DIVISION DE ESTUDIOS DE POSGRADO E INVESTIGACION

“Algoritmo de Búsqueda Semántica en Redes P2P Complejas”

PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS
EN
CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
ISC. Marco Antonio Aguirre Lam

DIRECTOR:
Dra. Laura Cruz Reyes

CODIRECTOR
Dra. Satu Elisa Schaeffer

Cd. Madero, Tams, Diciembre de 2008

Sistema Nacional de Educación Superior Tecnológica
Dirección General de Educación Superior Tecnológica

SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

SECRETARÍA DE EDUCACIÓN PUBLICA
SEP

Cd. Madero, Tam., a 26 de Noviembre de 2008.

Área: Posgrado
Nº Oficio: U5.423/08
Asunto: Autorización de Impresión de Tesis

C. ING. MARCO ANTONIO AGUIRRE LAM .
P r e s e n t e.

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su examen de grado de Maestro en Ciencias en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"ALGORITMO DE BÚSQUEDA SEMÁNTICA PARA REDES P2P COMPLEJAS"

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A t e n t a m e n t e
"POR MI PATRIA Y POR MI BIEN"

Ma. Yolanda Chavez Cinco
M.P. María Yolanda Chávez Cinco
Jefa de la División

S.E.P.
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACION
I T C M

" 2008, Año de la Educación Física y el Deporte "

MYCHC ' NLCO ' alm*

Ave. 1o. De Mayo y Sor Juana I. de la Cruz, Col. Los Mangos, C.P. 89440 Cd. Madero, Tam.
Teléfono : 01 (833) 357 48 20 al 29
Internet : www.itcm.edu.mx Correo Electrónico : itcm@itcm.edu.mx

AGRADECIMIENTOS

Mi profundo agradecimiento a los miembros del comité tutorial de esta tesis: Dra. Laura Cruz Reyes, Dra. Satu Elisa Schaeffer, Dr. Arturo Hernández Ramírez, M.C. Guadalupe Castilla Valdez y M.C. Claudia Guadalupe Gómez Santillán por las sugerencias dadas durante el desarrollo de esta tesis.

Mi sincero aprecio a la Dra. Laura Cruz Reyes, Dra. Elisa Schaeffer y M.C. Claudia Guadalupe Gómez Santillán por haber dirigido esta tesis y por las aportaciones realizadas a este trabajo.

ARTÍCULOS REALIZADOS:

Los resultados y aportaciones de esta investigación han sido presentados conjuntamente con mis asesores en los siguientes artículos internacionales:

a) "**A Comparative Study of Search Processes in Algorithms for Semantic Query Routing Systems in Complex Networks**", *In Proceedings of the 15th International Multi-Conference on Advanced Computer Systems*, Szczecin, Polonia.

b) "**NAS Algorithm for Semantic Query Routing Systems in Complex Networks**" *Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence*, Salamanca, España.

c) "**Statistical Methodology of Topological Features Selection in Complex Networks to Improve Distributed Resources Search Process**". *In Proceedings of the 15th International Congress on Computer Science Research*, Aguascalientes, México.

d) "**A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks**" *Computación y Sistemas Journal*, Publicación del Centro de Investigación en computación, IPN, México.

RESUMEN

Muchos sistemas del mundo real son modelados como redes complejas ya que son difíciles de describir debido a su gran tamaño y la interacción dinámica de sus elementos. Las redes complejas poseen una estructura topológica no trivial, lo que ha motivado el estudio de características topológicas de redes del mundo real. El conocimiento de estas características puede ser usado para optimizar el desempeño de los procesos que en ellas se llevan a cabo, tales como la búsqueda de recursos distribuidos, administración de tráfico y diseño de algoritmos de enrutamiento.

En este trabajo fue abordado el problema de búsqueda de información en redes Peer-to-Peer complejas usando estrategias locales, tales como, aprendizaje, caracterización y exploración. El proceso de búsqueda es un problema difícil de resolver ya que el ambiente en el cual se lleva a cabo éste proceso se torna complejo debido a que los usuarios realizan búsquedas de diferentes tipos de recursos en momentos diferentes y los nodos así como sus recursos pueden variar de manera dinámica. Las técnicas tradicionales de búsqueda de recursos tales como la caminata aleatoria y la inundación generan enormes cantidades de tráfico, mientras que los algoritmos de búsqueda mejorados basados en tablas *hash* introducen sobrecarga de recursos para mantener sus sistemas de búsqueda en diferentes nodos. En contraste, los sistemas auto-adaptativos tales como los algoritmos de hormigas proporcionan un paradigma adecuado de manera natural para controlar el tráfico de mensajes de consulta sobre redes P2P.

En esta tesis, se desarrolló un algoritmo evolutivo basado en algoritmos de colonia de hormigas que toma ventaja de la topología local de la red, aplicando la función de caracterización *Coeficiente de Dispersión de Grado* (*DDC*) y un método de exploración local denominado "*lookahead*", para encontrar recursos de manera eficiente sobre redes P2P. Los resultados de esta investigación muestran que al incluir información de la estructura local y aprendizaje en el algoritmo de búsqueda la eficiencia en el proceso de búsqueda obtiene una mejora del 75% utilizando instancias de prueba de un sistema real de intercambio de archivos.

SUMMARY

Many systems of the real world are modeled as complex networks since they are hard to describe because of their large size and dynamic interaction of their elements. Complex networks have a non-trivial topological structure, which has motivated the study of topological features of networks of the real world. Knowledge on such characteristics can be used to optimize the performance of processes carried out in these networks, such as the search of distributed resources, traffic management, and design of routing algorithms.

In this work, the problem of information search in Peer-to-Peer complex networks was treated, using local strategies, such as learning, characterization, and exploration. The process of search is a challenging problem because the environment in which is carried out this process becomes complex because users conducted searches of different types of resources at different times and nodes and their resources can vary in a dynamic way. Traditional resource discovery techniques such as random-walk and flooding generate enormous amounts of traffic, while improved P2P resource discovery mechanisms such as distributed hash tables introduce overload of resources to maintain their search systems in different nodes. In contrast, auto-adaptive systems such as ant algorithms provide a suitable paradigm in a natural way to control the traffic of messages for queries in Peer-to-Peer networks.

In this dissertation, an evolutionary algorithm based on ant colony algorithms was developed; it takes advantage of the local topology the network, applying the characterization function Degree Dispersion Coefficient (*DDC*) and a method of local exploration called "*lookahead*", with the objective to find resources efficiently in P2P networks. The results of this research show that include local structure information and learning in the search algorithm, the efficiency of the search process achieve an improvement of 75% using a real datasets for file-sharing Peer-to-Peer systems.

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

Capítulo 1

INTRODUCCIÓN

En este capítulo se muestran los antecedentes de la investigación realizada, se da una descripción general del problema de investigación que se aborda, y se presenta la justificación, hipótesis, objetivos y delimitaciones de este problema. Por último se presenta la organización del documento.

1.1. Antecedentes

Muchos problemas de la vida diaria pueden ser modelados como sistemas complejos que demandan entidades autónomas con capacidad de adaptarse a cambios en el ambiente y organizarse por ellas mismas para encontrar solución a nuevas condiciones. Los sistemas complejos cambian dinámicamente con el tiempo, y su estado presente es influido por los estados anteriores, entonces uno de sus principales objetivos será preservar su funcionalidad colectiva [Liu 2005].

Un aspecto importante en los sistemas complejos de gran escala es la configuración adaptable de las entidades autónomas, determinando localmente su comportamiento por ellas mismas y sin hacer uso de mecanismos de control global. Como se puede observar en la Figura 1.1, una entidad autónoma, denominada agente, recibe información del ambiente y de

otras entidades. Esta entidad tiene sus propias reglas de comportamiento locales, las cuales le ayudan a tomar decisiones acerca de su comportamiento en el ambiente, en función de la información que recibe. El término sistemas adaptables complejos fue acuñado por John H. Holland [Lewin 1992].

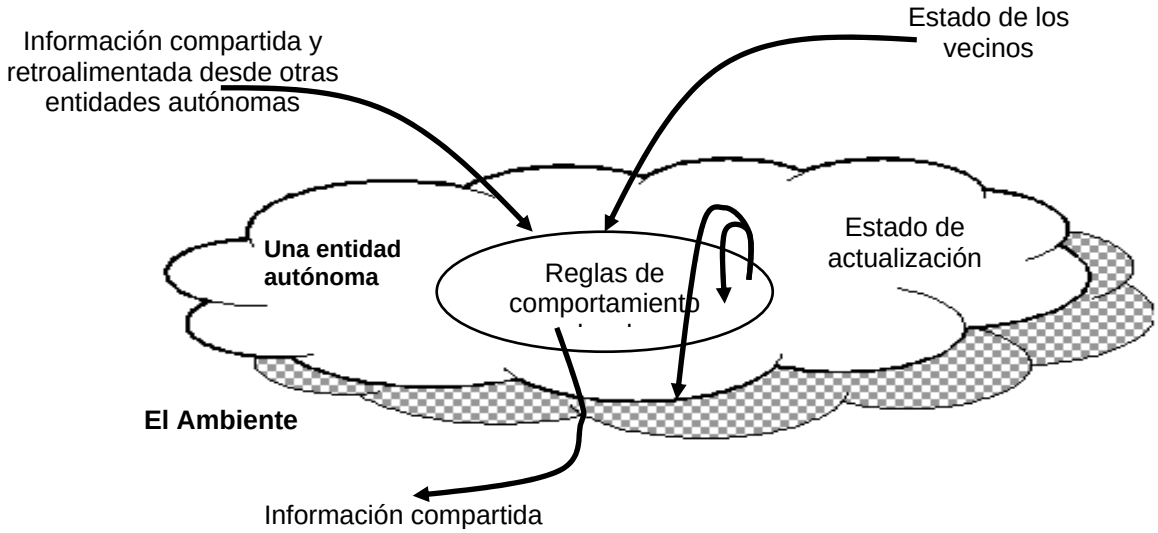

Figura 1.1. Módulos funcionales en una entidad autónoma

El campo de la informática contribuye proporcionando los instrumentos de investigación para biólogos, médicos, y sociólogos, quienes necesitan simular y representar fenómenos observados en la naturaleza, así como compartir los resultados de su investigación.

La necesidad de compartir recursos y desarrollar trabajo cooperativo ha generado que los ambientes en los que se encuentra la información se hayan vuelto complejos en su estructura, organización, distribución y acceso; constituyendo un sistema complejo. Por lo anterior, existe la necesidad de crear técnicas adaptativas que ayuden a los usuarios a encontrar la información que solicitan en tiempos de búsqueda razonables y con una mejor calidad de la información obtenida [Arenas 2003].

En la actualidad, se observa un creciente interés por formar comunidades especializadas entre organizaciones independientes con intereses comunes que comparten recursos (software, hardware y datos). Entre los esfuerzos internacionales más importantes

para integrar recursos oceanográficos a través de la Web se tiene al sistema de observación GOOS2 (The Global Ocean Observing System) y al sistema GODAE3 (The Global Ocean Data Assimilation Experiment).

En México, la Red de Observaciones y Predicciones de Variables Oceanicas (ROPVO) en las Costas y Puertos de Golfo de México (GM) tiene como objetivo compartir recursos oceanográficos a través de Internet [Meza 2002]. ROPVO-GM está integrada por un conjunto de organizaciones entre las que se encuentran: la Secretaría de Marina (SEMAR), la Universidad Nacional Autónoma de México (UNAM), el Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada Unidad Altamira del Instituto Politécnico Nacional (CICATA-IPN) y la Universidad de Texas A&M Corpus Christi (TAMUCC). Cada organización tiene sus propias políticas para compartir sus recursos, y los recursos se agregan o remueven de acuerdo a esas políticas. Por lo anterior, la centralización de recursos en una computadora no es una propuesta viable ya que la administración de recursos se vuelve un proceso complejo.

Este trabajo forma parte de un conjunto de investigaciones que se están llevando a cabo entre el CICATA-IPN y el ITCM (Instituto Tecnológico de Ciudad Madero), para dar respuesta a algunas de las necesidades que se generan al tratar de formar comunidades especializadas de investigación.

1.2. Contexto de la Investigación

El contexto de esta investigación se enfoca al estudio de búsqueda de información en redes complejas, en este caso particularmente se centra en las redes par-a-par (mejor conocidas como peer-to-peer o P2P). Un sistema complejo es aquel que contiene un gran número de componentes, los cuales pueden actuar de acuerdo a reglas que pueden cambiar a través del tiempo y que pueden no ser bien comprendidas. La conectividad de los componentes y sus roles son variables. Los sistemas complejos pueden ser modelados mediante grafos, y al ser representados se les da la denominación de redes complejas [Amaral 2004].

Una formulación matemática para una red esta dada por un grafo G, definido como un par de conjuntos $G = (V, E)$. El conjunto V vértices representa los nodos de la red, y sus elementos pueden ser etiquetados por números enteros 1, 2,..., n. El número de elementos de V, $n = |V|$ es el orden del grafo. Otra manera de representar los nodos es mediante letras minúsculas. Las conexiones de la red son representadas por el conjunto E de aristas. Se asume que una arista es un par ordenado de vértices distintos, denotado simplemente por $\{v, w\}$ [Schaeffer 2006].

Una red P2P se puede definir como un sistema distribuido formado de nodos interconectados, capaces de auto-organizarse con el propósito de compartir recursos, sin requerir de la intermediación o soporte de un servidor o autoridad centralizada [Androutsellis-Theotokis 2004].

Debido a las características antes mencionadas es posible generar diferentes tipos de consultas. Las consultas más populares son aquellas basadas en palabras clave, algunos sistemas P2P populares que utilizan este tipo de consulta son: Napster [Saroiu 2003], Gnutella [Clip2 2000], Freenet [Yang 2002], Kazaa [Liang 2004] y Limewire [Ivkovic 2001]. El área de investigación que se con relaciona las búsquedas de este tipo se les llama búsquedas basadas en contenido (content-based search) [Zhou 2004], y recientemente se les ha denominado ruteo de consultas semánticas (semantic query routing problem, SQRP) [Michmayr 2007]. Lo que distingue a éste tipo de búsquedas de otras existentes (tales como: búsquedas de imágenes, texto libre, etc.) es que en éstas los datos de búsqueda necesitan tener una descripción semántica asociada con ellos.

De acuerdo con Michmayr, en SQRP cada nodo esta conectado a través de un enlace de salida a otros nodos los cuales son llamados sus nodos vecinos, el nodo tiene que decidir basado en información local a cual nodo vecino debe enviar la consulta. La información local de un nodo se actualiza continuamente observando las consultas y respuestas que pasan a través del nodo local, y almacenando que tipo de consultas fueron capaces de responder sus

nodos vecinos en el pasado. El problema consiste en encontrar una ruta corta entre el nodo que emite la consulta y el nodo que puede contestar esa consulta de manera apropiada.

1.3. Descripción del Problema de Investigación

El problema que se aborda en esta tesis es de tipo SQRP, y se define formalmente como sigue:

INSTANCIA: Una red P2P representada por el grafo de conexiones (G), un conjunto de contenidos (R) distribuidos en sus nodos y un conjunto de consultas semánticas (C) emitidas por los nodos, donde cada consulta semántica puede ser originada desde cualquier nodo en el tiempo T_0 $(\forall T_0 \in \square)$ asumiendo un tiempo de reloj con unidades fijas. Un nodo que origina la consulta o recibe la consulta de otro nodo en el tiempo (T_0+i) $(\forall i \in \square^+ \cup \{0\})$ puede procesar la consulta localmente y/o reenviar una réplica de la consulta a un conjunto de vecinos inmediatos en el tiempo T_0+i+1. El procesamiento de la consulta termina cuando se cumple la condición de paro especificada, ya sea cuando se encuentra la cantidad máxima de recursos buscados o cuando el tiempo de vida determinado para la entidad de búsqueda se haya agotado.

PREGUNTA: ¿Existe un conjunto de rutas entre los nodos que emiten las consultas y los nodos que contienen los recursos, tal que maximice la cantidad de recursos encontrados y minimice la cantidad de saltos realizados por una entidad de búsqueda para encontrar dichos recursos?

A continuación se muestra un ejemplo sencillo de una instancia de SQRP y su solución:

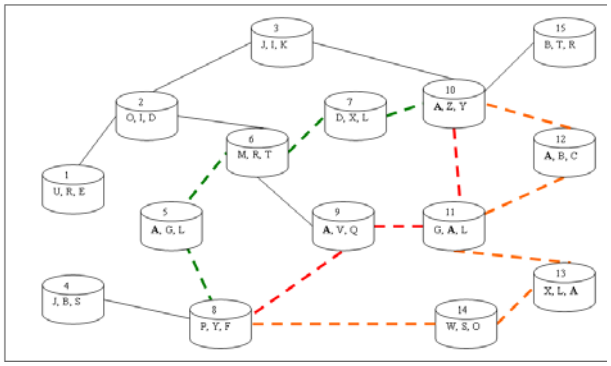

Figura 1.2. Ejemplo de una instancia SQRP

La Figura 1.2 muestra el ejemplo de una instancia de SQRP, de 15 nodos (con identificadores enumerados consecutivamente desde 1 hasta 15), cada uno tiene un repositorio con tres recursos que son representados como clases de documentos (con identificadores seleccionados del abecedario desde la letra A hasta la Z).

La Tabla 1.1 muestra los resultados de tres rutas generadas por consultas desde el nodo 8 buscando recursos de la clase A, se incluyen la ruta generada, la cantidad de saltos (que son contabilizados en cada movimiento de un nodo hacia otro), los recursos encontrados y una medida de desempeño que se calcula dividiendo la cantidad de saltos entre la cantidad de recursos encontrados, lo que da como resultado la cantidad de saltos necesarios para encontrar un recurso.

Tabla 1.1. Ejemplo de rutas generadas para una instancia de SQRP

Ruta		Cantidad de Saltos	Recursos encontrados	Saltos necesarios para Encontrar un recurso
1	{8, 5, 6, 7, 10}	5	2	2.5
2	{8, 9, 11, 10}	4	3	1.33
3	{8, 6, 14, 13, 11, 12}	5	4	1.25

Como se observa en la Tabla 1.1, la ruta 3 cumple con el objetivo de SQRP, debido a que maximiza la cantidad de recursos encontrados y minimiza la cantidad de saltos utilizados para encontrar dichos recursos.

1.4. Justificación

En la actualidad los usuarios de Internet forman comunidades virtuales científicas y sociales para compartir recursos e información. Las comunidades virtuales son sistemas administrados de manera centralizada o distribuida, donde los sistemas distribuidos, en especial las redes P2P, han adquirido una gran importancia debido a sus ventajas sobre los sistemas centralizados.

Sin embargo, el desarrollo de tales comunidades virtuales actualmente carece de métodos de búsqueda eficientes [Michlmayr 2007]. Entre las diferentes características que

hacen compleja la tarea de búsqueda de recursos distribuidos se encuentran: la naturaleza dinámica de la topología de conexión, los recursos contenidos en la red de los miembros de las comunidades, las limitaciones de hardware, que influyen debido a la gran escala de la información y del tamaño de la red, y por otra parte están los protocolos de conexión.

En este trabajo la motivación principal es crear técnicas adaptativas que ayuden a los usuarios a encontrar la información que solicitan en tiempos razonables de búsqueda y con una mejor calidad de la información obtenida, es decir, se pretende optimizar el desempeño de los procesos de búsqueda.

1.5. Hipótesis

Para dar solución a SQRP se propone modelar e implementar un sistema de consultas P2P. Bajo la metodología de AOC (Autonomy Oriented Computing) [Liu 2005], el sistema complejo P2P se puede definir como la tupla

$$<\{e_1,e_2,\ldots,e_i,\ldots,e_N\}, E,\Phi>$$

donde $\{e_1,e_2,\ldots,e_i,\ldots,e_N\}$ es un subconjunto del conjunto total de N entidades autónomas que operan en la red compleja P2P. E es el ambiente en el cual residen las entidades y esta formado por los conjuntos G, R y C descritos en la sección 1.3, y Φ es la función objetivo que consiste en maximizar la cantidad de recursos encontrados y minimizar la cantidad de saltos realizados por una entidad de búsqueda para encontrar dichos recursos.

El reto de este trabajo es el diseño de entidades autónomas que satisfagan los requerimientos dados por cualquier instancia de SQRP caracterizada en E y Φ. El objetivo es maximizar la cantidad de recursos encontrados y minimizar la cantidad de saltos realizados por una entidad de búsqueda para encontrar dichos recursos. El desarrollo de este trabajo se sustenta en las siguientes hipótesis:

Es posible optimizar el proceso de búsqueda de información en redes P2P mediante el diseño de entidades autónomas que permitan guiar la búsqueda a través de técnicas de aprendizaje automático y métricas de información local.

1.6. Objetivos

Objetivo General

Desarrollar un sistema de agentes inteligentes que tengan la capacidad de enrutar consultas para localizar información de texto en redes P2P modeladas como redes complejas, usando tanto aprendizaje automático como métricas topológicas y de importancia de contenido.

Objetivos Específicos

a) Diseñar una arquitectura de agentes autónomos para el problema de enrutamiento de consultas semánticas en redes P2P.
b) Caracterizar el ambiente del sistema de agentes mediante métricas topológicas que posibiliten que la navegación se oriente hacia nodos fuertemente conectados.
c) Formular un conjunto de métricas de importancia de contenido que posibiliten que la navegación se oriente hacia nodos con información relevante para la búsqueda.
d) Desarrollar un algoritmo metaheurístico con aprendizaje automático para implementar la arquitectura propuesta.

1.7. Delimitaciones

1. El ambiente de búsqueda (red P2P) y los repositorios son estáticos, solo las consultas son generadas de manera dinámica, pero creadas artificialmente.
2. El tamaño de las muestras de los repositorios (topología, consultas y contenido) dependen de la cantidad de memoria disponible.

1.8. Organización del Documento

El presente documento se encuentra organizado de la siguiente manera, en el capítulo 2 se aborda la comparación de los sistemas de organización de computadoras en Internet y la relación de los sistemas peer-to-peer con los sistemas complejos y los sistemas multi-agente.

En el capítulo 3 se aborda el marco teórico de los algoritmos de colonia de hormigas: los algoritmos de colonia de hormigas básicos y los distribuidos, así como una discusión acerca de los elementos que pueden ser adaptados al problema de enrutamiento de consultas semánticas que se aborda en esta tesis.

Con el motivo de evaluar el estado del arte de los trabajos relacionados con consultas semánticas y aquéllos trabajos que aplican la teoría de los algoritmos de hormigas, se realizó una discusión de estos trabajos en el capítulo 4.

En el capítulo 5 se presenta la metodología de solución a través del algoritmo de solución propuesto "*Neighboring-Ant Search"*, en el cual se incluye la arquitectura del sistema, las estructuras de datos manejadas y las funciones propias del algoritmo.

Para evaluar el desempeño del algoritmo de solución propuesto, en el capítulo 6 se presenta la experimentación realizada con diferentes configuraciones de experimentación.

Del desarrollo de esta tesis se desprendió una serie de conclusiones, y se identificó un conjunto de trabajos futuros que podrían contribuir al incremento de la eficiencia del algoritmo de solución propuesto. Lo anterior se presenta en el capítulo 7.

Capítulo 2

ORGANIZACIÓN DE INTERNET

En este capítulo se abordan características, ventajas y desventajas de dos tipos de organización existentes sobre la Internet, tales como la computación *grid* y *P2P*. Así como los métodos de búsqueda más populares sobre redes P2P.

2.1. Internet

Internet fue inventado en 1969 como resultado del proyecto administrado por el Departamento de Defensa de los Estados Unidos, que reconoció la importancia de las computadoras para la investigación, dirección, control y comunicación. Inicialmente, el Internet conectaba los servidores de la Universidad de California, Los Ángeles (UCLA), el Instituto de Investigación de Standford, la Universidad de California en Santa Bárbara, y la Universidad de Utah (UTAH) en Salt Lake City. Desde sus inicios, Internet fue diseñado como un sistema descentralizado capaz de sobrevivir durante ataques que pudieran destruir la arquitectura central [TechWeb 2001].

Originalmente, Internet era usado principalmente para correo electrónico, grupos de noticias, conexiones remotas a computadoras, y acceso a la información almacenada en archivos en otras máquinas. Todas estas operaciones eran complicadas y requerían un cierto

nivel de requisitos. La naturaleza descentralizada de la Internet significó que no hubiera un directorio central en donde un usuario pudiera buscar información [Bergman 2001].

Para resolver el problema de búsqueda de información, los investigadores de la Organización Europea para la Investigación Nuclear (CERN) propusieron la noción de hipermedia distribuida en 1989. Un documento hipertexto puede incluir texto, sonido, animación y gráficos en un simple documento. Además puede incluir enlaces a otros documentos aun si están almacenados en otras computadoras. Para 1993, fue desarrollado el primer software conocido como buscador, éste daba soporte a documentos de hipertexto para manejar la carga de documentos desde otros servidores en Internet. El World Wide Web fue creado a principios de 1993, con solo casi 50 sitios que soportaban documentos hipertexto [Bergman 2001].

La escala alcanzada por la Web y su naturaleza caótica y distribuida hace difícil encontrar la información que los usuarios necesitan. No existen directorios centrales que listan sitios disponibles con una explicación acerca de los contenidos de los sitios.

Por consiguiente, los usuarios de la Web confían principalmente en los resultados proveídos por máquinas de búsqueda tales como Google, Lycos, Yahoo, Altavista entre otras, las cuales emplean una base de datos de documentos Web indexados almacenados centralmente. Para obtener tales documentos e indexarlos, las máquinas de búsqueda constantemente buscan en la red. Google es el buscador más popular, y lleva a cabo más de 200 millones de búsquedas por día, construyendo el índice de páginas Web más grande y cubre un poco menos del 1% de la Web [Google 2003].

La necesidad de compartir recursos a través de Internet, la cual es considerada un Sistema Complejo [Guillaume 2006], ha motivado un incremento en la investigación de nuevas formas de organizar la estructura de las redes de comunicación. Actualmente dos de las organizaciones más estudiadas son: grid y peer-to-peer.

Estas dos clases de organizaciones tienen diferentes orígenes y líneas de evolución, ambas abordan el problema de compartir recursos en un ambiente virtual de colaboración entre organizaciones e individuos [Foster 2003]. En ambos ambientes de colaboración los mecanismos de búsqueda son básicos y sus diseños imponen nuevos retos de comunicación.

2.2. Arquitecturas de Computación: Grid vs. Peer-to-Peer

La malla computacional o mejor conocida como computación *grid* es un paradigma que propone el agregado de computación heterogénea, almacenaje y recursos de red geográficamente distribuidos para proveer un acceso generalizado a sus capacidades combinadas, en conjunto se le llama simplemente *grids* [Ardenghi 2007].

Los grids de datos [Chevernak 2000] [Hoschek 2000] están orientados principalmente a ofrecer servicios e infraestructura para aplicaciones intensivas sobre datos distribuidos que necesitan acceder, transferir y modificar conjuntos de datos masivos almacenados en recursos de almacenaje distribuidos. Algunos ejemplos de grids computacionales comprenden empresas de tecnologías de computación distribuida tales como la organización CORBA, el Ambiente del Grupo Abierto de Computación Distribuida (The Open Group's Distributed Computing Environment, DCE) que soporta compartir recursos seguros entre sitios, los Proveedores de Servicios de Almacenamiento (Storage Service Providers, SSP) y los Proveedores de Servicios de Aplicación (ASP) que permiten a las organizaciones subcontratar requisitos de almacenamiento y computación para otras empresas [Foster 2003].

Por otra parte, las arquitecturas par-a-par mejor conocidas por su nombre en inglés *peer-to-peer* (P2P), son la base de la operación de los sistemas de computación distribuida. Las motivaciones de su uso radican en sus características inherentes tales como escalabilidad, resistencia a la censura (libertad de publicación), y creciente acceso a recursos en presencia de una gran población de nodos, redes y fallas de computadoras sin necesidad de un servidor central y la sobrecarga de su administración. La administración está repartida. Aceleran

procesos de comunicación y reducen los costos de colaboración por medio de una administración de grupos de trabajo [Ardenghi 2007].

Algunos ejemplos de aplicaciones P2P comprenden redes académicas, tales como LionShare [Halm 2006], redes militares, tales como DARPA [DARPA 2008], Limewire, que es una aplicación para compartir archivos [Ivkovic 2001], y Skype [Guha 2005] que es una red de telecomunicaciones.

Los *grids* son sistemas distribuidos que habilitan el uso coordinado en gran escala de recursos distribuidos geográficamente, basados en almacenamiento de archivos, infraestructuras de servicios estandarizadas, frecuentemente con una orientación al alto rendimiento. Sin embargo, a medida que un sistema *grid* se incrementa en escala, requiere soluciones para la autoconfiguración, tolerancia a las fallas y escalabilidad. Sobre estas soluciones, la investigación en P2P tiene mucho para ofrecer.

Los sistemas P2P, por otro lado, se enfocan en tratar con la instantaneidad, las poblaciones transitorias, la tolerancia a las fallas y la autoadaptación. Con lo antes mencionado se concluye que los sistemas P2P ofrecen ventajas sobre otras arquitecturas de sistemas existentes [Ardenghi 2007].

2.3. Redes Peer-to-Peer (P2P)

Los sistemas P2P son definidos como sistemas distribuidos consistentes de nodos interconectados capaces de organizarse a si mismo dentro de diferentes topologías de red, con el propósito de compartir recursos tales como contenidos, ciclos de CPU, almacenaje y ancho de banda, capaces de adaptarse a las fallas y acomodarse a poblaciones transitorias de nodos mientras mantienen una aceptable conectividad y rendimiento, sin requerir intermediación o soporte de un servidor o autoridad global centralizada [Androutsellis-Theotokis 2004].

En los últimos años las redes P2P han ganado amplia popularidad sobre otros sistemas para compartir archivos debido a que son útiles para muchos propósitos, tales como compartir

archivos de audio, video, datos o cualquier archivo en formato digital, además de presentar las ventajas mencionadas en la sección 2.2.

Actualmente existen diferentes arquitecturas de redes P2P, éstas pueden ser de dos clases: estructuradas o no estructuradas. En este capítulo cuando se haga mención a la palabra par, se estará refiriendo a un nodo de la red P2P.

2.3.1. Redes P2P estructuradas y no estructuradas.

Las redes P2P son construidas en la capa de aplicación sobre una infraestructura de comunicación en la capa física, tal como la Internet. Estas pueden ser vistas como grafos en los cuales los nodos son pares y las aristas son los enlaces entre estos pares. Ya que sólo unos cuantos de todos los nodos de la capa física participan en la red P2P y los enlaces entre estos nodos pueden estar compuestos de varios enlaces físicos, las redes P2P son también conocidas como capas de red [Michlmayr 2007]. Existen diferentes formas para la estructura (topología) de capas de red, las cuales serán descritas a continuación.

Capas de red estructuradas

Estos sistemas no tienen un servidor de directorio central, y también son descentralizados pero tienen una cantidad significante de estructura. Por "estructura" se da a entender que la topología de red P2P es completamente controlada y que los archivos no son colocados en nodos aleatorios sino en direcciones que harán que las consultas subsecuentes sean más fáciles de satisfacer. En sistemas de "estructura débil" éste acomodo de archivos esta basado en pistas. En sistemas "altamente estructurados" tanto la topología de red y el acomodo de los archivos son determinados de manera precisa; esta estructura completamente controlada facilita al sistema satisfacer consultas muy eficientemente [Lv 2002].

Capas de red no estructuradas

Estos son sistemas en los cuales no existe un directorio centralizado ni ningún control preciso sobre la topología de red o acomodo de archivos. La red está formada por nodos que se unen a

la red siguiendo algunas reglas permisivas (Por ejemplo, las reglas de conexión, tener cierto tamaño de archivos para compartir, etc.). La topología resultante tiene ciertas propiedades, pero el acomodo de los archivos no está basado en algún conocimiento de la topología. Para encontrar un archivo, un nodo consulta a sus vecinos.

El método de consulta tradicional es por inundación o caminatas aleatorias donde la consulta es propagada a todos los vecinos dentro de un cierto radio [Clip2 2000]. Estos diseños no estructurados son extremadamente resistentes a la entrada y salida de los nodos del sistema. Sin embargo, los actuales mecanismos de búsqueda son extremadamente no escalables, generando grandes cargas sobre los participantes de la red [Lv 2002].

Este trabajo de tesis se enfoca al estudio de los mecanismos de búsqueda basados en redes P2P no estructuradas debido a que las características de estas redes les permiten ventajas sobre otros modelos de red P2P existentes. Otro factor importante para este enfoque es la falta de mecanismos de búsqueda inteligentes que provean a los usuarios accesos eficientes a los recursos disponibles en la red.

2.3.2. Métodos de búsqueda en redes P2P

Las redes P2P han llegado a ser uno de los mayores tópicos de investigación en los últimos años. En estos sistemas distribuidos la tarea principal que se lleva a cabo es la localización de recursos [Tsoumakos 2003].

Existen dos estrategias posibles para búsquedas en redes P2P: las que se llevan a cabo sobre redes P2P no estructuradas, denominadas *búsquedas ciegas*, que intentan propagar la consulta a un número suficiente de nodos para satisfacer la consulta; y las que se llevan a cabo en redes estructuradas, tomando información acerca de la ubicación de documentos, llamadas *búsquedas informadas*. La semántica de la información utilizada abarca desde el envío de pistas simples hasta direcciones de recurso exactas [Tsoumakos 2003]. Existen diferentes clasificaciones de los métodos de búsqueda en redes P2P. En la Figura 2.1 se muestra una

taxonomía de estas búsquedas que es presentada en [Tsoumakos 2003], en la cual fueron añadidos algunos métodos de búsqueda que han sido más citados en la literatura; líneas abajo se describen los métodos considerados.

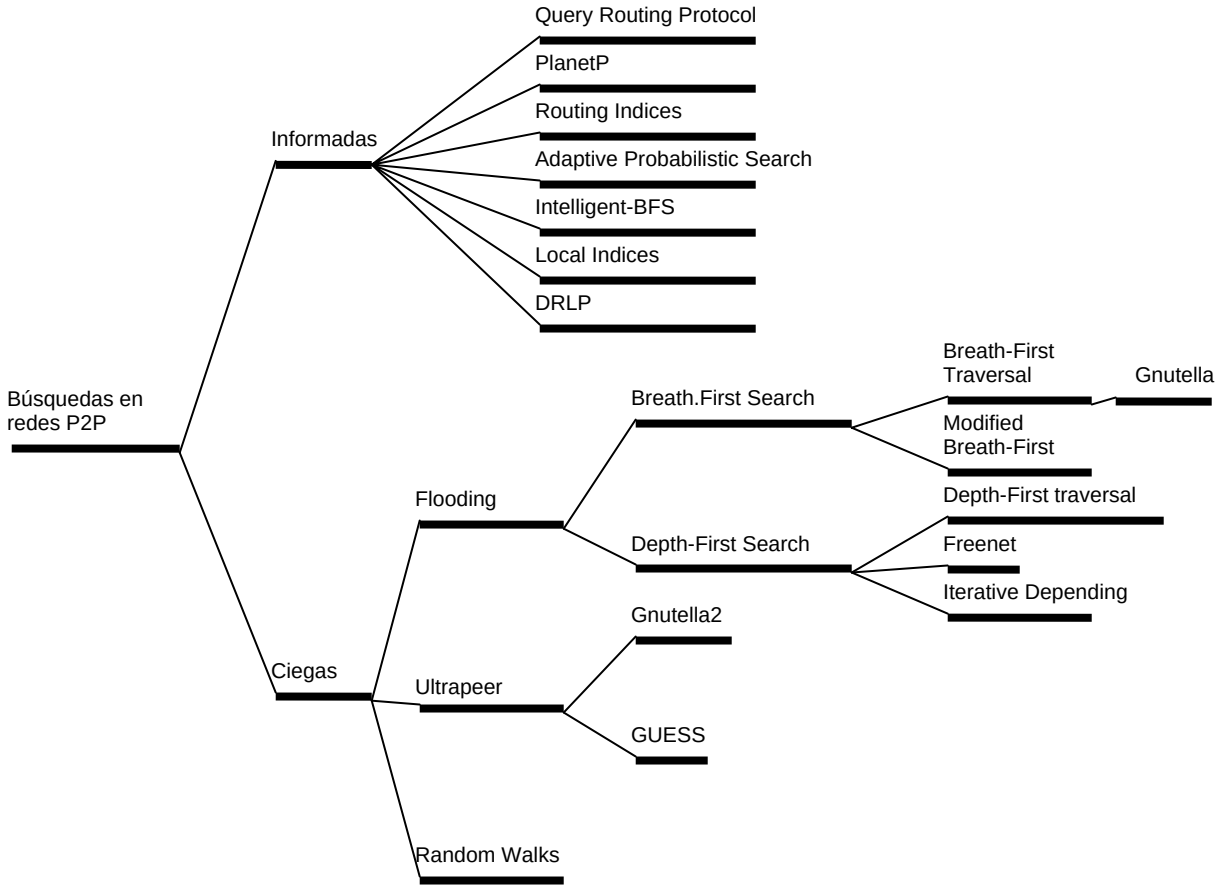

Figura 2.1. Taxonomía de las búsquedas en redes P2P

2.3.2.1 Métodos de búsqueda ciega

Búsqueda en Amplitud (Breadth-first search, BFS) y Búsqueda en Profundidad (Depth-first search, DFS)

Estos métodos de búsqueda son también conocidos como inundación (flooding). Las consultas son enviadas recursivamente a todos los nodos vecinos alcanzables. El alcance de la

transmisión, también referido como el parámetro time-to-live (TTL), es definido por la cantidad de saltos que se encuentran entre el nodo consultante y el nodo más lejano que intenta responder la consulta. En una búsqueda en profundidad, una consulta es transmitida recursivamente a todos los vecinos que pueden ser alcanzados dentro de un número definido de saltos. En la búsqueda en amplitud se utilizan menos recursos (en función del tiempo de ejecución) que en una búsqueda en profundidad, donde los nodos envían una consulta a todos sus vecinos secuencialmente, y termina después de que la consulta es satisfecha. La búsqueda en amplitud y la búsqueda en profundidad son técnicas conocidas en la teoría de grafos [Kocay 2005].

Búsqueda en Amplitud Modificada (Modified BFS)

Esta es una variación del esquema de inundación básico, con pares que eligen aleatoriamente solo una proporción de sus vecinos para enviar la consulta. Este algoritmo ciertamente reduce la producción del mensaje promedio comparado con el método de Gnutella, pero continúa contactando un gran número de puntos [Kalogeraki 2002].

Búsqueda en Amplitud Cruzada (Breadth-First Traversal)

Este método es parecido al método de búsqueda en amplitud de un árbol. En la búsqueda en amplitud de un árbol se realiza la búsqueda visitando los nodos en el orden de su profundidad en el árbol; primero visita todos los nodos en la profundidad cero (por ejemplo, la raíz), después todos los nodos en la profundidad uno, y demás.

Ya que un grafo no tiene raíz, cuando se realiza la búsqueda en amplitud cruzada, se debe especificar el vértice para iniciar el cruzamiento. Además se debe definir la profundidad del vértice dado para que sea la longitud de la ruta más corta desde el vértice inicial hasta el vértice dado. De esta manera, la búsqueda en amplitud cruzada visita el vértice inicial, después todos los vértices adyacentes al vértice inicial, y todos los vértices adyacentes a esos, y demás [Preiss 1999].

Gnutella

El algoritmo original de Gnutella utiliza inundación (basada en la búsqueda en amplitud cruzada) para encontrar recursos y visitar todos los pares accesibles dentro un valor de TTL definido. Sin embargo es simple y se dirige a encontrar el número máximo de recursos en esa región durante el tiempo de vida definido, éste método no tiene noción del tráfico generado, produciendo un enorme costo para un gran número de pares [Clip2 2000].

Búsqueda Iterativa en Profundidad (Iterative Deepening)

Estos algoritmos se basan en el método BFS, la diferencia es que la búsqueda iterativa en profundidad logra mejores resultados ya que define la condición de terminación de la búsqueda en relación a un número de pistas definido por el usuario (tales como cantidad de resultados deseados, etc.) y es posible que una "pequeña" inundación satisfaga la consulta. En caso de que no exista una condición de paro como el número de pistas, éstos producirán la misma carga de recursos que el mecanismo estándar de inundación [Lv 2002].

Búsqueda en Profundidad Cruzada (Depth First Transveral)

La búsqueda en profundidad cruzada es parecida a la búsqueda en profundidad cruzada de un árbol. Una búsqueda en profundidad cruzada de un árbol siempre comienza en la raíz de un árbol. Ya que un grafo no tiene raíz, cuando se realiza la búsqueda en profundidad cruzada, se debe especificar el nodo en el cual se debe comenzar la búsqueda.

En una búsqueda en profundidad cruzada de un árbol se visita un nodo y recursivamente los sub-árboles de ese nodo. De la misma manera, la búsqueda en profundidad cruzada visita un nodo y después recursivamente visita todos los nodos adyacentes de ese nodo.

El detalle, es que el grafo puede contener ciclos, pero el cruzamiento debe visitar cada vértice una vez como máximo. La solución a este problema es mantener la pista de los nodos

que han sido visitados, de tal manera que el cruzamiento no sufra una recursión infinita [Preiss 1999].

Freenet

Utiliza una búsqueda en profundidad (DFS) con un límite de profundidad definido. Cada nodo envía la consulta a un simple vecino, y espera por una respuesta definida desde el vecino, antes de enviar la consulta a otro vecino (si la consulta no fue satisfecha), o envía los resultados de regreso a la fuente de la consulta (si la consulta fue exitosa) [Yang 2002].

Gnutella2

Cuando un *superpar* (conocido también como *hub* o *ultrapar*) recibe una consulta de un par hoja, éste lo envía hacia sus nodos hoja relevantes (aquellos que tengan la información solicitada) y también hacia sus hubs vecinos. Estos hubs procesan la consulta localmente y la envían hacia sus hojas relevantes. Ningún otro par es visitado con este algoritmo. Los hubs vecinos regularmente intercambian tablas repositorio para filtrar tráfico innecesario entre ellos [Stokes 2003].

Guess

Este algoritmo construye sobre la noción de ultrapares (ultrapeers). Cada ultrapar es conectado a otros ultrapares y a un conjunto de pares hoja, actuando como su proxy (programa o dispositivo que realiza una acción en representación de otro). Cuando se realiza una búsqueda se lleva a cabo contactando diferentes ultrapares (los cuales tienen almacenada información de los contenidos de los nodos hoja en sus índices), hasta que un número de recursos son encontrados. El orden con el cual son escogidos los ultrapares no es especificada [Daswani 2000].

Caminata Aleatoria (Random Walks)

En el algoritmo de caminata aleatoria los nodos de la red no contienen información acerca de la ubicación del contenido de los recursos solicitados a menos que se encuentre en él mismo. Sea *G* un grafo que modela la red y *v* un nodo en *G*. Una caminata aleatoria de T-saltos desde *v* en *G* es una secuencia de variables aleatorias dependientes X_0 ,..., X_T definidas como sigue: $X_0 = v$ con una probabilidad de 1 y por cada $i = 1,... T$, el valor de X_i es seleccionado uniformemente aleatorio entre los nodos $\Gamma(X_{i-1})$, es decir, entre los nodos vecinos del nodo actual. En pocas palabras, una caminata aleatoria comienza en un nodo y en cada paso se mueve aleatoriamente hacia un nodo vecino del nodo actual, hasta llegar al nodo que contiene el recurso solicitado [Arora 2007].

2.3.2.2 Métodos de búsqueda informada

Protocolo de ruteo de consulta (Query Routing Protocol, QRP)

Para realizar una búsqueda más enfocada y para evitar flujo de red sin sentido, Rohrs propuso el algoritmo QRP [Rohrs 2001]. Los pares son responsables de crear sus propias tablas de ruteo que incluyen palabras clave *hash*, y regularmente las intercambian con sus vecinos. Estas palabras clave son usadas para describir los archivos ofrecidos localmente [Sakaryan 2004]. Las tablas de ruteo calculadas son propagadas a varios puntos de conexión que están alejados lo más posible del punto al que pertenecen, esto para reducir el número de conexiones requeridas. Si un par recibe tablas de ruteo de sus vecinos, este puede combinar las tablas y propagar la tabla combinada hacia sus vecinos.

PlanetP

La idea de usar filtros Bloom [Bloom 1970] para resumir el índice de contenido de cada par fue empleado en PlanetP [Matias 2002]. Un filtro Bloom es una estructura probabilística que es usada para probar si un elemento es miembro de un conjunto de datos. PlanetP usa filtros para crear un índice de contenido de la red P2P completa. Una vez que en un par ha sido

calculado su filtro Bloom, éste lo difunde a través de la red usando un algoritmo *gossiping* (rumor, chisme) [Demers 1987].

Para suministrar una búsqueda más selectiva que ayudaría a los usuarios a navegar mejor en un conjunto grande de documentos, PlanetP implementó un algoritmo de clasificación de la información distribuida en cada par, basándose en el modelo de vector de espacio [Witten 1999]. La única desventaja que tiene este algoritmo es la característica de escalabilidad, ya que entre más pares existan en la red se crea más tráfico y mayor consumo de recursos.

Indices de Ruteo (Routing Indices, RI)

Un método similar al QRP fue empleado por Índices de Ruteo [Crespo 2002]. En contraste con QRP, RI no usa filtros Bloom para resumir el contenido de los vecinos. En su lugar, usa categorización de documentos, que es una tarea complicada para aplicaciones reales. Cada par mantiene índices de ruteo para cada uno de los vecinos, indicando cuantos documentos y de que grupos podrían ser encontrados a través de ese vecino. Si una conexión nueva es instalada, un par agrega todos los índices de sus vecinos, añadiendo información acerca de documentos compartidos localmente y manda una tabla de ruteo combinada al nuevo vecino.

Para mantener la actualidad de los datos, los puntos deben informar a los vecinos acerca de cambios en la tabla de ruteo causados por añadir o remover la entrada o salida de los documentos de otros vecinos. Para propósito práctico, esto puede causar retransmisión frecuente de los datos dado que la red esta constantemente cambiando. Si más documentos pudieran ser encontrados a través de un vecino particular, ese vecino sería seleccionado [Sakaryan 2004].

Búsqueda Adaptativa Probabilística (Adaptive Probabilistic Search, APS)

La búsqueda Adaptativa Probabilística (APS) [Tsoumakos, 2003] utiliza la retroalimentación de búsquedas anteriores para realizar búsquedas futuras mas enfocadas. Cada punto mantiene un índice local que describe que recursos fueron solicitados por cada vecino. La probabilidad de escoger un vecino para encontrar un documento particular depende de los resultados previos de la búsqueda. La actualización toma una dirección inversa de regreso al punto de búsqueda inicial y puede tomar lugar después del éxito o fracaso, ajustando de acuerdo a la probabilidad.

El método usado guía a situaciones cuando archivos populares pudieron ser localizados más rápido, mientras otros archivos pudieron ser difícilmente localizados. Otras limitaciones son causadas por el hecho de que el primer par descubierto podría ser más usado para un ruteo futuro y entonces experimentar mas carga. Al mismo tiempo, otros puntos que están más cercanos (en tiempo o longitud de dirección) pudieran ser ignorados [Sakaryan 2004].

Búsqueda en Amplitud Inteligente (Intelligent-BFS)

Esta es una versión informada del algoritmo de búsqueda en amplitud modificada. Los nodos almacenan las tuplas consulta-vecinoID para responder peticiones desde (a través de) sus vecinos para clasificarlos. Las consultas son realizadas mediante texto libre, si no existe información de la consulta el método utilizado es la búsqueda en amplitud modificada. Si existe información de las consultas anteriores, un nodo identifica todas las consultas similares a la actual, de acuerdo a la métrica de similitud de la consulta; entonces se elige enviarla a un conjunto de sus vecinos que han regresado la mayoría de los resultados para estas consultas. Si encuentra el objetivo, la consulta toma la dirección anterior hacia el nodo que realizó la petición y actualiza los índices locales.

Algunas desventajas de éste método consisten en la gran cantidad de mensajes que se crean cuando no existe información de las consultas anteriores, no tiene un mecanismo de

castigo para rutas no exitosas, y la precisión de las consultas depende de que los nodos se especialicen en algún tipo de documento [Tsoumakos 2003].

Índices Locales (Local Indices)

Cada nodo indexa los archivos almacenados en todos los nodos dentro de un cierto radio *r* y puede responder consultas en beneficio de todos ellos. Una búsqueda es desarrollada de manera BFS, pero solo los pares que son accesibles desde el nodo que realizó la petición pueden contestar la consulta [Crespo 2002].

Por otra parte, la cantidad de mensajes enviados se compara con la cantidad de mensajes producidos en la técnica de inundación, también requiere de un $TTL = r$ ya sea cuando un par se une o abandona la red o cuando actualice su repositorio local, de esta manera existe una sobrecarga cuando los ambientes dinámicos se vuelven más grandes [Tsoumakos 2003].

Protocolo de Localización de Recursos Distribuidos (Distributed Resource Location Protocol, DRLP)

Los nodos sin información acerca de la localización de un archivo consultan a cada uno de sus vecinos con cierta probabilidad. Si un objeto es encontrado, la consulta toma la dirección del nodo que realizó la petición, almacenando la dirección del documento en cada uno de los nodos visitados. En peticiones subsecuentes, los nodos con información de la dirección indexada contactan directamente al nodo específico. Si ese nodo no obtiene el documento, de manera iterativa se inicia la misma búsqueda en los nodos vecinos seleccionados, como se describió anteriormente [Menascé 2002].

En redes dinámicas donde los cambios son rápidos, este método falla y los pares tienen la necesidad de realizar una búsqueda ciega. También esto afecta a la cantidad de recursos encontrados: si se realizan muchas búsquedas ciegas, entonces muchos resultados son encontrados lo que ocasiona una gran carga para la red. [Tsoumakos 2003].

2.3.3. Búsquedas Semánticas (Basadas en Contenido)

El tipo de consulta más popular en los sistemas P2P se realiza a través de palabras clave. A este tipo de problema se le llama búsqueda basada en contenido (content-based search) [Zhou 2004], y recientemente ruteo de consultas semánticas (semantic query routing problem, SQRP) [Michmayr 2007]. Estos métodos pueden considerarse como una hibridación entre los métodos de búsqueda ciegos y los informados.

El SQRP es un método de ruteo que se enfoca en la naturaleza de la consulta más que en la topología de red. Esencialmente el SQRP mejora los métodos de ruteo tradicional priorizando los pares que han sido anteriormente buenos al proporcionar información acerca de los tipos de contenido referido con la consulta.

Para poder realizar búsquedas semánticas en redes P2P, los datos necesitan tener una descripción semántica asociada con ellos. En la Figura 2.2 se muestra un ejemplo de una consulta semántica. El ruteo de consultas semánticas difiere fundamentalmente de otras técnicas de ruteo puesto que los pares potenciales son elegidos debido a la confianza de otros pares en su habilidad para responder correctamente una consulta dada sin considerar su posición dentro de la red.

Cada vez que un par responde una consulta sus pares vecinos ajustan su valor de confianza en que ése par está relacionado con ése tipo de consulta. La naturaleza de este ajuste depende de que el par haya respondido correctamente la consulta. La información asociada con los "tipos de consulta" depende en una gran parte del tipo de dato semántico que es tratado y del método de confianza del algoritmo que realiza la clasificación.

En SQRP el problema consiste principalmente en encontrar una ruta corta entre el nodo que emite la consulta y el nodo que puede contestar esa consulta de manera apropiada.

El SQRP es, razonablemente, una nueva idea y como tal, se encuentra en su infancia con lo cual existen pocos algoritmos de búsqueda que se relacionen con este enfoque. Aún no se alcanza un consenso universal en el cual exista un método algorítmico específico de SQRP en una rama de investigación. La llegada de SQRP es un tema importante en la actualidad, y es seguro que se incremente su interés en el futuro.

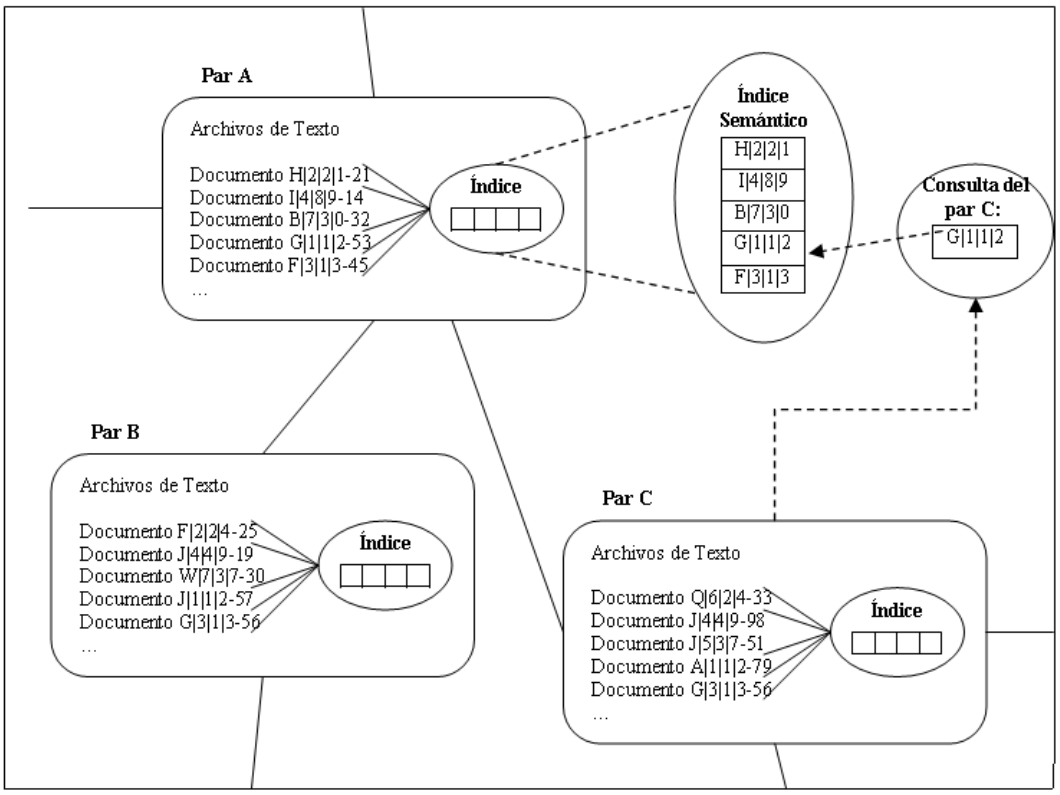

Figura 2.2. Ejemplo de una consulta semántica

2.3.4. Dificultad de la Búsqueda Semántica

En el peor caso el problema de búsqueda tiene una complejidad de $O(n^2)$ (n^2 en el caso de visitar todas las aristas del grafo) [Cormen 2001].

La justificación de aplicar algoritmos metaheurísticos para la solución de este problema no radica en la complejidad mencionada anteriormente; a pesar de que una instancia "pequeña" puede ser resuelta por un algoritmo exacto, entre más elementos interaccionen en el

problema (cantidad de nodos, enlaces y contenidos, entre otros factores dinámicos), la cantidad de recursos utilizados (tiempo de procesamiento, memoria, etc.) para resolver el problema aumenta haciendo casi imposible resolverlo.

Capítulo 3

REDES COMPLEJAS

En este capítulo se aborda la relación de los sistemas complejos con el ambiente en el cual se enfoca esta tesis, la Internet y la capa superpuesta P2P, así como también las características de los sistemas multi-agente en los cuales se basan los sistemas P2P.

Amaral en el 2004, publicó una clasificación para los diferentes tipos de sistemas y los categorizó en: sistemas simples, complicados y complejos. En el caso de los *sistemas simples* estos están formados por un pequeño número de componentes los cuales actúan de acuerdo a *leyes bien comprendidas*, después están los *sistemas complicados*, estos están formados por un gran número de componentes que tienen roles bien definidos y son gobernados por *leyes bien comprendidas* y por último los *sistemas complejos* están formados por un gran número de componentes, *que actúan de acuerdo a reglas que pueden cambiar con el tiempo* y que tal ves no sean bien comprendidas; además de que la conectividad de los componentes puede ser muy flexible y los roles pueden ser fluidos [Amaral 2004].

La teoría de grafos es una herramienta matemática poderosa y fundamental en el diseño y análisis de redes complejas, dado que la estructura topológica de una red compleja es un grafo [Xu 2001]. Un conjunto de vértices unidos por aristas forman una red simple también llamada grafo. Existen factores a través de los cuales las redes pueden llegar a ser más

complejas; por ejemplo, cuando se requiere modelar diferentes tipos de vértices o aristas los cuales pueden tener una variedad de propiedades, numéricas o de otro tipo [Newman 2003].

3.1 Tipos de Redes Complejas

Muchos sistemas del mundo real son modelados como redes complejas ya que son difíciles de describir debido a su gran tamaño y la interacción dinámica de sus elementos [Turrubiates 2007]. La Internet y la capa de red de los sistemas P2P son ejemplos de sistemas complejos que pueden ser modelados a través de la teoría de las redes complejas, ya que la red de comunicación y la administración de recursos sobre los cuales trabajan conforman el sistema complejo [Shahabi 2007].

Como se ha mencionado anteriormente, tradicionalmente el estudio de las redes complejas ha sido tratado con la teoría de grafos. La teoría de grafos inicialmente se enfocó a los grafos regulares, dado que en los años cincuenta las redes de gran escala no aparentaban tener principios de diseño y fueron descritas como grafos aleatorios, que fue la explicación más sencilla y directa de la definición de una red compleja. Esta definición continuó por décadas, pero el crecimiento del interés en redes complejas incitó a muchos científicos a reconsiderar este paradigma de modelado, con lo que se llegó a la conclusión que debido a la gran escala y dinamicidad de los elementos de estas redes, la teoría de grafos no es suficiente para describir las redes complejas, por lo que se han desarrollado diferentes modelos de redes, tales como los modelos de Erdös-Renyi [Newman 2003], Barabási-Albert [Barabási 1999] y Liu [Liu 2003].

Estos modelos permiten generar diferentes tipos de red de acuerdo a su distribución de grado, y se pueden clasificar en redes Aleatorias, Power-Law, Exponenciales y Small-world, las cuales serán tratadas a continuación.

3.1.1. Redes Aleatorias

La teoría de grafos se originó en el siglo XVIII con los trabajos de Leonard Euler acerca de la solución del problema de los puentes de Königsberg [Newman 2006]. En el siglo XX la teoría de grafos se volvió más estadística y algorítmica. Una particular fuente de ideas es el estudio de grafos aleatorios, grafos en los cuales las aristas se distribuyen aleatoriamente [Barabási 1999].

La teoría de grafos aleatorios fue publicada por Erdös y Rényi en una serie de artículos publicados a finales de los 50's [Erdős 1959] y comienzos de los 60's [Erdős 1960]. Erdös y Rényi se enfocaron a estudiar las propiedades estadísticas de grafos aleatorios con *n* nodos y *e* aristas. Erdös propuso el modelo de grafos aleatorios $G_{n,e}$ que considera a $M = \binom{n}{2}$ como el número máximo de aristas posibles entre los *n* nodos del grafo *G*. Por lo anterior, se considera a $G_{n,e}$ como el espacio de todos los $\binom{M}{e}$ grafos posibles con *n* nodos y *e* aristas. [Bollobás 2002]

En 1959, Gilbert introdujo su modelo de grafos aleatorios $G_{n,p}$ [Gilbert 1959]. La generación de grafos aleatorios se describe de la siguiente forma: Sea $\{X_{ij} : 1 \le i \le j \le n\}$ un arreglo de variables aleatorias de Bernoulli con $\Pr(X_{ij} = 1) = p$ y $\Pr(X_{ij} = 0) = 1 - p$, y sea $G_{n,p}$ un grafo con *n* nodos en el cual los nodos *i* y *j* son adyacentes, si $X_{ij} = 1$. En otras palabras, para construir un grafo aleatorio, $G_{n,p} \in G_{n,e}$, se agregan aristas con probabilidad independiente *p* [Bollobás 2002].

Más tarde a principios de los 80's diversas investigaciones arrojaron resultados más precisos, encontrando que, para un grafo aleatorio generado con una probabilidad de conexión *p* el grado k_i del nodo *i* sigue una distribución binomial con parámetros *n*-1 y *p*:

$$P(k) = \mathrm{C}_{n-1}^{k}\, p^{k} (1-p)^{n-1-k} = \binom{n-1}{k} p^{k} (1-p)^{n-1-k}. \tag{3.1}$$

Para valores grandes de *n*, la distribución del grado sigue una distribución de Poisson, de ahí que la probabilidad de que un nodo tenga grado *k* es:

$$P(k) \sim \frac{\langle k \rangle^{k}\, e^{-\langle k \rangle}}{k!}, \tag{3.2}$$

donde $\langle k \rangle = 2e / n = p(n-1)$ [Albert 2002, Barabási 2003, Newman 2003].

3.1.2. Redes Power-Law

En la década de los 90's diversos investigadores como [Albert 2000, Faloutsos 1999], descubrieron que la distribución del grado en las redes del mundo real como el World Wide Web, Internet, redes de proteínas y metabolismo, las redes de lenguaje y sociales, difieren a la distribución de Poisson, exhibiendo una distribución del grado Power-Law:

$$P(k) \sim k^{-\gamma}. \tag{3.3}$$

Las redes con distribución Power-Law son llamadas *Scale-Free* o *Libres de Escala*, debido a que independientemente de la escala, es decir del número de nodos, la distribución del grado no cambia. La distribución Zipf presenta las mismas características que la distribución Power-Law. La característica invariante de estas redes es que un conjunto reducido de nodos que conforman la red tienen un gran número de enlaces (un grado muy alto) y el resto de los nodos tienen pocos enlaces (un grado pequeño) [Barabási 2003, Newman 2003]. Esta característica permite que la tolerancia a fallas aleatorias sea grande, pero si los nodos con grado muy alto llamados nodos centrales son atacados intencionalmente esta red es muy vulnerable [Albert 2000].

En esta distribución el valor del parámetro γ se reduce desde ∞ hasta 0, siendo un parámetro de control que describe que tan rápido decae la frecuencia de aparición del grado *k*,

de manera que el grado promedio de la red se incrementa a medida que el valor de γ se reduce. En este caso $P(k)$ no tiene un pico definido, y para una *k* grande decae como una serie de potencias, apareciendo como una línea recta en una gráfica log-log.

3.1.3. Exponenciales

Aunque las redes Power-Law son comunes, también existen casos de redes que exhiben una distribución del grado exponencial como la red de energía del sur de California [Amaral 2000] y la red de vías de ferrocarril de la India [Sen 2003]. A estas redes se les llama redes exponenciales.

En estas redes la distribución del grado $P(k)$ muestra un pico en $\langle k \rangle$ y después cae exponencialmente para valores grandes de *k*:

$$P(k) \sim e^{-k}. \qquad (3.4)$$

Este tipo de red, no es muy tolerante a fallas aleatorias pues al eliminar cualquier nodo el daño es similar, debido a que cada nodo en la red se relaciona aproximadamente con la misma cantidad de nodos [Albert 2000].

3.1.4. Small World

El concepto "small world" es un término que describe el hecho de que a pesar de que las redes complejas tienen gran tamaño, en la mayoría de las redes la distancia entre dos nodos es relativamente corta. La distancia entre dos nodos esta definida como el número de nodos a lo largo de la ruta que los conecta.

La manifestación más popular de "small world" es el concepto de "seis grados de separación" descubierto por Stanley Milgram en 1967 quien concluyó que la ruta de conocidos entre pares de personas en los Estados Unidos tiene una distancia típica de seis. Esta propiedad

de “small world” aparece caracterizando a la mayoría de las redes complejas. [Albert 2002]. El concepto “small world” no es indicador de un principio organizacional particular, sino un fenómeno que puede darse en las redes complejas.

La naturaleza de las redes complejas, los cambios dinámicos de sus componentes tales como la topología, el contenido de los repositorios y la generación de consultas, hacen imposible caracterizar de manera global a este tipo de redes, por lo cual es necesario el desarrollo de modelos que reproduzcan redes reales y funciones de caracterización que obtengan en términos cuantitativos las características topológicas y de contenido de manera local. De esta manera sería posible comprender la dinámica y la estabilidad de las redes complejas de gran escala [Barabási 2003].

3.2. Sistemas Multi-agente para Redes Complejas

Recientemente se ha vuelto popular el estudio de la inteligencia que surge de las interacciones entre muchos agentes, conocidos como sistemas multiagente. Los sistemas multi-agente son complejos en naturaleza: estos son típicamente caracterizados por un gran número de partes que tienen muchas interacciones. Además esta complejidad no es accidental: esta es una propiedad innata de los tipos de tareas para las cuales estos sistemas son utilizados [Jennings 1999].

Los desarrollos sobre redes de gran escala de telecomunicaciones y sistemas distribuidos han visto incrementado su interés con los sistemas multi-agente [Jennings 1999]. La propia definición de agentes software ya nos indica que los agentes tienen características parecidas a la de los sistemas P2P, como son su funcionamiento de forma distribuida y autónoma. La extensión de los sistemas multi-agente puede ser facilitada si la tecnología de los sistemas de agentes pudiese interoperar con servicios P2P estándares [Mondéjar 2006].

Los agentes son los elementos principales que conforman un sistema multi-agente. Podemos definir un agente como [Wooldridge 2002]: “Un agente inteligente es un proceso

computacional capaz de realizar tareas de forma autónoma y que se comunica con otros agentes para resolver problemas mediante cooperación, coordinación y negociación. Habitan en un entorno complejo y dinámico con el cual interaccionan en tiempo real para conseguir un conjunto de objetivos". Las propiedades indispensables de un agente son:

a) *Autonomía*: es la capacidad de operar sin la intervención directa de los humanos, y de tener algún tipo de control sobre las propias acciones y el estado interno.
b) *Sociabilidad/Cooperación*: los agentes han de ser capaces de interactuar con otros agentes a través de algún tipo de lenguaje de comunicación.
c) *Reactividad*: los agentes perciben su entorno y responden en un tiempo razonable a los cambios detectados.
d) *Pro-actividad o iniciativa*: deben ser capaces de mostrar que pueden tomar la iniciativa en ciertos momentos.

Otras propiedades destacables serían:

a) *Movilidad*: posibilidad de moverse a otros entornos a través de una red electrónica.
b) *Continuidad temporal*: los agentes están continuamente ejecutando procesos.
c) *Veracidad*: un agente no comunicará información falsa premeditadamente.
d) *Benevolencia*: es la propiedad que indica que un agente no tendrá objetivos conflictivos, y que cada agente intentará hacer lo que se le pide.
e) *Racionalidad*: el agente ha de actuar para conseguir su objetivo.
f) *Aprendizaje*: mejoran su comportamiento con el tiempo.
g) *Inteligencia*: usan técnicas de IA para resolver los problemas y conseguir sus objetivos.

Una vez definidas las características de los agentes, podemos definir un sistema multi-agente (MAS) como aquél en el que un conjunto de agentes cooperan, coordinan y se comunican para conseguir un objetivo común. Las principales ventajas de la utilización de un sistema multi-agente son [Mondéjar 2006]:

a) *Modularidad***:** se reduce la complejidad de la programación al trabajar con unidades más pequeñas, que permiten una programación más estructurada.
b) *Eficiencia***:** la programación distribuida permite repartir las tareas entre los agentes, consiguiendo paralelismo (agentes trabajando en diferentes máquinas).
c) *Fiabilidad***:** el hecho de que un elemento del sistema deje de funcionar no tiene que significar que el resto también lo hagan; además, se puede conseguir más seguridad replicando servicios críticos y así conseguir redundancia.
d) *Flexibilidad***:** se pueden añadir y eliminar agentes dinámicamente.

En el estudio de los sistemas multi-agente, la estructura de red de los agentes juega un rol importante. Las interacciones entre todos los tipos de agentes son usualmente estructuradas con redes complejas. La idea de combinar los sistemas multi-agente y las redes complejas es un elemento esencial para dar solución al problema de SQRP, ya que la teoría de redes complejas permite modelar los elementos dinámicos y las interacciones del sistema multi-agente lo cual es importante para comprender y formar el comportamiento inteligente deseado sobre los elementos dinámicos de la red para cumplir con los objetivos del sistema.

Capítulo 4

ALGORITMOS DE COLONIA DE HORMIGAS

Este capítulo introduce la descripción y propiedades de dos clases de algoritmos de colonia de hormigas que han sido enfocados a diversos problemas con los cuales ha obtenido un buen desempeño. También se evalúa la aplicabilidad de éstos algoritmos elegidos para el problema de búsqueda en redes P2P.

Desde la década de los 80's, los fenómenos naturales que son simulados son llamados sistemas complejos y su estudio es llamado ciencia de la complejidad [Michlmayr 2007]. Ejemplos de sistemas complejos son la colonia de hormigas, organismos, ecologías y sociedades.

Los algoritmos de colonia de hormigas fueron elegidos como método de solución en esta tesis debido a que presentan las mismas propiedades que los sistemas complejos (que en este trabajo son las redes P2P) tales como la emergencia y auto-adaptación que son necesarias para que el algoritmo de búsqueda, pueda aprender y adaptarse al ambiente de búsqueda [De Wolf 2005]. En la siguiente sección se describen los algoritmos de colonia de hormigas.

4.1. Métodos Basados en Colonia de Hormigas

Los algoritmos de hormigas están inspirados en el comportamiento colectivo de la búsqueda de comida de ciertas especies de hormigas, en donde las hormigas se comunican de manera indirecta con otros miembros de su colonia de hormigas. La comunicación esta basada en la modificación del ambiente local depositando una sustancia química llamada *feromona*. En la búsqueda de comida, algunas especies usan el comportamiento llamado "dejar-rastro" y "seguir-rastro" para encontrar la ruta más corta entre el nido y la fuente de comida [Grassé 1959].

Las colonias de hormigas artificiales son sistemas multi-agente en el sentido que cada hormiga-agente tiene información o capacidades incompletas para resolver el problema, ya que no existe un sistema de control global, los datos son descentralizados, y la computación es asíncrona [Sycara 1998]. Además, las colonias de hormigas exhiben un comportamiento emergente debido a que construyen sus resultados de manera incremental. Por lo tanto, las colonias de hormigas son sistemas adaptativos complejos.

Existen varios algoritmos que modelan y explotan el comportamiento anteriormente mencionado para resolver problemas de optimización combinatoria NP-Hard basados en grafos. Éstos han sido extensamente aplicados al problema del Agente Viajero (Traveling Salesman Problem, TSP [Lawler 1985]).

Ant Colony Optimization, es una metodología basada en el comportamiento de las hormigas, y comprende diferentes tipos de sistemas de hormigas [Dorigo 2004]. En esta sección se ha elegido describir el algoritmo Ant Colony System [Dorigo 1997], que es uno de los algoritmos de hormigas con mejor rendimiento y más referenciados [Asmar 2005]. Sobre este algoritmo se realizan adaptaciones para el problema de enrutamiento de consultas semánticas que se aborda en esta tesis.

Ant Colony System (ACS)

ACS es un algoritmo en el cual un conjunto de agentes llamados hormigas, cooperan entre sí para hallar buenas soluciones a problemas de tipo TSP (Travelling Salesman Problem) y VRP (Vehicle Routing Problem). Las hormigas cooperan usando una forma de comunicación indirecta mediante feromonas que son depositadas en las aristas de un grafo mientras van construyendo las soluciones [Dorigo 1997].

De manera general el algoritmo funciona de la siguiente forma: en la Fase 1, se inicializan los parámetros que regulan el comportamiento del algoritmo; en la Fase 2, se van construyendo recorridos factibles aplicando una regla estocástica voraz (la regla de transición de estado), mientras se construyen los recorridos, las hormigas modifican la cantidad de feromonas sobre las aristas visitadas aplicando la regla de actualización local. Una vez que todas las hormigas han terminado de construir sus recorridos se realiza la Fase 3, donde todas las aristas que pertenecen al mejor recorrido generado son actualizadas nuevamente aplicando la regla de actualización global. El fin de este proceso iterativo se lleva a cabo en la Fase 4, donde se retorna la mejor solución encontrada. En la Tabla 4.1 se muestra de manera general el algoritmo ACS.

Tabla 4.1. Algoritmo ACS

```
Inicializar Parámetros                                  // Fase 1
Para cada Iteración                                     // Fase 2
    Para cada Hormiga
        Construye solución con la hormiga k
        Actualización LOCAL de feromona
    Guardar el mejor recorrido
    Actualización GLOBAL de feromona;                   // Fase 3
    Posicionar las hormigas en la ciudad inicial
Retornar la mejor solución encontrada                   // Fase 4
```

La Fase 1 se muestra en la Tabla 4.2, aquí se inicializan: la tabla de feromonas τ con el valor de τ_0, la cantidad total de ciudades n, la lista J_k de ciudades no visitadas por la hormiga k (líneas 01 y 02 de la Tabla 4.2) y la lista de ciudades donde cada una de las hormigas esta posicionada r_k, ésta asignación puede ser de manera aleatoria (líneas 03–06 de la Tabla 4.2).

Tabla 4.2. Fase de inicialización del algoritmo ACS

```
01  Para cada par (r,s)
02      τ (r, s) = τ0
03  Para cada hormiga k
04          Seleccionar rk1 como la ciudad inicial para la hormiga k
05          Jk(rk1) = {1…n}-rk1
06          rk = rk1
```

La Fase 2, mostrada en la Tabla 4.3 corresponde a la parte de construcción de la solución en donde cada hormiga va construyendo paso a paso la solución mientras va seleccionando la siguiente ciudad a la cual ir (de acuerdo a las Ecuaciones 4.1 y 4.2 que se mencionan mas adelante), se elimina de la lista de ciudades no visitadas y se añade al recorrido creado por la hormiga *k* (líneas 02-06 de la Tabla 4.3). Cuando se ha llegado a la última ciudad seleccionada esta se conecta con la ciudad inicial para cerrar el circuito (líneas 07-10 de la Tabla 4.3). Durante cada paso se actualiza de manera local la tabla de feromonas (líneas 11-13 de la Tabla 4.3) seleccionando la arista donde la hormiga eligió pasar (de acuerdo a la Ecuación 4.4).

Tabla 4.3. Fase de construcción del algoritmo ACS

```
01  Para i = 1 hasta el total de ciudades
02      Si i < total de ciudades n entonces
03          Para cada hormiga
04              Elegir la siguiente ciudad sk  // De acuerdo a las ec. 4.1 y 4.2
05              Jk(sk) = Jk(rk) – sk           // Quitar la ciudad visitada
06              recorridok(i) = (rk, sk)       // Añadir al recorrido
07      De lo contrario
08          Para cada hormiga
09              sk = rk1                       // La siguiente ciudad es la inicial
10              recorridok(i) = (rk, sk)       // Añadir sk al recorrido
11      Para cada hormiga
12          τ (rk, sk) = (1-ρ)τ (rk, sk) + ρτ0 // Actualizar localmente el rastro
13          rk = sk                            // sk es ahora la ciudad actual
```

La Fase 3, mostrada en la Tabla 4.4, corresponde a la actualización global, una vez que se han construido las soluciones de las hormigas en L_k se elige el recorrido que tenga la longitud más corta en L_{best} (líneas 01-03 de la Tabla 4.4), con este recorrido se actualiza la tabla de feromonas (de acuerdo a la Ecuación 4.3).

Tabla 4.4. Fase de actualización global del algoritmo ACS

01 **Para** cada hormiga
02 Calcular la longitud de los recorrido generados por cada hormiga en L_k
03 Obtener el recorrido L_k con la menor longitud en L_{best}
04 **Para** cada arista (r,s)
05 $\tau(r_k,s_k) = (1-\alpha)\tau(r_k, s_k) + \alpha(L_{best})^{-1}$ // Actualizar las aristas de L_{best} con la ec. 4.3

En la Tabla 4.5, se muestra la última Fase, en donde se comprueba que se cumpla la condición de terminación y se despliega el mejor resultado obtenido (líneas 01 y 02 de la Tabla 4.5). En el caso de no cumplirse la condición de terminación, se continúa realizando la Fase 2 (líneas 03 y 04 de la Tabla 4.5).

Tabla 4.5. Fase de terminación del algoritmo ACS

01 **Si** se cumple la condición de terminación entonces
02 Imprimir la ruta mas corta de L_k
03 **De lo contrario**
04 Ir a la Fase 2

Dentro del algoritmo se usan un conjunto de Ecuaciones, a través de las cuales se va guiando al algoritmo hacia la estimación de la mejor posible solución. Dichas ecuaciones se analizarán a continuación:

Regla Aleatoria Proporcional

La regla aleatoria proporcional de la Ecuación 4.1, define la probabilidad p_k con la cual una hormiga k elige moverse a la ciudad s. S es una variable aleatoria seleccionada de acuerdo con la probabilidad p_k

$$S = f(p_k(r,s)),\ p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)]\cdot[\eta(r,s)]^{\beta}}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^{\beta}}, & \text{si } s \in J_k(r) \\ 0, & \text{en otro caso,} \end{cases} \tag{4.1}$$

donde τ es la feromona, $\eta = 1/\delta$ es el inverso de la distancia desde la ciudad r a la ciudad s $\delta(r, s)$, $J_k(r)$ es el conjunto de ciudades a visitar por la hormiga k ubicada en la ciudad r y β es un parámetro que determina la importancia relativa entre la feromona y la distancia ($\beta > 0$). La

influencia del parámetro β al ser aplicado al valor de la distancia en $\eta(r,u)$ radica en que al incrementar el valor de β implica dar mayor importancia a través del tiempo en las distancias cortas que al valor almacenado en la feromona en $\tau(r,u)$.

Regla de transición de estado

La regla de transición de estado, donde una hormiga posicionada en un nodo *r* elige la ciudad *s* para moverse a ella aplicando la siguiente regla:

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\}, & \text{Si } q \le q_0 \text{ (explotación)} \\ S, \text{ en otro caso} & \text{(exploración parcial),} \end{cases} \tag{4.2}$$

donde *q,* es una variable aleatoria que representa un número uniforme distribuido entre [0-1], q_0 es un parámetro ($0 \le q_0 \le 1$) y *S* es una variable aleatoria seleccionada de acuerdo a la Ecuación 4.1. Esta regla, favorece las transiciones hacia nodos conectados mediante aristas cortas y con gran cantidad de feromona. El parámetro q_0 determina la importancia relativa entre la explotación y la exploración.

Regla de actualización global

La regla de actualización global, se realiza después de que todas las hormigas han construido sus recorridos, solo la mejor hormiga global (la hormiga que construyó el recorrido más corto desde el principio del proceso) es la que deposita feromona. Ésta elección junto al uso de la Ecuación 4.2, se hace con la intención de que la búsqueda sea dirigida. El nivel de feromona se actualizará mediante la siguiente regla:

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s), \tag{4.3}$$

donde

$$\Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1}, & \text{if } (r,s) \in \text{ mejor-recorrido global} \\ 0, & \text{en otro caso.} \end{cases}$$

Además α ($0 < \alpha < 1$), es el parámetro de evaporación global de la feromona, y L_{gb} es la longitud del mejor recorrido global desde el principio del proceso. Con esta actualización se intenta proveer de un gran monto de feromona a los recorridos cortos. Ésta ecuación dicta que sólo aquellas aristas que pertenecen al mejor recorrido global recibirán recompensa.

Regla de actualización local

La regla de actualización local, mientras se construye una solución, las hormigas recorren aristas y cambian su nivel de feromona aplicando actualización local con la Ecuación 4.4.

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \rho\cdot\Delta\tau(r,s), \qquad (4.4)$$

donde $0 < \rho < 1$, es un parámetro que define la tasa de evaporación local de la feromona y $\Delta\tau(r,s) = \tau_0$.

4.2. Métodos Distribuidos Basados en Colonias de Hormigas

Los primeros algoritmos de hormigas fueron diseñados para resolver problemas de optimización basados en grafos, los cuales utilizan una tabla de feromonas administrada globalmente. En la actualidad se ha incrementado la implementación de estos algoritmos en problemas distribuidos tales como el enrutamiento de paquetes en redes, en donde la información se administra de manera local. Los algoritmos de hormigas distribuidos más utilizados son:

a) AntNet. Este método fue publicado en 1998 por Di Caro y Dorigo [Di Caro 1998]. Fue diseñado para redes de intercambio de paquetes y su método de actualización de feromona es apropiado para redes simétricas y asimétricas. AntNet es el algoritmo distribuido más prominente.

b) AntHocNet. Este algoritmo fue publicado en 2004 por Di Caro, Ducatelle, y Gambardella [Di Caro 2004]. Fue diseñado para redes móviles ad-hoc (MANETS).

En esta sección sólo se describirá a grandes rasgos el algoritmo AntNet, que es el algoritmo que tiene características similares al problema que se aborda en esta tesis.

AntNet

En AntNet, las hormigas colaboran en la construcción de tablas de enrutamiento de paquetes de datos en una red de comunicaciones con el objetivo de optimizar el desempeño del enrutamiento en toda la red. El algoritmo es distribuido y adaptativo en el sentido que adapta la política de enrutamiento a las condiciones de tráfico actual en la red. El algoritmo de AntNet se muestra en la Tabla 4.6.

De forma general el algoritmo AntNet funciona de la siguiente manera: en la Fase 1, se inicializan los parámetros que regulan el comportamiento del algoritmo; en la Fase 2, cada nodo se administra de manera automática obteniendo los parámetros de configuración con los cuales se generan las hormigas de envío, durante este proceso de manera paralela ocurre la Fase 3, en donde se van construyendo rutas factibles aplicando una regla estocástica voraz (la regla de transición, Ecuación 4.5). Una vez que la hormiga de envío ha conseguido llegar al nodo destino se realiza la Fase 4, donde todos parámetros de los nodos que pertenecen a la ruta generada son actualizados (con las Ecuación 4.6 y 4.7).

Tabla 4.6. Algoritmo AntNet

```
Inicializar Parámetros                                  // Fase 1
Para cada Nodo  en paralelo                             // Fase 2
        Obtener parámetros
        Para cada Hormiga de envío en paralelo          // Fase 3
                Construye solución
                Si se encontró el destino
                        Para cada Hormiga de retroceso en paralelo   // Fase 4
                                Actualizar Parámetros
```

La Fase 1, mostrada en la Tabla 4.7, corresponde a la fase de inicialización, donde se definen los parámetros de control del algoritmo, los cuales son: el tiempo actual de la

simulación *t*, el tiempo total de la simulación *tend*, y el intervalo de tiempo entre la generación de hormigas Δt.

Tabla 4.7. Fase de inicialización del algoritmo AntNet

```
Inicializar
        T = tiempo actual
        tend = Longitud del tiempo de simulación
        Δt = Intervalo de tiempo entre la generación de hormigas
```

En la Tabla 4.8 se muestra la Fase 2, correspondiente a la fase de generación, en esta fase cada nodo de manera paralela obtiene un modelo del tráfico local de la red M (que corresponde a estructuras de datos que contienen simples factores de tráfico, tales como el mejor tiempo de llegada hacia un nodo, varianzas y desviaciones estándar de tiempos de llegada, etc.) y la tabla de enrutamiento τ de cada nodo (líneas 01-03 de la Tabla 4.8). Una vez que se obtiene esta información, mientras el tiempo de simulación no se haya agotado, cada nodo elige un nodo destino al cual deben llegar las hormigas de envío generadas por dicho nodo (líneas 04-08 de la Tabla 4.8), este el proceso que lleva a cabo la hormiga de envío de manera paralela, se describe en la Tabla 4.8.

Tabla 4.8. Fase de generación del algoritmo AntNet

```
01  Para cada Nodo                                          //Actividad concurrente sobre la red
02      Obtener el Modelo de tráfico local, M
03      Obtener la Tabla de enrutamiento del nodo, τ
04      Mientras t <= tend
05          En_paralelo                                      //Actividad concurrente en el nodo
06          Si t mod Δt == 0                                 // Si es intervalo de lanzamiento
07              Nodo destino = eligeNodoDestino(distribución_trafico_datos) //Elige el nodo destino
08              lanzarHormigaDeEnvio(nodo destino, nodo fuente)   //Genera hormiga de envío
09          Fin_en_paralelo
```

La Fase 3, corresponde a la etapa de construcción, este proceso se realiza de manera paralela cuando ha sido creada una hormiga de envío. Durante este proceso cada hormiga de envío se encarga de construir una ruta hacia un nodo destino eligiendo el siguiente nodo a moverse mediante la Ecuación 4.5, una vez que ha decidido moverse a un nodo, la hormiga almacena ese movimiento en una pila que almacena los enlaces visitados y el tiempo transcurrido hasta ese momento, posteriormente se mueve hacia el siguiente nodo (líneas 02-08 de la Tabla 4.9). Este proceso termina cuando ha sido encontrado el nodo destino, con lo

cual se crea una hormiga de retroceso con los datos de la hormiga de envío (nodo destino, nodo fuente y la ruta generada) (líneas 09-10 de la Tabla 4.9), posteriormente se elimina la hormiga de envío. En la Tabla 4.10 se describe el procedimiento que lleva a cabo la hormiga de retroceso.

Tabla 4.9. Fase de construcción del algoritmo AntNet

```
01  Para cada hormigaDeEnvioActiva(nodo fuente, nodo actual, nodo destino) // Actividad concurrente
02      Mientras nodo actual != nodo destino
03          Siguiente_nodo= eligeEnlace(nodo actual, nodo destino, τ, cola de enlaces) // Mediante ec. 3.5
04          ponerHormigaEnColadeEnlaces(nodo actual, siguiente nodo)
05          esperarDatoEnColaDeEnlaces(nodo actual, siguiente nodo)
06          cruzarElEnlace(nodo actual, siguiente nodo)
07          ponerEnLaPila(siguiente nodo, tiempo transcurrido)
08          nodo actual = siguiente nodo
09      lanzarHormigaDeRetroceso(nodo destino, nodo fuente, pila de datos)      // Actividad concurrente
10      eliminarHormigaDeEnvio
```

Por último, la Fase 4, corresponde a la Fase de actualización, la cual es llevada a cabo por la hormiga de retroceso. Durante este proceso cada hormiga de retroceso intenta llegar al nodo que emitió la búsqueda. Mientras el nodo que este evaluando en ese momento, no sea el nodo fuente, la hormiga selecciona el nodo que se encuentra en la pila de nodos visitados, y se mueve hacia ese nodo (líneas 03-05 de la Tabla 4.10). Una vez que ha realizado el movimiento hacia un nodo de la ruta, se actualiza el modelo de tráfico local del nodo evaluado y se actualiza la tabla de enrutamiento con las Ecuaciones 4.6 y 4.7.

Tabla 4.10. Fase de actualización del Algoritmo AntNet

```
01  Para cada hormigaDeRetrocesoActiva(nodo fuente, nodo actual, nodo destino)
02      Mientras nodo actual != nodo fuente
03          Siguiente nodo = sacarDeLaPila()
04          esperarSobreLaColaDeEnlacesDeAltaPrioridad(nodo actual, siguiente nodo)
05          cruzarElEnlace(nodo actual, siguiente nodo)
06          actualizarModeloDeTraficoLocal(M, nodo actual, nodo fuente, pila de datos)
07          Reforzamiento = obtenerReforzamiento(nodo actual, nodo fuente, pila de datos, M)
08          actualizarTablaDeEnrutamientoLocal(τ , nodo actual, nodo fuente, reforzamiento)
```

Las estructuras de datos manejadas en este algoritmo están en función de los elementos que componen la red. La red es mapeada sobre un grafo dirigido con pesos y N cantidad de nodos. Las aristas del grafo son enlaces entre los nodos. Los enlaces son vistos como tuberías de bits con ciertos costos (ancho de banda, medido en bits por segundo, y retraso en la

transmisión, medido en segundos) que dependen de la carga actual de los enlaces. Cada nodo *n* administra una tabla de enrutamiento τ_n, que almacena información acerca de los enlaces salientes y su cantidad de feromona. Las tablas de enrutamiento son matrices de tamaño *N* x *l*, donde *N* es el número de nodos en la red y *l* es el número de enlaces salientes. Cada entrada P_{nd}, de estas tablas es un valor probabilístico indicando la bondad de reelegir el enlace hacia el nodo vecino *n,* con respecto a un nodo destino dado *d*.

En AntNet existen dos tipos de agentes hormiga, que realizan las tareas de actualización en la tabla de feromonas: las hormigas de envío (Tabla 4.9) y las hormigas de retroceso (Tabla 4.10).

Las hormigas de envío son generadas en intervalos aleatorios en cada nodo fuente *s*, éstas construyen una ruta a un nodo destino *d,* seleccionado aleatoriamente mediante la regla de transición mostrada en la Ecuación (4.5). Cada vez que ésta llegue a un nodo *k* y tenga que seleccionar un enlace de salida. La regla de transición decide hacia que nodo avanzar basándose en la entrada P_{nd} encontrada en la tabla de enrutamiento e incluye el factor $l_n \in [0,1]$. Para cada enlace de salida *n*, el factor l_n es proporcional a la longitud actual de la cola del enlace hacia *n* en términos de bits esperando a ser enviados. El peso $\alpha \in [0,1]$ define la influencia del factor de longitud de la cola;

$$P'_{nd} = \frac{P_{nd} + \alpha l_n}{1 + \alpha\left(\left|vecinos(k) - 1\right|\right)}. \qquad (4.5)$$

Cuando una hormiga de envío ha alcanzado su nodo destino, ésta calcula el tiempo total del viaje *T* de la ruta usada. El valor *T,* indica la bondad de la ruta encontrada. Ya que cada nodo almacena información de enrutamiento localmente, la hormiga-de-envío no puede actualizar los valores de bondad en las tablas de enrutamiento directamente. En lugar de eso, se genera una hormiga-de-retroceso que retorna al nodo fuente a través de la misma ruta que fue utilizada por la hormiga de envío. Después de crear la hormiga de retroceso y pasarle una

copia de la pila de datos, almacenada mientras se viajaba al nodo destino, la hormiga de envío finaliza.

Las hormigas de retroceso son responsables de actualizar los valores de bondad y la información estadística de acuerdo a la información obtenida por la hormiga de envío al alterar las estructuras de datos τ y M, en cada nodo visitado. La hormiga de retroceso actualiza solo aquellas entradas que se refieren al nodo destino. Subsecuentemente las hormigas de envío toman sus decisiones de enrutamiento basadas en los valores alterados. En AntNet, la suma de todas las probabilidades que se refieren a un nodo destino siempre es 1. Esto es alcanzado al decrecer el valor de probabilidad para la ruta que fue usada, y al decrecer los otros valores para el nodo destino, como se muestra en las Ecuaciones 4.6 y 4.7. En estas ecuaciones, la hormiga de retroceso llega desde un nodo *x* a un nodo *k* y actualiza τ_k. La cantidad de feromona es determinada por el factor $r \in [0,1]$ que es derivado al comparar el tiempo de viaje *T* hacia el mejor tiempo de viaje W_{best_d}. Ya que la bondad de un tiempo de viaje es relativo a la carga actual de la red, la comparación toma la media y la varianza estimadas dentro de la cuenta.

$$P_{xd} \leftarrow P_{xd} + r\left(1 - P_{xd}\right), \tag{4.6}$$

$$P_{xd} \leftarrow P_{nd} - rP_{xd},\ n \in vecinos\left(k\right),\ n \neq x \tag{4.7}$$

Después de actualizar τ_k, la hormiga de retroceso también actualiza la estructura de datos M_k, con el tiempo de viaje almacenado desde el nodo *k* hacia el nodo *d*. La media y la varianza estimadas son actualizadas con los nuevos valores. Si el nuevo resultado es el mejor de los *w* ejemplos, este es almacenado en W_{best_d}.

Otra característica de éste algoritmo es que incluye una estrategia de prevención de ciclos. Cada hormiga de envío administra una pila de nodos anteriormente visitados. Cada vez que tiene que decidir a cual nodo siguiente visitar, esta excluye a todos los nodos visitados. Si esta detecta un ciclo debido a que es forzada a ir a un nodo anteriormente visitado *v*, debido

que todos los siguientes nodos posibles habían sido anteriormente visitados, ésta calcula el tiempo *t* gastado dentro del circuito. Si *t* es más grande que el 50% del tiempo total de la hormiga, esta es finalizada. De otra manera, la hormiga remueve todos los nodos que son parte del ciclo desde su pila y continúa viajando en el nodo *v*.

4.3. Comentarios Finales

Como se ha mencionado anteriormente, los sistemas de colonias de hormigas poseen características de los sistemas complejos (tales como emergencia y auto-adaptación), y debido a que los sistemas P2P se consideran complejos se ha llegado a la conclusión de que puede existir una adaptación de los sistemas de colonias de hormigas a los sistemas de enrutamiento de consultas en redes P2P. En esta sección se discute la aplicabilidad de los sistemas de hormigas anteriormente mencionados con las redes P2P: Algoritmos *ACS* y *AntNet.*

ACS fue diseñado originalmente para resolver problemas de optimización combinatoria para grafos (TSP, VRP, etc.) y, de la misma manera, SQRP puede considerarse como un problema de optimización combinatoria para grafos. Las diferencias entre estas dos perspectivas son las siguientes:

a) El grafo contemplado en el problema del TSP es completo, esto es, el objetivo es encontrar un circuito que pase por todas las ciudades y que tenga la menor distancia; en el caso del problema de enrutamiento de consultas se contempla una red compleja en donde el objetivo es encontrar la ruta mas corta que lleve a una mayor cantidad de recursos encontrados.

b) En el problema de TSP se manejan las distancias geográficas, en el caso del problema de enrutamiento de consultas que se plantea, las distancias se pueden contemplar como la cantidad de saltos hacia los recursos buscados.

c) Las estructuras de datos que son usadas en ACS, para el TSP consisten de una tabla de feromonas global que contiene un solo tipo de feromona (es decir solo se realiza aprendizaje sobre un solo objetivo), en el caso del problema de enrutamiento de consultas planteado, al ser éste un problema enfocado a redes P2P, la tabla de feromonas necesita ser administrada de manera local y distribuida [Michlmayr 2007].

Por otra parte, el algoritmo AntNet fue inventado para redes de intercambio de paquetes, administrado de manera distribuida, tal y como son administrados los sistemas P2P. Aunque las características de estos sistemas distribuidos son similares, existen las siguientes diferencias:

a) En el problema de intercambio de paquetes todos los nodos son conocidos, caso contrario del problema de enrutamiento de consultas planteado en esta tesis, en donde cada nodo solo necesita conocer a sus vecinos inmediatos.
b) La tabla de feromonas administrada en el intercambio de paquetes tiene diferentes tipos de feromona, pero solo enfocados al enrutamiento de nodos, en el caso del enrutamiento de consultas, además de necesitarse el enrutamiento de los nodos, es necesario guiar las consultas hacia los recursos buscados, creando con esto que las tablas de feromonas tengan más dimensiones [Michlmayr 2007].
c) Las funciones de selección utilizadas por AntNet se enfocan a la carga de los enlaces, en el caso del problema de enrutamiento planteado esa característica no esta contemplada.
d) La función de actualización obliga a que la suma de todos los valores de bondad que corresponden a un nodo destino sea 1, caso que no es necesario en el problema planteado [Michlmayr 2007].

Las características que si pueden ser tomadas en cuenta de los algoritmos anteriormente descritos son las siguientes:

a) La función de selección del algoritmo ACS es la más flexible, debido a que cuenta con parámetros que pueden ser adaptados con facilidad al problema de enrutamiento.
b) Las funciones de actualización local y global de ACS son las más adecuadas, debido a que los parámetros permiten ser adaptados para justificar los valores de bondad depositados en la tabla de feromonas, así como la facilidad de administrar de manera automática los niveles de feromona, permitiendo la convergencia del algoritmo.
c) Las estructuras de datos de AntNet pueden ser adaptadas al problema de enrutamiento agregando una dimensión a la tabla de feromonas para asignar referencias de rutas anteriormente visitadas de los nodos vecinos hacia recursos buscados anteriormente [Michlmayr 2007].
d) Para incrementar la eficiencia del sistema distribuido, el concepto de hormiga de envío y de retroceso de AntNet puede ser adaptado de la misma manera para realizar selección del siguiente nodo y la actualización global de la tabla de feromonas de manera simultánea [Michlmayr 2007].
e) El concepto de prevención de ciclos de AntNet puede ser adaptada sin hacer ningún cambio, de tal manera que al no existir un nodo vecino al cual ir, retorne al nodo anterior y busque una nueva ruta [Michlmayr 2007].

Capítulo 5

ESTADO DEL ARTE SOBRE CONSULTAS EN REDES P2P

En este capítulo se presenta el estado del arte de trabajos relacionados con consultas semánticas y trabajos que aplican la teoría de los algoritmos de hormigas a esta tarea. El capítulo finaliza con una discusión acerca de estos trabajos.

5.1 Trabajos Relacionados con consultas semánticas

Existen pocos trabajos relacionados con enrutamiento de consultas semánticas, también conocido como búsquedas basadas en contenido [Zhou 2004]. Dentro de estos trabajos existen diferentes tipos de búsquedas de contenido tales como: imágenes, palabras clave, texto libre y otras realizadas sobre documentos de tipo XML [Bray 2000] y RDF [Lassila 1999]. Estos últimos son documentos estructurados que contienen meta-etiquetas las cuales proporcionan información acerca de ciertos contenidos dentro del documento. A continuación se describen algunos de los trabajos más relevantes en esta línea de investigación, y un resumen de ellos se presenta en la Tabla 5.1.

Chrysakis [Chrysakis 2006] adopta una arquitectura basada en super-pares (super-peers [Yang 2003]) y sugiere un mecanismo de enrutamiento de consultas en la cual se propone una técnica de procesamiento de consultas de tipo top-*k*. En el contexto de los sistemas P2P, las

consultas top-*k* dan la oportunidad de filtrar los resultados y eliminar el tráfico de red, escogiendo los *k* resultados clasificados como más altos. También introduce dos versiones del algoritmo Hybrid Threshold [Yu 2005] adaptado a su escenario P2P no estructurado.

El algoritmo de ruteo funciona a través de super-pares; el super-par examina los esquemas RDF o XML de cada par que esta en su grupo y encuentra los pares que son relevantes de acuerdo al recurso solicitado, si el super-par encuentra pares relevantes en diferentes super-pares, éste los añade a un conjunto de nodos relevantes que esta relacionado con la consulta y los devuelve al par que emitió la consulta.

En este trabajo no se consideran condiciones acerca de la topología de la red, pero si de la distribución de contenido, la cual es de dos tipos: uniforme y zipfiana, comúnmente denominada distribución Zipf, esta distribución tiene las características de la distribución Power-Law, también conocida como Scale-free, descrita en el Tema 3.1.

Kokkinidis [Kokkinidis 2004] presenta el algoritmo ICS-FORTH SQPeer para procesamiento de consultas realizadas en los lenguajes de consulta RQL [Karvounarakis 2002] y RVL [Magkanaraki 2003]. El enfoque principal de este algoritmo es encontrar patrones de la consulta para elegir nodos relevantes para resolverla. El ruteo de consultas llevado a cabo por el algoritmo se realiza de la siguiente manera: un nodo recibe una consulta en lenguaje RQL o RVL, el nodo analiza la gramática de la consulta y crea un árbol de patrones de dicha consulta en cada nodo, y por cada patrón lleva a cabo un algoritmo de comparación. Si algún nodo contiene información de la consulta, éste es anotado en la ruta como parte de la respuesta de la consulta. La búsqueda se realiza en redes P2P estructuradas y está enfocada a las estructuras RDF. En este trabajo no se presentan los resultados experimentales.

Lu [Lu 2007] propone algoritmos locales para construir capas de red P2P basadas en contenido para soportar búsqueda de texto en múltiples bases de datos en redes grandes. El algoritmo utiliza un nodo considerado como proveedor, el cual se une a todos los nodos que tienen un alto nivel de contenido similar, si no encuentra nodos con alta similitud entonces se

conecta por lo menos con uno con similitud marginal, si el nodo proveedor contiene un nivel bajo de similitud con respecto a los demás, entonces se busca un nuevo proveedor que tenga un nuevo contenido para ser agrupado.

En contraste con trabajos existentes de capas de red diseñadas para soportar búsquedas sobre nombres de documentos, identificadores o palabras clave o un vocabulario controlado, este trabajo se encarga de auto-organizar dinámicamente los nodos de la red dentro de un control centralizado, es decir la búsqueda se lleva a cabo de manera central por cada nodo proveedor. Este trabajo funciona sobre una red P2P estructurada con características del modelo Small-world. Los datos de prueba utilizados fueron tomados de una muestra de la Conferencia de Recuperación de Texto (TREC, por sus siglas en inglés) WT10g [TREC 2002], que tienen una distribución de tipo zipfiana en sus contenidos.

Graga [Grega 2007] propone crear un simulador basado en bases de datos para resolver problemas de consultas de manera local, es decir, este trabajo se enfoca en resolver el problema de similitud de imágenes en cada repositorio local. En el algoritmo de consulta de este trabajo, el nodo que emite la consulta tiene que enviar valores de descripción de la imagen que busca a los nodos que son consultados y cada nodo que responde tiene que enviar una lista de distancias de la imagen solicitada desde los archivos que tiene en su repositorio. En este trabajo también se trata la generación de bancos de prueba (benchmarks) para que puedan ser utilizados como datos de comparación con otros trabajos. Los contenidos de la capa de red son dinámicos, pero no se mencionan las condiciones experimentales.

Zhou [Zhou 2004] propone un mecanismo de búsqueda basado en el protocolo de Gnutella. La idea principal de este algoritmo es enviar la consulta solicitada de acuerdo a la relevancia local de un nodo, determinada con una función de probabilidad que discrimina a que nodo elegir para continuar la consulta y de esta manera disminuir la congestión de la red. Sólo aquellos nodos estimados como relevantes recuperan la consulta y envían los mensajes de respuesta hacia el nodo fuente. El tipo de red que se utilizan es de tipo P2P no estructurada, cuyas conexiones son generadas de manera aleatoria con una distribución Power-Law

[Adamic 2000]. La búsqueda se realiza a través de palabras clave y sus contenidos fueron tomados de una muestra de la Conferencia de Recuperación de Texto (TREC) WT10g [TREC 2002].

Tempich [Tempich 2004] presenta el algoritmo REMINDIN' que emplea metáforas sociales para la creación de atajos, de la misma manera que los humanos, trata de encontrar respuestas a sus preguntas en un escenario del mundo real. El algoritmo define tres tipos de atajos que son creados cada vez que una consulta realiza una comparación del recurso solicitado en el nodo visitado, lo cual proporciona información que es utilizada para generar dichos atajos. El primero, son tipos de enlaces hacia los nodos que procesan los recursos que son respuestas adecuadas para una consulta dada (nodos proveedores de contenido). El segundo, son tipos de enlaces hacia nodos que emiten consultas similares en el pasado (nodos recomendadores). El tercero, son tipos de atajos que proveen enlaces hacia nodos que tienen una alta cantidad de conocimiento acerca de otros nodos (nodos de arranque).

El tipo de red en el que trabaja es una red P2P semi-estructurada con propiedades de Small-world. Los contenidos se encuentran en estructuras RDF, y son una muestra de una categoría contenida en la base de datos DMOZ [DMOZ 2004], cuya distribución de contenido es zipfiana.

Tabla 5.1. Características de trabajos relacionados con consultas semánticas

Trabajo Relacionado	Tipo de Red P2P	Distribución de contenido	Distribución de las consultas	Búsqueda de contenido
[Chrysakis 2006]	No estructurada	Uniforme y Zipf	Uniforme	XML y RDF
[Kokkinidis 2004]	Estructurada	---	---	RDF
[Grega 2007]	Estructurada	---	---	Imágenes
[Lu 2007]	Estructurada	Zipf	Zipf	Texto Libre
[Zhou 2004]	No Estructurada	Zipf	Zipf	Palabras clave
[Tempich 2004]	Semi-Estructurada	Zipf	Uniforme	RDF
Ésta tesis	No estructurada	Uniforme y Zipf	Zipf y Uniforme	Palabras clave

5.2. Trabajos Relacionados con Consultas Semánticas y Algoritmos de Hormigas

En esta sección se revisan los trabajos relacionados con algoritmos de hormigas en redes P2P, y un resumen de ellos se presenta en la Tabla 5.2.

Babaöglu [Babaöglu 2002] presenta una aplicación para compartir archivos llamada *Gnutant*, basada en la plataforma *AntHill* [Babaöglu 2002], que es una plataforma de código abierto escrita en Java, para el diseño, implementación y evaluación de algoritmos de hormigas en redes P2P. En la aplicación *Gnutant*, cada archivo tiene un único identificador asociado con meta-datos, los cuales están formados por palabras-clave de texto. El algoritmo de enrutamiento se basa en la metáfora de las hormigas, pero no en los principios del clásico meta-heurístico de *Optimización de Colonia de Hormigas* [Dorigo 2004]. Tres tipos diferentes de hormigas son responsables de construir un índice distribuido que contiene direcciones URL de archivos compartidos y de administrar las tablas de enrutamiento. El tipo de red P2P utilizada es no estructurada y esta basada en la red Gnutella [Clip 2002].

Schelfthout [Schelfthout 2003] evalúa el principio de feromona sintética que es empleada para coordinación de agentes en ambientes distribuidos. Su plataforma esta basada en la idea de objetos espaciales conocidos en la computación concurrente. De manera similar al trabajo de Babaöglu, este trabajo genera rastros de feromona para cada consulta y permite que los agentes sigan los rastros que representan solo una de las múltiples palabras en la consulta. La evaporación es usada con un factor equivalente a 0.1. El ambiente experimental es muy pequeño (alrededor de 200 nodos).

La tabla de feromonas es actualizada con valores de probabilidad que son determinados en función del recurso encontrado, es decir, si el resultado fue encontrado en un nodo, ese nodo tendrá la probabilidad de 1 de ser seleccionado, en el caso de que se encuentre en diferentes nodos la probabilidad se divide entre ellos. La elección del siguiente nodo para moverse depende del rastro de feromona, es decir, se elige aquel nodo que ha obtenido más cantidad de feromona de acuerdo al resultado obtenido; en el caso de que los valores de

feromona sean iguales, la selección se realiza de manera aleatoria. Se maneja una red P2P no estructurada generada de manera aleatoria. La búsqueda se realiza a través de nombres de archivo.

Yang [Yang 2006] propone un algoritmo llamado *AntSearch* para redes P2P no estructuradas. La feromona almacena valores correspondientes a la tasa de éxitos de consultas anteriores sobre los nodos que contienen los resultados encontrados. Este trabajo fue motivado por la necesitad de mejorar el proceso de búsqueda en términos del tráfico generado en la red y el nivel de información recuperada.

El algoritmo de enrutamiento, primero genera una exploración mediante un método de inundación con la cual inicializa la tabla de feromonas, asignando probabilidades hacia recursos con mayor probabilidad de ser buscados. Posteriormente cada vez que se realiza una búsqueda se eligen los nodos con mayor cantidad de feromona para lanzar pequeñas inundaciones y disminuir el congestionamiento de la red al evitar nodos que no proporcionan la información deseada. La instancia de red que se utiliza en este trabajo es una muestra de una red Gnutella de 160,000 nodos. La búsqueda se realiza a través de nombres de archivo.

Michlmayr [Michlmayr 2007] propone un algoritmo distribuido para el enrutamiento de consultas semánticas en redes P2P llamado *SemAnt*. En este trabajo se incluye la evaluación de los parámetros de configuración del algoritmo mediante experimentación basada en fuerza bruta eligiendo un conjunto de valores de parámetros para evaluar su desempeño. Propone un algoritmo para enrutamiento de consultas que trata de establecer un balance entre el tráfico de red y la cantidad de recursos encontrados, con lo cual obtiene un valor de bondad de los resultados que es almacenado en su tabla de feromonas.

El algoritmo de enrutamiento que utiliza esta basado en las reglas de selección de ACS [Dorigo 1997] y en la Metodología ruteo de AntNet [Di Caro 1998], pero incluye una diferencia en la función de actualización global de la feromona, la cual almacena la tasa de éxitos y el tiempo de vida del agente. La base experimental de este trabajo es una red P2P no

estructurada, con topología small-world, de 1,024 nodos, con contenido y consultas generadas de manera aleatoria de los tópicos de la clasificación de ACM Computing Classification System (ACM CCS).

Tabla 5.2. Características de trabajos relacionados con algoritmos de hormigas

Trabajos Relacionados	Algoritmo de Hormigas	Distribución de contenido	Distribución de las consultas
[Babaöglu 2002]	---	Zipf	Zipf
[Schelfthout 2003]	---	Aleatoria	Aleatoria
[Yang 2006]	---	Zipf	Zipf
[Michlmayr 2007]	ACS[Dorigo 1997] y AntNet [Di Caro 1998]	Uniforme	Uniforme
Ésta tesis	ACS [Dorigo 1997] SemAnt[Michlmayr 2007]	Zipf y Uniforme	Zipf y Uniforme

5.3. Discusión de Trabajos Relacionados

A continuación se hace una discusión de los trabajos relacionados.

Trabajos relacionados con consultas semánticas

Como se ha mencionado anteriormente, existen pocos trabajos que abordan el enrutamiento de consultas semánticas (como se muestra en la Tabla 5.3), dentro de los cuales se manejan diferentes tipos de red P2P y la búsqueda de contenidos también varía de acuerdo al enfoque que se esta manejando.

Tabla 5.3. Análisis de trabajos relacionados con consultas semánticas

Trabajo Relacionado	Topología	Técnicas de Aprendizaje	Caracterización Topológica
[Chrysakis 2006]	Estática	---	---
[Kokkinidis 2004]	Estática	---	---
[Grega 2007]	Estática	---	---
[Lu 2007]	Dinámica	---	---
[Zhou 2004]	Estática	---	---
[Tempich 2004]	Dinámica	Sí	---
Ésta tesis	Estática	Sí	DDC

Trabajos como el de Lu y Tempich realizan cambios en la topología para tomar ventaja de la localización de documentos, de tal manera que organizan los archivos en ciertos nodos

conocidos de la red, con lo cual la búsqueda se hace más dirigida. La desventaja de estos métodos es que consumen muchos recursos para realizar esta tarea, a pesar de que las consultas obtienen una buena cantidad de recursos deseados.

En la mayoría de estos trabajos, con excepción del trabajo de Tempich, no se utilizan técnicas de aprendizaje para optimizar la búsqueda. Aunque hay cierta noción de la ubicación de los recursos, éstos se enfocan principalmente a usar filtros para tener información similar a lo que se esta buscando, y no abordan el problema de disminuir la cantidad de saltos necesarios para satisfacer la consulta y disminuir el tráfico de la red.

A diferencia de otros trabajos relacionados mencionados anteriormente, en este trabajo se toma ventaja de la topología de red utilizando métricas que permiten discriminar aquellos nodos que pueden ser más aptos para continuar con la búsqueda, aprendiendo de manera automática las rutas que contengan más recursos encontrados y reduciendo la cantidad de saltos necesarios para satisfacer la consulta.

Trabajos relacionados con Consultas Semánticas y Algoritmos de Hormigas

Los algoritmos de hormigas son muy populares y efectivos en la solución de problemas de grafos, pero no ha sido, sino hasta estos últimos años en donde se ha incrementado el uso de estos algoritmos en la solución de problemas de enrutamiento de consultas. En la Tabla 5.4, se muestran los trabajos relevantes que utilizan algoritmos de hormigas para esta clase de problemas.

Tabla 5.4. Análisis de los algoritmos de hormiga aplicados a enrutamiento de consultas

Trabajos Relacionados	Tiempo de vida de los agentes	Caracterización Topológica Local	Topología de Red
[Babaöglu 2002]	Constante	---	Gnutella
[Schelfthout 2003]	Constante	---	Aleatoria
[Yang 2006]	Estimado	---	Gnutella
[Michlmayr 2007]	Constante	---	Small-world
Ésta tesis	Constante	DDC	Scale-free y Small-world

Existen muy pocos trabajos en los cuales se haya aplicado la teoría de hormigas y solo el trabajo de Michlmayr ha aplicado la teoría de optimización de colonia de hormigas [Dorigo 2004]. La diferencia entre el trabajo de Michlmayr y esta tesis, con respecto a esta teoría, es que el algoritmo propuesto toma los mecanismos de actualización de feromona tal y como fueron argumentados y sustentados para el algoritmo ACS, permitiendo de manera automática realizar evaporación local sin necesitad de que cada nodo tenga su propio tiempo de evaporación, lo que podría causar algún efecto extraño al estar manejando la evaporación por intervalos fijos de tiempo.

Con excepción de Yang todos los trabajos relacionados utilizan un tiempo de vida del agente de manera estática, definido de manera experimental.

Todos los algoritmos de hormigas para enrutamiento de consultas semánticas ofrecen, en cierta medida, aprendizaje generado por las consultas anteriores; sin embargo, la principal diferencia entre dichos algoritmos y el propuesto en esta tesis, reside principalmente en que el algoritmo propuesto toma ventaja del ambiente local donde se lleva a cabo la búsqueda, en términos del método local de exploración *lookahead* [Mihail 2007] y la métrica topológica local *DDC* [Ortega 2007].

Capítulo 6

ALGORITMO DE SOLUCION PROPUESTO "*Neighboring-Ant Search*" (NAS)

En este capítulo se presenta el algoritmo de solución propuesto para el problema de enrutamiento de consultas semánticas. Se presenta la arquitectura del sistema, las estructuras de datos manejadas, las funciones propias del algoritmo y la descripción del algoritmo de búsqueda NAS.

6.1. Arquitectura del Sistema Multi-agente.

La arquitectura general del sistema se muestra en la Figura 6.1 y comprende dos elementos principales: *i*) El *ambiente E* es una red compleja con una distribución de tipo Scale-free [Barabási 1999], comprendida como un conjunto de nodos enlazados que contienen información local acerca de sus nodos vecinos, *ii*) Los *agentes* $\{e_1, e_2, e_3\}$, que pueden ser de tres tipos dependiendo del rol que desempeñen, son representados como *hormigas* del algoritmo NAS que modifican el medio ambiente o como el nodo que emite o recupera las consultas.

El *agente de consulta* $\{e_1\}$, es representado por el nodo que emite la consulta y su rol es crear el *agente de búsqueda*. El *agente de búsqueda* $\{e_2\}$ es representado por una hormiga, la cual fue creada por el nodo que emite la consulta y avanza a través de los nodos

dependiendo de un conjunto de reglas definidas hasta satisfacer un conjunto de objetivos deseados. El *agente de recuperación* $\{e_3\}$, es creado cuando se encuentra un recurso buscando en un nodo.

El agente de búsqueda esta formado por un conjunto de módulos que permiten su funcionamiento, tales como el *selector de acciones*, el cual es responsable de elegir hacia que nodo avanzar seleccionando el *generador de acciones exploratorias o explotatorias*, implementadas por las Ecuaciones 6.4 y 6.5. Cada vez que se genera una acción se activa el módulo de *actualización local,* el cual utiliza la Ecuación 6.6 para llevar a cabo la evaporación local de la tabla de feromonas. Esta tabla actúa como una memoria que recuerda implícitamente las mejores rutas que han sido seleccionadas.

Durante la búsqueda, cada vez que se encuentra un recurso buscado, éste genera un *agente de recuperación*. El periodo de vida del agente de búsqueda es válido hasta que la cantidad de saltos (TTL) sea igual a cero (siendo 25 el tiempo de vida asignado inicialmente y decrementando este valor en una unidad por salto) o la cantidad de recursos solicitados sea satisfecha.

El *agente de recuperación*, que es creado cuando se encuentra un recurso, es responsable de recuperar el recurso y evaluar el desempeño del *agente de búsqueda*. Con esta información se retroalimenta la tabla de feromonas sobre la ruta que fue utilizada por el agente de búsqueda hasta el *agente de consulta*. Para la actualización se utiliza le Ecuación 6.7. Todos los módulos usan parámetros de control se configuran apropiadamente para asegurar el buen desempeño del sistema.

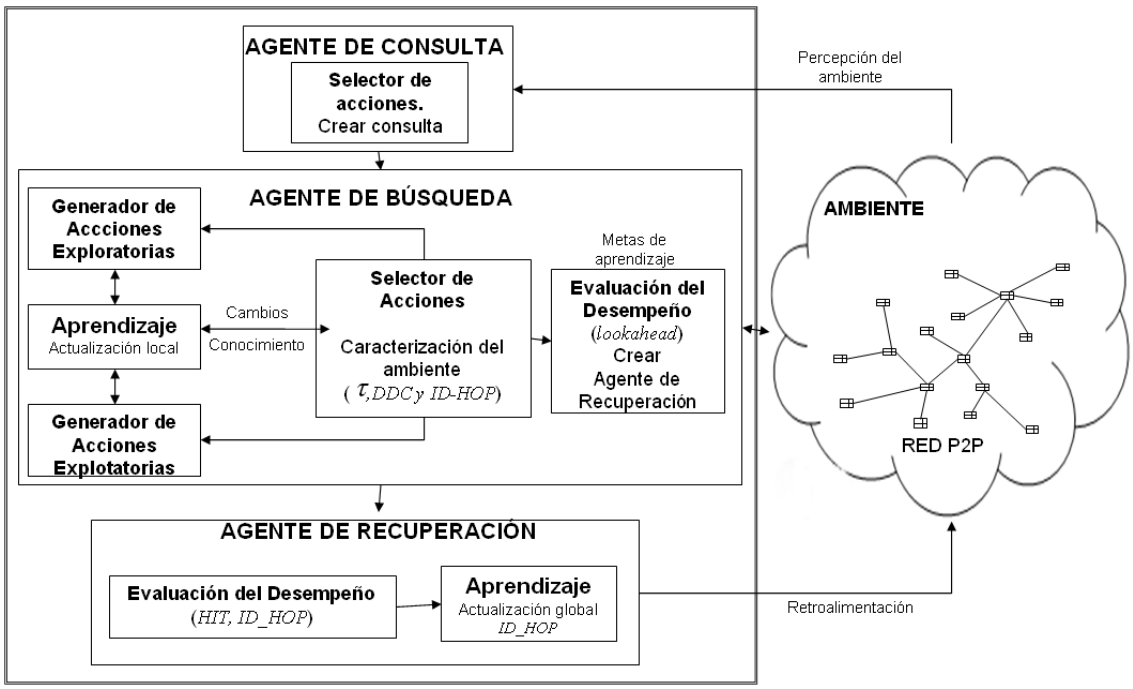

Figura 6.1. Arquitectura general del sistema multi-agente propuesto.

6.2. Estructuras de Datos

Las estructuras de datos utilizadas en este sistema multi-agente son esenciales para el funcionamiento del algoritmo ya que contienen toda la información necesaria concerniente a los nodos, sus vecinos y los agentes que se encargan de modificar el ambiente, así como los datos necesarios para representar el aprendizaje de consultas pasadas.

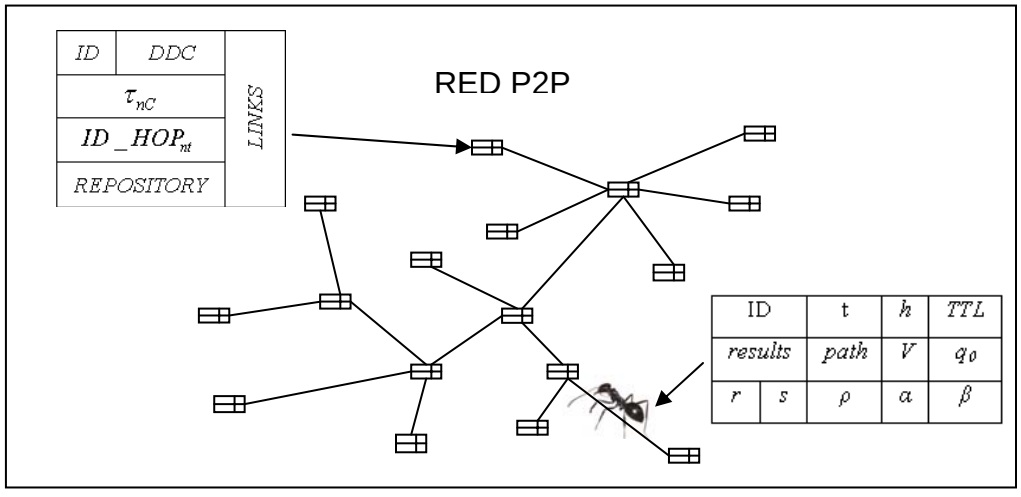

Figura 6.2. Estructuras principales del sistema multi-agente.

Existen dos elementos principales que definen el funcionamiento del algoritmo sobre una red compleja: la estructura perteneciente al nodo y la estructura del agente (hormiga), como se observa en la Figura 6.2.

Las estructuras de datos del algoritmo original ACS solo hacen referencia a la tabla de feromonas, la cual sólo contiene una sola clase de feromona porque el aprendizaje es sobre un solo objetivo buscado. Debido a esto la tabla de feromonas fue modificada con el motivo de tener diferentes clases de feromonas, que permitieran indicar la cantidad de feromona depositada en cada nodo para cierto recurso buscado, además de incluir otros parámetros que son utilizados como valores heurísticos que permiten obtener mayor conocimiento del ambiente local.

La estructura de los nodos (como se muestra en la Figura 6.3) consta de elementos que permiten tener conocimiento local de sus nodos vecinos, cada nodo contiene: un identificador único (ID) para distinguirse de los demás nodos, el Coeficiente de Dispersión de Grado (DDC), una tabla de feromonas (τ) de tamaño n x C que funciona como tabla de enrutamiento, donde n es la cantidad de vecinos y C es la cantidad de recursos que han sido utilizados en las consultas que han pasado por esos nodos, la importancia de la distancia (ID_HOP) que corresponde a una tabla de n x C elementos, este factor indica la importancia de ir hacia cada nodo vecino en función de la longitud de la ruta, y el repositorio local de cada nodo (*REPOSITORY*), que comprende una tabla de dispersión local (tabla hash) con los datos de cada uno de los recursos del repositorio local y por último los enlaces hacia otros nodos (*LINKS*).

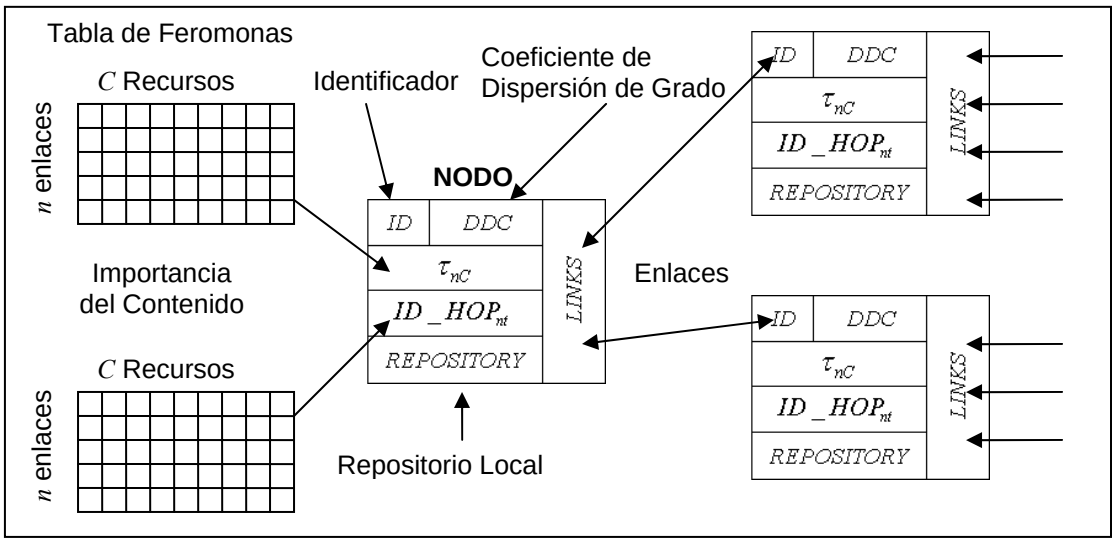

Figura 6.3. Estructura de un nodo

La estructura del agente (hormiga, que representa al agente de búsqueda o de recuperación), mostrada en la Figura 6.4, esta formada de parámetros, los cuales se incorporan a las funciones que le permiten discriminar y actualizar información de los nodos que visita. El agente contiene: un identificador único (ID) para distinguirse de los demás agentes, t que es recurso a buscar, la cantidad de saltos (h) que ha realizado, el tiempo de vida restante (TTL), la ruta seleccionada ($path$), los nodos visitados (V), el nodo que inicio la consulta (r), el nodo que esta evaluando (s), los parámetros de configuración de las ecuaciones de selección (β, q_0) y de actualización (α, ρ) y la cantidad de recursos encontrados ($results$).

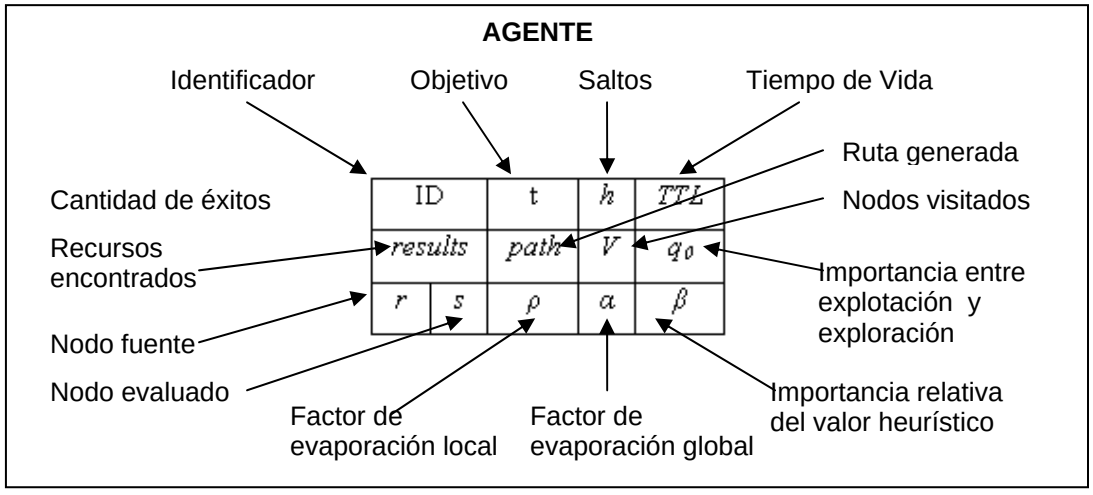

Figura 6.4. Estructura de un agente (hormiga)

6.3. Funciones Heurísticas para el Aprendizaje

El algoritmo clásico de ACS esta formado de reglas (selección y actualización) que permiten la convergencia del sistema hacia mejores resultados. Dado que el problema para el que fue diseñado el algoritmo ACS no tiene el mismo objetivo que el problema que se aborda en esta tesis, se propusieron modificaciones de estas reglas con el objetivo de adaptar el algoritmo base ACS al problema de enrutamiento de consultas semánticas.

También fueron añadidas nuevas funciones (tales como la métrica topológica DDC, la importancia de los éxitos (HIT), la importancia del tiempo de vida del agente (ITL_HOP) y la distancia hacia un recurso buscado (ID_HOP)) que mejoran el desempeño del sistema multi-agente en términos de la función objetivo del problema. En esta sección se describen los ajustes realizados.

Función de Importancia de los Éxitos (HIT)

Esta función califica el desempeño del *agente de búsqueda* basado en los recursos encontrados y los recursos esperados, se aplica en la función de actualización global (6.7) con el propósito de guiar las consultas hacia rutas que provean la mayor cantidad de recursos encontrados:

$$HIT = \frac{results_k}{maxResults}, \tag{6.1}$$

donde $results_k$ representa la cantidad de recursos encontrados por la hormiga k, y $maxResults$ es la cantidad máxima de recursos esperados.

Función de Importancia del Tiempo de Vida (ITL_HOP)

Esta función califica el desempeño del *agente de búsqueda* basándose en el tiempo de vida parcial del agente y en el tiempo originalmente asignado. Se aplica en la función de actualización global con el objetivo general de disminuir el tiempo de vida de la hormiga k, necesario para encontrar un conjunto de recursos.

$$ITL\_HOP = \frac{TTL_{max}}{2 \cdot TTL_k}, \tag{6.2}$$

donde TTL_k es el tiempo de vida parcial de la hormiga k hasta ese momento, y $TTLmax$ es el tiempo de vida máximo asignado a una hormiga de búsqueda para una consulta.

Función de la Importancia de la Distancia (ID_HOP)

Esta función califica a cada nodo dependiendo de la distancia hacia cada recurso anteriormente encontrado. Una vez que una hormiga k genera una ruta hasta un recurso, la función $ID\_HOP_{r,s,t}$ se aplica a cada nodo perteneciente a esta ruta para incrementar su importancia en términos de la longitud de ruta con respecto a la distancia hacia el recurso encontrado. Esta función se obtiene mediante el inverso de la cantidad de saltos hasta el nodo evaluado $h_{r,s,t}$ dividido por el total de saltos realizados por la hormiga hasta el último recurso encontrado h_k:

$$ID\_HOP_{r,s,t} = \left(\frac{h_k}{h_{r,s,t}} \right)^{-1} \quad \forall r,s,t \in ruta\ k. \tag{6.3}$$

6.4. Reglas Modificadas de Selección de Acciones y de Actualización del Aprendizaje

La regla de transición de estado y la regla de exploración originales del algoritmo ACS fueron modificadas cambiando el valor heurístico original η (Ecuaciones 6.1 y 6.2), por el valor de la métrica *DDC* más el valor de la función de importancia de la distancia (*ID_HOP*), con el objetivo de dar mayor preferencia a los nodos con mejor conexión y menor distancia hacia un objetivo buscado.

La función de transición de estado de NAS da la probabilidad con la cual una hormiga k en un nodo r elige moverse hacia el nodo s. El parámetro q_0 determina la importancia relativa de la explotación contra la exploración. Cada vez que una hormiga en un nodo r ha elegido un nodo s para moverse, esta genera un número aleatorio uniforme $0 \le q \le 1$. Si $q \le q_0$ entonces el mejor nodo es elegido de acuerdo a la Ecuación 6.4 (explotación). Se refiere al

nodo que posea mayor cantidad de feromona, mejor conectividad y menor cantidad de saltos hacia un recurso buscado, de otra manera se elige el nodo de manera aleatoria proporcional de acuerdo a la Ecuación 6.5, que se utiliza para exploración.

En el caso de que el valor aleatorio generado sea menor que q, se elige la *estrategia de explotación* que selecciona un nodo que provea más información para continuar el enrutamiento hacia un nodo vecino, es decir, elige un nodo que proporcione una mayor cantidad de feromona y mejor conectividad con la menor cantidad de saltos hacia un recurso.

$$s = \begin{cases} \arg\max_{u \in \Gamma(r) \wedge u \notin V_k} \left\{ \left[\tau_{r,u,t} \right] \cdot \left[DDC_u + ID\_HOP_{r,u,t} \right]^{\beta} \right\}, & \text{si } q \le q_0 \text{ (explotación)} \\ S, & \text{de lo contrario (exploración parcial),} \end{cases} \tag{6.4}$$

donde r es el nodo actual donde se localiza la hormiga k, u pertenece al conjunto de nodos vecinos de r, V_k es el conjunto de nodos visitados por la hormiga k, t es el recurso buscado, β es el parámetro que determina la importancia relativa entre la feromona y el DDC con la función de importancia de la distancia *ID_HOP* (6.3), y S es una variable pseudo-aleatoria determinada por la *estrategia de exploración* (6.5).

La *estrategia de exploración* (6.5) estimula a las hormigas a buscar nuevas rutas. Esta estrategia aplica la función de selección pseudo-aleatoria f, basada en la técnica de la ruleta [Goldberg 1990] para favorecer de manera pseudo-aleatoria a los nodos con una gran cantidad de feromona y mejor conectividad con la menor cantidad de saltos hacia un recurso solicitado.

$$S = f(p_{r,u,t}), \quad p_{r,u,t} = \frac{\left[\tau_{r,u,t} \right] \cdot \left[DDC_u + ID\_HOP_{r,u,t} \right]^{\beta}}{\sum_{u \in \Gamma(r) \wedge u \notin V_k} \left[\tau_{r,u,t} \right] \cdot \left[DDC_u + ID\_HOP_{r,u,t} \right]^{\beta}}. \tag{6.5}$$

Las reglas de actualización originales del algoritmo ACS son de dos tipos: actualización local y actualización global. Con respecto a la regla de actualización no se realizó ninguna modificación ya que el mecanismo de evaporación automático tiene el funcionamiento deseado para este problema. La única modificación fue realizada sobre la regla de actualización global, debido a que ésta asigna la recompensa a cada uno de los estados

visitados por el agente, en el algoritmo original esta recompensa era asignada con el valor de la longitud de circuito más corto, que era el objetivo original de ACS, encontrar el circuito con menor distancia.

En el caso del problema abordado, la regla de actualización requiere resolver más objetivos: disminuir la cantidad de saltos necesarios para encontrar un recurso y recuperar la mayor cantidad de recursos buscados. Por lo cual se realizó una única modificación en la regla de actualización global, cambiando el valor del circuito con menor distancia $\Delta\tau(r,s)$ (6.3), por las Ecuaciones 6.1 (HIT) y 6.2 (ITL_HOP), con el objetivo de asignar una recompensa que fuera acorde a los objetivos deseados, maximizar la cantidad de recursos encontrados y minimizar la cantidad de saltos utilizados por las hormigas de búsqueda.

Las reglas de actualización son usadas para retroalimentar el sistema sobre rutas exitosas. Cada ves que una hormiga decide moverse hacia un nodo a través de la selección, la feromona se evapora en cada nodo que ha sido visitado, de acuerdo a la regla de *actualización local* (6.6) con el fin de reducir la cantidad de feromona en ese nodo, evitando así que ciertos nodos sean predominantes sobre otros y manteniendo los niveles de feromona en un rango estable [Sutton 1996].

$$\tau_{r,s,t} \leftarrow (1-\rho)\cdot\tau_{r,s,t} + \rho\cdot\tau_0, \tag{6.6}$$

donde τ_0 es el valor de inicialización de la tabla de feromonas y ρ es el factor de evaporación local de la feromona.

La regla de actualización global (6.7) se lleva a cabo cada vez que un recurso es encontrado y se aplica a cada nodo perteneciente a una ruta exitosa. La cantidad de feromona que es depositada depende de la calidad de la solución obtenida, mediante la cantidad de recursos encontrados y el tiempo de vida restante de la hormiga.

$$\tau_{r,s,t} \leftarrow (1-\alpha)\cdot\tau_{r,s,t} + \alpha\cdot\left[w_d HIT + (1-w_d) ITL\_HOP\right] \quad \forall r,s,t \in ruta\ k \tag{6.7}$$

donde α es el factor de evaporación global de la feromona, w_d es la influencia entre los recursos encontrados (*HIT*) y el tiempo de vida de la hormiga (*ITL_HOP*).

6.5. Algoritmo NAS

El algoritmo NAS es un algoritmo metaheurístico basado en los algoritmos ACS [Dorigo 1997] y SemAnt [Michlmayr 2007], donde un conjunto de agentes independientes cooperan indirecta y esporádicamente para alcanzar un conjunto de objetivos en común. La operación de cada agente es distribuida, la colaboración indirecta se alcanza a través de la tabla local de feromonas en cada nodo, donde cada hormiga deposita "rastros" de acuerdo a la calidad del resultado obtenido. La Tabla 6.1 muestra el algoritmo de NAS.

Tabla 6.1. Pseudocódigo del algoritmo NAS

```
01  en_paralelo  { // Actividad Concurrente en cada nodo
02        Para cada consulta en r_k crear agente de búsqueda
03           Mientras agente de búsqueda este vivo  // Actividad Concurrente del agente de bús.
04              Si Hits < maxResults y TTL > 0                          // Fase 1
05                 Si en r_k el nodo vecino s_k  tiene el recurso buscado // Estrategia lookahead
06                    r_k = añadir s_k  a Path_k
07                    TTL_k = TTL_k -1
08                    actualizaciónGlobal  // Actividad concurrente del agente de recuperación (5.7)
09                 De lo contrario                                       // Fase 2
10                    s_k = Regla de Transición de Estado (5.4)
11                    Si r_k es nodo hoja o s_k había sido visitado anteriormente
12                       remover el último nodo de Path_k y retroceder en la ruta
13                    De lo contrario
14                       r_k = añadir s_k  a Path_k
15                       TTL_k = TTL_k -1
16                       actualizacionLocal (5.6)
17              De lo contrario
18                 Eliminar agente de búsquedat
19  } // Fin de la actividad concurrente
```

El algoritmo tiene dos objetivos principales: guiar las consultas hacia nodos mejor conectados usando la métrica topológica local *DDC* [Ortega 2007] y minimizar la cantidad de saltos hacia recursos anteriormente encontrados. En este algoritmo una consulta es representada por una hormiga de búsqueda, y entre más frecuente sea una consulta hacia un recurso, la mejor ruta es elegida; es decir el grado de optimización de una consulta depende directamente de su popularidad.

Otra adaptación realizada al algoritmo es la técnica clásica *lookahead* [Mihail 2007], cuando una hormiga se localiza en un nodo *r*, ésta tiene conocimiento de su conjunto de nodos vecinos de nivel 1. A través de esta técnica la hormiga es responsable de solicitar el recurso buscado a cada uno de sus vecinos no visitados.

El algoritmo NAS consiste de dos fases principales. La Fase 1 corresponde a la *evaluación de resultados* (líneas 05-08 en el pseudocódigo de la Tabla 6.1). En esta fase a través de la técnica *lookahead*, la *hormiga de búsqueda k* localizada en el nodo *r*, consulta a sus nodos vecinos, por el recurso solicitado. Si el recurso es encontrado, una *hormiga de recuperación* es creada, esta hormiga recupera el recurso y a través de la función de actualización global dada en la Ecuación (6.7) la ruta utilizada por la *hormiga de búsqueda* es retroalimentada con la bondad del resultado obtenido, al mismo tiempo, *ID_HOP* (6.3) es actualizado. En el caso de que en la fase de evaluación no haya encontrado algún recurso, la fase dos se lleva a cabo.

La Fase 2 corresponde a la transición de estados (líneas 9-16 en el pseudocódigo de la Tabla 6.1) llevada a cabo por la *hormiga de búsqueda*, en la cual a través de *q*, se selecciona un nodo vecino *s* por la función de explotación (6.4) o exploración (6.5). En el caso de que no exista un nodo hacia el cual avanzar (es decir se encuentra en un nodo hoja o todos los nodos vecino han sido visitados) se realiza un salto hacia atrás sobre la ruta realizada por el *agente de búsqueda*, de lo contrario el *agente de búsqueda* añade el nodo *s* a su ruta, aplicando la actualización local (6.6) y reduciendo TTL_k en un salto. El proceso de la consulta termina cuando el número de resultados esperados ha sido satisfecho o cuando TTL_k es igual a cero. El *agente de búsqueda* es eliminado indicando el fin de la consulta.

6.6. Parámetros de configuración

Las funciones del algoritmo NAS contienen parámetros que permiten su buen desempeño. La configuración de los parámetros fue tomada del trabajo de Michlmayr en el cual se llevó a cabo un ajuste de parámetros realizando pruebas con diferentes valores de los parámetros. A continuación en la tabla 6.2 se muestra la configuración de los parámetros utilizados.

Tabla 6.2. Parámetros de configuración de NAS

$\rho = 0.07$	Factor de evaporación local de la feromona
$\alpha = 0.07$	Factor de evaporación global de la feromona
$\tau_0 = 0.009$	Parámetro de inicialización de la tabla de feromonas
$ID\_HOP_0 = 0.001$	Parámetro de inicialización de la tabla heurística de distancias hacia los recursos anteriormente encontrados.
$\beta = 2$	Importancia relativa de *DDC* y *ID_HOP* con respecto a la feromona
$q_0 = 0.9$	Importancia entre *exploración* y *explotación*
maxResults =5	Cantidad máxima de resultados deseados
$TTL_{max} = 25$	Tiempo máximo de vida de los *agentes de búsqueda*
$w_d = 0.5$	Importancia entre los recursos encontrados y el tiempo de vida del *agente de búsqueda*

Capítulo 7

EXPERIMENTACIÓN

En este capítulo se describen los experimentos que fueron llevados a cabo para evaluar el desempeño del algoritmo NAS propuesto para la búsqueda de información en redes P2P y enfocado al problema de Enrutamiento de Consultas Semánticas (SQRP).

7.1. Ambiente Experimental

La siguiente configuración corresponde a las condiciones experimentales que son comunes para las pruebas descritas en este capítulo.

Software

a) Sistema operativo, Ubuntu 7.1.
b) Plataforma de Java, JDK 1.6.
c) Entorno de desarrollo integrado, NetBeans 6.1.
d) Herramienta de modelado de agentes, Repast 3.1.
e) Librería de código abierto para alto desempeño en computación científica y técnica, Colt 1.2.0.

Hardware

Equipo de cómputo con 2 procesadores Xeon (TM) CPU 3.06GHz en paralelo y memoria en RAM de 4 GB.

Métricas de Desempeño de Algoritmos

En la actualidad existen pocas aplicaciones para el enrutamiento de consultas semánticas, las cuales son difíciles de contrastar ya que no existe un estándar en la abstracción del problema del mundo real y en las métricas utilizadas por los investigadores para mostrar el desempeño de sus algoritmos. En este trabajo, como en algunos otros, se aplican las métricas de desempeño en función de la cantidad de consultas emitidas, en otros casos se miden en función del tiempo de ejecución. Además, debido a que la descripción de los algoritmos es muchas veces incompleta, los lenguajes de programación son diferentes y las computadoras donde se lleva a cabo la experimentación son de diferente capacidad, entre otros factores, es muy difícil la reproducibilidad de dichos experimentos, imposibilitando la comparación entre algoritmos de diferentes investigadores.

Las métricas aplicadas a ésta experimentación fueron seleccionadas de cuatro trabajos que utilizan métricas más prácticas para obtener el desempeño de los algoritmos de búsqueda en redes P2P [Balakrishnan 2003], [Chen 2005], [Michlmayr 2007] y [Yang 2002].

La diferencia de las métricas utilizadas en esta tesis con respecto a los trabajos anteriormente mencionados es que las mediciones fueron tomadas con respecto al número de consultas satisfechas y no con respecto al tiempo de ejecución, con el fin de estandarizar los resultados independientemente del equipo de cómputo y otros factores, ya que el tiempo de procesamiento varía con respecto al hardware y software utilizado.

Las métricas utilizadas en esta experimentación, son las siguientes:

a) *Saltos promedio* (Average hops) es definido como la cantidad de enlaces viajados por el agente de búsqueda (hormiga) hasta su muerte (al encontrar todos los recursos buscados o terminar su tiempo de vida)
b) *Tasa de éxitos promedio* (Average hit-rate) es definido como el número de recursos encontrados por cada agente de búsqueda (hormiga) hasta su muerte (al encontrar todos los recursos buscados o terminar su tiempo de vida)
c) *Eficiencia Promedio* (Average efficiency) es el radio de saltos promedio para encontrar un recurso en la red. Éste se obtiene dividiendo los saltos promedio entre la tasa de éxitos promedio, lo que retorna la cantidad de saltos necesarios para encontrar un recurso.

Cada una de las pruebas realizadas tiene diferentes configuraciones que dependen del objetivo del experimento. A continuación se presenta la experimentación realizada.

7.2. Evaluación del algoritmo NAS utilizando el DDC

Existen pocos trabajos relacionados con enrutamiento de consultas semánticas que aplican los fundamentos de los algoritmos de colonias de hormigas. Los algoritmos de colonias de hormigas fueron diseñados para resolver problemas de grafos obteniendo muy buenos resultados, por lo cual se diseñó esta aplicación basada en el trabajo de Ant Colony System [Dorigo 1997] que es el algoritmo que ofrece un mejor rendimiento entre todos los algoritmos de hormigas existentes.

Para modelar esta aplicación fue necesario adaptar los diferentes parámetros del algoritmo original, además de modificar las estructuras de datos manejadas ya que el algoritmo original maneja un solo tipo de feromona y fue diseñado para un grafo totalmente conexo. El algoritmo propuesto fue mejorado con dos estrategias: la búsqueda en los repositorios locales (utilizando la técnica lookahead [Mihail 2007]) y la métrica topológica

local DDC (para guiar la búsqueda hacia subgrafos mejor conectados). A continuación se muestran los resultados obtenidos para la prueba de desempeño del algoritmo NAS.

Objetivo

Mostrar la mejora del desempeño del algoritmo de búsqueda NAS al incluir información topológica local.

Descripción

Para este experimento fue creada una red de 1,024 nodos de tipo Scale-free con el modelo de Barabási-Albert [Barabási 1999]. Los recursos contenidos en los nodos corresponden a números entre 0 y 1,023 creados uniformemente con el generador de distribución uniforme de Repast [ROAD 2005], a cada nodo le fueron asignados recursos y replicados con una distribución Zipfiana. Cada unidad de tiempo fue tomada cada 100 milisegundos, los experimentos se llevaron a cabo durante 10,000 unidades de tiempo. Durante cada unidad de tiempo fue asignado a cada uno de los nodos, el 10% de probabilidad uniforme de lanzar una consulta, eligiendo aleatoriamente el recurso a buscar (números entre 0 y 1,023).

Resultados

En la Figura 7.1(a), se muestra la cantidad de saltos usados por el algoritmo NAS para satisfacer un conjunto de consultas, al usar y no usar el DDC. El algoritmo NAS sin DDC comienza con un promedio de 19 saltos, y conforme el algoritmo evoluciona, alrededor de la consulta 1,000, la cantidad de saltos se reduce a un rango de 12 a 14 saltos. Al incorporar el DDC al algoritmo NAS éste comienza con un promedio de 18 saltos y mientras el algoritmo evoluciona, alrededor de la consulta 1,000, la cantidad de saltos se reduce a un rango de 11 a 13 saltos.

En la Figura 7.1(b) se muestra la tasa promedio de éxitos (tasa de éxitos entre cantidad máxima de resultados esperados) desempeñada por el algoritmo NAS con y sin DDC. Para ambas estrategias la cantidad de resultados encontrados comienza alrededor del 85% al 90%, y

mientras el algoritmo evoluciona alrededor de la consulta 10,000 se puede observar que la tasa de éxitos se incrementa en un rango del 88% a un 95%.

En la Figura 7.1(c) se muestra la eficiencia promedio del algoritmo NAS como una función de la cantidad de saltos necesaria para satisfacer una consulta (saltos promedio entre tasa de éxitos promedio). El algoritmo sin DDC en la primera etapa necesita alrededor de 5 saltos promedio para encontrar un recurso. Alrededor de la consulta 1,000, el promedio de saltos se reduce a un rango de 2.7 a 3.4, y al incorporar el DDC, en la primera etapa necesita alrededor de 4.5 saltos promedio para encontrar un recurso. Alrededor de la consulta 1,000, la cantidad de saltos promedio se reduce a un rango de 2.5 a 3 saltos para encontrar un recurso.

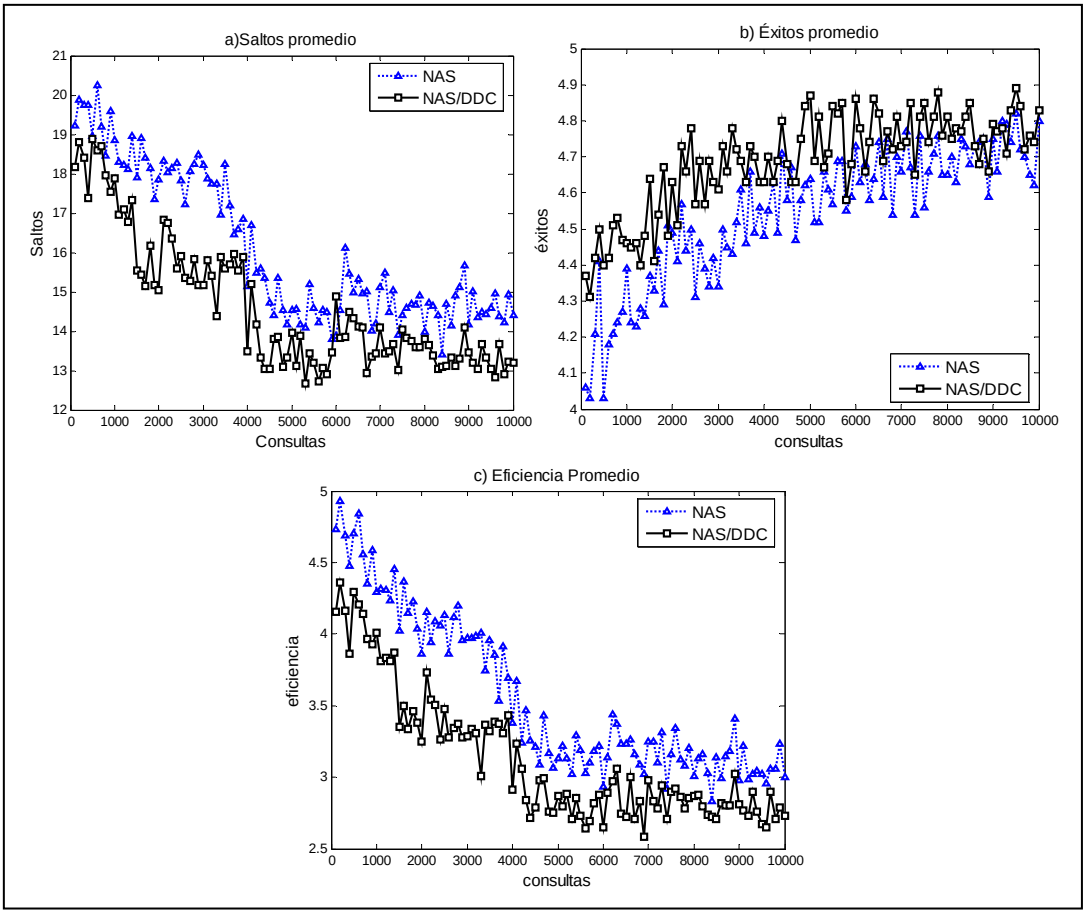

Figura 7.1. Gráficas de resultados sobre una red Scale-free cada 100 consultas de las métricas a) saltos promedio, b) tasa de éxitos promedio y c) eficiencia promedio del algoritmo NAS con y sin DDC.

Análisis de Resultados

Como resultado del decremento en la cantidad de saltos necesarios para satisfacer una consulta, podemos concluir que las hormigas de búsqueda aprenden a seleccionar mejores rutas hacia los recursos solicitados. Además, la variabilidad en los resultados se mantiene en un rango pequeño, esto es debido a la capacidad de exploración permanente del algoritmo NAS, lo cual es necesario para incorporar en el futuro otras características de redes complejas (por ejemplo, cambios en las conexiones de los nodos y la información de los recursos).

Al incluir información de la estructura local en el algoritmo se observa que la cantidad de saltos se reduce en 35% desde el inicio a la etapa estable reportada, y que la eficiencia se incrementa en un 35% en términos de la cantidad de saltos necesaria para encontrar un recurso. Estos resultados producen una mejora del 8%. También se alcanzó un desempeño promedio bueno en la tasa de éxitos: entre un 80% y 90%.

7.3 Comparación de Algoritmos de Búsqueda utilizando el DDC

En el experimento anterior se comprobó que el DDC aporta una mejora en el proceso de búsqueda del algoritmo propuesto. Con el motivo de corroborar lo anterior y realizar comparaciones, fue diseñado un algoritmo basado en la conocida Caminata aleatoria (RandomWalk) [Arora 2007], modificado para elegir aleatoriamente entre dos nodos vecinos y elegir moverse hacia aquél cuyo DDC sea mayor. A continuación se muestran los resultados del desempeño de ambos algoritmos aplicando el DDC en redes de dos modelos diferentes.

Objetivo

Comparar el desempeño de dos algoritmos de búsqueda al incorporar la métrica topológica DDC (Degree Dispersion Coefficient) sobre dos topologías de red diferentes.

Descripción

Para este experimento fueron generadas redes de 1024 nodos con dos diferentes modelos. El primer modelo es una red de tipo Small-world creada con el modelo de Watts y Strogatz [Watts 1998]. El segundo modelo es de tipo Scale-free con el modelo de Barabási-Albert

[Barabasí 1999]. Cada nodo de la red contiene 10 datos numéricos generados con una distribución uniforme entre 0 y 1,023. Cada unidad de tiempo fue tomada cada 100 milisegundos, los experimentos se llevaron a cabo durante 10,000 unidades de tiempo. Durante cada unidad de tiempo fue asignado a cada uno de los nodos, el 10% de probabilidad uniforme de lanzar una consulta, eligiendo aleatoriamente el recurso a buscar del total de recursos existentes (números entre 0 y 1,023).

Los algoritmos utilizados para la comparación fueron el algoritmo NAS (NeighboringAnt-Search) y el RW (Random-Walk), ambos utilizando y no utilizando la métrica topológica local DDC. Los dos algoritmos fueron configurados con 25 saltos de tiempo de vida máximo para realizar cada consulta

Resultados

En la Figura 7.2(a), se muestra que en las redes de tipo Scale-free, la cantidad de saltos utilizados por el algoritmo RW es muy cercana a 25 saltos al utilizar y no utilizar el DDC. Para el algoritmo NAS con y sin DDC, fueron necesarios 19 saltos. Al incorporar el DDC al algoritmo NAS se reduce la cantidad de saltos hasta 10. Para la topología de red Small-world, mostrada en la Figura 7.2(b), el algoritmo RW se comporta de la misma manera, utilizando todo el tiempo de vida asignado, Para NAS sin DDC, se necesitaron 24 saltos y utilizando DDC, 18 saltos.

En la Figura 7.2(c) se muestra la cantidad promedio de éxitos. En la topología de red Scale-free, el algoritmo RW obtiene entre 2 y 2.5 recursos por consulta sin utilizar el DDC, y al utilizarlo, la cantidad de recursos encontrados se incrementa entre 2.5 y 3 recursos encontrados. En el mismo tipo de red, el algoritmo NAS obtiene de 3 a 3.5 recursos encontrados por consulta sin usar el DDC, y de 4 a 4.5 al usar el DDC.

La Figura 7.2(d) muestra la misma métrica para la redes Small-world, donde el algoritmo RW obtiene de1.3 a 1.4 recursos encontrados al utilizar y no utilizar el DDC, y el algoritmo NAS sin DDC obtiene de 1.2 a 1.6 recursos encontrados, y con DDC de 1.3 a 1.8.

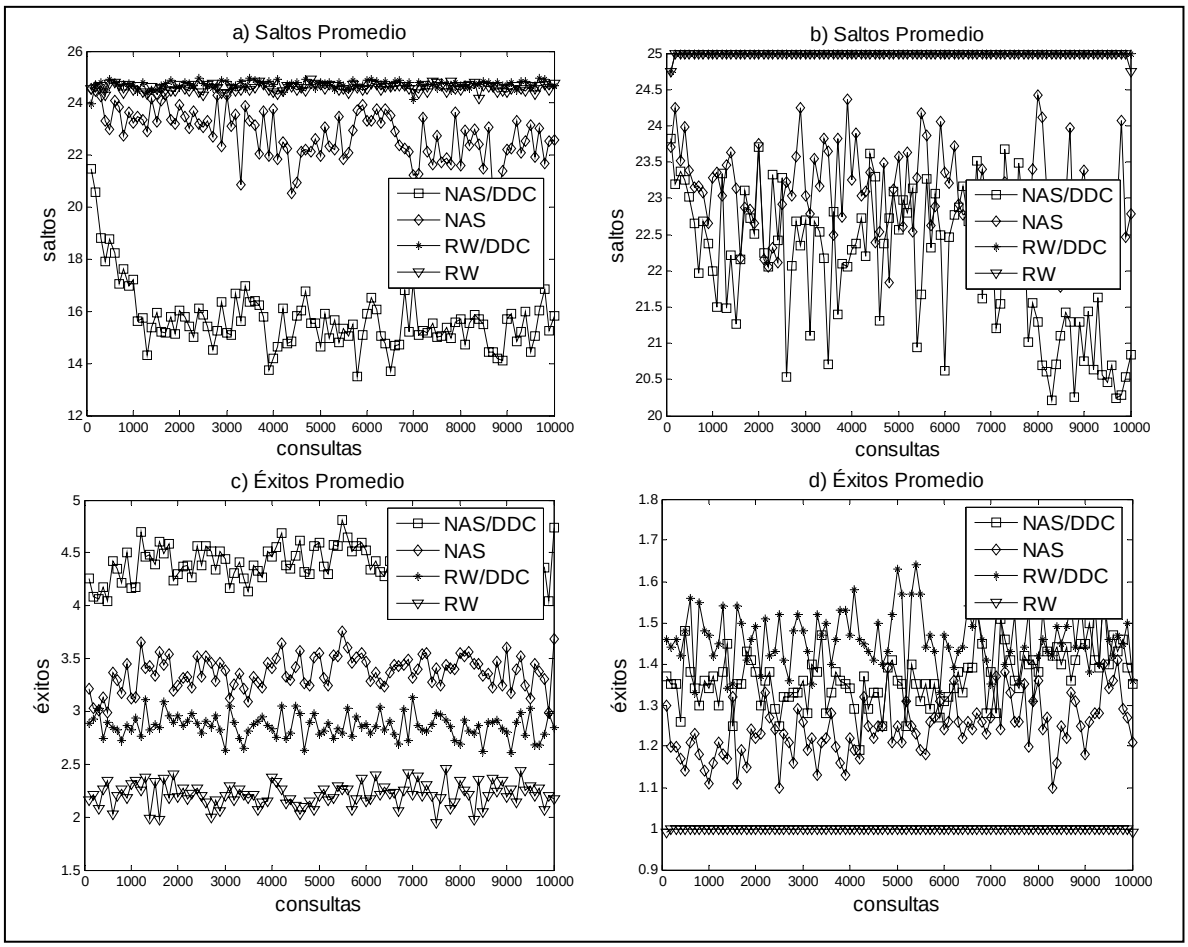

Figura 7.2. Gráficas de resultados sobre una red Small-world cada 100 consultas de las métricas a) saltos promedio, b) tasa de éxitos promedio de los algoritmos NAS y Random-walk con y sin DDC. Y resultados sobre una red Scale-free cada 100 consultas de las métricas c) saltos promedio, d) tasa de éxitos promedio de los algoritmos NAS y Random-walk con y sin DDC.

Análisis de Resultados

Las observaciones confirman la intuición de que la métrica topológica local DDC, siendo una medida de diversidad de grado, no proporciona ventajas significativas en topologías con distribuciones de grado Poissonianas, tales como las del modelo Watts-Strogatz. Sin embargo, al aplicar esta métrica en presencia de redes de distribución Scale-free permite una mejora significativa en el desempeño de la búsqueda, lo cual también implica que el algoritmo NAS funciona mejor en tales topologías en comparación con los métodos existentes que no incorporan información estructural local. Con esto podemos concluir que al aplicar la

información estructural en el algoritmo la cantidad de saltos necesarios para satisfacer una consulta se reduce hasta en un 50% y que en topologías de tipo Scale-free la cantidad de resultados encontrados se incrementa hasta en un 15% y un 5% en redes de tipo Small-world.

7.4. Evaluación de la aportación del DDC y lookahead en el algoritmo NAS

Conforme la experimentación ha avanzado se han integrado elementos que corresponden al comportamiento de los sistemas del mundo real. Con esta información se ha realizado el siguiente experimento para evaluar la aportación de la métrica topológica local DDC y la técnica de exploración local lookahead. A continuación se muestran los resultados obtenidos de la prueba de desempeño del algoritmo NAS con ambos métodos.

Objetivo

Mostrar la aportación de la métrica topológica local DDC y del método de búsqueda local Lookahead en el algoritmo de búsqueda NAS.

Descripción

Para este experimento fue creada una red de 1,024 nodos de tipo scale-free con el modelo de Barabási-Albert. Los recursos contenidos en los nodos corresponden a tópicos de la base de datos de ACM Computing Classification System (ACM CCS), la cual contiene 910 tópicos. Estos tópicos fueron replicados aproximadamente 30 veces cada uno, los cuales fueron ubicados con una distribución Zipfiana. Cada unidad de tiempo fue tomada cada 100 milisegundos, los experimentos se llevaron a cabo durante 10,000 unidades de tiempo. Durante cada unidad de tiempo fue asignado a cada uno de los nodos, el 10% de probabilidad uniforme de lanzar una consulta, eligiendo aleatoriamente el recurso a buscar del total de tópicos existentes en el ACM CCS.

Resultados

La Figura 7.3(a) muestra los saltos promedios para desempeñar un conjunto de consultas con diferentes configuraciones del algoritmo NAS: NAS básico, NAS con DDC, NAS con

Lookahead y NAS con DDC y Lookahead. Para los algoritmos NAS básico y con DDC el comportamiento es el mismo: la cantidad de saltos promedios se mantiene en 25 saltos desde el comienzo hasta el final. Para NAS con lookahead el comportamiento evoluciona, comenzando en 20 saltos promedio y alrededor de la consulta 2,000 la cantidad de saltos se reduce hasta un rango de 16 a 14. NAS con DDC y lookahead comienza con 18 saltos promedio y alrededor de la consulta 2,000 la cantidad de saltos se reduce en un rango de 14 a 12 saltos.

La Figura 7.3(b) muestra la tasa promedio de éxitos para desempeñar un conjunto de consultas con diferentes combinaciones de NAS. Para los algoritmos NAS básico con y sin DDC el comportamiento es el mismo: la tasa promedio de éxitos se mantiene alrededor de 0.3 recursos encontrados por consulta desde el inicio hasta el final. Para NAS con lookahead el comportamiento evoluciona comenzando en 4.2 recursos encontrados por consulta y alrededor de la consulta 500 la tasa de éxitos se incrementa y se mantiene en un rango entre 4.9 y 5. Para NAS con lookahead y DDC el comportamiento es el mismo que NAS con lookahead.

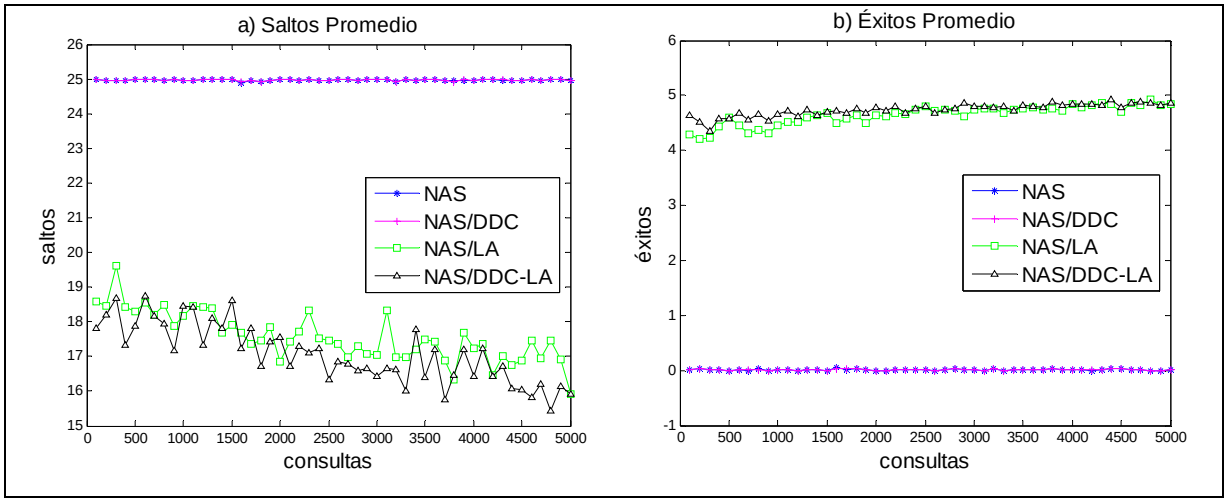

Figura 7.3. Gráficas de resultados del algoritmo NAS con diferentes configuraciones: NAS básico, NAS con DDC, NAS con lookahead y NAS con lookahead y DDC, en a) saltos promedio y b) tasa promedio de éxitos.

Análisis de Resultados

Los resultados del experimento confirman que la combinación del DDC y el lookahead, en presencia de la distribución de red Scale-free permiten una mejora significativa en el desempeño de la búsqueda. Al incluir información estructural local y aprendizaje en el algoritmo la cantidad de saltos se reduce en un 90% desde el comienzo hasta el estado estable reportado, y que la eficiencia se incrementa en 90% en términos de la cantidad de la cantidad de saltos necesarios para encontrar un recurso. Estos resultados resultan en una mejora del 90%. También se muestra que se alcanza un buen desempeño promedio en la tasa de éxitos: entre un 90% y 95%.

Como se observó, los mejores resultados fueron obtenidos con la combinación de ambas estrategias. En el caso del DDC que es una métrica topológica local su comportamiento es dirigido hacia nodos con mejor conexión y en ese punto busca información desde el nodo, sin tomar en cuenta los contenidos de los nodos vecinos, siendo esta la principal diferencia con el lookahead, ya que éste, toma información requerida por la consulta en el nodo y en los nodos vecinos de primer nivel. La combinación de estas estrategias permite que la búsqueda sea dirigida hacia regiones con vecindarios densos, lo cual incrementa la probabilidad de encontrar los recursos solicitados.

7.5. Evaluación de las Trazas de elegibilidad

Las trazas de elegibilidad reemplazante son un mecanismo usado en el aprendizaje reforzado para manipular la recompensa atrasada y las variables responsables de almacenar información acerca de que tan recientemente un estado fue visitado. La función de las trazas es incrementar la velocidad de convergencia hacia la ruta óptima. Cada vez que un estado es visitado comienza un proceso de memoria de corto término, con lo cual una traza decae gradualmente con el tiempo. Esta traza permite al estado ser elegible para ser tomado en cuenta para el aprendizaje debido a que permite más exploración para mantener la traza dentro de un rango de valores que no permite a otros estados ser predominantes [Sutton 1996]. A continuación se muestra el resultado de las muestras de las trazas de elegibilidad del algoritmo NAS.

Objetivo

Comprobar el comportamiento de los niveles de feromona del algoritmo NAS con respecto a la teoría de las trazas de feromona de los algoritmos de hormigas.

Descripción

La configuración de este experimento es la misma que para el experimento 7.4.

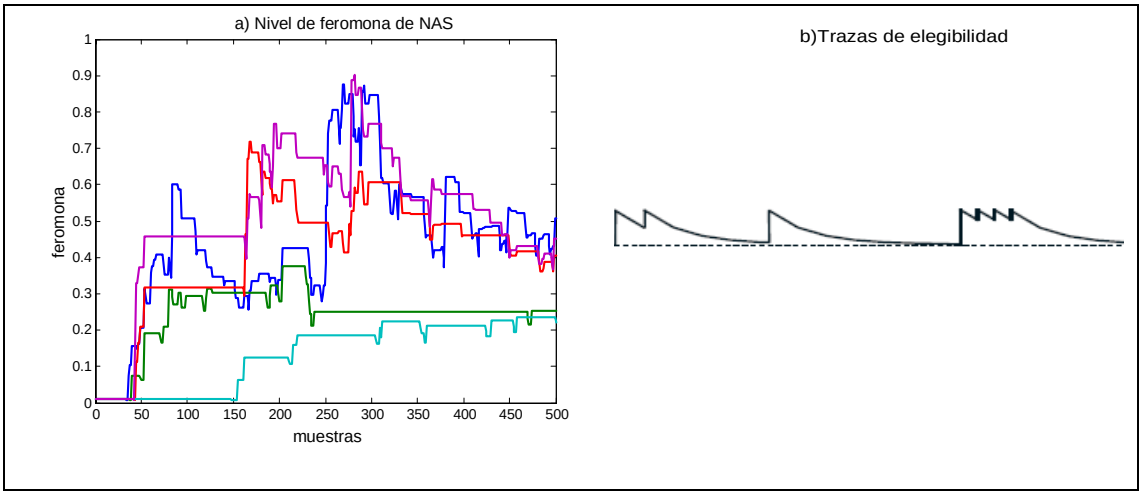

Figura 7.4.Trazas de elegibilidad a) del algoritmo NAS y b) creadas por Sutton.

Resultados

La Figura 7.4(a) muestra el nivel de feromona de cinco aristas (seleccionadas aleatoriamente) en una red de 1,024 nodos y 14,250 enlaces. En este experimento fueron tomadas las primeras 500 muestras, y fueron observadas cada vez que las hormigas realizaron una actualización sobre la tabla de feromonas. Los incrementos en las líneas que corresponden a las conexiones entre los nodos reflejan la implementación de la función de actualización global (Ecuación 6.7) indicando la bondad del resultado obtenido. El decremento en la feromona en estas conexiones corresponde al uso de la función de actualización local (Ecuación 6.6, conocida también como evaporación local), cuyo objetivo es disminuir el nivel de feromona de las conexiones por donde viajan las hormigas, para mantener en un rango estable los niveles y no permitir que ciertas conexiones sean más predominantes que otras. Las líneas que no presentan

cambios corresponden a aquellas conexiones que no fueron seleccionadas en ninguna ruta llevada a cabo por las hormigas.

Análisis de Resultados

La grafica en la Figura 7.4(a) muestra que la memoria de aprendizaje del algoritmo NAS (feromona), sigue un comportamiento como el de las trazas reemplazantes de elegibilidad de Sutton como se muestra en la Figura 7.4(b). En NAS, una vez que se alcanza la estabilidad, el rango de valores que puede alcanzar el nivel de feromona se mantiene entre un valor muy pequeño, sin llegar a ser cero y un valor máximo de 2.5.

7.6. Evaluación del algoritmo NAS con instancias P2P reales

En los experimentos anteriores se han ido añadiendo elementos que acercan el funcionamiento del sistema al comportamiento real de los sistemas P2P complejos. Como última evaluación se presenta la aplicación del algoritmo NAS a instancias de prueba reales. A continuación se muestran los resultados obtenidos.

Objetivo

Evaluar el desempeño del algoritmo NAS con instancias P2P reales.

Descripción

Para este experimento fue creada una red de 1,024 nodos de tipo scale-free con el modelo de Barabási-Albert. Los recursos contenidos en los nodos corresponden a tópicos de canciones en una red Gnutella de 2,000 usuarios [Goh 2005], de la cual fueron extraídos 1,024 de manera aleatoria para esta experimentación. La cantidad de tópicos manejados (títulos de canciones) es 1,066 y tienen una distribución Zipf, es decir, la frecuencia de los datos en la red varía desde 14 hasta 203 datos por tópico. La distribución de los tiempos de lanzamiento de cada consulta es determinada por una distribución Zipf. Los recursos a buscar fueron elegidos con una distribución Zipf. En resumen, la distribución de las consultas, los contenidos en los repositorios y en el tiempo de lanzamiento en las consultas es de tipo Zipfiana, ya que es la distribución que se encuentra en las aplicaciones del mundo real [Liu 2004]. Las mediciones

fueron realizadas de dos maneras: por tiempo (cada 10,000 milisegundos) y por consulta (cada 100 consultas).

Resultados

La Figura 7.5 muestra los resultados obtenidos medidos cada 100 consultas finalizadas. La Figura 7.5(a) muestra la tasa de éxitos promedio, inicialmente el algoritmo comienza obteniendo 4.3 recursos encontrados y conforme evoluciona el algoritmo la cantidad de recursos encontrados se encuentra en un rango de 4.9 a 5 recursos encontrados.

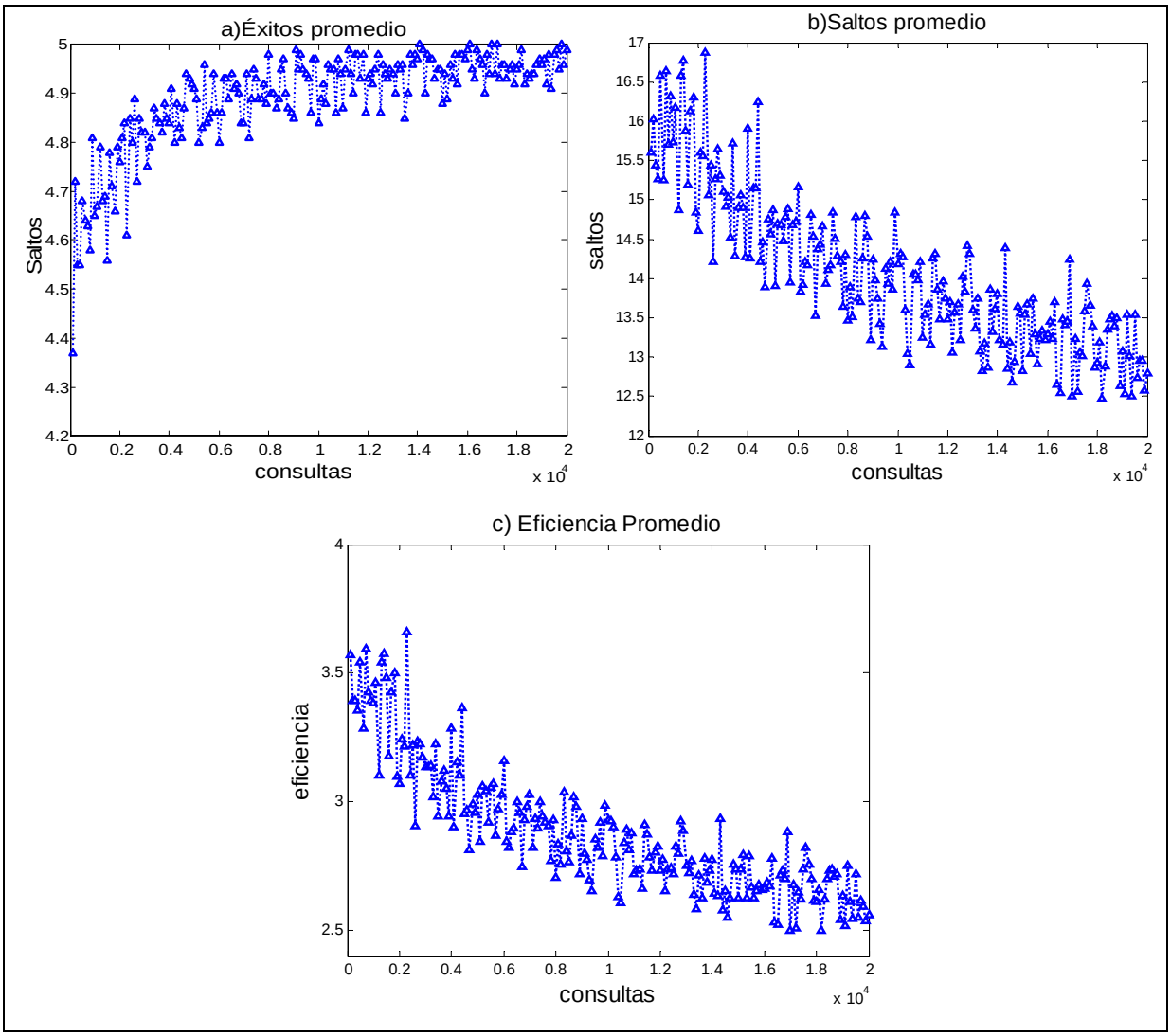

Figura 7.5. Gráficas de resultados del algoritmo NAS con instancias reales en a) Tasa promedi0 de éxitos y b) Saltos promedio y c) Eficiencia promedio medidas en consultas

La Figura 7.5(b) muestra los saltos promedios utilizados por consulta, en un inicio el algoritmo comienza con alrededor de 17 saltos y conforme evoluciona el algoritmo la cantidad de salto disminuye hasta 11 saltos. En la Figura 7.5(c) se muestra la eficiencia del algoritmo, que representa la cantidad de saltos necesaria para encontrar un recurso en la red, en la etapa inicial comienza necesitando alrededor de 3.7 saltos para encontrar un recurso y conforme evoluciona el algoritmo, la cantidad de saltos necesario disminuya hasta 2.2 saltos para encontrar un recurso.

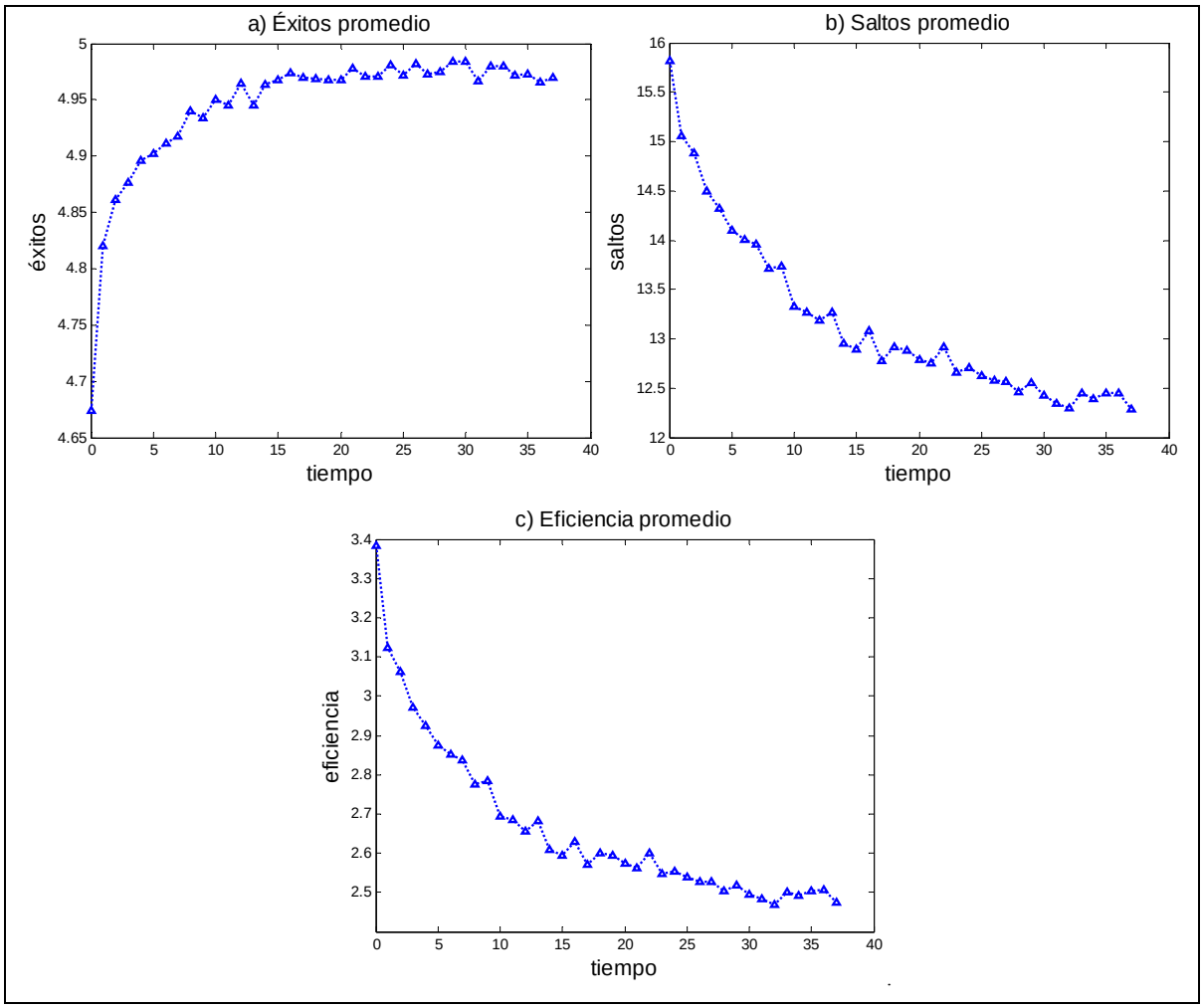

Figura 7.6. Gráficas de resultados del algoritmo NAS con instancias reales en a) Tasa promedio de éxitos y b) Saltos promedio y c) Eficiencia promedio medidas en tiempo de ejecución cada 10,000 milisegundos.

La Figura 7.6 muestra los resultados obtenidos medidos en tiempo cada 10,000 milisegundos. La Figura 7.6(a) muestra la tasa de éxitos promedio, inicialmente el algoritmo comienza obteniendo 4.67 recursos encontrados y conforme evoluciona el algoritmo la cantidad de recursos encontrados se encuentra en un rango de 4.95 a 5 recursos encontrados.

La Figura 7.6(b) muestra los saltos promedios utilizados por consulta, en un inicio el algoritmo comienza con alrededor de 16 saltos y conforme evoluciona el algoritmo la cantidad de salto disminuye hasta 12 saltos. En la Figura 7.6(c) se muestra la eficiencia del algoritmo, que representa la cantidad de saltos necesaria para encontrar un recurso en la red, en la etapa inicial comienza necesitando alrededor de 3.4 saltos para encontrar un recurso y conforme evoluciona el algoritmo la cantidad de saltos necesario disminuya hasta 2.5 saltos para encontrar un recurso.

Análisis de Resultados

Como se puede observar en la experimentación, el desempeño del algoritmo de búsqueda con las instancias reales se mantiene cercano a los resultados obtenidos con las instancias artificiales generadas. En general el desempeño del algoritmo obtiene un 75% de mejora en comparación con sus resultados iniciales, lo que significa que el algoritmo aprende las rutas que son más cortas hacia los recursos buscados. Las mediciones realizadas a través de las consultas terminadas y con el tiempo varían en pequeñas proporciones pero en general se comportan de la misma manera, la diferencia es que las medidas tomadas en cuestión del tiempo son dependientes del hardware y software utilizado, con lo que se llega a la conclusión que la medición en cuestión en consultas proporciona ventajas en términos de la reproducibilidad del experimento, ya que no es dependiente de ninguno de los elementos anteriormente mencionados.

Capítulo 8

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se presentan las aportaciones de esta investigación y algunas directrices para el desarrollo de futuros trabajos.

8.1 Conclusiones

El presente trabajo de investigación propone un sistema multi-agente para mejorar la solución del problema de enrutamiento de consultas semánticas en redes P2P no estructuradas con una topología Scale-free. Dada una red de nodos interconectados con una distribución Scale-free, un conjunto de consultas y un conjunto de datos almacenados en el repositorio de cada nodo, ambos con una distribución Zipfiana, el objetivo del sistema multi-agente es reducir la cantidad de saltos necesarios para encontrar la mayor cantidad posible de recursos solicitados.

A la fecha, los algoritmos propuestos para enrutamiento de consultas semánticas son pocos. La mayoría carece de fundamentos asociados con los algoritmos metaheurísticos de aprendizaje, que por su naturaleza de auto-adaptación ofrecen mayores posibilidades de encontrar resultados óptimos en redes complejas y de gran escala, como lo son los sistemas P2P. Estos se enfocan principalmente en la parte de filtrado de resultados o recuperación de la información. Por otra parte la mayoría los algoritmos de hormigas aplicados a este problema

son difíciles de reproducir ya que no son muy claros tanto en su fundamentación, algoritmos, experimentación y métricas de evaluación. La principal diferencia entre el algoritmo NAS propuesto y los trabajos mencionados radica en que este algoritmo toma ventaja de la topología local de la red de tal manera que permite discriminar que nodo proporciona mayor información.

El algoritmo NAS fue adaptado de dos algoritmos de hormigas: *ACS* y *SemAnt* y enriquecido con la métrica de caracterización topológica local, DDC.

El algoritmo NAS obtiene un 75% de reducción en el número de saltos por consulta, el cual se determina en función de su etapa inicial de búsqueda hasta su etapa estable de aprendizaje. Además, NAS logra un 20% de mejora en cuestión de los recursos encontrados. En general, comparado con métodos de búsqueda ciegos como el conocido Random Walk, el algoritmo NAS mejora los resultados obtenidos en un 60% tanto en la cantidad de saltos por consulta como en la cantidad de recursos encontrados.

Las principales contribuciones de esta tesis son las siguientes:

a) Se generaron instancias de prueba para redes P2P, cada una formada por tres elementos: la red de nodos y conexiones, las consultas emitidas por los nodos y el contenido de los repositorios de los nodos. Todas con distribuciones aleatorias y del mundo real.
b) Se crearon tres funciones heurísticas de caracterización local para ser aplicadas en la evaluación de los resultados obtenidos por el agente de búsqueda.
c) Para la solución del problema de enrutamiento de consultas semánticas se adaptaron y modificaron las funciones heurísticas del algoritmo de colonia de hormigas ACS, enfocado originalmente a la solución del TSP.
d) Se incluyó la aplicación de la métrica topológica local *DDC* y la función de exploración *lookahead*, en las funciones heurísticas del sistema multi-agente para obtener ventaja de la topología de red. Los mejores resultados fueron obtenidos al combinar ambas estrategias propuestas. En el caso del DDC, que es una métrica

topológica, su comportamiento es dirigido hacia nodos mejor conectados, sin tomar en cuenta los contenidos de los vecinos, siendo ésta la principal diferencia con la técnica "lookahead", ya que ésta busca la información solicitada por la consulta en el nodo actual y en los vecinos de primer nivel. La combinación de estas estrategias permiten que la búsqueda se dirija hacia regiones con vecindarios densos, lo cual incrementa la probabilidad de encontrar más resultados.

e) Se diseñaron la arquitectura y las estructuras de datos del sistema multi-agente propuesto, así como sus parámetros de configuración.
f) Se implementó un nuevo sistema multi-agente con las características mencionadas, el cual obtuvo en general una mejora del 75% en el desempeño, en función de su etapa inicial de búsqueda hasta su etapa estable de aprendizaje aplicando métricas locales.

8.2 Trabajos Futuros

Para dar continuidad a este trabajo de investigación, a partir de la experiencia obtenida se proponen los siguientes trabajos:

a) Definir el tamaño de la vecindad. En esta tesis se considera que la vecindad es de diámetro 1.
b) Estudiar profundamente el impacto de la métrica topológica local en el aprendizaje del sistema multi-agente y su efecto en las medidas de saltos y tasa de éxitos.
c) Contemplar el uso de más de una hormiga por consulta.
d) Implementar de manera dinámica: la entrada y salida de nodos en la red y cambios en los datos de los repositorios.
e) Analizar el impacto de cada uno de los parámetros del algoritmo.
f) Diseñar una estrategia de administración de la memoria que permita evaluar instancias más grandes.

REFERENCIAS

[Adamic 2000] Adamic L. y Huberman B., "Power-Law Distribution of the World Wide Web", *Science*, vol. 287. No. 5461, p. 2115, 2000.

[Albert 2000] Albert R., Jeong H. y Barabási A.L., "Error and Attack Tolerance of Complex Networks", *Nature*, vol. 506, pp. 378~382, 2000.

[Albert 2002] Albert R. y Barabási A.L., "Statistical Mechanics of Complex Networks" *Reviews of Modern Physics*, vol. 74, No.1, pp. 47~97, 2002.

[Amaral 2000] Amaral L.A.N., Scala A., Barthelemy M. y Stanley H.E., "Classes of Small World Networks", *Proceedings of the National Academy of Sciences*, vol. 97, No. 21, pp. 11149~11152, 2000.

[Amaral 2004] Amaral L.A.N. y Ottino J.M., "Complex Systems and Networks: Challenges and Opportunities for Chemical and Biological Engineers", *Chemical Engineering Scientist*, vol. 59, pp. 1653~1666, 2004.

[Androutsellis-Theotokis 2004] Androutsellis-Theotokis S. y Spinellis D., "A Survey of Peer-to-Peer Content Distribution Technologies", *ACM Computing Surveys*, vol. 36, No. 4, 2004.

[Ardenghi 2007] Ardenghi J., Echaiz J., Cenci K., Chuburu M., Friedrich G., García R., Gutierrez, de Matteis L. y Caballero J.P., "Características de Grids vs. Sistemas Peer-to-Peer y su posible Conjunción", *IX Workshop de Investigadores en Ciencias de la Computación (WICC 2007)*, ISBN 978-950-763-075-0, pp. 587~590, 2007.

[Arenas 2003] Arenas A., Cabrales A., Díaz-Guilera A., Guimerá R., y Vega-Redondo F., "Search and Congestion in Complex Networks", *Proceedings of the Conference "Statistical Mechanics of Complex Networks"*, 2003.

[Arora 2007] Arora S., y Barak B., *Complexity Theory: A Modern Approach*, Libro en preparación para ser publicado por la Prensa de la Universidad de Cambridge, 2007.

[Asmar 2005] Asmar D.C., Elshamli A. y Areibi S., "A comparative assessment of ACO algorithms within a TSP environment", *Dynamics of Continous Discrete and Impulsive Systems-Series B-Applications & Algorithms*, vol. 1, pp. 462~467, 2005.

[Babaöglu 2002] Babaöglu O., Meling H. y Alberto Montresor, "Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems", *In Proceedings of the 22nd International Conference on Distributed Computing Systems* (*ICDCS 02*), IEEE, 2002.

[Balakrishnan 2003] Balakrishnan H., Kaashoek M.F., Karger D., Morris R., Stoica I., "Looking up data in p2p systems", *Communications of the ACM*, vol. 46 No. 2, pp. 43~48, 2003.

[Barabási 1999] Barabási A.L., Albert R., Jeong H., "Mean-Field theory for Scale-free Random Networks", *Physica A*., vol. 272 pp. 173~189, 1999.

[Barabási 2003] Barabási A.L., *Emergence of Scaling in Complex Networks*. Handbook of Graphs and Networks, Wiley-VCH, pp. 69~82, 2003.

[Bergman 2001] Bergman M, "The Deep Web: Surfacing Hidden Value", *The Journal of Electronic Publishing*, vol. 7, 2001.

[Bloom 1970] Bloom B.H., "Space/time trade-offs in hash coding with allowable errors" *Communication of ACM*, vol. 13, No. 7, pp. 422~426, 1970.

[Bollobás 2002] Bollobás B. y Riordan O.M., *Mathematical Results on Scale-free Random Graphs*, Handbook of Graphs and Networks, Wiley-VCH, pp. 1~32, 2002.

[Bray 2000] Bray T., Paoli J., Sperberg-McQueen C.M. y Maler E., "Extensible Markup Language (XML) 1.0, W3C Recommendation", http://www.w3.org/TR/2000/REC-xml-20001006/, 2002.

[Chen 2005] Chen H., Gong Z. y Huang Z., "Self-learning Routing in Unstructured P2P Network", *International Journal of Information Technology*, vol. 11, No. 12, pp. 59~67, 2005.

[Chervenak 2000] Chervenak A., Foster I., Kesselman C., Salisbury C. y Tuecke S., "The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets", *J. Net. Comput. Appl.*, vol. 23, No. 3, pp. 187~200, 2000.

[Chrysakis 2006] Chrysakis I. y Plexousakis D., *Semantic Query Routing and Distributed Top-k Query Processing in Peer-to-Peer Networks*, Reporte Técnico. Institute of Computer Science – FORTH, 2006.

[Clip2 2000] Clip2.com, "The gnutella protocol specification v0.4", http://www9.limewire.com/developer/gnutella protocol 0.4.pdf, 2000.

[Crespo 2002] Crespo A. y Garcia-Molina H., "Routing indices for peer-to-peer systems", *In Proceedings of the 22nd International Conference on Distributed Computing Systems* (*ICDCS 02*), IEEE, 2002.

[Cormen 2001] Cormen T., Leiserson C., Rivest R. y Stein C., *Introduction to Algorithms*, The MIT Press, McGraw Hill, 2001.

[DARPA 2008] Defense Advanced Research Project Agency, http://www.darpa.mil, 2008.

[Daswani 2000] Daswani S. y Fisk A., "Gnutella UDP Extension for Scalable Searches (GUESS) v0.2", https://www.limewire.org/fisheye/browse/~raw,r=1.1.2.12/limecvs/core/UnicastRFC.html, 2000.

[De Wolf 2005] De Wolf T. y Holvoet T., "Emergence Versus Self-Organisation: Different Concepts but Promising When Combined", *In Proceedings of the 3rd International Workshop on Engineering Self-Organising Applications* (*ESOA'05*), *4th International Joint Conference on Autonomous Agents and Multi-Agent Systems* (*AAMAS'05*), pp. 1~15, 2005.

[Demers 1987] Demers A., Greene D., Hauser C., Irish W., Larson J., Shenker S., Sturgis H., Swinehart D. y Terry D., "Epidemic algorithms for replicated database maintenance", *In Sixth Annual ACM Symposium on Principles of DistributedComputing*, pp. 1~12, 1987.

[DMOZ 2004] DMOZ open directory, http://www.dmoz.org/, 2004.

[Di Caro 1998] Di Caro G. y Dorigo M., "AntNet: Distributed Stigmergy Control for Communications Networks", *Journal of Artificial Intelligence Research* (*JAIR*), vol. 9, pp. 317~365, 1998.

[Di Caro 2004] Di Caro G., Ducatelle F. y Gambardella L.M., "AntHocNet: An Ant-based Hybrid Routing Algorithm for Mobile and Ad Hoc Networks", *In Proceedings of Parallel Problem Solving from Nature*, vol. 3242, 2004.

[Dorigo 1997] Dorigo M. y Gambardella L.M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation*, vol.1, No.1, 1997.

[Dorigo 2004] Dorigo M. y Stützle T., *Ant Colony Optimization*. MIT Press, 2004.

[Erdős 1959] Erdős P. y Rényi A., "On random graphs. I.", *Publ. Math. Debrecen*, vol. 6, pp. 290~297, 1959.

[Erdős 1960] Erdős P. y Rényi A. "The Evolution of Random Graphs", *Magyar Tud. Akad. Mat. Kutató Int. Közl*, vol. 5, pp. 17~61, 1960.

[Faloutsos 1999] Faloutsos M., Faloutsos P. y Faloutsos C., "On Power-Law Relationship of the Internet Topology", *ACM SIGCOMM Computer Communication Review*, vol. 29, No. 4, pp. 251~262, 1999.

[Foster 2003] Foster I., "The Grid: Computing Without Bounds", *Scientific American*, vol. 288, No. 4, p. 78, 2003.

[Gilbert 1959] Gilbert E.N., "The Annals of Mathematical Statistics", vol. 30, No. 4, pp. 1141~1144, 1959.

[Goh 2005] Goh S.T., Kalnis P., Bakiras S. y Tan K. "Real Datasets for File-sharing Peer-to-Peer Systems", *Lecture Notes in Computer Science*, vol. 3453, No. 12, pp. 201~213, 2005.

[Goldberg 1990] Goldberg D. y Deb K., "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", *In Proceedings of the 1st Workshop on Foundations of Genetic Algorithms*, pp. 69~93, 1990.

[Gómez 2007] Gómez C., *Algoritmos de Aproximación para Problemas de Búsqueda en Redes Complejas*. Propuesta de Tesis Doctoral, CICATA, Altamira, México, 2007.

[Google 2003] Google's Technical Highlights, http://www.google.com/intl/en/press/zeitgeist/archive2003.html, 2003.

[Grassé 1959] Grassé P.-P., "La reconstruction du nid et les coordinations inter-individuelles chez bellicoitermes natalenis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs", *Insect Sociaux*, vol. 6, pp. 41~81, 1959.

[Grega 2007] Grega M., Kluska B., Leszczuk M., Papir Z., *Content-based Search for Peer-to-Peer Overlays*. Taller de Estudiantes de Doctorado (MedHocNet\\\'07), 2007.

[Guha 2005] Guha S. y Daswani N., *An Experimental Study of the Skype Peer-to-Peer VoIP System*. Reporte Técnico, Universidad de Cornell, Estados Unidos, 2005.

[Guillaume 2006] Guillaume J.-L., Latapy M., "Complex Network Metrology", Complex Systems Publications Inc., vol. 16, No. 1, pp. 83~94, 2006.

[Halm 2006] Halm M., "LionShare: Secure P2P Collaboration for Academic Networks", *EDUCAUSE Annual Conference*, 2006.

[Hoschek 2000] Hoschek W., Jaén-Martínez F. J., Samar A. , Stockinger H. y Stockinger K., "Data Management in an International Data Grid Project", *Proceedings of the First IEEE/ACM International Workshop on Grid Computing*, pp.77~90, 2000.

[Ivkovic 2001] Ivkovic I., "Improving Gnutella Protocol: Protocol Analysis and Research Proposals", *Prize-Winning Paper for LimeWire Gnutella Research Contest*, 2001.

[Jennings 1999] Jennings N.R., "Agent-oriented software engineering", *In Procedding of the 12th Internacional Conference on Industrial and Engineering Application of AI*, 1999.

[Kalogeraki 2002] Kalogeraki V., Gunopulos D. y Zeinalipour-Yazti D. "A Local Search Mechanism for Peer-to-Peer Networks", *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 300~307, 2002.

[Karvounarakis 2002] Karvounarakis G., Alexaki S., Christophides V., Plexousakis, D. y Scholl M., "RQL: A Declarative Query Language for RDF", *In Proceedings of the 11th International World Wide Web Conference*, 2002.

[Kocay 2005] Kocay W. y Kreher D., *Graphs, Algorithms and Optimization*. *Discrete Mathematics and its Applications*. Chapman and Hall/CRC, 2005.

[Kokkinidis 2004] Kokkinidis G. y Christophides V., "Semantic Query Routing and Processing in P2P Database Systems, The ICS-FORTH SQPeer Middleware", *Lecture Notes in Computer Science*, *Springer-Verlag*, vol. 3268, pp. 486~495, 2004.

[Lassila 1999] Lassila O. y Swick R., "Resource Description Framework Model and Syntax Specification, W3C Recommendation", http://www.w3.org/TR/REC-rdf-syntax/, 1999.

[Lawler 1985] Lawler E., Lenstra J., Hendrik A., Kan G. y Shmoys D., *The Traveling Salesman Problem: A Guided tour of Combinatorial Optimization*, Wiley Series in Discrete Mathematics & Optimization, JohnWiley & Sons, 1985.

[Lewin 1992] Lewin R., *Complexity, life at the edge of chaos*, Maxwell Macmillan International, 1992.

[Liang 2004] Liang J., Kumar R. y Ross. K., "Understanding KaZaA", http://cis.poly.edu/~ross/papers/UnderstandingKaZaA.pdf., 2004.

[Liu 2004] Liu Z., Wu I., Lee I. y Cho J., *Understanding Hidden-Web Traffic from The Perspective of A Metasearcher*, Reporte Técnico, Departamento de Ciencias de la Computación, Universidad de California, Los Angeles, 2004.

[Liu 2005] Liu J., XiaoLong J. y Kwok C.T., *Autonomy Oriented Computing /From Problem Solving to Complex System Modeling*, Springer Science + Business Media Inc, pp. 27~54, 2005.

[Lu 2007] Lu J. y Jamie Callan, "Content-based peer-to-peer network overlay for full-text federated search", *8th RIAO Conference on Large-Scale Semantic Access to Content* (*RIAO '07*), 2007.

[Lv 2002]Lv Q., Cao P., Cohen E., Li K., y Shenker S., "Search and Replication in Unstructured Peer-to-Peer Networks", *Proceedings of the 16th international conference on Supercomputing*, pp. 8495, 2002.

[Magkanaraki 2003] Magkanaraki, A., Tannen, V., Christophides, V., Plexousakis, D.: Viewing the SemanticWeb Through RVL Lenses. In Proceedings of the 2nd International Semantic Web Conference (ISWC), 2003.

[Matias 2002] Matias F. y Nguyen T., *Text-based content search and retrieval in ad hoc P2P communities*. Reporte Técnico DCSTR-483, Universidad de Rutgers, Estados Unidos, 2002.

[Menascé 2002] Menascé D. y Kanchanapalli L., "Probabilistic Scalable P2P Resource Location Services", SIGMETRICS Perf. Eval. Review, pp. 48~58, 2002.

[Meza 2002] Meza E., *Red de Observaciones y Predicciones de Variables Oceánicas (ROPVO) en las Costas y Puertos del Golfo de México*. Propuesta proyecto CONACyT, 2002.

[Michlmayr 2007] Michlmayr E., *Ant Algorithms for Self-Organization in Social Networks*. Tesis Doctoral, Women's Postgraduate College for Internet Technologies (WIT), Institute of Software Technology and Interactive Systems, Universidad de Tecnología deVienna, 2007.

[Mihail 2007] Mihail M., Saberi A., Tetali P., "Random Walks with Lookahead in Power Law Random graphs", *Internet Mathematics*, vol. 3, pp. 1~3, 2007.

[Mondéjar 2006] Mondéjar R., Pujol J., Garcia P. y Pairot C., *Sistemas multi-agente en entornos P2P*, Reporte Técnico DEIM-RR-06-002, Departament d'Enginyeria Informàtica i Matemàtiques, 2006.

[Newman 2005] Newman M, "Power laws, Pareto distributions and Zipf's law", *Contemporary Physics*, vol. 46, No. 5, pp. 323~35, 2005.

[Newman 2006] Newman M.E.J., Barabási A.L. y Watts, D.J., "The Structure and Dynamics of Networks", *Princeton University Press*, 2006.

[Ortega 2007] Ortega R., Meza E., Gómez G., Cruz L., Turrubiates T., "Impact of Dynamic Growing on the Internet Degree Distribution", *In Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, *Lecture Notes in Computer Science*, vol. 4743, pp. 119~122, 2007.

[Preiss 1999] Preiss B., *Data Structures and Algorithms with Object-Oriented Design Patterns in Java* (*Worldwide Series in Computer Science*), John Wiley & Sons ISBN 0471-34613-6, 1999.

[Ripeanu 2002] Ripeanu .M y Foster I., "Mapping the Gnutella network: Macroscopic properties of large-scale peer-to-peer systems", *In Peer-to-Peer Systems, First International Workshop, IPTPS 2002*, vol. 2429, pp. 85-93, 2002.

[ROAD 2005] ROAD (Repast Organization for Architecture and Design). Repast Home Page, http://repast.sourceforge.net, 2005.

[Rohrs 2001] Rohrs R., "Query routing for the Gnutella network", http://rfc-gnutella.sourceforge.net/src/qrp.html, 2001.

[Rubinstein 1981] Rubinstein R., *Simulation and the Monte Carlo Method*, Wiley, ISBN 0-471-08917-6, 1981.

[Sakaryan 2004] Sakaryab G., *A Content-Oriented Approach to Topology Evolution and Search in Peer-to-Peer Systems*, Tesis Doctoral, Universidad de Rostock, Alemania, 2004.

[Saroiu 2003] Saroiu S., Gummadi K. y Gribble S., "Measuring and analyzing the characteristics of Napster and Gnutella hosts", *Springer-Verlag New York, Inc*., vol. 9, No. 2 pp.170~184, 2003.

[Schaeffer 2006] Schaeffer S.E., *Algorithms for Non-uniform Networks*, Tesis Doctoral, Laboratorio de Tecnología para Ciencia Computacional Teórica de la Universidad de Helsinki, 2006.

[Schelfthout 2003] Schelfthout K. y Holvoet Tom, "A Pheromone-Based Coordination Mechanism Applied in Peer-to-Peer", *In 2nd International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2003)*, vol. 2872, pp. 71~76, 2003.

[Sen 2003] Sen P., Dasgupta S., Chatterjee A., Sreeram P.A., Mukherjee G. y Manna S.S., "Small-world Properties of the Indian Railway Network", *Physical Review E.*, vol 67, 2003.

[Shahabi 2007] Shahabi C., "Modelling P2P data networks under complex system theory", *International Journal of Computational Science and Engineering*, vol. 3, No.2, pp. 103~111, 2007.

[Stokes 2003] Stokes M, "Gnutella2 Specifications Part One"
http://www.gnutella2.com/gnutella2 search.htm, 2003.

[Sutton 1996] Sutton R. y Singh S., "Reinforcement Learning with Replacing Eligibility Trace", *Kluwer Academic Publisher*, pp. 36, 1996.

[Sycara 1998] Sycara K., "Multiagent systems", *AI Magazine*, vol. 19, No. 2, pp. 79~92, 1998.

[TechWeb 2001] TechWeb, http://www.techweb.com/wire/story/twb20010110s0020, 2001.

[Tempich 2004] Tempich C., Staab S. y Wranik A., "REMINDIN': Semantic Query Routing in Peer-to-Peer Networks Based on Social Metaphors" *In W3C, Proceedings of the 13th International World Wide Web Conference* (*WWW 2004*), pp. 640-649, 2004.

[TREC 2002] Trec REtrieval Conference, http://trec.nist.gov/, 2002.

[Tsoumakos, 2003] Tsoumakos D. y Roussopoulos N. "A comparison of Peer-to-Peer Search Methods", *International Workshop on the Web and Databases* (*WebDB*), 2003.

[Turrubiates 2007] Turrubiates T., *Clasificador de redes complejas basado en métricas que caracterizan diferentes Topologías*. Tesis de Maestría, Instituto Tecnológico de Cd. Madero. Cd. Madero, Tamps, México, 2007.

[Watts 1998] Watts D.J. y Strogatz S. "Collective dynamics of 'Small-World' networks" *Nature*, vol. 393, No. 6684, pp. 440~442, 1998.

[Witten 1999] Witten I., Moffat A. y Bell T., *Managing Gigabytes: Compressing and Indexing Documents and Images*, Morgan Kaufmann, second edition, 1999.

[Wooldridge 2002] Wooldridge M., "An introduction to multiagent systems". John Wiley Ed., 2002.

[Xu 2001] Xu J.-M., "Topological Structure and Analysis of Interconnection Networks", *Kluwer Academic Publishers*, 2001.

[Yang 2002] Yang B. y García-Molina H., "Efficient Search in Peer-to-Peer Networks", *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pp. 271~272, 2004.

[Yang 2003]Yang B. y García-Molina H. "Designing a super-peer network", *Proceedings 19th International Conference*, 2003.

[Yang 2006] Yang K., Wu C. y Ho J., "AntSearch: An Ant Search Algorithm in Unstructured Peer-to-Peer Networks", *IEICE Transactions on Communications*, vol. 89, No. 9, pp. 2300~2308, 2006.

[Yu 2005] Yu H., Li H., Wu P., Agrawal D., Abbadi A.E., "Efficient Processing of Distributed Top-k Queries", *In Proceedings of the 16th International Conference on Database and Expert Systems Applications* (*DEXA 2005*), 2005.

[Zhou 2004] Zhou Y., Croft W., Levine B., *Content-based search in peer-to-peer networks*, Reporte Técnico IR-367, Universidad de Masachusets, Estados Unidos, 2004.