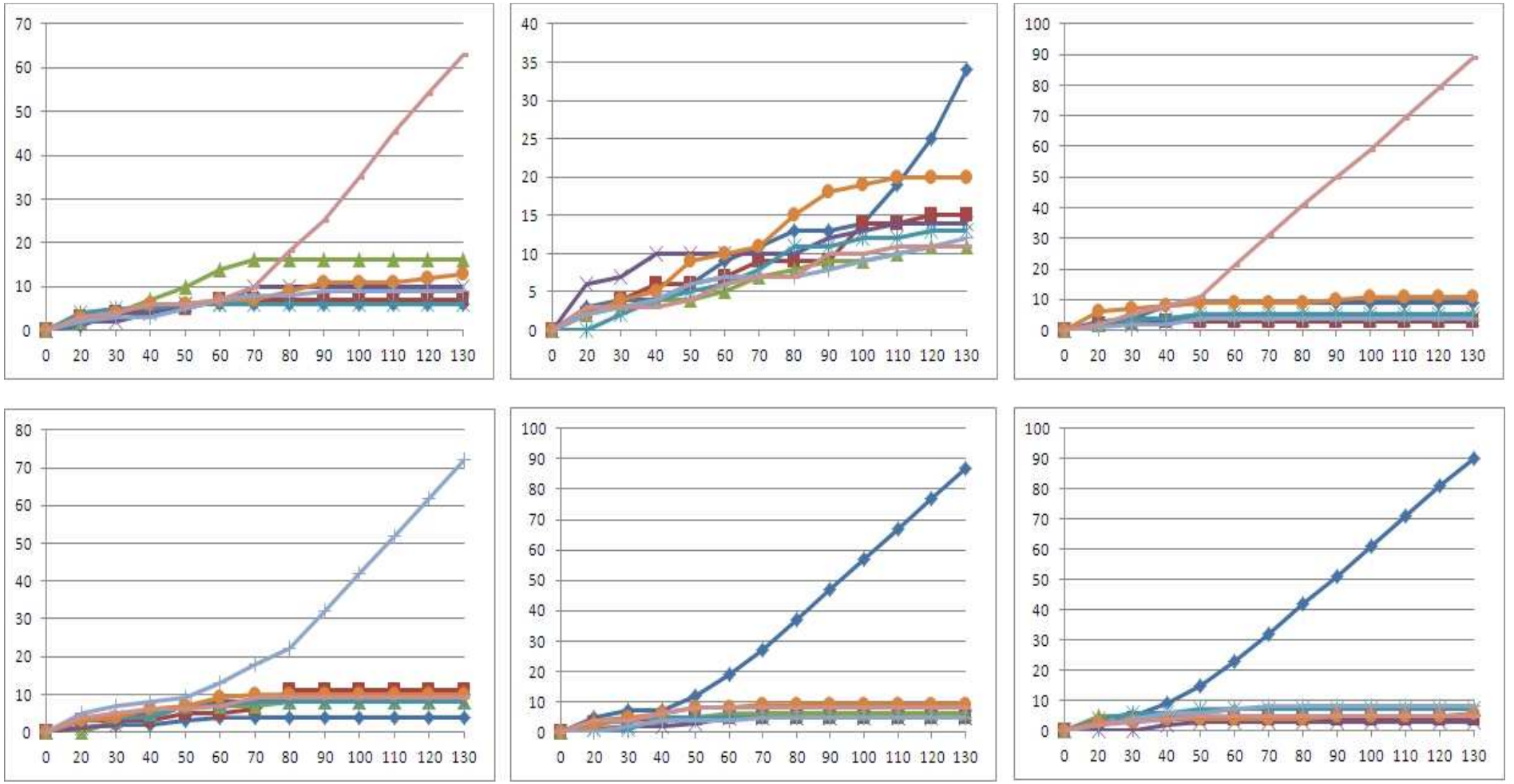


Figura 37: Evolución de las elecciones de cada itinerario en algunas pruebas realizadas con el algoritmo para el grafo de la Figura 36. En el eje de ordenadas se muestra el número de veces que cada itinerario ha sido elegido, y en el de abscisas el número de hormigas ACO que han terminado su itinerario.

Para el estudio de calibración se utilizará el grafo resultante de procesar el Grafo de Itinerarios de Aprendizaje propuesto como escenario de este trabajo (Figura 13) resultando el grafo de la Figura 38:

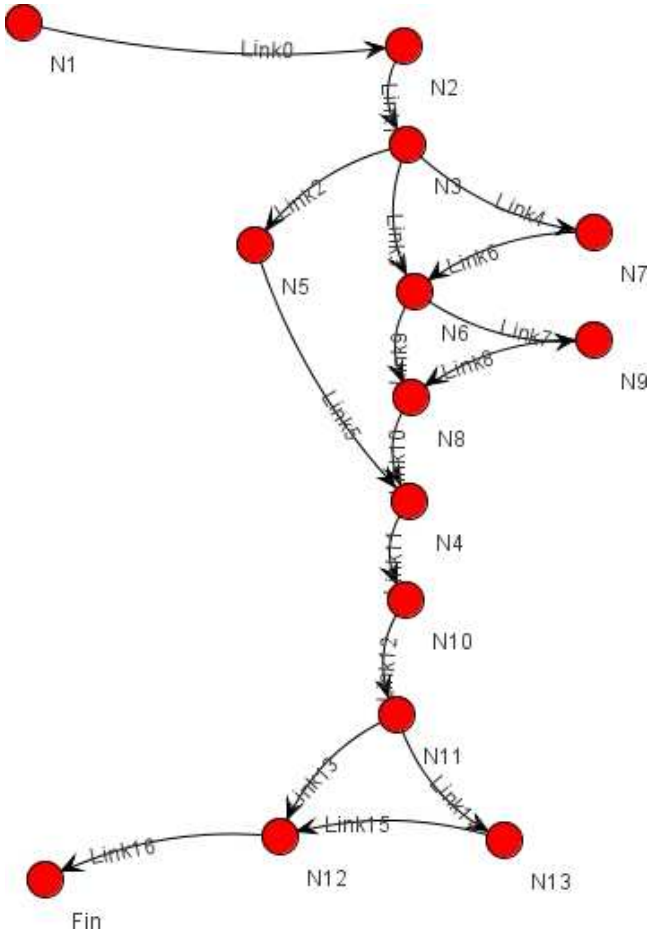


Figura 38: Grafo de itinerarios de aprendizaje una vez eliminadas las restricciones de navegación

La colonia de hormigas estará compuesta por cien hormigas, que simularán el comportamiento de 100 alumnos, que realizarán un recorrido cada una. Tras la ejecución, los pesos de los arcos del grafo quedarán modificados con la función de ajuste calculada para cada uno de ellos. Para una siguiente iteración, se volverá a utilizar el grafo en las condiciones iniciales y no en el estado (funciones de ajuste y feromonas en cada arco) que hubiera quedado tras la iteración anterior. La Tabla 13 muestra los valores iniciales para cada nodo del grafo, siendo los principales puntos de interés en el grafo los llamados Nodos de Decisión: aquellos nodos con grado de salida mayor que uno (*N3*, *N6* y *N11)*.

Tabla 13: Estado inicial de los nodos del grafo

Nodo	W_i	m	M	$\bar{\varepsilon}$	Grado Salida	Arcos
N1	1	5	10	0	1	Link0
N2	1	5	10	0	1	Link1
N3	1	5	10	0	3	Link2, Link3, Link4
N4	1	5	10	0	1	Link11
N5	1	5	10	0	1	Link5
N6	0,5	8	10	0	2	Link7, Link9
N7	0,25	8	10	0	1	Link6
N8	0,75	6	10	0	1	Link10
N9	0,6	7	10	0	1	Link8
N10	1	5	10	0	1	Link12
N11	1	5	10	0	2	Link13, Link14
N12	1	5	10	0	1	Link16
N13	0,7	5	10	0	1	Link15
Fin	-	-	-	-	-	-

Los itinerarios posibles se identifican en la Tabla 14 indicando para cada itinerario la suma de los pesos pedagógicos de los nodos que lo componen.

Tabla 14: Itinerarios posibles

ID	Itinerario	Peso
A	N1→ N2→N3→N6→N9→N8→N4→N10→N11→N12→Fin	9.85
B	N1→N2→N3→N6→N9→N8→N4→N10→N11→N13→N12→Fin	10.55
C	N1→N2→N3→N6→N8→N4→N10→N11→N12→Fin	9.25
D	N1→N2→N3→N6→N8→N4→N10→N11→N13→N12→Fin	9.95
E	N1→N2→N3→N7→N6→N9→N8→N4→N10→N11→N12→Fin	10.10
F	N1→N2→N3→N7→N6→N9→N8→N4→N10→N11→N13→N12→Fin	10.80
G	N1→N2→N3→N7→N6→N8→N4→N10→N11→N12→Fin	9.50
H	N1→N2→N3→N7→N6→N8→N4→N10→N11→N13→N12→Fin	10.20
I	N1→N2→N3→N5→N4→N10→N11→N12→Fin	9.00
J	N1→N2→N3→N5→N4→N10→N11→N13→N12→Fin	9.70

El equipo pedagógico ha diseñado un itinerario base que se corresponde con el itinerario I. Es el itinerario más corto y en el que todos sus nodos tienen peso 1. El resto de itinerarios alternativos son itinerarios de repaso de contenido. Por ejemplo el itinerario C, sustituye el curso del nodo 5 por dos cursos, situados en los nodos N6 y N8. El N6 está enfocado a reforzar el andamiaje de conocimiento necesario e introducir los conocimientos que se presentan en el N5 y el N8 sería una continuación, repitiendo y ampliando los conceptos del N5. Es decir, los nodos N6 y N8 representan la misma carga didáctica que el N5 pero ralentizando el aprendizaje (con más ejemplos, más

ejercicios, más tests, etc). Por tanto el itinerario C es un itinerario más lento que el itinerario I. Lo normal es que las calificaciones en el itinerario C sean ligeramente más alta, lo que se contrarresta con los pesos que el equipo asigna a estos nodos, más bajos. De la misma forma, otros itinerarios posibles plantean más cursos de repaso para contenidos anteriores o de introducción a los siguientes. Por ejemplo el nodo N7 podría ser un repaso de los contenidos del N3, el nodo N9 podría ser una introducción a los contenidos del N8 y el N13 repaso del N11 e introducción a los del N12.

Estudio marginal del parámetro β

Para estudiar el comportamiento del algoritmo según la variación de β, se ejecutará el algoritmo ASALI en el grafo de la Figura 38 con varios valores para el parámetro β (0.5, 1, 2, 5, 10 y 25), realizando además varias ejecuciones con diferentes valores de α: 0, 1, 2, 5, 10, 25 y 50 para comprobar igualmente su influencia.

Recordemos la interpretación de los parámetros α y β:

- El parámetro α es un intensificador de la influencia de las feromonas. Cuando α = 0 éstas no influyen y el algoritmo se convierte en el clásico algoritmo voraz que seleccionará como siguiente nodo al más cercano según la función de distancia definida para el problema, en nuestro caso, según la función de ajuste. En este caso sólo el valor heurístico tiene influencia sobre la probabilidad de selección de cada nodo. Cuando α > 0, las feromonas comienzan a tener efecto en la decisión de los nodos, siendo más determinantes cuanto mayor sea el parámetro α.
- El parámetro β controla la influencia del valor heurístico, que deja de tener influencia cuando β = 0, influyendo en este caso únicamente en la decisión del siguiente nodo las feromonas depositadas previamente en cada arco, lo que conduciría rápidamente a una situación de bloqueo en el camino que aleatoriamente se hubiera seleccionado primero, lo que generalmente lleva a soluciones no óptimas.

En nuestro algoritmo, los elementos de la tabla de decisión se calculan según la expresión (3) y con $\eta_{ij} = f_{ij}$, dónde f_{ij} es la función de ajuste para el arco l_{ij} y se calcula según la ecuación (39).

Aunque las feromonas Φ se evaporan al finalizar la iteración (cuando toda la colonia ha construido una solución) en cambio son depositadas al vuelo, por lo que la

acción de una hormiga puede tener un efecto inmediato sobre las siguientes en la misma iteración. Por tanto el valor de β puede aumentar el impacto de la actuación de la hormiga anterior si β > 1 o minimizarlo β < 1.

En cuanto al parámetro α, no debe tener influencia alguna, puesto que no se realizará más de una iteración por cada prueba y se partirá de un grafo en cuyo estado inicial las feromonas τ son las mismas en todos los arcos y positivas ($\tau_0 = {}^{1}/_{N_i}$). En este caso:

$$a_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[f_{ij}]^{\beta}}{\sum_{l\in N_i}[\tau_{il}(t)]^{\alpha}[f_{il}]^{\beta}} = \frac{[\tau_0]^{\alpha}[f_{ij}]^{\beta}}{[\tau_0]^{\alpha}\sum_{l\in N_i}[f_{il}]^{\beta}} = \frac{[f_{ij}]^{\beta}}{\sum_{l\in N_i}[f_{il}]^{\beta}} \qquad (49)$$

El parámetro β actuará modificando los valores de la tabla de decisión, aumentando las diferencias de probabilidad de selección entre los arcos vecinos, otorgando mayor probabilidad de ser seleccionado a aquel arco cuya función de ajuste sea mayor y disminuyéndola a aquellos cuya función de ajuste sea menor, con respecto a un valor de β menor. Por ejemplo, sean los arcos de la Figura 39:

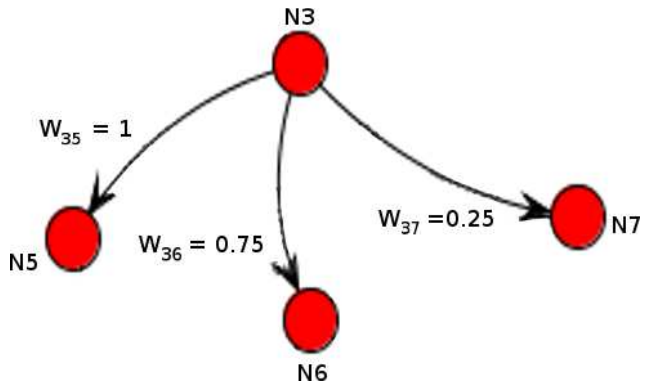


Figura 39: Nodo de decisión para el cálculo de la influencia del parámetro β

En la primera ejecución del algoritmo ASALI sobre un grafo de itinerarios de aprendizaje, el factor de idoneidad de cada arco, al no haber calificaciones disponibles se incializa a 1. Inicialmente las feromonas Φ se inicializan a cero, aunque durante la iteración se irán depositando feromonas. Por tanto la función de ajuste, inicialmente depende en gran medida del peso pedagógico del nodo destino. Aplicando la ecuación (49) la variación inicial de los elementos de la tabla de decisión para los arcos que van del nodo N3 a los nodos N5, N6 y N7 es la que se muestra en la Tabla 15. Esas probabilidades se verían modificadas cada vez que se recalcule la tabla de decisión, es

decir, cada vez que una hormiga se disponga a seleccionar el siguiente nodo, por lo que las acciones de una hormiga influirán en la siguiente.

Tabla 15: Influencia del parámetro β en la probabilidad de decisión de los nodos

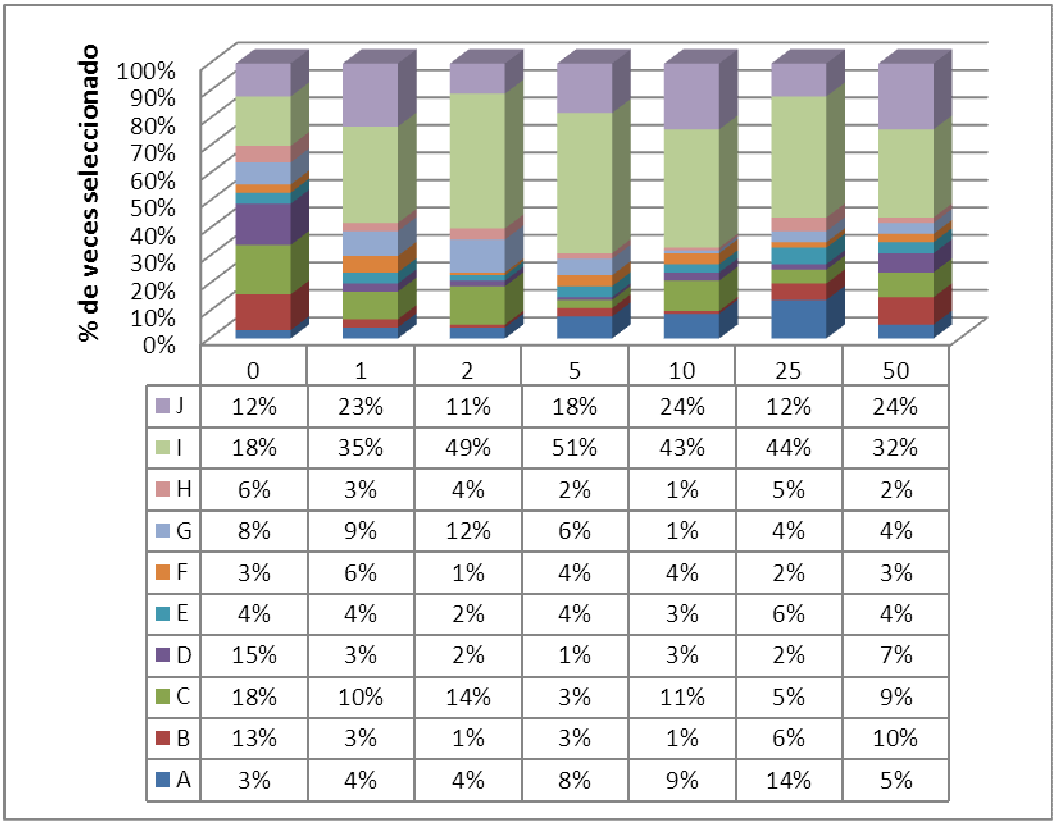
	β = 0.5	β = 1	β = 2	β = 5	β = 10	β = 25
N3→N5	42%	50%	61%	81%	95%	100%
N3→N6	37%	38%	35%	19%	5%	0%
N3→N7	21%	12%	4%	0%	0%	0%
Total	1	1	1	1	1	1

En el grafo utilizado, los arcos que deberían verse beneficiados por el aumento de β son los itinerarios I y J, puesto que son los que cuentan con un nodo destino de peso 1, por lo que al aumentar el valor de β la mayor parte de las hormigas seleccionaría los arcos que van del nodo N3 al nodo N5 y del nodo N11 al N12 en perjuicio de sus respectivos nodos vecinos.

De los resultados obtenidos (Figura 41) no se deduce que la variación de α afecte de alguna manera a la probabilidad de selección de los itinerarios posibles, pues no existe una relación causa-efecto entre el aumento o disminución de α sobre un mismo valor de β en los resultados obtenidos.

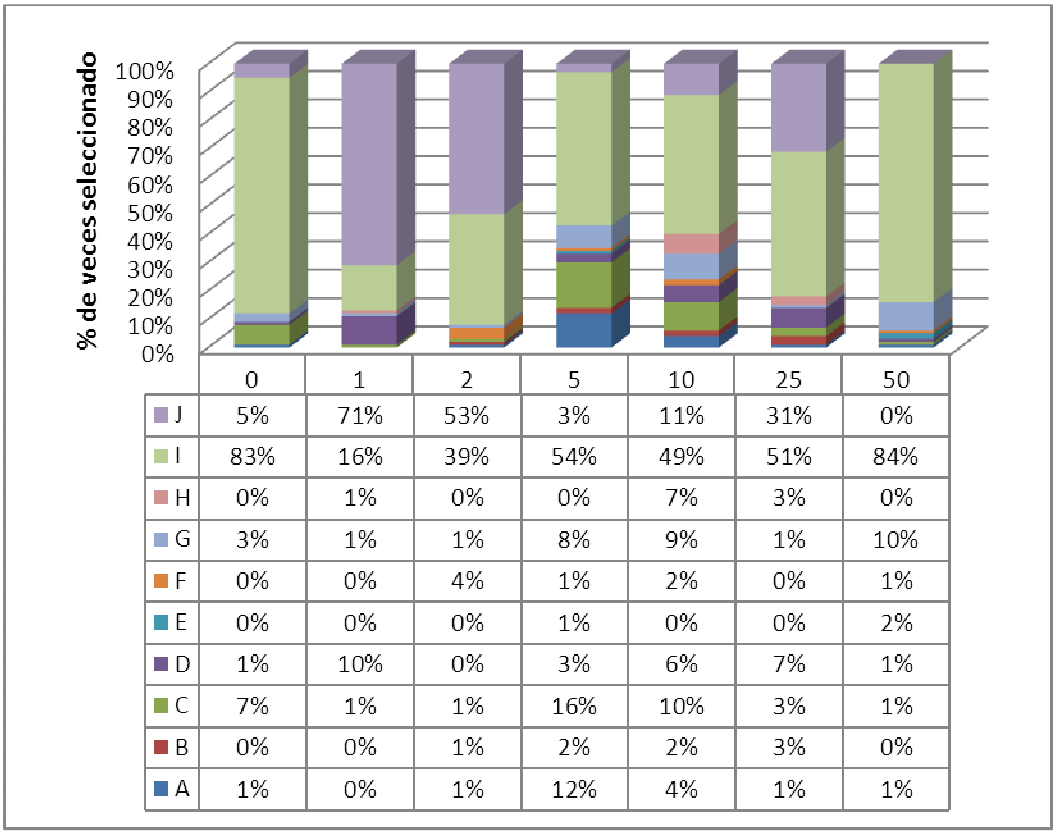
Para β = 0.5, los itinerarios "recomendados" (aquellos en los que el nodo de destino del arco tiene mayor peso), I y J, son los elegidos entre el 30% y el 69% de las veces. Todos los itinerarios son recorridos por al menos una hormiga. En la mayoría de las ocasiones (salvo en una) el reparto de las hormigas por los distintos itinerarios es relativamente homogéneo, no existiendo un único camino que atraiga a la mayoría absoluta (más del 50%) de las hormigas.

Tampoco se observa influencia alguna de la variable α cuando β = 1 (Figura 40). En cambio si se observa en este caso que los itinerarios I y J aglutinan una mayor proporción de hormigas que los recorren, concretamente entre un 57% y el 92%. Como consecuencia, comienzan a aparecer varios itinerarios que no son visitados por ninguna de las 100 hormigas de la colonia, siendo los primeros itinerarios perjudicados aquellos que en los nodos de decisión no incluyen al arco cuyo nodo destino tiene mayor peso. En todas las ejecuciones de esta prueba, independientemente del valor de α, hay al menos un itinerario que no ha sido elegido por ni una sola hormiga.



	0	1	2	5	10	25	50
J	12%	23%	11%	18%	24%	12%	24%
I	18%	35%	49%	51%	43%	44%	32%
H	6%	3%	4%	2%	1%	5%	2%
G	8%	9%	12%	6%	1%	4%	4%
F	3%	6%	1%	4%	4%	2%	3%
E	4%	4%	2%	4%	3%	6%	4%
D	15%	3%	2%	1%	3%	2%	7%
C	18%	10%	14%	3%	11%	5%	9%
B	13%	3%	1%	3%	1%	6%	10%
A	3%	4%	4%	8%	9%	14%	5%

Figura 41: Estudio marginal de β. Reparto de itinerarios para β=0.5



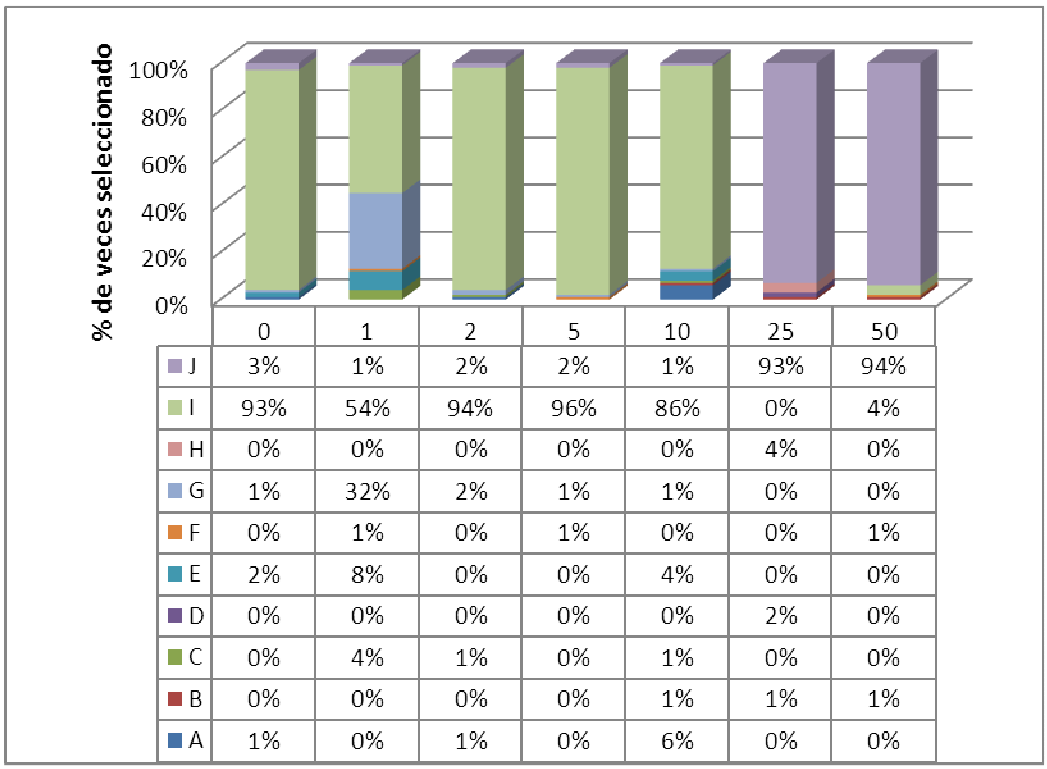
	0	1	2	5	10	25	50
J	5%	71%	53%	3%	11%	31%	0%
I	83%	16%	39%	54%	49%	51%	84%
H	0%	1%	0%	0%	7%	3%	0%
G	3%	1%	1%	8%	9%	1%	10%
F	0%	0%	4%	1%	2%	0%	1%
E	0%	0%	0%	1%	0%	0%	2%
D	1%	10%	0%	3%	6%	7%	1%
C	7%	1%	1%	16%	10%	3%	1%
B	0%	0%	1%	2%	2%	3%	0%
A	1%	0%	1%	12%	4%	1%	1%

Figura 40: Estudio marginal de β. Reparto de itinerarios para β=1

Es frecuente cuando β = 1 que las hormigas alternen entre dos itinerarios (en este caso el I y el J). Cuando β = 2 ya se puede apreciar como los dos itinerarios "mayoritarios" se convierten prácticamente en los únicos elegidos (Figura 43). Tan sólo en una de las siete pruebas el itinerario G consiguió atraer a un porcentaje relevante de hormigas, concretamente a un 32% de la colonia. En el resto de pruebas o bien el itinerario I o bien el J, aglutinaron entre el 86% y el 96% de las elecciones. A diferencia del caso anterior, con β = 2 se observa que las hormigas ya no se dividen entre dos posibles itinerarios, y ahora optan ya claramente por uno sólo de ellos. Puede haber ocasiones en las que las hormigas alternen entre dos itinerarios, como ocurre en la segunda prueba, pero no es frecuente como cuando β=1. Las elecciones de los itinerarios diferentes a los predominantes I y J se producen, en la mayor parte de las veces, al principio de la prueba.

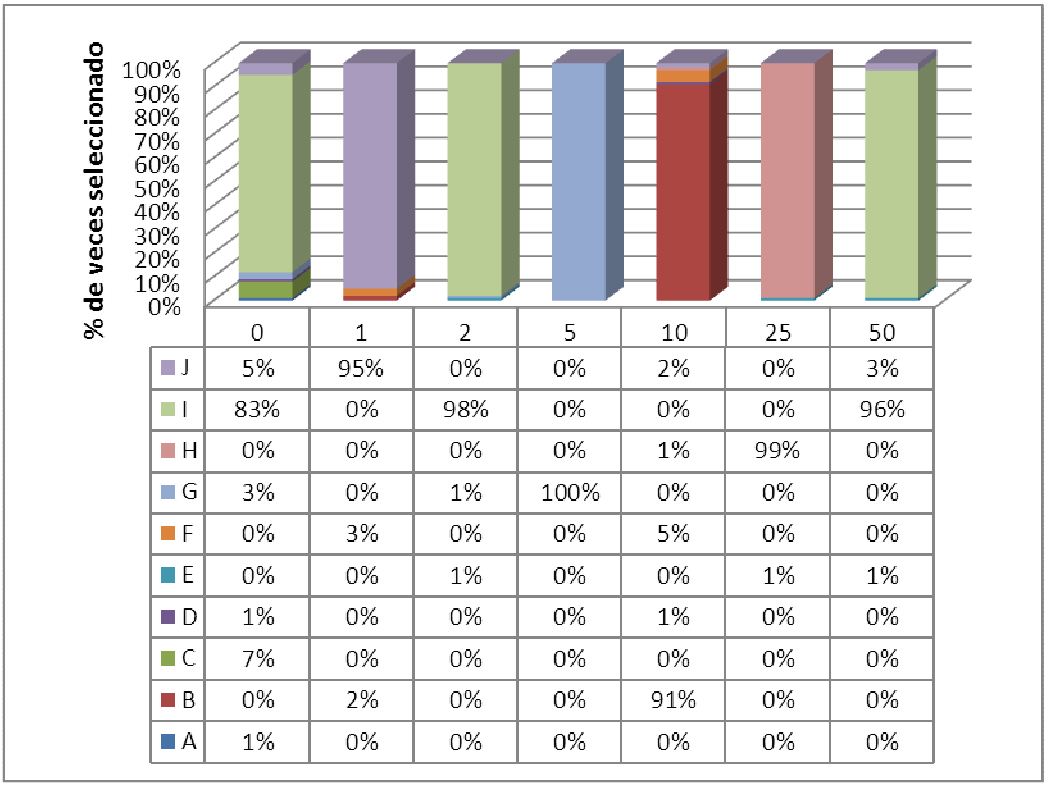
Cuando el valor de β crece hasta β=5 (Figura 42) el impacto de las feromonas Φ es tan fuerte que cualquier itinerario puede convertirse en el predominante sin necesidad de que sea elegido por un alto número de hormigas. Sólo es necesario que la calificación en uno de los nodos destino de un arco que parte de un nodo de decisión sea lo suficientemente alta como para que ese arco tenga más probabilidad de ser elegido (su valor en la tabla de decisión sea bastante más alto que el de los otros arcos). El efecto potenciador de las feromonas Φ por parte de β será tan alto que conducirá al resto de hormigas por este arco. Incluso cuando, debido a la elección aleatoria del nodo, el algoritmo seleccione otro arco que tuviera muy poca probabilidad, si la calificación obtenida en él no es muy alta, lo más probable es que la siguiente hormiga recupere la senda anterior. Por ejemplo es lo que ocurre en el caso en que α = 10. Tras 62 hormigas eligiendo consecutivamente el itinerario B, la siguiente es encaminada por el arco N6→N8 en lugar de por el N6→N9 pero no obtiene una calificación lo suficientemente alta en N8 como para que el arco N6→N8 pase a tener una mayor probabilidad de ser elegido en la tabla de decisión que el N6→N9. La consecuencia es que las siguientes hormigas vuelven a ser encaminada por el N6→N9.

Como consecuencia de este comportamiento, los itinerarios predominantes ya no son únicamente los recomendados, sino que el bloqueo puede producirse en cualquier itinerario, que será el elegido por una proporción de hormigas superior al 80% y que en algunas ocasiones puede ser el 100%. Para valores menores a 1 el algoritmo se muestra excesivamente explorador repartiendo en gran medida a las hormigas por cada uno de los caminos posibles. La convergencia hacia un camino predominante acaba alcanzándose pero muy lentamente. Esta opción podría ser adecuada para utilizarse en un entorno de aprendizaje en el que explícitamente se desea que los alumnos opten por diferentes itinerarios.



	0	1	2	5	10	25	50
J	3%	1%	2%	2%	1%	93%	94%
I	93%	54%	94%	96%	86%	0%	4%
H	0%	0%	0%	0%	0%	4%	0%
G	1%	32%	2%	1%	1%	0%	0%
F	0%	1%	0%	1%	0%	0%	1%
E	2%	8%	0%	0%	4%	0%	0%
D	0%	0%	0%	0%	0%	2%	0%
C	0%	4%	1%	0%	1%	0%	0%
B	0%	0%	0%	0%	1%	1%	1%
A	1%	0%	1%	0%	6%	0%	0%

Figura 43: Estudio marginal de β. Reparto de itinerarios para β=2



	0	1	2	5	10	25	50
J	5%	95%	0%	0%	2%	0%	3%
I	83%	0%	98%	0%	0%	0%	96%
H	0%	0%	0%	0%	1%	99%	0%
G	3%	0%	1%	100%	0%	0%	0%
F	0%	3%	0%	0%	5%	0%	0%
E	0%	0%	1%	0%	0%	1%	1%
D	1%	0%	0%	0%	1%	0%	0%
C	7%	0%	0%	0%	0%	0%	0%
B	0%	2%	0%	0%	91%	0%	0%
A	1%	0%	0%	0%	0%	0%	0%

Figura 42: Estudio marginal de β. Reparto de itinerarios para β=5

En el escenario planteado inicialmente, los valores que mejor se ajustan son los comprendidos entre $\beta = 1$ y $\beta = 2$, puesto que potenciarán los caminos que el equipo pedagógico defina como recomendables, pero a su vez, permitirán la evolución hacia otros caminos si las calificaciones en estos no es inferior a la deseada, sin que por ello se produzca un bloqueo fácilmente en alguno de estos caminos. Cuanto más cercano a $\beta=2$ más rígido al cambio se mostrará el algoritmo y cuanto más cercano a $\beta=1$ más flexible pero también menos propenso a producir un itinerario predominante.

Estudio marginal del parámetro α

Para estudiar la influencia del parámetro α en el comportamiento del algoritmo, repetiremos las mismas pruebas que para el estudio marginal del parámetro β pero conservando el estado de las feromonas del grafo entre cada prueba. Para ello configuramos el algoritmo para que cada hormiga construya 10 soluciones, con lo que simularemos diez ediciones diferentes del mismo curso, al ejecutarse cada iteración de la colonia secuencialmente. De esta forma podremos analizar la influencia de las feromonas τ de una edición en otra. A diferencia de las feromonas Φ que tienen influencia en una misma iteración, las feromonas τ, que se depositan al final de cada iteración, refuerzan el mejor camino de cada iteración con un extra adicional de feromonas. Este mejor camino se corresponde con el itinerario en el que se ha obtenido un mejor rendimiento, es decir una calificación media más alta. Se trata de un refuerzo elitista, que deposita una cantidad extra de feromonas en el recorrido seguido por la hormiga que ha obtenido el mejor resultado en cada iteración, tras haber evaporado las feromonas de todos los arcos.

El valor de α tendrá influencia en la rapidez con la que se cambie la probabilidad de selección de un itinerario a otro entre dos iteraciones diferentes (dos promociones de alumnos en la vida real). En nuestro escenario siempre se requiere de un nodo final único que incluirá una evaluación de los contenidos de toda la programación, independientemente del itinerario seguido, siendo la calificación obtenida en este nodo la que el algoritmo utilice para reforzar el mejor itinerario (aquél itinerario seguido con el que se obtuvo calificación media más alta). Es decir, el mejor itinerario es aquél cuya calificación media es más alta, pero para el refuerzo del itinerario, en lugar de la calificación media se utilizará la calificación obtenida en el nodo final.

Analicemos como influye el valor de $\alpha > 0$ en la selección de un arco. Sea el arco L_{35} el que une los nodos N_3 y N_5 en la Figura 39, al final de cada iteración puede ocurrir que:

- El arco L_{35} pertenezca al itinerario más visitado pero no al itinerario a través del cual se obtuvo la mejor calificación. La evaporación reducirá la cantidad de feromonas τ, lo que por un lado puede reducir el valor correspondiente al arco L_{35} en la tabla de decisión. Sin embargo, al estar L_{35} en el itinerario más veces visitado en esta iteración, la cantidad de feromonas Φ depositadas on-line en él y por tanto su función de ajuste f_{35} será superior a la de los arcos vecinos L_{36} y $L_{37,}$ por lo que la variación de la probabilidad de ser seleccionado en la siguiente iteración dependerá de los valores que siga tomando f_{35} y de cómo está potenciado por β. No se puede predecir un comportamiento claro, al depender la función de ajuste de las calificaciones obtenidas.
- De forma similar al caso anterior, puede suceder que el arco L_{35} sí pertenezca al itinerario con el que mayor calificación se ha obtenido pero no al más visitado. En este caso, tras la evaporación, L_{35} recibirá un aporte extra de feromonas τ, pero su función de ajuste pudiera ser inferior a la de alguno de los arcos vecinos, por lo que la probabilidad de ser elegido en la siguiente iteración dependerá también de los valores de f_{35}, f_{36} y f_{37} así como del valor de β.
- Los únicos casos claros que nos llevan a un valor de probabilidad predecible son aquellos en los que, o bien el arco L_{35} forme parte del itinerario con el que hubiera obtenido la mayor calificación de la iteración y además sea el itinerario más veces recorrido por las hormigas, o bien L_{35} no forme parte ni del itinerario de mejor calificación ni tampoco del itinerario más visitado. En el primero de los casos se aumentará la probabilidad de ser elegido en la siguiente iteración mientras que en el segundo, la probabilidad disminuirá, facilitando la selección de otros caminos. No obstante no es posible asegurar que en la siguiente iteración se mantenga el itinerario anterior como predominante, pues la función de ajuste depende de las calificaciones obtenidas y están sujetas a la evaporación de feromonas al final de la iteración.

Al depender el valor de la función de ajuste de las calificaciones obtenidas en cada nodo, el refuerzo para un camino será tanto más fuerte cuanto mejores calificaciones se hayan obtenido en él, y sobre todo, cuanto mayor sea la diferencia a favor respecto a las calificaciones obtenidas en los nodos vecinos. A continuación se describen los casos descritos anteriormente con datos obtenidos de la experimentación.

El eje de abscisas de ambas gráficas (Figuras Figura 44 y Figura 45), indica el número de hormigas que han acabado su camino. La colonia sigue siendo de cien hormigas y se ejecutan 10 iteraciones, por lo que cada 100 hormigas, se produce una actualización y evaporación de todas las feromonas.

La gráfica de la Figura 44 muestra cómo cuando $\alpha = 0$ las feromonas τ no tienen efecto en la tabla de decisión, y el algoritmo sólo tiene en cuenta en la siguiente iteración a la función de ajuste, una vez evaporadas las feromonas Φ. Esto hace que en la segunda iteración, se produzca un bloqueo en el camino que más fue recorrido en la primera iteración, manteniéndose en sucesivas iteraciones con un porcentaje cercano al 100%.

El efecto de aumentar el valor del parámetro α puede verse en la Figura 45. Si en una iteración la máxima calificación media se obtuvo siguiendo un itinerario *H* diferente al más utilizado, *I*, tras la evaporación de feromonas y el posterior refuerzo en el itinerario *H*, éste itinerario será quien vea aumentada sus probabilidades de ser elegido gracias al efecto de las feromonas τ, que ahora, al ser $\alpha > 0$, sí tienen efecto en la tabla de decisión. En este caso, el arco que va del nodo N_3 al nodo N_7 se ve reforzado respecto al que va del nodo N_3 al nodo N_5.

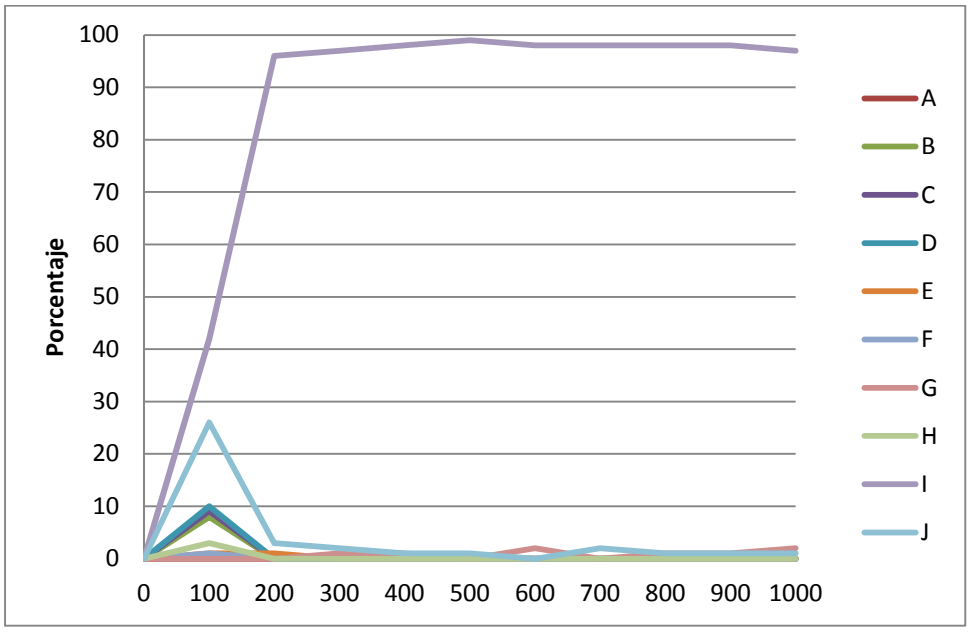


Figura 44: Evolución de la distribución de hormigas por los itinerarios del grafo para α=0 y β=1.5

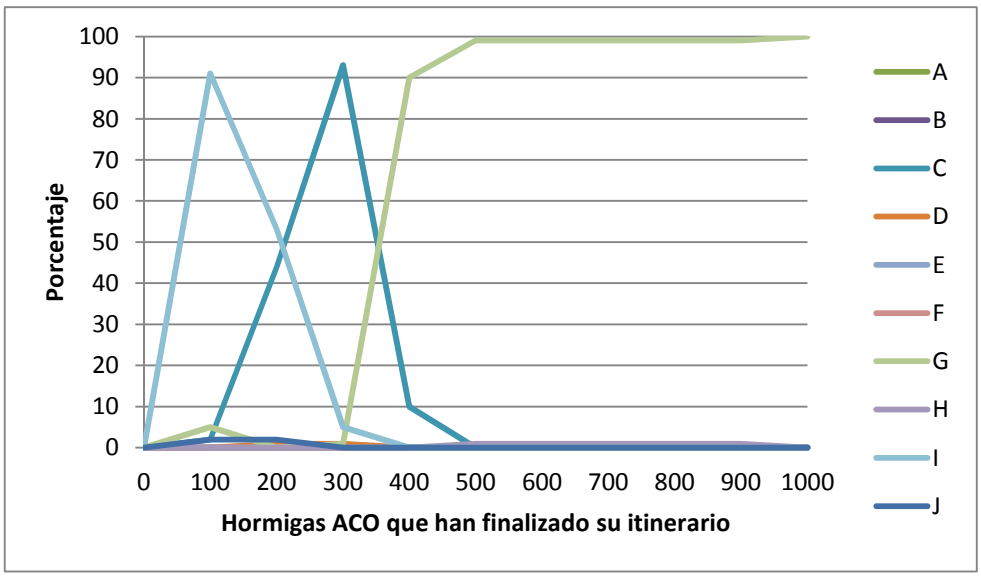


Figura 45: Evolución de la distribución de hormigas por los itinerarios del grafo para α=1.0 y β=1.5

La Tabla 16 muestra los itinerarios seguidos en cada iteración así como el itinerario que obtuvo un mayor refuerzo (por ser el itinerario seguido por la hormiga que obtuvo una calificación media más alta) en cada iteración.

Tabla 16: Itinerarios elegidos en cada iteración, para el experimento de la Figura 45

Mejor Camino	C	C	G	G	H	H	H	H	H	G
Mejor Calificación	7.13	6.73	7.25	7.29	7.26	7.24	7.29	7.26	7.27	7.17
Itinerario	1ª	2ª	3ª	4 ª	5 ª	6 ª	7 ª	8 ª	9 ª	10 ª
A	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0
C	2	44	**93**	10	0	0	0	0	0	0
D	0	1	1	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0
G	5	0	1	**90**	**99**	**99**	**99**	**99**	**99**	**100**
H	0	0	0	0	1	1	1	1	1	0
I	**91**	**53**	5	0	0	0	0	0	0	0
J	2	2	0	0	0	0	0	0	0	0

En la tabla se aprecia como en la primera iteración (primera solución de cada una de las 100 hormigas que componen la colonia), el recorrido predominante fue el del itinerario *I*, elegido por el 91% de las hormigas. Uno de los factores que potencia la

elección de este itinerario, al no existir aún feromonas depositadas en los arcos, es el valor heurístico, en nuestro caso, la función de ajuste, que es mayor cuanto mayor es el peso pedagógico del nodo destino. En este caso, en los nodos de decisión, los arcos $N_3 \rightarrow N_5$ y $N_{11} \rightarrow N_{12}$, se ven favorecidos al contar los nodos N5 y N12 con el máximo peso pedagógico. Una vez finalizada la iteración comienza el proceso de actualización y evaporación de feromonas. El refuerzo se realizará en el mejor itinerario. En este caso, la mejor calificación se obtiene a través del nodo C por lo que los arcos reforzados serán el $N_3 \rightarrow N_6$, y el $N_{11} \rightarrow N_{12}$, evaporándose el 80% de las feromonas del resto de arcos.

En la prueba, podemos ver como en la primera iteración, a pesar de ser *I* el itinerario más transitado, la mayor calificación media se alcanza en el nodo *C*, que al verse reforzado por las feromonas experimenta un crecimiento del porcentaje de elecciones en la siguiente iteración, llegando hasta el 44%. En esta segunda iteración, *I* vuelve a ser itinerario más veces elegido y *C*, repite como el itinerario a través del cual se alcanza la mayor calificación media, aunque en esta ocasión ligeramente inferior a la alcanzada en la primera iteración. Esto provoca que en la tercera iteración, el itinerario *C* sea ya el elegido en un 93% de los casos. Sin embargo, en la tercera iteración, se alcanza a través del itinerario *G* una calificación bastante mejor que la obtenida en las dos iteraciones anteriores, lo que produce un refuerzo suficiente para, a pesar de haber sido elegido el itinerario sólo una vez, aumentar hasta el 90% en la siguiente iteración. Como caso especial, cabe destacar que en las siguientes iteraciones, a pesar de conseguirse la mejor calificación siguiendo el itinerario *H,* esto no tiene ningún impacto especial en siguientes iteraciones. Esto se debe a que la única diferencia entre el itinerario *G* y el *H* es arco a elegir para llegar del nodo N_{11} al N_{12}, o bien vía N_{13} (itinerario H), o bien directamente (itinerario *G)*. Otros itinerarios que también utilizan el arco $N_{11} \rightarrow N_{13}$ son el *B*, el *F* y el *J*. Entre todos ellos, tan sólo suman 4 recorridos por el arco $N_{11} \rightarrow N_{13}$ en las cuatro primeras iteraciones, por lo que la cantidad de feromonas on-line (Φ) depositadas es mínima y además, puesto que el arco fue recorrido en las dos primeras iteraciones, prácticamente ya han desaparecido por efecto de la evaporación, de ahí que su probabilidad de ser seleccionado se siga manteniendo muy baja y tan sólo se elija el itinerario *H* un 1% en cada una de las siguientes iteraciones.

Las gráficas siguientes muestran la evolución de feromonas τ (Figura 46), Φ (Figura 47) y del valor de la función de ajuste, f_{ij} (Figura 48), en el primer nodo de decisión del grafo utilizado durante esta experimentación.

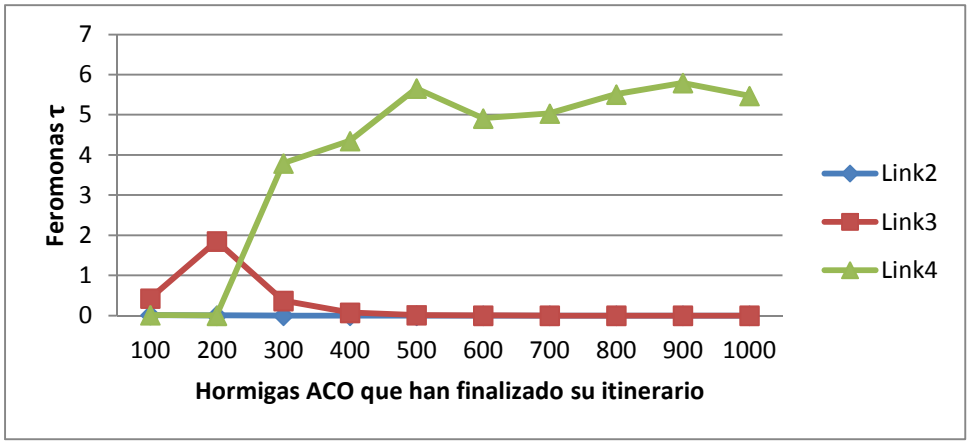


Figura 46: Evolución de feromonas τ en el primer nodo de decisión del grafo de la Figura 38

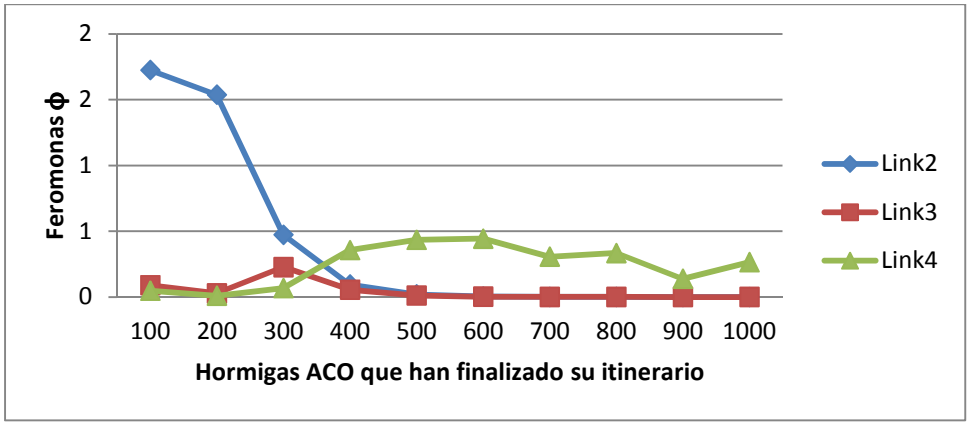


Figura 47: Evolución de feromonas Φ en el primer nodo de decisión del grafo de la Figura 38

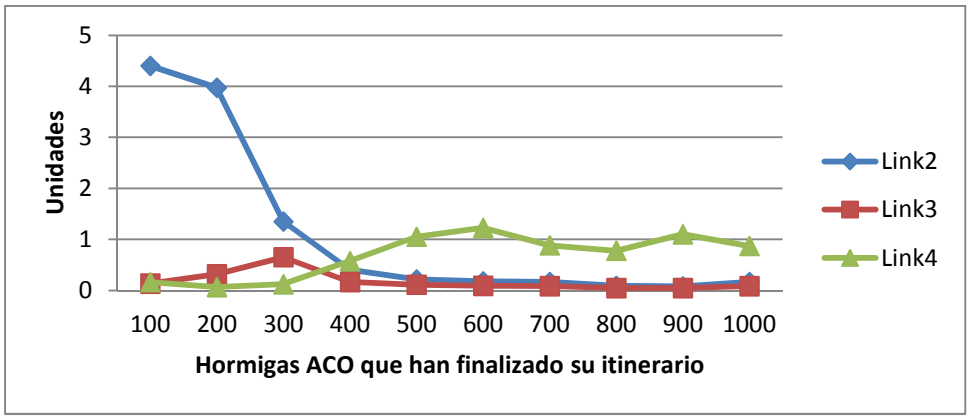


Figura 48: Evolución de la f_{ij} en el primer nodo de decisión de la Figura 38

Se han observado resultados similares en 100 ejecuciones diferentes de la misma prueba, variando únicamente los itinerarios elegidos.

El resultado de aumentar el valor del parámetro α de 0 a 1 tiene como resultado un mayor impacto de la función de ajuste en la tabla de decisión, tal y como se explica a continuación.

La ecuación (3) representa la expresión que calcula el valor de cada elemento a_{ij} de la tabla de decisión, que será el que determine la probabilidad de ser elegido un arco l_{ij}. Teniendo en cuenta el proceso de evaporación y refuerzo del mejor itinerario (aquel por el que se ha obtenido la mejor calificación) con el valor de la función de ajuste, desarrollamos la ecuación:

$$a_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l\in N_i}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} = \frac{\left[(1-\rho)\tau_{ij}(t-1) + f_{ij}\right]^{\alpha}\left[f_{ij}\right]^{\beta}}{\sum_{l\in N_i}[(1-\rho)\tau_{il}(t-1) + f_{il}]^{\alpha}[f_{il}]^{\beta}}$$

que para α=1, se convierte en

$$a_{ij}(t) = \frac{\left[(1-\rho)\tau_{ij}(t-1) + f_{ij}\right]\left[f_{ij}\right]^{\beta}}{\sum_{l\in N_i}[(1-\rho)\tau_{il}(t-1) + f_{il}][f_{il}]^{\beta}} \rightarrow \frac{\left[f_{ij}\right]^{\beta+1}}{\sum_{l\in N_i}[f_{il}]^{\beta+1}} \qquad (50)$$

En el instante inicial, para cualquier arco, el número de feromonas antes de realizar la primera actualización, τ_0, es despreciable. Al final de la primera iteración, cada elemento de la tabla de decisión tiende a depender casi exclusivamente de la función de ajuste (39):

$$f_{ij} = \omega_1 W_j P_{ij} + \omega_2 \Phi_{ij}$$

El valor de esta función de ajuste puede controlarse con las variables de calibración ω_1 y ω_2. Teniendo en cuenta que el primer sumando es menor que 1 si $\omega_1<1$ la influencia del segundo sumando en la magnitud del valor final de f_{ij} puede aproximarse como sigue:

Caso extremo: todas las hormigas eligen el arco l_{ij} en la iteración k. La función de ajuste en ese caso será como máximo:

$$f_{ij} = \omega_1 W_j P_{ij} + \omega_2 \Phi_{ij} = 1 + \omega_2 \phi_{max} \qquad (51)$$

Para la estimación de ϕ_{ij} suponemos que se obtiene la calificación máxima, que normalizada al intervalo [0,1] es $\varepsilon_j = 1$ y que la mínima calificación deseada en el arco de destino es el 50% de la máxima calificación posible ($m_j = M_j/2$). Puesto que $W_j<1$ y $P_{ij}<1$, el mayor valor posible para el primer sumando se alcanza con $\omega_1=1$ y vale 1. Para ello sustituyendo estos valores en la ecuación (38) y utilizando la expresión de la suma total de depósitos de la ecuación (40) obtenemos el valor máximo de feromonas que puede ser depositado en un arco al final de una iteración:

$$\phi_{max} = \sum_{k=1}^{n} \phi_{max}^{k}(\varepsilon_{max}) = \sum_{k=1}^{n} \varepsilon_j^k \varphi + \omega_3 \left(\varepsilon_j^k - \frac{m_i}{M}\right) = \sum_{k=1}^{n} \varphi + \frac{\omega_3}{2} \tag{52}$$

Puesto que todas las hormigas de la colonia han escogido el mismo arco y han obtenido la máxima calificación en curso del nodo destino del arco l_{ij}, la cantidad de feromonas Φ máxima en ese arco será:

$$\Phi_{max} = N \left(\varphi + \frac{\omega_3}{2}\right) \tag{53}$$

donde N es el número de hormigas de la colonia y φ es un valor constante que simboliza la unidad mínima de feromonas a depositar. Suponiendo el caso más extremo (mayor valor de la función de ajuste) y sustituyendo en la ecuación (51):

$$f_{ij} = \omega_1 W_j P_{ij} + \omega_2 \Phi_{ij} = 1 + \omega_2 N \left(\varphi + \frac{\omega_3}{2}\right) \tag{54}$$

La ecuación (54) se utilizará como base para la calibración del algoritmo a través de las variables ω_1, ω_2 y ω_3. Una vez establecido $\omega_1=1$, para que el segundo sumando de la ecuación de la función de ajuste, las feromonas depositadas on-line por cada hormiga, no tenga una influencia excesiva con respecto al peso pedagógico, se decide igualarlo al máximo caso posible, esto es a 1:

$$1 = \omega_2 N \left(\varphi + \frac{\omega_3}{2}\right) \tag{55}$$

Y por tanto, podemos definir una variable en función de la otra como:

$$\omega_2 = \frac{1}{N \left(\varphi + \frac{\omega_3}{2}\right)} \tag{56}$$

Con una hoja de cálculo es fácil calcular varios valores posibles conociendo la población de la colonia (que se haría coincidir con la población real de alumnos) y el valor unitario de una feromona, que en nuestro caso es constante y se ha definido como φ=0.01. Por ejemplo para una colonia de 100 hormigas, la Tabla 17 muestra los valores adecuados para ω_2 y ω_3.

Tabla 17: Valores guía para la calibración fina del algoritmo ASALI con las variables ω_2 y ω_3 para una colonia de 100 hormigas y con un valor de φ=0.01

ω_2	ω_3
1.0000	0.00
0.2857	0.05
0.1666	0.10
0.1176	0.15
0.0909	0.20
0.0740	0.25
0.0384	0.50
0.0259	0.75
0.0196	1.00
0.0157	1.25
0.0131	1.50
0.0112	1.75
0.0099	2.00
0.0088	2.25
0.0079	2.50
0.0072	2.75
0.0066	3.00
0.0061	3.25
0.0056	3.50
0.0053	3.75
0.0049	4.00
0.0047	4.25
0.0044	4.50
0.0042	4.75
0.0040	5.00

Con ω_3 puede regularse el refuerzo en función de la calificación obtenida en un nodo, mientras que con ω_2 puede darse más o menos importancia al número de hormigas que han escogido un arco. Así, aumentando ω_2 potenciamos los arcos que más veces se han recorrido, mientras que aumentando ω_3 y manteniendo ω_2 bajo estaremos potenciando, los arcos que han obtenido una mayor calificación.

Para seleccionar unos valores de calibración adecuados, conociendo la población de la colonia, realizamos un número de simulaciones suficiente (en nuestro caso 20 simulaciones para cada valor) y analizamos el comportamiento del algoritmo en función de los valores obtenidos. A continuación mostramos algunos comportamientos típicos para diferentes combinaciones.

Para los valores ω_1=1; ω_2=0,02 y ω_3=1, el impacto de obtener la mejor calificación en un nodo diferente al más elegido en una iteración es prácticamente nulo, tal y como puede apreciar en los datos de la Tabla 18, representados a su vez en la Figura 49. Para intentar potenciarlas mejores calificaciones, aumentamos el valor de ω_3 hasta 2.

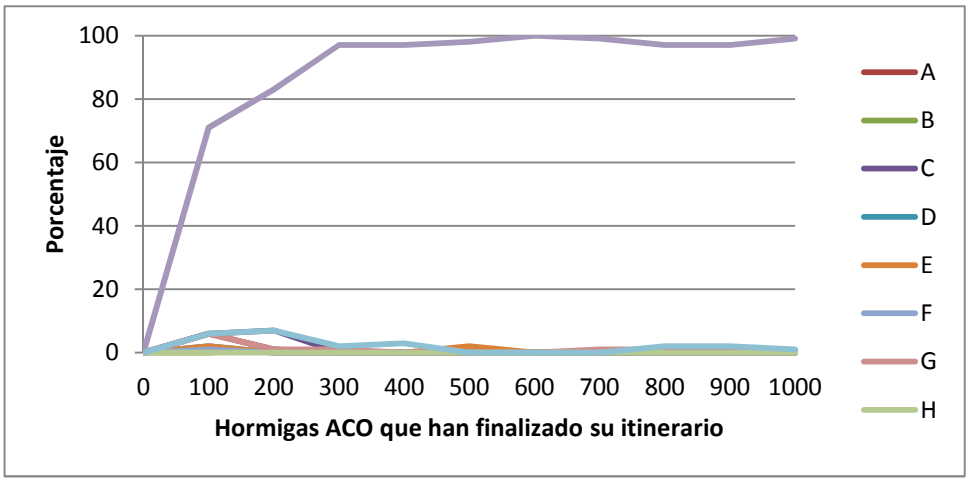


Figura 49: Evolución del porcentaje de elección de cada itinerario en cada iteración. En cada iteración participan 100 hormigas. α=1.0; β=1.5; ω_1=1.0; ω_2=0.02; ω_3=1.0.

Tabla 18: Datos representados en la Figura 49

Mejor Camino -->	E	G	G	J	E	I	G	G	G	J	
Mejor Calificación -->	8,96	7,08	7,11	6,48	7,36	6,19	7,13	7,14	7,10	6,43	
Itinerario	100	200	300	400	500	600	700	800	900	1000	Total
A	6	1	0	0	0	0	0	0	0	0	7
B	0	1	0	0	0	0	0	0	0	0	1
C	6	7	0	0	0	0	0	0	0	0	13
D	2	0	0	0	0	0	0	0	0	0	2
E	2	0	0	0	2	0	0	0	0	0	4
F	1	0	0	0	0	0	0	0	0	0	1
G	6	1	1	0	0	0	1	1	1	0	11
H	0	0	0	0	0	0	0	0	0	0	0
I	**71**	**83**	**97**	**97**	**98**	**100**	**99**	**97**	**97**	**99**	938
J	6	7	2	3	0	0	0	2	2	1	23

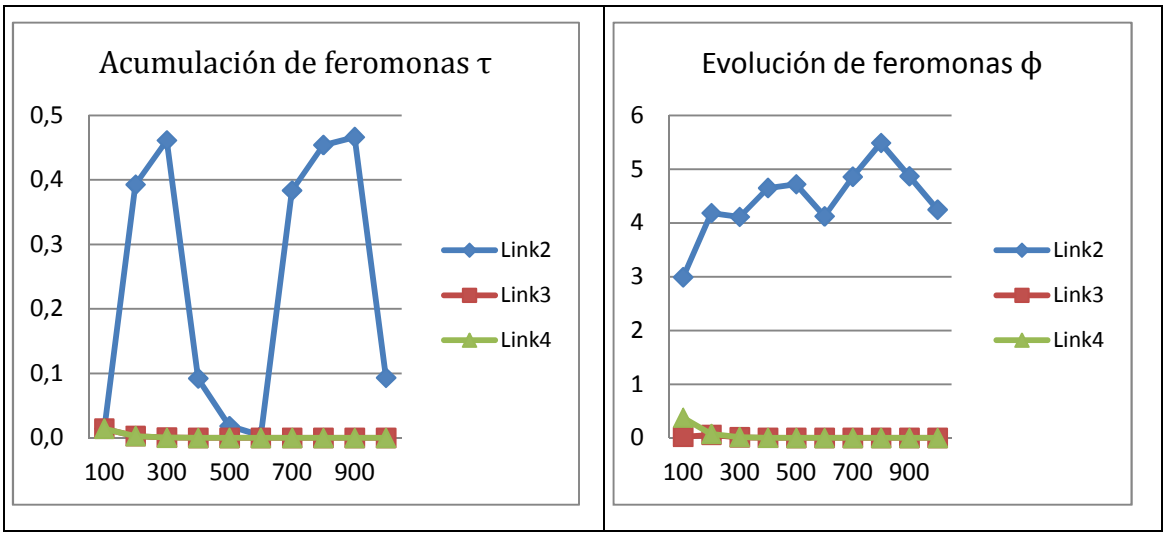


Figura 50: Evolución de feromonas τ y ϕ en cada iteración. α=1.0; β=1.5; ω_1=1.0; ω_2=0.02 y ω_3=1.0

Como se puede observar en la gráfica de la función de ajuste de cada arco en el primer nodo de decisión (Figura 51), cuando el valor de la función de ajuste de uno de los arcos disminuye, aumentan la del resto de nodos.

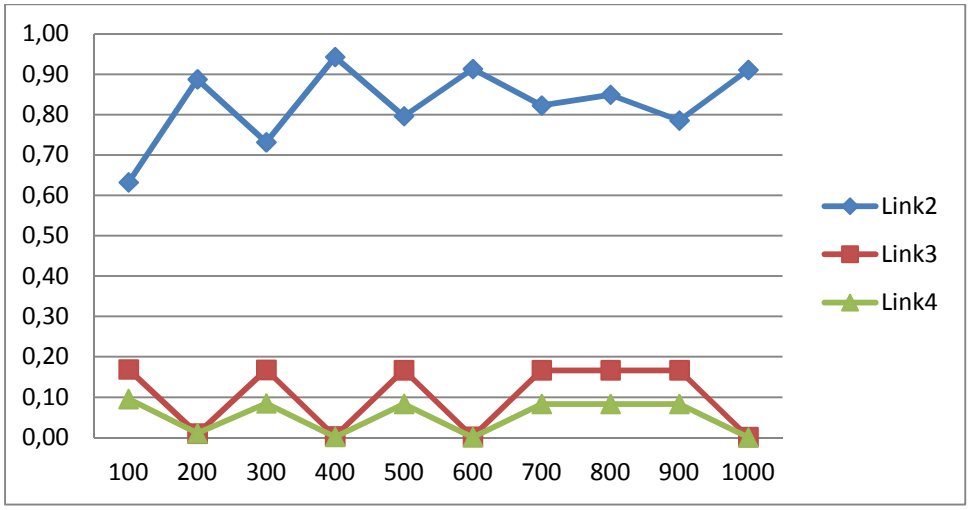


Figura 51: Evolución de la función de ajuste en 10 iteraciones con α=1.0; β=1.5; ω_1=1.0; ω_2=0.02 y ω_3=1.0

Tras aumentar el valor de ω_3 hasta ω_3=2, observamos el comportamiento deseado, que refuerza significativamente al camino por el que se obtiene la mejor calificación, favoreciendo la posibilidad de cambio en el itinerario "preferido" por el algoritmo. La Figura 52 representa gráficamente los datos de la Tabla 19, en la que

podemos observar como en las tres primeras iteraciones la mejor calificación se obtiene en el itinerario G, que implica la elección del arco que va del nodo N3 al nodo N7. A pesar de no ser G el itinerario elegido en mayor número de veces en las dos primeras iteraciones (200 primeras hormigas), el hecho de alcanzar la máxima calificación en cada iteración a través de él, aumenta en la siguiente iteración la probabilidad de ser elegido, lo que consigue en la tercera iteración.

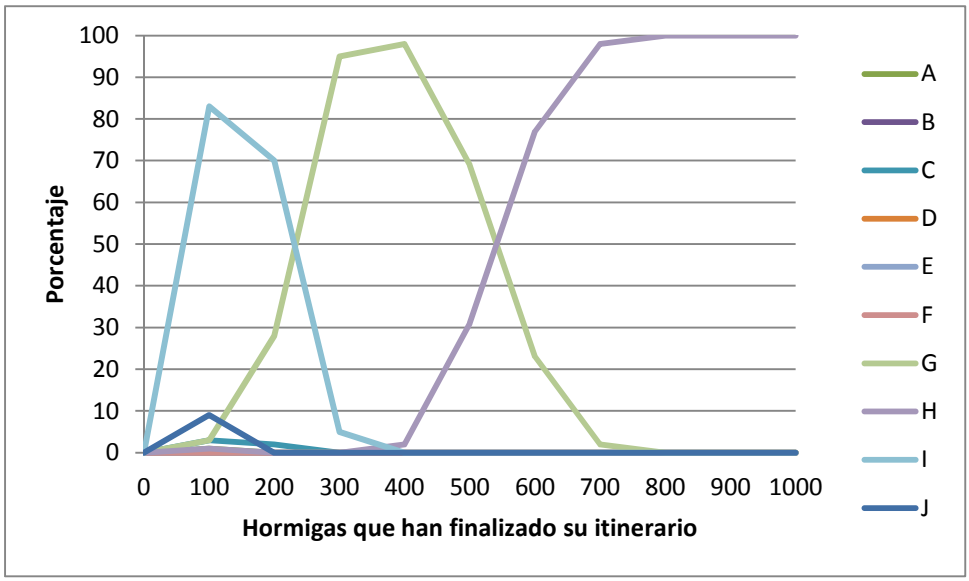


Figura 52: Evolución del porcentaje de elección de cada itinerario en cada iteración. En cada iteración participan 100 hormigas. α=1.0; β=1.5; ω_1=1.0; ω_2=0.02 y ω_3=2.0.

Tabla 19: Datos representados en la Figura 52

Mejor Camino	G	G	G	H	H	H	H	H	H	H	
Mejor Calificación	7.16	7.00	7.13	7.38	7.39	7.40	7.41	7.43	7.42	7.42	
ID	100	200	300	400	500	600	700	800	900	1000	Total
A	1	0	0	0	0	0	0	0	0	0	1
B	0	0	0	0	0	0	0	0	0	0	0
C	3	2	0	0	0	0	0	0	0	0	5
D	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0
G	3	28	**95**	**98**	**69**	23	2	0	0	0	318
H	1	0	0	2	31	**77**	**98**	**100**	**100**	**100**	509
I	**83**	**70**	5	0	0	0	0	0	0	0	158
J	9	0	0	0	0	0	0	0	0	0	9

Este fenómeno se vuelve a repetir con el itinerario H en la cuarta iteración, consiguiendo aumentar la probabilidad de elección progresivamente y minimizando la del nodo G, desde un 98% en la cuarta iteración hasta un 69% en la quinta y 23% en la

sexta. Como contrapartida, frente al comportamiento del algoritmo con ω_3=1, podemos destacar la mayor concentración en la elección observada en cada iteración. A diferencia de los experimentos realizados con ω_3=1, en los que se observaba gran diversidad en la elección de los caminos en las primeras iteraciones, con ω_3=2, el número de itinerarios que en la primera iteración no han sido elegidos por ninguna hormiga, ha aumentado. Es decir, el comportamiento del algoritmo es más explotador de soluciones que explorador. Otra característica observada con el aumento de esta variable de calibración es una mayor rapidez para alcanzar el bloqueo (todas las hormigas eligen el mismo itinerario).

Las gráficas de la Figura 53 muestran la evolución de las feromonas τ y ϕ a lo largo de las 10 iteraciones de los test. La Figura 54 muestra la evolución de la función de ajuste, f_{ij}, para los valores indicados.

A partir de ω_3=2, la influencia de las feromonas ϕ depositadas en función de la calificación es excesiva y el bloqueo se produce prácticamente desde la primera iteración, una vez que se produce la actualización y evaporación de feromonas. La explicación es que la suma de las feromonas ϕ acumuladas durante la primera iteración es mucho mayor al refuerzo obtenido al itinerario de mayor calificación, por lo que éste refuerzo deja de tener influencia, bloqueándose el algoritmo en el itinerario más veces seleccionado en la primera iteración.

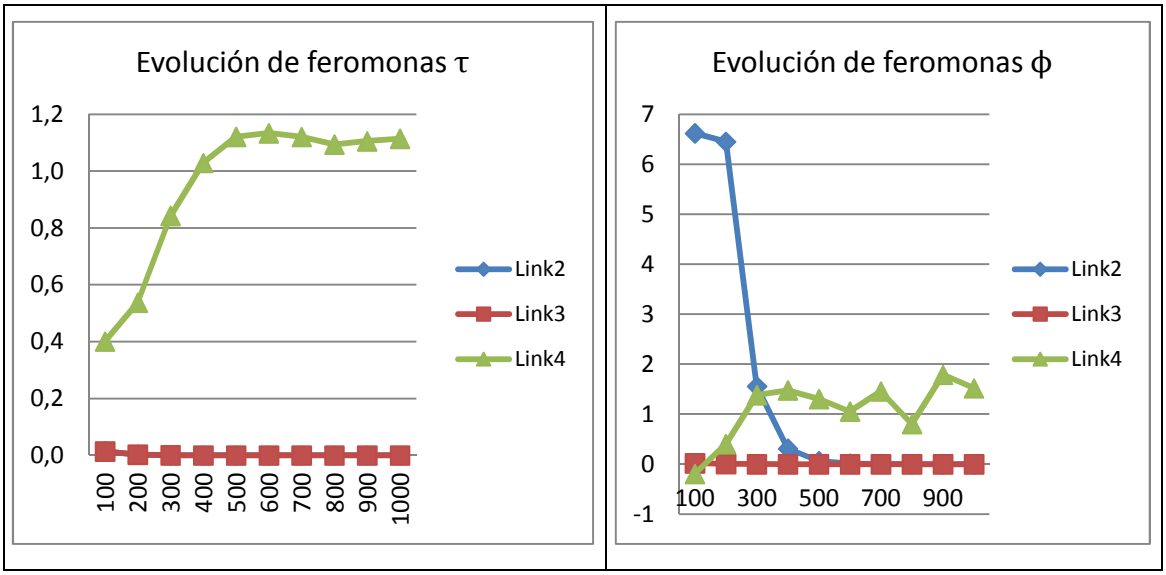


Figura 53: Evolución de feromonas τ y ϕ en cada iteración. α=1.0; β=1.5; ω_1=1.0; ω_2=0.02 y ω_3=2.0

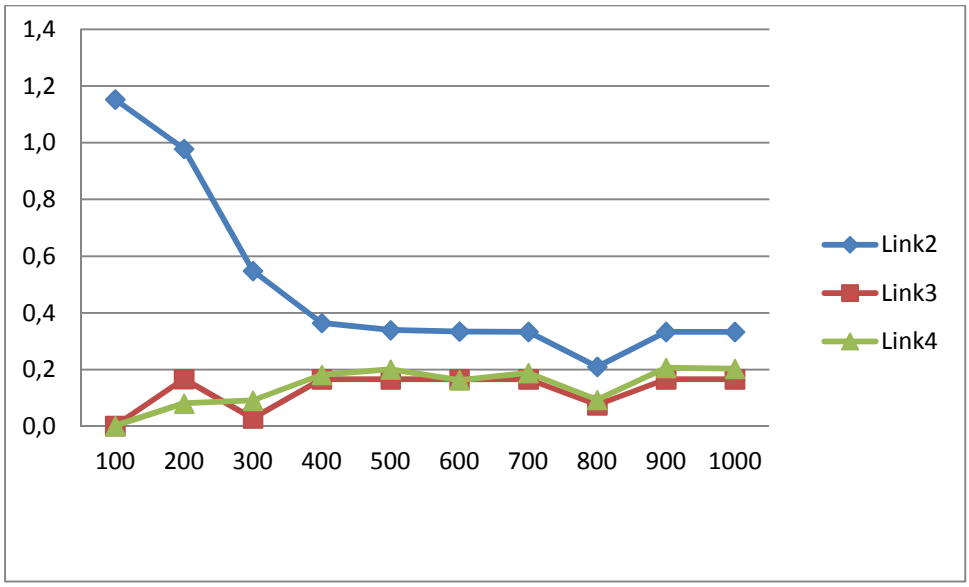


Figura 54: Evolución de la función de ajuste en 10 iteraciones. α=1.0; β=1.5; ω_1=1.0; ω_2=0.02 y ω_3=2.0

Tras una larga fase de experimentación, en la que se simularon más de 25 veces para cada valor de las variables de calibración, obtenemos las siguientes conclusiones:

- Partiendo de ω_1=1, aumentar el valor de ω_1 aumenta la importancia de los pesos pedagógicos y el factor de idoneidad, frente al impacto de las feromonas.
- Partiendo de ω_2=0,02 y ω_3=1, aumentar ω_3 hasta un máximo de ω_3=2 fortalece en los nodos de decisión la probabilidad de selección del arco que pertenece al itinerario a través del cual se obtuvo la mayor calificación en la iteración anterior.
- Aumentar ω_2 a un valor mayor que 1, tiene el mismo efecto que aumentar ω_3 con un valor superior a 2. El bloqueo se produce desde la primera iteración en el itinerario más veces seleccionado.

Llegados a este punto de la experimentación, los valores seleccionados son: α=1.2, β=1.5, ω_1=1, ω_2=0.02 y ω_3=1.5.

Como último estudio, resta analizar la influencia de la tasa de evaporación.

Estudio marginal de la tasa de evaporación

La tasa de evaporación utilizada es ρ=0.8, lo que significa que el 80% de las feromonas se perderán por el efecto de la evaporación, permaneciendo en cada arco, sólo un 20% de las existentes en el instante anterior. En nuestro algoritmo, el tiempo se computa mediante iteraciones, y las hormigas inician la construcción de su solución de forma secuencial. Cuando todas las hormigas de la colonia han construido su solución se produce la actualización y evaporación de las feromonas.

Tras la experimentación se comprueba que para valores pequeños de la tasa de evaporación, la primera iteración es determinante, conduciendo al bloqueo del algoritmo, gracias al mantenimiento de un mayor número de feromonas que condiciona la elección de los arcos en la siguiente iteración. Este comportamiento comienza a minimizarse con valores de ρ>0.5. Para valores de 0.5<ρ<0.8 el algoritmo se comporta como deseamos, facilitando, sobre todo en las primeras iteraciones, cambios de itinerarios en función de las calificaciones obtenidas.

La Tabla 20, cuyos valores se representan en la Figura 55, muestra el número de hormigas que elige cada itinerario durante la ejecución de una simulación, teniendo en cuenta los valores de los parámetros α, β y variables de calibración (ω_1, ω_2 y ω_3) elegidos anteriormente y una tasa de evaporación del 50%.

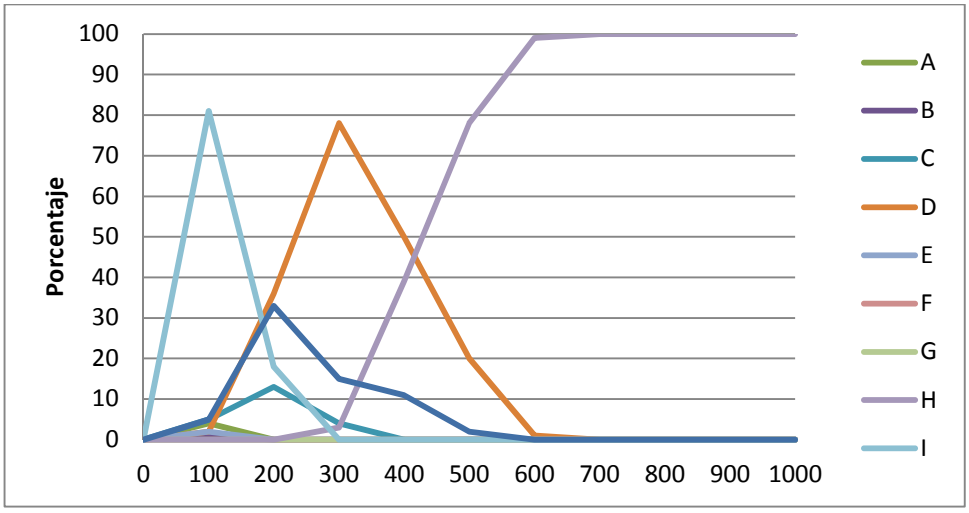


Figura 55: Ejemplo de comportamiento del algoritmo ASALI para los valores α=1.2, β=1.5, ω_1=1.0, ω_2=0.02, ω_3=1.5 y ρ=0.5

Tabla 20: Número de hormigas que eligen cada itinerario por iteración para los valores α=1.2, β=1.5, ω_1=1.0, ω_2=0.02, ω_3=1.5 y ρ=0.5 en una prueba durante la experimentación.

Mejor Camino	D	D	H	H	H	H	H	H	H	H	
Mejor Calificación	6.50	6.90	7.33	7.43	7.40	7.39	7.40	7.41	7.40	7.40	
ID	100	200	300	400	500	600	700	800	900	1000	Total
A	4	0	0	0	0	0	0	0	0	0	4
B	1	0	0	0	0	0	0	0	0	0	1
C	5	13	4	0	0	0	0	0	0	0	22
D	2	**36**	**78**	**50**	20	1	0	0	0	0	187
E	2	0	0	0	0	0	0	0	0	0	2
F	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0
H	0	0	3	39	**78**	**99**	**100**	**100**	**100**	**100**	619
I	**81**	18	0	0	0	0	0	0	0	0	99
J	5	33	15	11	2	0	0	0	0	0	66

La tabla anterior muestra un comportamiento del algoritmo que permite una adaptación a lo largo del tiempo según el rendimiento de los alumnos. Salvo en la primera iteración, para la que no se cuenta con información que pueda condicionar la elección de uno u otro itinerario, el resto se ve influenciado por las iteraciones anteriores. Así en la primera iteración, aunque el itinerario I fue el más veces recorrido, la máxima calificación fue obtenida por una hormiga que recorrió los arcos del itinerario D, por lo que éstos obtuvieron un refuerzo extra que sirvió para incrementar el número de hormigas que eligieron el itinerario D en la segunda iteración.

Cabe destacar que los receptores de este refuerzo son los arcos del itinerario. Este refuerzo condiciona la decisión de los itinerarios posibles en un nodo de decisión, reforzando las posibilidades de los itinerarios comunes al arco elegido. Esto explica el aumento del número de hormigas que eligen el itinerario C que se observa en la segunda iteración, al verse este itinerario beneficiado del refuerzo los arcos $N_3 \rightarrow N_6$ y $N_6 \rightarrow N_8$ comunes a los dos itinerarios. De la misma forma, el refuerzo obtenido por el arco $N_{11} \rightarrow N_{13}$ también aumenta la probabilidad de ser elegido a los itinerarios que, además del D, contienen a este arco: B, F, H y J. En este sentido, el itinerario J se beneficia de este refuerzo y en menor medida de las feromonas que, tras la evaporación de feromonas al final de la primera iteración, aún quedan en los arcos del itinerario I (el elegido en un mayor número de veces), con el que coincide totalmente hasta llegar al nodo de decisión que conforma el nodo N_{11} con los nodos N_{12} y N_{13}.

A partir de la sexta iteración, el nivel de feromonas acumulados en los arcos del nodo H, provoca el bloqueo del algoritmo (siempre elige el mismo itinerario) debido

a que la cantidad de feromonas τ y ϕ acumuladas provoca que la probabilidad de decisión se maximice en uno de los itinerarios (Figura 56).

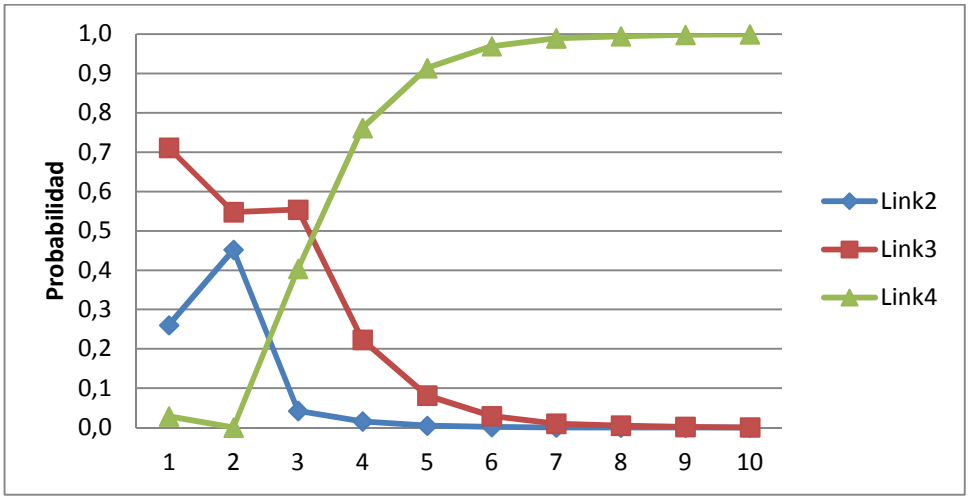


Figura 56: Probabilidad de decisión para los arcos Link2 (N3→N5), Link3 (N3→N6) y Link4 (N3→N7) en el experimento anterior.

Aunque el comportamiento depende de las calificaciones obtenidas en cada nodo durante cada iteración, la convergencia del algoritmo a una solución puede suavizarse aumentando el valor de la variable de calibración ω_1, lo que dará más peso al peso pedagógico frente a las feromonas, o disminuyendo el valor de ω_3 hasta alcanzar valores inferiores al correspondiente al de la combinación indicada en la Tabla 17 para un valor de ω_2 dado.

4.2.3 Estudio de la adaptabilidad

La adaptación del itinerario depende de una secuencia de resultados (calificaciones) y de la cantidad de feromonas existentes en cada arco. Dependiendo de estos valores, la probabilidad de elegir un nodo u otro variará, permitiendo al algoritmo cambiar a otro itinerario. Con este estudio de adaptabilidad se pretende demostrar que el algoritmo puede evolucionar y cambiar de forma autónoma.

Metodología

La primera fase consiste en simular la finalización del curso por parte de un conjunto de alumnos, al que llamaremos promoción. Una promoción tiene una duración temporal determinada, un periodo de tiempo en el que los alumnos participantes deben finalizar su itinerario. Tras la finalización analizaremos cómo evoluciona el grafo de

itinerarios de aprendizaje según las diferentes proporciones de perfiles de alumnos. Toda la experimentación se realizará simulando a través del software desarrollado a tal efecto. La configuración inicial de la audiencia (que porcentaje de alumnos de cada perfil participan) es irrelevante, interesando la evolución en función de las configuraciones siguientes y de su comportamiento.

Utilizaremos al algoritmo ASALI para simular el comportamiento de los alumnos mediante las hormigas ACO, utilizando diferentes proporciones de los perfiles definidos (High, Medium y Low), siendo los experimentos planteados son:

a) Simulaciones de la edición de un curso con las proporciones de alumnos, según sus perfiles, indicadas en la Tabla 21.

b) Simulación de la edición de un curso con alumnos sin perfil determinados - calificaciones aleatorias con distribución normal N(5,2).

Tabla 21: Casos de prueba de simulación de la edición de un curso para un número de participantes fijo.

Experimento	% High	% Medium	% Low
E1	10	30	60
E2	10	60	30
E3	30	10	60
E4	30	60	10
E5	60	10	30
E6	60	30	10
E7	50	0	50
E8	50	50	0
E9	0	50	50
E10	100	0	0
E11	0	100	0
E12	0	0	100

Las simulaciones del primer caso estudian la variabilidad de los itinerarios proporcionados por el algoritmo en función del perfil del alumno a lo largo de una edición del curso para 100 alumnos. Los resultados del experimento del segundo caso servirán para estudiar la estabilidad del sistema independientemente de los perfiles de alumnos que compongan la audiencia del curso. Se realizarán veinticinco simulaciones de cada caso, para analizar la variación promedio en la evolución del algoritmo, que deberá potenciar la elección del camino con mejor calificación tras la primera promoción. Se analizará en qué porcentaje varía esta elección en función de los perfiles de alumnos.

Estado Inicial

El estado inicial es el resultado de la finalización de la formación en la primera promoción de 100 alumnos, cuyas calificaciones y el itinerario seguido se muestran en las dos tablas siguientes (Tabla 22 y Tabla 23).

Tabla 22: Calificaciones e itinerario seguidos por cada alumno en la simulación de la primera promoción.

Orden	Tipo	Media	Camino	Orden	Tipo	Media	Camino
1	3	4,4379	J	26	1	5,7588	I
2	1	6,2916	J	27	2	5,7595	I
3	1	6,9140	J	28	2	5,7892	I
4	2	6,2536	I	29	3	6,0008	J
5	3	6,3068	J	30	2	6,0183	J
6	2	5,9133	I	31	3	5,6769	I
7	3	5,5906	I	32	2	6,3596	A
8	1	6,3053	J	33	3	5,6207	I
9	3	7,2688	H	34	2	5,6314	I
10	2	5,7891	I	35	2	6,7059	G
11	3	5,5840	I	36	2	6,2998	A
12	1	6,9274	D	37	2	5,6482	I
13	3	6,5693	B	38	3	5,5984	I
14	1	5,6785	I	39	2	5,9110	J
15	1	6,1296	J	40	2	5,5910	I
16	3	5,6188	I	41	2	5,6328	I
17	2	6,2894	C	42	2	5,9459	J
18	1	5,6544	I	43	2	5,9762	J
19	3	5,5708	I	44	2	6,0095	J
20	3	5,8269	J	45	3	5,6147	I
21	1	5,5177	I	46	3	6,4748	D
22	2	5,5558	I	47	3	5,5568	I
23	2	6,2621	A	48	3	5,8572	J
24	2	5,8809	J	49	2	6,7488	H
25	1	5,6728	I	50	3	5,5037	I

Partiendo de los datos de las Tabla 22 (continúa en la Tabla 23), el software desarrollado nos proporciona además las calificaciones medias de cada nodo, así como la cantidad de feromonas τ y Φ existentes en cada arco a la finalización de la promoción (Tabla 24). En la simulación se han utilizado los valores α=1.2, β=1.5, ρ=0.65, ω_1=1.0, ω_2=0.02 y ω_3=1.5.

Tabla 23: Itinerario seguido por cada alumno y calificación media obtenida en la simulación de la primera promoción (continuación).

Orden	Tipo	Media	Camino	Orden	Tipo	Media	Camino
51	3	5,8010	J	76	3	5,2565	I
52	3	5,4311	I	77	3	5,2415	I
53	2	5,4560	I	78	3	5,2216	I
54	2	5,7775	J	79	3	5,7996	A
55	2	6,5670	E	80	3	5,1963	I
56	3	5,7428	J	81	3	5,8600	D
57	3	5,7191	J	82	3	5,1754	I
58	3	5,9622	C	83	3	6,2076	E
59	3	5,3737	I	84	3	5,1581	I
60	3	5,3649	I	85	3	5,1438	I
61	2	5,6865	J	86	3	5,1292	I
62	2	5,6905	J	87	3	5,1166	I
63	3	5,3711	I	88	3	5,1033	I
64	3	6,0384	A	89	3	5,0848	I
65	2	5,3482	I	90	3	5,0721	I
66	2	5,3704	I	91	3	5,7931	D
67	2	5,3860	I	92	3	5,0608	I
68	3	5,8779	C	93	3	5,0549	I
69	3	5,8201	C	94	3	6,0490	E
70	3	6,1742	B	95	3	5,0197	I
71	3	5,3155	I	96	3	5,0099	I
72	3	5,6037	J	97	3	5,3034	J
73	3	5,2889	I	98	3	4,9937	I
74	3	5,2763	I	99	3	4,9803	I
75	3	5,5510	J	100	3	4,9676	I

Los datos de las tablas anteriores se representan en la gráfica de barras de la Figura 57, en la que se aprecia como los itinerarios I y J son los que más ampliamente han sido elegidos por el algoritmo, con un 55% y un 24% de las veces respectivamente. Esta elección está claramente condicionada por los pesos pedagógicos y la longitud del itinerario, al ser inicialmente nula la cantidad de feromonas existente en cada arco[24]. No en vano son los dos itinerarios más cortos en cuanto a número de nodos se refiere, lo que influye positivamente a la hora de calcular la calificación media final del itinerario, objetivo que se desea maximizar.

[24] El proceso de inicialización establece una cantidad de feromonas inicial igual a la inversa del número de nodos del grafo, para evitar así indeterminaciones en el cálculo de la probabilidad de decisión de cada nodo por el algoritmo.

Tabla 24: Estado inicial para el estudio de adaptabilidad[25]

L_{ij}	N_i	N_j	$\overline{\varepsilon_i}$	$\overline{\varepsilon_j}$	τ_{ij}	φ_{ij}	f_{ij}
Link0	N1	N2	4,8103	4,9810	0,4130	4,5943	1,2753
Link1	N2	N3	4,9810	5,2617	0,4130	5,0689	1,3024
Link2	N3	N5	5,2617	4,8605	0,0250	3,7265	0,5463
Link3	N3	N6	5,2617	5,4129	0,0250	0,1927	0,1777
Link4	N3	N7	5,2617	5,4617	0,4130	-0,1446	0,0878
Link6	N7	N6	5,4617	5,4129	0,4130	-0,1884	0,5020
Link7	N6	N9	5,4129	4,7860	0,0250	0,1959	0,3112
Link8	N9	N8	4,7860	4,9243	0,0250	0,1336	0,7576
Link9	N6	N8	5,4129	4,9243	0,4130	0,0050	0,3880
Link10	N8	N4	4,9243	4,9776	0,4130	0,5022	1,0403
Link11	N4	N10	4,9776	4,9935	0,4130	4,4622	1,2678
Link12	N10	N11	4,9935	5,0008	0,4130	4,5791	1,2721
Link13	N11	N12	5,0008	4,8361	0,0250	2,2370	0,6266
Link14	N11	N13	5,0008	5,3484	0,4130	0,4660	0,2894
Link15	N13	N12	5,3484	4,8361	0,4130	1,8457	1,1182
Link16	N12	Fin	4,8361	4,9865	0,4130	4,4487	1,2670
Link17	N5	N4	4,8605	4,9776	0,0250	3,8404	1,2195

A pesar de que en esta primera promoción los itinerarios I y J han sido los que más veces se han recorrido, la máxima calificación se ha obtenido siguiendo el itinerario H, por lo que el refuerzo adicional de feromonas se ha destinado a los nodos de este itinerario, como se puede apreciar en la Figura 58. La probabilidad de decisión de cada uno de los arcos en los tres nodos de decisión del grafo (nodos N3, N6 y N11) se indica en la Tabla 25.

[25] Lij: Arco; Ni: nodo origen; Nj: nodo destino; $\overline{\varepsilon_i}$: Calificación media en el nodo origen; $\overline{\varepsilon_j}$:Calificación media en el nodo destino; τ_{ij}: feromonas en el arco L_{ij}; φ_{ij}: feromonas online en el arco L_{ij}; f_{ij}: función de ajuste del arco L_{ij}.

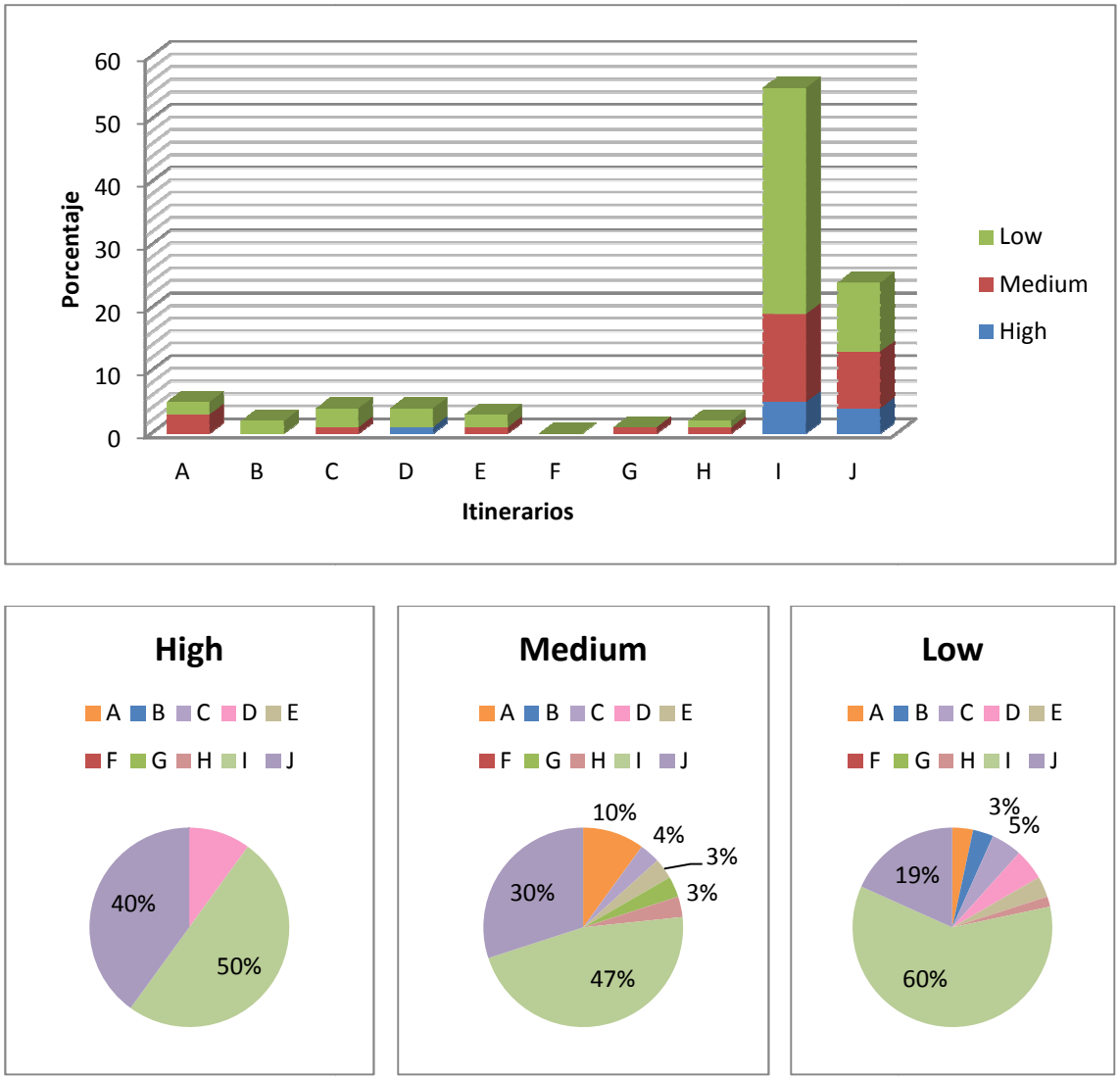


Figura 57: Elecciones de itinerarios tras la primera iteración (estado inicial).

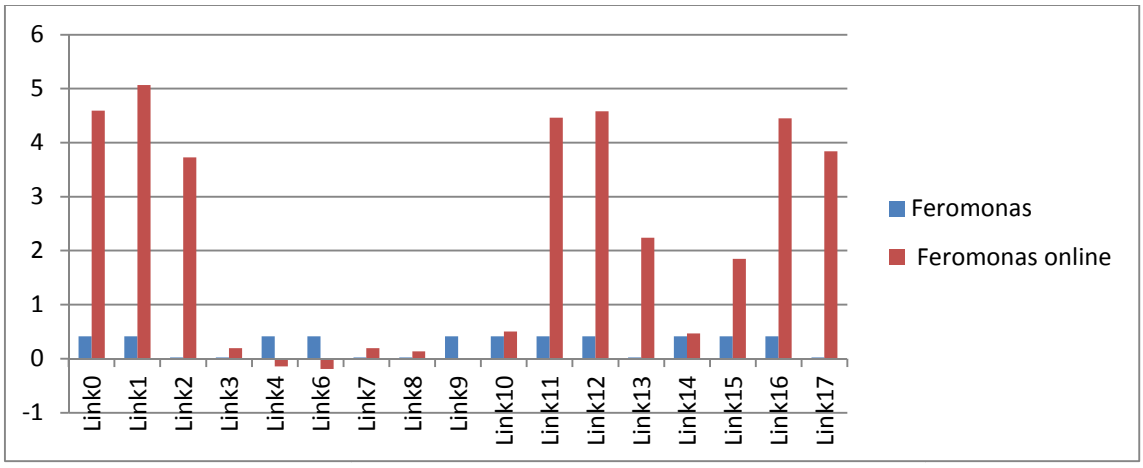


Figura 58: Distribución inicial de feromonas en los arcos del grafo de itinerarios de aprendizaje.

Tabla 25: Probabilidad de los arcos en los nodos de decisión tras finalizar la primera promoción

Nodo de decisión de N3	
De N3 a N5	33%
De N3 a N6	6%
De N3 a N7	61%
Nodo de decisión de N11	
De N11 a N12	10%
De N11 a N13	90%
Nodo de decisión de N6	
De N6 a N8	21%
De N6 a N9	79%

Análisis de Resultados

Un primer análisis de los resultados nos sugiere que el porcentaje medio alumnos que eligen el itinerario H en la segunda promoción es menor cuanto mayor es la proporción de alumnos de perfil Low (Tabla 26). Lo primero que pensamos es que esto pudiera ser debido al comportamiento de los alumnos de perfiles Medium y High que en la mayoría de los experimentos, debido a sus buenas calificaciones en promedio a través del itinerario H, podrían contribuir a potenciar positivamente este itinerario. Sin embargo, un análisis detallado de las pruebas nos lleva a afirmar que no puede considerarse este comportamiento como infalible, puesto que en algunas de las ejecuciones se obtuvo un comportamiento diferente, aunque sí que se puede constatar que según los perfiles de alumnos configurados, fue el comportamiento que más frecuentemente se obtuvo.

Sí que es destacable, en cambio, el aumento de recorridos que experimenta el itinerario H, que lleva consigo un descenso en los recorridos del itinerario I. La explicación a este hecho se debe a los refuerzos recibidos en los nodos del itinerario H al final de la primera promoción, entre ellos el arco que va del nodo N_3 al N_7 en detrimento del que lo une con el N_5, perteneciente tanto al itinerario I como al J. En cambio el descenso de recorridos del nodo J no es tan notable como el del itinerario I, puesto que aunque el descenso de la probabilidad de selección del nodo $N_3 \rightarrow N_5$ afecta a ambos itinerarios, una vez seleccionado, al llegar al siguiente nodo de decisión en el nodo N_{11}, la elección entre el nodo N_{12} o el N_{13} como siguiente nodo, favorece al segundo gracias al refuerzo recibido por el arco $N_{11} \rightarrow N_{13}$, común con el itinerario H. De ahí que el descenso del itinerario J no sea tan notable como el del I.

Tabla 26: Comparativa entre los experimentos E1 y E2

Experimento E1. H=10%; M=30%; L=60%				
	High	Medium	Low	Total
A	0%	2%	1%	1%
B	0%	0%	0%	0%
C	8%	2%	1%	2%
D	8%	2%	3%	3%
E	0%	0%	0%	0%
F	0%	0%	0%	0%
G	14%	7%	6%	7%
H	22%	46%	53%	48%
I	24%	17%	16%	17%
J	24%	23%	20%	21%

Experimento E2. H=10%; M=60%; L=30%				
	High	Medium	Low	Total
A	0%	0%	0%	0%
B	0%	0%	0%	0%
C	8%	1%	0%	1%
D	20%	6%	7%	8%
E	0%	0%	0%	0%
F	0%	0%	0%	0%
G	6%	6%	9%	7%
H	42%	63%	63%	61%
I	0%	2%	2%	2%
J	24%	21%	19%	21%

Las conclusiones a raíz de los experimentos E3 y E4 (Tabla 27), coinciden con las extraídas de los dos experimentos anteriores: el cambio en la proporción no justifica por sí mismo un cambio en la selección de itinerarios, sino que es la sucesión de calificaciones la que decide los refuerzos online (feromonas Φ) a depositar en cada arco y por tanto la variación de probabilidad de selección de cada uno. Sigue siendo destacable el descenso del porcentaje de alumnos encaminados a través del itinerario I, mientras que el descenso a través del itinerario J es menor.

Tabla 27: Comparativas entre los experimentos E3 y E4.

Experimento E3. H=30%; M=10%; L=60%				
	High	Medium	Low	Total
A	0%	0%	0%	0%
B	0%	0%	0%	0%
C	1%	6%	0%	1%
D	8%	8%	4%	6%
E	0%	0%	0%	0%
F	0%	2%	0%	0%
G	17%	8%	7%	10%
H	61%	54%	74%	68%
I	5%	4%	1%	3%
J	9%	18%	13%	13%

Experimento E4. H=30%; M=60%; L=10%				
	High	Medium	Low	Total
A	0%	0%	0%	0%
B	0%	0%	0%	0%
C	2%	0%	2%	1%
D	3%	3%	12%	4%
E	0%	0%	0%	0%
F	1%	1%	0%	1%
G	7%	5%	6%	6%
H	47%	64%	48%	58%
I	4%	1%	8%	2%
J	37%	26%	24%	29%

Las mismas conclusiones extraídas de los cuatro experimentos anteriores pueden extraerse de los experimentos E5 y E6 (Tabla 28).

Tabla 28: Comparativa entre los experimentos E5 y E6.

Experimento E5. H=60%; M=10%; L=30%				
	High	Medium	Low	Total
A	0%	0%	0%	0%
B	0%	0%	0%	0%
C	1%	0%	0%	0%
D	6%	4%	3%	5%
E	0%	0%	0%	0%
F	1%	0%	1%	1%
G	7%	12%	3%	6%
H	59%	62%	79%	65%
I	2%	0%	0%	1%
J	25%	22%	15%	21%

Experimento E6. H=60%; M=30%; L=10%				
	High	Medium	Low	Total
A	0%	0%	0%	0%
B	0%	0%	0%	0%
C	0%	0%	0%	0%
D	6%	7%	14%	7%
E	0%	0%	0%	0%
F	1%	1%	2%	1%
G	7%	6%	12%	7%
H	74%	64%	58%	69%
I	0%	4%	0%	1%
J	12%	19%	14%	14%

Los experimentos E7, E8 y E9, para los que sólo se tienen cuenta dos perfiles de alumnos, corroboran que el comportamiento del algoritmo cuando la colonia está dividida al 50% en individuos de dos perfiles diferentes, es el mismo descrito en los experimentos anteriores. La Figura 59 presenta el porcentaje de individuos que recibió la formación a través de los itinerarios H, I o J, que son los que han experimentado cambios considerables en los experimentos anteriores.

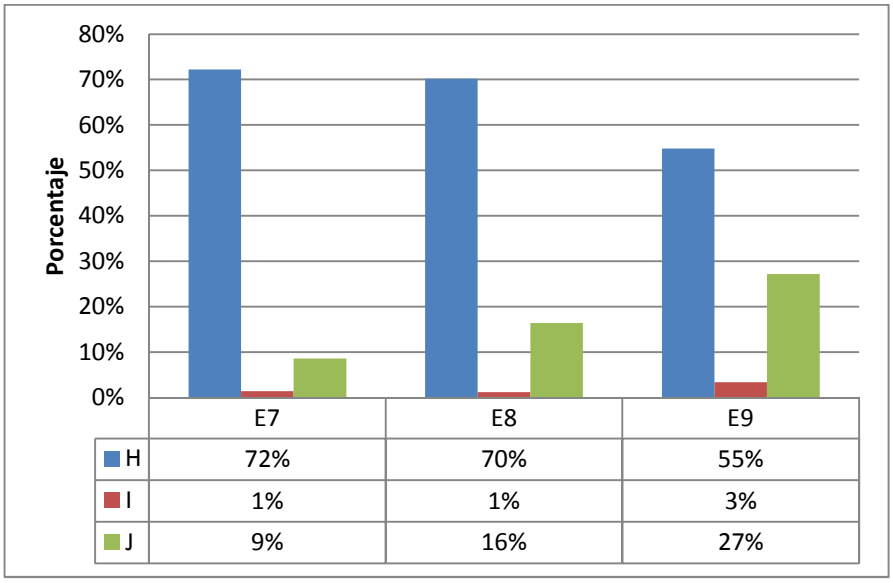


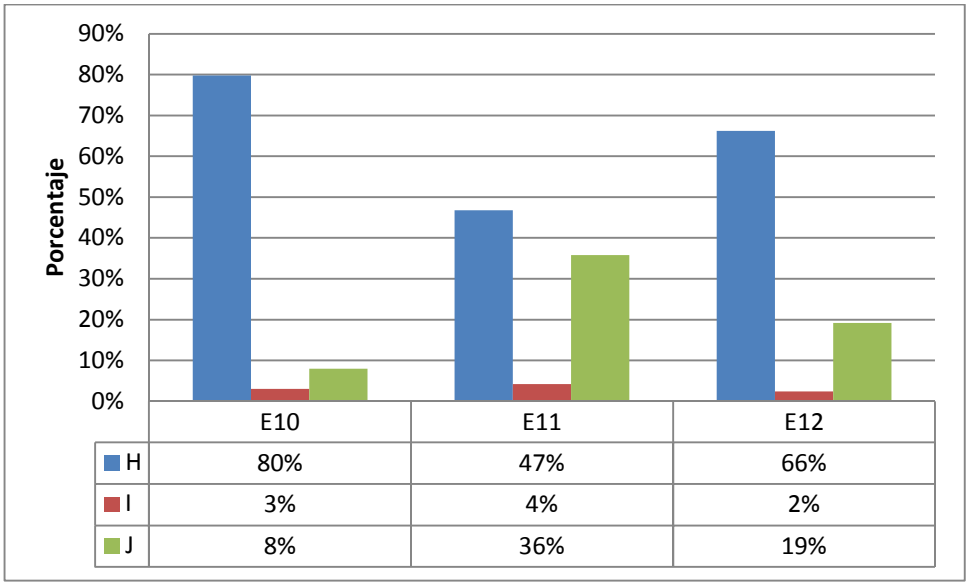
Figura 59: Comparativa en los experimentos E7, E8 y E9.

Los dos primeros experimentos arrojan resultados prácticamente parejos, en cambio en el último, la ausencia de alumnos de perfil más alto, resulta en una menor

propensión al cambio, si bien como ya se ha indicado anteriormente, este hecho depende de las calificaciones obtenidas y no propiamente del perfil.

La transición hacia el itinerario H es demasiado alta, teniendo en cuenta que, aunque en la primera promoción la calificación media más alta se obtuvo en este itinerario con un 7.26 sobre 10, tan sólo dos individuos siguieron este itinerario.

Por último, los experimentos con un único perfil también corroboran el comportamiento observado en los experimentos anteriores: notable aumento de probabilidad de elección del itinerario H y disminución drástica de la probabilidad de selección del itinerario I, perjudicado por el refuerzo del arco N11→N13 que el itinerario J tiene en común con el itinerario H.



	E10	E11	E12
H	80%	47%	66%
I	3%	4%	2%
J	8%	36%	19%

Figura 60: Comparativa en los experimentos E10, E11 y E12.

Como conclusión del primer grupo de experimentos podemos corroborar que el algoritmo se comporta como esperamos, reforzando el itinerario a través del cual se obtuvo una mejor calificación. Como efecto colateral se refuerzan también, aunque en menor grado, otros itinerarios que compartan arcos con el de mayor éxito en algún nodo de decisión (nodos con más de un arco de salida).

Si simulamos una población de alumnos sin un perfil determinado, cuyas calificaciones pueden oscilar entre el 1 y el 9 sobre 10, el comportamiento del algoritmo, como se aprecia en la gráfica, es prácticamente el mismo.

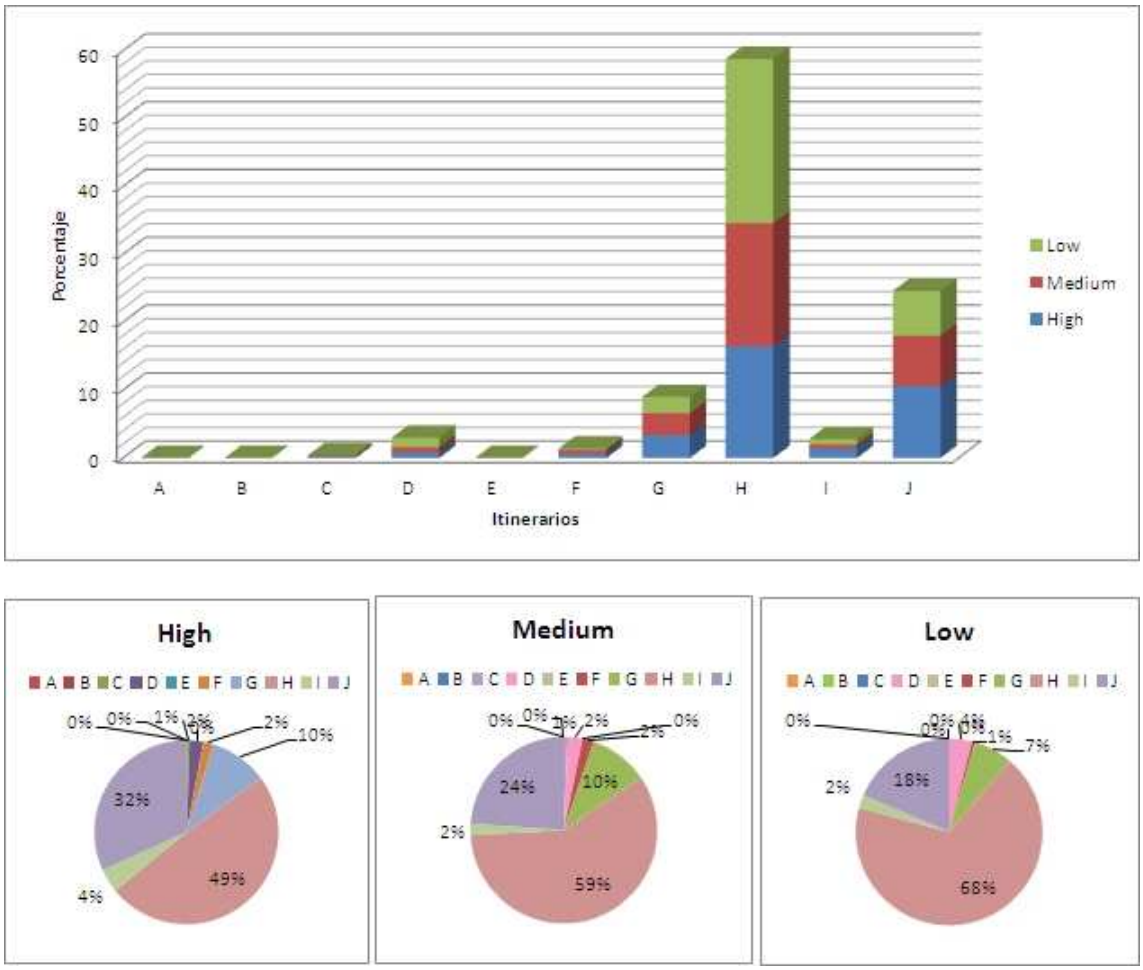


Figura 61: Resultados tras cinco ejecuciones con alumnos de perfil escogido aleatoriamente

Repitiendo el experimento durante 100 iteraciones, a pesar de ser aleatoria la proporción de alumnos, el itinerario H siempre fue el escogido, como mínimo, por el 39% de los participantes, y como máximo por el 81%, siendo en el 80% de las iteraciones el camino más utilizado.

Para analizar la distribución del reparto de alumnos por cada camino, utilizaremos el Coeficiente de Gini [123], que es una medida de la desigualdad ideada por el estadístico y economista italiano Corrado Gini. Se utiliza principalmente para medir la desigualdad en los ingresos, pero puede utilizarse para medir cualquier forma de distribución desigual.

El coeficiente de Gini es un número entre 0 y 1, en donde 0 se corresponde con la perfecta igualdad (el reparto se hace a partes iguales) y donde el valor 1 se

corresponde con la perfecta desigualdad (un elemento persona tiene todos los bienes y los demás ninguno). Se define como el cociente entre el área de Lorenz y el área de máxima desigualdad posible con *N* observaciones. Cuando el tamaño *N* es grande, el área de máxima desigualdad tiende a convertirse en el área de todo el triángulo bajo la línea de equidistribución (Figura 62).

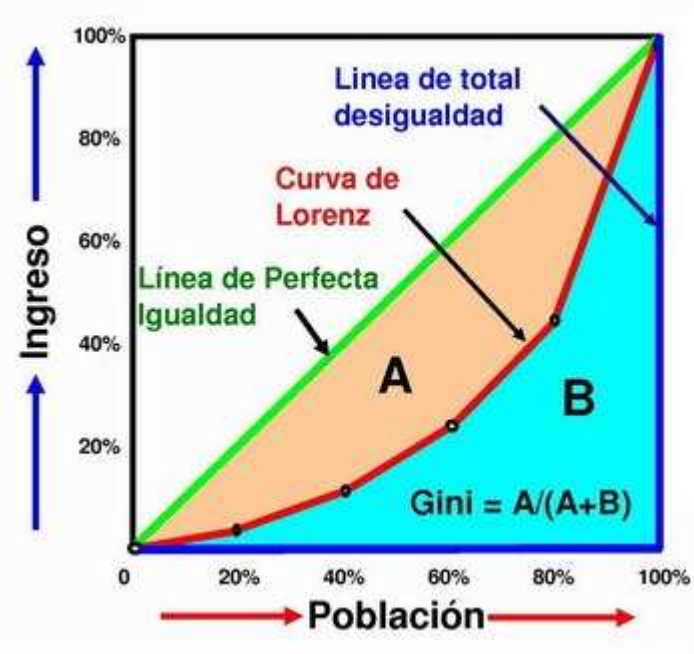


Figura 62: Relación del Coeficiente de Gini y la Curva de Lorenz

El coeficiente de Gini se calcula a menudo con la Fórmula de Brown [125], que es más práctica:

$$G = \left| 1 - \sum_{k=1}^{n-1} (X_{k+1} - X_k)(Y_{k+1} - Y_k) \right|$$

El índice de Gini no es más que el coeficiente de Gini expresado en porcentaje, y es igual al coeficiente de Gini multiplicado por 100. Las propiedades del índice de Gini son comparables con las del cuadrado del coeficiente de variación [124].

Para poder usar el coeficiente de Gini, es necesario que ningún elemento disponga de unos ingresos negativos.

La Tabla 29 muestra los índices de Gini de cada experimento.

Tabla 29: Índices de Gini en los experimentos del estudio de adaptabilidad

Experimento	H	M	L	Índice de Gini
E1	10	30	60	41,68%
E2	10	60	30	42,22%
E3	30	10	60	46,95%
E4	30	60	10	49,47%
E5	60	10	30	48,40%
E6	60	30	10	50,31%
E7	50	0	50	46,70%
E8	50	50	0	48,33%
E9	0	50	50	46,39%
E10	100	0	0	54,12%
E11	0	100	0	52,33%
E12	0	0	100	48,58%

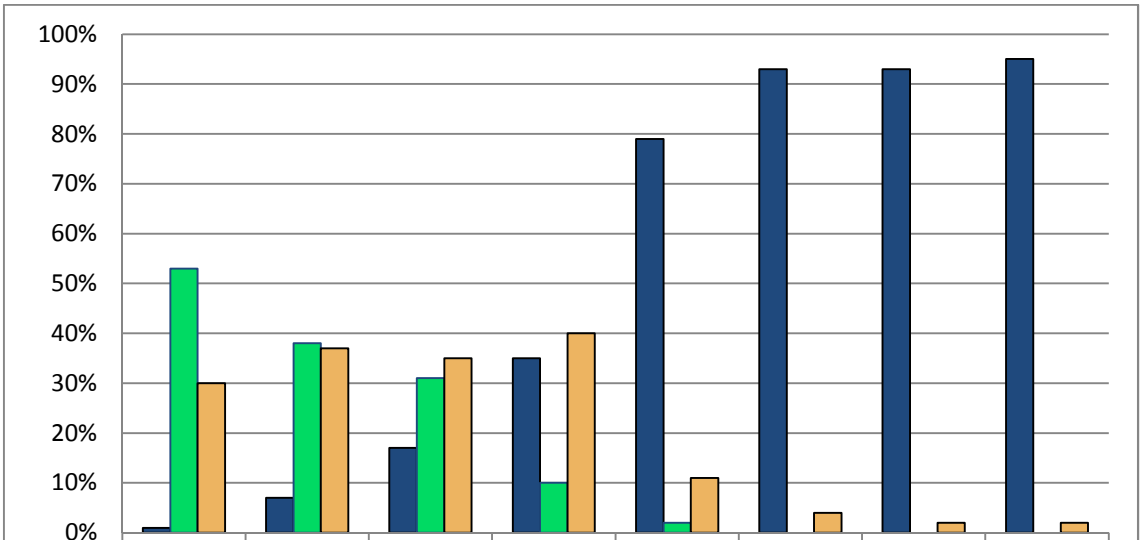
La Tabla 29 nos permite concluir que la proporción de alumnos influye en el reparto de itinerarios, siendo los alumnos con mejor rendimiento (perfil High) los que en mayor medida influyen en el itinerario asignado al resto, debido a que las feromonas depositadas al depender de la calificación obtenida, ofrecen un mayor refuerzo a los caminos seguidos por estos. De la misma forma los alumnos de perfil Medium tienen una influencia mayor que los de perfil Low. Esto se traduce en una mayor concentración de las elecciones unos pocos caminos posibles del grafo cuanto mayor es la proporción de alumnos del perfil High y Medium respecto a los alumnos de perfil Low, más tendentes a obtener calificaciones inferiores al mínimo deseado.

Cuando existen alumnos de los tres perfiles, la concentración es menor cuanto mayor es la proporción de alumnos del perfil más bajo, lo que significa que el algoritmo ASALI adopta una comportamiento ligeramente más explorador que explotador de soluciones.

El Índice de Gini para el experimento con alumnos cuyos perfiles son seleccionados aleatoriamente fue del 45,17%. El experimento simuló la participación de 500 alumnos, en cinco promociones, partiendo todas ellas del mismo estado inicial y resultando la distribución final de perfiles como sigue: 167 alumnos de perfil High (33,4%), 155 de perfil Medium (31%) y 178 alumnos de perfil Low (35,6%). La distribución de elecciones se corresponde con la gráfica de la Figura 61, en la que se

observa cómo la mayor parte de alumnos opta por los itinerarios H, J y G, por ese orden de preferencia.

El comportamiento del algoritmo puede variarse a través de los parámetros de calibración, como ya se ha comentado anteriormente, favoreciendo la explotación o exploración de soluciones mediante el aumento o disminución, respectivamente, del valor asignado al parámetro β, así como aumentando o disminuyendo el efecto memoria respecto a promociones anteriores con el aumento o disminución del valor asignado al parámetro α. El gráfica de la Figura 63 muestra como un valor de α igual a 0.25 prácticamente mantiene igual respecto a la promoción anterior las proporciones de alumnos que eligen uno u otro itinerario, mientras que conforme se aumenta el valor, el efecto del refuerzo de feromonas recibido en los arcos del itinerario H se incrementa, disminuyendo el porcentaje de alumnos que escogen el itinerario I y aumentando el porcentaje de los que escogen el H.



	0,25	0,5	0,75	1	1,25	1,5	1,75	2
%H	1%	7%	17%	35%	79%	93%	93%	95%
%I	53%	38%	31%	10%	2%	0%	0%	0%
%J	30%	37%	35%	40%	11%	4%	2%	2%

Figura 63: Porcentaje de alumnos que eligen H, I y J en la segunda promoción en función del valor de α

Un excesivo refuerzo puede llevar al sistema al bloqueo en una solución particular, tal y como ocurre en el mundo real de las hormigas, cuando tras un periodo de exploración todas acaban transitando en fila india por el camino óptimo entre el nido y la fuente de alimento. Si el porcentaje de alumnos que sigue un itinerario es muy alto,

la probabilidad de que el alumno que mejor calificación obtenga lo haga siguiendo un itinerario diferente serán mínimas, lo que evitará el refuerzo en otro itinerario diferente y será prácticamente imposible que el sistema pueda explorar itinerarios alternativos, incluso aunque la secuencia de malas calificaciones sea increíblemente alta. Para evitarlo pueden utilizarse las acciones globales, contempladas en la meta-heurística ACO, de forma que el refuerzo recibido en un itinerario se reduzca cuando el porcentaje de alumnos que lo transitó es igualmente alto:

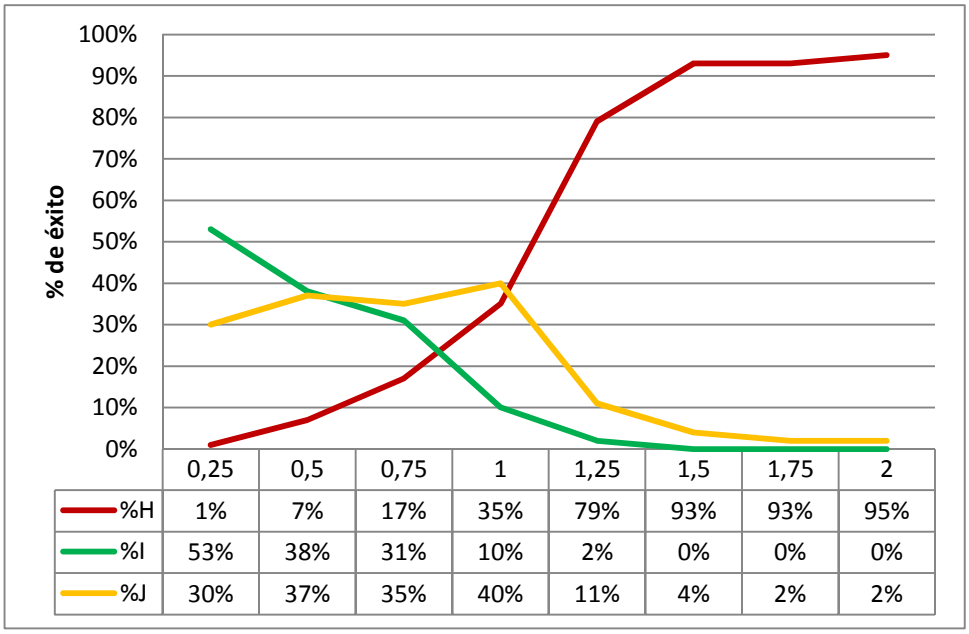
$$\tau_{ij}(t) = \left(1 - Q_{ij}\right)\tau_{ij}(t)\ \forall l_{ij}\ \in\ R^k$$

siendo $Q_{ij} \in [0,1]$ el porcentaje de alumnos que siguieron el arco l_{ij} del itinerario R^k. Esta acción global contribuye a que el refuerzo de feromonas en arcos que han sido masivamente transitados, sea mínimo o nulo, minimizando así el bloqueo en ellos en la siguiente promoción, puesto que ya sólo recibirían el refuerzo procedente de las feromonas ϕ.

Conclusiones

Tras el período experimental a través de simulación se observa como a partir de un estado inicial dado, es posible que un itinerario seguido por un 2% de los alumnos evoluciones en la siguiente promoción a ser el itinerario asignado a un porcentaje comprendido entre el 40% y el 70% de los alumnos. El comportamiento del algoritmo se puede afinar, de forma que la variación no sea tan brusca, utilizando los parámetros α, β, ω_1, ω_2 y ω_3 según el comportamiento deseado por el equipo pedagógico.

Partiendo de los valores por defecto α=1, β=1.5, ω_1=1.0, ω_2=0.02 y ω_3=1.5, con una tasa de evaporación del 65% (ρ=0.65) el comportamiento del algoritmo puede aumentar o disminuir la confianza en el mejor itinerario de la promoción anterior según el valor del parámetro *α*. Puede disminuirse la influencia excesiva del camino que mejor calificación obtuvo en la promoción anterior disminuyendo la variable *α*, y aumentarla, incrementando su valor. Al recibir como refuerzo en τ_{ij} un valor proporcional al de la función de ajuste, f_{ij}, cuando el arco l_{ij} pertenece al camino de mayor éxito, y al estar fuertemente influenciada esta función por el efecto de las feromonas depositadas online, un valor de α menor reducirá el efecto de dichas feromonas online. Con la experimentación con el grafo se han obtenidos los resultados indicados en la Figura 64:



	0,25	0,5	0,75	1	1,25	1,5	1,75	2
%H	1%	7%	17%	35%	79%	93%	93%	95%
%I	53%	38%	31%	10%	2%	0%	0%	0%
%J	30%	37%	35%	40%	11%	4%	2%	2%

Figura 64: Porcentaje de éxito de los itinerarios H, I y J respecto a la variación del parámetro α en la segunda promoción

El parámetro α, como ya se indicó durante el proceso de calibración, ha tomado valores comprendidos en el intervalo $0 < \alpha \leq 2$.

Básicamente, el comportamiento del algoritmo puede ser controlado, una vez calibrado para una población de alumnos establecida (en nuestro estudio, para una población de 100 alumnos) mediante el parámetro α. Esta característica facilita la integración de este algoritmo en LMS en los casos en los que la población de alumnos se decide fija (o se fija un máximo número de alumnos por promoción), permitiendo configurar el resto de parámetros, una vez calibrado para la población en concreto de alumnos, como propiedades fijas del entorno.

Capítulo 5. Incorporación de itinerarios adaptativos a un LMS

Uno de los principales problemas en la industria del e-learning es la interoperatividad entre los distintos LMS, incluso en los de código abierto. Esto es debido principalmente a que, a pesar de que la mayoría de ellos implementan una o más especificaciones de las ampliamente aceptadas, no todos implementan las mismas para la misma funcionalidad e incluso algunos optan por formatos propietarios rompiendo intencionadamente la compatibilidad. En cuanto a la duda entre optar por aplicaciones que implementan estándares abiertos pero de código propietario o aplicaciones de código libre, J. Dalziel apunta en [88] que son las aplicaciones comerciales las que se ven más obligadas a cumplir estándares abiertos para asegurar el éxito de éstas, puesto que en las aplicaciones de código abierto, siempre cabe la posibilidad de a partir del código, realizar las modificaciones oportunas para cumplir los estándares necesarios (a menudo dando lugar a nuevos productos).

Para ilustrar esta falta de entendimiento a pesar las especificaciones existentes para garantizar la interoperatividad podemos citar el caso de la especificación IMS DRI (Digital Repositories Interoperatibility) creada precisamente para garantizar la interoperatividad entre almacenes de recursos digitales: El proyecto OKI, basado en APIs de código abierto, y en estrecha colaboración con el consorcio IMS durante el mismo periodo en que la especificación DRI fue desarrollada, eligió su propio formato de almacenes de recursos digitales. Como consecuencia, el Centro de Investigación de e-learning de la Universidad de Cambridge (CARET) intentó implementar las dos aproximaciones, IMS DRI y el formato de OKI, y la conclusión final fue que "era más difícil de lo que esperábamos" [88].

5.1 Arquitecturas de los sistemas de enseñanza a distancia

El estándar 1484.1 del IEEE, LTSA (Learning Technology Systems Architecture), define una arquitectura neutral con respecto a los contenidos, metodologías pedagógicas y la tecnología de la plataforma (Figura 65). Se trata de una arquitectura a muy alto nivel para sistemas de aprendizaje que usen la tecnología como medio principal de transmisión y soporte para los procesos educativos. LTSA

proporciona un marco en el que entender los sistemas de e-learning actuales y futuros, pero no especifica detalles concretos para su implementación (como lenguajes de programación, herramientas de autor o sistemas operativos) ni para su gestión (como ciclo de vida de los componentes u objetos educativos, aseguramiento de la calidad, control de acceso o administración de usuarios).

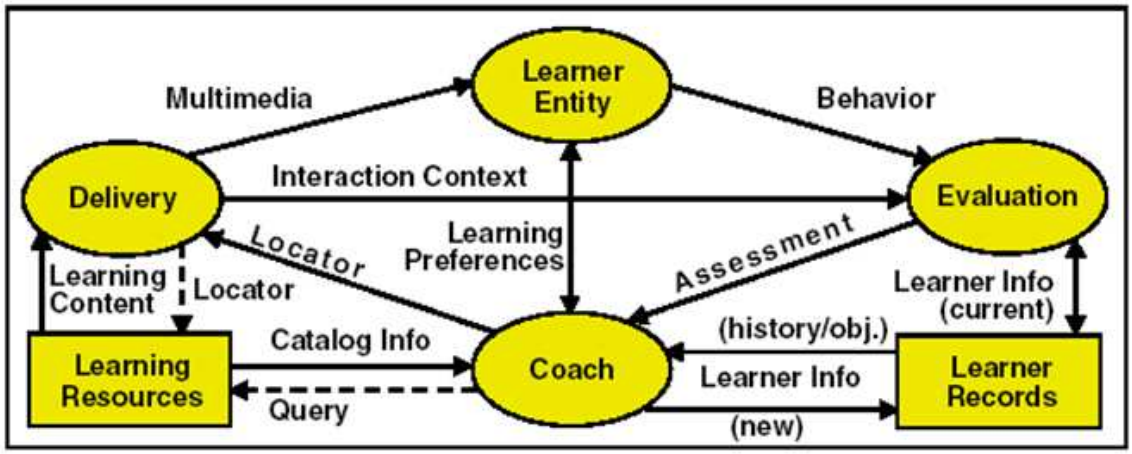


Figura 65: Modelo funcional de la especificación LTSA de IEEE

La producción de numerosas especificaciones es una de las características de los procesos de estandarización en el área del e-learning, las cuales han sido publicadas por diversos organismos, centrándose sus esfuerzos en la definición de metadatos, los mecanismos de agregación de contenidos, la interoperatividad entre plataformas, la definición de lenguajes de modelado de los procesos de enseñanza, la definición de reproductores de cursos interactivos, etc. El resultado de todo este esfuerzo a lo largo de los años ha sido la aparición de muchas plataformas diferentes, generalmente incompatibles entre sí, pero que gracias al agrupamiento de un conjunto de especificaciones en el estándar de facto SCORM, consiguen poder reutilizar, no a veces sin esfuerzo, cursos desarrollados para diferentes plataformas. La gran mayoría de estas plataformas se basan en el modelo funcional LTSA o en alguna variación de éste, como por ejemplo el modelo propuesto por Liu et al. [89], para considerar funcionalidades adicionales como entornos específicos para el trabajo colaborativo, la creación de contenidos o la gestión de cursos (Figura 66). Aparecieron así las diferencias entre los entornos meramente gestores del aprendizaje (en inglés *Learning Management System* o LMS) de los gestores de contenidos educativos (en inglés *Learning Content Management System* o LCMS).

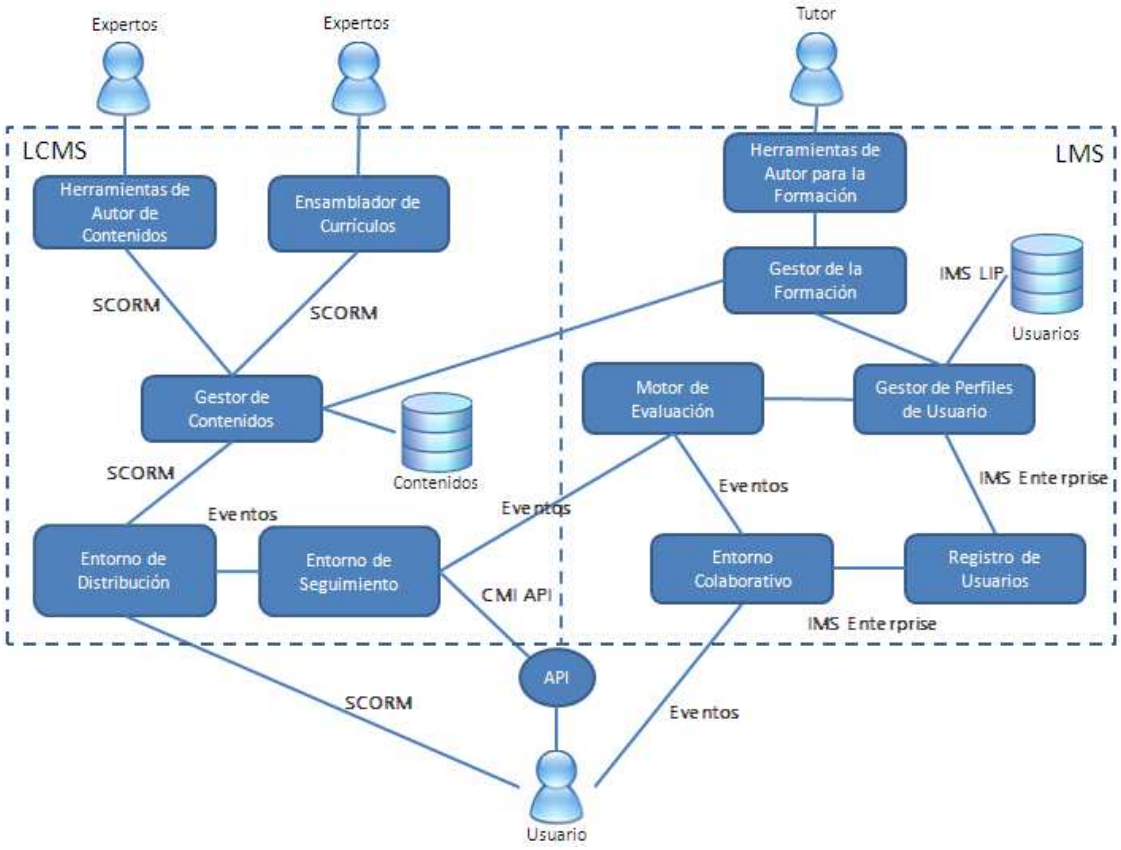


Figura 66: Modelo funcional del Sistema e-learning de Liu, El Saddik y Georganas

Aunque no todas las implementaciones de sistemas e-learning toman como referencia el estándar de IEEE, sí que coinciden la mayoría en la necesidad de garantizar la interoperatividad con otros sistemas independientemente del lenguaje de programación utilizado o de la arquitectura interna de cada uno.

Para la integración de nuestra propuesta en una plataforma de e-learning, hay que tener en cuenta que no existe el concepto de itinerario de aprendizaje. Las especificaciones actuales no contemplan esta posibilidad: el seguimiento y gestión del aprendizaje se circunscribe al curso SCORM, empaquetando en él todos los recursos educativos necesarios, incluidos test de autoevaluación. Se fuerza de esta forma a diseñar las posibles estrategias de adaptación a través del descriptor del paquete del curso [9], añadiendo metadatos en descriptores adicionales para mantener la compatibilidad con el estándar de facto de la industria. Esto reduce la variabilidad al mismo tiempo que aumenta el tamaño del curso. Otras especificaciones como IMS Learning Design (IMS LD), permiten describir los procesos de enseñanza-aprendizaje, definiendo etapas de aprendizaje on-line, aprendizaje presencial, ejercicios, trabajos en grupo, siguiendo un enfoque constructivista frente al predominantemente conductista de SCORM. Por su complejidad y falta de plataformas que lo soporten, IMS LD no ha

conseguido imponerse en la industria del e-learning, siendo SCORM, primero en su versión 1.2 y luego en la actual versión 2004, el estándar de facto de la industria, por lo que el principal interés de esta sección se centra en las acciones a realizar para dotar a un LMS conforme con SCORM, para poder soportar itinerarios de aprendizaje adaptativos según el enfoque presentado en este trabajo.

Antes de describir la estrategia a seguir para adaptar un LMS para trabajar con itinerarios de aprendizaje adaptativos, repasamos los conceptos básicos de la especificación SCORM, estándar de facto de la industria.

El estándar SCORM especifica el API de comunicación entre el SCO (los contenidos de un curso: páginas, objetos, contenido multimedia, etc.) pero no como implementa cada LMS el control del avance o cualquier otra tarea. En el esquema de comunicación de SCORM, el API que asegura la interoperabilidad de un curso SCORM en diferentes LMS, se incluye en el propio paquete distribuible de SCORM (Figura 67).

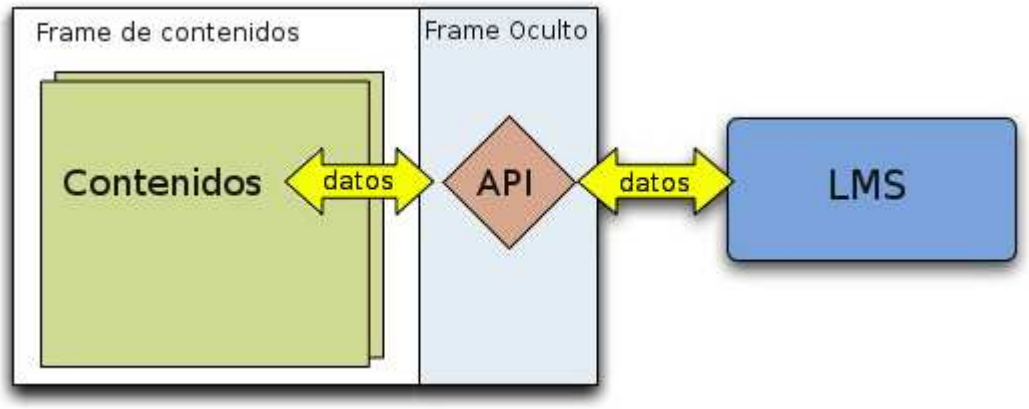


Figura 67: Esquema de comunicación con SCORM

La especificación SCORM define el API de comunicación y el lenguaje en que ésta debe ser implementada: el estándar ISO 16262, conocido como *ECMAScript* o más comúnmente como *Javascript*. También define como los contenidos deben buscar el API (Figura 68), e incluso proporciona algoritmos específicos para ello (Listado 38). En cambio la implementación de la comunicación entre el API y el LMS depende del propio LMS que distribuye el paquete SCORM. Es decir, se asegura la compatibilidad de un paquete SCORM (un curso) en diferentes LMS porque se estandariza la forma de localizar el API SCORM y como los contenidos pueden intercambiar datos con el LMS a través de esta API, pero es el distribuidor del paquete quién se encarga de inyectar su propia implementación del API a la hora de distribuir el paquete SCORM.

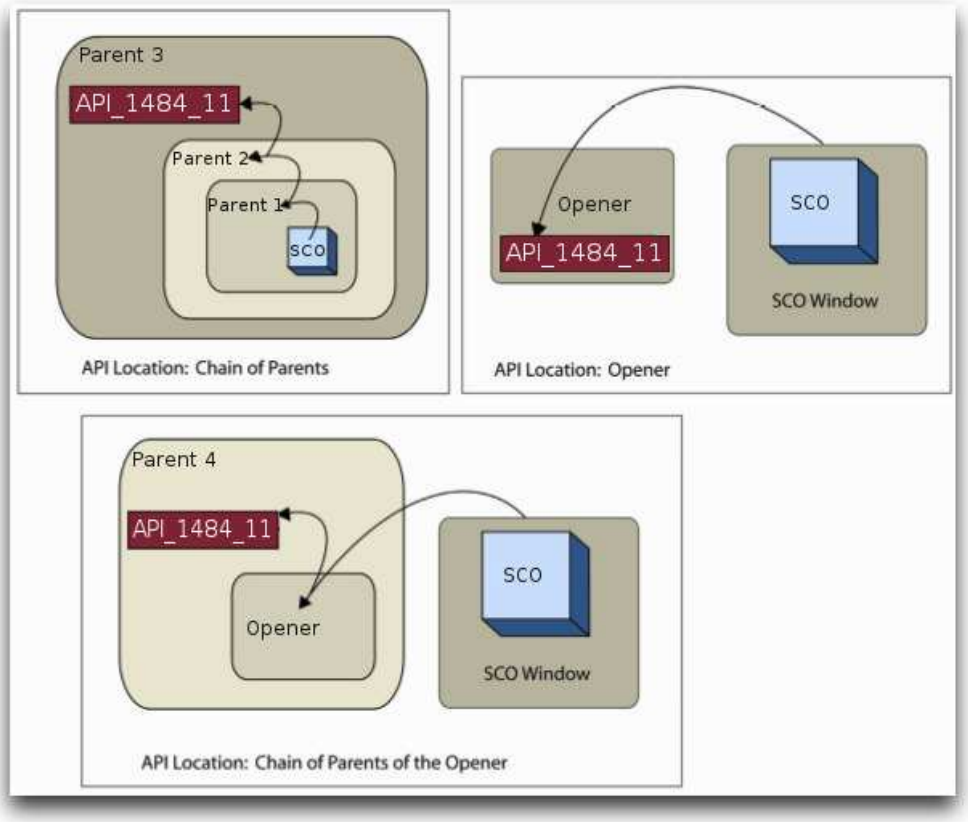


Figura 68: Mecanismos de localización del API de SCORM

El API de SCORM 2004, define ocho funciones básicas:

- **Initialize:** Inicializa la comunicación con una instancia de la API de SCORM. Devuelve un valor lógico que se representa con las cadenas "true" o "false" para indicar, respectivamente, la correcta o incorrecta inicialización del curso. Se define en *JavaScript* como: `Initialize(""): bool` .
- **Terminate:** Finaliza la comunicación con una instancia de la API de SCORM. Devuelve un valor lógico para indicar la correcta o incorrecta terminación del curso. Se define en *JavaScript* como `Terminate(""): bool`.
- **GetValue: S**olicita datos al entorno de ejecución. Recibe como parámetro el elemento del modelo de datos de CMI (*Computer Managed Instruction*) de AICC que se quiere consultar[26]. Se define en *JavaScript* como `GetValue( element: CMIElement ): string`.

[26] Como se describe en la memoria del periodo investigador [5], SCORM se apoya en un conjunto de especificaciones para garantizar la interoperabilidad.

- **SetValue:** Esta función permite enviar datos para su almacenamiento en el entorno de ejecución. Para ello toma como parámetros el elemento de datos del modelo CMI que se quiere actualizar, y el valor con el que se quiere actualizar. Se define en *JavaScript*: `SetValue(element: CMIElement, Value: string): string`.
- **GetLastError:** Recupera un código que representa el estado de error de la sesión de comunicación después de la última llamada a la API. Se define en *JavaScript* como: `GetLastError(): CMIErrorCode`.
- **GetErrorString:** Recupera el mensaje asociado a un estado error que se pasa como parámetro. Su definición en *JavaScript* es la siguiente: `GetErrorString(errorCode: CMIErrorCode): string`.
- **GetDiagnostic:** Permite al LMS devolver información detallada sobre el primer error que puede ser útil en el diagnóstico del problema. Por ejemplo, la información de diagnóstico para el error "406" podría ser "El valor 'cero' no está permitido para cmi.score.raw. El elemento cmi.score.raw debe contener un número válido representado sólo con dígitos". Su definición en *JavaScript* es: `GetDiagnostic(errocCode: CMIErrorCode): string`.
- **Commit:** Indica al LMS que una porción significante de datos ha sido salvada y que debería asegurarse que se persisten apropiadamente. SCORM no impone ningún requisito acerca de cómo el LMS debe implementar esta persistencia. La definición en el API *JavaScript* es: `Commit(""): bool`.

Muchos de los LMS actuales usan AJAX (*Asynchronous Javascript And XML*) en su implementación del API para comunicar los contenidos SCORM de forma asíncrona con el propio LMS. AJAX es una técnica de desarrollo web que permite a aplicaciones que se ejecutan en el cliente (generalmente en el navegador web) intercambiar datos con el servidor de forma asíncrona, en segundo plano, sin tener que recargar todos los datos (toda la página) que el usuario está visualizando. De esta forma, se mejora la interactividad, velocidad y usabilidad en las aplicaciones. *JavaScript* es el lenguaje en el que normalmente se implementan las funciones de llamada de AJAX que hacen uso del objeto *XMLHttpRequest*, disponible en los navegadores actuales, para realizar peticiones HTTP al servidor. El uso de esta técnica condicionaba a los LMS a que sus cursos fueran reproducibles únicamente desde ordenadores personales puesto que hasta hace poco, la mayoría de navegadores de dispositivos móviles (teléfonos móviles, PDA, Tablet PC, etc.) no incluían el objeto *XMLHttpRequest.*

```
var nFindAPITries = 0;
var API = null;
var maxTries = 500;

// The ScanForAPI() function searches for an object named API_1484_11
// in the window that is passed into the function. If the object is
// found a reference to the object is returned to the calling function.
// If the instance is found the SCO now has a handle to the LMS
// provided API Instance. The function searches a maximum number
// of parents of the current window. If no object is found the
// function returns a null reference. This function also reassigns a
// value to the win parameter passed in, based on the number of
// parents. At the end of the function call, the win variable will be
// set to the upper most parent in the chain of parents.

function ScanForAPI(win)
{
  while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent !=
win))
  {
   nFindAPITries++;
   if (nFindAPITries > maxTries)
   {
     return null;
   }
   win = win.parent;
  }
  return win.API_1484_11;
}

// The GetAPI() function begins the process of searching for the LMS
// provided API Instance. The function takes in a parameter that
// represents the current window. The function is built to search in a
// specific order and stop when the LMS provided API Instance is found.
// The function begins by searching the current window's parent, if the
// current window has a parent. If the API Instance is not found, the
// function then checks to see if there are any opener windows. If
// the window has an opener, the function begins to look for the
// API Instance in the opener window.

function GetAPI(win)
{
  if ((win.parent != null) && (win.parent != win))
  {
   API = ScanForAPI(win.parent);
  }
  if ((API == null) && (win.opener != null))
  {
   API = ScanForAPI(win.opener);
  }
}
```

Listado 38: Mecanismo de descubrimiento del API de SCORM

A partir de junio 2008, BlackBerry añadió soporte para el uso de *XMLHttpRequest* en la versión 4.6 de su sistema operativo[27], lo que habilitó a los *smartphones* BlackBerry para el uso de AJAX, facilitando la reproducción de cursos SCORM con interfaz de usuario rica, hasta ahora reservados a entornos PC.

5.2 Pasos para la adaptación de un LMS SCORM

Para adaptar un LMS que soporta SCORM de forma que pueda trabajar con itinerarios de aprendizaje adaptativos, utilizando el algoritmo de optimización por colonias de hormigas para itinerarios de aprendizaje (ASALI), es necesario adaptar diferentes entornos o módulos de su arquitectura para añadir funcionalidad adicional que permita al LMS gestionar Grafos de Itinerarios de Aprendizaje. El enfoque presentado en esta tesis describe los cursos, representados como nodos del grafo de itinerarios de aprendizaje, como cursos SCORM independientes.

La especificación para la navegación y secuenciación de contenidos en SCORM, IMS Simple Sequencing[28] (IMS-SS), permite definir reglas condicionales para la reproducción o no de contenidos de un curso, como por ejemplo, un test, en función de diferentes criterios: por ejemplo, que se hayan visualizado todos los contenidos del curso antes de iniciar el test o que una vez que se haya realizado el test, éste deje de estar activo. Pero, estas reglas de secuenciación y navegación, siempre deben estar referidas a objetos de contenidos reutilizables o SCOs (del inglés *Shareable Content Objects*) que para ser conformes con la especificación, como ya se ha comentado al principio de este Capítulo, deben incluirse empaquetados dentro del mismo archivo que conforma la unidad mínima de distribución de contenidos: el paquete SCORM. Esto es lógico para garantizar la reutilización del paquete, puesto que los contenidos de éste no dependerán de otros externos que pudieran cambiar con el paso del tiempo. Sin embargo en nuestro enfoque, el grafo de itinerarios de aprendizaje, hace referencia a paquetes SCORM independientes y gestiona la navegación entre ellos desde un punto de vista externo, por lo que no es suficiente únicamente con la modificación del LMS para depositar la cantidad de feromonas resultante tras la evaluación de un alumno en el test

[27] http://docs.blackberry.com/en/developers/deliverables/11844/Feature_AJAX_512507_11.jsp
[28] http://www.imsglobal.org/simplesequencing/

final de un curso, sino que será necesario dotar del modelo de datos adecuado para la gestión del grafo de itinerarios de aprendizaje.

Las adaptaciones necesarias son:

a) **Incorporar un entorno que permita la construcción del Grafo de Itinerarios de Aprendizaje**, así como la importación de grafos a partir de ficheros utilizando un lenguaje formal como el propuesto en este trabajo.

b) **Soportar operaciones CRUD[29] sobre elementos de los Grafos de Itinerarios de Aprendizaje**. Esto permitirá evolucionar los grafos según las necesidades, añadiendo o suprimiendo caminos alternativos, modificando las calificaciones mínimas deseadas en los nodos, etc. Algunas operaciones, como la actualización de la calificación media de cada nodo, pueden automatizarse. La modificación de la estructura del Grafo de Itinerarios de Aprendizaje, mediante la adición o eliminación de nodos, así como la modificación de restricciones de navegación, requerirá volver a ejecutar el proceso de transformación del GIA en un grafo dirigido procesable por el algoritmo. Estas operaciones, sólo podrían realizarse una vez una promoción haya finalizado y antes de comenzar la siguiente. Sin embargo puede evitarse este proceso si la modificación se realiza en el grafo dirigido resultante de la transformación inicial, por lo que si se ofrece una interfaz adecuada a los autores (equipo pedagógico) y éstos tienen un conocimiento suficiente del funcionamiento del algoritmo, podrían añadirse y eliminarse nodos en cualquier momento. En este caso para que un nodo recién añadido no se vea perjudicado por la diferente del rastro de feromonas con otros ya existentes, puede inicializarse el arco resultante con destino en el nodo añadido con el nivel medio de feromonas del resto de arcos vecinos (que tienen el mismo nodo origen que el recién creado). La eliminación de un arco no plantea mayor problema y tan sólo habría que sincronizarla para que no coincidiera con el proceso de cálculo de la probabilidad de elección de ningún alumno. Esto último puede resolverse con simple mecanismo de semáforos.

c) **Adaptar capa de presentación para presentar el avance en el itinerario de aprendizaje y obtener el curso siguiente.** Utilizaría las operaciones CRUD anteriores, para obtener el curso siguiente y mostrar información relevante acerca del itinerario completo: cursos ya realizados, calificación media hasta el

[29] CRUD (Create, Read, Update, Delete) son las siglas que definen las operaciones básicas de gestión de datos: crear, leer, actualizar y eliminar.

momento y si el curso siguiente es el final o no. Otras implementaciones podrían optar, en lugar de asignación del siguiente curso por el algoritmo, por la recomendación de las alternativas dejando la elección del siguiente curso a criterio del alumno. Para ello sería necesario también contar con los métodos necesarios para la obtención de la descripción del curso e información de interés, como el nivel de dificultad, que ayude al alumno a tomar la decisión.

d) **Gestionar las transiciones entre nodos**. En la especificación SCORM, el evento de finalización se produce cuando se invoca al método `Terminate`[30] del API. Cuando un curso se finaliza se han de evaluar los conocimientos del mismo para calcular la cantidad de feromonas a depositar y comenzar el proceso de decisión del siguiente curso en caso de que hubiera más de una opción posible. Por tanto, tras la señal finalización del curso se debe:

   - **Iniciar la evaluación para el nodo cuyo curso ha finalizado.** El sistema debe responsabilizarse de presentar al usuario un examen de evaluación. Este examen de evaluación puede estar incluido en el propio paquete de contenidos SCORM (lo que facilitaría su inicio) o puede generarse automáticamente por el sistema, por ejemplo para crear test automáticos a partir de un conjunto de preguntas y respuestas relacionadas entre sí, que pueden escogerse de forma aleatoria [97]. Tanto una como otra opción deberá indicarse en el GIA, especificando la URI desde la que se puede iniciar la evaluación de este curso. Podría apuntar a un Web Service externo encargado de proporcionar el test o apuntar a un recurso interno del paquete SCORM. El cliente se comunicará a través del API SCORM con el LMS para comunicar la calificación y realizar tareas de seguimiento. Esta comunicación se realiza mediante el establecimiento de valores a propiedades del modelo CMI, utilizando el método `SetValue()` del API SCORM. Es necesario modificar el LMS para que al recibir la calificación correspondiente a la evaluación de un curso, pueda calcular las feromonas a depositar en el arco seguido hacia ese nodo.
   - **Comprobar si el nodo finalizado es el último del grafo**. En caso de ser el último nodo se debe calcular la calificación media del alumno a lo largo de su itinerario y almacenarlo hasta que todos los alumnos de la promoción (la colonia de hormigas en terminología ACO) hayan

[30] Como para el resto de métodos del API, en la versión de 1.2 de SCORM el nombre del método viene prefijado con las siglas LMS.

finalizado o hasta que la promoción haya finalizado (se supere la fecha de finalización del programa formativo).

- **Depositar feromonas on-line (ϕ).** Cada vez que un alumno finalice la evaluación de un curso, el sistema debe calcular las feromonas ϕ a depositar, lo que tendrá influencia en la función de ajuste de los arcos vecinos en un nodo de decisión.
- **Obtener el curso asociado al siguiente nodo.** A partir de la tabla de decisión, el sistema decidirá el siguiente nodo para cada alumno, tras la finalización de un curso que no pertenece al último nodo.
- **Evaporar feromonas y reforzar el mejor camino.** Cuando todos los alumnos hayan finalizado la promoción, se procederá a la evaporación de las feromonas de cada arco y se reforzará al mejor camino. Este caso debe ser iniciado por medio de un evento, que se disparará cuando todos los alumnos de la promoción hayan finalizado.

e) **Proporcionar entorno de gestión del aprendizaje adecuado.** El entorno gestión del aprendizaje debe informar al tutor del itinerario seguido por cada alumno, por lo que será necesario persistir (por ejemplo en BD) las transiciones de un curso a otro así como diferenciar cuando un alumno ha realizado un curso de forma aislada, a cuando lo ha realizado incluido en un Programa Formativo que emplea Grafos de Itinerarios de Aprendizaje. El entorno de gestión de aprendizaje debe mostrar el camino seguido hasta el momento por cada usuario.

Para la comunicación con el LMS es necesario envolver la interfaz SCORM del gestor del aprendizaje, de forma que se mantenga la compatibilidad con la especificación. Para ello es necesario interceptar las llamadas al API de SCORM del curso con un componente Proxy (Figura 69) que será el encargado de recibir las llamadas desde el cliente SCORM del curso, remitiendo la llamada al LMS en caso de tratarse de eventos diferentes a la terminación del curso o a la evaluación, o al gestor de grafos de itinerarios de aprendizaje en caso contrario. Este componente se encargará de mantener las sesiones activas de aquellos alumnos que están en cada momento realizando un curso, así como la información necesaria para saber si ese curso pertenece a un GIA o no.

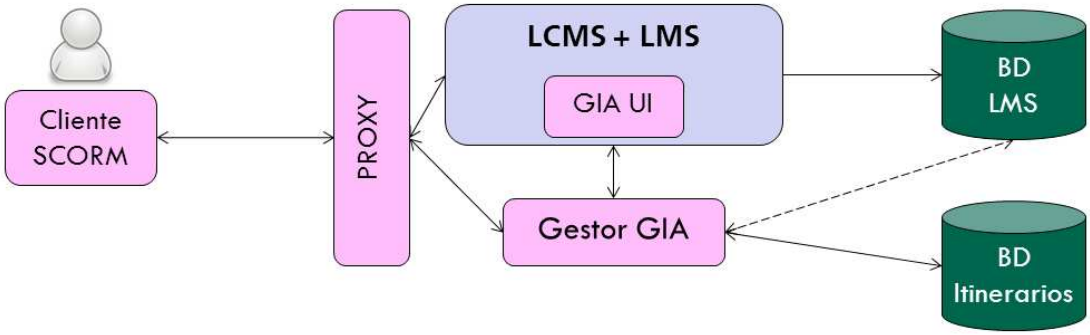


Figura 69: Arquitectura para la integración de ASALI en un LMS

Hay que tener en cuenta además que, a diferencia de la simulación empleada para la experimentación en este trabajo, el algoritmo carecerá del bucle que regula su funcionamiento. En el caso de integración en un LMS el número de iteraciones será siempre 1, y la evaporación y refuerzo ser producirá al terminar todos los alumnos de una promoción su itinerario o bien al finalizar la promoción (teniendo en cuenta sólo los que hayan completado la formación). El concepto de promoción, ligado al espacio de tiempo en el que se desarrolla la formación, con una fecha de inicio y otra de fin, es pues muy importante.

A diferencia del algoritmo empleado para la simulación, la aplicación de los puntos anteriores para la adaptación de un LMS SCORM obliga a almacenar los datos, en un sistema de almacenamiento de datos persistente, por ejemplo en una base de datos (Figura 70).

La integración del modelo de datos propuesto en la Figura 70 representa la integración del modelo de datos necesario para soportar Grafos de Itinerarios de Aprendizaje en un LMS. La integración a nivel de modelo de datos se realiza a través de las tablas que representan al usuario (*User*) y a los Recursos (*Resource*), puesto que todos los LMS analizados durante el periodo investigador [5] incluyen estas entidades en su modelo de datos.

Este modelo de datos permite conocer en cada momento, para cada GIA, el nodo en el que se encuentra cada alumno y los nodos ya recorridos por éste. Además permite conocer en cuantos GIA participa cada alumno, así como la topología de cada GIA a través de las relaciones N:M GIA-Node y GIA-Edge.

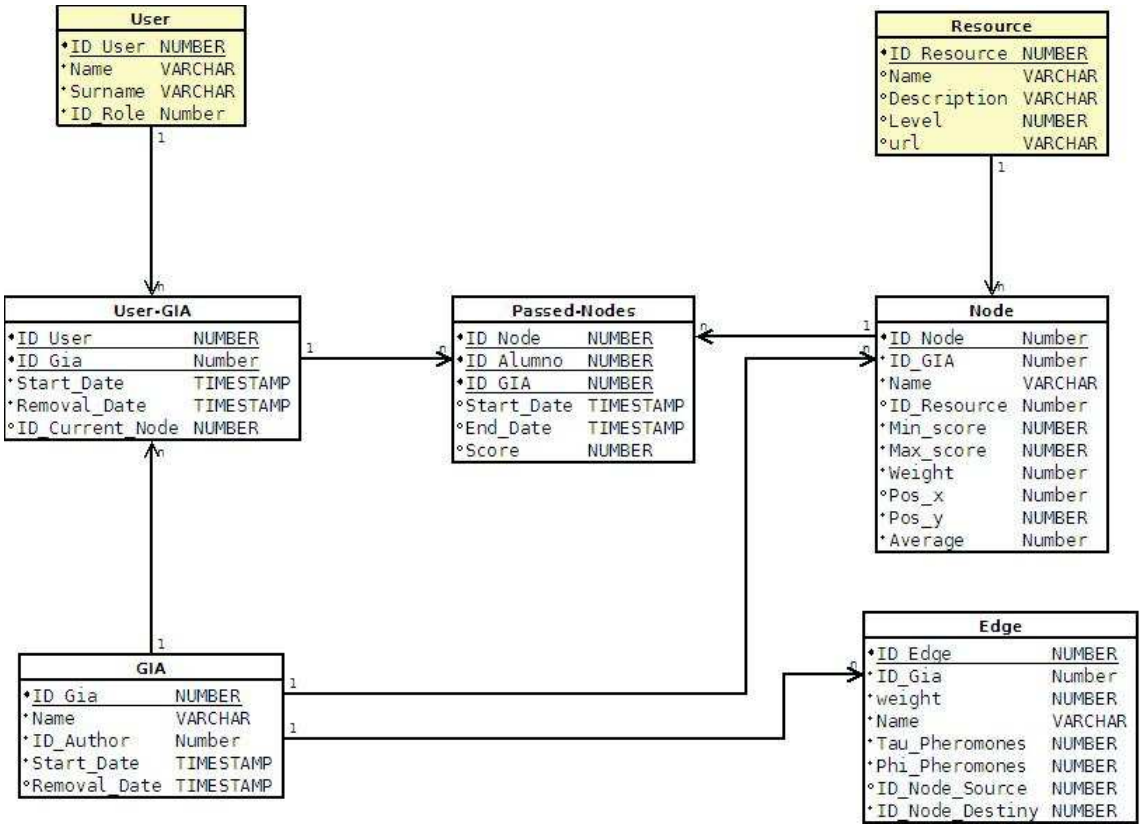


Figura 70: Modelo de datos para persistir el GIA e integrarlo en el modelo de datos de un LMS

5.3 Implantación de plataformas de e-learning en SOA

En contra del enfoque vertical y aglutinador que sobre los procesos de negocio involucrados en la industria del e-learning tienen los LMS actuales, muchos autores comienzan a animar a la adopción de arquitecturas orientadas a servicios (en inglés *Service Oriented Architecture, SOA*), que podrían dejar obsoleta los actuales estándares de facto [95].

SOA es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio. Permite la creación de sistemas de información escalables que reflejan el negocio de la organización, facilitando la exposición e invocación de servicios (comúnmente pero no exclusivamente servicios web), lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

Como definición de servicio podríamos decir que "es una entidad que puede ser usada por una persona, programa u otro servicio, que realiza funciones de cómputo,

de almacenamiento o de comunicación con otros usuarios, dispositivos físicos o incluso con otros servicios" [93].

Los servicios pueden ser orquestados con el fin de construir y ofrecer nuevas funcionalidades más complejas, abriendo así una nueva vía para la colaboración entre diferentes plataformas de e-learning. Varias propuestas [90]-[92] sugieren la publicación como servicio de la funcionalidad que puede ser utilizada por otras plataformas, apostando claramente por una adaptación a SOA de las especificaciones más extendidas en la industria del e-learning [96]. Según Vossen y Westerkamp [95], esta evolución hacia una arquitectura orientada a servicios podría revolucionar la industria mediante la especialización de los diferentes actores implicados (Figura 71), permitiendo la aparición de proveedores de contenidos, de aplicaciones (por ejemplo, aplicaciones colaborativas como wikis, blogs, chats, foros, etc.), de metadatos, de cursos, o de itinerarios formativos.

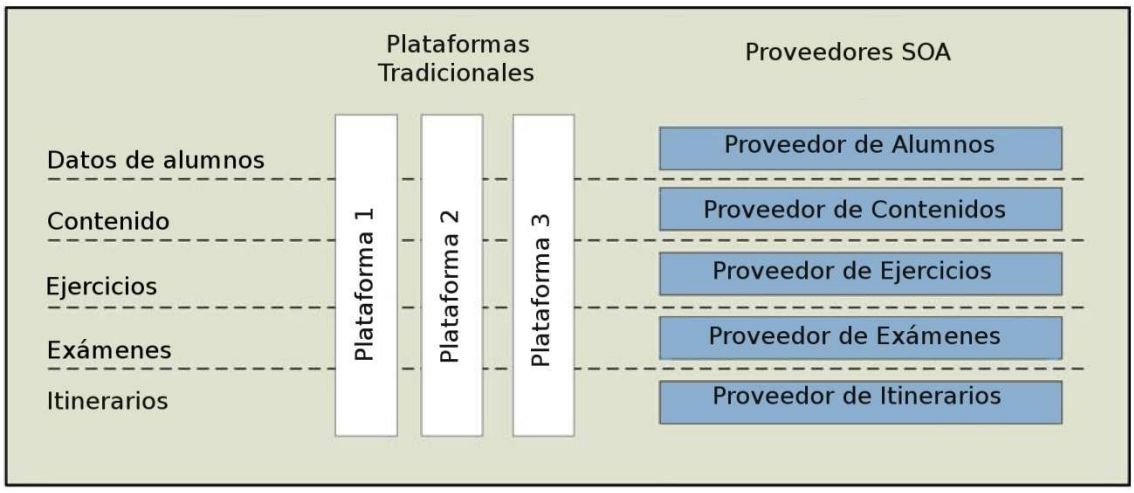


Figura 71: Relación entre plataformas tradicionales de e-learning con una orientación SOA

La visión de Vossen y Westerkamp acerca de la desaparición de los LMS como actualmente los conocemos y su disgregación en servicios, facilitaría la integración de las funcionalidades básicas de un LMS en la las empresas de mediano y gran tamaño.

Al margen del soporte que dan las diferentes tecnologías al concepto de servicio, una arquitectura orientada a servicios se organiza de acuerdo a ciertas reglas bien definidas: los servicios deben tener interfaces descritas mediante un lenguaje formal que defina las operaciones que puedan solicitar otros servicios. Para completar el dinamismo del modelo es además necesario que los servicios puedan ser publicados en registros accesibles mediante mecanismos estandarizados [94], que permitan su descubrimiento y su posterior utilización a otros servicios o usuarios. WSDL (*Web*

Service Description Language) y WADL (*Web Application Description Language*) son los lenguajes estándares para la descripción de servicios y aplicaciones (típicamente servicios web REST[31] [102]) web respectivamente. Ambos son lenguajes basados en XML, pero mientras el primero es el estándar definido por W3C, este organismo no piensa dedicar trabajo a la evolución del segundo, que fue enviado para su aprobación por varios de sus miembros. Los servicios web son invocados normalmente en combinación con el protocolo SOAP, especialmente indicado para el intercambio de información estructurada sobre protocolos de la capa de aplicación (especialmente HTTP o SMTP).

Adquieren relevancia en arquitecturas orientadas a servicios los proveedores de servicios y los almacenes o registros de servicios (y las agencias de descubrimiento de servicios que trabajan directamente con los registros) con los que estos proveedores trabajan (Figura 72).

Existen distintas tecnologías que pueden ser catalogadas dentro del paradigma SOA, entre las que cabe mencionar: Servicios Web, OSGi o Jini[32]. Cada una de ellas tiene un ámbito de aplicación y objetivos diferentes; por ejemplo, Jini se orienta a la publicación de servicios en redes locales y de corto alcance, los Servicios Web tratan de definir las interacciones entre servicios disponibles en Internet, y OSGi busca la integración de los servicios domésticos, de área local y personal con Internet.

Si bien SOA no está estrictamente relacionado con Servicios Web, éstos cada día adquieren mayor relevancia debido a sus características, que permiten la interoperabilidad entre aplicaciones a través del uso de estándares abiertos, facilitando la integración en el sistema de información de una organización de diferentes aplicaciones informáticas, independientemente del lenguaje de programación en el que estuvieran programadas.

Según Thomas Erl [103] los principios de la Orientación a Servicios son:

- **Reusabilidad.** Los servicios deben ser reutilizables, bien dentro de la propia aplicación, del dominio la organización (en una Intranet) o en dominio público (en Internet).

[31] REpresentational State Transfer
[32] Jini, inicialmente desarrollado por Sun Microsystems, fue liberado con licencia de código abierto de Apache, siendo transferido posteriormente a Apache Foundation bajo el proyecto River. http://river.apache.org/

Figura 72: Esquema de relaciones en SOA (particularizado para Web

- **Contrato formal.** Los servicios deben ser definidos a través de un lenguaje formal. Esta definición constituirá el contrato del servicio y en él figurará el nombre del servicio, la forma de acceso, las operaciones que ofrece, los datos de entrada de cada una de las operaciones y los datos de salida. De esta manera, todo consumidor del servicio. En el caso de los Servicios Web, se utiliza WSDL para la definición del servicio.
- **Acoplamiento débil entre servicios.** Los servicios serán independientes unos de otros de forma que cada vez que se vaya a invocar a un método de un servicio, se accederá a él a través del contrato, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama.
- **Composición de Servicios.** Se pueden definir un servicio más complejo (o de más alto nivel) como composición se servicios más simples (o de más bajo nivel).
- **Autonomía.** Los Servicios deben de ser autónomos, teniendo cada uno su propio entorno de ejecución.
- **Ausencia de estado.** Un servicio no debe guardar ningún tipo de información de estado. Una aplicación está formada por un conjunto de servicios, lo que implica que si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución es que un servicio sólo contenga lógica, y que toda información esté almacenada en algún sistema de información (BD, Ficheros, Colas de mensajes, etc).

- **Descubrimiento de servicios.** Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de los Servicios Web, el descubrimiento se logra publicando los interfaces de los servicios en registros UDDI (*Universal Description, Discovery, and Integration*) [94].

La integración se debe llevar a cabo mediante un mecanismo que permita a los servicios cooperar entre ellos, diferenciándose entre orquestación (cuando dicho mecanismo es controlado por una única unidad) y coreografía (cuando o existe un punto único de control en la colaboración entre servicios).

Los esfuerzos por estandarizar un lenguaje formal para la definición de coreografías de servicios web ser cerraron el 10 de julio de 2009 con la publicación de la especificación WS-CDL (*Web Services – Choreography Description Language*), aunque en versión *Recomendación para Candidato*. WS-CDL es un lenguaje para definir como colaboran los participantes en relaciones punto a punto, es decir, por pares. El lenguaje está basado en XML y utiliza algunos aspectos inspirados en el Cálculo-π [104] pero no ha tenido relativo éxito en la industria hasta ahora. En cambio, sí han adquirido gran relevancia la orquestación de procesos, para los que BPEL (*Business Process Execution Language*) se ha impuesto como lenguaje estándar para la composición de servicios web, a través de la cual se consigue definir la ejecución de procesos de negocio complejos.

Una arquitectura SOA puede representarse de forma esquemática como una pila de capas de diferente nivel de abstracción (Figura 73). Esta representación abarca desde la capa de almacenamiento de datos a las capas de presentación de las aplicaciones. Las aplicaciones hacen uso de procesos de negocio que orquestan la secuencia de pasos a realizar para obtener los datos necesarios. Esta obtención de los datos es llevada a cabo a través del consumo de los servicios, los cuales a su vez acceden a los componentes de negocio de la organización, que brinda el procesamiento requerido para la función deseada. Los componentes de negocio por lo general tienen una capa de acceso a datos que les permite acceder distintos repositorios de datos de aplicaciones que funcionan actualmente en la organización y que pueden involucrar a sistemas ya obsoletos pero aún en uso.

Un LMS cuya arquitectura aplique la filosofía de orientación a servicios, facilitará la adaptación del mismo para trabajar con itinerarios de aprendizaje

adaptativos, mediante la definición de una capa de servicios que proporcionen acceso a la funcionalidad propia del LMS. Como ilustra la Figura 72, SOA diferencia tres entidades principalmente: consumidor, proveedor y registro de servicios. Siguiendo la propuesta de Bahrami et al. [105] estas tres entidades pueden verse reflejadas en las siguientes entidades de una arquitectura de e-learning basada en SOA:

- **El portal de e-learning.** Actúa como consumidor de servicios. Utiliza los servicios de e-learning proporcionados en la web por los Centro de Servicios de e-learning e interactúa con los usuarios para proporcionarles dichos servicios. La comunicación entre el portal y el centro de servicios ser realiza a través de SOAP.
- **Proveedor de Servicios de e-learning.** Pueden existir más de uno. Los portales buscan en la red los servicios ofrecidos por éstos, por lo que sus servicios deben quedar registrados en el Registro de Servicios de e-learning.
- **Registro de Servicios de e-learning.** El Registro de Servicios actúa como bróker de servicios. Recibe los descriptores de servicios por parte de los proveedores y las peticiones de búsqueda de los consumidores.

5.3.1 El Portal de e-learning

Aplicando la separación de servicios por su nivel de abstracción de la Figura 73 y la filosofía de arquitectura SOA, Bahrami et al. presenta en [105] un modelo de arquitectura para los portales de e-learning en el que se diferencian claramente tres capas del portal de e-learning, Front-end, Middleware y Back-end, y que hemos modificado para recoger la necesidad de contar con un servicio de gestión de itinerarios (Figura 74).

El Front-end constituye la capa de presentación y hace uso de los servicios ofrecidos por la capa intermedia o middleware. El contenido puede ser presentado a los usuarios finales (administradores, instructores, alumnos y diseñadores) en un entorno web o en un cliente específico de la plataforma del cliente (por ejemplo, los alumnos podrían usar un cliente específico en su dispositivo móvil para realizar los cursos). La comunicación entre las dos capas se realiza a través del protocolo SOAP. EL Front-end encapsula la funcionalidad para consumir los servicios proporcionados por la capa Middleware, los cuales conoce a través del contrato WSDL de antemano.

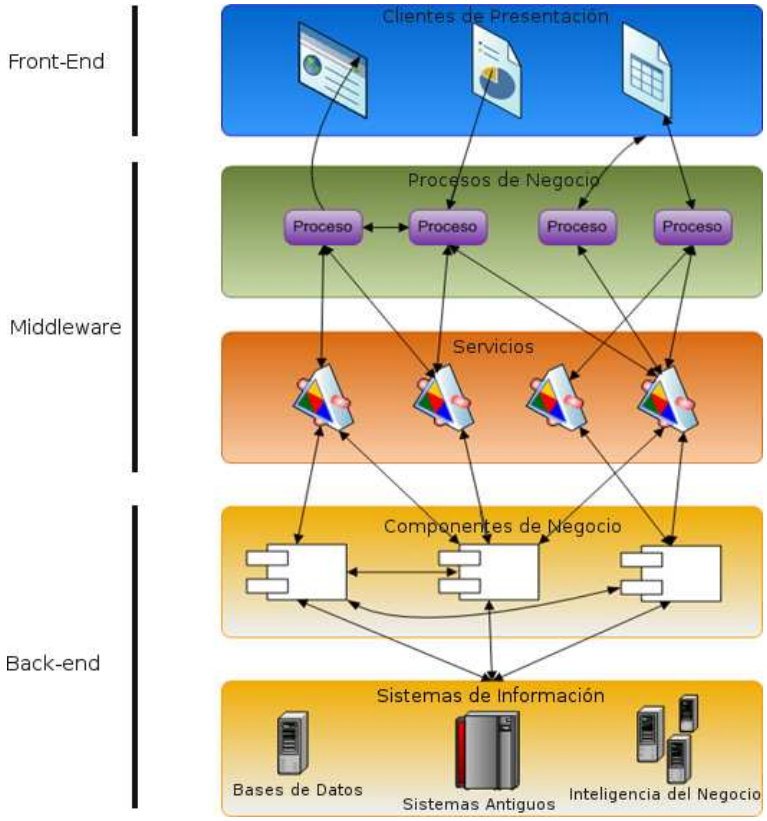


Figura 73: Representación por capas de una Arquitectura SOA.

El Middleware es el motor del portal de e-learning y proporciona funciones lógicas y de control a la capa de presentación (Front-end). En esta capa se definen principalmente seis servicios, si bien tienen cabida otros servicios encargados de realizar operaciones de utilidad (por ejemplo traducción de identificadores de usuario para el acceso desde dispositivos móviles). Además, en esta capa intermedia pueden realizarse divisiones de servicios en función de su granularidad, distinguiendo entre servicios compuestos (cuya interfaz ofrece métodos que ofrecen datos compuestos de los datos obtenidos de dos o más servicios más simples) y servicios atómicos, o entre servicios encargados de orquestar a otros servicios más simples, tal y como se representa en la Figura 73.

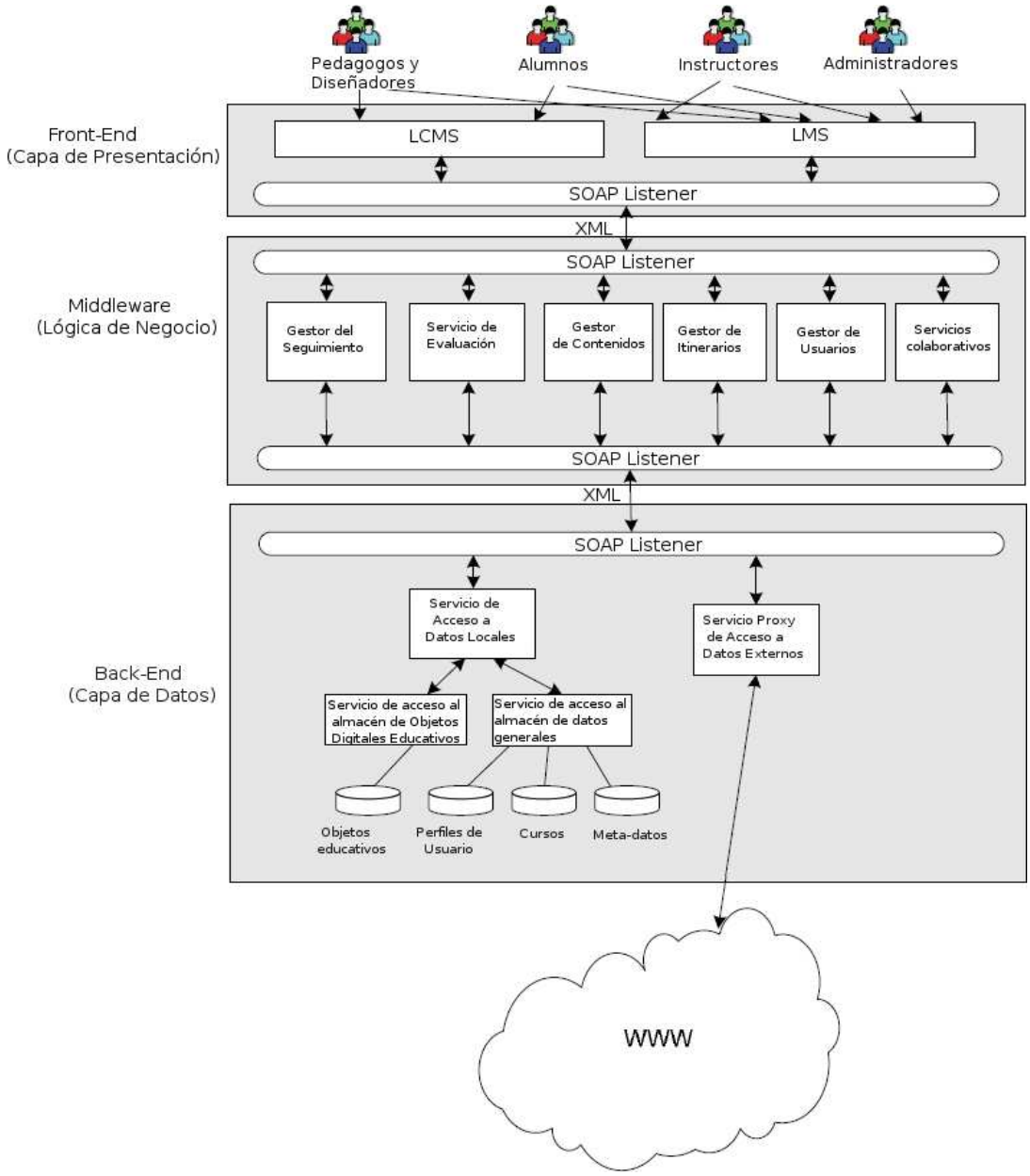


Figura 74: Esquema de arquitectura del portal de e-learning con arquitectura SOA

Los servicios destacados que la capa Middleware debe ofrecer en un LMS según la arquitectura SOA son:

- **Entorno de seguimiento y evaluación** (proveedor de servicios de seguimiento y evaluación). Permite llevar a cabo acciones relacionadas con el seguimiento del avance del curso, tales como establecer un nuevo hito de avance (porcentaje de avance) de un determinado curso y actualizar dedicación (tiempo invertido) de un alumno. Para adaptar un LMS a una arquitectura orientada a servicios, el LMS o un proveedor de servicios ha de exponer esta funcionalidad como servicio, por ejemplo publicando servicios

web que actúen de envoltorio[33] de la funcionalidad encargada del seguimiento en el LMS.

- **Servicio de Evaluación.** El servicio de evaluación se encarga de proporcionar a la capa de presentación los mecanismos para realizar la evaluación de un curso. Su complejidad depende del método de evaluación empleado, pudiendo desde actuar como envoltorio de la funcionalidad relacionada en el LMS (recibir y procesar las calificaciones de un test dentro de un curso) hasta ofrecer funcionalidades extras como, por ejemplo, la creación bajo demanda de tests a partir de bases de datos de preguntas y respuestas relacionadas con el curso en cuestión [97]. Estas funcionalidades no son provistas por el Servicio de Evaluación del Middleware, sino que este actúa de proxy de un proveedor de servicios encargado de ofrecer realmente esta funcionalidad.
- **Servicio de Gestión de Contenidos**. Permite añadir, eliminar, obtener y modificar contenidos educativos. Típicamente estos contenidos serán paquetes SCORM o recursos didácticos susceptibles de incorporarse en un paquete SCORM, pero depende de la funcionalidad ofrecida por el sistema de e-learning. Algunos LMS como OLAT integran funcionalidades de LCMS, proporcionando entornos de producción online de contenidos que pueden ser incorporados a un repositorio para su uso por otros autores, o bien publicados directamente en algún espacio de aprendizaje. Este entorno también es el encargado de distribuir un curso determinado a través de su ubicación (URL). Esta funcionalidad es proporcionada por proveedores de servicios, actuando el Servicio de Gestión de Contenidos como intermediario.
- **Entorno de Gestión de Usuarios.** Este servicio de la capa intermedia se encarga de las operaciones de alta, baja y modificación tanto de usuarios como de grupos de usuarios (casi todos los entornos de e-learning permiten la creación de grupos de usuarios por nivel educativo, por idioma o por cualquier otro atributo de los usuarios). Para adaptar un LMS no SOA a esta arquitectura habría que, al igual que para el resto de entornos de esta capa, proporcionar un envoltorio como Servicio Web de los módulos del LMS que se encargan de esta funcionalidad, de forma que el propio LMS actúe como proveedor de servicios, siendo el Servicio de Gestión de Usuarios de la capa Middleware un mero intermediario.

[33] Ver patrón de diseño envoltorio (*wrapper*) en [86].

- **Servicio de Gestión de Itinerarios Adaptativos de Aprendizaje.** El entorno de Gestión de Itinerarios de Aprendizaje es el encargado de orquestar al resto de servicios para integrar la propuesta de adaptabilidad del itinerario de aprendizaje de este trabajo. Es un servicio que además de ofrecer la funcionalidad propiamente relacionada con la gestión y manipulación de itinerarios de aprendizaje, actúa como orquestador del resto de servicios del middleware (Figura 75). Así, utiliza los métodos del servicio web "Gestor del Seguimiento" para actualizar la información relativa al seguimiento del alumno en un curso (porcentaje de avance, tiempo invertido, etc); hace uso del Servicio Web "Gestor de Evaluación" para comunicar al LMS las calificaciones obtenidas por el alumno en cada curso del itinerario; delega en el Servicio Web "Gestor de Usuarios" las consultas acerca del perfil del alumno, sus calificaciones históricas y privilegios en el sistema; y por último, se apoya en el Servicio Web "Gestor de Contenidos" para obtener los cursos pertenecientes a cada nodo del itinerario. Como servicio, aporta al sistema la funcionalidad propia del Gestor de Itinerarios de aprendizaje adaptativos: métodos para la parametrización del algoritmo ASALI, operaciones de manipulación de los elementos de un GIA (sirven dan soporte a aplicaciones de autor en el front-end) y gestión de las feromonas del GIA, entre otras.

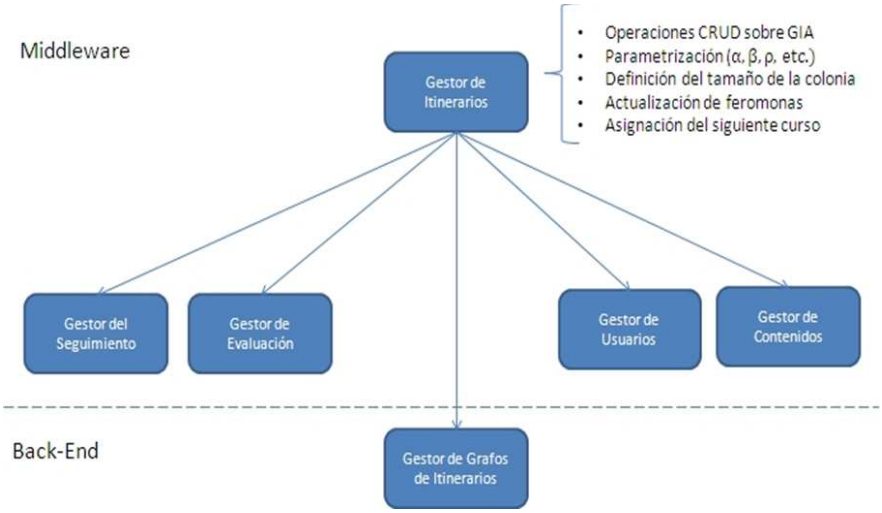


Figura 75: Servicio de Gestión de Itinerarios como orquestador del resto de servicios del Middleware del LMS

El servicio Gestor de Grafos de Itinerarios es el encargado de trabajar con el modelo de datos específico para los Grafos de Itinerarios de Aprendizaje de este trabajo e incluye los atributos necesarios en las tablas de usuario y de

recursos del modelo de datos propio del LMS, tal y como se muestra en la Figura 70.

- **Servicios de colaboración.** Muchos de los LMS actuales ofrecen funcionalidades relacionadas con la realización de tareas de forma colaborativa: chats, blogs, foros de discusión, gestión documental, integración con herramientas de comunicación multimedia (audio-conferencias y video-conferencias) y, más recientemente, integración con conocidas plataformas de redes sociales (Twitter, Facebook, LinkedIn, etc.). Generalmente, estas características ya siguen la filosofía SOA en la mayoría de los LMS actuales, por lo que estos servicios serían ofrecidos por los LMS, quedando en el middleware unos simples servicios proxy, que faciliten el cambio de un LMS a otro. Recientemente, debido a la explosión de las redes sociales, se está produciendo el camino inverso en muchas organizaciones: en lugar de incorporar accesos a redes sociales en sus LMS, están incorporando estos últimos a plataformas de redes sociales corporativas, como por ejemplo Jive[34], aunque utilizando unos mecanismos de integración ligeramente diferente:
    - **Publicación vía RSS.** Los cursos a los que un usuario puede acceder se publican mediante un Servicio Web o REST utilizando RSS (Figura 76).
    - **Implementar Single-Sign On.** De esta forma cuando un usuario del portal de la red social accede a un curso del LMS no tiene que volver a autenticarse en el LMS. De esta forma los usuarios de la red social perciben que el curso que han seleccionado del listado RSS anterior forma parte del propio portal de red social. Esto, obviamente requiere que el LMS soporte este mecanismo de autenticación.
    - **Personalizar espacios en la Red Social.** Para una integración total en este tipo de entornos, es necesaria la creación de espacios privados para cada tipo de grupo, promoción o curso, con acceso a los cursos e itinerarios de su grupo. Se suelen desarrollar lógica adicional que automáticamente enrole a cada usuario en el espacio que necesita.
    - **Desarrollar componentes específicos para el seguimiento del curso.** Recientemente se están desarrollando componentes web específicos (*widgets*) que permiten hacer uso de los servicios anteriormente indicados, de forma que se facilite la integración de un

[34] www.jivesoftware.com

LMS y LCMS con portales de Redes Sociales con el objetivo de integrar mecanismos de enseñanza-aprendizaje guiados con los procesos de construcción informal de conocimiento que aparecen en las redes sociales [106]-[107].

La capa denominada Back-End, incluye los servicios de acceso a los datos tanto de contenidos educativos digitales como del modelo de datos del LMS en el que se relacionan usuarios, perfiles de usuarios, privilegios de acceso, grupos, niveles educativos, etc. El acceso a los almacenes de contenidos digitales también se ha de exponer como servicios web para facilitar la integración con el *middleware* y con otras aplicaciones [108].

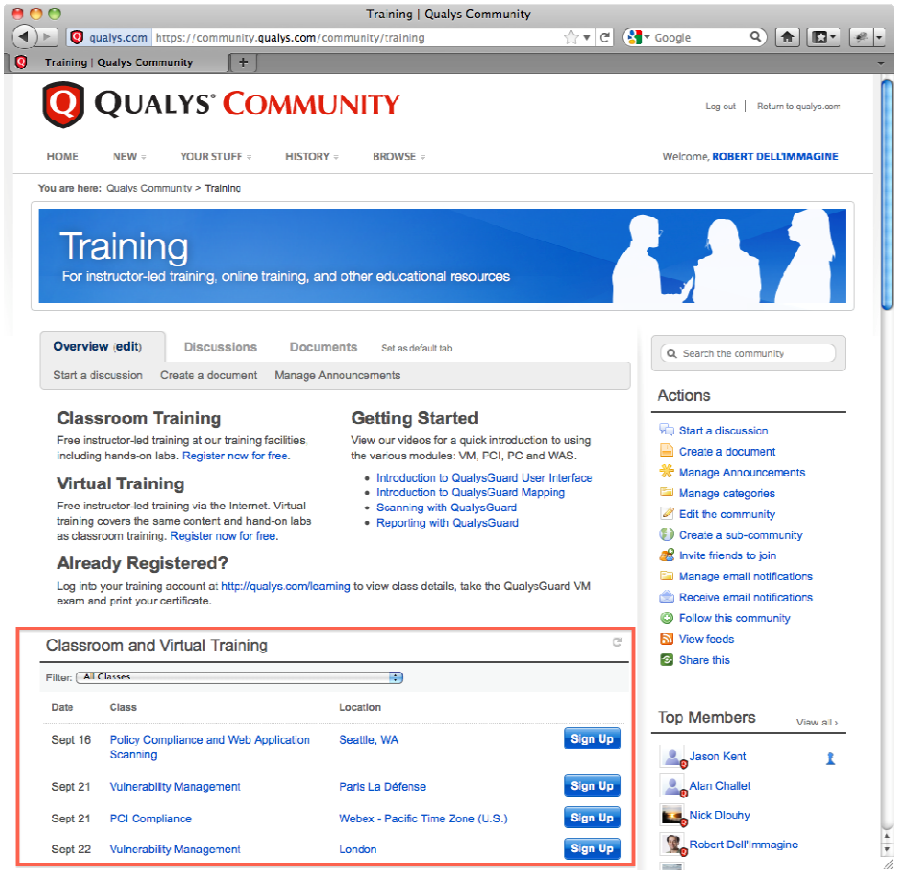


Figura 76: Ejemplo de integración de LMS con Jive a través de RSS (recuadro rojo)

5.3.2 Proveedor de Servicios de e-learning

No es único. Pueden existir proveedores especializados en cada una de las necesidades: gestión de objetos educativos, herramientas de creación de contenidos

(herramientas de autor), servicios para el seguimiento y gestión del aprendizaje, servicios de colaboración, etc.

Ofrecen los servicios que son expuestos directamente al front-end por los servicios de la capa Middleware. Los servicios de la capa Middleware actúan como de envoltorio de los servicios proporcionados por el proveedor, independizando a los clientes del front-end de la tecnología de comunicación empleada y de protocolos de comunicación.

Figura 77: Representación del rol "proxy" de los servicios del middleware

En una arquitectura SOA, un único LMS puede ofrecer los servicios de gestión del seguimiento, gestión de contenidos, gestión de usuarios y herramientas colaborativas, publicando su funcionalidad como un Servicio Web. La exposición de esta funcionalidad típica de un LMS como Servicios Web, facilita la integración de los entornos de aprendizaje en otras aplicaciones y portales de una organización, por ejemplo un ERP o con una plataforma de redes sociales corporativas, facilitando de esta forma el acceso a cursos de formación. Además, facilita la integración con un servicio de gestión de grafos de itinerarios de aprendizaje, haciendo transparente esta característica para el usuario final y haciendo posible la propuesta de este trabajo.

En los últimos años, la investigación relacionada con la integración de los LMS actuales con otras aplicaciones en las organizaciones ha sido muy intensa [109][113]. Algunas especificaciones como IMS-TI (*IMS Tools Interoperability*) o CCSI (*CopperCore Service Integration*) promueven la integración de herramientas externas en los propios LMS. IMS TI propone el uso combinado de Servicios Web y soluciones proxy, mientras que CCSI, sin mencionar Servicios Web, propone una capa intermedia en la arquitectura de los LMS, situada entre la lógica de presentación y la de las herramientas a integrar, y cuyo objetivo es adaptar las llamadas entre ambas.

Una de las iniciativas más interesantes desde el punto de vista de la integración de un LMS con otras aplicaciones dentro de una organización es la que ofrece la solución comercial de Firmwater Inc.[35], Firmwater LMS, que ofrece una API publicada como Servicios Web, que permite integrar una funcionalidad limitada: autenticación del usuario, crear sesión de usuario y realizar búsquedas en el LMS.

5.3.3 Registro de Servicios de e-learning

El estándar de facto para el registro de servicios en SOA es UDDI (Universal Description, Discovery, and Integration). El objetivo de UDDI es proporcionar un marco abierto para registrar y localizar servicios de negocios.

La localización de Servicios Web se realiza exactamente igual que con sitios Web estándar o aplicaciones basadas en la Web: o se conoce la URL o se busca un motor de búsqueda para obtener un sitio web que reúna los criterios de búsqueda. Si se conoce la dirección del servicio web que se quiere invocar o la dirección de su interfaz (WSDL, algún tipo de XML, interfaz Java estándar, etc.), es posible contactar el servicio directamente. En cambio, si no se conoce la dirección del servicio deseado, es necesario consultar un registro, una especie de "páginas amarillas", y esa es la función que realiza UDDI, proporcionando un mecanismo neutral de publicación y localización de descripciones de servicios.

Un nodo de registro UDDI actúa como proveedor de servicios, publicando servicios de negocios a demanda; como registro de servicios, exponiendo un directorio explorable de servicios web; y como solicitante de servicios, localizando un servicio solicitado y enlazando al cliente con ese servicio. Tanto el registro como la localización se efectúan enviando comandos UDDI en el cuerpo de un mensaje SOAP a un registro UDDI. Para acceder a los servicios registrados, el cliente envía mensajes SOAP codificados según el estilo de codificación del esquema UDDI. La solicitud SOAP es enviada al servidor e interpretada por el procesador SOAP del nodo de registro UDDI. Las llamadas UDDI se hacen en el servicio de registro que, a su vez, consulta la base de datos UDDI y tras obtener los resultados, prepara la respuesta y luego se la remite al servidor Web para su envío a la aplicación cliente (Figura 78).

[35] www.firmwater.com

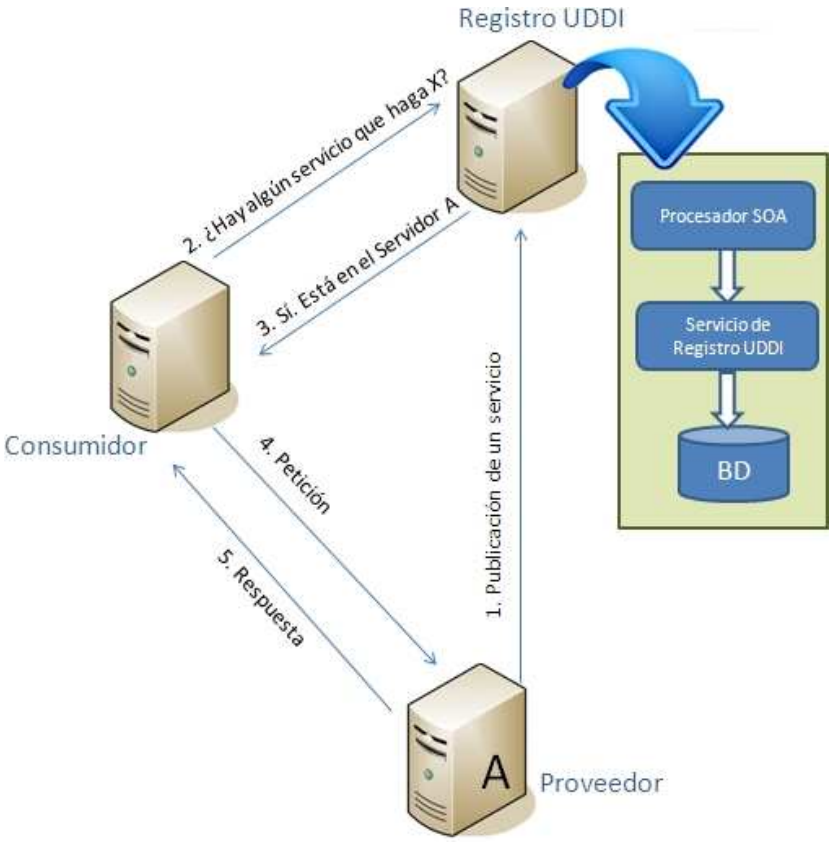


Figura 78: Esquema de funcionamiento de un registro UDDI

Una de las iniciativas más interesantes en cuanto a utilización de un registro de servicios es el Proyecto Agrega [114], una federación de repositorios de contenidos digitales, con nodos instalados en cada una de las Comunidades Autónomas españolas. Cada nodo permite almacenar objetos digitales SCORM 2004 etiquetados con metadatos. Estos nodos facilitan un conjunto de servicios tales como búsqueda, visualización o creación de nuevo material, que están disponibles tanto para aplicaciones propias del nodo como para aplicaciones externas al mismo. Además, se ha implementado una interfaz de interoperabilidad en la que se publican los servicios en forma de servicios Web, muchos de ellos siguiendo la filosofía Web 2.0 [115].

A través de esta federación de repositorios es posible localizar contenidos educativos mediante búsquedas por sus metadatos, expresados según el perfil LOM-ES[36], así como servicios de específicos de búsqueda y catalogación de contenidos.

[36] LOM-ES v 1.0 es el estándar de etiquetado de Objetos Digitales Educativos que satisfacen las necesidades específicas de la comunidad educativa española. Es una extensión del LOM de IEEE-LTSC. http://www.lom-es.es/

Aunque inicialmente basado en SOAP, se pretende mejorar el rendimiento de Agrega mediante una migración de sus interfaces a servicios REST, que presentan como ventaja un mejor grado de procesamiento de XML [116].

5.3.4 Gestión de Itinerarios Adaptativos como Servicio Web

La integración del LMS en una arquitectura orientada a servicios y la incorporación de servicios de gestión del Grafo de Itinerarios de Aprendizaje facilita la puesta en práctica de la propuesta presentada en este trabajo

Para controlar el acceso restringido a usuarios autorizados, se pueden emplear diferentes técnicas como el empleo de cookies o Single Sign-on entre el Servicio de Gestión de Itinerarios y el LMS.

Los métodos expuestos al cliente final son aquellos relacionados con la gestión de los eventos que produce un cambio en el estado de un usuario en el grafo: iniciar y abandonar una sesión, solicitar la lista de grafos en los que participa o está matriculado el alumno, preguntar por estado actual, o avanzar al siguiente nodo. Algunos de los métodos necesarios son:

- **start**: Inicia una sesión de un usuario. El método devuelve la sesión que será utilizada por el cliente en el resto de métodos.
- **stop**: Finaliza la sesión. Es llamado por el cliente, bien explícitamente o bien implícitamente al cerrar el cliente (por ejemplo como resultado del evento de finalización del cliente o el que se produce al cerrar el navegador en aplicaciones web).
- **getStatus**: Devuelve un objeto con el estado del usuario. Este objeto contiene información acerca del nodo actual en el que se encuentra el usuario, la calificación media hasta el momento, el identificador del grafo en uso (sólo uno en uso simultáneamente) y si el itinerario ha sido finalizado o no.
- **walk**: Este método es el encargado de iniciar el proceso de cálculo de feromonas Φ y del siguiente nodo en el grafo, modificando así el estado actual del usuario. Devuelve la URL desde la que obtener el curso SCORM del siguiente nodo, facilitando así el inicio del siguiente curso en el cliente.
- **launchTest**: Inicia un test de evaluación externo a un curso. Este método es invocado para la ejecuión de tests externos a los cursos SCORM de un nodo, permitiendo la integracón de LMS que utilizan generación automática de tests y en general evaluación externa a los cursos SCORM. Una vez

realizado el test, este método se encarga de iniciar los procesos de actualización de estado y avance implementados en el método `walk` anterior. Este método recibe como parámetros la sesión de usuario, el identificador del grafo actual y el identificador del nodo actual en dicho grafo. A partir del identificador del nodo, obtiene la ubicación del test de evaluación, que puede ser una URL o el *endpoint* de un servicio web del LMS encargado de generar los tests automáticamente, por lo que es necesario que el sistema gestor de itinerarios, almacene en su modelo de datos la ubicación del test de evaluación para cada nodo.

- **listGraphs**: Ofrece la lista de identificadores de los GIA en los que puede participar el alumno, de forma que se facilite al cliente la representación de las opciones posibles y la alternancia entre diferentes grafos, si bien sólo uno de ellos estará activo en cada instante.
- **setCurrentGraph**: Se utiliza para cambiar de GIA entre los posibles ofrecidos por el método anterior. Permite cambiar de programa formativo entre los suscritos por el alumno.

Otros métodos susceptibles de incorporarse al servicio de gestión de itinerarios de aprendizaje son los relacionados con las operaciones de creación, lectura, modificación y eliminación de grafos, nodos a un grafo, etc. Es decir, herramientas de autor para la creación de grafos de itinerarios de aprendizaje. Estas herramientas permitirían definir las restricciones de navegación del grafo (Figura 7) así como procesarlo para obtener el Grafo de Itinerarios de Aprendizaje en formato procesable mediante computación (Figura 38).

Capítulo 6. Conclusiones y trabajo futuro

6.1 Retrospectiva

El origen de esta tesis es el proyecto fin de carrera del autor, defendido en el año 2002. En él se define un entorno de enseñanza on-line basado en la videoconferencia en tiempo real, utilizando el protocolo RTP sobre UDP. A partir de este proyecto, el doctorando comienza a participar en proyectos de I+D+i europeos, desarrollando una pasarela residencial (Figura 79) basada en una arquitectura orientada a servicios en el proyecto Osmose y, abordando el problema de la adaptación de contenidos multimedia y su aplicación a la educación en el proyecto Passepartout, en el que, junto a otros investigadores europeos, se desarrolló un LMS compatible con SCORM aprovechando la infraestructura software del proyecto Osmose. Como rasgo distintivo, en el proyecto Passepartout se implementó un LMS compatible con la especificación SCORM, definiendo la incorporación de reglas de adaptación y personalización de contenidos dentro de los cursos, que eran distribuidos como bundles OSGi. El objetivo del proyecto Passepartout era construir un escenario en el que investigar nuevas formas de interacción hombre-máquina, dentro del marco de la investigación en nuevas técnicas de inteligencia ambiental.

Aprovechando los resultados del proyecto OSMOSE (la pasarela residencial desarrollada y el framework OSGi desarrollado en este proyecto), se desarrolló un reproductor de cursos, t-Maestro, capaz de personalizar el contenido y adaptar su comportamiento al perfil del usuario. Para su demostración práctica se desarrollaron varios cursos SCORM, entre los cuales destaca un curso sobre la historia de la Unión Europea, aprovechando el quincuagésimo aniversario de la creación de la Unión (Figura 80). En estos cursos, se utilizaron novedades como personajes virtuales para la audio-descripción de los contenidos, música de fondo personalizable según los gustos del usuario y contraste de los subtítulos adaptados a las necesidades visuales del alumno, para poner en práctica los trabajos sobre adaptación de los contenidos a las necesidades de cada alumno.

Los resultados finales del proyecto fueron presentados en el ITEA2 Symposium, celebrado los días 5 y 6 de octubre de 2006, en Paris, al que asistieron los ministros de Industria, Ciencia y Tecnología de varios países europeos, obteniendo el equipo del proyecto la medalla de oro[37] a la mejor demostración y exposición de los resultados.

Figura 79: Pasarela residencial desarrollada en el proyecto OSMOSE (2003-2005)

Figura 80: Escena de un curso de t-Maestro en el proyecto Passepartout

[37] http://www.itea2.org/paris_2006-10-19

Tras estos trabajos previos surgió el interés por la definición de aplicar reglas de adaptación, no al contenido de un curso, sino a un itinerario formativo, entendido como una secuencia de cursos, siendo el objetivo principal de esta tesis, investigar la viabilidad de la aplicación de la metodología de Optimización por Colonia de Hormigas (ACO) en el cálculo del itinerario alternativo más recomendable.

Esta Tesis presenta las características principales de la optimización por colonias de hormiga y un resumen de los principales algoritmos ACO existentes en la literatura, tanto para problemas de un único objetivo como multi-objetivos. El problema de las rutas de aprendizaje se ha planteado como un problema de un único objetivo: mejorar el rendimiento medio del grupo de alumnos, con la particularidad de que el rol de hormigas virtuales del resto de algoritmos ACO es desempeñado por los alumnos reales.

Se ha desarrollado una aplicación Java con el objeto de simular el comportamiento del algoritmo y analizar su comportamiento, que además ofrece un framework que facilita la implementación de nuevos algoritmos ACO mono-objetivos para la experimentación.

6.2 Conclusiones

En la literatura relacionada, ACO se aplica en todas las ocasiones a procesos en los que no existe interacción humana. Esta es la principal diferencia con los algoritmos ACO hasta el momento, dado que en todos los algoritmos hasta el momento, las hormigas virtuales de estos algoritmos son agentes software cuyas acciones se producen como resultado de la ejecución del algoritmo. En el algoritmo propuesto, en cambio, las acciones de estas hormigas virtuales son propiciadas por la actividad humana, a partir de la superación de los cursos asociados a cada nodo de un grafo que define los diferentes itinerarios de aprendizaje posibles. El enfoque de este trabajo defiende la adaptación del itinerario como fruto de un proceso colaborativo y social, como en el resto de técnicas ACO, pero fruto de las acciones de los seres humanos. Como demuestra el trabajo experimental de esta Tesis, es posible la aplicación de ACO con el objeto de obtener un sistema evolutivo, que adapte su respuesta de forma progresiva a los cambios del entorno, en este caso, a las calificaciones obtenidas por el conjunto de alumnos.

Además de otras aproximaciones a problemas similares utilizando optimización por colonia de hormigas [118]-[121], también hay propuestas que persiguen la adaptación del aprendizaje utilizando otras técnicas, destacando el trabajo

de Azough et al. [116] que propone el uso de algoritmos genéticos para adaptar los contenidos de un curso al perfil del alumno.

En todos ellos, uno de los problemas encontrados radica en la definición de una heurística que ayude a definir el concepto de distancia o visibilidad de cada nodo, η_{ij}, puesto que a diferencia de otros problemas en los que el concepto de distancia es fijo, en nuestro problema esta distancia está en constante evolución influenciada por los resultados obtenidos por los alumnos.

En esta Tesis se presenta el algoritmo *Ant System for Adaptive Learning Itineraries* (ASALI), como modificación del algoritmo *Ant System* (AS) para cubrir los requisitos del escenario particular de nuestro problema. El concepto de distancia entre los nodos se define mediante una función de ajuste apoyada en el peso pedagógico, definido a priori por el equipo pedagógico; la probabilidad de éxito y las evidencias de los resultados obtenidos en las evaluaciones de cada curso, reforzados en una cantidad variable por medio de las típicas feromonas de los algoritmos ACO. El algoritmo ASALI utiliza una estrategia elitista para el refuerzo del mejor camino, cuyo valor depende de la calificación media obtenida por los alumnos. Como novedad, se usan dos tipos de feromonas, off-line y on-line, influyendo las segundas en las decisiones de las hormigas del algoritmo ACO en cada iteración de la colonia. El concepto de iteración de una colonia ACO, se asocia a una promoción de un itinerario de aprendizaje, cuyo comienzo significa el inicio de las actividades (el inicio de los cursos) y que no termina hasta que todos los alumnos hayan finalizado todos los cursos programados, hayan abandonado o se haya alcanzado la fecha de finalización.

La viabilidad de la aplicación de ACO al problema de las rutas de aprendizaje depende en gran medida de la sensibilidad del algoritmo para variar el itinerario en función de los estímulos obtenidos del entorno en forma de feromonas. En esta Tesis se ha realizado un estudio de calibración que nos ha permitido comprobar que es posible variar la sensibilidad del algoritmo propuesto a través de la modificación de una serie de parámetros de calibración, lo que asegura la viabilidad de nuestra propuesta, al permitir a un equipo de expertos modificar el comportamiento del algoritmo para conseguir una respuesta más o menos rápida a los estímulos.

La experimentación mediante simulación nos lleva a deducir que es viable la aplicación de la metodología ACO para conseguir entornos de aprendizaje adaptativos, permitiendo adaptar el itinerario de aprendizaje (en cuanto a secuencias de cursos se refiere) a las necesidades del colectivo. Esta investigación no pretende demostrar si es

mejor o peor desde el punto de vista pedagógico esta solución, sino que ACO puede ser aplicado también para procesos de adaptación empleando. Es interesante el enfoque novedoso que presentamos, en el que se utilizan las acciones propias de seres humanos como las hormigas virtuales de ACO.

Los resultados obtenidos mediante la simulación nos indican que es viable la aplicación de esta técnica para conseguir la adaptación del camino formativo al rendimiento medio observado en el grupo de alumnos.

6.3 Aplicaciones

Entre las aplicaciones de la construcción de itinerarios adaptativos de aprendizaje y su gestión en plataformas de e-learning podemos destacar la personalización de itinerarios adaptados al estilo de aprendizaje de cada alumno la recomendación del siguiente curso.

6.3.1 Adaptación del itinerario al estilo de aprendizaje del alumno

En ambos casos es necesario poder describir el estilo de aprendizaje que se utiliza en cada curso. Sin embargo el estilo de aprendizaje de las personas dependen de múltiples factores: de cómo la información es seleccionada, organizada y procesada; de las diferentes habilidades relacionadas con el tratamiento de la información, del estado emocional, etc.

En función del sistema de representación de la información, podemos encontrar personas que destacan por mayor facilidad para procesar la información visual, otros que se adaptan mejor a sistemas de representación auditiva y otros aprenden mejor determinada información mediante las sensaciones experimentadas por su propio cuerpo, por los movimientos (por ejemplo a la hora de aprender un deporte). Este criterio clasifica a los alumnos en perfiles de aprendizaje visual, auditivo o kinestésico, que obviamente, no son perfiles mutuamente excluyentes.

Atendiendo a los mecanismos disponibles para organizar la información, los estilos de aprendizaje pueden clasificarse en función de los modos de pensamiento controlados por los dos hemisferios cerebrales. El hemisferio izquierdo es el encargado de reconocer grupos de letras formando palabras y éstas formando frases tanto en lo que

se refiere al habla, la escritura, la numeración, las matemáticas y la lógica, como a las facultades que intervienen en la transformación de un conjunto de informaciones en palabras, gestos y pensamientos. El hemisferio derecho, en cambio, está relacionado con facultades no verbales, sentimientos, emociones, y habilidades especiales como son el procesado de información visual y auditiva no relacionada con procesos verbales como pueden ser la música o el arte. Teniendo en cuenta la diferente especialización de los dos hemisferios cerebrales, existen modelos de estilos de aprendizaje que se basan en los diferentes modos de razonamiento de cada hemisferio [131]: analítico, secuencial, abstracto, lineal, realista, verbal, temporal, simbólico o cuantitativo para el hemisferio izquierdo; intuitivo, global, aleatorio, concreto, fantástico, no verbal, atemporal, literal o cualitativo para el hemisferio derecho.

En la época de la explosión de la sociedad de la información, de Internet, la televisión digital y de los dispositivos móviles, la cantidad de información que llega a los individuos es ingente, pero tan sólo una parte de la recibida es seleccionada y procesada. Ya hemos comentado que en función de cómo se selecciona la información podemos distinguir entre alumnos visuales, auditivos y kinestésicos. También que tras seleccionar la información de interés, cada individuo tiene que organizar esta información, y según cómo sea organizada podemos distinguir entre alumnos de hemisferio derecho o izquierdo.

Pero además toda esa información, una vez organizada ha de ser procesada. En este sentido no existe un único modelo o inventario de estilos de aprendizaje, sino que a lo largo de las últimas décadas han aparecido numerosas propuestas de modelos de aprendizaje en base a diferentes criterios. Entre estos diferentes modelos podemos destacar el Inventario de Estilos de Aprendizaje de Kolb [127] o KLSI (*Kolb's Learning Style Inventory*) [38] y el Modelo de Estilos de Aprendizaje de Felder-Silverman o FLSM (FSLSM, *Felder-Silverman Learning Style Model*) [129], que se centran en cómo se trabaja con la información para construir conocimiento, sobre todo a partir de la experiencia.

[38] La versión 4 se ofrece en exclusiva en la web de la consultora HayGroup: www.haygroup.com

En KLSI, Kolb desarrolla un modelo de aprendizaje basado en la formación del conocimiento a partir de la experiencia, siguiendo la secuencia cíclica: experiencia concreta, observación y reflexión, formación de conceptos abstractos y generalización, definición nuevas situaciones para estos conceptos y experimentación en estas nuevas situaciones (experimentación activa). Kolb añade que es necesario trabajar esas cuatro fases para que el aprendizaje sea efectivo, pero en la práctica la mayoría de individuos tiende a especializarse en una o dos de ellas, dada la relación de cada una de estas fases con los procesos de percepción (experiencia vs razonamiento abstracto) o procesamiento de la información (reflexión vs experimentación activa). El modelo de Kolb, contrapone la percepción y el procesamiento de la información para describir un modelo de cuatro cuadrantes (Figura 81) que clasifique a los alumnos en función de sus dos fases preferidas, definiendo así cuatro estilos de aprendizaje: *acomodador*, *divergente*, *asimilador* y *convergente*.

Figura 81: Cuadrante de Estilos de Aprendizaje de Kolb

De forma similar, el Modelo de Felder-Silverman clasifica las preferencias de aprendizaje según cuatro dimensiones: activa o reflexiva, sensitiva o intuitiva, visual o verbal, y secuencial o global. Tanto Kolb como Felder, han llevado a cabo estudios experimentales con alumnos reales [128] [130], para definir el porcentaje de alumnos que prefieren uno u otro estilo de aprendizaje según la especialización (arquitectura, ingeniería, agricultura, psicología, humanidades, etc.), nivel académico y cómo cambia esta preferencia con la edad, género y otras características de los alumnos. En muchas especialidades, no existe una gran diferencia entre los porcentajes de cada uno de los diferentes perfiles de alumnos, lo que indica que no es algo determinante y cualquier estilo de aprendizaje puede obtener éxito en esa especialidad. En cambio sí se observa en algunas una mayor concentración de alumnos de un determinado estilo de

aprendizaje, lo que pudiera llevar a pensar que existe una cierta relación entre dicho estilo de aprendizaje y la naturaleza de ciertos estudios, por ejemplo, eminentemente práctica y activa, como en Educación Física o abstracta como en Matemáticas (Figura 82).

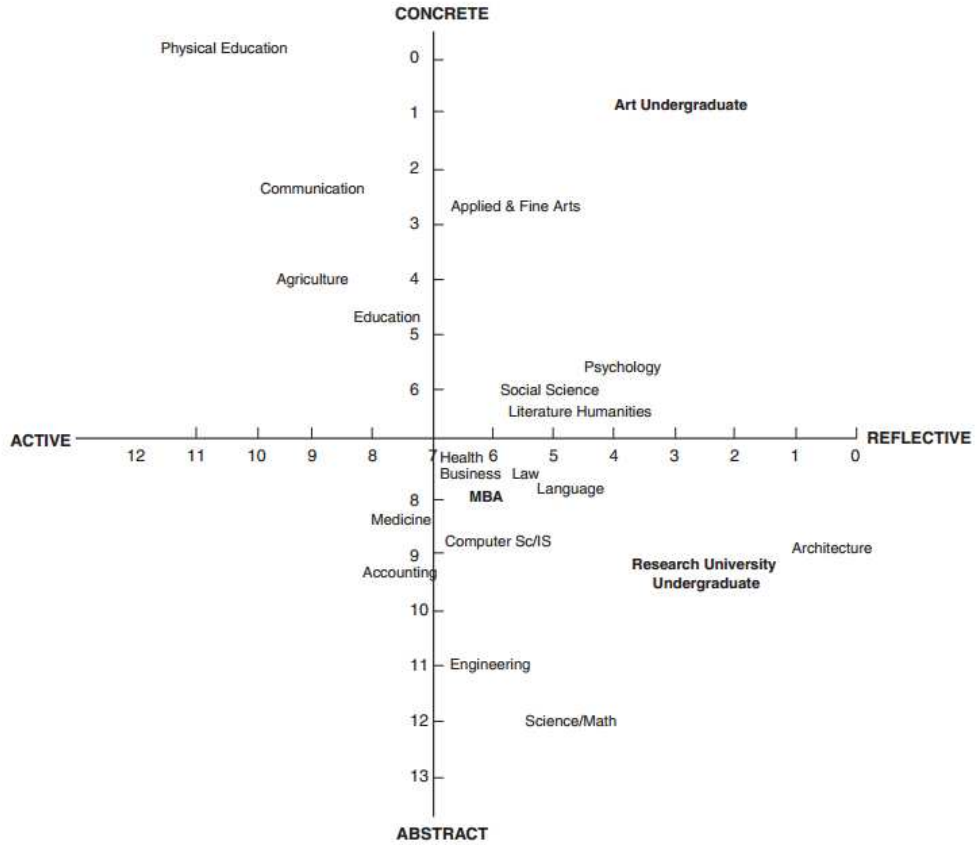


Figura 82: Clasificación de estudios universitarios según estilos de aprendizaje de Kolb

En aquellos estudios no tan marcados por un determinado estilo, como Informática, Medicina o la realización de un Máster en Administración de Empresas (MBA) puede resultar útil la personalización del itinerario de aprendizaje al estilo de aprendizaje que mejor se adapte al alumno. Para ello es necesario contar con unos meta-datos que describan el estilo de aprendizaje que mejor se ajusta a cada curso en función del enfoque de percepción y procesamiento de la información utilizado en el curso, es decir, si se fomenta en el contenido del curso el aprendizaje a partir de experiencias concretas o de conceptos abstractos, o si se persigue un aprendizaje a través de la observación del alumno o de la experimentación con nuevas situaciones. Por ejemplo, algunos alumnos aprenden más rápidamente con cursos que utilizan una estrategia de presentación de los conocimientos siguiendo un método inductivo, partiendo de ejemplos particulares para generalizar luego un principio absoluto (aunque esto sea

falso) que un método deductivo, que les confunde en el principio general haciéndoles difícil luego entender su aplicación en los casos particulares.

La aplicación de ASALI a la personalización del itinerario de aprendizaje del alumno implica la diferenciación de los cursos que se corresponden con el estilo de aprendizaje del alumno. La adaptación del algoritmo consiste en la aplicación del algoritmo ASALI a los cursos destino que coinciden con el estilo de aprendizaje del alumno. Esto implica la construcción de una estructura de datos (por ejemplo, una lista o un vector) en la que incluir estos cursos de forma que el cálculo probabilístico y la elección del siguiente curso sólo tenga en cuenta los cursos que coinciden con las preferencias del alumno. Los atributos para describir los estilos de aprendizaje más adecuados en cada curso pueden ser añadidos como atributos de cada nodo en el grafo de itinerarios de aprendizaje (Listado 39).

```
...
<key id= "learningType" for= "node">
     <desc>Learning Type</desc>
     <default>0</default>
  </key>
...
 <node id="Node1">
  <data key="y">226.0</data>
  <data key="x">271.0</data>
  <data key=”learningType”>1,3</data>
  ...
 </node>
 ...
</graph>
```

Listado 39: Atributo learningType para representar los estilos de aprendizaje de un nodo

En caso de no existir ningún curso que coincida con el estilo de aprendizaje del alumno, el algoritmo tomaría los existentes sin importar el estilo de aprendizaje de éstos. Esta estrategia implicaría la descripción tanto de los cursos, para los que habría que incluir uno o más atributos a cada nodo del grafo para indicar los estilos de aprendizaje adecuados y además para cada alumno, incluir su estilo de aprendizaje. De entre todos los posibles que se adapten al perfil del alumno, el algoritmo seleccionará el de mayor probabilidad de éxito (evaluando las feromonas depositadas en el arco que lleva a cada nodo) teniendo en cuenta el peso pedagógico de los mismos.

Esta estrategia, divide el grafo de itinerarios de aprendizaje en tantos itinerarios alternativos como estilos de aprendizaje existentes (Figura 83).

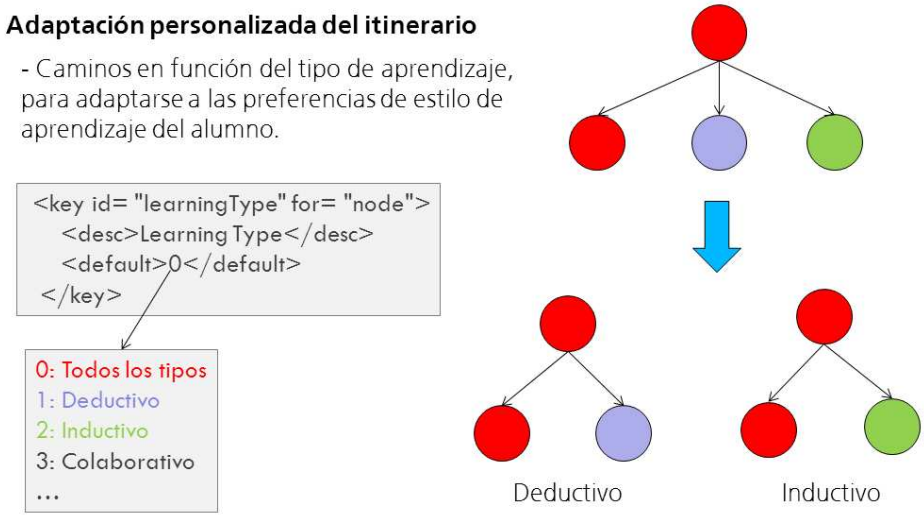


Figura 83: Representación de itinerarios alternativos en función del estilo de aprendizaje

Una opción más fácil de implementar, cosiste en la modificación del factor de idoneidad de la función de ajuste (39) de nuestra aproximación, para asignar el valor 1 si el estilo adecuado para el curso siguiente coincide con alguno de los preferidos por el alumno o el valor 0 (cero) en caso contrario. En cambio esta estrategia, no conduce siempre al alumno por un curso que coincida con su estilo de aprendizaje, puesto que si el valor de feromonas ϕ es suficientemente alto en arcos que llevan a cursos de otros estilos de aprendizaje, el valor de la función de ajuste para estos arcos sería mayor y por tanto con más probabilidades de ser elegidos.

Uno de los principales inconvenientes de la personalización del itinerario es la dificultad de conocer el estilo de aprendizaje de cada alumno. A diferencia de los metadatos que describen los estilos de aprendizaje más acordes con cada curso, en el caso de los alumnos es muy probable no disponer de información acerca de su estilo de aprendizaje preferente.

Determinar los estilos de aprendizaje más adecuados en cada área de conocimiento no es una tarea sencilla, debido a las diferentes características que determinan el estilo de aprendizaje. Generalmente, como ya se ha comentado anteriormente, un alumno no desarrolla únicamente un estilo (de los cuatro descritos por Kolb) sino que suele desarrollar dos, lo que no significa que si se presenta y organiza la información de acuerdo a los estilos restantes, el alumno no asimile los conocimientos de forma satisfactoria. En este sentido, existen numerosos instrumentos para averiguar el estilo de aprendizaje de cada alumno, generalmente empleando un test con una serie

de preguntas adaptadas al ámbito en el que se desarrolla el aprendizaje (académico, empresarial…) y que el alumno debe contestar de forma sincera. Un ejemplo de test, aplicado al contexto académico español, para calcular el estilo de aprendizaje de los alumnos es el CHAEA (Cuestionario Honey-Alonso sobre Estilos de Aprendizaje) [132] aunque existen numerosos[39] cuestionarios y encuestas aplicables a diferentes ámbitos [133]. El principal problema de la mayoría de estos instrumentos, es que tienen el riesgo de etiquetar a los alumnos en un estilo de aprendizaje determinado, sobre todo cuando el alumno presenta más de un estilo de preferencia fuertemente desarrollado [134]. La aplicación de un test de determinación del estilo de aprendizaje a cada alumno en el nodo de inicio del Grafo de Itinerarios de Aprendizaje, como primera actividad del curso, ayudaría a determinar los estilos de aprendizaje de cada alumno. Una alternativa, más compleja pero que ayudaría a minimizar el número de estilos de aprendizaje, consiste en la aplicación de técnicas de minería de datos [135], para el descubrimiento de los dos estilos de aprendizaje más adecuado para la mayoría de alumnos de una determinada área de conocimiento.

6.3.2 Recomendación del siguiente curso

En nuestro trabajo se ha estudiado la adaptación del itinerario tomando como base de decisión la calificación obtenida en los tests de evaluación de los cursos, verificando que el algoritmo ASALI tiene la capacidad de adaptar la ruta al itinerario de mayor éxito. Al momento de integrar ASALI en el LMS, puede decirse que en lugar de seleccionar el siguiente curso, se ofrezca al alumno las diferentes opciones posibles (los cursos posibles) en cada nodo de decisión, indicando el nivel de éxito alcanzado en cada uno de los cursos candidatos. Esta información está disponible en el modelo de datos, que almacena la cantidad de feromonas de cada arco y que podría normalizarse para ofrecer un indicador que sirva para recomendar las diferentes opciones. Además, también podría ofrecerse como información extra que ayude al alumno a tomar la decisión, la calificación media obtenida por otros alumnos en el curso destino, si bien, en esta podrían tenerse en cuenta alumnos que han llegado al curso destino desde otros cursos diferentes al de origen (siguiendo otros arcos). Si existe la información acerca del estilo de aprendizaje tanto en el perfil del alumno como en los metadatos del curso, podrían recomendarse además los cursos en función del estilo de aprendizaje.

[39] Puede consultarse una lista de estos instrumentos en la tesis de Daniela Melaré Vieira Barros [133].

6.4 Trabajo Futuro

En esta Tesis se ha analizado el problema de la adaptación de itinerarios de aprendizaje desde una perspectiva mono-objetivo. La complejidad del estudio se incrementa notablemente para problemas multi-objetivo, que se presenta como trabajo futuro de continuación de esta tesis. De esta forma resta por analizar la viabilidad de algoritmos ACO multi-objetivos en el escenario propuesto, teniendo presente la particularidad del doble rol jugado por los alumnos (como alumnos propiamente dichos y como hormigas virtuales del algoritmo ACO) y la optimización de dos o más objetivos. En este aspecto, la revisión del estado del arte que se ofrece en el capítulo 2 de esta Tesis puede resultar de gran utilidad para futuros trabajos en este sentido.

Como trabajo futuro aún falta completar la integración de un LMS con adaptación de itinerarios de aprendizaje, incorporando herramientas colaborativas que faciliten la creación de conocimiento mediante la colaboración de los propios alumnos y antiguos alumnos, conjugando así metodologías de aprendizaje conductista y construccionista, sacando provecho del paradigma de la Web 2.0.

En este sentido el autor dirige el proyecto de Universidad Corporativa en una importante empresa multinacional con sede en Sevilla. Este proyecto, recién comenzado en el momento de escribir estas líneas, es pionero en Europa e integrará un LMS comercial con una plataforma[40] de red social corporativa que facilitará la interacción y colaboración a antiguos alumnos de los diferentes programas de desarrollo de competencias de la compañía (programas de formación de jefes de proyecto, de formación en dirección de obras de ingeniería, de desarrollo de potenciales directivos, etc.) en busca de una mayor riqueza de la formación y una mejor gestión del conocimiento.

La integración de ASALI en un LMS puede ayudar a la adaptación del itinerario al estilo de aprendizaje de los alumnos, y constituye una de las aplicaciones más complejas y ambiciosas de este trabajo.

[40] Jive Social Business Software, http://www.jivesoftware.com

Curriculum Vitae

José Manuel Márquez Vázquez es Ingeniero en Informática (E.T.S. de Ingeniería Informática. Universidad de Sevilla, julio de 2002) y obtuvo el Diploma de Estudios Avanzados (DEA) del Programa de Doctorado en Lenguajes y Sistemas Informáticos en junio de 2007. Ese mismo año, recibió también el Certificado de Aptitud Pedagógica (Instituto de Ciencias de la Educación de Sevilla).

En el año 2003, fue galardonado con el Premio "Mejor Investigación Docente 2001/2002 de la Universidad de Sevilla" y con el Premio "Mejor Proyecto Fin de Carrera 2003" patrocinado por FIDETIA, por el proyecto "Eduka, una primera aproximación a un sistema de educación semipresencial".

Desde mayo de 2001 ha desarrollado su carrera profesional en Abengoa, comenzando como desarrollador de Sistemas de Adquisición y Control de Datos (SCADA) en Sainco, continuando como investigador en proyectos de I+D+i europeos en Telvent y por último, desde finales de 2008, como Jefe de Proyectos de desarrollo de software de la División de Aplicaciones de Negocio en Simosa IT.

Entre los meses de diciembre de 2002 y marzo de 2003, fue profesor del curso de Formación Profesional Ocupacional "Programador de aplicaciones de Redes Internet", impartido en el I.E.S. "Los Viveros" de Sevilla.

Ha colaborado de forma continuada con el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla desde el año 2002, siendo nombrado Asistente Honorario durante los cursos 2006/07, 2009/10 y 2010/11.

Cuenta con varias publicaciones en revistas y congresos internacionales, así como en capítulos de libros relacionados con las tecnologías de la información y la educación. A continuación mostramos únicamente las publicaciones relacionadas con esta tesis:

Artículos publicados en revistas

- Performance improvement using Adaptive Learning Itineraries. José Manuel Márquez Vázquez, Luis González Abril, Francisco Velasco Morente and Juan Antonio Ortega Ramírez. **Computational Intelligence**. Edited by: Ali Ghorbani, and Evangelos Milios. Factores de Impacto JCR: 5.378 (2009), 0.704 (2010). Aceptado y pendiente de publicación en 2012.
- Designing adaptive learning itineraries using features modelling and swarm intelligence. José Manuel Márquez Vázquez, Juan Antonio Ortega Ramírez, Luis González Abril and Francisco Velasco Morente. **Neural Computing and Applications**, 2011. Vol. 20, No. 5, 623-639, Factores de Impacto JCR: 0.812 (2009), 0.563 (2010). http://dx.doi.org/10.1007/s00521-011-0524-7.
- Aplicación de las nuevas tecnologías: docencia interactiva. J. A. Ortega, J. Torres, J. M. Márquez. **Revista de Enseñanza Universitaria**. Diciembre 2001, Número 18. Universidad de Sevilla. (Sevilla, 2001). ISSN: 1131-5245. http://institucional.us.es/revistas/universitaria/18/art_13.pdf

Artículos publicados en congresos

- Creating Adaptive Learning Paths Using Ant Colony Optimization and Bayesian Networks. José Manuel Márquez, Juan Antonio Ortega, Luis González Abril, Francisco Velasco. **2008 IEEE World Congress on Computational Intelligence** (WCCI2008). Hong Kong, China. 1-6 de junio de 2008. ISBN: 978-1-4244-1821-3. http://dx.doi.org/10.1109/IJCNN.2008.4634349
- Defining adaptive learning paths for competence-oriented learning. José Manuel Márquez, Juan Antonio Ortega, Luis González Abril, Francisco Velasco. In Proceedings of the **IADIS International Conference e-LEARNING 2008** (part of the IADIS Multi Conference on Computer Science and Information Systems 2008) Amsterdam, July 22-25, 2008. Volume I, pp 403-410. Edited by Miguel Baptista Nunes and Maggie McPherson. ISBN: 978-972-8924-58-4. http://www.iadisportal.org/digital-library/defining-adaptive-learning-paths-for-competence-oriented-learning
- Distributing OSGi services: The OSIRIS Domain Connector. José Manuel Márquez, Javier Álamo, Juan Antonio Ortega. **4th International Conference on Networked Computing and Advanced Information Management**, NCM2008, Volume I, pp. 341-346. Gyeongju, Corea del Sur, 2-4 de septiembre de 2008. Published by IEEE Computer Society, Order Number P3322. ISBN: 978-0-7695-3322-3. http://dx.doi.org/10.1109/NCM.2008.172

- Secure real-time integration of services in a OSGi distributed environment. José Manuel Márquez, Javier Jiménez, Isaac Agudo. **4th International Conference on Networked Computing and Advanced Information Management**, NCM2008, Volume I, pp. 631-636. Gyeongju, Corea del Sur, 2-4 de septiembre de 2008. Published by IEEE Computer Society, Order Number P3322. ISBN: 978-0-7695-3322-3. http://dx.doi.org/10.1109/NCM.2008.173

- Catálogo de aplicaciones de OSMOSE: Una experiencia real en el uso de servicios web en plataformas OSGi. José Manuel Márquez, Juan Antonio Ortega, Guillermo Hernández. Actas de las **I Jornadas Científico-Técnicas en Servicios Web, JSWEB'2005. I Congreso Español de Informática**, CEDI 2005. Editado por Thomson. (Granada, del 13 al 16 de septiembre de 2005). Pags. 91-98. ISBN: 84-9732-455-2. Presentación disponible en: http://www.w3c.es/Eventos/ServiciosWeb/presentaciones/cedi2005TELVENT.pdf

- Sistema avanzado de despliegue de aplicaciones en pasarelas residenciales. Miguel García Longarón, Fernando Usero Fuentes, José Manuel Márquez, Miguel Ángel Oltra Rodríguez, José Luis Ruiz Revuelta y Juan Carlos Dueñas López. **XV Jornadas Telecom I+D** (Madrid, del 22 al 24 de noviembre de 2005). ISBN: 84-689-3794-0

Capítulos en libros

- Calibración de un algoritmo ACO para el cálculo de itinerarios alternativos en sistemas de e-learning. José Manuel Márquez Vázquez, Luis González Abril, Francisco Velasco Morente, Juan Antonio Ortega Ramírez. **XIII Jornadas de Arca** (JARCA'11). Sistemas Cualitativos y sus Aplicaciones en Diagnosis, Robótica e Inteligencia Ambiental. ISBN: 978-84-615-5513-0. http://madeira.lsi.us.es/JARCA/Actas%20Jarca%202011.pdf

- Métodos de aprendizaje por refuerzo aplicados a un escenario de e-Learning. José Manuel Márquez Vázquez, Francisco Velasco Morente, Luis González Abril, Juan Antonio Ortega Ramirez, Cristóbal Chamizo Guerra. **XI Jornadas de Arca** (JARCA'09). Sistemas Cualitativos, Diagnosis, Robótica, Sistemas Domóticos y Computación Ubicua. Sevilla. España. Edición Digital @ Tres, S.L.L. Vol. 50. 2010. Pag. 69-75. ISBN: 978-84-613-71

- Comparativa y limitaciones de los sistemas de e-learning y m-learning. Juan Antonio Ortega Ramírez, José Manuel Márquez, Jesús Torres Valderrama, Juan Antonio Álvarez García, Alejandro Fernández y Manuel Cruz. **Experiencia de Innovación Universitaria (I).** Colección "Innovación y Desarrollo de la

Calidad de la Enseñanza Universitaria". Instituto de Ciencias de la Educación. Vicerrectorado de Docencia. Universidad de Sevilla. Vol. 1. 2007. Pág. 549-559. ISBN: 978-84-86849-51-1.

- Evolución del Aprendizaje Semipresencial: del e-Learning al M-Learning. Juan Antonio Ortega Ramírez, José Manuel Márquez, Jesús Torres Valderrama, Antonia Reina Quintero, María José Escalona Cuaresma. **La Innovación en la Enseñanza Superior (I)**. Sevilla, España. Instituto de Ciencias de la Educación. Vicerrectorado de Docencia. Universidad de Sevilla. Vol. 1. 2006. Pag. 563-576. ISBN: 84-86849-40-3.
- Aplicación de Sistemas Hipermedia al desarrollo de clases virtuales y semipresenciales. J. A. Ortega, J. Torres, J. M. Márquez. **Innovaciones Docentes en la Universidad de Sevilla. Curso 2001-2002**. Sevilla. Universidad de Sevilla. 2002. Pag. 229-241. ISBN: 84-86849-29-2
- Un Sistema de Aula Virtual que Combina Sistemas de Hipermedia con Aplicaciones de Videoconferencia. Juan A. Ortega Ramírez, José Manuel Márquez Várquez, Jesús Torres Valderrama, María José Escalona Cuaresma. **Innovaciones Docentes en la Universidad de Sevilla, Curso 2002-2003**: Áreas de Arte y Humanidades, Ciencias Exactas y Naturales, Ciencias de la Salud e Ingeniería y Tecnología. Universidad de Sevilla. Instituto de Ciencias de la Educación, Vicerrectorado de Calidad y Nuevas Tecnologías. Vol. 1. 2004. Pag. 299-312. ISBN: 84-86849-31-4

Proyectos Fin de Carrera dirigidos

Título	Análisis y Prueba de Concepto para la Adopción de SOA en un Entorno Empresarial
Autor	Jesús Burgers Oñate
Tutor	José Manuel Márquez Vázquez
Ponente	Dr. Alejandro Carballar Rincón
Titulación	Ingeniero en Telecomunicación
Universidad	Universidad de Sevilla
Año	2011
Resumen	Estudio del arte de SOA, análisis de viabilidad y prueba de concepto para la migración a una arquitectura SOA del sistema de información móvil de una multinacional, utilizando como elemento clave de la arquitectura un bus de servicios empresariales (ESB).
Calificación	Matrícula de Honor

Proyectos de Investigación en los que ha participado

Título	JULES VERNE
Código	ITEA 02002
Empresa	Telvent Interactiva, S.A.
Duración	2 años. (2003 y 2004)
Coordinador	Keith Baker {keith.baker@philips.com}
Financiación	Ministerio de Industria, Turismo y Comercio
Descripción	Proyecto de investigación en el que participan las mayores empresas de la industria de difusión digital interactiva para realizar pruebas e investigar las tecnologías disponibles para la creación de contenidos y capacidad de los futuros terminales y redes domésticas.
Entidades participantes	Philips, Philips Semiconductors, Thomson Multimedia, Telvent, Cybercultus, Cardinal Information Systems Ltd., Centre Henri Tudor, Institut National de Recherche en Informatique et en Automatique (INRIA) Loria, Institut National des Telecommunications.
Web	http://www.citi.tudor.lu/julesverne

Título	OSMOSE (Open Source Middleware for Open Systems in Europe)
Código	ITEA 02003
Empresa	Telvent Interactiva, S.A.
Duración	2 años. (Junio de 2003 a Junio de 2005)
Coordinador	Jesús Bermejo. {jesus.bermejo@telvent.abengoa.com}
Financiación	Ministerio de Industria, Turismo y Comercio
Descripción	Definir un conjunto de elementos hardware y software de código libre, que facilitase la comunicación entre dispositivos en el entorno residencial, así como que garantizase, en la medida de lo posible, la ejecución de aplicaciones multimedia y su integración con la TV digital.
Entidades participantes	Bull, France Télécom, Philips, Telefonica, Telvent, Thales, Bantry Technologies, VICORE and Whitestein Technologies, CharlesUniversity, École Polytechnique Fédérale de Lausanne (EPFL), Institut National de Recherche en Informatique et en Automatique (INRIA), Institut National des Télécommunications, Laboratoire d'Informatique Fondamentale de Lille, y Universidad Politécnica de Madrid.
Web	http://osmose.objectweb.org/

Título	FAMILIES (Madurez basada en hechos mediante lecciones aprendidas en la institucionalización y la exploración asociada a la ingeniería de familias de sistemas)
Código	ITEA ip02009
Empresa	Telvent Interactiva, S.A.
Duración	2 años. (Junio de 2003 a Junio de 2005)
Coordinador	Dr. Frank van der Linden {frank.van.der.linden@philips.com}
Financiación	Ministerio de Industria, Turismo y Comercio.
Descripción	Ingeniería del software aplicada a las familias de sistemas y productos.
Entidades participantes	TU Wien, Thales, Institut National de Recherche en Informatique et en Automatique (INRIA), Ivorium, CEA-List, Nokia, University of Helsinki, VTT, MetaCase Consulting, Siemens, Robert Bosh GmbH, Market Maker, University of Duisburg-Essen, IESE Fraunhofer, ICT Norway, SINTEF, DNV Software, EDB Telesciences, Ericsson Norway, SuperOffice, Visma Software, European Software Institute, Universidad Politécnica de Madrid, Telvent, Philips, Rijks Universiteit Groningen.
Web	http://www.esi.es/Families/

Título	Passepartout
Código	ITEA 04017
Empresa	Telvent Interactiva, S.A.
Duración	2 años. (Marzo de 2005 a Marzo de 2007)
Coordinador	Keith Baker {keith.baker@philips.com}
Financiación	Ministerio de Industria, Turismo y Comercio.
Descripción	Este proyecto se centra en la convergencia de sistemas y aplicaciones digitales en media-centers en el hogar. Se espera que de este proyecto surjan nuevas tecnologías que impulsen a las industrias del software europeas hacia la convergencia sobre terminales y red con el objetivo final de la "inteligencia ambiental". El proyecto persigue aunar los media-centers con las redes domésticas para proporcionar contenidos escalables desde la televisión de alta definición (HDTV) a definiciones inferiores de una forma adecuada.
Entidades participantes	ARTEMIS Institut National des Telecommunications, BCE, Centre Henri Tudor, CharToon, CWI-Amsterdam, Cybercultus, Electronics and Telecommunications Research Institute (ETRI), GRADIENT/LARES/IRUTIC, INRIA Loria, Philips Applied Technologies, Prewise, Stoneroos, Technische Universiteit Eindhoven, Telvent, Thomson R&D, Universidad Politécnica de Madrid, Universidad de Vigo, V2_, Institute for the Unstable Media.
Web	http://www.itea2.org/public/project_leaflets/PASSEPARTOUT_profile_oct-05.pdf

Título	OSIRIS
Código	ITEA 04040
Empresa	Telvent Interactiva, S.A.
Duración	3 años. (2006, 2007 y 2008)
Coordinador	Jesús Bermejo {jesus.bermejo@telvent.abengoa.com}
Financiación	501.942,70 € (total de ayudas concedidas en la convocatoria de 200584)
Descripción	Integración de tecnologías en tiempo de ejecución en una plataforma orientada a servicios, abierta y escalable, con despliegue, agregación y distribución de servicios a través de un bus de aplicaciones.
Entidades participantes	ASL, Charles University, CNR-ISTI, CognIT, Ericsson, Eteration, Gatespace, ICT-Norway, inAccess Network, Instituto de Telecomunicações, Italtel, Karde, Norwegian Computing Centre, Norwegian Tax Administration, Philips, Portugal Telecom, RadioNor, RedIRIS, SINTEF, SouJava, SuperOffice, Telefónica, Telvent, Thales, Universidad de Málaga, Universidad Politécnica de Madrid.
Web	http://www.itea2.org/public/project_leaflets/OSIRIS_profile_oct-05.pdf

Referencias

[1] Márquez, J.M.; J. A. Ortega; L. González Abril; F. Velasco and C. Angulo (2008). Plataforma Educativa Basada en una Arquitectura Abierta Orientada a Servicios. Innovación Educativa en las Titulaciones de Informática en la Universidad Española. Huelva. Universidad de Huelva, Servicio de Publicaciones. Vol. 300, pp. 91-105.

[2] Castells, M. (1997). La era de la información. Madrid. Alianza Editorial. 1997

[3] Messerschmitt, D. and C. Szyperski. (2003). Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press, 2003

[4] Weiss, J. (2005). Aligning relationships: Optimizing the value of strategic outsourcing. Global services report, IBM, 2005

[5] Márquez, J.M. (2007). Estado del arte del eLearning. Ideas para la definición de una plataforma universal. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla. March 2007.
http://www.lsi.us.es/docs/doctorado/memorias/Marquez,%20Jose%20M.pdf (recuperado en marzo de 2012)

[6] Weber, G. (1999). Adaptive learning systems in the World Wide Web. In: J. Kay (Ed.), Proceedings of the international conference on user modeling, UM99, Banff, Canada, June 20–24, 1999 (pp. 371–378). Wien, NewYork: Springer.

[7] Manouselis, N. and Sampson, D. (2003). Agent-based e-learning course recommendation: Matching learner characteristics with content attributes. International Journal of Computers and Applications (IJCA), 25(1).

[8] Melis, E.; E. Andres; J. Büdenberder; A. Frishauf; G. Goguadse; P. Libbrecht, M. Pollet and C. Ullrich. (2001). ActiveMath: A generic and adaptive webbased learning environment. International Journal of Artificial Intelligence in Education, 12(4), 385–407.

[9] Rey-López, M.; A. Fernández-Vilas; R. P. Díaz Redondo and J. Pazos-Arias. (2006). Providing SCORM with adaptivity. In Proceedings of the 15th International Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 981-982

[10] Huang, Y. M.; J. N. Chen; T. C. Huang; Y. L. Jeng and Y. H. Kuo (2007). Standardized course generation process using dynamic fuzzy Petri nets. Expert Systems with Applications, 34 (1).

[11] Chen, J. N.; Y. M. Huang and W. C. Chu (2005). Applying dynamic fuzzy Petri net to web learning system. Interactive Learning, 13(3), 159–178.

[12] Semet, Y.; E. Lutton and P. Collet (2003). Ant colony optimization for e-learning: Observing the emergence of pedagogic suggestions, in Proceedings of IEEE Swarm Intelligence Symposium, Indianapolis, IN, 2003, pp. 24–26

[13] Canales, A.; A. Peña; R. Peredo; H. Sossa and A. Gutiérrez (2007). Adaptive and intelligent web based education system: Towards an integral architecture and framework. Expert Systems with Applications, 33, 1076–1089.

[14] Power G.; H. C. Davis; R. I. Cristea; C. Stewart and H. Ashman. (2005). Goal Oriented Personalisation with SCORM. In Proceedings of 5th IEEE International Conference on Advanced Learning Technologies, ICALT, pp. 5-8

[15] Dorigo, M. (1992). Optimization, Learning and Natural Algorithms, *PhD thesis,* Politecnico di Milano, Italie, 1992.

[16] Zhu, F.; H.H.S. Ip; A. W. P. Fok and J. Cao (2007). PeRES: A Personalized Recommendation Education System Based on Multi-agents & SCORM. In Proceedings of 6th International Conference on Web Based Learning – ICWL 2007 Edinburgh, UK, August 15-17, 2007. Lecture Notes in Computer Science, 2008, Volume 4823/2008, pp. 31-42

[17] Huang, M-J.; H-S. Huang; M-Y. Chen (2007). Constructing a personalized e-learning system based on genetic algorithm and case-based reasoning approach. Expert Systems with Applications. Volume 33, Issue 3, October 2007, pp. 551-564. doi:10.1016/j.eswa.2006.05.019

[18] Gutiérrez, S. (2007). Secuenciamiento de actividades educativas orientado a la reutilización y la auto-organización en tutoría inteligente. Tesis Doctoral. Universidad Carlos III de Madrid.

[19] Coates, J. (2006). Generational Learning Styles. LERN Books, Learning Resources Network. ISBN 1-57722-032-3.

[20] Griffiths, D.; J. Blat; T. Navarrete; J. L. Santos; P. García and J. Pujol. (2006). PlanetDR, a scalable architecture for federated repositories supporting IMS Learning Design. In Proceedings of International Workshop in Learning Networks for Lifelong Competence Development, TENCompetence Conference. March 30th-31st, Sofia, Bulgaria

[21] González-Ferrer, A.; L. Castillo; J. Fdez-Olivares and L. Morales (2008). Towards the use of XPDL as planning and scheduling modeling tool: the workflow patterns approach. In Proceedings of 11th Ibero-American Conference on AI, Lisbon, Portugal, October 14-17, 2008. Advances in Artificial Intelligence.

Lecture Notes in Computer Science, Vol. 5290. Geffner, H.; Prada, R.; Machado Alexandre, I.; David, N. (Eds.) Springer Verlag.

[22] Mariño, O.; R. Casallas; J. Villalobos; D. Correal and J. Contamines (2007). Bridging the gap between e-learning modeling and delivery through the transformation of learnflows into workflows. E-learning networked environments and architectures. A knowledge processing perspective. Samuel Pierre (Ed.). Springer Verlag, London, 2007.

[23] Márquez, J. M.; C. Cetina, F. Velasco; L. González- Abril and J. A. Ortega (2008) Modelado de características para itinerarios formativos adaptativos**.** X Jornadas de ARCA. Sistemas Cualitativos y Diagnosis, Robótica, Sistemas Domóticos y Computación Ubícua (JARCA). Tenerife, Spain. 2008.

[24] Márquez, J. M.; J. Álamo and J. A. Ortega (2008). Distributing OSGi services: The Osiris Domain Connector. In Proceedings of Fourth International Conference on Networked Computing and Advanced Information Management, NCM'08. Gyeongju, Korea Rep. 2-4 Sept. 2008. Vol. 1, pp 341-346.

[25] Márquez, J. M.; J. Jiménez and I. Agudo (2008). Secure Real-Time Integration of Services in a OSGi Distributed Environment. In Proceedings of Fourth International Conference on Networked Computing and Advanced Information Management, NCM'08. Gyeongju, Korea Rep. 2-4 Sept. 2008. Vol. 1, pp. 631-635.

[26] Márquez, J. M.; J. A. Ortega; L. González Abril and F. Velasco (2008). Creating adaptive learning paths using Ant Colony Optimization and Bayesian Networks. In Proceedings of International Joint Conference on Neural Networks, IJCNN'08. Hong Kong, June 1-8, 2008, pp. 3834-3839.

[27] Dorigo, M. and Gambardella, L.M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1997. 1(1): p. 53-66.

[28] Stützle, T. and M. Dorigo (1999). ACO algorithms for the quadratic assignment problem. New Ideas in Optimization. 1999, McGraw-Hill.

[29] van der Zwaan, S. and C. Marques (1999). Ant Colony Optimisation for Job Shop Scheduling. Proceedings of the Third Workshop on Genetic Algorithms and Artificial Life (GAAL 99), 1999.

[30] Schoonderwoerd, R.; O. Holland; J. Bruten and L. Rothkrantz (1996). Ant-based load balancing in telecommunications networks. Adaptative Behavior, 1996. 5: 2: p. 169-207.

[31] Costa, D. and A. Hertz (1997). Ant can colour graphs. Journal of the Operational Research Society, 1997. 48: p. 295-305.

[32] Parpinelli, R.S.; H.S. Lopes and A.A. Freitas (2002). Data mining with an Ant Colony Optimization algorithm. IEEE Transaction on Evolutionary Computation, 2002. 6: 4: p. 321-332.

[33] Casillas, J.; O. Cordón and F. Herrera (2000). Learning fuzzy rules using ant colony optimization algorithms. International Workshops on Ant Algorithms. Université Libre de Bruxelles, Belgium, 2000: p. 13-21.

[34] Campos, L.M.d.; J.A. Gámez and J.M. Puerta (2002). Learning bayesian networks by ant colony optimization: searching in two different spaces. Mathware & Soft Computing, 2002. 9: 2-3: p. 251-268.

[35] Deneubourg, J.-L.; S. Aron; S. Goss and J.-M. Pasteels (1990). The self-organizing exploratory pattern of the argentine ant. Journal of Insect Behavior, 3:159-168, 1990.

[36] Grassé. P. P. (1959). La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. Insectes Sociaux, 6:41-81, 1959.

[37] Dorigo, M.; V. Maniezzo and A. Colorni. (1991) Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.

[38] Vallés d., G. (2009). Selección del alimento en la hormiga argentina, Linepithema humile (Mayr, 1868) (Hymenoptera, Formicidae). Anales Universitarios de Etología, 3:13-17.

[39] Dorigo, M.; V. Maniezzo and A. Colorni (1996). The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41, 1996.

[40] Gambardella, L. M. and M. Dorigo (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In Proceedings of the Twelfth International Conference on Machine Learning, ML-95, pages 252-260. Palo Alto, CA: Morgan Kaufmann, 1995.

[41] Stützle, T. and H. Hoos (1997). The MAX-MIN ant system and local search for the traveling salesman problem. In T. Baeck, Z. Michalewicz, and X. Yao, editors, Proceedings of IEEE-ICEC-EPS'97, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference, pages 309-314. IEEE Press, 1997.

[42] Bullnheimer, B.; R. F. Hartl and C. Strauss (1997). A new rank-based version of the ant system: a computational study. Technical Report POM-03/97, Institute of Management Science, University of Vienna, 1997. Accepted for publication in the Central European Journal for Operations Research and Economics.

[43] Maniezzo, V. and A. Colorni (1998). The ant system applied to the quadratic assignment problem. IEEE Trans. Knowledge and Data Engineering, 1998, in press.

[44] Stützle, T. and H. Hoos (1998). MAX-MIN Ant system and local search for combinatorial optimization problems. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pages 137-154. Kluwer, Boston, 1998.

[45] Gambardella, L. M.; E. D. Taillard and M. Dorigo (1999). Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society (1999) 50, pp. 167-176. doi:10.1057/palgrave.jors.2600676

[46] Colorni, A.; M. Dorigo; V. Maniezzo and M. Trubian (1994). Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL), 34:39-53, 1994.

[47] Bullnheimer, B.; R. F. Hartl and C. Strauss (1998). Applying the ant system to the vehicle routing problem. In I. H. Osman S. Voβ, S. Martello and C. Roucairol, editors, Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pages 109-120. Kluwer Academics, 1998.

[48] Gambardella, L. M.; E. Taillard and G. Agazzi (1999). Ant colonies for vehicle routing problems. New Ideas in Optimization. McGraw-Hill, 1999.

[49] Gambardella, L. M. and M. Dorigo (1997). HAS-SOP: An hybrid ant system for the sequential ordering problem. Technical Report 11-97, IDSIA, Lugano, CH, 1997.

[50] Michel, R. and M. Middendorf (1998). An island model based ant system with lookahead for the shortest supersequence problem. In Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, pages 692-701. Springer-Verlag, 1998.

[51] White, T.; B. Pagurek and F. Oppacher (1998). Connection management using adaptive mobile agents. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), pp. 802-809. CSREA Press, 1998.

[52] Di Caro, G. and M. Dorigo (1998). Extending AntNet for best-effort Quality-of-Service routing. Unpublished presentation at ANTS'98 - From Ant Colonies to Artificial Ants: First International Workshop on Ant Colony Optimization. http://iridia.ulb.ac.be/ants98/ants98.html. October 15-16 1998.

[53] Bonabeau, E.; F. Henaux; S. Guérin; D. Snyers; P. Kuntz and G. Théraulaz (1998). Routing in telecommunication networks with "Smart" ant-like agents telecommunication applications. In Proceedings of IATA'98, Second Int.

Workshop on Intelligent Agents for Telecommunication Applications. Lectures Notes in AI vol. 1437, Springer-Verlag, 1998.

[54] Di Caro, G. and M. Dorigo (1998). Two ant colony algorithms for best-effort routing in datagram networks. In Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98), pages 541-546. IASTED/ACTA Press, 1998.

[55] Subramanian, D.; P. Druschel and J. Chen (1997). Ants and reinforcement learning: A case study in routing in dynamic networks. In Proceedings of IJCAI-97, International Joint Conference on Artifcial Intelligence, pages 832-838. Morgan Kaufmann, 1997.

[56] Heusse, M.; S. Guérin; D. Snyers and P. Kuntz (1998). Adaptive agent-driven routing and load balancing in communication networks. Advances in Complex Systems, 1, pp. 237-254, 1998.

[57] van der Put; R. and L. Rothkrantz (1999). Routing in packet switched networks using agents. Simulation Practice and Theory, 1999, in press.

[58] Mariano, C. and E. Morales (1999). A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks. Technical Report HC-9904, Instituto Mexicano de Tecnología del Agua, Mexico, June, 1999.

[59] Iredi, S.; D. Merkle and M. Middendorf (2001). Bi-Criterion Optimization with Multi Colony Ant Algorithms. In Proceedings of the First International Conference on Evolutionary Multi-criterion Optimization (EMO'01), Lecture Notes in Computer Science 1993, 359-372. 2001.

[60] Doerner, K.; W. Gutjahr; R. Hartl; C. Strauss and C. Stummer (2002). "Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection". Proceedings of the 4th. Metaheuristics International Conference. Porto, 243-248. 2002.

[61] Barán, B. and M. Schaerer (2003). A multiobjective Ant Colony System for Vehicle Routing Problems with Time Window*s"*. Proc. Twenty first IASTED International Conference on Applied Informatics, pg. 97-102. Insbruck, Austria. 2003.

[62] Pinto, D. and B. Barán (2005). Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimization approach. In Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking LANC'05, pp. 11-19. Cali, Colombia. 2005.

[63] Doerner, K.; R. Hartl and M. Reimann (2003). Are COMPETants more competent for problem solving? – the case of a multiple objective transportation problem. Central European Journal of Operations Research, 11:2, 115-141. 2003.

[64] Gardel, P.; H. Estigarribia; U. Fernández and B. Barán (2005). Aplicación del Ómicron ACO al problema de compensación de potencia reactiva en un contexto multiobjetivo. Congreso Argentino de Ciencias de la Computación - CACIC'2005. Concordia – Argentina. 2005.

[65] Paciello, J.; H. Martínez and C. Lezcano (2006). Algoritmos de Optimización multi-objetivos basados en colonias de hormigas. XXXII Conferencia Latinoamericana de Informática – CLEI'2006. Santiago de Chile

[66] Lin, S (1965). Computer solutions of the traveling salesman problem. Bell Systems Journal, 44:2245-2269, 1965

[67] Koopmans T. C. and M. J. Beckmann (1957). Assignment problems and the location of economic activities, Econometrica 25, 1957, pp. 53–76

[68] Dantzig, G. B. and J. H. Ramser (1959). The Truck Dispatching Problem. Management Science. Vol. 6, No. 1, October 1959, pp. 80-91.

[69] Gambardella, L. M.; É. Taillard and G. Agazzi (1999). MACS-VRPTW: A Multiple Ant Colony System for vehicle routing problem with time windows. In D. Corne, M. Dorigo and F. Glover, editors. New Ideas in Optimization. McGraw-Hill, London, UK, pp. 63-76.

[70] Escudero, L.F. (1988). An inexact algorithm for the sequential ordering problem. European Journal of Operational Research, Volume 37, Issue 2, November 1988, pp. 236-249.

[71] Ascheuer, N.; L. F. Escudero; M. Grotschel and M. Stoer (1993). A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). SIAM Journal on Optimization 3, pp. 25–42.

[72] Balas, E.; M. Fischetti and W.R. Pulleyblank (1995). The precedence-constrained asymmetric traveling salesman polytope. Mathematical Programming 65, pp. 241–265.

[73] Leighton, F. (1979) A graph coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards, 84, pp. 489-505.

[74] Brélaz, D. (1979). New Methods to color the vertices of a graph. Communications of the ACM, 22(4), pp. 251-256.

[75] Branke, J.; M. Middendorf and F. Schneider (1998). Improved heuristics and a genetic algorithm for finding short supersequences. OR-Spektrum, Vol. 20, pp. 39-46.

[76] Stützle, T. and M. Dorigo (2002). A short convergence proof for a class of Ant Colony Optimization algorithms. IEEE Transactions on Evolutionary Computation, Vol. 6, No. 4, August 2002.

[77] IEEE Standard for Learning Technology - Data Model for Reusable Competency Definitions. Versión 1. 29th August 2008. Official Standard (IEEE Std 1484.20.1™-2007). Editors: Claude Ostyn and Scott Lewis.

[78] IEEE. Proposed Draft Standard for Learning Technology - Simple Reusable Competency Map. Revision 4. 22 February 2006. Editor: Claude Ostyn.

[79] Márquez, J. M.; L. González Abril; F. Velasco and J. A. Ortega (2011). Designing adaptive learning itineraries using feature modelling and swarm intelligence. Neural Computing & Applications. Feb. 2011. doi:10.1007/s00521-011-0524-7

[80] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik 1: 269–271.

[81] Valigiani, G.; R. Biojout; Y. Jamont; E. Lutton; C. Bourgeois-Republique and P. Collet (2005). Experimenting with a real-size man-hill to optimize pedagogical paths. In Proceedings of the 2005 ACM symposium on Applied computing, SAC'05. ACM, New York.

[82] Márquez, J. M.; L. González Abril; F. Velasco and J. A. Ortega (2012). Performance improvement using adaptive learning itineraries. Computational Intelligence Journal. Wiley-Blackwell. Edited by Ali Ghorbani and Evangelos Milios. [Aceptado y pendiente de publicación]

[83] Nkambou, R and J. P. M. Tchetagni (2004) Diagnosing Student Errors in E-Learning Environment using MPE Theory. In Proceedings of Web-based Education – 2004. Innsbruck, Austria. 16-18 Februray 2004.

[84] Gamboa, H. and A. Fred (2002). Designing intelligent tutoring systems: a bayesian approach. In Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS 2001). J. Filipe et al. (Eds). Enterprise Information Systems III, 146-152.

[85] García, P.; A. Amandi; S. Schiaffino and M. Campo. (2005). Using Bayesian networks to detect students' learning styles in a Web-based education system. In Proceedings of ASAI 2005, Argentine Symposium on Artificial Intelligence, ISSN 1666 1079 – August 2005 - Rosario, Argentina. pp. 115 – 126.

[86] Gamma, E.; R. Helm; R. Johnson and J. Vlissides (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN 0-201-63361-2.

[87] Tapscott, D. (2008). Grown Up Digital: How the Net Generation is Changing Your World. McGraw-Hill, 2008, ISBN 978-0-07-150863-6, pp.15-16

[88] Dalziel, J. (2004). Open standards versus open source in e-learning. EDUCAUSE Quarterly, 4. 2004. http://www.educause.edu/ir/library/pdf/EQM0340.pdf (recuperado en marzo de 2012)

[89] Liu, X.; A. El Saddik and N. D. Georganas, (2003). An implementable architecture of an e-learning system. CCECE 2003 – CCGEI 2003, Montreal, 2003

[90] Sun Microsystems (2003). E-Learning Framework. Tehcnical White Paper. http://www.sun.com/products-n-solutions/edu/whitepapers/pdf/framework.pdf

[91] Leal Musa, D. and J. Palazzo Moreira de Oliveira, (2004). Sharing Learner Information through a Web Services-based Learning Architecture. International Workshop on Web Information Systems Modeling, WISM 2004.

[92] Vidal, J. C.; A. Novegil; M. Lama and E. Sánchez (2008). Service Oriented Architecture to Manage Units of Learning based on the IMS LD specification. In Proceedings of European University Information Systems, EUNIS 2008. 24th-27th of June. Århus, Denmark.

[93] Dueñas, J. C.; J. L. Ruiz; J. Bermejo; J. A. Alonso; C. Acuña and C. Díaz. (2004) Plataformas abiertas para la provisión de servicios, XIV JORNADAS TELECOM I+D. Madrid 23-25 Noviembre 2004.

[94] Oasis. Universal Description, Discovery and Integration v3.0.2 UDDI Specificaction (2005). http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm (recuperado en marzo de 2012)

[95] Vossen, G. and P. Westerkamp (2008). Why service-orientation could make e-learning standards obsolete. Int. J. Technology Enhanced Learning, Vol. 1, Nos. 1/2, pp.85–97.

[96] Vossen, G. and P. Westerkamp (2006). Towards the Next Generation of E-Learning Standards: SCORM for Service-Oriented Environments. ICALT, pp.1031-1035, Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06), 2006.

[97] Hu, X.-M.; J. Zhang; H. Shu-Hung Chung; O. Liu and J. Xiao (2009). An Intelligent Testing System Embedded with an Ant-Colony-Optimization-Based Test Composition Method. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, v.39, n.6, p. 659-669, November 2009. doi:10.1109/TSMCC.2009.2021952

[98] Brusilovsky, P. (1996) Methods and Techniques of Adaptive Hypermedia. In User Modeling and User-Adapted Interaction, 6(2-3), pages 87–129, 1996.

[99] Boyle, C. D. B. and A. O. Encarnación (1994). Metadoc: An Adaptive Hypertext Reading System. In User Model. User-Adapt. Interact., 4(1), pages 1–19, 1994.

[100] Encarnacão, L. M. (1995). Adaptivity in graphical user interfaces: An experimental framework. In Computers & Graphics 19, (6), pages 873–884, 1995.

[101] Henze, N. (2000). Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources. PhD Thesis, University of Hannover, 2000.

[102] Fielding, R. (2000) Architectural Styles and the Design of Network-based Software Architectures, Chapter (3): Representational State Transfer (REST). Doctoral thesis (PhD), University Of California.

[103] Erl, T. (2004). Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. 2004. Prentice Hall.

[104] Milner, R.; J. Parrow and D. Walker (1992). A calculus of mobile processes. Information and Computation 100 (100): 1–40. doi:10.1016/0890-5401(92)90008-4.

[105] Bahrami, K.; M. Abedi and B. Daemi (2007). A Web Service based portal framework for distance learning on Power Line Network. Sixth International Conference on Information, Communication and Signal Processing. ICICS 2007.

[106] Sallán J. M. (2006). Caracterización de los grupos informales de transmisión de conocimiento mediante el análisis de redes sociales. Intangible Capital. Marzo 2006, vol. 2, núm. 1, p. 21-36.

[107] Gaete, J. M. and J. I. Vásquez (2008). Conocimiento y estructura en la investigación académica: una aproximación desde el análisis de redes sociales. Redes: Revista hipana para el análisis de redes sociales, Nº 14, 2008.

[108] Navaro, L. I.; M. M. Such; D. M. Martin and P. P. Peco (2006). Architecture Oriented towards the management of Learning Objects Repositories (LOR@). ICALT, pp. 255-256. Sixth International Conference on Advanced Learning Technologies (ICALT'06), 2006.

[109] Xu, Z.; Z. Yin and A. El Saddik (2003). A Web Services oriented framework for dynamic e-learning systems. In Proceedings of Canadian Conference on Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Vol. 2, pp. 943-946.

[110] Chu, C-P.; C-P. Chang; C-W. Yeh and Y-F. Yeh (2004). A Web-Service Oriented Framework for building SCORM Compatible Learning Management Systems. ITCC, vol. 1, pp.156, International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 1, 2004.

[111] Dodero, J. M. and E. Ghiglione (2008). ReST-Based Web Access to Learning Design Services. IEEE Transactions on Learning Technologies, vol. 1, no. 3, pp. 190-195.

[112] Fontela, J.; M. Caeiro and M. Llamas (2009). Una Arquitectura SOA para sistemas de e-learning a través de la integración de Web Services. V Congreso Iberoamericano de Telemática. CITA 2009.

[113] Rojas, M. and J. Montilva (2011). Una arquitectura de software para la integración de objetos de aprendizaje basada en servicios web. Ninth Latin American and Caribbean Conference (LACCEI'2011).

[114] Canabal, M. and A. Sarasa (2007). Agrega, Plataforma de Objetos Digitales Educativos. IV Simposio Pluridisciplinar sobre Diseño, Evaluación y Desarrollo de Contenidos Educativos Reutilizables. SPDECE'2007, Bilbao. Available: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-318/ (recuperado en marzo de 2012)

[115] O'Reilly, T. (2005). What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Retrieved Sept. 27, 2011, from http://oreilly.com/web2/archive/what-is-web-20.html (recuperado en marzo de 2012)

[116] Sarasa, A. and M. Canabal (2011). Agrega: Un proyecto de software libre Web 2.0 de la Administración Pública. Retrieved Sept. 27, 2011, from http://www.proyectoagrega.es/client/documentoLocal/Agrega%20Software%20Libre.pdf (recuperado en marzo de 2012)

[117] Azough, S.; M. Bellafkih and E. H. Bouyakhf (2010). Adaptive e-learning using Genetic Algorithms. International Journal of Computer Science and Network Security, IJCSNS, Vol. 10, No. 7, July 2010.

[118] Gutiérrez, S.; A. Pardo and C. Delgado Kloos (2006). Finding a learning path: toward a swarm intelligence approach. In Proceedings of International Conference on Web-based Education 2006, pp. 94-99. Puerto Vallarta, México.

[119] Van den Berg, B.; R. Van Es; C. Tattersall; J. Janssen; J. Manderveld; F. Brouns; H. Kurvers and R. Koper (2005). Swarm-based sequencing recommendations in e-learning. In Proceedings of International Conference on Intelligence Systems Design and Applications 2005, pp. 488-493, Wroclaw, Poland.

[120] Janssen, J.; C. Tattersall; W. Waterink; B. Van den Berg; R. Van Es; C. Bolman and R. Koper (2007). Self-organizing navigational support in lifelong learning: How predecessors can lead the way. Computers & Education, 49, pp. 781-793.

[121] Wang, T-I.; K-T. Wang and Y-M. Huang (2008). Using a style-based ant colony system for adaptive learning. Expert Systems with Applications, 34(4), 2449-2464.

[122] Otero, F. E. B.; A. A. Freitas, and C. Johnson (2008). cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes. In Proceedings of the 6th International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS 2008), Lecture Notes in Computer Science 5217, pp. 48-59. Springer-Verlag, 2008.

[123] Gini, C. (1912). Variabilità e Mutabilità. Studi Economico-Giuridici dell'Univ. Di Cagliari, 3, part 2, pp.1-158.

[124] González, L.; F. Velasco; J. M. Gavilán and L. M. Sánchez-Reyes (2010). The Similarity between the Square of the Coefficient of Variation and the Gini Index

of a General Random Variable. Revista de Métodos Cuantitativos para la Economía y la Empresa (10), pp. 5-18, Diciembre de 2010.

[125] Brown, M. (1994). Using Gini-style indices to evaluate the spatial patterns of health practitioners: theoretical considerations and an application based on Alberta data. Soc. Sci. Med. Vol. 38, No. 9. pp. 1243-1256. 1994

[126] Hölldobler, B and E. O. Wilson (1990). The Ants. Harvard University Press. Premio Pulitzer 1991.

[127] Kolb, D. A. (1971). Individual learning styles and the learning process. Working Paper #535-71, Sloan School of Management, Massachusetts Institute of Technology.

[128] Kolb, D. A. and A. Y. Kolb (2005). The Kolb Learning Style Inventory – Version 3.1. 2005 Technical Specifications. http://www.whitewater-rescue.com/support/pagepics/lsitechmanual.pdf (recuperado en marzo de 2012)

[129] Felder, R. M. and L. K. Silverman (1988). Learning and Teaching Styles in Engineering Education. Engineering Education, Vol. 78, No. 7, pp. 674–681. http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/LS-1988.pdf (recuperado en marzo de 2012).

[130] Felder, R. M. and J. Spurlin (2005). Applications, Reliability and Validity of the Index of Learning Styles. International Journal on Engineering Education, Vol. 21, No. 1, pp. 103-112.

[131] Herman, N. (1996). The whole brain business book. Ed. McGraw-Hill. México.

[132] Alonso, C., D. J. Gallego and P. Honey (1999). Los estilos de aprendizaje: procedimientos de diagnóstico y mejora. Ediciones Mensajero, Bilbao, pp. 104-116.

[133] Melaré, D. (2011). Estilos de aprendizaje y medios didácticos en contextos virtuales. Tesis Doctoral. UNED. Madrid, 2011. http://e-spacio.uned.es/fez/eserv.php?pid=tesisuned:Educacion-Dmelare&dsID=Documento.pdf (recuperado en marzo de 2012)

[134] García, J. L. (2006). Tecnologías de la Información y Comunicación en la Formación del Profesorado. Tesis Doctoral. UNED. Madrid, 20016.

[135] Durán, E. and R. Costaguta (2007). Minería de datos para descubrir estilos de aprendizaje. Revista Iberoamericana de Educación. Nº. 42/2. 10-03-2007. Editado por la Organización de Estados Iberoamericanos para la Educación, la Ciencia y la Cultura.

Anexo

En este anexo se detallan los contenidos del disco compacto que acompaña a esta tesis. La estructura de carpetas del cd es la que se muestra en la Figura 84 se describe a continuación.

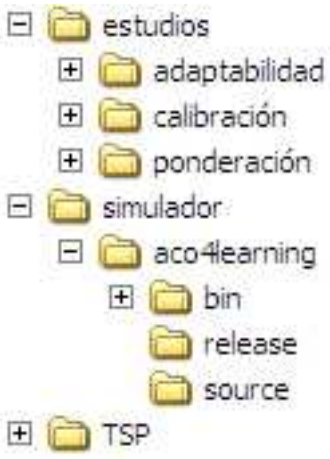


Figura 84: Jerarquía de carpetas del CD

Estudios

En la carpeta estudios, se incluyen los estudios sobre la estrategia de ponderación (carpeta ponderación), sobre la calibración del simulador (carpeta calibración) y sobre la adaptabilidad del algoritmo (carpeta adaptabilidad). Cada una de estas carpetas contiene un conjunto de hojas de cálculo, en formato Microsoft Excel 2010, que describimos a continuación. Siguiendo el orden en el que aparecen en la Tesis, se describe a continuación el contenido de cada una.

Ponderación

En esta carpeta se encuentra una hoja de cálculo con la comparativa (fichero `comparativa.xslx`) de las estrategias de equiponderación, cuyo estudio se incluye en la carpeta `equiponderación`; y baremación, en la carpeta `baremación`. En la carpeta de cada estrategia se incluye el grafo utilizado en los experimentos (`grafo_equiponderado.xml` y `grafo_baremado.xml`) y una hoja de cálculo que recoge los resultados de la comparativa realizada con diferentes colonias de hormigas: homogéneas (compuestas por todas las hormigas de un mismo perfil de rendimiento:

carpetas `Todos_Low`, `Todos_Medium` o `Todos_High` respectivamente) y colonias heterogéneas (compuestas por una población de hormigas de diferentes perfiles, en una proporción del 30%-60% de hormigas High, Medium y Low respectivamente y cuyos tests se incluyen en la carpeta `30-60-10`). Los resultados de cada test se muestran en ficheros en formato CSV (texto separados por el delimitador '`;`'), cuyo nombre sigue el patrón `asalicities-`*i*`.csv`, dónde i es el número de secuencia que representa el orden en el que se realizó el experimento. Para cada colonia se realizaron 4 experimentos con 1000 hormigas cada uno, obteniendo en una hoja de cálculo en formato Microsoft Excel 2010 una comparativa, con los valores medios y desviaciones típicas.

Calibración

Esta carpeta contiene los ficheros con los resultados del estudio de calibración descrito en el Capítulo 4. En ella se encuentra el grafo utilizado en todos los experimentos (fichero `learning_graph.xml`). Los estudios relativos a los parámetros α, β, ρ y las tres variables ω_1, ω_2 y ω_3, se incluyen respectivamente en las carpetas `alfa`, `beta`, `rho` y `omegas`. El contenido de cada una de estas carpetas sigue el mismo formato que el descrito para cada una de las subcarpetas de la carpeta Ponderación, conteniendo las carpetas `alfa`, `beta`, `rho` y `omegas` un fichero CSV resultado de cada experimento y una hoja de cálculo en formato Microsoft Excel 2010 que contiene la comparativa de todos ellos. Los resultados de los experimentos derivados de la variación de los parámetros α y β se agrupan en carpetas con el formato `b`*i*`.d` que indica el valor del parámetro β. Así, la carpeta `beta/b2.0` contiene los resultados obtenidos de variar el parámetro α manteniendo fijo el valor de β=2.0.

Adaptabilidad

En esta carpeta se incluyen los ficheros resultantes del estudio de adaptabilidad del algoritmo ASALI, descrito en la sección 4.2.3 de esta Tesis. Dentro de esta carpeta encontramos dos subcarpetas: `estado_previo` y `experimentos`, así como el fichero que define en XML al grafo utilizado para el estudio de adaptabilidad, `grafo_adaptabilidad.xml`, que representa el estado inicial tras la simulación de una edición del curso por 100 alumnos a partir del grafo inicial (`learning_graph.xml`).

El estado inicial se muestra en la carpeta `estado_previo`, compuesto por el fichero `asalicities.csv`, en el que se detallan los caminos realizados por un grupo de alumnos en una edición anterior del curso; y por el fichero `asalicities_ph.csv`,

que complementa al anterior con la información de las feromonas existentes en cada arco, así como las calificaciones medias en cada nodo. La hoja de cálculo `estado-previo.xslx` resume gráficamente los datos de los ficheros anteriores, mostrando las elecciones de los alumnos en cada nodo de decisión del grafo.

La carpeta experimentos contiene una carpeta con los ficheros resultantes de cada uno de los casos descritos en la sección 4.2.3, según la proporción de alumnos de cada perfil. El contenido de cada una de estas carpetas, nombradas con el identificador del experimento (del E1 al E12) mantiene el mismo formato que los estudios ya descritos anteriormente, incluyendo un archivo de texto, en formato CSV, para cada ejecución del experimento y una hoja de cálculo en la que se reúnen todos los resultados para obtener datos estadísticos y representar gráficamente los resultados. Se incluye también la concentración de feromonas en cada arco al finalizar la ejecución en ficheros nombrados como `asalicities_ph-`*`i`*`.csv`, dónde el sufijo –i es un número de secuencia que actúa meramente como diferenciador del resto. Para la ejecución de cada experimento se utiliza el software desarrollado a tal efecto, con el objeto de simular la edición de un curso en el que participan 100 alumnos. Además de los doce experimentos definidos en base a las diferentes proporciones de perfiles de alumnos escogidos, también se estudia el comportamiento del algoritmo ASALI con una composición de alumnos de perfil aleatorio (carpeta `random`) y un experimento en el que se modifica la tasa de evaporación, pasando del 20% al 65% (carpeta `rho0.65`).

Simulador

La carpeta simulador contiene el software desarrollado para llevar a cabo el trabajo de experimentación, basado en la simulación del recorrido seguido por un conjunto de alumnos, teniendo en cuenta únicamente para el estudio, la calificación obtenida en la evaluación final en cada curso asociado a un nodo. Dentro de la carpeta aco4learning, se encuentran el código fuente (carpeta `source`), el código binario (carpeta `bin`) y el ejecutable (carpeta `release`). Para ejecutar el simulador haga doble click en el fichero asali.jar ubicado en la carpeta `release` o ejecute el comando `java –jar asali.jar` en un terminal línea de comandos de su sistema operativo. Necesitará el tener Java instalado en su sistema operativo (al menos el Java Runtime Environment, en su versión 1.5 o superior).

El simulador permite utilizar varios algoritmos: ASALI (presentado en esta Tesis), ASALI-Sim, una variación del algoritmo ASALI orientada a la recomendación de contenidos tras la ejecución simulada de varias iteraciones, y los algoritmos ACO y

AS de Marco Dorigo. Todos ellos pueden ejecutarse a través sus correspondientes opciones del menú `Run`.

El software desarrollado, cuya interfaz gráfica puede verse en la Figura 85, no sólo permite simular el comportamiento de determinados algoritmos en un grafo, sino que además permite la construcción de nuevos grafos. Para añadir nodos tan sólo es necesario, tras seleccionar el modo de edición (opción "`EDITING`" del menú "`Mouse Mode`") y hacer click en el panel blanco de la derecha, en una zona en la que no exista ya un nodo. Los arcos pueden crearse haciendo click en un nodo y, manteniendo el botón izquierdo del ratón pulsado, llevando el cursor hasta el nodo destino. La posición de un nodo puede modificarse, accediendo al modo de menú `PICKING`, que nos permitirá ajustar la posición de cada nodo de forma individual. La opción de menú `TRANSFORMING` nos permitirá mover el grafo de posición, manteniendo la forma mediante el desplazamiento solidario de todos los nodos, por igual. Los atributos de un nodo o de un arco del grafo pueden modificarse a través de la opción de edición que aparece al hacer click con el botón derecho del ratón sobre cualquier elemento del grafo.

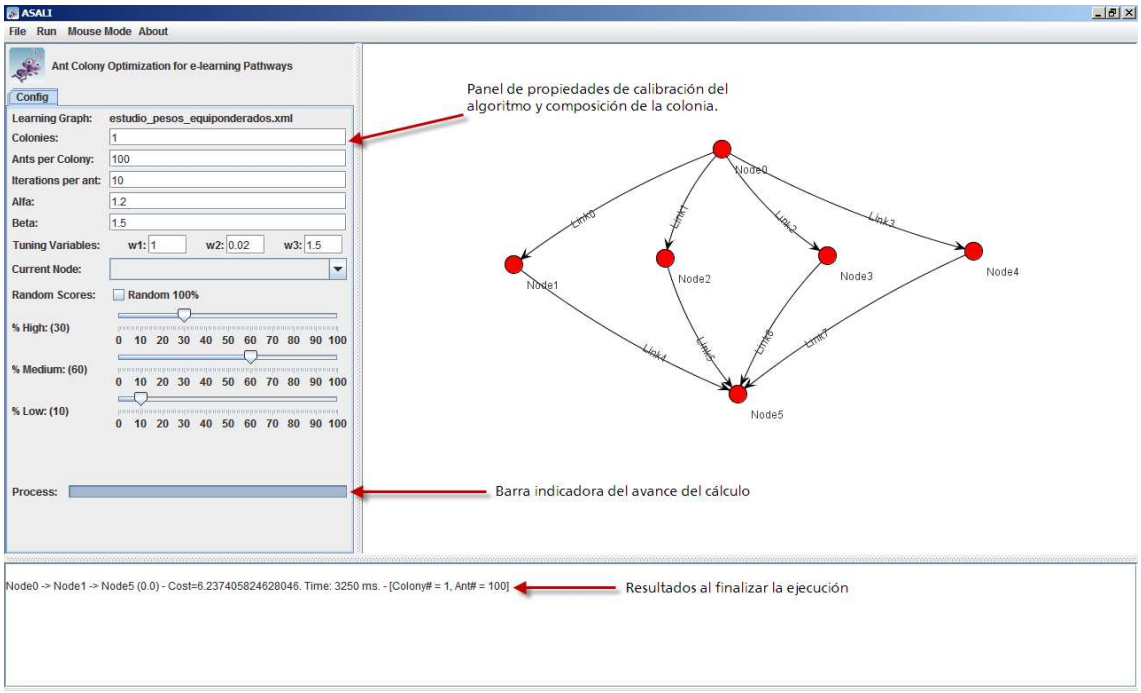


Figura 85: Interfaz gráfica del software desarrollado para simulación

Los grafos pueden salvarse en XML siguiendo la sintaxis descrita en la sección 3.3.2 de esta Tesis. Además de este mismo formato, se aceptan también otros formatos, como el utilizado para representar los grafos del problema del viajante, mucho más sencillo, pues bastar con representar por columnas las coordenadas de la posición de

cada nodo (en algunas ocasiones se representan coordenadas GPS y en otras coordenadas euclídeas, resultantes de la representación del punto en un mapa). En la carpeta TSP se muestran algunos ejemplos que pueden modificarse y utilizarse en el simulador.

TSP

La carpeta TSP contiene ejemplos relacionados con el problema del viajante. El algoritmo AS puede emplearse para encontrar el ciclo *hamiltoniano* más corto en estos grafos.

El software desarrollado acepta, además del formato definido para representar al Grafo de Itinerarios de Aprendizaje, los formatos de ficheros de TSPLIB[41], una librería para el problema del viajante publicada por la Universidad de Heidelberg.

[41] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Esta tesis se acabó de imprimir en Triana el 19 de marzo de 2012,

Bicentenario de la primera Constitución española.