INSTITUTO TECNOLÓGICO
DE CIUDAD MADERO

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN

Tesis:

Ajuste Adaptativo de un Algoritmo de Enrutamiento de Consultas Semánticas en Redes P2P.

Para obtener el grado de:

Maestro en Ciencias en Ciencias de la Computación

Presenta:

I.S.C. Gilberto Rivera Zárate

Número de Control: G02071037

Director

Dra. Claudia Guadalupe Gómez Santillán

Co-Director

Dra. Elisa Satu Schaeffer

Ciudad Madero, Tamaulipas, México. Diciembre 2009

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

SECRETARÍA DE EDUCACIÓN PÚBLICA

SEP

Cd. Madero, Tamps; a 26 de Noviembre de 2009.

OFICIO No.: U5.439/09
AREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. ING. GILBERTO RIVERA ZARATE
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 Maestría en Ciencias en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"AJUSTE ADAPTATIVO DE UN ALGORITMO DE ENRUTAMIENTO DE CONSULTAS SEMÁNTICAS EN REDES P2P"

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

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

c.c.p.: Archivo

MYCHC 'NLCO 'aygc*

"2009, Año de la Reforma Liberal"

Ave. 1o. De Mayo y Sor Juana I. De la Cruz, Col. Los Mangos, C.P. 89440 Cd. Madero, Tam.
Tels. (833) 3 57 48 24, Fax: (833) 3 57 48 20, Ext. 1002, email: itcm@itcm.edu.mx
www.itcm.edu.mx

Resumen

En el área de las Ciencias Computacionales, el *problema de ajuste de parámetros* se define como la selección de los valores de los parámetros que regulan el comportamiento de los algoritmos. Es muy conocido que la falta de un ajuste adecuado en los parámetros consume una gran cantidad de recursos humanos y computacionales. Debido a su relevancia en los ámbitos científicos y tecnológicos, el problema ha recibido mucha atención. Sin embargo, son pocos los trabajos que abordan el ajuste de parámetros en condiciones dinámicas y de gran escala como lo es la búsqueda de recursos de información en la Internet.

En este trabajo se aborda el ajuste de parámetros del algoritmo AdaNAS propuesto para el *Direccionamiento de Consultas Semánticas* (SQRP, por sus siglas en inglés) en la Internet. El algoritmo busca los nodos que contienen información relacionada con una palabra clave, y su objetivo es minimizar la cantidad de pasos y maximizar la cantidad de recursos encontrados (éxitos) para mejorar la eficiencia de sistemas de consulta en redes punto a punto (P2P).

El algoritmo de búsqueda **AdaNAS** se basa en un sistema de colonias de hormigas para guiar a un grupo de *agentes que aprenden de forma adaptativa*. Los agentes a su vez son algoritmos que navegan en la Internet de manera autónoma con el objetivo de adquirir experiencia durante la búsqueda la cual se registra mediante métodos, funciones y reglas. Cada agente tiene asociado un valor para el parámetro ***Tiempo de Vida*** (*TTL,* por sus siglas en inglés), el cual define su periodo de existencia y es calculado a través del número de pasos dados.

La **función *f*** propuesta para el *TTL* de AdaNAS incluye: *caracterización local de la estructura de* SQRP (métricas topológicas), *aprendizaje de corto alcance* del desempeño parcial obtenido en consultas previas y el *aprendizaje de largo alcance* del desempeño final obtenido también en consultas previas (tabla de feromonas). La feromona es una sustancia utilizada por hormigas reales para registrar información de los caminos recorridos durante la búsqueda de comida; los mejores caminos son favorecidos con mayor cantidad de feromona.

El problema de ajuste de parámetros ha sido abordado usando métodos globales que tienen una naturaleza muy distinta al enfoque local que se presenta en este trabajo, y para el cual se ha acuñado el término *control dinámico local*. En la literatura, al ajuste de parámetros durante la ejecución de un algoritmo se le denomina control dinámico. El término local se ha añadido porque todas las tareas involucradas son locales: caracterización del entorno, procesamiento de información histórica de desempeño y ajuste de parámetros. Este enfoque permite la **adaptación** a cambios frecuentes en los datos de entrada, ya que esto se considera un escenario común en sistemas complejos del mundo real. Esta es un área emergente que tiene mucho camino por explorar.

La estrategia de **ajuste de control adaptativo** propuesta, muestra que ***es factible resolver de manera eficiente*** *el problema de direccionamiento de consultas semánticas en la Internet, modelado como un sistema complejo, controlando adaptivamente el parámetro TTL del algoritmo AdaNAS, mediante el uso de información* ***local.*** El *algoritmo adaptativo AdaNAS* al ser comparado con algoritmos de la literatura que utilizan estrategias de ajuste global obtuvo una mejora significativa, la eficiencia medida en éxitos por paso incrementó en un 49%.

Abstract

In the area of Computer Science, the problem of tuning parameters is defined, as the selection of the parameter values that regulate the behavior of the algorithms. It is well known that the lack of an adequate tuning on the parameters consumes a great amount of human and computational resources.

Due to its importance in the scientific and technological areas, the problem has received a lot of attention. However, few work approaches the tuning of the parameters in dynamic conditions and great scale; such as the search of information resources in the internet.

In this work, the tuning of the parameter of the Internet Semantic Query Routing (SQR) algorithm is approached . The algorithms searches nodes that contain information relationed with a key word with the objective to minimize the amount of steps and to maximize the amount of resources found to improve the efficiency of the SQRP.

The search algorithm **AdaNAS** is based on an ant colony system that guides a group of agents who learn in an auto-adaptive manner. Agents are algorithms that surf autonomously on the internet with the objective to acquire experience during the search registrated by means of methods, functions and rules. Each agent has an associated value for the **TTL** (Time-To-Live) parameter which defines its existence period calculated by the number of given steps.

The **function *f*** proposed for TTL of AdaNAS includes: local characterization of SQRP structure (topological metrics), short-range learning of partial performance obtained in previous queries and long-range learning of final performance also obtained in previous queries (pheromone table). Pheromone is a substance used by real ants to registrate information of paths taken during food search; best paths are benefited with greater amount of pheromone.

The tuning parameter problem has been approached using global methods which have a different nature than the local approach presented in this work; the term " local dynamic control " has been designated. In literature, the tuning of parameters during algorithm execution is called dynamic control. The local term has been added because all of the tasks involved are local: environment characterization, historical information of the performance process and parameter tuning. This approach allows the **adaptation** to frequent changes in the input data. Frequent changes are a common scenario in complex systems of the real world. This is an emergent area with a long way to go.

The proposed strategy for adaptive control tuning shows that it is feasible to resolve in an efficient way the problem of semantic query routing (SQRP) on the Internet, modeled as a complex system, adaptively controlling the TTL parameter of AdaNAS algorithm, by means of local information. The adaptative algorithm AdaNAS obtained a significant improvement compared to algorithms in literature which use global adjustment strategies, the efficiency obtained in successes per step **increased** in a 49%.

TABLA DE CONTENIDO

Página

Introducción

Motivaciones 1
Contexto de la Investigación 2
Direccionamiento de Consultas Semánticas (SQRP) 2
Definición Formal de SQRP 3
Problema de Investigación: Ajuste de Parámetros 5
Hipótesis 7
Objetivos 7
Organización del Documento 8

1 Algoritmos de Búsqueda en Redes Complejas Punto-a-Punto (P2P)

1.1 Optimización Combinatoria 9
1.1.1 Problemas de Optimización Combinatoria 9
1.1.2 Algoritmos Metaheurísticos 10
1.2 Proceso de Búsqueda de Recursos sobre Redes P2P Complejas 10
1.2.1 Redes Complejas 11
1.2.2 Redes P2P Complejas 16
1.2.3 Proceso de Búsqueda Semántica 17
1.3 Algoritmos de Búsqueda en Redes P2P Complejas 17
1.3.1 Búsqueda por Caminatas Aleatorias (Random-Walk) 18
1.3.2 Búsqueda por Inundación (Flooding) 18
1.3.3 Búsqueda en Amplitud Inteligente 19
1.3.4 Búsqueda Adaptativa Probabilística: Colonias de Hormigas .. 19
1.4 Estado del Arte de las Búsquedas Semánticas en Redes P2P Complejas 23
1.5 Comentarios Finales 25

2 Ajuste de Parámetros del Algoritmo de Optimización

2.1 Clasificación del Ajuste de Parámetros 27

2.2 Técnicas de Ajuste de Parámetros Global 29
2.2.1 Análisis y Diseño de Experimentos (DOE) 29
2.2.2 Modelado Causal 31
2.3 Ajuste del Control Adaptativo de Parámetros: Aprendizaje Reforzado 34
2.3.1 La Propiedad de Markov 35
2.4 Estado del Arte: Ajuste del Control Adaptativo de Parámetros Aplicado a los Algoritmos de Hormigas y Consultas Semánticas 36

3 Metodología Propuesta para el Ajuste de Parámetros

3.1 Metodología de Solución 39
3.2 Integración de la Infraestructura de Referencia 42
3.2.1 Caracterización de Instancias Reales 42
3.2.2 Generación de Instancias 43
3.2.3 Desarrollo de Algoritmos de Referencia 45
3.3 Identificación de Factores de Desempeño y sus Relaciones 47
3.3.1 Selección de las Características de SQRP que Afectan el Rendimiento del Algoritmo NAS 47
3.3.2 Preparación del Algoritmo NAS para la Selección de Parámetros Significantes 49
3.3.3 Selección de los Parámetros Significantes de NAS 50
3.3.4 Modelado de las Relaciones entre los Parámetros del Algoritmo NAS y las Características de SQRP 51
3.4 Desarrollo de un Sistema de Búsqueda Adaptativo 52
3.5 Evaluación de la Eficiencia de los Algoritmos NAS y AdaNAS 52
3.5.1 Pruebas de Diferencias Significativas Usando Wilcoxon 53

4 Experimentación y Resultados

4.1 Ambiente Experimental 55
4.2 Integración de una Infraestructura de Referencia 56
4.2.1 Caracterización de Redes Reales 56
4.2.2 Creación de Generadores de Redes 60
4.2.3 Desarrollos de Algoritmos de Referencias 62
4.3 Identificación de Factores de Desempeño y sus Relaciones 74
4.3.1 Selección de las Características de SQRP, que afectan

el rendimiento del Algoritmo NAS 75
4.3.2 Preparación del Algoritmo NAS, para la Selección de los Parámetros Significantes 78
4.3.3 Selección de los Parámetros Significantes del Algoritmo NAS 79
4.3.4 Modelado de las relaciones entre los parámetros del algoritmo NAS y las características de SQRP 83
4.4 Desarrollo de un Sistema de Búsqueda Adaptativo 80
4.4.1 Diseño del Modelo Adaptativo Basado en Agentes 84
4.4.2 Desarrollo de un Algoritmo Adaptativo: AdaNAS 84
4.4.3 Diseño de la Función de Ajuste de Control Adaptativo del Parámetro TTL 92
4.5 Evaluación de la eficiencia del Algoritmo AdaNAS 91
4.5.1 Diferencias Significativas: Prueba de Wilcoxon 100

Conclusiones y Trabajos Futuros

Conclusiones 104
Aportaciones de la Investigación 111
Trabajos Futuros 113

Referencias

Anexo A: Redes Complejas 114

Anexo B: Complejidad de SQRP 115

Anexo C: Sistema Multiagente 116

128

145

Índice de Figuras

Figura 1.	Elementos que conforman SQRP	3
Figura 2.	Diagrama general del problema de ajuste de control adaptativo de parámetros	6
Figura 1.1.	Red aleatoria: a) estructura topológica, b) gráfica de la distribución del grado	14
Figura 1.2.	Red Scale-free: a) estructura topológica, b) gráfica de la distribución del grado	15
Figura 1.3.	Distribución y comunicación entre los nodos de una red P2P	16
Figura 1.4.	Clasificación de las Búsquedas	18
Figura 2.1.	Clasificación general del ajustes de parámetros	28
Figura 2.2.	Modelo general de un proceso	29
Figura 2.3	Objetivos de un diseño experimental	30
Figura 2.4	Aprendizaje por refuerzo	34
Figura 4.1.	Clasificación de los subgrafos de muestra INT – 1997 con la métrica topológica local DDC(*i*)	57
Figura 4.2.	Distribución del grado *P*(*k*) de la muestra INT – 1997	57
Figura 4.3.	Clasificación de los subgrafos de muestra INT – 2000 con la métrica topológica local DDC(*i*)	58
Figura 4.4.	Distribución del grado *P*(*k*) de la muestra INT – 2000	58
Figura 4.5.	Clasificación de los subgrafos de muestra INT – 2003 con la métrica topológica local DDC(*i*)	59
Figura 4.6.	Distribución del grado *P*(*k*) de la muestra INT – 2003	59
Figura 4.7.	Arquitectura General del Sistema Multiagente Propuesto	63
Figura 4.8.	Rendimiento del algoritmo NAS, a) promedio del porcentaje de éxitos, *b*) Promedio de saltos y c) Promedio de la eficiencia	68
Figura 4.9.	Comparación del algoritmo NAS contra el algoritmo Random-Walk. a) Promedio del porcentaje de éxitos , *b*) Promedio de saltos y *c*) Promedio de la eficiencia.	70
Figura 4.10.	Comparación de las estrategias del medio ambiente local de NAS. a) promedio del porcentaje de éxitos, b)	72

Promedio de saltos y c) Promedio de la eficiencia.

Figura 4.11.	Resultados de los efectos principales: *respuesta1* (HITS) y *respuesta2* (HOPS)	77
Figura 4.12.	Resultados de las interacciones: *respuesta1* (HITS) y *respuesta2* (HOPS)	77
Figura 4.13.	Resultados del rendimiento del algoritmo NAS para el DDC y el GRADO	81
Figura 4.14.	Resultados del rendimiento del algoritmo NAS para el DDC y el GRADO	81
Figura 4.15.	Diagrama de causalidad entre los parámetros y el desempeño de NAS.	84
Figura 4.16.	Estructura de la tabla ID_HOP	87
Figura 4.17.	Modelo Adaptativo de AdaNAS	90
Figura 4.18.	Modelo Adaptativo de AdaNAS.	91
Figura 4.19.	Estructuras principales del sistema multiagente	95
Figura 4.20.	Estructura de un nodo	96
Figura 4.21.	Estructura de un agente (hormiga).	97
Figura 4.22.	Estructura para la tabla de feromonas del algoritmo AdaNAS	98
Figura 4.23.	Estructura de Datos de la Tabla Lookahead para el algoritmo AdaNAS	99
Figura 4.24.	Estructura de la tabla *D*	100
Figura 4.25.	Estructura de la tabla *H*	100
Figura 4.26.	Comparación del promedio de éxitos de NAS contra AdaNAS.	106
Figura 4.27.	Comparación del promedio de los pasos de NAS contra AdaNAS.	107
Figura 4.28.	Comparación de la eficiencia de NAS contra AA_NAS.	107
Figura 4.29.	Resultado de la eficiencia promedio para NAS y AdaNAS, con las primeras 45 instancias.	110
Figura 4.30.	Resultado de la eficiencia promedio para NAS y AdaNAS, con 45 instancias (45-90)	110

Índice de Tablas

Tabla 1.1. Fase de inicialización del algoritmo Ant Colony System (ACS) 21
Tabla 1.2. Fase de construcción del algoritmo ACS 21
Tabla 1.3. Fase de actualización global del algoritmo ACS 22
Tabla 1.4. Fase de terminación del algoritmo ACS 22
Tabla 1.5. Algoritmo SemAnt 23
Tabla 1.6. Estado del arte: consultas semánticas y algoritmos de hormigas 25

Tabla 2.1. Estado del arte del ajuste de parámetros 38

Tabla 3.1. Etapas de la metodología general 40
Tabla 3.2. Descripción de las etapas con su correspondiente producto 41
Tabla 3.3. Método de caracterización que permite identificar la topología local o global de la red 42
Tabla 3.4. Algoritmo del modelo Erdos y Renyi 44
Tabla 3.5. Pseudocódigo del algoritmo NAS 46
Tabla 3.6. Pseudocódigo del algoritmo RW 47

Tabla 4.1. Información de muestras de la Internet para los años 1997, 2000 y 2003 56
Tabla 4.2. Información acerca de las redes generadas con el modelo Erdos-Rényi 55
Tabla 4.3. Información acerca de las redes generadas con el modelo de Barabási 56
Tabla 4.4. Configuración de los parámetros del algoritmo NAS 67
Tabla 4.5. Estimación de los efectos para la variable de respuesta de los éxitos 78
Tabla 4.6. Resultados generados de las experimentaciones de DDC y el grado 80
Tabla 4.7. Tabla de experiencia del nodo 86
Tabla 4.8. Algoritmo del agente de consulta 92
Tabla 4.9. Algoritmo de la hormiga de búsqueda 93
Tabla 4.10. Algoritmo de la hormiga de recuperación 94
Tabla 4.11. Algoritmo de la hormiga de actualización 94
Tabla 4.12. Tabla de feromonas del algoritmo NAS 97
Tabla 4.13 Estructura de Datos para Lookahead del algoritmo NAS 98

Tabla 4.14. Configuración de los parámetros del algoritmo NAS 104
Tabla 4.15. Configuración de los parámetros del algoritmo AdaNAS 105
Tabla 4.16. Resultados del algoritmo NAS, aplicado a 10 instancias 108
Tabla 4.17. Resultados del algoritmo AdaNAS, aplicado a 10 instancias 108
Tabla 4.18. Resultados del % de mejora del promedio de la eficiencia, al comparar los algoritmos NAS y AdaNAS 109

INTRODUCCIÓN

En esta tesis se aborda el problema de ajuste de parámetros usando estrategias de control adaptativo, aplicadas a un algoritmo de optimización que resuelve el problema de direccionamiento de consultas semánticas sobre la Internet. En las secciones de este capítulo se presenta un panorama general de la tesis; el cual inicia con la descripción de los motivos que llevaron a esta investigación, continuando con la descripción del contexto del direccionamiento de consultas semánticas y la definición del problema de investigación. Posteriormente se presentan la hipótesis y los objetivos planteados para el ajuste de parámetros, terminando con una breve descripción de cada uno de los capítulos de la tesis.

Motivaciones

Los parámetros de una función necesitan ajustarse para obtener resultados de buena calidad. Las funciones matemáticas son de gran utilidad en muchas disciplinas del conocimiento como pueden ser: las industriales, científicas y tecnológicas. Los investigadores frecuentemente invierten mucho tiempo en tratar de ajustar los parámetros de las funciones representadas como algoritmos que serán usados en la solución computacional de problemas [1,2].

Debido a las afirmaciones antes establecidas, la configuración de los mejores valores de los parámetros, se ha considerado por sí misma como un área de estudio que amerita más atención de la que ha recibido hasta este momento. Esta es un área donde el método científico y el análisis estadístico pueden y deben ser empleados [3]. Otros investigadores como Adenso-Díaz B. y Laguna M. [1], afirmaron que el 90% del tiempo que se invierte en la solución de un problema es consumido por el ajuste de parámetros y que en la actualidad este es un problema de investigación abierto [1,4,5].

Las técnicas para el ajuste de parámetros son aplicadas a una gran diversidad de problemas en diferentes áreas de estudio. Un área de investigación que ha estado tomando gran relevancia en los últimos años, es el estudio de los procesos que se llevan acabo sobre la Internet.

La *Internet* es una red de redes de computadoras distribuidas geográficamente y diseñadas como un sistema descentralizado [6,7,8]. Por sus características inherentes es considerada un sistema complejo [5,6,9].

Los *sistemas complejos* son definidos como aquellos sistemas que tienen grandes cantidades de elementos, las conexiones entre los elementos que lo forman evolucionan con el tiempo y los roles de los elementos pueden variar. Estos sistemas para su estudio son modelados como grafos o redes, dando lugar al concepto de *redes complejas*. Los parámetros del ambiente y de los elementos de estos sistemas, requieren ser configurados de manera dinámica y local para alcanzar la autonomía en presencia de condiciones que generalmente cambian de manera local [6,10].

El reto para esta área de investigación es dotar a los sistemas complejos de estrategias que les permitan auto configurarse de manera eficiente.

Contexto de la Investigación

Esta investigación es desarrollada en el contexto y terminología de los trabajos revisados en el estado del arte, aunado al uso de terminología propia. Por lo que se considera imprescindible normar conceptos que serán usados en esta área de investigación.

a) Direccionamiento de Consultas Semánticas (SQRP)

El problema de buscar información textual a través de palabras claves en la Internet es conocido como *Direccionamiento de Consultas Semánticas* (Semantic Query Routing Problem, SQRP). Este es un problema que ha venido tomando una gran relevancia con el crecimiento de las comunidades que comparten información de manera directa, conocidas como comunidades punto a punto (P2P, peer-to-peer), que trabajan sobre ambientes distribuidos y de gran escala como la Internet [5,8,11].

El objetivo de SQRP es determinar la ruta mas corta, desde un nodo que emite una consulta hasta donde están los nodos que pueden contestarla apropiadamente proporcionando la información requerida. La consulta viaja moviéndose a través de la red desde un nodo de inicio y después a un vecino, seguido del vecino del nodo vecino, y así sucesivamente hasta localizar el recurso requerido, o en caso contrario considerarlo ausente [5,12,13].

Como se observa en la Figura 1, a cada nodo *i* está asociada una base de datos o colección de documentos r_i (*repositorio*). Estos están disponibles a todos los nodos conectados en la red P2P. Un nodo busca información enviando mensajes a los nodos que están directamente conectados llamados vecinos. Las *consultas* c_j son enviadas por los nodos en forma de palabras claves, las cuales se usan para encontrar todos los documentos que contienen dichas palabras claves. Un nodo recibe un mensaje de consulta y lo evalúa entre sus documentos locales. Si la evaluación tiene éxito, el nodo actual genera una replica del mensaje, enviando su dirección al nodo que solicitó la búsqueda. Este proceso se repite hasta cumplir con los requerimientos de paro de la búsqueda de información [14,15].

La estructura del ambiente en el cual se lleva a cabo el proceso antes descrito se denomina *topología*, y se refiere al patrón de conexiones que forman los nodos de la red. La topología representa una de las principales dificultades de este problema debido a la posibilidad de que el patrón de conexiones cambie de un punto a otro, dando lugar a subgrupos heterogéneos en sus conexiones, los cuales pudieran causar efectos diferenciados en la búsqueda [5,16].

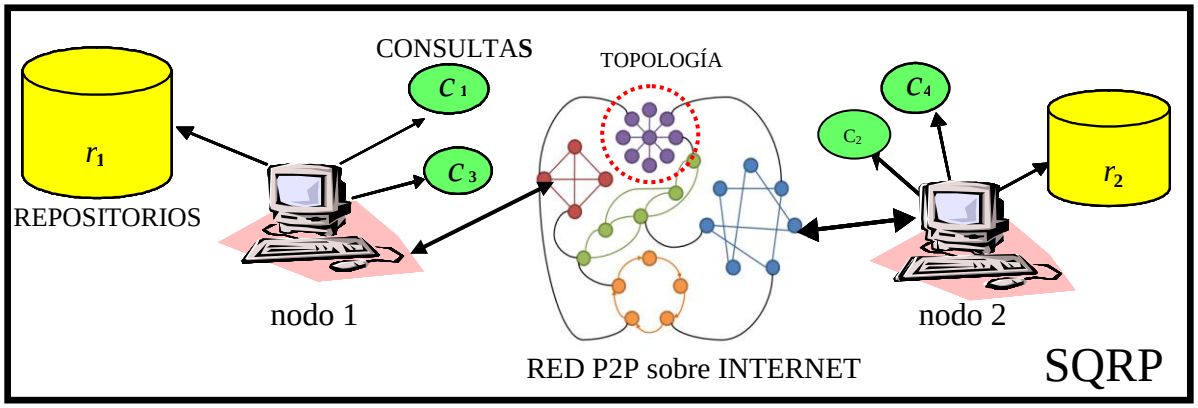


Figura 1. Elementos que conforman SQRP.

En general, los sistemas complejos como lo es SQRP involucran *elementos* como el *ambiente* (topología), *entidades* que interactúan en el sistema (nodos, repositorios y consultas) y una *función* objetivo (minimizar pasos y maximizar resultados) [6,17]. Cada uno de estos elementos contiene un conjunto de parámetros que capturan valores que representan el estado del sistema en un instante de tiempo. Dichos valores alimentarán los parámetros propios del proceso de búsqueda, activando estrategias necesarias para lograr el objetivo del proceso. En el anexo B se puede encontrar una descripción detallada de SQRP.

Debido a que el sistema bajo estudio es dinámico, los cambios en sus elementos, incluyendo la heterogeneidad de la topología, inciden directamente en el rendimiento del proceso de búsqueda [6,9,10]. En otras palabras, para dar una solución óptima a SQRP, los parámetros del algoritmo que implemente el proceso de búsqueda necesitan ser ajustados localmente usando estrategias de **ajuste de control adaptativo de parámetros**.

b) Definición Formal de SQRP

En la literatura especializada, la definición del problema SQRP no ha sido abordada formalmente. En este trabajo se presenta un modelo matemático para SQRP. En el anexo B se demuestra que este problema es NP-duro.

Dados

$G = (V,E)$	la red P2P modelada,
CC	la matriz de costos de transmisión,
$L = \{l_1, l_2 , l_3 , \ldots , l_z \}$	el diccionario de palabras clave,
$R = \{r_1, r_2, r_3, \ldots , r_n\}$	los repositorios locales de la red, donde $r_i=\{d_{i1}(L_1),d_{i2}(L_2),d_{i3}(L_3),\ldots,d_{izi}(L_s)\}$, $L_i \subseteq L$ y d_{ij} representa el j-ésimo documento almacenado en el nodo i y z_i es la cantidad máxima de documentos contenidos en r_i,
$C = \{c_1, c_2, c_3, \ldots, c_m\}$	las consultas emitidas por los usuarios, donde $\{v, k, tt_i\}$ con $v \in V$, $k \in L$ y tt_i es el instante en el cual se solicitó la consulta,
maxResult	cantidad mínima de documentos para satisfacer una consulta, y
TTL	costo máximo permitido en una consulta.

El problema SQRP consiste en determinar para cada consulta c_i una trayectoria $T \subset G$, que maximice la relación dada en la Ecuación (1):

$$Z_i = \frac{\sum_{c_i \in C} NumDoc(c_i, T)}{\sum_{c_i \in C} Costos(c_i)}; \qquad (1)$$

donde el Costo(c_i) es calculado por $Costo(c_i) = \sum_{u,v \in T} cc(u,v)$, y $NumDoc(c_i,T)$ esta dado por $NumDoc(c_i, T) = \sum_{v \in T} \sum_{d_v \in R_v} f(c_i, d_v)$.

Sujeta a las siguientes restricciones:

a) $NumDoc(c_i,T) > 0; \forall c_i \in C$ (se satisfacen todas las consultas C).
b) $NumDoc(c_i ,T) \geq maxResult; \forall c_i \in C$
c) $Costo(c_i) \leq TTL; \forall c_i \in C$

La dureza de un problema, como lo es SQRP, implica que *no existe un algoritmo exacto con complejidad polinomial que encuentre la solución óptima a dicho problema*. Además, la *cardinalidad del espacio de búsqueda de estos problemas suele ser muy grande*, lo cual hace inviable el uso de algoritmos exactos ya que la cantidad de tiempo que se necesitaría para encontrar una solución es inaceptable. Debido a estos dos motivos, se necesita utilizar algoritmos metaheurísticos que permitan obtener una solución de calidad en un tiempo razonable [27,28,31,34].

c) Problema de Investigación: Ajuste de Parámetros

El *ajuste de parámetros*, antes y durante la ejecución de un algoritmo de solución es clave para obtener soluciones de alta calidad en un tiempo razonable; impactando en la reducción significativa del tiempo de solución y uso de recursos [1,2,18]. Los investigadores que han estudiado este problema lo han hecho a través de la aplicación de estrategias que caracterizan el ambiente de manera global y local.

El ajuste **global** se aplica ***antes*** de la ejecución de un algoritmo, y por lo tanto, los parámetros toman valores promedio de los datos. Esto implica que los elementos del problema sean considerados homogéneos en todos los puntos sin distinguir los subgrupos inherentes. En otras palabras, se aplica la misma estrategia de solución aunque los elementos cambien de un punto a otro [5,18,19]. En uno de los trabajos que aplican esté tipo de ajuste se desarrolló la herramienta Calibra para la afinación de parámetros usando técnicas de diseño de experimentos. Calibra sólo permite afinar un máximo de cinco parámetros sin tomar en cuenta las relaciones entre ellos [1]. En otro trabajo se presenta una afinación manual de cinco parámetros a través de prueba y error, y tampoco se toman en cuenta las relaciones [5]. Las propuestas de este tipo resuelven parcialmente el problema del ajuste de parámetros, pero bajo condiciones dinámicas y heterogéneas las soluciones obtenidas son de baja calidad.

Bajo el enfoque de ajuste **local**, los parámetros se configuran **durante** la solución del problema [20]. Esta es una alternativa viable para aquellos problemas que presentan condiciones dinámicas y heterogéneas, ya que los valores de los parámetros requieren ser modificados en función de los cambios locales de los elementos del problema [21]. En la literatura se encontró que la mayoría de los trabajos que usan el enfoque de ajuste local lo aplican a algoritmos genéticos que resuelven problemas estáticos; el objetivo es adecuar los parámetros al estado actual de la solución y a características puntuales del problema. En esos trabajos no se considera la dinámica del problema, ni las características de entornos heterogéneos [20,21,22].

Dentro de la presente investigación, hasta este momento sólo se han identificado dos trabajos de ajuste local para condiciones dinámicas. El primero propone ajustar un parámetro a través de estructuras adaptativas; sin embargo, algunas medidas utilizadas en el ajuste son globales. Además, la estrategia de búsqueda se realiza mediante inundación de mensajes en la red; congestionando el tráfico y aumentando los costos de la búsqueda [23].

El segundo trabajo es aplicado a búsquedas de rutas para guiar automovilistas, haciéndose necesario conocer el inicio y el fin de la ruta. La técnica aplicada para su solución es el Filtro de Kalman, y es necesario que los datos del problema se ajusten a una distribución de probabilidad conocida [21]. Ambos trabajos usan información global de la red, lo cual no es aplicable a procesos que operan en Internet por su gran magnitud y dinamismo.

En esta tesis se propone una función de control adaptativo de parámetros utilizando **información local**. En la Figura 2, se presenta un esquema general de la propuesta. El ajuste de parámetros es para un algoritmo de búsqueda semántica en redes P2P. Inicialmente, para el nodo *j* y para cualquier consulta *i*, el algoritmo recibe los primeros valores de sus parámetros Θ_i configurados globalmente. Para localizar los documentos requeridos por una consulta *i*, el algoritmo transita entre varios nodos. El nodo *j*, proporciona información local x_j relacionada con elementos SQRP

(topología y repositorios). Con la información recibida se determina el desempeño parcial del algoritmo $z_{i,j}$, se actualizan las tablas de aprendizaje α_j del nodo j y se invoca a la función f que ajusta localmente los parámetros del algoritmo $\theta_{i,j}$ adaptándolos al nodo que se está evaluando. Para el ajuste utiliza información de los vecinos no visitados $\Gamma(x_j)$, el desempeño $z_{i,j}$ y lo aprendido en consultas previas α_j esta información también se usa para seleccionar el siguiente nodo a visitar. El desempeño final Z_i se determina cuando se alcanzan las condiciones de paro especificadas para la consulta.

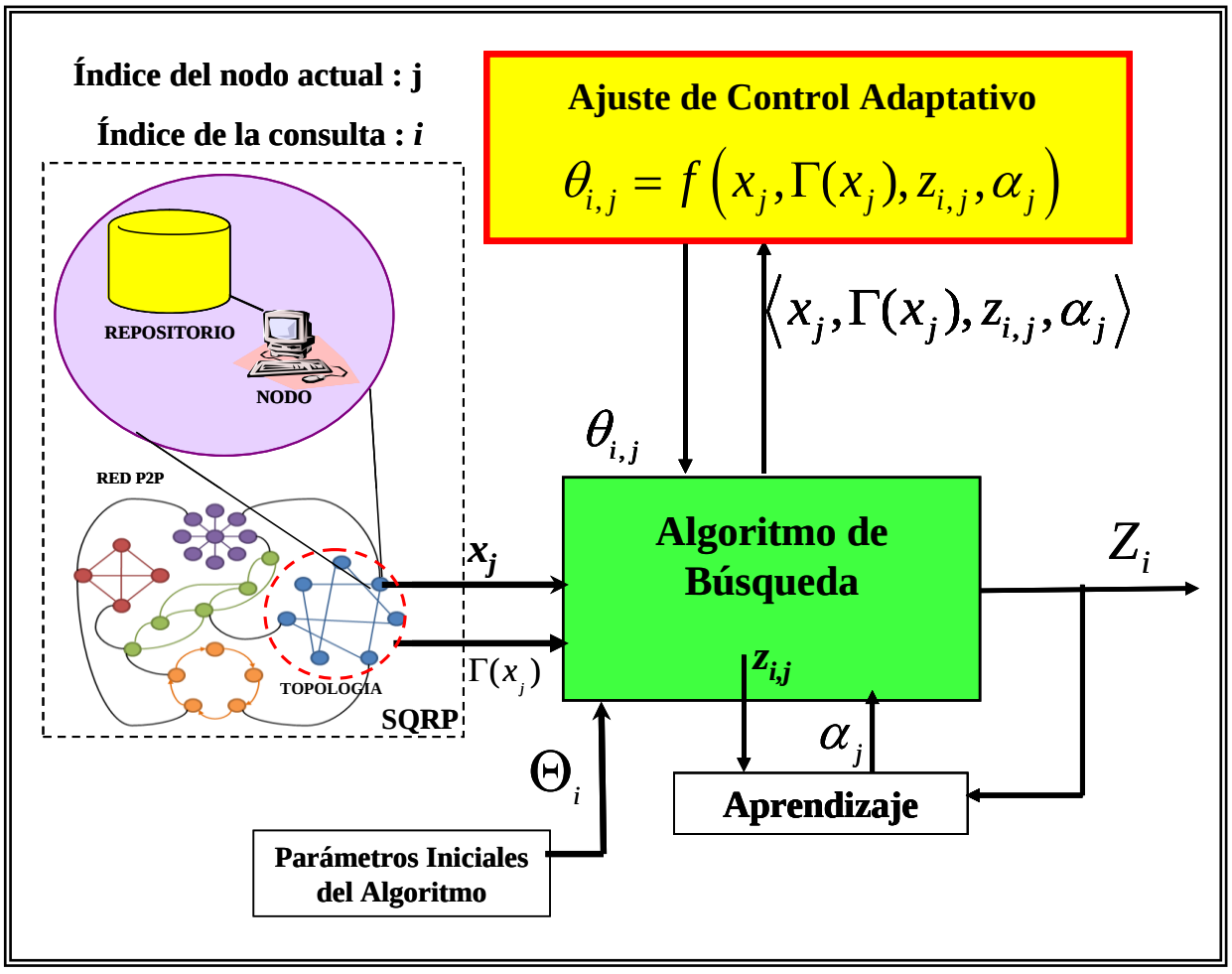


Figura 2. Diagrama general del problema de ajuste de parámetros con control local adaptativo.

La **función *f*** propuesta incluye: *caracterización local de la estructura del problema* SQRP (métricas topológicas), *aprendizaje de corto alcance* del desempeño parcial obtenido en consultas previas y el *aprendizaje de largo alcance* del desempeño final obtenido en consultas previas (tabla de feromonas). La feromona es una sustancia utilizada por hormigas reales para registrar información de los caminos recorridos durante la búsqueda de comida; los mejores caminos son favorecidos con mayor cantidad de feromona. Los algoritmos de colonias de hormigas implementan esta metáfora con estructuras de aprendizaje llamadas tablas de feromonas [24]. Por lo anterior, la búsqueda semántica abordada en esta tesis está basada en colonias de hormigas.

Para evitar el congestionamiento de la red cada consulta es ejecutada por una hormiga con una duración limitada, que en lo sucesivo llamaremos ***tiempo de vida*** (Time-To-Live, TTL).

Hipótesis

Para dar solución al problema de SQRP se propone modelar un sistema de consultas P2P el cual haga uso de una función de ajuste de control adaptativo para el tiempo de vida del proceso de consulta. Con este fin, el desarrollo de este trabajo sustenta la siguiente hipótesis:

> *Es posible resolver de manera eficiente SQRP* ***ajustando*** *los parámetros de un algoritmo de búsqueda basado en colonia de hormigas, mediante técnicas de* ***control adaptativo*** *que utilicen información* ***local*** *derivada de: la configuración inicial, el desempeño de consultas anteriores y las condiciones topológicas heterogéneas.*

Objetivos

A continuación se presentan el objetivo general y los objetivos específicos planteados para esta investigación.

a) Objetivo General

Formular una función de ***control adaptativo f*** *que determine de manera local el valor óptimo del* ***p****arámetro Tiempo de Vida* (***TTL***) *de un algoritmo de búsqueda basado en colonia de hormigas aplicado en la solución de SQRP.*

b) Objetivos Específicos

- *Diseñar camas de prueba* formadas por instancias reales y artificiales de SQRP.
- Diseñar un algoritmo metaheurístico básico para la solución de SQRP, que esté basado en colonias de hormigas y utilice estrategias de ajuste global de parámetros. Este algoritmo es denominado *Neighboring-Ant Search* (*NAS*).
- Identificar las variables relevantes de SQRP que son críticas para el desempeño del algoritmo metaheurístico NAS.
- Identificar las variables relevantes del algoritmo NAS que son críticas para su desempeño.
- *Rediseñar las estrategias de aprendizaje del algoritmo NAS,* utilizando el conocimiento adquirido sobre variables críticas para el desempeño. La nueva versión del algoritmo se denomina Adaptative Neighboring-Ant Search (AdaNAS).
- Diseñar una función de ajuste de control adaptativo para el parámetro *TTL* del algoritmo AdaNAS.

- Formular un modelo adaptativo de agentes de consulta del algoritmo AdaNAS.

- Demostrar la complejidad de SQRP utilizando la teoría de la NP_completitud.

Organización del Documento

La tesis está organizada como se describe a continuación:

En el Capítulo 1, se describen algoritmos de búsqueda que han sido propuestos para la solución de problemas relacionados con SQRP. Se inicia revisando conceptos de optimización combinatoria con la finalidad de determinar la complejidad de SQRP. Posteriormente se describen los elementos que intervienen en la búsqueda de recursos. Además se revisan los elementos del *ambiente* en el cual se lleva acabo el proceso de búsqueda, siendo las redes complejas uno de los aspectos más importantes. Se continúa con la descripción de técnicas de comunicación entre los nodos de la red, a través del modelo de redes punto a punto complejas. El capítulo finaliza con una discusión sobre los métodos de búsqueda analizados.

En el Capítulo 2 se revisan las técnicas de ajuste de parámetros: antes de la ejecución del algoritmo o ajuste global y durante la ejecución del algoritmo o ajuste de control adaptativo; así como las estrategias relacionadas con cada una de estas técnicas. El capítulo finaliza con una revisión al estado del arte para el ajuste de control adaptativo de parámetros aplicado principalmente a algoritmos de colonias de hormigas y consultas semánticas.

En el Capítulo 3 se detalla la metodología seguida para el ajuste de parámetros. Así mismo se describen los pasos necesarios para llevar acabo la afinación estática global del algoritmo NAS. Posteriormente se detalla el ajuste de control adaptativo de parámetros y la formulación de una función de ajuste adaptativa para el tiempo de vida (TTL) del algoritmo AdaNAS. Ambos algoritmos fueron propuestos para la solución de SQRP.

En el Capítulo 4 se describen el conjunto de experimentos diseñados para validar el cumplimiento de la hipótesis y los objetivos establecidos en esta investigación. Se presentan y discuten los resultados obtenidos para posteriormente establecer conclusiones sobre el conjunto de pruebas llevadas acabo.

En la última sección se presentan las conclusiones derivadas del desarrollo de esta investigación, además de las contribuciones obtenidas, terminando con la descripción de algunos posibles trabajos futuros con los que se puede dar continuidad a ésta investigación.

Capítulo 1

ALGORITMOS DE BÚSQUEDA EN REDES COMPLEJAS PUNTO-A-PUNTO (P2P)

En la actualidad los usuarios de Internet forman comunidades virtuales científicas y sociales para compartir recursos e información. La necesidad de compartir recursos a través de Internet, la cual es considerada un sistema complejo, ha motivado un incremento en la investigación de nuevas formas de organizar la estructura de las redes de comunicación. Las redes P2P han surgido como una respuesta para satisfacer los requerimientos de organización generados al compartir recursos en un ambiente virtual de colaboración entre organizaciones e individuos.

Actualmente existen varias alternativas de algoritmos de búsqueda que trabajan sobre redes complejas P2P, dentro de las cuales se pueden encontrar algoritmos de: caminatas aleatorias, inundación, tablas de índices, colonias de hormigas, entre otros. En esté capítulo se abordan los algoritmos de búsqueda más populares sobre las redes P2P y que han sido propuestos para la solución de SQRP.

1.1 Optimización Combinatoria

En matemáticas aplicadas y ciencias de la computación, la *optimización combinatoria* es una rama de la optimización, relacionada a la investigación de operaciones, teoría de algoritmos y teoría de la complejidad computacional. El análisis combinatorio es el estudio matemático del acomodo de un grupo, el ordenamiento o la selección de objetos discretos, usualmente finito en número cuando las combinaciones posibles son demasiadas para ser analizadas una por una [20,25,26].

Mediante el estudio de la teoría de la complejidad computacional es posible comprender la importancia de la optimización combinatoria. Los algoritmos de optimización combinatoria se relacionan comúnmente con problemas difíciles de resolver conocidos como NP-duros [27].

1.1.1 Problemas de Optimización Combinatoria

Este tipo de problemas se pueden definir como aquellos que consisten en encontrar la mejor solución dentro de un gran número de soluciones potenciales, para conseguir ciertos objetivos. Si

las configuraciones son discretas, el problema es *combinatorio*. La instancia está compuesta por un conjunto de configuraciones, un conjunto de restricciones y además una función objetivo [20,26,27,28].

El objetivo es identificar cuales de las configuraciones son *factibles*, es decir, las que cumplen con todas las restricciones, y generan el mejor valor de la función objetivo. Si se busca el mejor valor entonces se desea *maximizar;* o el menor valor y entonces se dice que se esta *minimizando*. La configuración factible con el mejor valor se llama *solución óptima* de la instancia [26,27].

Los algoritmos metaheurísticos son una técnica utilizada para la solución de problemas de optimización combinatoria ya que exploran grandes espacios de soluciones, encontrando soluciones aproximadas en tiempos razonables para problemas que son difíciles de resolver [29,30].

1.1.2 Algoritmos Metaheurísticos

Los metaheurísticos son procedimientos genéricos de alto nivel que guían a los heurísticos para explorar eficientemente el espacio de soluciones, ofrecen soluciones cercanas a la solución óptima del problema con una cantidad de recursos razonable. El término heurística está relacionado con el concepto de encontrar y, en el contexto de la inteligencia artificial, se refiere a la tarea de resolver de un modo inteligente problemas reales basándose en el conocimiento disponible. En la actualidad existe una gran diversidad de metaheurísticos disponibles para tratar de resolver un conjunto de problemas en diferentes áreas de la ciencia, la ingeniería y para las que no se dispone de algoritmos exactos que permitan encontrar soluciones de calidad en tiempos razonables [20,29,30,31].

Este tipo de métodos combinan ideas que provienen de cuatro distintos campos de investigación: las técnicas de diseño de algoritmos (resuelven una colección de problemas), algoritmos específicos (dependientes del problema que se quiere resolver), fuentes de inspiración (del mundo real) y métodos estadísticos. El término metaheurístico fue acuñado por Fred Glover [31]. Las propiedades fundamentales que las caracterizan son: plantillas genéricas que no son específicas de un problema en concreto. Son usadas para guiar el proceso de búsqueda, con el objetivo de explorar eficientemente el espacio de búsqueda y así encontrar soluciones cercanas a las óptimas. Lo anterior a través del uso de estrategias que van desde la búsqueda local hasta procesos complejos de aprendizaje [4,32].

Los metaheurísticos, son considerados actualmente herramientas prometedoras para la solución aproximada de instancias grandes de problemas NP-duros [20, 33,34]; como mencionó en la sección de introducción, SQRP pertenece a esta clase.

1.2 Proceso de Búsqueda de Recursos Sobre Redes P2P Complejas

El problema de búsqueda de recursos en redes P2P complejas, se ve afectado por varios factores los cuales hacen que el problema sea complejo. Uno de los principales factores es el ambiente en el cual se lleva acabo éste proceso en donde los usuarios realizan búsquedas de diferentes tipos de

recursos. En éstas búsquedas, las condiciones son variables es decir, las conexiones pueden cambiar, así como la información que se encuentra en los repositorios [5,35,36,37].

Las técnicas tradicionales tales como la caminata aleatoria y la inundación generan enormes cantidades de tráfico, mientras que los algoritmos basados en tablas fragmentadas (tablas hash) generan una sobrecarga de recursos para mantener un sistema de búsqueda en diferentes nodos. En contraste, las técnicas adaptativas, proporcionan un paradigma novedoso para controlar el tráfico de mensajes de consulta sobre redes P2P complejas [5,38,39].

1.2.1 Redes Complejas

Un *sistema* es un conjunto de elementos relacionados entre sí con un objetivo común. Los *sistemas complejos* son aquellos que tienen comúnmente un gran número de elementos, los cuales actúan de acuerdo a reglas que pueden cambiar a través del tiempo, es decir la conectividad de los elementos y sus roles son variables [10,40].

Los sistemas complejos son modelados de manera abstracta como grafos, siendo así como nace el término de *red compleja* ó *no uniforme* [41,42]. Este tipo de redes han ganado mucha popularidad ya que describen un amplio rango de problemas en la naturaleza y la sociedad [35,41,42,43]. Existen sistemas complejos naturales y artificiales. Por ejemplo, un sistema natural modelado como una red compleja es la estructura de una célula, las neuronas y enlaces químicos.

Dentro de los ejemplos de sistemas artificiales se encuentra la Web y la Internet [8,35,44]. Ambos pueden modelarse como una red compleja, donde los nodos o vértices son los elementos que conforman el sistema y las aristas o lados son las interacciones entre los elementos, conformando la topología de una red compleja que es el patrón que forman los nodos y las aristas de la red [45,46]. En el anexo A, se describen más conceptos relacionados con las redes complejas.

Funciones de Caracterización

Para el estudio y clasificación de las redes complejas se han desarrollado un conjunto de funciones de caracterización. Estas tienen el objetivo de encontrar y destacar las propiedades que caracterizan la estructura y el comportamiento de la red compleja, ofreciendo herramientas para medir propiedades; y así crear modelos de redes que ayuden a entender el significado de esas propiedades. Las funciones de caracterización pueden clasificarse en dos tipos [16,47,48]:

- Basadas en *información global*: requieren información de toda la red *G*.
- Basadas en *información local*: cada nodo *i* en la red *G*, tiene asociado un subgrafo G_i, formado por el nodo *i* y el conjunto de sus nodos vecinos ($\Gamma(i)$). Una función de caracterización local es una métrica que se calcula para cada G_i en *G*.

En SQRP las redes crecen y evolucionan por eventos locales, tales como la adición de nuevos nodos y enlaces, ó la eliminación de enlaces entre nodos. Cada vez que se genera un evento local la topología de la red cambia. Una forma de medir estos cambios es a través de las funciones de

caracterización ya que tienen como objetivo proveer información de la red en términos cuantitativos, y de esta forma caracterizar y analizar las redes complejas [17,19].

A continuación se describen algunas las funciones de caracterización local importantes para este trabajo de investigación, y que serán aplicadas en los experimentos posteriores.

a) Distribución del grado

El grado k_i de un nodo *i*, es el número de nodos adyacentes a *i*, y dos nodos son adyacentes si un lado los une. Sea $P(k)$ la fracción de nodos en la red que tienen grado *k*, es decir, $P(k)$ es la probabilidad de que un nodo seleccionado aleatoriamente tenga grado *k*. Los valores de $P(k)$ para $k \in [0, n-1]$ (suponiendo que puede haber al menos una conexión entre cada par de nodos distintos). Una gráfica de $P(k)$ para cualquier red dada puede formarse haciendo un histograma de los grados de los nodos, este histograma es la distribución de probabilidad del grado de una red y esta basada en información global, y generalmente es referida como *distribución del grado* [49,50,51].

Con base en lo anterior, definimos a X_k^G como el número de nodos en *G* con grado *k*. Así, la probabilidad de que un nodo en *G* tenga grado *k* se expresa por la Ecuación (1.1):

$$P(k) = \frac{X_k^G}{n} \tag{1.1}$$

b) Coeficiente de Dispersión del Grado (DDC por sus siglas en inglés)

El *DDC* [16,17,48,52] se define como una función basada en información local, la cual tiene como objetivo medir la dispersión entre el grado de un nodo *i* y $\Gamma(i)$. El *DDC* es formulado por la Ecuación (1.2) y para el nodo *i* se define como:

$$DDC(i) = \frac{\sigma(i)}{\mu(i)} \tag{1.2}$$

donde,

$$\sigma(i) = \sqrt{\frac{\sum_{j \in \Gamma(i)} [k_j - \mu_i]^2 + [k_i - \mu_i]^2}{N_i}} \tag{1.3}$$

y,

$$\mu(i) = \frac{\sum_{j \in \Gamma(i)} [k_j] + k_i}{N_i} \tag{1.4}$$

donde $\sigma(i)$ representa la variación del grado del conjunto $\{i \cup \Gamma(i)\}$, calculada con la Ecuación (1.3) y $\mu(i)$ es el grado promedio encontrado en el conjunto $\{i \cup \Gamma(i)\}$ y $N_i = |\{i \cup \Gamma(i)\}|$, calculada con la Ecuación (1.4). Además, k_i y k_j representan los grados de los nodos *i* y *j* respectivamente.

Con las funciones de caracterización definidas en esta sección se establecerán los tipos de redes complejas. Específicamente se usará la función de distribución del grado P(*k*) para la siguiente clasificación.

Tipos de Redes Complejas

De acuerdo con la distribución del grado P(*k*), las redes complejas se clasifican en: redes *aleatorias* (random) y redes de escala libre (*scale-free*), entre otras. Cabe señalar que en este trabajo sólo se describirán las características de las redes antes mencionadas ya que son las recomendadas por la literatura para modelar los elementos de SQRP [5,52,53,54].

a) Redes Aleatorias (Random)

A principios de los 80's diversas investigaciones arrojaron resultados sobre un grafo aleatorio generado con una probabilidad de conexión *p* y el grado k_i del nodo *i*. Este último sigue una distribución binomial con parámetros *n*-1 y *p*, como se observa en la Ecuación (1.5):

$$P(k) = \mathrm{C}_{n-1}^{k}\, p^k (1-p)^{n-1-k} = \binom{n-1}{k} p^k (1-p)^{n-1-k} \tag{1.5}$$

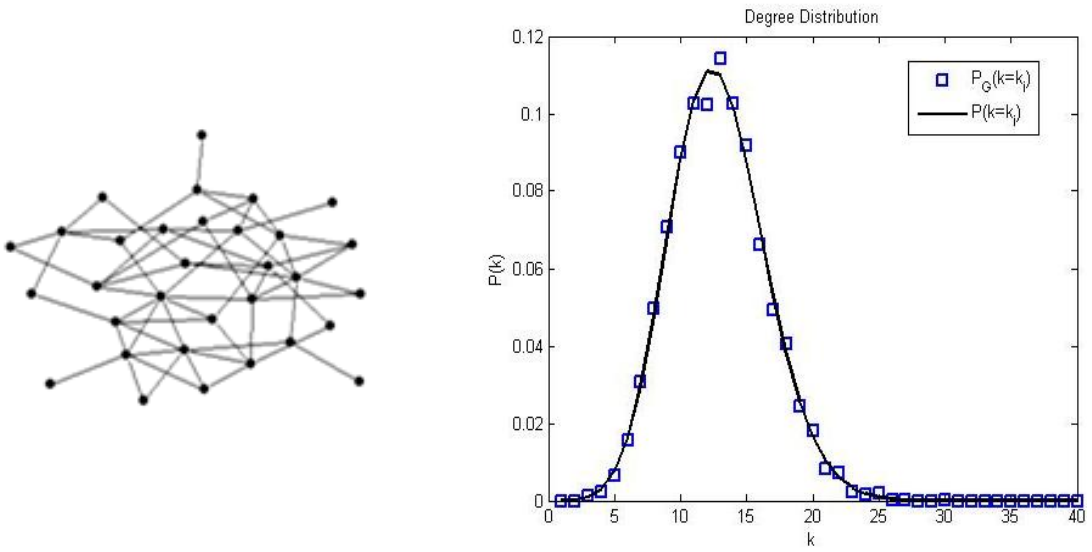
Para valores grandes de *n*, la distribución del grado sigue una distribución de Poisson, de ahí que la probabilidad de que un nodo tenga grado *k* es definida en la Ecuación (1.6):

$$P(k) \sim \frac{\langle k \rangle^k e^{-\langle k \rangle}}{k!} \tag{1.6}$$

donde $\langle k \rangle = 2e/n = p(n-1)$ [35,41,42].

Dado que en una red aleatoria las aristas entre los nodos son establecidas aleatoriamente, la mayoría de los nodos tiene aproximadamente el mismo grado, es decir el grado es cercano al grado promedio de la red $\langle k \rangle$. En la Figura 1.1.a, se observa la estructura topológica de una red

aleatoria y en la Figura 1.1.b, se muestra la gráfica de la distribución del grado obtenida para una red aleatoria con 3072 nodos, 19778 aristas, grado mínimo $\delta(G)$=3, grado máximo $\Delta(G)=30$, grado promedio $\langle k \rangle = 12$, los cuadros representan la distribución del grado real de la red y la línea continua representa la distribución del grado calculada con la Ecuación (1.6) y se observa como la distribución del grado sigue una distribución de Poisson.



a) estructura topológica b) distribución del grado

Figura 1.1. Red Aleatoria.

b) Redes de Escala Libre (Scale-free)

En la década de los 90's investigadores como Albert R., [55], Faloutsos M., [43] y Adamic L., [44] desarrollaron trabajos acerca de 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. Encontrando que las redes del mundo real exhiben una distribución del grado power-law, con valores en el intervalo $2 \leq \gamma \leq 4$ y esté tipo de red es generada con la Ecuación (1.7):

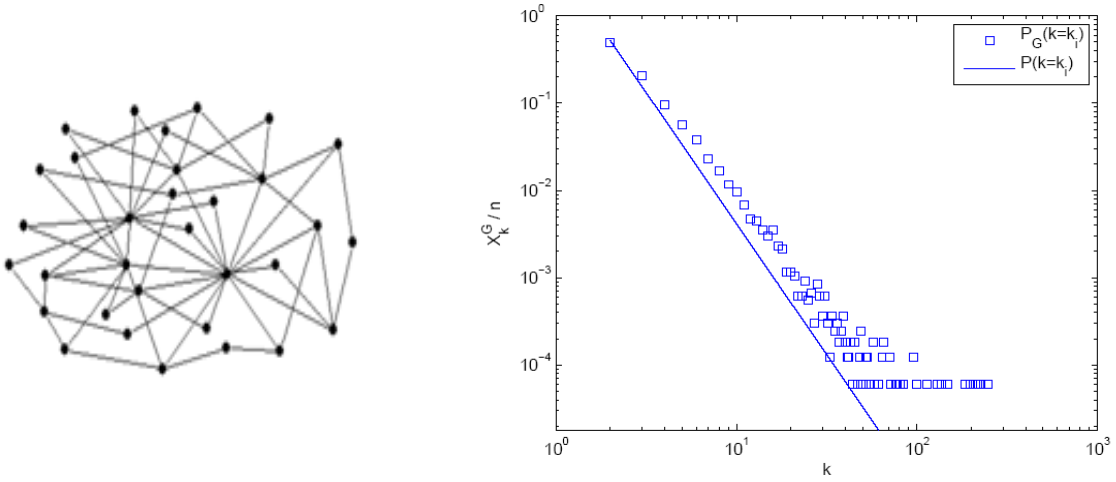
$$P(k) \sim k^{-\gamma} \tag{1.7}$$

Las redes s*cale-free*, presentan la característica invariante de que un número reducido de nodos que conforman la red tienen grado muy alto y una gran cantidad de nodos tienen un grado pequeño [41,42]. 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 [55,56].

En esta distribución el parámetro γ decrece 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 γ se decrementa. 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 [16,41,44].

En la Figura 1.2.a se aprecia la estructura topológica de una red scale-free. La Figura 1.2.b muestra la gráfica distribución del grado obtenida para una red scale-free con 3072 nodos, 9209 aristas, grado mínimo $\delta(G)$=3, grado máximo $\Delta(G)=139$, grado promedio $\langle k \rangle = 6$, los cuadros representan la distribución del grado real de la red y la línea continua representa la distribución del grado calculada con la Ecuación (1.7).



a) estructura topológica b) distribución del grado

Figura 1.2. Red Scale-free.

La Internet presenta una distribución del grado power-law en su topología, y es considerada una red scale-free en promedio [5,9,17,43]. Sin embargo para estudiar las características de los datos lo hace por medio de una familia de distribuciones de probabilidad que forman parte de la familia de la distribución power-law, ésta es conocida como la distribución Zipf. La ley de Zipfs es una ley empírica formulada usando estadística matemática y esta relacionada con el hecho de que muchos tipos de datos estudiados en la física y las ciencias sociales pueden ser aproximados con una distribución zipfiana [41,57]. En la actualidad para estudiar las propiedades estadísticas de grandes conjuntos de datos, tales como la Internet, la ley de Zipf predice que de una población de *N* elementos, la frecuencia de elementos de orden *k* esta dada por la Ecuación (1.8):

$$f(k;\gamma,N)=\frac{\frac{1}{k^{\gamma}}}{\sum_{n=1}^{N}\left(\frac{1}{n^{\gamma}}\right)}, \qquad (1.8)$$

donde *N* es el número total de elementos, *k* es el rango de los elementos y γ es el valor del exponente que caracteriza la distribución. Por ejemplo, en un texto sólo un pequeño número de

palabras son utilizadas con mucha frecuencia, mientras que frecuentemente ocurre que un gran número de palabras son poco empleadas [57].

1.2.2 Redes P2P Complejas

Después de haber modelado la topología de las redes complejas, el siguiente paso es establecer la forma de comunicación entre los nodos de la red compleja. 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. Una red P2P es un sistema distribuido donde todos los nodos son iguales en términos de funcionalidad y las tareas que desarrollan en el sistema [8,13,58,59,60].

El objetivo de una red P2P es compartir recursos tales como información, música, video, entre otros. La popularidad de los sistemas P2P es motivada por los beneficios ofrecidos a los usuarios finales, en contraste a la Web tradicional, un sistema P2P no necesita contar con ningún servidor centralizado dedicado, lo cual hace a las redes P2P confiables y tolerantes a fallas [58,59,60].

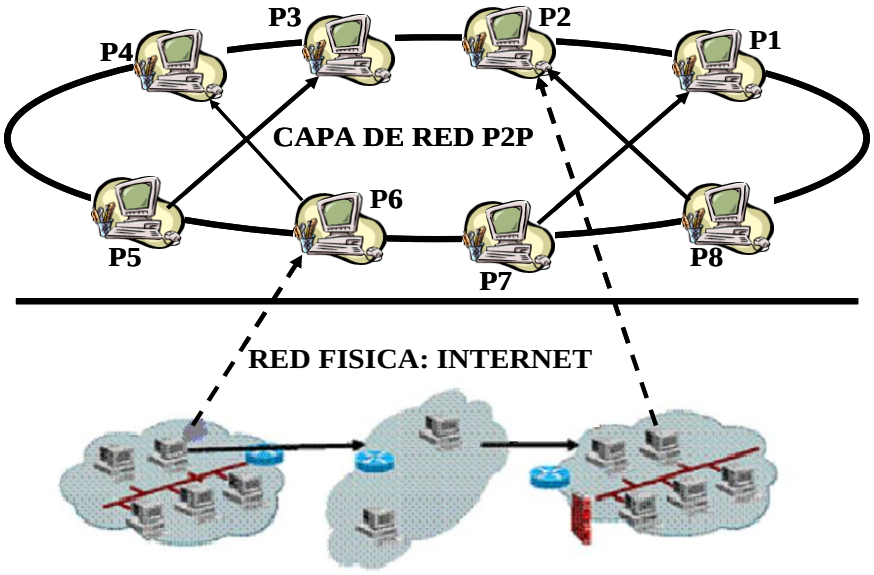


Figura 1.3: Distribución y comunicación entre los nodos de una red P2P.

Los usuarios pueden fácilmente unirse a una red y salirse cuando sea necesario, formando redes no-estructuradas auto-organizadas. Debido a la naturaleza no-estructurada, estas aplicaciones emplean principalmente mecanismos de búsqueda de datos basados en inundación, éstos mecanismos generan mucho tráfico extra en la Internet y limitan el crecimiento de los sistemas P2P. Como se observa en la Figura 1.3. Los sistemas P2P junto con la capa de comunicaciones de red (típicamente la Internet) forma un sistema complejo que requiere operación autónoma a través de mecanismos de búsqueda inteligentes [5,13].

En las redes P2P distribuidas complejas, la recolección de conocimiento global no es un enfoque factible para desarrollar consultas sobre recursos compartidos. En estas circunstancias el proceso de búsqueda cuando se inicia una consulta necesita determinar localmente su comportamiento sin recurrir a un mecanismo de control global.

1.2.3 Proceso de Búsqueda Semántica

El problema de la búsqueda de documentos en redes P2P consiste en determinar la ruta más corta de un nodo que emite una consulta a nodos que pueden contestarla apropiadamente, los cuales le proveerán la información requerida [5,8,11]. La consulta viaja moviéndose a través de la red desde un nodo de inicio y después a un vecino del nodo vecino y así hasta localizar el recurso requerido o considerarlo ausente en la red [8,14,15,38].

Los algoritmos de búsqueda en redes P2P deben considerar algunos factores que van desde el comportamiento de los usuarios, hasta el software y hardware, entre otros. Debido a esta complejidad las soluciones propuestas para la búsqueda se han limitado a casos especiales como pueden ser la congestión, el comportamiento de los usuarios, el mecanismo de búsqueda, entre otros, sin llegar a un consenso [1, 2, 4, 5, 23].

En la presente investigación se considera la búsqueda de documentos basada en claves que representan a dichos documentos. Éste es el tipo de búsqueda más popular en redes P2P y se denomina *búsqueda semántica* SQRP [5]. En la introducción de la tesis, SQRP se formula matemáticamente como un problema de optimización combinatoria y su complejidad se demuestra en el anexo B.

1.3 Algoritmos de Búsqueda Semántica en Redes P2P Complejas

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. Existen dos estrategias posibles para búsquedas en redes P2P: las denominadas *búsquedas ciegas*, que propagan la consulta a un número suficiente de nodos para satisfacer la consulta. Otra estrategia que se lleva a cabo es tomar la información acerca de la ubicación de documentos, llamados *búsquedas informadas*. La semántica de la información utilizada abarca desde el envío de pistas simples hasta direcciones de recurso exactas [38,61,62].

Existen diferentes clasificaciones de los métodos de búsqueda semántica en redes P2P. En la Figura 1.4 se muestra una parte de la taxonomía de los algoritmos de búsquedas la cual fue presentada en [8,38]. Independientemente del tipo de búsqueda, para evitar el congestionamiento de la red P2P y la degradación del sistema subyacente, el alcance de la transmisión se delimita por el parámetro **TTL** (**Time-To-Live**) que controla el tiempo de vida del proceso de búsqueda.

1.3.1 Búsqueda por Caminatas Aleatorias (Random-Walk)

El algoritmo Random-Walk (RW) es una técnica de búsqueda ciega clásica que ha sido muy usada [36,14,39]; donde los nodos de la red no poseen información sobre la localización o contenido de los recursos requeridos, a menos que el recurso se encuentre en el mismo nodo. Esta tipo de búsqueda se basa en TTL y en un recurso que se desea localizar. El recurso a buscar puede ser un archivo, programa, ó dato.

Sea *G* un grafo que modela la red y *v* un nodo en *G*. Una caminata aleatoria RW 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, RW 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 o se agote TTL [36].

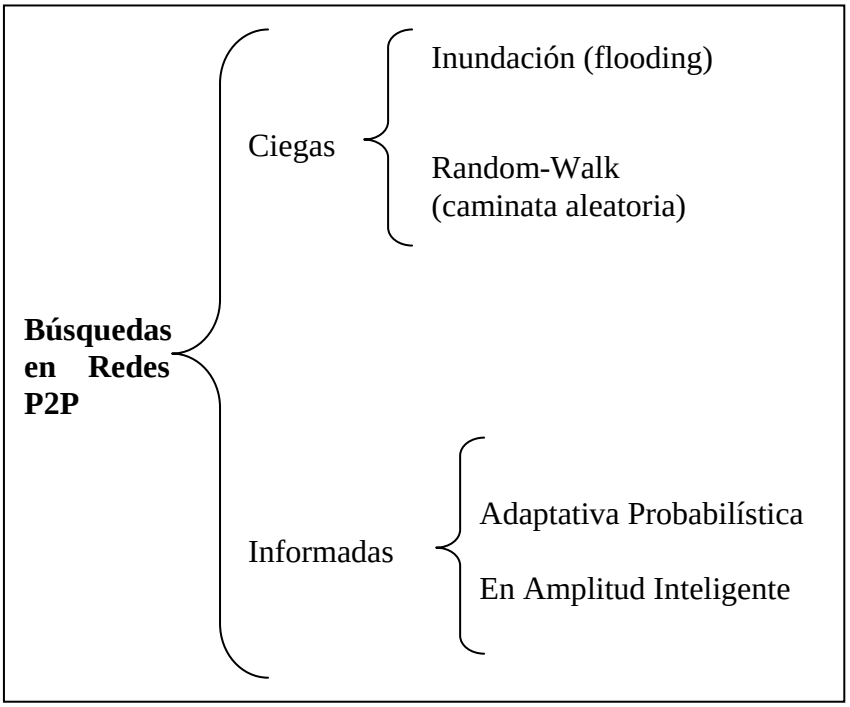


Figura 1.4. Clasificación de las búsquedas

1.3.2 Búsqueda por Inundación (Flooding)

El algoritmo Flooding (FL) es la técnica más popular en sistemas P2P. Al igual que RW, el algoritmo FL se basa en TTL y en un recurso que se desea localizar.

En FL, un nodo *i* inicia el proceso de búsqueda enviando la descripción del recurso y el valor de TTL hacia todos sus nodos vecinos. Cuando un nodo vecino *j* recibe lo enviado por el nodo *i*, éste realiza una búsqueda local y responde al nodo *i* indicando si tiene o no el recurso. Además, si

TTL > 0 el nodo *j* decrementa el valor de TTL en una unidad e inmediatamente propaga la descripción del recurso y TTL hacia todos sus nodos vecinos. Cuando TTL= 0 se deja de propagar la solicitud de búsqueda. Cuando, un nodo recibe la misma solicitud de búsqueda varias veces, no realiza ningún procesamiento para atender la solicitud ni la propagación. Este método de búsqueda tiene la desventaja de generar una gran cantidad de mensajes que hacen que los canales de comunicación se congestionen rápidamente [14,38,51].

1.3.3 Búsqueda en Amplitud Inteligente

Esta es una versión informada del algoritmo de búsqueda en amplitud modificada. Los nodos almacenan las consultas exitosas hacia los vecinos para responder peticiones y clasificarlas. Las consultas son realizadas mediante texto libre, si no existe información de la consulta el método utilizado es la búsqueda en amplitud, el cual va inundando por niveles hasta encontrar la información o TTL=0.

Si existe información de consultas anteriores, el nodo iniciador identifica todas las consultas similares a la actual, de acuerdo a una métrica de similitud de la consulta; entonces se elige enviarla a un conjunto de vecinos que han regresado la mayoría de los resultados para estas consultas. Si encuentra el objetivo, la consulta envía 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 [14,51].

1.3.4 Búsqueda Adaptativa Probabilística: Colonias de Hormigas

Este tipo de búsqueda usa la retroalimentación de búsquedas anteriores para realizar búsquedas futuras eficientes dentro de un periodo de tiempo delimitado por TTL. Cada par 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 se realiza en el camino de retorno al par de búsqueda inicial y puede tomar lugar después del éxito o fracaso, ajustandose de acuerdo a la probabilidad [13,38].

Los algoritmos de colonias de hormigas están inspirados en el comportamiento colectivo de la búsqueda de comida, en donde las hormigas se comunican de manera indirecta con otros miembros de su colonia. 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” con el objetivo de encontrar la ruta más corta entre el nido y la fuente de comida [24,63].

Las colonias de hormigas artificiales son representadas como sistemas multiagente. Es decir cada hormiga-agente tiene información o capacidades parciales para resolver una parte del problema. La información se va registrando en la tabla de feromonas, la cual es administrada de manera local, los datos son descentralizados, y la computación es asíncrona [64]. 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 simuladas como sistemas adaptativos complejos [24].

Ejemplos de sistemas complejos son la colonia de hormigas, organismos, ecologías y sociedades. En esta tesis, los algoritmos de colonia de hormigas fueron elegidos como método de solución debido a que pueden ser modelados como un sistema multiagente y comparten muchas de las características importantes como la emergencia y adaptación que son necesarias para que el algoritmo de búsqueda, pueda aprender y adaptarse al ambiente de búsqueda [6,65].

a) Sistema Multiagente

El estudio de la inteligencia que surge de las interacciones entre muchos agentes, es conocido como *sistemas multiagente*. Dichos sistemas son complejos por naturaleza y se caracterizan por un gran número de partes que tienen muchas interacciones. Esta complejidad no es accidental, es una propiedad innata de los tipos de tareas para las cuales estos sistemas son utilizados [66,67,68].

Los agentes son los elementos principales que conforman un sistema multiagente. Entonces 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 [67,68].

Para el estudio de los sistemas multiagente, la estructura de la red donde navegarán los agentes juega un rol importante. Y en la mayoría de los problemas existirán mas de un tipo de agente dando lugar a que el sistema se vuelva complejo. La idea de combinar los sistemas multiagente y las redes complejas es un elemento esencial para dar solución a problemas, ya que la teoría de redes complejas permite modelar los elementos dinámicos y las interacciones del sistema multiagente 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 [5,6,18,].

Los desarrollos de aplicaciones sobre redes de gran escala de telecomunicaciones y sistemas distribuidos han incrementado su interés con el uso de sistemas multiagente [67,68]. La propia definición de agentes de software, indica que los agentes tienen características parecidas a la de los sistemas P2P, como son su funcionamiento de forma distribuida y autónoma. El trabajo con sistemas multiagente se facilita cuando dichos agentes trabajan con servicios P2P estándares [69].

Los algoritmos metaheurísticos basados en colonias de hormigas, pueden ser modelados como un sistema multiagente, cada hormiga agente contribuye a lograr un objetivo global, los datos son descentralizados y la computación es asíncrona. Las colonias de hormigas exhiben un comportamiento emergente debido a que construyen sus resultados de manera incremental por lo que son consideradas como sistemas adaptativos complejos [5,17,23,24].

b) Algoritmos de Colonia de Hormigas Básicos

Ant Colony Optimization (ACO), es una metodología basada en el comportamiento de las hormigas, y comprende diferentes tipos de sistemas de hormigas. Los primeros algoritmos de hormigas fueron diseñados para resolver problemas de optimización basados en grafos y con una tabla de feromonas global. En la actualidad se ha diversificado la metodología original con el objetivo de atacar problemas con estructuras distribuidas tales como el enrutamiento de paquetes en redes que usan tablas de feromonas distribuidas. En los siguientes párrafos se describen los algoritmos ACS y SemAnt que son variantes de ACO [63,70,71].

c) Ant Colony System (ACS)

Algoritmo metaheurístico en el cual, un conjunto de agentes llamados hormigas, cooperan entre sí con el objetivo de encontrar buenas soluciones a problemas de direccionamiento. 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 [24,63].

La Fase 1 se muestra en la Tabla 1.1, aquí se inicializan: la tabla de feromonas τ con el valor de τ_0 (Líneas 01 y 02), la lista J_k de nodos no visitados por la hormiga k y la lista de nodos iniciales donde cada una de las hormigas esta posicionada r_k, ésta asignación puede ser de manera aleatoria (Líneas 03–06).

Tabla 1.1. Fase de inicialización del algoritmo
Ant Colony System (ACS)

```
01  Para cada par (r,s)
02      τ(r, s) = τ0
03  Para cada hormiga k
04          Seleccionar r_k1 como el nodo inicial para la hormiga k
05          J_k(r_k1) = {1…n}-r_k1
06          r_k = r_k1
```

Para la Fase 2, mostrada en la Tabla 1.2, corresponde con la parte de construcción de la solución en donde cada hormiga va construyendo paso a paso la solución, mientras va seleccionando el siguiente nodo al cual ir, se elimina de la lista de nodos no visitados y se añade al recorrido creado por la hormiga k (Líneas 02-06).

Tabla 1.2. 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  //
05              J_k(s_k) = J_k(r_k) – s_k          // Quitar la ciudad visitada
06              recorrido_k(i) = (r_k, s_k)        // Añadir al recorrido
07      De lo contrario
08          Para cada hormiga
09              s_k = r_k1                         // La siguiente ciudad es la inicial
10              recorrido_k(i) = (r_k, s_k)        // Añadir s_k al recorrido
11      Para cada hormiga
12          τ (r_k, s_k) = (1-ρ)τ (r_k, s_k) + ρτ0 // Actualizar localmente el rastro
13          r_k = s_k                              // s_k es ahora la ciudad actual
```

Cuando se ha llegado al último nodo seleccionado este se conecta con el nodo inicial para cerrar el circuito (líneas 07-10). Durante cada paso se actualiza de manera local la tabla de feromonas (Líneas 11-13) seleccionando la arista donde la hormiga eligió pasar.

La Fase 3, mostrada en la Tabla 1.3, 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), con este recorrido se actualiza la tabla de feromonas.

Tabla 1.3. 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}

En la Tabla 1.4, 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). En el caso de no cumplirse la condición de terminación, se continúa realizando la Fase 2 (Líneas 03 y 04). El algoritmo ACS fue originalmente propuesto para resolver problemas del tipo Travel Salesman Problem (TSP, Problema del Agente Viajero), en los cuales el objetivo es encontrar la mejor ruta, minimizando los costos de viaje [24].

Tabla 1.4. 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

d) SemAnt

El algoritmo metaheurístico *SemAnt* tiene por objetivo resolver búsquedas basadas en contenido en redes P2P. Esté algoritmo esta basado en el algoritmo clásico de ACS propuesto en [24] y fue desarrollado por la investigadora Michlmayr en su tesis de doctorado [5].

La *topología* en la cual navegan los agentes es estática y su distribución de probabilidad es aleatoria, es decir que el grado de cada uno de los nodos es cercano al grado promedio. Cada nodo tiene una dirección única que la usa como un identificador. Los enlaces entre los nodos son bidireccionales. La red esta formada por 1024 nodos. Su *repositorio* es estático y solo pude manejar 30 palabras, es decir su tabla de ruteo es una matriz n x C, donde n son la cantidad de vecinos y C = 30 que es número de palabras por nodo. La distribución del contenido de los repositorios es Scale-free [56], es decir hay un pequeño número de palabras que se repiten y las demás se repiten pocas veces en toda la red. Las consultas siguen una distribución aleatoria, donde todos los nodos tienen una probabilidad de lanzar una consulta a intervalos regulares de tiempo.

El algoritmo de búsqueda tiene el objetivo de encontrar la ruta mas corta entre cada nodo que lanza una consulta y los nodos que la contestan. Para lograr su objetivo en una parte de la investigación se lleva acabo un ajuste de parámetros global de manera manual, y no toma en cuenta la interacción entre los parámetros. La Tabla 1.5 muestra el algoritmo SemAnt.

Los pasos necesarios para contestar la consulta q lanzada en un punto p_q son los siguientes:

Tabla 1.5 Algoritmo SemAnt

Paso 1:	Checar el repositorio del nodo $P_{q.}$ Si el resultado es encontrado, enviárselo al usuario. Si el número de resultados es menor que la cantidad máxima de recursos deseados *rmax*, ir al paso 2. De lo contario terminar el algoritmo.
Paso 2:	Crear una hormiga de avance F_q e inicializa el tiempo de la consulta tInicio en el nodo $P_{q.}$ Agregar el identificador del nodo P_q a la pila de nodos visitados (F_q). Inicializar la lista del costo de los enlaces $lc(F_q)$ para resolver la consulta $P_{q.}$
Paso 3:	Usar en procedimiento de selección de enlaces para seleccionar el vecino P_j la hormiga de avance F_q deberá seleccionar de los vecinos que no hayan sido visitados (x $\in$ [1..n], n $\in$ □ ; lista de vecinos no visitados). Enviar la hormiga F_q al nodo P_j.
Paso 4:	Para cada hormiga de avance F_q que llega al nodo P_j. Checar si el nodo P_j ya había sido visitado. Si ya había sido visitado terminar. De lo contrario checar el repositorio del nodo P_j y verificar si cuenta con la información solicitada por la consulta. Si no se encontró información ir al paso 6. De lo contrario agregar los identificadores de todos los recursos encontrados en el repositorio del nodo P_j a R, que lleva el control de los recursos encontrados.
Paso 5:	Generar una hormiga de retorno B_q. Pasarle a *R*, el identificador del nodo P_j que almacena *R*, la pila de nodos ya visitados $s(F_q)$, y el registro del costo de los enlaces $lc(F_q)$. Enviar B_q de regreso al punto donde se lanzo la consulta P_q. Terminar la hormiga de avance $F_{q.}$ De lo contrario continuar en el paso 6.
Paso 6:	Agregar el identificador del nodo P_j a la pila de nodos ya visitados $s(F_q)$. Agregar el costo de los últimos enlaces usados a la lista de enlaces $lc(F_q)$.
Paso 7:	Sea TTL el máximo tiempo de procesamiento para la consulta. Si tInicio+TTL < TiempoActual, continuar en el paso 3. De lo contrario, terminar $F_{q.}$

1.4. Estado del Arte de las Búsquedas Semánticas en Redes P2P Complejas

El éxito de los algoritmos de direccionamiento de consultas semánticas en redes de compartición de archivos P2P, descansa sobre los mecanismos de búsqueda, los cuales están recibiendo una

atención especial [5,14,15,17]. Aquí se resumen algunas de las más relevantes propuestas para SQRP. Los investigadores que han tratado de dar solución a SQRP lo han abordado desde diferentes enfoques de solución en los cuales no se resuelve el problema completo sino solo una parte de él.

Dada la complejidad de SQRP, se tomaron algunas ideas de los trabajos investigados en la formulación del algoritmo propuesto y así se planteó a una propuesta innovadora que incluye aspectos no solucionados hasta el momento.

Wu [15] 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 de Wu, primero genera una exploración a una pequeña área 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. Cabe señalar que este autor no usa las reglas clásicas de selección y actualización propuestas por Dorigo [24]. La característica de su trabajo es el uso de una técnica denominada DQ+ que simula a una inundación controlada, es decir no inunda a todos los vecinos solo aquellos que cumplen con una condición.

Michlmayr [5] propone un algoritmo distribuido para el enrutamiento de consultas semánticas en redes P2P llamado *SemAnt* ver sección 2.3.4. 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 Michlmayr esta basado en las reglas de selección de ACS [24] y en la técnica de ruteo de AntNet [71], 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).

Kalogeraki [14] propone un algoritmo distribuido para el enrutamiento de consultas semánticas en redes P2P llamado *Mecanismo de Búsqueda Inteligente* basado en *Búsquedas en Amplitud Modificadas* (*ISM-BFS*, por sus siglas en inglés). En este trabajo se propone una modificación al método clásico BFS, a través de una inundación controlada a través de un parámetro el cual representa el valor del número de vecinos a seleccionar de manera probabilista. El objetivo es ayudar al nodo que esta consultando a encontrar las respuestas más relevantes a la consulta de

manera rápida y eficiente. De esta manera Kalogeraki espera reducir costos en las comunicaciones a través de los mecanismos de recuperación de la información.

1.5 Comentarios Finales

El algoritmo AntSearch [15] y el algoritmo ISM-BFS [14] realizan inundaciones controladas por algún parámetro de popularidad, a través del uso de una estrategia de estructuras de datos para llevar los registros de consultas pasadas. Sin embargo, el uso de la estrategia de inundación durante la evolución del algoritmo sobrecargará la red con mensajes de búsqueda.

Por otro lado, el algoritmo SemAnt [5] esta basado en el algoritmo de Sistema de Colonia de Hormigas, y propone un conjunto de parámetros iniciales para su buen funcionamiento, uno de los parámetros calculados es el tiempo de vida (TTL) de los agentes de búsqueda para satisfacer un conjunto de consultas. Los valores de los parámetros fueron calculados por fuerza bruta y una vez fijados no cambian su valor durante todo el experimento. Otra característica es el uso de tablas de aprendizaje para registrar experiencias pasadas y tratar de optimizar los recursos de almacenamiento y así evitar inundar las líneas de comunicación con mensajes de búsqueda.

En la Tabla 1.6 se resumen las características principales de los trabajos del estado del arte. En la primera columna se presenta información del investigador, seguido del nombre del algoritmo de búsqueda desarrollado. En la tercera columna se establece la técnica de búsqueda usada y en la cuarta columna se identifican los trabajos que realizan caracterización local de la red compleja. Después, en la quinta columna se continúa con las funciones históricas de aprendizaje del rendimiento parcial. Terminando con el tipo de cálculo utilizado para el parámetro TTL.

Tabla 1.6. Estado del Arte: Consultas Semánticas y Algoritmos de Hormigas

INVESTIGADOR	ALGORITMO DE BÚSQUEDA USADO	TECNICA DE BÚSQUEDA	FUNCIONES DE CARACTERIZACION DEL AMBIENTE LOCAL	APRENDIZAJE DEL RENDIMIENTO	TIEMPO DE VIDA (TTL)
[Wu 2007]	AntSearch	Flooding	*	Muestreo	Dinámico Global
[Michlmayr 2007]	SemAnt	Adaptativa Probabilística: Colonia de Hormigas	*	Largo alcance	Estático Global
[GÓMEZ 2009] Esta tesis	NAS (Neighboring-Ant Search)	Adaptativa Probabilística: Colonia de Hormigas	DDC y Lookahead	Largo alcance	Estático Global
[GÓMEZ 2009] Esta tesis	AdaNAS (Adaptative Neighboring-Ant Search)	Adaptativa Probabilística: Colonia de Hormigas	Grado, Lookahead	Largo y corto alcance	Dinámico Local Adaptativo

Los aspectos más relevantes de los trabajos del estado del arte han sido incorporados en el algoritmo NAS propuesto en esta tesis [17]. Por ejemplo: NAS y SemAnt están centrados en las mismas condiciones del problema, y ambos basados en el algoritmo ACS. Sin embargo, dentro de las diferencias más significativas entre SemAnt y NAS, se encuentra que SemAnt solamente aprende de experiencias pasadas, y el algoritmo NAS además toma ventaja del medio ambiente local. Esto significa que la búsqueda en NAS toma lugar en términos del método de exploración clásicos Lookahead [39], la métrica DDC, y un aprendizaje local de largo alcance. El aprendizaje es de largo alcance porque registra el desempeño final obtenido en consultas previas a través de la tabla de feromonas [72].

Inicialmente, los parámetros del algoritmo NAS se configuraron usando estrategias de ajuste global de parámetros [19]. En la siguiente etapa el algoritmo NAS es estudiado detalladamente para identificar las características que formarán la función de ajuste de control adaptativo del parámetro TTL. Al incorporar la función adaptativa al algoritmo este cambia su nombre a AdaNAS.

El algoritmo AdaNAS hereda todas las estrategias incorporadas en NAS, pero esta versión usa estrategias de ajuste de control adaptativo para el parámetro TTL las cuales le permiten extender la búsqueda de manera controlada, para tratar de identificar más documentos y así obtener un mayor rendimiento. La función incluye: caracterización local de la estructura del problema SQRP (métricas topológicas), aprendizaje de corto alcance del desempeño parcial obtenido en consultas previas y el aprendizaje de largo alcance del desempeño final obtenido en consultas previas (tabla de feromonas).

Los detalles del desarrollo de los algoritmos de búsqueda semántica NAS y AdaNAS se presentan en el capítulo 4. En ese mismo capítulo se presenta la aportación más importante de la tesis que es el desarrollo de una función de adaptación del parámetro TTL que logre un alto rendimiento de la búsqueda en redes P2P.

Capítulo 2

AJUSTE DE PARÁMETROS DE ALGORITMOS DE OPTIMIZACIÓN

En esta sección se describen los temas relacionados con el ajuste de parámetros y los trabajos que han sido desarrollados aplicando alguna de las estrategias de ajuste de parámetros. Las estrategias de ajuste se dividen en dos etapas dependiendo en que parte del experimento se apliquen, antes o durante la ejecución del experimento. Además se describe la estrategia de ajuste del control adaptativo de parámetros, la cual es usada en el algoritmo de optimización de la colonia de hormigas.

2.1 Clasificación del Ajuste de Parámetros

Las dos etapas principales en la aplicación de un metaheurístico a un problema específico son el esquema de representación y la determinación de la función objetivo. Estos dos elementos forman el puente entre el contexto original del problema y el algoritmo de solución. Al crear cada uno de estos elementos se necesitan seleccionar valores para cada uno de sus parámetros disponibles, de tal manera que se logre una mejor representación del problema y una mayor eficiencia en el proceso de resolución del mismo. A cada una de las combinaciones de valores para los parámetros se le llama *configuración paramétrica*, y al problema de seleccionar la mejor configuración se le llama *problema de configuración* o *ajuste de parámetros* [2,73].

Las investigaciones realizadas por Angeline [74], Hinterding [75], y Smith [76] les llevó a proponer una clasificación de tipos de ajustes de parámetros. En particular, el enfoque que plantea Angeline [74], esta basado en niveles de adaptación y reglas de actualización. Para la adaptación propone tres niveles: de la población, individual y de componentes. Mientras que para la actualización propone dos tipos de mecanismos de actualización: reglas absolutas y empíricas.

Posteriormente Eiben, Hinterding y Michalewicz [20] extienden la taxonomía de Angeline; ellos distinguen dos tipos de adaptación: Estática (afinación de parámetros) y Dinámica (control de parámetros). La afinación de parámetros, se realiza antes de la ejecución del algoritmo; y el control de parámetros se realiza al mismo tiempo que se ejecuta el algoritmo. La clasificación general del ajuste de parámetros se observa en la Figura 2.1.

La *afinación de parámetros* se caracteriza por ofrecer una configuración inicial estática durante toda la ejecución del algoritmo, es decir, evalúa el desempeño general del algoritmo, y no garantiza que los valores propuestos sean los mejores a lo largo de toda la ejecución. Generalmente sus valores representan promedios, medianas o alguna otra medida que represente a un conjunto de valores para los cuáles el algoritmo trabajó eficientemente [20,77].

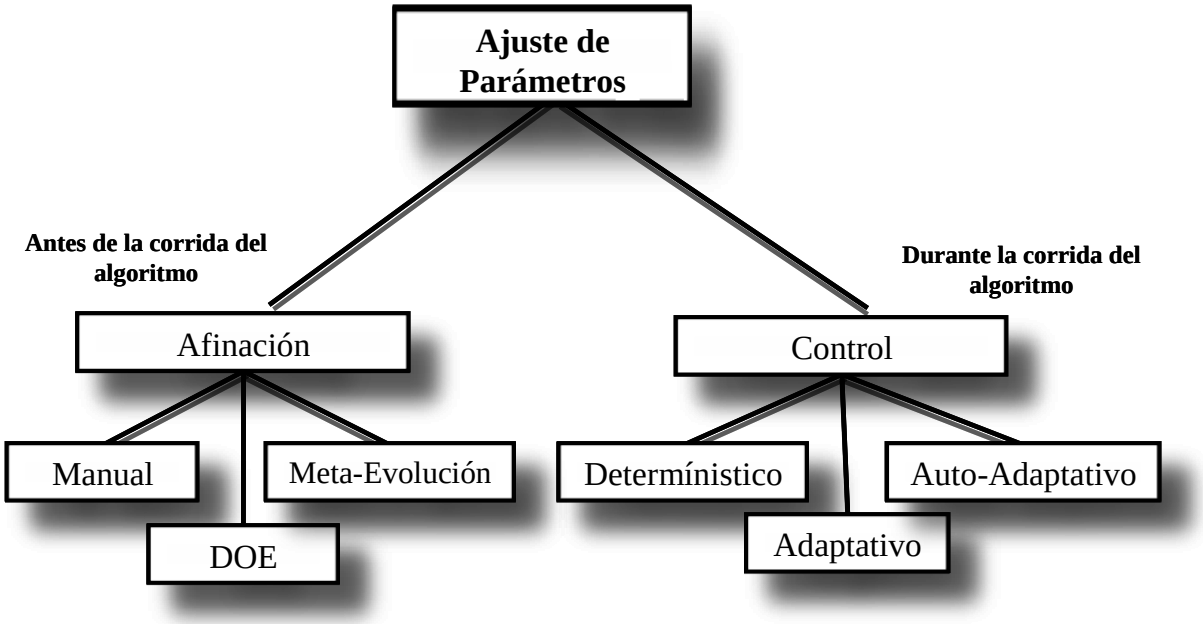


Figura 2.1. Clasificación general del ajuste de parámetros

La estrategia más usual para obtener una configuración inicial de parámetros consiste en probar muchas configuraciones de manera manual y escoger la de mejor resultado. Los resultados obtenidos a través de este método son muy específicos de las instancias analizadas, además de consumir mucho tiempo del investigador y recursos computacionales. Otra alternativa que esta tomando auge es el D*iseño de Experimentos* (DOE, por sus siglas en inglés) ya que proporciona configuraciones sustentadas en el uso de herramientas estadística [18,20].

La estrategia menos aplicada es la meta-evolución, la cual busca configurar los parámetros de un algoritmo metaheurístico para una instancia específica invocando a otro metaheurístico. Su principal desventaja es que el segundo metaheurístico también requiere de una buena configuración [20,77].

El *Control de Parámetros,* ofrece la ventaja de estar monitoreando los cambios en el ambiente al mismo tiempo que considera el estado actual del algoritmo. Eiben [73] propone obtener una configuración paramétrica inicial a partir de técnicas de afinación y posteriormente ejercer control sobre los parámetros, esto brinda mejores resultados. El control de parámetros *determinístico* toma lugar cuando a algún parámetro se le asignan valores a partir de alguna regla determinista que modifica su valor sin considerar ninguna retroalimentación del algoritmo en ejecución [77].

El control *adaptativo* de parámetros se realiza cuando existe alguna forma de retroalimentación del pasado que determina un cambio en sentido y magnitud del parámetro. Si la retroalimentación es obtenida por el mismo mecanismo de ajuste entonces el control de parámetros es *autoadaptativo* [77,78]. Se ejerce control de parámetros *autoadaptativo* cuando los parámetros son codificados dentro de la estructura del algoritmo metaheurístico, de esta forma las mejores soluciones se encuentran asociadas a una configuración paramétrica la cual también irá evolucionando así como la solución; a esto se llama *evolución de la evolución* [20,75].

2.2 Técnicas de Ajuste de Parámetros Global

Las técnicas de ajuste de parámetros globales en la actualidad son mas usadas por los investigadores del área de diseño de algoritmos metaheurísticos. De las estrategias de ajuste global la que esta tomando mayor relevancia son las de diseño de experimentos, debido a que proporciona información estadística para la toma de decisiones y la validez de los experimentos.

2.2.1 Análisis y Diseño de Experimentos (DOE)

Una de las estrategias más usadas en la afinación de parámetros es el *diseño de experimentos*, (DOE) ya que ofrece una manera de determinar el ajuste óptimo global de parámetros a través de diversas herramientas estadísticas [79,80]. DOE es una colección de herramientas estadísticas que se relacionan con la planeación, la ejecución y la interpretación de un experimento para obtener conclusiones validas y objetivas. Un experimento puede definirse como una prueba planeada donde se introducen cambios controlados en las variables de un proceso con el fin de analizar cambios que pudieran observarse en las salidas del sistema. Este proceso puede representarse con el modelo mostrado en la Figura 2.2 [80]. Se puede observar que el proceso recibe datos de entrada (instancia de un problema) para producir una salida (solución del problema). El proceso (metaheurístico) para alcanzar la solución puede ser regulado por factores controlables (parámetros configurables) o no controlables (parámetros no configurables).

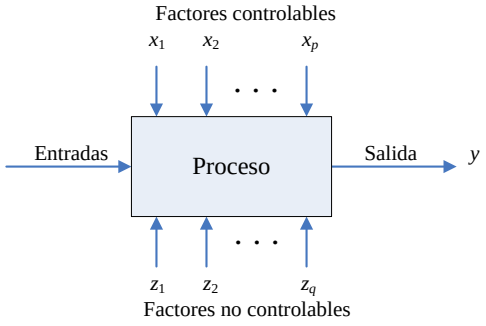


Figura 2.2 Modelo general de un proceso

Para aplicar el enfoque estadístico en un diseño y análisis de un experimento, es necesario contar con un esquema general de procedimientos a seguir. Montgomery [80] presenta una explicación de este esquema:

a) Identificar y enunciar el problema: un enunciado claro del problema contribuye a una mejor comprensión de los aspectos bajo estudio y de la solución final.
b) Elección de factores, niveles y rangos: en este punto se deben considerar los factores que pueden influir en el proceso, y determinar cuáles pueden ser controlados por el experimentador y cuáles no. Determinar los niveles de los factores, es decir, valores específicos del factor, si son cualitativos o cuantitativos y el rango de valores que pueden tomar.
c) Selección de variables respuesta: identificar la variable o las variables que proporciona información útil acerca del proceso de estudio.
d) Elección del diseño experimental: implica la consideración del tamaño de la muestra, el número de factores y niveles, y el objetivo experimental.
e) Realización del experimento: que fue diseñado con la información antes descrita.
f) Análisis estadísticos de los datos: en esta etapa deberán usarse métodos estadísticos para analizar los datos a fin de que las conclusiones sean objetivas.
g) Conclusiones y recomendaciones: es aquí donde se obtienen conclusiones prácticas acerca de los resultados y se recomiendan las acciones a seguir.

Diseño Factorial

Es una de las estrategias de diseño de experimentos mas usadas, ya que cuenta con un conjunto de herramientas que soportan los resultados estadísticos de los experimentos de forma gráfica [80]. En muchos experimentos se tiene principal interés en estudiar los efectos o la influencia de dos o más factores sobre una variable de respuesta; para este tipo de experimentación, los diseños factoriales son los más eficientes. Casos especiales del diseño factorial general son usados ampliamente en trabajos de investigación debido a que constituyen las bases de otros diseños de gran valor práctico [81,82].

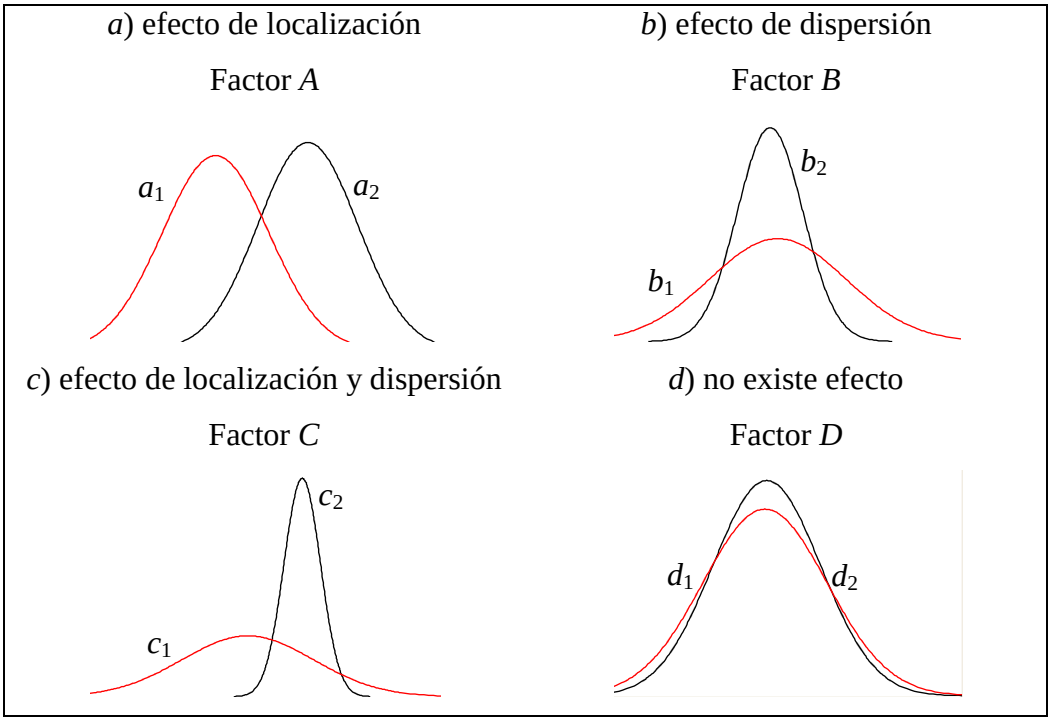


Figura 2.3 Objetivos de un diseño experimental

Por *diseño factorial* se entiende que en cada ensayo o réplica completa del experimento se investigan todas las combinaciones posibles de los valores de los factores llamados *niveles*. El *efecto* de un factor se define como el cambio en la variable de respuesta producido por un cambio en el nivel del factor, con frecuencia se le llama *efecto principal* porque se refiere a los factores de interés primario en el experimento. En algunos experimentos puede encontrarse que la diferencia en la respuesta entre los niveles de un factor no es la misma para todos los niveles de los otros factores, cuando esto ocurre existe una *interacción* entre los factores [79,80].

Hay situaciones en que las conclusiones se aplicarán únicamente a los niveles del factor considerado en el análisis, es decir, las conclusiones no pueden extenderse a niveles del factor que no se consideraron, entonces se dice que el factor es *fijo* y por lo regular se aplica cuando el factor tiene un número muy reducido de niveles. Existen situaciones en las que el factor tiene un gran número de posibles niveles y es deseable extender las conclusiones a la totalidad de niveles se hayan o no considerado en el análisis, entonces se dice que el factor es *aleatorio* [80].

En general un diseño experimental, en este caso de un diseño factorial, tiene un conjunto de objetivos [82,83] los cuales se ilustran en la Figura 2.3 y se enlistan a continuación:

a) Identificar factores que cambian la media de la variable respuesta pero no la variabilidad (*efecto de localización*, ver Figura 2.3a).
b) Identificar qué factores contribuyen a la variabilidad de la variable respuesta pero no afectan la media (*efecto de dispersión*, ver Figura 2.3b).
c) Identificar qué factores cambian la media y la variabilidad de la variable respuesta (ver Figura 2.3c).
d) Identificar qué factores no tienen efecto alguno sobre la variable respuesta (ver Figura 2.3d).

Los diseños factoriales ofrecen varias ventajas, son más eficientes que los experimentos de un factor a la vez, ya que realizan las interacciones entre los factores y evitan llegar a conclusiones incorrectas; permiten la estimación de los efectos de un factor con varios niveles de los factores restantes, produciendo conclusiones que son válidas para un rango de condiciones experimentales.

2.2.2 Modelado Causal

El modelado causal es un método multivariado que puede ser usado en la afinación global de parámetros de algoritmos, ya que hace una representación generalizada del conocimiento, a través de encontrar dependencias en los datos, que impliquen relaciones de causa y efecto [84]. En esta aplicación, las *causas* son los valores de los parámetros, mientras que los *efectos* corresponden al desempeño del algoritmo obtenido con esos valores.

La causalidad es el término que de manera general relaciona las acciones con sus consecuencias, o las causas y con sus efectos. Para poder identificar la causalidad es necesario primero poder asociar eventos. Por ejemplo "buscar un documento con información popular necesita pocos pasos", relaciona documento popular con la disminución de la cantidad de pasos. Después de asociar los eventos de causa-efecto, lo siguiente es realizar un experimento donde se busque un

grupo de documentos con información popular e ir registrando la cantidad de pasos necesarios para satisfacer las consultas. De acuerdo a los resultados se podría llegar a la conclusión de qué tan cierta es nuestra aseveración; a esta actividad o experimento se le conoce como intervención [86].

En el ejemplo, la causalidad puede interpretarse como el proceso de identificar la relación directa que existe entre dos eventos. Si la relación causal no se modifica al presentarse nuevas intervenciones o cambios se puede establecer predicción de acciones. La causalidad además de ayudar a establecer relaciones entre fenómenos sociales, físicos, psicológicos, y de otras índoles, ayuda a identificar los elementos o variables que afectan la calidad de los productos o servicios que se ofrecen en la industria por ejemplo, al identificar las causas que afectan la calidad de los productos o servicios se pueden poner en practica medidas de prevención o corrección que ayuden a minimizar o eliminar por completo los efectos de estas variables [84,85].

a) Modelos Causales

Un modelo causal es una representación generalizada del conocimiento, el cual es obtenido al encontrar dependencias en los datos, que impliquen relaciones de causa y efecto. La causalidad requiere identificar la relación directa existente entre eventos o variables. Las variables que intervienen en el modelo son de naturaleza aleatoria, y algunas pueden tener relación causal con otras [84]. Las relaciones causales son transitivas, irreflexivas y antisimétricas. Esto quiere decir que: 1) si A es causa de B y B es causa de C, entonces A es además causa de C, 2) un evento A no puede causarse a si mismo, y 3) si A es causa de B entonces B no es causa de A.

En la práctica las variables se dividen en dos conjuntos, las variables exógenas, cuyos valores son determinados por factores fuera del modelo y las endógenas, que tienen valores descritos por un modelo de ecuaciones estructurales. La representación de un sistema causal se puede hacer a través de un grafo acíclico dirigido (DAG), que muestra las influencias causales entre las variables del sistema y ayuda a estimar los efectos totales y parciales que resultan de la manipulación de una variable [84].

b) Algoritmos para la Creación de Grafos Causales

La mayoría de los métodos utilizados para crear modelos causales están basados en pruebas de independencia condicional, ésta define un conjunto de restricciones que deben ser satisfechas para inducir la estructura causal. Existe independencia estadística cuando la probabilidad de un evento no depende o se ve afectada por la presencia de otro. Dos variables X y Y son condicionalmente independientes dada una tercera variable W si para cualquier conjunto medible S de posibles valores de W, X y Y son condicionalmente dependientes dado el evento $W \in S$. Si se conoce el estado de W, el conocimiento del estado de Y es irrelevante para saber algo de X.

El algoritmo PC tiene como objetivo encontrar el grafo causal para muestras lo suficientemente grandes. El grafo obtenido representa las mismas condiciones de independencia condicional de la población bajo las siguientes suposiciones: el grafo causal en la población es acíclico y confiable, y el conjunto de variables causales es causalmente suficiente [86,87].

Este algoritmo consiste de dos etapas. La primera etapa es la *eliminación de aristas*. Esta etapa inicia con la construcción de un grafo completo que relaciona a todas las variables de entrada. Posteriormente en base a pruebas de dependencia condicional se van eliminando aquellas aristas entre variables que son condicionalmente independientes. La segunda etapa es la *Orientación estadística de aristas*. En esta segunda etapa se crea la orientación del grafo mediante operadores condicionales.

Existen herramientas computacionales disponibles para la creación de modelos causales, como TETRAD [Carnegie06]. Además de recursos desarrollados para la enseñanza de modelos causales como el laboratorio Causality Lab de la Universidad de Carnegie-Mellon [89].

c) Creación de modelos causales con TETRAD

TETRAD [89] es un software gratuito desarrollado con el objetivo de crear, simular datos, estimar, probar, predecir y buscar modelos estadísticamente causales. Ofrece métodos para el descubrimiento causal en una interfaz amigable y sencilla de usar. Si los datos que se van utilizar provienen de un tamaño de muestra lo suficientemente grande y las suposiciones de normalidad están razonablemente satisfechas, TETRAD puede ayudar con los siguientes problemas entre otros:

- Reconocer cuándo un conjunto de datos proporciona poca o ninguna información sobre los procesos subyacentes.
- Encontrar alternativas que expliquen adecuadamente los datos del modelo dado.
- Seleccionar las variables que influyen directamente en una variable de resultado.
- Reducir el número de variables requeridas para la predicción.
- Encontrar la ecuación estructural planteada con las variables latentes.
- Usar una red bayesiana para clasificar y predecir.

Si el propósito del análisis de los datos es estimar las influencias causales individuales que cada variable ejerce en el resultado, entonces los procedimientos que aplica TETRAD son teóricamente inestables. Dependiendo de la verdadera estructura causal y qué variables han sido modeladas, es posible identificar qué variables son las causas directas y cuales son los efectos directos. Así, cuando las suposiciones referentes a la distribución de los datos están justificadas, TETRAD puede usarse para ayudar a la selección de relaciones causales para después estimar la influencia precisa con cualquier paquete de datos. A continuación se describen las posibles orientaciones que pueden obtenerse en el grafo causal [89].

$\rightarrow$ Flecha sencilla. Indica una causa genuina directa.
$\leftrightarrow$ Flecha doble. Indica la presencia de una causa latente común entre dos variables.
o$\rightarrow$ Flecha sencilla con círculo en un extremo. Indica la incapacidad de TETRAD para deducir si existe una influencia directa entre dos variables ó si existe una causa latente entre ambas.
o–o Arista con círculos en ambos extremos. Indica la incapacidad de TETRAD para deducir si hay una influencia directa entre dos variables, y si fuera así, cual sería la dirección, o la causa latente entre ellas.

2.3 Ajuste del Control Adaptativo de Parámetros: Aprendizaje Reforzado

Como se describió en la sección 2.1, las técnicas de control de parámetros son: deterministas, adaptativas y autoadaptativas. Este trabajo se enfoca en las técnicas adaptativas, las cuales se basan en la selección de los mejores valores a través diversos mecanismos que ayudan a los agentes a aprender por medio del aprendizaje reforzado.

El aprendizaje reforzado es considerado como uno de los problemas más difíciles del aprendizaje máquina. Los algoritmos para solucionar éste tipo de problemas requieren muchos recursos en comparación con otro tipo de problemas. En éste tipo de aprendizaje un agente trata de aprender un comportamiento mediante interacciones de prueba y error en un ambiente dinámico e incierto. Las principales diferencias con otros tipos de aprendizajes son: no se presentan pares de entradas y salidas, el agente obtiene experiencia a través de los estados, acciones, transiciones y recompensas. La evaluación del sistema ocurre en forma concurrente con el aprendizaje [67,91].

En la Figura 2.4 se observa que un agente está conectado a un ambiente por medio de percepción y acción. En cada interacción el agente recibe como entrada una indicación de su estado actual ($s \in S$) y relaciona una acción ($a \in A$). La acción cambia el estado y el agente recibe una señal de refuerzo o recompensa ($r \in R$). El comportamiento del agente debe ser encaminado a escoger acciones que tiendan a incrementar a largo plazo las sumas de las recompensas totales [90,91].

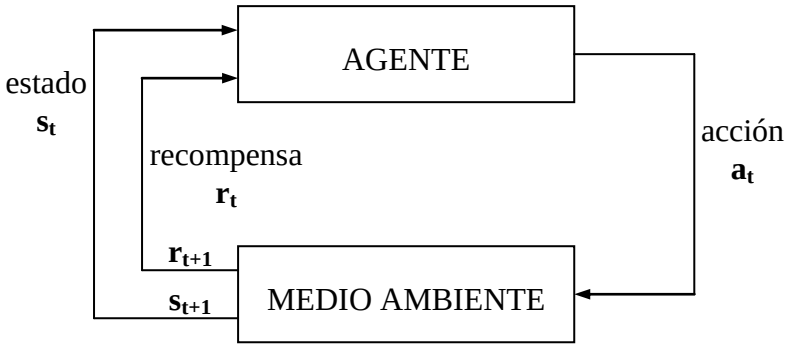


Figura 2.4. Aprendizaje por refuerzo

En los trabajos desarrollados por Sutton [90] se define el aprendizaje reforzado como aquel aprendizaje de -como hacer un mapa de situaciones para las acciones- así como maximizar una señal con una recompensa numérica. Este tipo de aprendizaje no dice cual acción tomar como en la mayoría de las técnica de aprendizaje máquina, en lugar de eso se deben descubrir que acciones se mantienen como las mas recompensadas para hacer uso de ellas.

En la mayoría de los casos las acciones pueden no tener efecto solamente en la recompensa inmediata sino a través de varias situaciones y así acumular un número de recompensas. Estas dos características -la búsqueda del intento y error, así como la recompensa retardada- son dos de las características más importantes del aprendizaje reforzado.

Para que un agente obtenga una buena ganancia es importante que tenga un balance entre *exploración* a través de descubrir o adaptar el valor de las acciones, y *explotación* donde

aprovecha lo que ya conoce. La caracterización de la problemática de encontrar el balance entre exploración y explotación está dada por procesos de decisión de Markov (PDM) [67,78].

Un MDP modela un problema de decisión secuencial en donde el sistema evoluciona en el tiempo y es controlado por un agente. La dinámica del sistema esta determinada por una función de transición de probabilidad que mapea estados y acciones a otros estados.

2.3.1 La propiedad de Markov

Los agentes toman decisiones como una función de una señal del medio ambiente llamada estado del medio ambiente. El estado significa, cualquier información que este disponible para el agente y es dado por algún sistema de reprocesamiento que es parte del medio ambiente. El estado proporcionará una señal para tomar una decisión de que acción realizar a través de una función.

Las señales del estado deben incluir medidas que puedan ser estructuradas sobre el tiempo de una secuencia de estados. Dichas señales no debe esperar informar al agente de todo lo que sucede en el medio ambiente para tomar una decisión. Si no que sólo se debe sumar sensaciones pasadas de manera compacta, información relevante. Es decir, sensaciones inmediatas que reflejen parte de la historia pasada. Una señal de estado exitosa contiene toda la información relevante y se dice que es Markov, o que tiene la propiedad de Markov [67,78,90].

La propiedad de Markov para el problema del aprendizaje reforzado asume que hay un número finito de estados y valores de recompensa y considera como un medio ambiente general debe responder en un tiempo t+1 para la acción tomada en un tiempo t. En este caso la dinámica puede ser definida solamente por la especificación de la distribución de probabilidad completa. Como se observa en la Ecuación (2.1).

$$\Pr \{ s_{t+1} = s' , r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots , r_1 , s_0 , a_0 \} \quad (2.1)$$

Para todo los estados y recompensas nuevas calculadas s', r y además todos los valores posibles de los eventos anteriores: s_t, a_t, r_t, s_{t-1}, a_{t-1}, . . . , r_1 , s_0 , a_0, si la señal del estado tiene una propiedad de Markov, entonces el medio ambiente responde en t+1, dependiendo solamente de la representación del estado y la acción en t, en tal caso el medio ambiente dinámico pude ser definido por la Ecuación (2.2).

$$\Pr \{ s_{t+1} = s' , r_{t+1} = r \mid s_t, a_t \} \quad (2.2)$$

Para todo estado y recompensa calculado y actual s', r, s_t y a_t, la señal del estado tiene una propiedad de Markov, si y solo si, la Ecuación (2.1) y (2.2) son equivalentes.

Si un ambiente tiene la propiedad de Markov, entonces su dinámica de un paso representada en la ecuación (2.2), permite predecir el estado y recompensa siguiente, dado el estado actual y acción. En otras palabras, la propiedad de Markov es importante en el reforzamiento del aprendizaje debido a que las decisiones y valores se determinan por una función solamente del estado actual [67,78,90].

2.3.2 Proceso de Decisión de Markov (PDM, por sus siglas en inglés)

Formalmente un PDM es una tupla M = < S, A, Φ, R > donde sus elementos básicos son:

- S = {s_1, s_2, s_3, ... , s_n}; conjunto finito de estados.
- A= { a_1, a_2, a_3, ... , a_n }; conjunto finito de acciones, que pueden depender de cada estado.
- R función de recompensa, define la meta, mapea cada estado-acción a una recompensa que indica la bondad del estado.
- Φ: A x S --> S ; modelo del ambiente. Es una función de transición de estados.

La tarea de aprendizaje reforzado que satisface la propiedad de Markov es llamada proceso de decisión de markov. Este proceso busca estimar las funciones de valor a través de evaluar la bondad de un estado o realizar la siguiente acción [91].

El aprendizaje reforzado es una estrategia usada por los algoritmos de colonia de hormigas, el cual es aplicado en la tabla de feromonas. La tabla de feromonas es una memoria a través de la cual se registran las rutas más usadas por la colonia de hormigas.

2.4 Estado del Arte: Ajuste de Parámetros de Algoritmos de Búsqueda

El problema de ajuste de parámetros analizado desde alguna de sus dos estrategias de solución, por si sólo representa un problema de optimización, el cual ha sido abordado desde varios enfoques por distintos investigadores, entre los cuáles destacan los siguientes.

Belardino Adenso Díaz [1] lleva a cabo experimentos usando afinación de parámetros con diseño de experimentos. El proyecto desarrollado se llama CALIBRA, y este permite al usuario sugerir valores para 5 parámetros. CALIBRA asume que las relaciones entre el desempeño y los parámetros son lineales y entonces empieza a obtener el efecto de cada factor (en magnitud y dirección sobre la respuesta) al realizar distintas combinaciones de posibles valores. El diseño de experimentos es utilizado en una etapa inicial y después la solución es mejorada mediante una búsqueda local entre las soluciones vecinas. El ajuste de parámetros se hace de manera global, es decir, sólo considera el desempeño final del algoritmo para ajustar sus valores.

Constantinos Antoniou [21] en su tesis doctoral presenta un enfoque basado en un modelo *espacio-estado* para la calibración de parámetros en línea para el sistema *DynaMIT-R* para el problema de *asignación de tráfico*. Utiliza tres métodos basados en el *Filtro de Kalman*: Filtro de Kalman Extendido (*Extended Kalman Filter*, EKF), EKF con límites y Filtro de Kalman sin Aroma (*Unscented Kalman Filter*, UKF). Para ser usada esta técnica, es necesario ajustar los datos a una distribución de probabilidad, ya que las correcciones que hace el algoritmo las hace entre el valor real y el valor de la distribución.

Mauro Birattari [2] en su tesis doctoral aborda el problema de ajuste de parámetros desde la perspectiva de aprendizaje automático y propone el algoritmo *F-Race*, el cual se ejecuta en paralelo con el algoritmo que soluciona el problema, construyendo un modelo estadístico que represente el comportamiento del algoritmo. También se proponen algunas guías para mejorar la metodología de la investigación en el campo de los metaheurísticos. Aplica el ajuste a dos algoritmos: *búsqueda local* para el problema de *asignación cuadrática* con nueve parámetros y MMAS para TSP con cinco parámetros.

Enda Ridge [18] presenta un método que utiliza *Diseño de Experimentos* para construir un modelo predictivo del desempeño de un heurístico basado en la configuración de los parámetros y las características de las instancias. Aplica su método al algoritmo ACS para el *Problema del Agente Viajero* (*Traveler Salesman Problem*, TSP), considerando 10 parámetros a configurar y dos características de las instancias de TSP. Crea *Modelos de la Superficie de Respuesta* (*Response Surface Model*, RSM) tanto para el desempeño como para el tiempo de solución, y son usados para encontrar la configuración de parámetros que permiten el mejor desempeño del algoritmo en términos de calidad y tiempo.

Michlmayr [5] 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.

En la técnica de ajuste **global**, los parámetros de un algoritmo toman valores promedio de los datos ya que el problema que se resuelve se aborda como si fuera homogéneo en todos sus puntos, no distinguiendo los subgrupos inherentes. En otras palabras, aplica la misma estrategia de solución aunque las condiciones cambien de un punto a otro [5][18].

La Tabla 2.1 presenta las características de los trabajos más relevantes del estado del arte. En la primera columna aparece el nombre del investigador y el año. Enseguida, las columnas dos a la seis identifican los investigadores que han abordado, respectivamente: el análisis de las características del problema, el tipo de dato utilizado en el ajuste de parámetros (local o global) y el tiempo en el que se efectúa el ajuste de parámetros (antes o durante la ejecución del algoritmo). Finalmente se señala quienes trabajan con SQRP y el algoritmo que utilizan.

Como resultado del análisis de la Tabla 2.1 se observa, que hasta este momento, solamente dos investigadores han abordado SQRP con algoritmos cuyos parámetros son configurados globalmente, resolviendo parcialmente el problema del ajuste de parámetros.

Los trabajos que usan el enfoque de ajuste local lo aplican a algoritmos genéticos que resuelven problemas estáticos; el objetivo es adecuar los parámetros al estado actual de la solución y a características puntuales del problema. En esos trabajos no se considera la dinámica del problema, ni las características de entornos heterogéneos [22]. A diferencia de los trabajos previos, esta tesis sí propone una función para el ajuste local de parámetros e incorpora los principales aspectos de los trabajos revisados en dos algoritmos de búsqueda semántica: NAS y AdaNAS. Con fines de prueba, la función propuesta para el ajuste de parámetros se incorporó en

el algoritmo de búsqueda AdaNAS, que por su composición general pudiera ser integrada en algoritmos que resuelven otros problemas de optimización de naturaleza dinámica y heterogénea.

Tabla 2.1 Estado del Arte del Ajuste de Parámetros

Investigador	Análisis de las Características del Problema a Solucionar	Datos del Problema de		Ajuste de Parámetros		Técnica de Ajuste de Parámetros	SQRP	Algoritmo de Colonia de Hormigas
		Locales	Globales	Antes	Durante			
Birattari 2004			✓	✓		Meta-Evolución (F-Race)		
Adenso-Díaz 2001			✓	✓		DOE (Diseño Factorial Fraccional)		
Bo Yuan 2006			✓	✓		Meta-Evolución (Racing Meta-EA)		
Wu 2007			✓		✓	Reglas Deterministicas	✓	AntSearch
Ridge 2007	✓		✓	✓		DOE (Diseño Factorial Fraccional)		
Michlmayr 2007			✓	✓		Fuerza Bruta	✓	SemAnt
Antoniou 2004			✓		✓	Adaptacion (Filtro de Kalman)		
Gómez 2009 en esta tesis	✓		✓	✓		DOE (Diseño factorial)	✓	NAS
Gómez 2009 en esta tesis	✓	✓	✓	✓	✓	DOE y Control Adaptativo (Aprendizaje, caracterización)	✓	AdaNAS

Capítulo 3

METODOLOGÍA PROPUESTA PARA EL AJUSTE DE PARÁMETROS

En este capítulo se presenta una metodología propuesta para resolver el problema de ajuste de control adaptativo del parámetro *TTL*. Este parámetro define el alcance de transmisión de los algoritmos de búsqueda en redes P2P y tiene un efecto significativo en el desempeño algorítmico.

3.1 Metodología de Solución

Esta sección describe detalladamente la metodología propuesta para el ajuste de parámetros de algoritmos metaheurísticos, en particular para algoritmos que dan solución al problema de búsqueda semántica SQRP. La metodología esta diseñada en cuatro etapas. La segunda etapa se inspiró en el trabajo de Ridge [18], en el cual se trabaja solo el ajuste global a través de DOE. La parte innovadora de la presente investigación es la propuesta de ajuste local adaptativo que se incorpora en las etapas posteriores al ajuste global.

Los pasos y productos de cada etapa se relacionan en las Tablas 3.1 y 3.2, respectivamente.

La etapa inicial de la metodología es la integración de una infraestructura de referencia (benchmark). Esta etapa incluye la generación de instancias de SQRP, el desarrollo de un algoritmo de solución basado en colonias de hormigas (NAS) y un algoritmo de búsqueda clásico (Random-Walk).

En la segunda etapa se identifican factores que impactan al desempeño algorítmico, así como relaciones entre estos factores que también inciden en el desempeño.

En la tercera etapa se formula una función para el ajuste de control adaptativo del parámetro *TTL*, la cual se incorpora en un nuevo algoritmo (AdaNAS) derivado del algoritmo de referencia NAS.

La metodología propuesta finaliza con la evaluación de la eficiencia del algoritmo AdaNAS con respecto a los algoritmos de referencia. Las etapas de la metodología se muestran en la Tabla 3.1 y en la Tabla 3.2 se complementa la metodología con el producto obtenido por actividad.

Tabla 3.1 Etapas de la metodología general

Etapa 1. Integración de una infraestructura de referencia
a. Caracterización de instancias reales, b. Generación de instancias, c. Desarrollo de algoritmos de referencia: Neighboring Ant-Search (NAS) y Random-Walk (RW).
Etapa 2. Identificación de factores de desempeño y sus relaciones
a. Selección de las características de SQRP que afectan el rendimiento del algoritmo NAS, b. Preparación del algoritmo NAS para la selección de parámetros significantes, c. Selección de los parámetros significantes del algoritmo NAS, d. Modelado de las relaciones entre los parámetros del algoritmo NAS y las características de SQRP.
Etapa 3. Desarrollo de un sistema de búsqueda adaptativo
a. Diseño de un modelo adaptativo basado en agentes, b. Desarrollo de un algoritmo adaptivo: AdaNAS, *c.* ***Diseño de la función de ajuste de control adaptativo del parámetro TTL***
Etapa 4. Evaluación de la eficiencia del algoritmo adaptativo AdaNAS a. Pruebas de diferencias significativas usando Wilcoxon

Cabe mencionar que para el desarrollo de la metodología fue necesario es necesario identificar y describir algunos elementos considerados claves para el diseño de la función de control dinámica local:

- *Espacio o ambiente del problema*, es fundamental identificar el tipo de ambiente en el que se va a trabajar, si es estático o dinámico. El objetivo es buscar un modelado para responder la solución. En este problema el ambiente es una red P2P compleja con distribución de probabilidad power-law estática.

- *Espacio de los parámetros*, son todos las posibles valores de las combinaciones de los parámetros a justar.

- *Métricas de rendimiento múltiple*, el rendimiento debe ser analizado en términos de la calidad y el tiempo de la solución. En este problema las métricas de rendimiento se establecieron de la siguiente forma:

  a) *Promedio de los saltos* (*HOPS*), cantidad promedio de enlaces viajados (o saltos) por un agente de búsqueda hasta que este muere.
  b) *Promedio de los éxitos* (HITS), definido como el promedio del número de recursos encontrado (éxitos) por cada agente de búsqueda hasta su muerte.
  c) *Promedio de la eficiencia*, definido como el promedio del porcentaje de los éxitos dividido por el promedio de los saltos (éxitos/saltos).

Tabla 3.2 Descripción de las etapas con su correspondiente producto

Etapa	**Producto**
1. Integración de una infraestructura de referencia	
Caracterización de instancias reales usando DOE y análisis multivariado.	Topología caracterizada con *DDC*.
Generación de instancias usando información de la literatura.	Consultas con distribución aleatoria (grado de repetición de la consulta), Grafo con distribución power-law (grado de conexiones del grafo = topología) Repositorios con distribución power-law (grado de repetición de llaves en los rep.)
Desarrollo de algoritmos de referencia usando información de la literatura: NAS y RW.	Algoritmos de búsqueda NAS: DDC, Regla de transición, Funciones de aprendizaje de corto alcance y Tabla de aprendizaje de largo alcance (feromona).
2. Identificación de factores de desempeño y sus relaciones	
Selección de características de SQRP que afectan el rendimiento del algoritmo NAS usando DOE.	Factor significante: topología. Relaciones: repositorios y consultas.
Preparación del algoritmo NAS para la selección de parámetros significantes usando información de la literatura.	Topología caracterizada con el grado de las conexiones. Parámetros de control de factores de desempeño: β_1 (intensificación de medidas locales), β_2 (intensificación de feromona), w_h (factor de equilibrio entre *hits* y *hops*), w_d (importancia del grado) y w_i (importancia de la distancia).
Selección de los parámetros significantes del algoritmo NAS usando modelado causal.	q (influye sobre *hits* y *hops*),W_d, β_1 , ρ y ***TTL*** (influyen sobre la relación *Hits*/*Hops*).
Modelado de las relaciones entre los parámetros del algoritmo NAS y las características de SQRP.	Regla de transición modificada, Tabla de aprendizaje de corto alcance modificada.
3. Desarrollo de un sistema de búsqueda adaptativo	
Diseño de un modelo adaptativo basado en agentes	Modelo adaptivo de la búsqueda.
Desarrollo de un algoritmo adaptivo: AdaNAS.	Grado, Regla de transición modificada, Tablas de aprendizaje de corto y largo alcance (feromona), Parámetros de control de factores de desempeño.
Diseño de la función de ajuste de control adaptativo del parámetro TTL	Configuración-voraz: q=1, w_d =0, β_2 =0, Regla de transición modificada-voraz, Función de ajuste para TTL.
4. Evaluación de la eficiencia del algoritmo adaptativo AdaNAS	
Evaluación eficiencia	Tablas y gráficas comparativas con sustento estadístico de la prueba de Wilcoxon.

3.2 Integración de la infraestructura de referencia

La infraestructura de referencia esta formada por las instancias artificiales de prueba y los algoritmos de referencia necesarios para este proyecto.

3.2.1 Caracterización de instancias reales

La primera actividad propuesta en esta investigación es identificar características topológicas que permitan distinguir cuantitativamente en cada punto de la red su topología local, para que de esta manera procesos como la búsqueda puedan adaptarse a las características locales cambiantes. La topología es un factor importante debido a que es el ambiente en el cual navegan los agentes hormiga.

Se obtuvieron instancias reales de subgrafos de la Internet disponibles en [92]. Estas instancias reflejan la topología de Internet a nivel de sistemas autónomos para los años de 1997, 2000 y 2003. En [16] se propone la métrica local DDC para medir la dispersión del grado de una vecindad. A partir de esta métrica se propone un método de caracterización que permite identificar la topología local o global de la red. El algoritmo de la Tabla 3.3 determina la topología global. Si el grafo es de un solo nodo, la topología obtenida es local.

Tabla 3.3 Método de caracterización que permite identificar la topología local o global de la red

Entrada: Instancias reales de 1997, 2000 y 2003. **Salida:** Tipo de red clasificada.
1. Para cada uno de los nodos *i* de la red, determinar la dispersión del grado 1.1. Calcular $DDC(i)$ usando: $DDC(i) = \frac{\sigma(i)}{\mu(i)}$. 1.2 $SumDDC = SumDDC + DDC(i)$. 2. Calcular $DDC(G)$ usando: $DDC(G) = \frac{SumDDC}{n}$. 3. Clasificar $DDC(G)$ bajo los siguientes criterios [93]: 3.1. Si $DDC(G) \in [0.015 - 0.3025]$, Tipo de red = Aleatoria. 3.2 Si $DDC(G) \in [0.3026 - 0.4718]$, Tipo de red = Exponencial. 3.3 Si $DDC(G) \in [0.4719 - 1.2]$, Tipo de red = Scale-free. 4. Regresar (*Tipo-de-red*) 5. Terminar

La descripción del tipo de red puede ser interpretado a través de la cantidad de conexiones para cada uno de los nodos de la red, como sigue:

- En la redes con distribución del grado aleatorias, $DDC(G) \in [0.015 - 0.3025]$. Esto es debido a que la mayoría de los nodos tienen aproximadamente el mismo grado, cercano al grado promedio de la red $\langle k \rangle$; se puede decir que la dispersión del grado en promedio es nula o muy pequeña.

- En la redes con distribución del grado exponenciales, $DDC(G) \in [0.3026 - 0.4718]$. Esto es debido a que la mayoría de los nodos tienen un grado cercano al grado promedio de la red $\langle k \rangle$ y se pueden observar un número muy reducido de nodos con un alto grado por arriba de $\langle k \rangle$.

- Para las redes scale-free $DDC(G) \in [0.4719 - 1.2]$. Esto es debido a que solo unos cuantos nodos poseen un grado muy alto y la mayoría de los nodos un grado muy pequeño, por lo que la dispersión del grado en promedio tiende a ser alta.

Los rangos a través de los cuales se hace la clasificación de redes son establecidos en el trabajo de Turrubiates [93] utilizando DOE y técnicas de análisis multivariado. En ese trabajo se comprobó que la métrica DDC aplicada como un clasificador tiene una confiabilidad del 99.8%, para clasificar la topología de una instancia real o artificial.

3.2.2 Generación de Instancias

Para contar con instancias SQRP, éstas se generaron artificialmente con características tomadas de la literatura [17,70,5,23]. De acuerdo con la definición de SQRP, dada en la introducción, para cada instancia se requiere la red, los repositorios y las consultas que acceden a los repositorios.

Los generadores de redes, repositorios y consultas, se diseñan utilizando los métodos descritos a continuación:

a. Generación de Redes Aleatorias

El modelo de Erdös y Rényi [35,41] es un modelo sin crecimiento que comienza con *n* nodos desconectados, cada par de nodos es conectado con una probabilidad *p*. Por tanto el número total de enlaces es un variable aleatoria con un valor esperado de $p[n(n-1)/2]$. El algoritmo del modelo de Erdös y Rényi se muestra en la Tabla 3.4.

Con este modelo la red aleatoria generada sigue una distribución del grado binomial con parámetros $N-1$ y p, como se observa en la Ecuación (3.1):

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

Para valores grandes de n, la distribución del grado sigue una distribución de Poisson, Ecuación (3.2). De ahí que la probabilidad de que un vértice tenga grado k es:

$$P(k) \approx \frac{\langle k \rangle^{k} e^{-\langle k \rangle}}{k!} \qquad (3.2)$$

donde $\langle k \rangle = 2e/n = p(n-1)$ [9, 35,42].

Tabla 3.4. Algoritmo del Modelo Erdös y Rényi

Entradas :	*N* = Número de nodos
	P= Probabilidad de enlace [0,1]
	Na= Número aleatorio entre [0,1]
Generador_de_Redes_Aleatorias (*n*)	
1	**Inicio**
2	**Inicializar** M[n,n] = {0};
3	**Para** (*i* =1, y *j* = *i*+1) hasta n
4	**Si (** *i* > *j* **) y (** *P* > *Na* **)**
5	M[*i* , *j*] = 1;
6	*i* = *i* + 1*;*
7	*J* = *j* + 1;
8	**End_Para**

b. Generación de Redes Scale-free

Después de haber estudiado el comportamiento de muchas redes reales hacia 1999, se introduce el modelo Barabási-Albert [45] inspirado en el crecimiento y el enlace preferencial, este fue el primer modelo que reprodujo redes con una distribución del grado scale-free. El modelo considera dos elementos: crecimiento y enlace preferencial.

- El *crecimiento* comienza con un pequeño número de nodos m_0. A cada paso *t* se añade un nuevo nodo con $m \leq m_0$ aristas que enlacen al nuevo nodo con *m* diferentes nodos ya existentes en el sistema.

- El *enlace preferencial*, mostrado en la Ecuación (3.3), se aplica cuando se selecciona los nodos a los cuales un nuevo nodo se va a conectar. Se asume que la probabilidad Π de que un nuevo nodo sea conectado al nodo *i* depende del grado k_i del nodo *i*, tal que:

$$\Pi(k_i) = \frac{k_i}{\sum_{j \in N} k_j} \qquad (3.3)$$

  donde *N* es el conjunto de nodos en la red.

- Después de *t* lapsos de tiempo, el procedimiento resulta en una red con $n = t + m_0$ nodos con *m.t* aristas.

c. Generación de Repositorios y Consultas

En las redes P2P, cada punto administra un *repositorio local* (*R*) de recursos y ofrece sus recursos a otros puntos de la red. De acuerdo con la literatura, el contenido de los repositorios en Internet

sigue una distribución del grado de repetición de llaves power-law. En otras palabras, unos pocos nodos contienen muchos tópicos en sus repositorios y el resto de los nodos contiene pocos tópicos.

Para la generación de las *consultas* (*Q*), siguiendo las recomendaciones de la literatura, cada nodo tiene una probabilidad de 0.1 de lanzar una consulta, seleccionando un tópico aleatoriamente dentro de una lista de posibles tópicos del nodo. La distribución del grado de la activación consultas *Q* es aleatoria y determina que tan a menudo se repite la consulta de una misma clave en la red.

La topología de la red y los repositorios son estáticos durante la ejecución del algoritmo de búsqueda. Mientras que las consultas son lanzadas aleatoriamente durante la simulación.

3.2.3 Desarrollo de algoritmos de referencia

Como última actividad de esta etapa se describe la implementación realizada de los algoritmos de referencia: el metaheurístico de colonia de hormigas Neighboring-Ant Search (NAS) con ajuste global de parámetros y el algoritmo Random-Walk (RW). Estas versiones son diseñadas usando ajuste de parámetros con enfoque manual, es decir se usan valores globales para los parámetros, en los cuales los valores no cambian durante la ejecución del algoritmo independientemente del tipo de subestructura que se este analizando en cada nodo.

Para los dos algoritmo NAS y RW, la configuración inicial es tomada de otras investigaciones reportadas en la literatura [5,18,24]. Posteriormente se valida el funcionamiento de los algoritmos a través de una serie de experimentos descritos en el capítulo 4.

a. Pseudocódigo del algoritmo NAS

NAS es un algoritmo metaheurístico, donde un conjunto de agentes independientes llamados hormigas cooperan independiente y esporádicamente para lograr una meta común. El algoritmo tiene dos objetivos: busca maximizar el número de recursos (*hits*) encontrados por las hormigas y minimizar el número de saltos (*hops*) dados por la hormiga. NAS guía las consultas hacia nodos que tienen mejor conectividad usando la métrica estructural local DDC [100]. Ya que el DDC, para minimizar la cantidad de saltos, mide las diferencias entre el grado del nodo y el grado de sus vecinos, entre mas frecuentemente una consulta sea llevada hacia un recurso, una mejor ruta será seleccionada. Esto es, el porcentaje de optimización de una consulta depende directamente de su popularidad.

El algoritmo NAS ejecuta en paralelo todas las consultas usando agentes de consulta. El rol de cada agente de consulta es crear el agente de búsqueda cuando una consulta es lanzada desde un nodo. La actividad de cada agente de búsqueda consiste de dos fases principales.

La **Fase 1** corresponde a la evaluación de resultados (líneas 04-10 en el pseudocódigo de la Tabla 3.5). implementa la técnica clásica Lookahead. Esto es, una hormiga *k* llamada agente de búsqueda y localizada en el nodo r_k, verifica si el recurso existe en un nodo no visitado s_k que viene a ser de su vecindario incluyéndose ella misma. Si el recurso es encontrado, la hormiga añade el nodo s_k a su ruta, actualizando el número de ocurrencias del recurso consultado HIT_k,

reduce el tiempo de la hormiga TTL_k en el costo del enlace de r_k a s_k, realiza la actualización de la feromona local, y realiza la actualización de la feromona global. La actualización global de las feromonas es una actividad concurrente de las hormigas llamadas agentes de recuperación; la ruta al recurso es actualizado en la tabla de feromonas y regresada al usuario final. En el caso de que la fase de evaluación falle, el algoritmo continua con la Fase 2.

En la **Fase 2** el estado de transición (líneas 11-19 del Pseudocódigo en Tabla 3.5) es llevada a cabo. Esta fase selecciona mediante números aleatorios *q*, un nodo vecino *s*. En el caso de que no existan nuevos nodos hacia los cuales moverse, esto quiere decir, el nodo es una hoja o todos los nodos vecinos han sido visitados, un salto hacia atrás es llevado a cabo en la ruta, de otra manera la hormiga añade el nodo seleccionado s_k a su ruta actualizando localmente la feromona, y reduce TTL_k en un salto. El proceso de consulta termina cuando el número esperado de resultados ha sido encontrado o TTL_k llega a cero. En ambos casos el agente de búsqueda es matado indicando el fin de la consulta.

Tabla 3.5. Pseudocódigo del algoritmo NAS

```
01  en_paralelo  { // Actividad Concurrente del agente de consulta
02        Para cada consulta en r_k crear agente de búsqueda k con TTL_k=maxTTL y Hits_k = 0;
03           Mientras Hits_k < maxResults y TTL > 0;  // Actividad Concurrente del agente de búsqueda
04                    // Paso 1: Evaluación de Resultados
05                    Si  el nodo no se ha visitado s_k ∈ { r_k ∪ Γ( r_k)} y tiene el recurso búscado // Estrategía Lookahead
06                          r_k = agregar s_k a la ruta_k
07                       Hits_k = Hits_k +1
08                       TTL_k = TTL_k - CC_rs
09                          Actualización Feromona Local
10                       Actualización Feromona Global  // Actividad Concurrente del agente de   recuperación
11                    De lo contrario // Paso 2: Estado de Transición
12                          s_k = aplicar la regla de transición con la función DDC
13                          Si r_k es un nodo hoja o no tiene vecinos sin  visitar  s_k,
14                              remover el último nodo de la ruta_k
15                          De lo contrario
16                              r_k  = agregar s_k a la ruta_k
17                            TTL_k = TTL_k - CC_rs
18                              Actualizacion Feromona Local
19              Matar el agente de búsqueda
```

b. Pseudocódigo del Algoritmo RW

El algoritmo RW es un algoritmo de búsqueda ciego, es decir, no cuenta con información adicional como tablas de enrutamiento que le proporcionen información para guiar la búsqueda. En esta versión del algoritmo RW le fue incluido la técnica clásica de exploración llamada Lookahead para explorar rápidamente los nodos vecinos. El algoritmo RW ejecuta en paralelo todas las consultas usando agentes.

La **Fase 1** corresponde a la evaluación de resultados (líneas 04-8 en el pseudocódigo de la Tabla 3.6). implementa la técnica clásica Lookahead. Esto es, *k* es llamada agente de búsqueda y localizada en el nodo r_k, verifica si el recurso existe en un nodo no visitado s_k que viene a ser de su vecindario incluyéndose ella misma. Si el recurso es encontrado, el agente añade el nodo s_k a su ruta, actualizando el número de ocurrencias del recurso consultado HIT_k, y reduce el tiempo del agente TTL_k.

En la **Fase 2** el estado de selección del siguiente nodo (líneas 9-15 del Pseudocódigo en Tabla 3.6) es llevada a cabo. Esta fase selecciona aleatoriamente un vecino que no haya sido visitado *s*. En el caso de que no haya nuevos nodos hacia los cuales moverse, esto quiere decir, que el nodo es una hoja o todos los nodos vecinos han sido visitados, un salto hacia atrás es llevado a cabo en la ruta, de otra manera el agente añade el nodo seleccionado s_k a su ruta y reduce TTL_k en el costo del enlace de r_k a s_k. El proceso de consulta termina cuando el número esperado de resultados ha sido encontrado o TTL_k llega a cero. En ambos casos el agente es matado indicando el fin de la consulta.

Tabla 3.6. Pseudocódigo del algoritmo RW

```
01  en_paralelo  { // Actividad Concurrente del agente
02        Para cada consulta en r_k crear agente  k con TTL_k=maxTTL y Hits_k = 0;
03           Mientras Hits_k < maxResults y TTL > 0;  // Actividad Concurrente del agente
04                    // Paso 1: Evaluación de Resultados
05                   Si  el nodo no se ha visitado s_k ∈ { r_k ∪ Γ( r_k)} y tiene el recurso búscado // Estrategía Lookahead
06                        r_k = agregar s_k a la ruta_k
07                      Hits_k = Hits_k +1
08                      TTL_k = TTL_k - CC_rs
09                   De lo contrario // Paso 2: Selección del siguiente nodo
10                        s_k = seleccionar el siguiente nodo aleatoriamente y que no se haya visitado s_k ∈ { r_k ∪ Γ( r_k)}
11                        Si r_k es un nodo hoja o no tiene vecinos sin  visitar  s_k,
12                            remover el último nodo de la ruta_k
13                        De lo contrario
14                            r_k  = agregar s_k a la ruta_k
15                          TTL_k = TTL_k - CC_rs
16              Matar el agente
```

Las experimentaciones y resultados de los dos algoritmos son incluidos en el capítulo 4.

3.3. Identificación de factores de desempeño y sus relaciones

Para comprobar la suposición de relaciones entre los elementos que se sospecha influyen en el desempeño del algoritmo, se realizar un análisis estadístico a través de diversas estrategias como: diseño de experimentos factorial, análisis de correlaciones, análisis multivariado [80]. Estos tipos de diseños son utilizados para identificar los efectos de las interacciones de un conjunto de factores relacionados con una variable respuesta.

3.3.1 Selección de las características de SQRP que afectan el rendimiento del algoritmo NAS.

En esta sección se identifican las características principales de SQRP que influyen en el rendimiento del algoritmo NAS. Es importante identificar las características que influyen en el rendimiento debido a que dicha influencia puede ser positiva o negativa. De manera que la influencia de los parámetros en el rendimiento pude hacer que este aumente o disminuya.

a. Características de SQRP

La consulta viaja a través de la red moviéndose desde el nodo fuente a un nodo vecino y después a un vecino del vecino y así hasta localizar la información requerida, o darla como ausente. El reto descansa en el diseño de algoritmos que naveguen en la Internet de manera inteligente y autónoma. Para alcanzar esta meta, el algoritmo NAS selecciona el siguiente nodo a visitar usando información de los nodos cercanos al nodo actual, por ejemplo, información de la topología local del nodo actual [24].

Se consideraron los tres elementos de SQRP y sus distribuciones del grado para estudiar los factores de impacto en el rendimiento del algoritmo NAS. El primer factor seleccionado es la distribución del grado de las conexiones de los nodos, el cual define la topología de la red. Aunado a este factor se selecciona, la distribución del grado de las repeticiones de llaves de los repositorios. Finalmente se incorpora la distribución del grado de la activación de las consultas, la cual define cuantas veces la misma consulta es repetida dentro de los diferentes nodos de la red.

Para simplificar la descripción del experimento, los siguientes nombres cortos serán usados respectivamente: distribución de topología, repositorios y consultas. Las distribuciones más comunes sobre la Internet son: power-law [45] y aleatoria [94]. La distribución power-law se usará en la topología [43] y los repositorios [95,96], para las consultas se usará una distribución aleatoria [5,13].

Los supuestos de la investigación para este experimento pueden ser establecidas como a partir del siguiente cuestionamiento: ¿ Afecta el conjunto de características de SQRP el rendimiento del algoritmo NAS?. Estos supuestos son establecidos de la siguiente forma:

$\mathbf{H_o}$: La distribución de la topología, repositorios y consultas de SQRP, No tienen efectos significantes en la eficiencia del algoritmo NAS.

$\mathbf{H_1}$: La distribución de la topología, repositorios y consultas de SQRP, tienen efectos significantes en la eficiencia del algoritmo NAS.

b. Diseño Experimental

Los supuestos establecidos en el punto anterior requieren estudiar todas las posibles combinaciones de las características de SQRP contra los tipos de distribuciones de probabilidad establecidos en el capítulo 1. Cuando es necesario estudiar los efectos de dos factores, la literatura recomienda llevar a cabo un experimento factorial completo 2^k. En esta clase de experimentos es necesario identificar los factores con sus niveles, las variables de respuesta y las replicas. El impacto de los niveles del factor es medido a través de las variables de respuesta en cada replica [80].

Dos ***factores*** fueron estudiados: el tipo de distribución de probabilidad y las características de SQRP. Los ***niveles del primer factor*** son: scale-free (SF) y random (RM). Los ***niveles para el segundo factor*** son: la topología, repositorios y consultas.

Las variables de respuesta son: ***respuesta*1** que es el promedio de la cantidad de información encontrada (*hits*) y ***respuesta*2** que es el promedio del número de saltos dados para buscar la

información solicitada (*hops*). En el experimento, las ***replicas*** son la diferentes corridas del algoritmo metaheurístico NAS. Bajo estas condiciones se determinó que el detalle de este estudio se presenta en el experimento de la selección de características de SQRP del capítulo 4.

3.3.2 Preparación del algoritmo NAS para la selección de parámetros significantes.

Para la selección de los factores significantes de NAS, se hace un análisis de éstos parámetros en las reglas de transición y actualización, con el propósito de hacer mejoras en su diseño.

En el algoritmo NAS, en su regla de transición se incluyó la métrica topológica local llamada DDC [16] que esta definida en la sección 1.2.1. Esta métrica local mide la dispersión entre el grado de un nodo y sus vecinos, se espera que la métrica guíe al algoritmo hacia nodos que tengan el mayor número de conexiones, es decir que tengan muchos vecinos. Otra métrica relacionada es el grado de conexiones del nodo k_i , que está definida en la sección 1.2.1.

En ésta sección se prueban las dos métricas para identificar cual de ellas guía mejor al algoritmo NAS hacia nodos mejor conectados. La métrica topológica local seleccionada se incluye en la regla de transición que forma parte del algoritmo NAS.

Para llevar a cabo el estudio se incorporaron en la regla de transición cada una de las métricas a evaluar por separado. Obteniéndose la eficiencia del algoritmo NAS para cada una de ellas. Esto se llevó a cabo con diferentes instancias para probar cual de las dos métricas es la que aporta mayor información en la función que guía el algoritmo hacia nodos mejor conectados.

La variable de respuesta es el promedio de la eficiencia final del algoritmo NAS, y se midió ejecutando el algoritmo 30 veces con la misma configuración sobre el conjunto de instancias SQRP. La eficiencia del algoritmo se mide a través del promedio de la tasa de aprendizaje *hits/hops,* que indica cuantos recursos encontró el algoritmo en cada nodo que visitó. El conjunto de instancias SQRP son las descritas en la sección 4.2.2 en la que se presenta información detallada sobre su generación.

En total se utilizaron 90 instancias SQRP, el repositorio local de recursos su contenido sigue una distribución del grado de repetición de llaves power-law. Para las consultas, la distribución del grado de la activación consultas es aleatoria. La topología de la red y los repositorios son estáticos durante la ejecución del algoritmo de búsqueda. Mientras que las consultas son lanzadas aleatoriamente durante la simulación.

En este punto se espera identificar la métrica que mejor guíe a el algoritmo hacia nodos con mayor número de conexiones, esperando que estos nodos por las características de la topología pueden garantizar satisfacer las condiciones de la consulta rápidamente.

Otro análisis efectuado es en el dominio de cada uno de los parámetros de las reglas de transición y actualización de la feromona del algoritmo NAS. Las reglas están formadas por un conjunto de parámetros: la tabla de aprendizaje de largo alcance (feromona) τ, el DDC ó el grado del nodo y un conjunto de funciones de aprendizaje de corto alcance, que son: la métrica que evalúa las distancias ID_HOP, los éxitos HIT e importancia del tiempo de vida ITL_HOPS. Al combinar éstos parámetros las diferencias entre los rangos de los dominios en algunos casos son grandes y

eso hace que el algoritmo seleccione mas a unos parámetros que a otros. Debido a esto se analiza los rangos de valores de cada uno de los parámetros se normalizan para que todos se muevan dentro del mismo rango los que se encuentren juntos en un mismo término.

Aunado a la normalización de los rangos se identificó que en algunos casos es necesario equilibrar o dar mayor importancia a algunos de los factores de control de la regla de transición. Se analizan cada uno de los parámetros de la regla de transición para identificar la importancia de cada uno y rediseñar la ecuación en caso de ser necesario agregando parámetros de control de factores del desempeño, que ayudaran a intensificar los valores de algunos parámetros.

3.3.3 Selección de los parámetros significantes del algoritmo NAS

Michlmayr [5] llevó a cabo el análisis de los parámetros de su algoritmo metaheurístico llamado SemAnt. Aplicando la técnica de ajuste global denominada manual o fuerza bruta, a cuatro parámetros de su algoritmo SemAnt. Una de las desventajas de su experimentación fue no considerar las relaciones entre los parámetros, es decir, solo analizaba un parámetro a la vez sin considerar si su valor influía en los otros parámetros.

El estudio de las relaciones de los parámetros es importante ya que existen casos donde los parámetros estudiados de manera independiente no son significantes, pero al combinarlos con otros parámetros pudieran volverse significantes para el rendimiento del algoritmo [18].

Este experimento se hizo a través de un análisis causal, dicha técnica pertenece a las estrategias de análisis multivariado. Esta técnica fue seleccionada debido a la cantidad de parámetros que se tienen que observar. No se aplicó diseño factorial debido a que se generarían 3^9 combinaciones para ser analizadas. El análisis que se aplico en esta sección fue el análisis causal. Para llevar a cabo éste al igual que en DOE se corren primero los experimentos y después se analizan los datos.

El resultado de este análisis es un grafo causal, el cual muestra las relaciones entre los parámetros que se están analizando. La ventaja de usar esta estrategia también cae dentro del ajuste global de parámetros y la cantidad de experimentos necesarios para encontrar las relaciones y su influencia en el rendimiento fue de 800 corridas, y con diseños factoriales se hubieran necesitado 19683 corridas.

En el algoritmo NAS se identificaron nueve parámetros que definen parte de su comportamiento algorítmico, dos de ellos son ID_HOP_0 y τ_0. Éstos parámetros son usados para inicializar estructuras de aprendizaje a corto y largo plazo. Los valores de inicialización deben ser valores pequeños cercanos a cero, razón por lo cual no se considerarán estos parámetros en la búsqueda de la configuración inicial del algoritmo y se tomarán los valores de ellos tal y cual se recomiendan por la literatura $ID\_HOP_0 = 0.001$ y $\tau_0 = 0.009$.

Los supuestos por corroborar en este experimento giran alrededor de los parámetros de NAS y su significancia en él. Se dice que un parámetro es significante cuando al haber un cambio en su valor existe un impacto, positivo o negativo, sobre el desempeño del algoritmo. Para este experimento, se consideran un conjunto de parámetros de control de factores de desempeño los

cuales son: β_1, β_2, w_h, w_d, w_i y además dos parámetros que son parte del algoritmo de colonia de hormigas $\boldsymbol{\rho}$, $\boldsymbol{q}$.

Los parámetros estudiados son: q (establece la relación entre la exploración y la explotación), ρ (factor de evaporación), β_1 (intensificación de medidas locales), β_2 (intensificación de feromona), w_h (factor de equilibrio entre HIT e ITL_HOPS), w_d (importancia del grado) y w_i (importancia del ID_HOP).

La *variable de respuesta* es el promedio de la eficiencia del algoritmo NAS al ejecutarlo 30 veces con la misma configuración sobre el conjunto de instancias SQRP. El desempeño del algoritmo se mide a través del promedio de la eficiencia (*hits/hops)*, que indica cuantos recursos encontró el algoritmo en cada nodo que visitó. El conjunto de instancias SQRP son las descritas en la sección 4.2.3 en la que se presenta información detallada sobre su generación. El resultado esperado para este experimento es el grafo causal, a través del cual se muestran las relaciones de influencia de los parámetros. Los detalles del experimento se muestran en el capítulo 4.

3.3.4. Modelado de las Relaciones entre los Parámetros del Algoritmo NAS y las Características de SQRP

En esta sección se usa la información obtenida en secciones anteriores con el propósito de relacionar los parámetros del algoritmo NAS con las características de SQRP. Esta actividad es necesaria para el diseño de las estructuras que servirán de base para la creación de la función de ajuste adaptativa para el tiempo de vida (TTL) .

La topología es uno de los resultados importantes de las características de SQRP, la cual esta siendo caracterizada mediante la métrica del grado del nodo. Ésta característica ya esta incluida en la regla de transición. Ahora se analizan las relaciones de las características de SQRP, las cuales son repositorios y consultas. En esta sección se diseña una nueva estructura para la función de la distancia hacia el siguiente éxito medida en saltos *ID_HOP*. Esta función califica a cada nodo vecino del nodo actual, dependiendo de la distancia hacia el recurso encontrado en consultas previas, calificando mejor a los recurso más cercano a partir del nodo vecino [17].

Para éste experimento se diseño la estructura de datos perteneciente a la función *ID_HOP*, esta es usadas por los agentes de búsqueda. La estructura diseñada recibe el nombre de *tabla de aprendizaje de corto alcance modificada* la cual esta compuesta por la siguiente información: nombre del vecino, nombre de la clave a buscar y cantidad de pasos hacia el nodo que tenga la clave buscada y que sea el más cercano. El agente de búsqueda cuando avanza necesita hacerlo a través de la regla de transición, dentro de dicha regla el *ID_HOP* es uno de los términos considerados como uno de los criterios para seleccionar el nodo siguiente en la trayectoria. La hormiga de retroceso actualiza el valor del *ID_HOP*, según la cantidad de nodos recorridos. Rutas más largas son menos deseables que las rutas cortas. Bajo estas condiciones se muestran los resultados obtenidos en el capítulo 4.

3.4. Desarrollo de un sistema de búsqueda adaptativo

La construcción de una función de ajuste de control adaptativo del algoritmo AdaNAS, es la parte medular de este trabajo de investigación. Todos los resultados obtenidos en las etapas anteriores se incorporan en al algoritmo AdaNAS.

El ajuste de *Control de Parámetros*, ofrece la ventaja de estar monitoreando los cambios en el ambiente al mismo tiempo que considera el estado actual del algoritmo. El control *adaptativo* se realiza cuando existe alguna forma de retroalimentación del pasado que determina un cambio en sentido y magnitud del parámetro [77]. Dentro de los cambios realizados a el algoritmo NAS para diseñar la versión adaptativa llamada AdaNAS se encuentran:

a. Diseño de un modelo adaptativo basado en agentes

Se usará un modelo discreto de adaptación basado sobre la propuesta de Holland [101], en este se asume que el sistema realiza las acciones en pasos discretos $t = 1, 2, 3, \ldots$, esta suposición aplica prácticamente a todos los sistemas computacionales. El sistema es subdividido en cuatro partes: la estructura *A* para adaptar el ambiente (llamados agentes), los planes de adaptación *P*, la memoria *M* y los operadores *O*. Obteniéndose un modelo adaptativo basado en agentes para solucionar SQRP.

b. Desarrollo de un algoritmo adaptivo: AdaNAS

AdaNAS es un algoritmo metaheurístico, donde un conjunto de agentes independientes llamados hormigas cooperan independiente y esporádicamente para lograr una meta común. El algoritmo tiene dos objetivos: busca maximizar el número de recursos encontrados por las hormigas y minimizar el número de saltos dados por la hormiga. AdaNAS guía las consultas hacia nodos que tienen mejor conectividad usando la métrica estructural del grado del nodo (definido en el capítulo 1). Además el uso de la técnica Lookahead, la cual por medio de las estructuras permite conocer los repositorios de los nodos vecinos de un nodo especifico. Tablas de aprendizaje de corto y largo alcance (feromona) y parámetros de control de factores de desempeño.

c. Diseño de la función de ajuste de control adaptativo del parámetro TTL

La función de ajuste de control adaptativo del parámetro TTL, es desarrollada con información de las tablas de aprendizaje de corto alcance. A través de estas estructuras se toma la decisión de extender el tiempo de vida y se aplica una configuración voraz para los parámetros: $q = 1$, $\beta_2 = 0$ y $w_d = 0$. Haciendo que la regla de transición se modifique al cancelar algunos parámetros de la ecuación. En otras palabras la función de ajuste de control adaptativo del parámetro TTL depende solamente de la tabla de aprendizaje de corto alcance D.

Cada uno de estos puntos son detallados y documentados en el capítulo 4.

3.5 Evaluación de la Eficiencia de los Algoritmos NAS y AdaNAS

La última actividad propuesta en esta metodología es la evaluación de la eficiencia, donde se evalúa el rendimiento de los dos algoritmos NAS y AdaNAS. Con la finalidad de determinar la

contribución de la técnica de control adaptativo a la eficiencia del algoritmo AdaNAS. Además de evaluar si logra satisfacer los objetivos de SQRP, minimizando la cantidad de pasos (*hops*) y maximizando la cantidad de éxitos (*hits*).

Los dos algoritmos AdaNAS y NAS, recibirán la misma configuración e información para las distribuciones de: topología, repositorios y consultas. La variable de respuesta es el promedio de la eficiencia de los algoritmos al ejecutarlos 30 veces con la misma configuración sobre el conjunto de instancias SQRP. La eficiencia promedio de los algoritmos se mide a través de la tasa de aprendizaje *hits/hops*, que indica cuantos recursos encontró el algoritmo en cada nodo que visitó. El conjunto de instancias SQRP son las descritas en la sección 4.2.2 en la que se presenta información detallada sobre su generación. En total se utilizaron 90 instancias SQRP. Las hipótesis se verificarán en el capítulo 4, a través de los diferentes experimentos desarrollados y el análisis de los resultados obtenido.

3.5.1 Pruebas de diferencias significativas usando Wilcoxon

La estadística no paramétrica es una rama de la estadística que estudia las pruebas y modelos estadísticos cuya distribución subyacente no se ajusta a los llamados criterios paramétricos. Su distribución no puede ser definida a priori, pues son los datos observados los que la determinan. La utilización de estos métodos se hace recomendable cuando no se puede asumir que los datos se ajusten a una distribución conocida, cuando el nivel de medida empleado no sea, como mínimo, de intervalo.

Ventajas de los Métodos No Paramétricos

1. Los métodos no paramétricos pueden ser aplicados a una amplia variedad de situaciones porque ellos no tienen los requisitos rígidos de los métodos paramétricos correspondientes. En particular, los métodos no paramétricos no requieren poblaciones normalmente distribuidas.

2. Diferente a los métodos paramétricos, los métodos no paramétricos pueden frecuentemente ser aplicados a datos no numéricos, tal como el género de los que contestan una encuesta.
3. Los métodos no paramétricos usualmente involucran simples computaciones que los correspondientes en los métodos paramétricos y son por lo tanto, más fáciles para entender y aplicar.

Desventajas de los Métodos No Paramétricos

1. Los métodos no paramétricos tienden a perder información porque datos numéricos exactos son frecuentemente reducidos a una forma cualitativa.

2. Las pruebas no paramétricas no son tan eficientes como las pruebas paramétricas, de manera que con una prueba no paramétrica generalmente se necesita evidencia más fuerte (así como una muestra más grande o mayores diferencias) antes de rechazar una hipótesis nula.

Cuando los requisitos de la distribución de una población son satisfechos, las pruebas no paramétricas son generalmente menos eficientes que sus contrapartes paramétricas, pero la reducción de eficiencia puede ser compensada por un aumento en el tamaño de la muestra.

El procedimiento para aplicar la prueba de Wilcoxon con muestras correlacionadas es:

1. Primero se calcula la diferencia entre las dos muestras, es decir $D_1 – D_2$
2. Se calcula el valor absoluto de el paso anterior $|D_1 – D_2|$
3. Se ordenan los datos con relación al valor absoluto
4. Los valores en cero se eliminan
5. Se ordenan las diferencias absolutas, de menor a mayor
6. Se asigna una categoría creciente. Si hay diferencias iguales se les asignará el valor promedio de sus diferencias
7. Por último, a las calificaciones se les asigna el signo de la diferencia entre las muestras $D_1 – D_2$, y se realiza una suma de estos últimos valores.

Capítulo 4

EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se describe la experimentación realizada para validar los objetivos e hipótesis del trabajo de investigación. Los experimentos se presentan siguiendo el orden de la metodología descrita en el capítulo 3. Para cada experimento se describe el producto obtenido en el paso correspondiente de la metodología y se dan evidencias experimentales para sustentar las conclusiones emitidas.

Los experimentos empiezan con la creación de la infraestructura de referencia. Continuando con la identificación de factores de desempeño y sus relaciones. Enseguida se lleva a cabo el desarrollo de un sistema de búsqueda adaptativo. Para terminar con la evaluación de la eficiencia del algoritmo adaptativo AdaNAS.

4.1 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 7.10.
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.
f) MatLab 7.

Hardware

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

4.2 Integración de una Infraestructura de Referencia

En esta etapa se analizan redes reales y desarrolla instancias artificiales para llevar a acabo los experimentos. Así como el desarrollo de la primera versión del algoritmo NAS el cual el la base para trabajar el problema de configuración de parámetros adaptativos.

4.2.1 Caracterización de Redes Reales

Para este experimento se tomó información de subgrafos de la Internet disponibles en [71] para formar posteriormente instancias de prueba, que refleje la topología de Internet. Se analizaron los años de 1997, 2000 y 2003.

La Tabla 4.1 presenta la información de muestras de Internet usadas en el experimento. En la primera columna aparece el nombre de la instancia, en la segunda y tercera columna aparece el número de nodos y el número de aristas respectivamente, en la cuarta columna el grado promedio de la red, en la quinta y sexta columna se muestra el grado mínimo y el grado máximo respectivamente y en la última columna aparece el coeficiente de dispersión del grado global. De acuerdo con diversas investigaciones [43,55] se puede afirmar que Internet es una red de tipo scale-free, eso se reafirma con los valores calculados para las muestras de Internet.

Para la caracterización de las muestras de las muestras de Internet se usó la métrica topológica *DDC*(G), la cual fue validada en [93] como una medida que determina de manera confiable los tres modelos, redes: aleatorias, scale-free y exponenciales.

Tabla 4.1 Información de muestras de la Internet para los años 1997, 2000 y 2003

Muestras	**N**	***E***	$\langle k \rangle$	$\delta(G)$	$\Delta(G)$	**DDC(G)**
INT – 1997	3015	5,156	3.42	1	590	1.02
INT – 2000	6474	12572	3.88	1	1458	1.06
INT – 2003	192244	609066	6.33	1	1071	0.75

En la muestra de Internet que corresponde al año 1997 de los 3015 subgrafos, el 72% corresponde con las características de las redes scale-free, el 23% corresponde con las características de las redes aleatorias y el 5 % del total corresponde con redes de tipo exponencial. Estos resultados se presentan en la Figura 4.1.

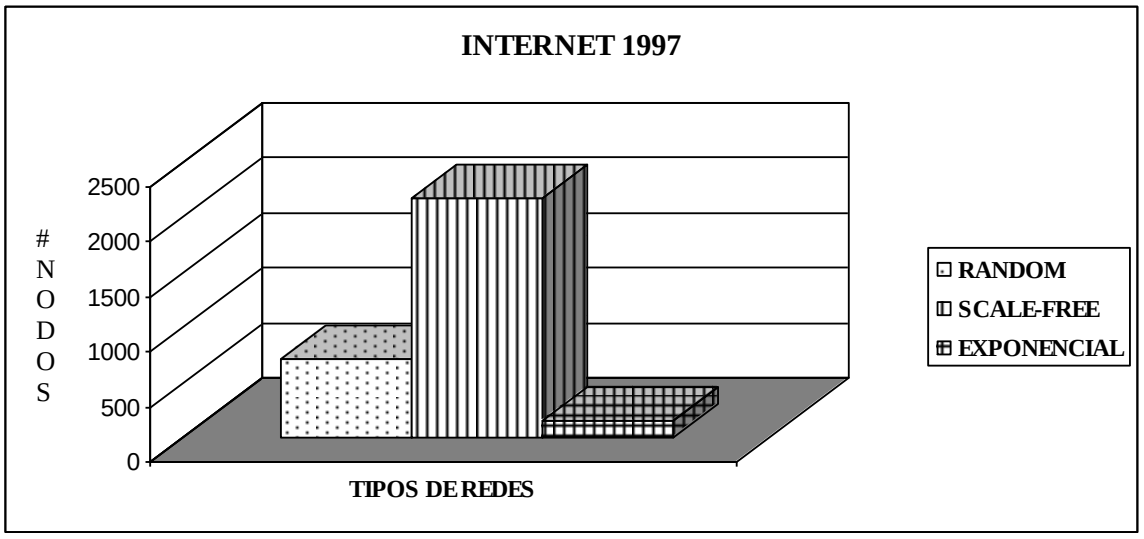


Figura 4.1 Clasificación de los subgrafos de muestra INT – 1997 con la métrica topológica local DDC(*i*) .

En la Figura 4.2 se presenta la distribución del grado en escala doble logarítmica para el conjunto de datos de INT-1997, la cual confirma que la distribución del grado es power-law (red scale-free).

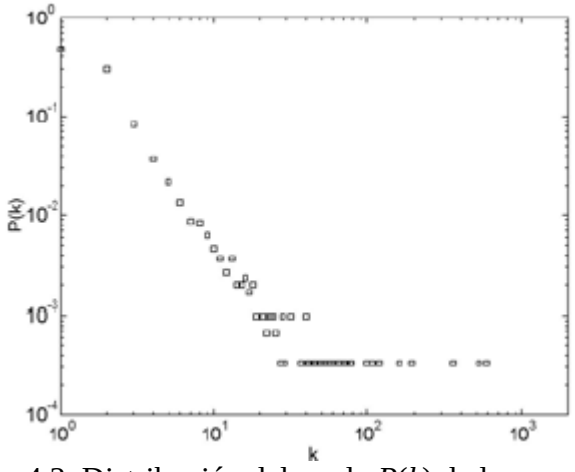


Figura 4.2. Distribución del grado *P*(*k*) de la muestra INT – 1997

En la muestra de Internet que corresponde al año 2000, de los 6474 subgrafos, 4426 que corresponden al 68% de los subgrafos presentan una distribución del grado de tipo power-law (red scale-free), 1134 subgrafos que representan el 18% de los existentes tienen una distribución del grado aleatoria y 914 subgrafos que representan el 14 % del total corresponden con una distribución del grado de tipo exponencial. Estos resultados se muestran en la Figura 4.3.

En la Figura 4.4 se presenta la distribución del grado en escala doble logarítmica para el conjunto de datos de INT-2000. Se puede apreciar que ha habido una variación en el número de nodos y lados, pero la distribución del grado se mantiene en power-law (red scale-free).

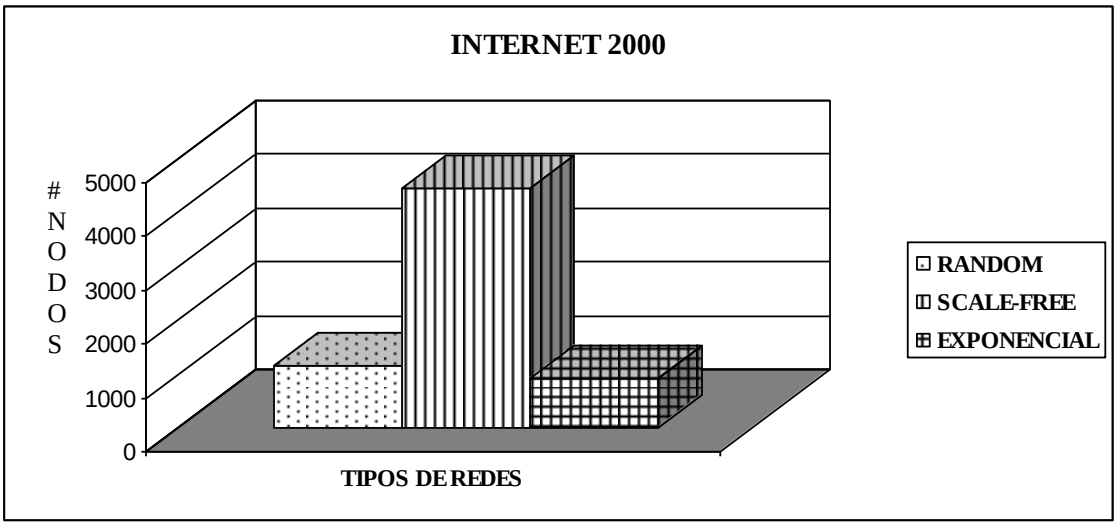


Figura 4.3. Clasificación de los subgrafos de muestra INT – 2000 con la métrica topológica local DDC(*i*).

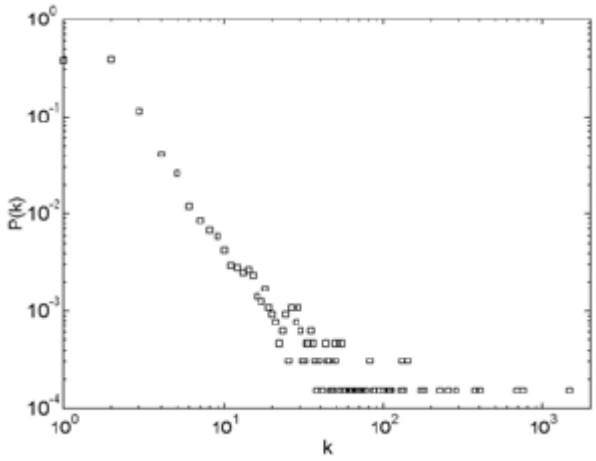


Figura 4.4 Distribución del grado $P(k)$ de la muestra INT – 2000

En la muestra de Internet que corresponde al año 2003, de los 192244 subgrafos 139150, que corresponden al 73% de los subgrafos su distribución del grado de tipo power-law (red scale-free), 29717 subgrafos que representan el 15% de los existentes corresponden con una distribución del grado aleatoria y 23377 subgrafos que representan el 12 % del total corresponden con una distribución del grado exponencial. Estos resultados se muestran en la Figura 4.5.

En la Figura 4.6 se presenta la distribución del grado en escala doble logarítmica para el conjunto de datos de INT-2003. También se observa una variación en el número de nodos y lados, pero la red mantiene sus características promedio como tipo scale-free.

Los resultados de los experimentos han mostrado que la caracterización de redes complejas permite analizar la topología de redes naturales y artificiales. El análisis confirma que la Internet, y aunque tiene un crecimiento dinámico, en términos generales se ha mantenido con una distribución del grado power-law es decir es una red de tipo scale-free.

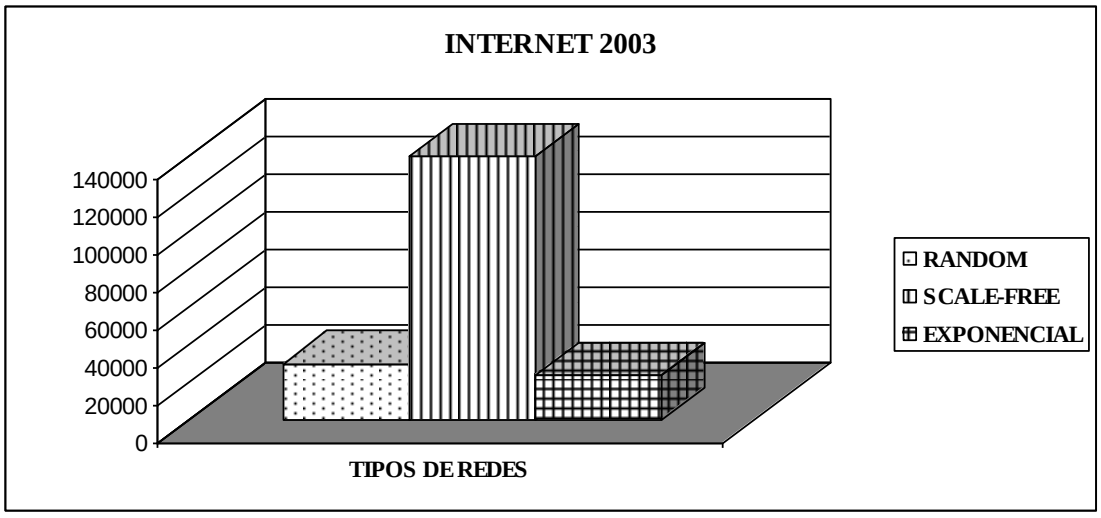


Figura 4.5 Clasificación de los subgrafos de la muestra INT – 2003 con la métrica topológica local DDC(*i*).

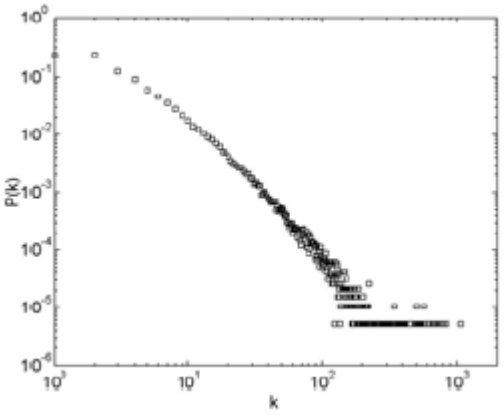


Figura 4.6 Distribución del grado $P(k)$ de la instancia INT – 2003

En Int-1997 y Int-2000 un incremento en el número de nodos y lados motiva un incremento en el grado máximo $\Delta(G)$. Sin embargo, debido a que Int-2003 contiene un número exponencial de nodos y lados, hay una reducción en $\Delta(G)$. Esta observación es muy importante ya que se observa que la adición de nuevos nodos y lados a una red con distribución del grado power-law como la Internet no ha incrementado el grado máximo y preserva sus características de la distribución del grado. Sin embargo, Int-2003 hace evidente que el crecimiento dinámico de la Internet es afectado por los factores tecnológicos que evitan el crecimiento del grado máximo y motivan la reducción del número de nodos de poco grado. Estas características deben ser tomadas en cuenta para generar redes artificiales con distribución del grado power.law.

En suma los resultados presentan indicios de una mejor distribución de las conexiones. Además se observa que en las tres muestras reales analizadas co-existen subgrafos no-homogéneos, es decir, se demuestra que las redes que se abordan tienen subestructuras con patrones distintos con propiedades diferentes entre sí.

Entonces es válido asegurar que es posible que un agente que se desplace por la red navegará en subredes con características topológicas no uniformes de un punto a otro punto de la red, por lo que es deseable que el agente aproveche las condiciones locales actuales para maximizar su desempeño. Otro punto importante que presentan las muestras de Internet es que la topología se ha mantenido a través del tiempo con una red de tipo scale-free, siendo esto un factor muy importante para la investigación.

4.2.2 Creación de Generadores de Redes

En esta etapa se describe la generación de instancias artificiales para topología, repositorios y consultas usando dos modelos de generación de redes complejas.

a. Redes Aleatorias

En la Tabla 4.2 se muestra la información para redes creadas con el modelo de Erdös y Rényi. Con n = {50, 50, 200, 200} y sus correspondientes probabilidades P ={0.266152, 0.867890, 0.602100, 0.135000}. En la tercera columna se muestra el grado mínimo $\delta(G)$, en la cuarta columna el grado máximo de la red $\Delta(G)$, en la quinta columna se muestra el grado promedio $\langle k \rangle$ y sexta columna se muestra la desviación estándar σ y en la séptima columna se muestra el coeficiente de agrupamiento global $C(G)$, el cual mide la tendencia a formar grupos [9,35] y en la última columna se muestra el DDC(G). Los resultados de la Tabla 4.3 muestran que las redes tienen su grado mínimo y máximo cercano al grado promedio. Además la desviación estándar indica que los grados de los nodos no están alejados del grado promedio, y el coeficiente de agrupamiento es aproximadamente igual a la probabilidad de en lace P. Además que al aplicarles el DDC(G) esta dentro del rango de valores que corresponde al tipo de red aleatoria.

Tabla 4.2. Información acerca de las redes generadas con el Modelo Erdös-Rényi.

n	P	$\delta(G)$	$\Delta(G)$	$\langle k \rangle$	σ	$C(G)$	DDC(G)
50	0.266152	14	20	13	3.32445	0.27417	0.0234
50	0.867890	37	48	43	2.0629	0.887626	0.0056
200	0.602100	98	139	118	6.72873	0.593355	0.0080
200	0.135000	13	38	26	4.60808	0.135805	0.0280

En la literatura se encontró que las consultas siguen una distribución de probabilidad aleatoria. Debido a que los usuarios de la red pueden lanzar cualquier tipo de consulta en cualquier momento [5, 14,92].

b. Redes Scale-free

En la Tabla 4.3 muestra los datos para las redes con m_0 nodos iniciales, la segunda columna corresponde con m aristas de enlace. En la tercera columna se muestra t que es el paso o tiempo que se esta ejecutando, en la cuarta columna se encuentra el grado mínimo $\delta(G)$, en la quinta columna se muestra el grado máximo de la red $\Delta(G)$ y en la sexta columna se muestra el grado

promedio $\langle k \rangle$. En la séptima columna se muestra la desviación estándar σ y en la última columna se muestra el DDC(G). Los resultados de la Tabla 4.3 muestran que el grado promedio es muy pequeño en comparación con el grado máximo, existiendo una gran diferencia entre el grado máximo y el grado mínimo. Debido a que la desviación estándar es un poco mayor con respecto al grado promedio se puede confirmar que los nodos altamente conectados son pocos en relación con los nodos que no están muy conectados. Además que al aplicarles el DDC(G) esta dentro del rango de valores que corresponde al tipo de red scale-free.

Tabla 4.3. Información acerca de las redes generadas con el Modelo de Barabási.

m_0	m	t	$\delta(G)$	$\Delta(G)$	$\langle k \rangle$	σ	DDC(G)
4	3	100	3	33	5	4.98351	0.6034
5	3	1000	3	124	5	8.00114	0.8044
4	3	1000	3	82	5	6.84551	0.5943
5	2	1500	2	120	3	5.33489	0.8133

Al consultar en la literatura se ha encontrado que la distribución del grado para generar repositorios y consultas es las misma que se ha explicado para redes en esta sección [15,95,97]. pero con diferente significado.

A continuación se presentan las condiciones que son usadas para generar una instancia para SQRP, y estas son usadas para toda la experimentación:

La *topología* (*T*) fue generada con el modelo de Barabási et al. [45], para crear redes no uniformes con una distribución del grado power-law (red scale-free). Esta distribución es formada por un conjunto de nodos con un alto grado de conexiones y el resto de los nodos tienen pocas conexiones. Todas las redes generadas tienen 1024 nodos y sus enlaces son bidireccionales. El número de nodos fue seleccionado basado en recomendaciones de la literatura [5,70].

En las redes P2P, cada punto administra un *repositorio local* (*R*) de recursos y ofrece sus recursos a otros puntos de la red. Estos repositorios fueron generados usando "tópicos" obtenidos de ACM Computing Classification System taxonomy (ACMCCS). Esta base de datos contiene un total de 910 tópicos distintos. La distribución del contenido de los repositorios sigue una distribución del grado power-law, es decir, unos pocos nodos contienen muchos tópicos en sus repositorios y el resto de los nodos contiene pocos tópicos.

Para la generación de las *consultas* (*Q*), a cada nodo le fue asignado una lista de posibles tópicos a buscar. Esta lista esta limitada por la cantidad total de tópicos de ACMCCS. Durante cada paso del experimento, cada nodo tiene una probabilidad de 0.1 de lanzar una consulta, seleccionando un tópico aleatoriamente dentro de una lista de posibles tópicos del nodo. La distribución del grado de las consultas *Q* es aleatoria y determina que tan a menudo se repite una consulta en la red. Cuando la distribución del grado es aleatoria cada consulta es duplicada 100 veces en promedio.

La topología y los repositorios son estáticos durante la ejecución del experimento, mientras que las consultas son lanzadas aleatoriamente durante la simulación. Cada simulación se ejecuta

durante 10,000 unidades de tiempo (consultas). El promedio del rendimiento fue obtenido mediante tres medidas las cuales fueron obtenidas cada 100 unidades de tiempo.

Cuando se generan instancias artificiales es necesario que se generen con el siguiente orden: primero se tiene que generar la red, posteriormente el repositorio de datos y finalmente el archivo de consultas; de esta forma se asegura que siempre se consultará algo que existe en algún repositorio de datos. Lo que no se puede asegurar es cuantas replicas existen de cada uno de los datos en la red. Las redes generadas a través de los modelos de [45,98], se ajustan con las características de los modelos reales. Estos modelos sirven para generar instancias de SQRP con distribución especifica en topología, repositorios y consultas. Las cuales son usadas en los experimentos.

4.2.3 Desarrollo de Algoritmos de Referencia: NAS y RW

En este punto se describirá el desarrollo de los algoritmos de referencia NAS y RW. Estos algoritmos son construidos para resolver SQRP, y para probarlos se usarán las instancias construidas en el la sección 4.2.2.

a. Algoritmo Neighboring-Ant Search (*NAS*)

El algoritmo NAS descrito en la sección 3.2.3 es un algoritmo metaheurístico basado en los algoritmos ACS [24] y SemAnt [5]. 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.

El algoritmo tiene dos objetivos principales: guiar las consultas hacia nodos mejor conectados 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.

NAS está hibridizado con estrategias del ambiente local de aprendizaje, caracterización y exploración. Dos reglas de aprendizaje (LR, por sus siglas en inglés) son usadas para aprender del rendimiento pasado, esas reglas son modificadas por tres Funciones de Aprendizaje (LF, por sus siglas en inglés). El DDC es usada como una métrica topológica local para la caracterización estructural. Una adaptación del bien conocido método de exploración de adelanto (one-step Lookahead) es usado para explorar el ambiente cercano. Estas estrategias locales proveen a NAS la capacidad de mejorar el rendimiento de la búsqueda distribuida.

b. Arquitectura del Sistema Multiagente para NAS

La arquitectura general del sistema se muestra en la Figura 4.7 y comprende dos elementos principales: *i*) El *ambiente E* es una red compleja con una distribución de probabilidad para la topología de la red (*T*), entendida como un conjunto de nodos enlazados con información local acerca de sus vecinos inmediatos. *ii*) Los *agentes* pueden ser de cuatro tipos dependiendo del rol

que desempeñan: de consulta, búsqueda, recuperación y actualización. En el algoritmo NAS cada agente lleva una consulta (*Q*), y puede ser representado por un nodo o una hormiga.

- El *agente de consulta* $\{e_1\}$ es representado por el nodo y su rol es lanzar consultas y crear *agentes de búsqueda*. Cada nodo tiene un repositorio propio *R* donde se almacenan los documentos a compartir y una tabla de feromonas que representa el aprendizaje del sistema. En cada unidad tiempo este agente realiza una evaporación local de la tabla de feromonas a través de la Ecuación (4.9), con el objetivo de favorecer la fase de exploración del algoritmo.

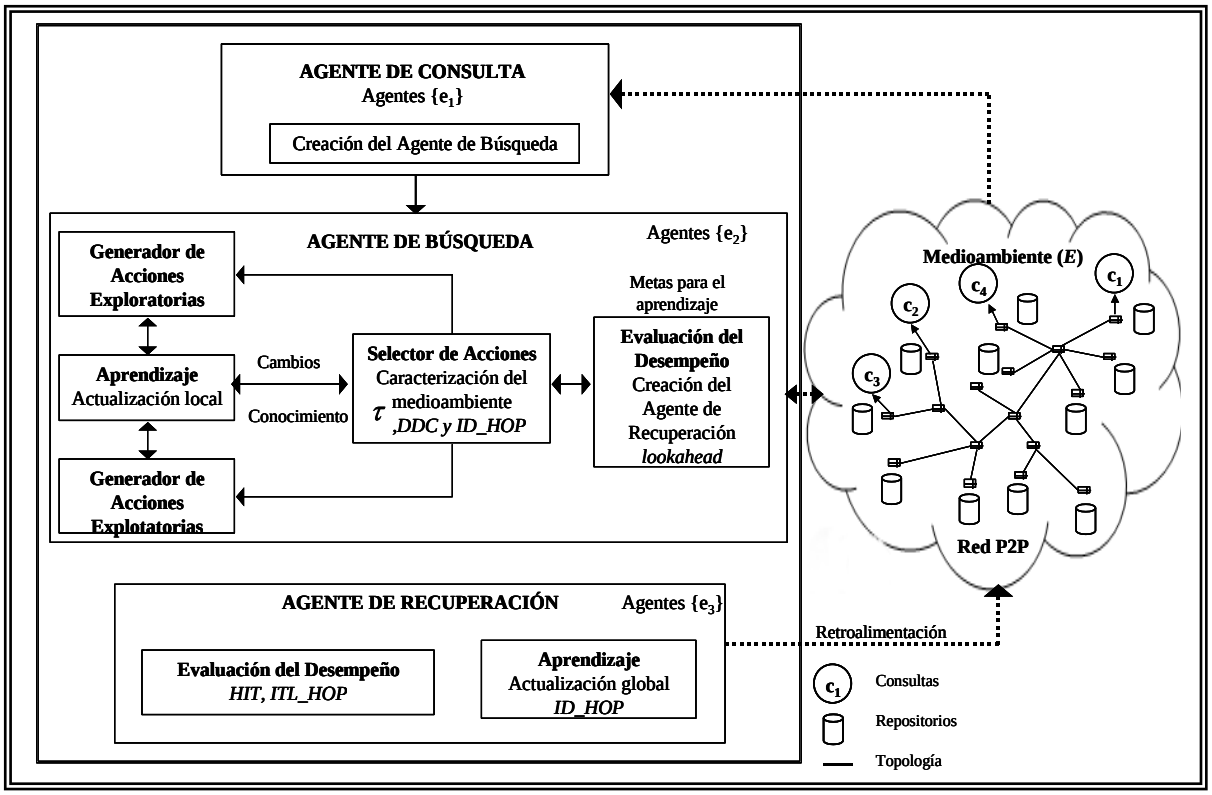


Figura 4.7. Arquitectura General del Sistema Multiagente Propuesto.

- El *agente de búsqueda* $\{e_2\}$ es representado por una hormiga que nace en el nodo que lanza la consulta. Su rol es moverse a través de la red siguiendo un conjunto de reglas. Estos agentes operan a través del selector de acciones que selecciona entre movimientos de exploración o explotación a través de las Ecuaciones (4.7) y (4.8). Para evaluar el rendimiento, la hormiga va registrando la ruta que ha sido seleccionada, y cada vez que encuentra un recurso, crea un agente de recuperación. La estrategia Lookahead verifica la existencia de los recursos en el nodo actual y en los vecinos. El tiempo de vida de un agente de búsqueda es fijado como *maxTTL*, pero el agente también puede terminar la operación de búsqueda al considerar que se han encontrado el máximo de recursos esperados establecidos en *maxResults*. Cuando el agente termina la operación de búsqueda genera un agente de actualización que mantiene actualizadas las tablas de feromona de los nodos transitados. Todos los módulos tienen parámetros de control que deben ser configurados correctamente para asegurar el buen rendimiento del sistema.

- El *agente de recuperación* $\{e_3\}$ es creado cuando se encuentra un conjunto de recursos en un nodo. Estos agentes son responsables de devolver los recursos encontrados al nodo que los solicito. Al realizar su tarea este agente recorre en sentido inverso la trayectoria generada por el agente de búsqueda hasta el momento, y actualiza la métrica ID_HOP en cada nodo visitado a través de la Ecuación (4.9).

c. Funciones de Aprendizaje Histórico Propuestas

El algoritmo clásico de ACS esta formado de reglas de selección y actualización que permiten la convergencia del sistema hacia mejores resultados. Las modificaciones de esas reglas fueron hechas con el objetivo de adaptar el algoritmo ACS para dar solución a SQRP. Dichas modificaciones se hicieron agregando métricas locales tales como: el Coeficiente de Dispersión del grado (DDC), la importancia de los éxitos (*HIT*), la importancia del tiempo de vida (*ITL_HOP*) y la distancia hacia un recurso buscado (*ID_HOP*)); también se incluyó el método Lookahead. Estos cambios favorecen el desempeño del sistema multiagente en términos de la función objetivo del problema. En esta sección se describen los ajustes realizados.

Función de Importancia de los Éxitos (*HIT*, por sus siglas en inglés)

La función *HIT*, mostrada en la Ecuación (4.4), califica el rendimiento del *agente de búsqueda k* (que es una hormiga que transporta una consulta), basado sobre los resultados ($result_k$) encontrados y los recursos esperados (*maxResults*):

$$HIT = \frac{results_k}{maxResults}, \tag{4.4}$$

donde $results_k$ representa la cantidad de recursos encontrados por la hormiga *k*, y *maxResults* es la cantidad máxima de recursos esperados. El valor de la función de *HIT* es aplicado en la función de actualización global, Ecuación (4.10), por el agente de actualización con el propósito de guiar las consultas hacia rutas que provean la mayor cantidad de recursos encontrados.

Función de Importancia del Tiempo de Vida (*ITL_HOP*, por sus siglas en inglés)

La función *ITL_HOP*, mostrada en la Ecuación (4.5), califica el rendimiento del agente de búsqueda *k* basándose en el tiempo de vida usado hasta encontrar un recurso (TTL_k):

$$ITL\_HOP = \frac{TTL_{max}}{2 \cdot TTL_k}, \tag{4.5}$$

donde TTL_k es el tiempo de vida parcial de la hormiga *k* hasta el momento. Estas mediciones de tiempo están dadas en términos del número de saltos y corresponden respectivamente con los pasos dados y los pasos máximos permitidos a cada hormiga. El resultado es aplicado en la regla de actualización global, Ecuación (4.10), con el objetivo de disminuir el tiempo de vida necesario para encontrar un conjunto de recursos.

Función de la Importancia de la Distancia (*ID_HOP*, por sus siglas en inglés)

Para el agente *k* la función *ID_HOP*, mostrada en la Ecuación (4.6), califica a cada nodo *s*, el cual es un vecino del nodo actual *r*, dependiendo de la distancia hacia el recurso *t* encontrado en consultas previas y que es el recurso mas cercano a partir del nodo s*:*

$$ID\_HOP_{r,s,t} = \left(\frac{h_k}{h_{r,s,t}} \right)^{-1} \quad \forall r, s, t \in ruta\ k. \tag{4.6}$$

donde $h_{r,s,t}$ es el número de saltos para llegar al recurso *t* desde el nodo *r*, transitando por el nodo *s*. Una vez que una hormiga *k* genera una ruta hasta un recurso *t*, la función $ID\_HOP_{r,s,t}$ se aplica a cada nodo perteneciente a la ruta para incrementar su importancia en términos de la distancia hacia el recurso encontrado con respecto a la longitud de la ruta. Esta función es usada en la regla de selección implementada en la Ecuación (4.7).

d. Reglas Clásicas de Aprendizaje Modificadas para NAS

Esta sección describe las modificaciones hechas para las dos reglas de aprendizaje clásicas, originalmente propuestas por Dorigo [24] en el algoritmo ACS. La primera regla es llamada de transición de estado; con esta regla el siguiente nodo a ser visitado es seleccionado. La regla de transición usa dos estrategias: explotación y exploración. La segunda regla es llamada de actualización; con esta regla la hormiga actualiza los nodos en la ruta viajada. La regla de actualización usa dos estrategias: actualización local y global de la feromona. Estas estrategias son usadas para retroalimentar el sistema sobre rutas exitosas.

La regla de transición de estado modificada para NAS es formada en las Ecuaciones (4.7) y (4.8). La regla de selección es:

$$s = \begin{cases} \arg\max_{L_n \forall n \in \{0,|L|\}} \left\{ \left[\tau_{r,L_n,t} \right] \cdot \left[DDC_{L_n} + ID\_HOP_{r,L_n,t} \right]^{\beta} \right\}, & \text{if } q \le q_0 \text{ (explotación)} \\ S, & \text{de otra forma (exploración),} \end{cases} \tag{4.7}$$

$$S = f(p_{r,u,t}), \quad p_{r,u,t} = \frac{\left[\tau_{r,u,t} \right] \cdot \left[DDC_u + ID\_HOP_{r,u,t} \right]^{\beta}}{\sum_{\forall i \in \Gamma(r) \wedge i \notin V_k} \left[\tau_{r,i,t} \right] \cdot \left[DDC_i + ID\_HOP_{r,i,t} \right]^{\beta}} \tag{4.8}$$

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*, τ es la tabla de feromonas, *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*, L_n es la lista de nodos disponibles, *q* es un número aleatorio entre [0,1] y q_0 determina la importancia relativa de la exploración contra la explotación. En caso que $q \le q_0$, la estrategia de explotación es seleccionada: ésta selecciona un nodo que provee una gran cantidad de feromona y una mejor

conectividad con un número pequeño de saltos hacia un recurso. De otra forma la estrategia de exploración es seleccionada a través de la Ecuación (4.8).

Donde *S* selecciona un nodo aplicando una función de selección pseudo-aleatoria *f* basada sobre la bien conocida técnica de la ruleta [80] para favorecer a los nodos con una gran cantidad de feromona, alta conectividad y una distancia corta hacia el recurso requerido. Esta estrategia de exploración estimula a las hormigas a buscar nuevas rutas. Cabe resaltar que las variables pseudo-aleatorias son modificadas por las funciones de aprendizaje *DDC* e *ID_HOP*.

Las reglas de actualización de ACS están compuestas de actualización local y global. En consecuencia la regla de actualización modificada en este trabajo es dada en las Ecuaciones (4.9) y (4.10). La regla de actualización local de NAS esta formulada por:

$$\tau_{r,s,t} \leftarrow (1-\rho)\cdot\tau_{r,s,t} + \rho\cdot\tau_0, \qquad (4.9)$$

donde *r* es el nodo actual en el cual la hormiga *k* esta localizada, *s* es el nodo vecino actual al cual la hormiga se va a mover, *t* es el recurso buscado, τ es la tabla de feromonas usadas por la hormiga para hacer la actualización local, τ_0 es el valor de inicialización de la tabla de feromonas y ρ es el factor de evaporación local de la feromona. Cada vez que una hormiga decide moverse hacia un nodo mediante la regla de selección, algo de feromona es depositado en cada nodo que ha sido visitado para establecer una ruta hacia los nodos mas frecuentemente visitados. Por otro lado, la regla de actualización global modificada, dada por la Ecuación (4.7), es calculada cada vez que un recurso es encontrado y es aplicada en cada uno de los nodos que pertenecen a la ruta a través de la cual se descubrieron los recursos :

$$\tau_{r,s,t} \leftarrow (1-\alpha)\cdot\tau_{r,s,t} + \alpha\cdot\left[w\,HIT + (1-w)ITL\_HOP\right] \quad \forall r,s,t \in ruta\ k \qquad (4.10)$$

donde α es el factor de evaporación global de la feromona, *w* es el factor de peso que controla la importancia relativa entre los recursos encontrados (*HIT*) y el tiempo de vida de la hormiga (*ITL_HOP*). La cantidad de feromona depositada depende de la calidad de la solución obtenida, la cantidad de recursos encontrados y la importancia del tiempo de vida de la hormiga.

e. Estrategias de Caracterización Estructural Local Incorporadas a NAS: Lookahead y DDC

Un nodo *i* es vecino del nodo *j* si los dos nodos *i* y *j* están conectados por un lado (*i, j*) en el grafo del modelo del sistema. El conjunto de todos los vecinos del nodo *i* es denotado por $\Gamma(i)$. En un grafo simple sin dirección, que es, un grafo en el cual los lados son considerados como canales de comunicación bidireccional y cada par de nodos puede ser conectado por al menos un lado, el *grado* de un nodo es el número de vecinos que tiene.

Una estrategia basada en información local es la bien conocida estrategia de exploración *Lookahead* de nivel 1. Dicha estrategia es empleada en algoritmos para examinar los recursos de los vecinos hacia un cierto nivel, antes de decidir como proceder con la búsqueda. En este trabajo, nosotros asumimos que cada nodo conoce cuales son los recursos que los vecinos de nivel 1 [39].

Para enfocar localmente las estrategias de exploración, se uso la función de DDC [100]. DDC es basado en información local a través de la dispersión del grado de un nodo que mide las diferencias entre el grado de un nodo y los grados de sus vecinos. Esta métrica fue definida en el capítulo 2 y esta dada por la Ecuación (4.11):

$$DDC(i) = \frac{\sigma(i)}{\mu(i)}, \qquad (4.11)$$

f. Configuración del Experimento

El algoritmo NAS fue implementado para resolver las instancias del problema de direccionamiento de consultas semánticas SQRP. La aplicación del algoritmo NAS requiere la especificación de las instancias del problema a resolver y la definición de los parámetros de control del algoritmo. En esta implementación una instancia del problema esta compuesta por tres archivos separados: topología, repositorios y consultas. Las instancias fueron generadas con la información de la sección 4.2.2.

Las medidas de rendimiento que van a son evaluadas son:

a) *Promedio de los saltos*, definido como la cantidad promedio de enlaces viajados (o saltos) por un agente de búsqueda hasta que este muere (esto es *maxResults* = 10 o *maxTTL* = 25).
b) *Promedio de los éxitos* (HITS), definido como el promedio del número de recursos encontrado (éxitos) por cada agente de búsqueda hasta su muerte.
c) *Promedio de la eficiencia*, definido como el promedio del porcentaje de los éxitos dividido por el promedio de los saltos (éxitos/saltos).

Tabla 4.4. Configuración de los parámetros del algoritmo NAS

Parámetros	**Definiciones**
ρ = 0.07	Factor de evaporación local de la feromona
α = 0.07	Factor de evaporación global de la feromona
τ_0 = 0.009	valor de inicialización de la tabla de feromonas
ID_HOP_0 = 0.001	Valor inicial de la función de importancia de la distancia
β = 2	Importancia relativa entre el DDC e *ID_HOP*
Q_0 = 0.9	Importancia relativa entre el exploración y explotación
maxResults =10	Número máximo de resultados recuperados
maxTTL = 25	Máximo tiempo de vida del agente
w = 0.5	Importancia relativa entre HIT e ITL_HOP

Los parámetros de control del algoritmo son especificados en un archivo que contiene la configuración estática global de los parámetros del algoritmo NAS. La configuración del algoritmo NAS usada en la experimentación es mostrada en la Tabla 4.4, en la primera columna se encuentra el parámetro con su valor y en la segunda columna una breve descripción del parámetro. Los valores de los parámetros fueron establecidos de recomendaciones de la literatura [5,18,24].

g. Estudio Comparativo del Algoritmo NAS

En este experimento, el rendimiento del algoritmo NAS es comparado contra los algoritmos SemAnt [5] y Random-Walk [101]. Para la experimentación, para los tres algoritmos se usaron las especificaciones generales descritas en la sección anterior.

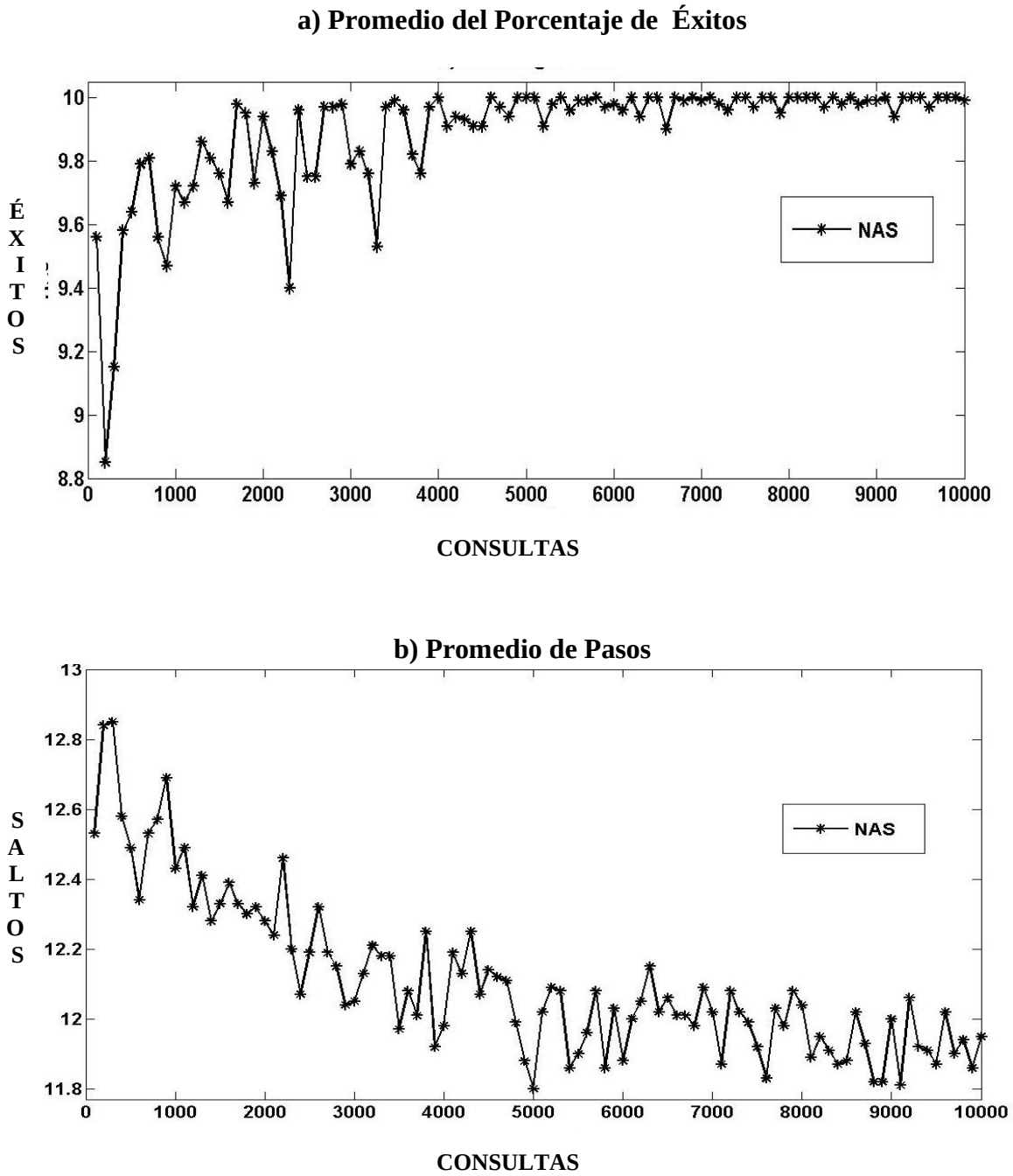


Figura 4.8. Rendimiento del algoritmo NAS, a) promedio del porcentaje de éxitos, *b*) Promedio de saltos y c) Promedio de la eficiencia

c) Promedio de la Eficiencia

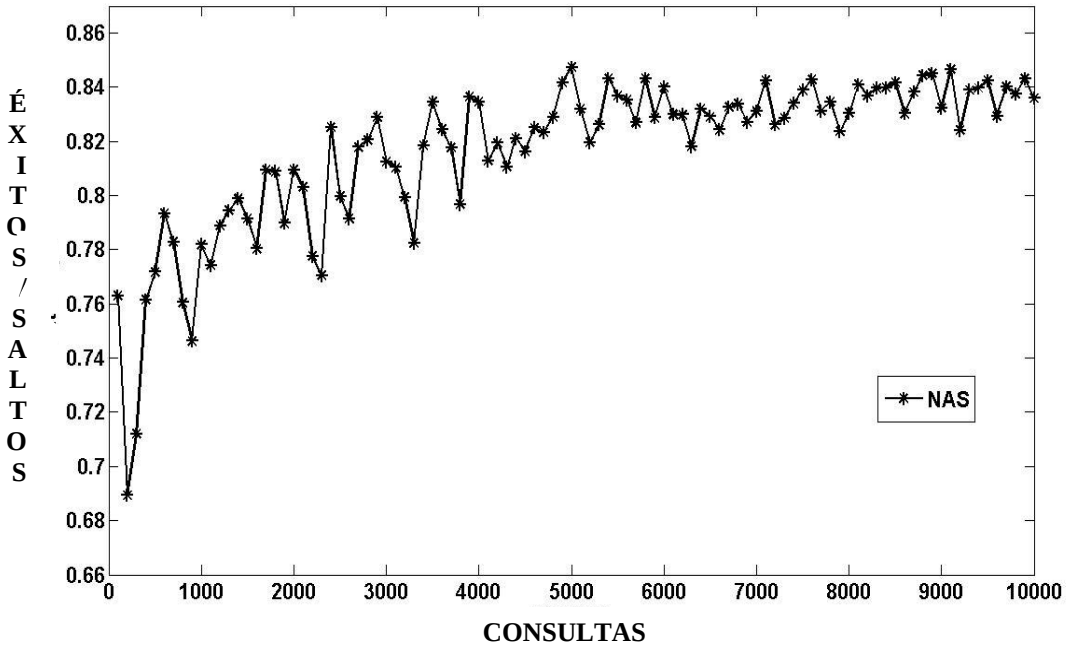


Figura 4.8. Continuación.

Para llevar a cabo la experimentación con el algoritmo SemAnt bajo las mismas condiciones, se cambió el parámetro número de enlaces. Para SemAnt, este parámetro va desde un rango de 4,000 a 10,000 conexiones, obteniéndose resultados significativamente iguales de manera que no afecta al rendimiento del algoritmo tomar un valor intermedio. Por lo tanto, en NAS este parámetro fue fijado a 7,000 enlaces.

Los valores experimentales para el algoritmo SemAnt fueron obtenidos de [5]. Para el parámetro promedio del porcentaje de éxitos (*average hit)* el algoritmo SemAnt muestra que sus valores van desde 0.8 a 2.1 éxitos por consulta. Sin embargo, para el algoritmo NAS, como se muestra en la Figura 4.8(a), Los valores para el parámetro promedio del porcentaje de éxitos (*average hit*) van desde 9.5 a 10 éxitos por consulta.

Por otro lado, para el promedio de la cantidad de saltos (*average hop*) para el algoritmo SemAnt empieza con 23 saltos y baja a 16 saltos por consulta durante la ejecución del algoritmo. Mientras que el algoritmo NAS, como se muestra en la Figura 4.8(b), empieza en un promedio de saltos de 12.5 y disminuye hasta 12 saltos por consulta.

Finalmente, el promedio de la eficiencia (*average efficiency*) para el algoritmo SemAnt mejora de 0.034 a 0.13 éxitos por saltos, Mientras tanto para el algoritmo NAS, como se muestra en la Figura 4.8(c), éste incrementa de un valor inicial de 0.76 a 0.84 éxitos por saltos.

Para el experimento con el algoritmo Random-Walk (RW), se usaron las especificaciones generales descritas en la sección 3.2.3. Los resultados de los experimentos se muestra en la Figura 4.9(a), para el promedio del porcentaje de éxitos (*average hit*) de ambos algoritmos RW y

NAS. El comportamiento cambia ligeramente sobre el tiempo. Esto es, el promedio del porcentaje de los éxitos en RW varia entre 1 a 0.5 éxitos por consulta, mientras en NAS, el promedio del porcentaje de los éxitos varia entre 9.6 y 10 éxitos por consulta.

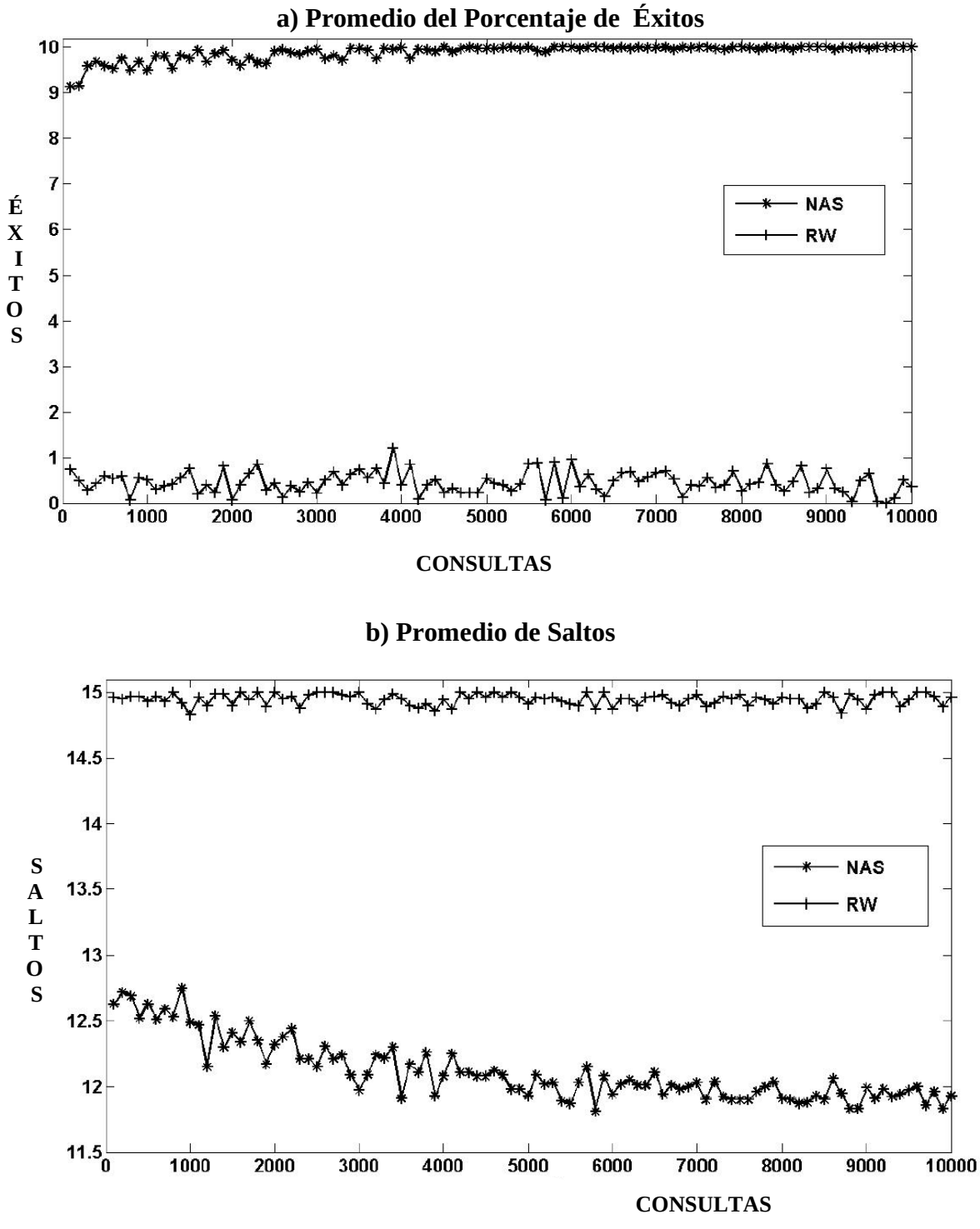


Figura 4.9. Comparación del algoritmo NAS contra el algoritmo Random-Walk. a) Promedio del porcentaje de éxitos , *b*) Promedio de saltos y *c*) Promedio de la eficiencia.

c) Promedio de Eficiencia

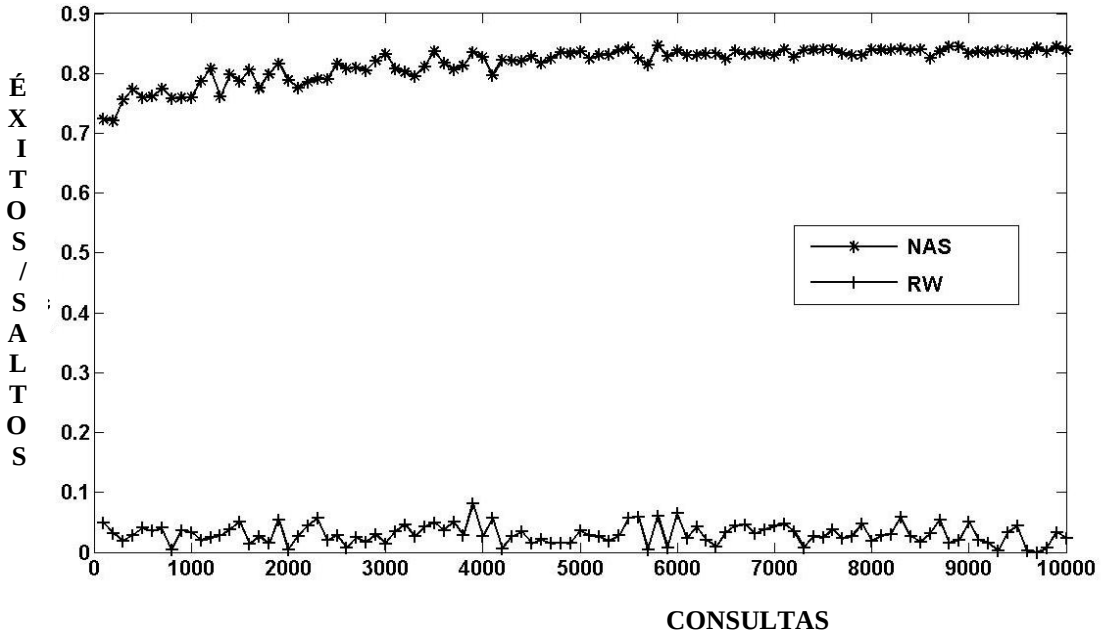


Figura 4.9 Continuación.

Similarmente, la Figura 4.9(b) muestra el promedio de la cantidad de saltos (*average hop*) para ambos algoritmos RW y NAS. El comportamiento del algoritmo RW no evoluciona; el promedio de la cantidad de saltos se mantiene alrededor de 15 saltos por consulta, desde el inicio hasta el fin. Sin embargo, en NAS el comportamiento evoluciona; empezando en 12.5 y baja a un promedio de 12 saltos.

Finalmente, la Figura 4.9(c) muestra el promedio de la eficiencia (*average efficiency*) para ambos algoritmos RW y NAS. En el algoritmo RW, el comportamiento no evoluciona; ya que el promedio de la eficiencia se mantiene al rededor de 0.05 éxitos por salto. Para NAS, el comportamiento sí evoluciona; ya que el promedio de la eficiencia se incrementa de 0.76 a 0.84 éxitos por salto.

Las observaciones anteriores confirman que el algoritmo NAS supera a ambos algoritmos SemAnt y Random-Walk. Específicamente, en la eficiencia final el algoritmo NAS muestra una eficiencia que es seis veces mejor cuando es comparado con el algoritmo SemAnt, y una eficiencia que es diez y siete veces mejor al compararlo con el algoritmo Random-Walk. Estás observaciones demuestran una mejora significante en el rendimiento de la búsqueda en presencia de la distribución scale-free, cuando las reglas de aprendizaje, el DDC y el Lookahead están activos. Esto implica que el algoritmo NAS supera esos métodos que no han incorporado información del medio ambiente local.

h. Experimentación para el Análisis de las Estrategias del Medio Ambiente Local del Algoritmo NAS

En este experimento, el rendimiento del algoritmo NAS es analizado experimentalmente para determinar la contribución de cada una de las tres estrategias locales usadas - LR modificado, DDC y Lookahead - para incrementar el rendimiento del algoritmo NAS.

Para evaluar la contribución de las estrategias relacionadas con las reglas de aprendizaje modificadas, Lookahead y DDC fueron eliminados. La contribución del DDC fue evaluado eliminando solamente Lookahead; LR modificado se mantuvo porque el DDC esta incluido en él. Similarmente, Lookahead fue evaluado sin LR modificado y DDC. Para el experimento con el algoritmo NAS se usó las especificaciones generales descritas en la sección 3.2.2, con la excepción de tres parámetros: *maxResult* = 5, *maxTTL* = 10, y q_0 = 0.90.

La Figura 4.10(a) muestra el rendimiento promedio del porcentaje de éxitos durante la obtención de un conjunto de consultas con NAS y tres diferentes configuraciones de NAS: LR modificado, DDC y Lookahead. Para el LR modificado y el DDC, los algoritmos arrancan aproximadamente en 0.5 éxitos por consulta; al final, el promedio del porcentaje de éxitos se incrementa a 2 éxitos por consulta. Para Lookahead y NAS el promedio del porcentaje de éxitos arranca en 4.3 y después de 1000 consultas el comportamiento cambia; para Lookahead el promedio de porcentaje de éxitos termina en 4 y para NAS termina en 5.

a) Promedio del Porcentaje de Éxitos

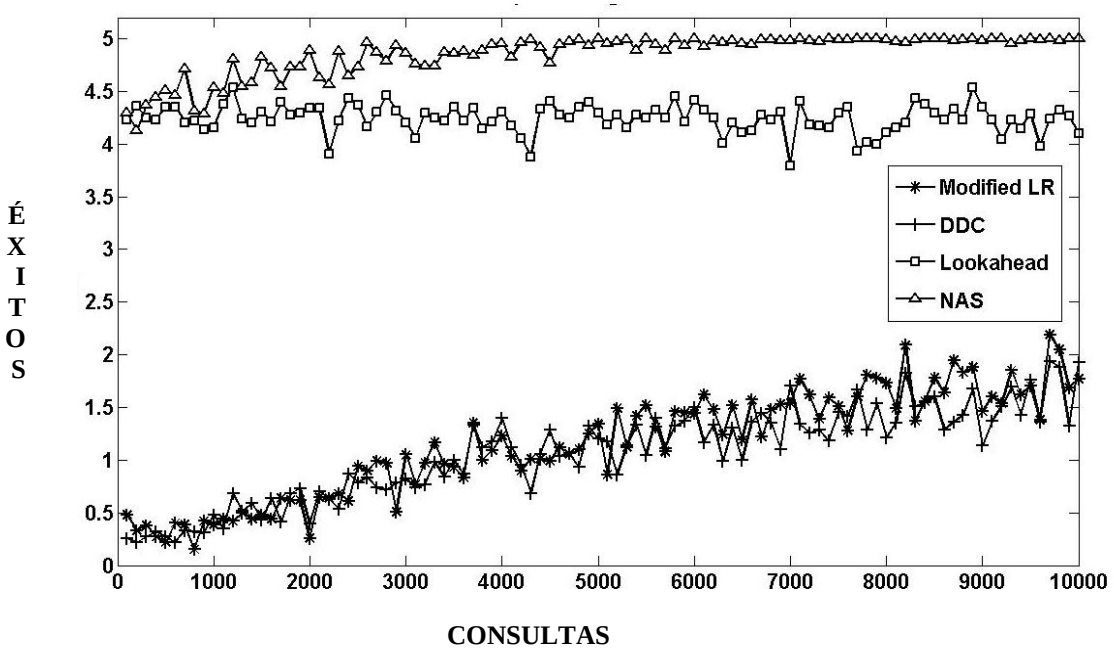
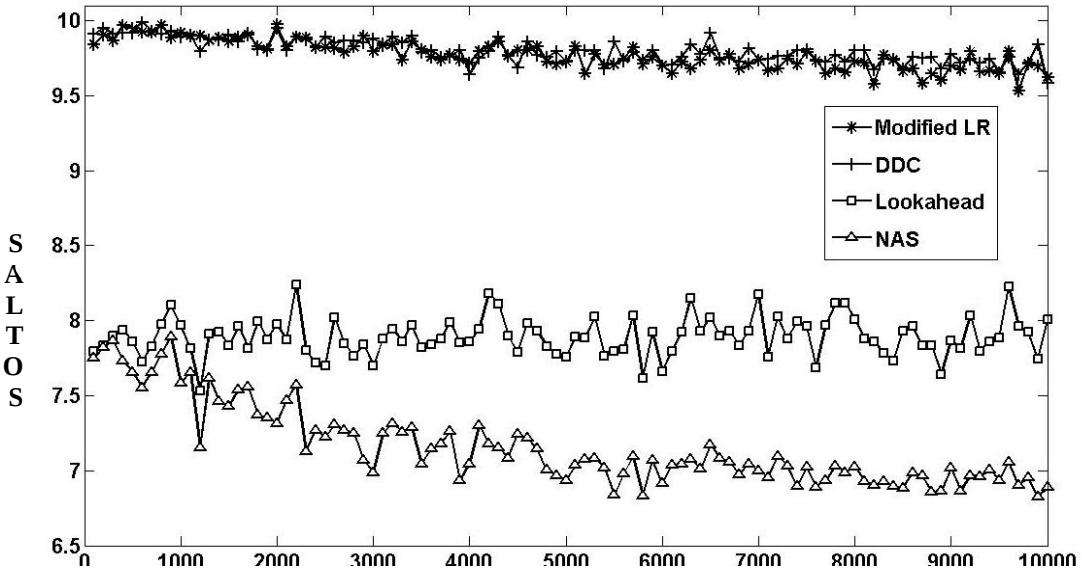


Figura 4.10. Comparación de las estrategias del medio ambiente local de NAS. a) promedio del porcentaje de éxitos, b) Promedio de saltos y c) Promedio de la eficiencia.

a) Promedio de Saltos



c) Promedio de la Eficiencia

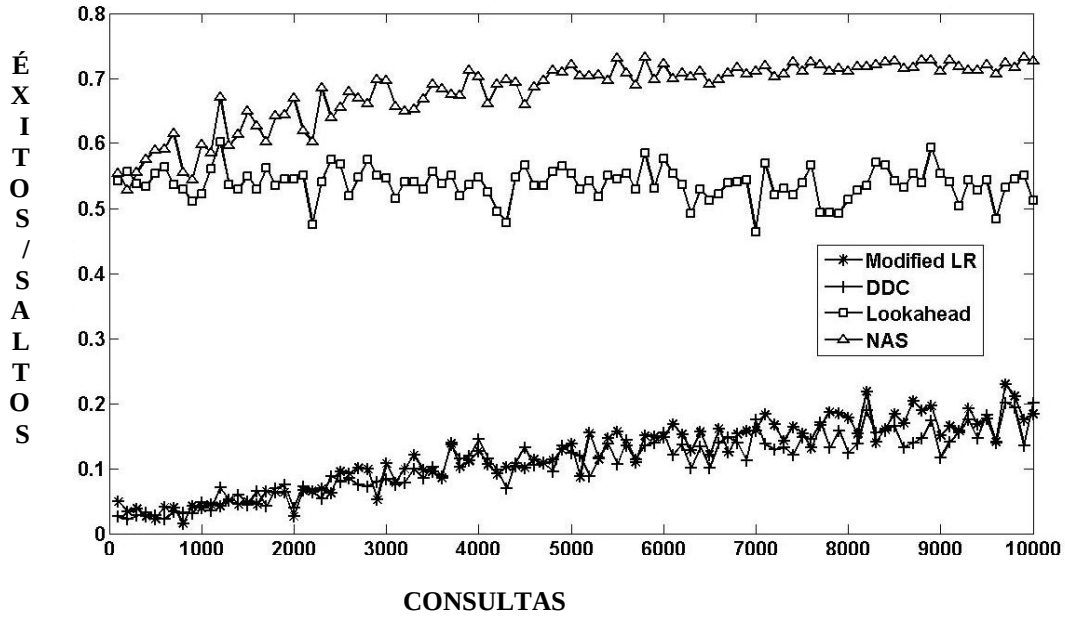


Figura 4.10. Continuación.

Por otro lado, la Figura 4.10 (b) muestra el promedio de saltos realizados durante un conjunto de consultas con NAS y tres diferentes configuraciones de él. Para LR modificado y DDC, el comportamiento es aproximadamente el mismo; el conteo del promedio de saltos empieza en 10 y termina en 9.5 saltos por consulta. Sin embargo, el Lookahead y NAS empiezan en 7.8 y después de 1500 consultas, el comportamiento cambia; para Lookahead el promedio del conteo de saltos termina en 8 y para NAS termina en 6.9 saltos por consulta.

Finalmente la Figura 4.10 (c) muestra la eficiencia promedio, para el LR modificado y DDC, el comportamiento es aproximadamente el mismo; en el inicio la eficiencia es 0.05, y al final la eficiencia se incrementa a 0.2 éxitos por salto. Sin embargo, para el Lookahead y NAS el comportamiento evoluciona de tal manera que la eficiencia promedio empieza en 0.55 y después de 1,500 consultas, el comportamiento cambia; para Lookahead, el promedio de la eficiencia termina en 0.5, para NAS, el promedio de la eficiencia termina en 0.7. Así de esta manera, el mejor rendimiento se obtuvo con la combinación de tres estrategias.

Además, los resultados revelan que para la configuración especificada, el método Lookahead muestra la mayor contribución al final del rendimiento de NAS, dando una eficiencia de 0.5 éxitos por salto. Mientras el DDC y el LR modificado tienen un impacto similar de 0.2 éxitos por salto. En otras experimentaciones y con otras configuraciones [93], las estrategias analizadas han mostrado diferentes contribuciones. Debido a estos resultados, es necesario un estudio a profundidad para identificar, las relaciones que existen entre las características del problema y la configuración de los parámetros del algoritmo, con el objetivo de encontrar los mayores beneficios de cada una de las estrategias propuestas para NAS.

En resumen para la solución de SQRP, se propuso un novedoso algoritmo llamado NAS que esta basado sobre algoritmos de colonias de hormigas existentes pero incorporándole estrategias del medioambiente local: Reglas de Aprendizaje modificadas (LR), DDC y Lookahead. Tres funciones son usadas para aprender del rendimiento pasado: importancia de los éxitos (HIT), importancia del tiempo de vida (ITL_HOP) e importancia de la distancia (ID_HOP). El resultado de estás combinaciones es una baja cantidad de saltos y una alta cantidad de éxitos, superando a los dos algoritmos propuestos en trabajos relacionados, Random-Walk y SemAnt.

El análisis y la simulación confirman que las técnicas propuestas son mas efectivas para mejorar la eficiencia de la búsqueda. Específicamente el algoritmo NAS en la eficiencia muestra una mejoría de seis veces mas eficiencia, que el algoritmo SemAnt y más de diez veces de mejoría que el algoritmo Random-Walk.

Se observa que al incluir el aprendizaje y la caracterización con LR modificado y DDC respectivamente, el algoritmo evoluciona hasta alcanzar un promedio de 2 éxitos con 9.5 saltos por consulta. Para el caso de la exploración con Lookahead, el algoritmo se mantiene constante en su rendimiento de 4 éxitos con 8 saltos. Combinando las tres estrategias el algoritmo NAS evoluciona hasta alcanzar 5 éxitos con 6.9 saltos. Como se observa el mejor resultado fue obtenido en la combinación de las estrategias propuestas.

4.3 Identificación de Factores de Desempeño y sus Relaciones

Identificar los factores y sus relaciones es una actividad importante, debido a que proporciona información para la toma de decisiones en los experimentos.

4.3.1 Selección de las Características de SQRP que afectan el rendimiento del algoritmo NAS

En esta etapa se identifican las características principales de SQRP para comprobar el efecto que producen en la respuesta del experimento, así como sus relaciones con otros factores que también influyen en el desempeño del algoritmo. Se propone realizar un diseño de experimentos factorial completo [80]. Este tipo de diseños son utilizados cuando se desea analizar los efectos de mas de dos factores así como las interacciones de un conjunto de factores relacionados con una o más variables de respuesta. Posteriormente se realiza el análisis de los parámetros que influyen en el rendimiento del algoritmo NAS.

Características Significantes de SQRP

Se consideraron las principales características de SQRP y sus distribuciones de probabilidad para estudiar el impacto en el rendimiento del algoritmo NAS. La primer característica a analizar fue las distribuciones de probabilidad de las conexiones de los nodos, las cuales definen la topología de la red. Enseguida las distribuciones de probabilidad del contenido de los repositorios, la cual es la frecuencia de la distribución de la información contenida en los nodos. Finalmente, las distribuciones de probabilidad de las consultas, las cuales definen cuantas veces la misma consulta es repetida dentro de los diferentes nodos de la red. Para simplificar la descripción del experimento, los siguientes nombres cortos serán usados respectivamente: distribución de topología, repositorios y consultas. La distribución más común sobre la Internet es: scale-free [45] y aleatoria [94].

Supuestos del Experimento

Los supuestos del experimento pueden ser establecidas como: ¿Afecta el conjunto de características de SQRP a el rendimiento del algoritmo NAS?. Los supuestos se establecen de la siguiente forma:

H_0: La distribución de la topología, repositorios y consultas del problema de SQRP, No tienen efectos significantes en el rendimiento del algoritmo NAS.
H_1: La distribución de la topología, repositorios y consultas del problema de SQRP, tienen efectos significantes en el rendimiento del algoritmo NAS.

Diseño Experimental

Dos ***factores*** fueron estudiados: el tipo de distribución de probabilidad y las características de SQRP. Los ***niveles del primer factor*** son: scale-free (SF) y aleatorias (random, RM). Los ***niveles para el segundo factor*** son: la topología, repositorios y consultas.

Las variables de respuesta son: ***respuesta*1** que es el promedio de la cantidad de información descubierta (*hits*) y ***respuesta*2** que es el promedio del número de nodos visitados (*hops*). En el experimento las ***replicas*** son la diferentes corridas del algoritmo metaheurístico NAS.

Desarrollo del experimento

Cada instancia esta formada por tres archivos que corresponden con las características del problema: topología, repositorios y consultas. La generación de esos archivos será explicada a continuación. En este experimento 2^3 instancias fueron generadas. Estas instancias son las combinaciones de las distribuciones de probabilidad de las características del problema. Cada instancia fue replicada 10 veces con NAS. La topología y los repositorios fueron simulados de manera estática, mientras que el conjunto de consultas fue dinámico. Se utilizaron las instancias generadas en la sección 4.2.2. para los dos experimentos.

En cada simulación se resolvieron 10,000 consultas. El promedio del rendimiento fue calculado tomando las medidas de rendimiento cada 100 corridas. El rendimiento fue medido a través de dos variables de respuesta. La variable de *respuesta 1* mide el promedio de éxitos; el máximo número de éxitos fue puesto en 5; la variable de *respuesta 2* mide el número de saltos; en cada consulta el máximo número de saltos fue puesto en 25.

El experimento estadístico fue desarrollado con el paquete estadístico MINITAB. Se llevó a cabo un diseño factorial completo 2^k con 2 factores, 3 niveles (k), y 10 replicas. El nivel de significancia se fijo en $\alpha = 0.05$, y los valores de las variables para cada instancia. El diseño factorial fue analizado para determinar la importancia de las interacciones de los factores, en otras palabras, que combinaciones de los factores tienen los efectos mas grandes sobre el rendimiento. Con esta información, se corrió un Modelo General Lineal (MGL), el cual es basado principalmente sobre las pruebas de la ANOVA. MGL fue usado para confirmar, si las combinaciones seleccionadas de los factores son significantes estadísticamente. Los resultados son descritos en la siguiente sección con las gráficas de los efectos principales y las interacciones.

Análisis de los Resultados Estadísticos

La Figura 4.11, revela que el tipo de distribución de probabilidad de los factores principales tiene efectos sobre el promedio del rendimiento del algoritmo NAS. Esto se observa en las gráficas ya que al cambiar de una distribución a otra, hay una variación en la inclinación de la línea.

La topología (T) es el factor que tiene el mayor efecto sobre el promedio de la calidad y la eficiencia de NAS. Para las dos variables de respuesta, el promedio de los éxitos y el promedio de los saltos, la topología reacciona de manera diferente, es decir, cuando la respuesta es el promedio de los éxitos los mejores valores se encuentran cuando la topología es scale-free y para la variable de respuesta promedio de los saltos los mejores valores se encuentran cuando la topología es aleatoria.

Además del parámetro topología es importante ver los tres parámetros juntos para analizar las relaciones entre ellos. Los otros dos factores (repositorios y consultas) muestran efectos irrelevantes sobre el promedio de la calidad y la eficiencia de NAS. El tamaño y la dirección en las gráficas confirman esta afirmación.

Las gráficas de la Figura 4.12 revelan que el tipo de distribución de probabilidad de los factores tienen efectos significantes sobre el promedio del rendimiento de NAS. Para las dos variables de respuesta, las interacciones entre repositorios y consultas, tienen los efectos mas grandes sobre el promedio de la calidad y eficiencia de NAS. Las otras dos interacciones muestran efectos irrelevantes por los resultados aquí obtenidos, se puede afirmar que la significancia estadística de efectos relevantes en un proceso de optimización se pueden identificar con la metodología MLG.

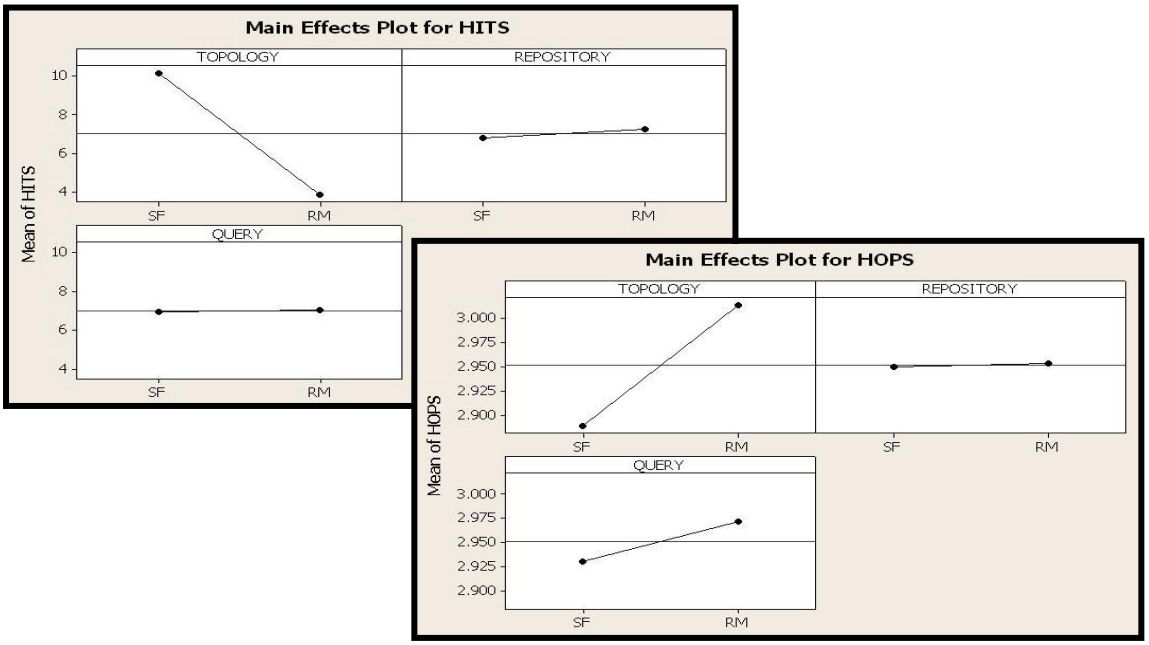


Figure 4.11. Resultados de los efectos principales: *respuesta1* (HITS) y *respuesta2* (HOPS)

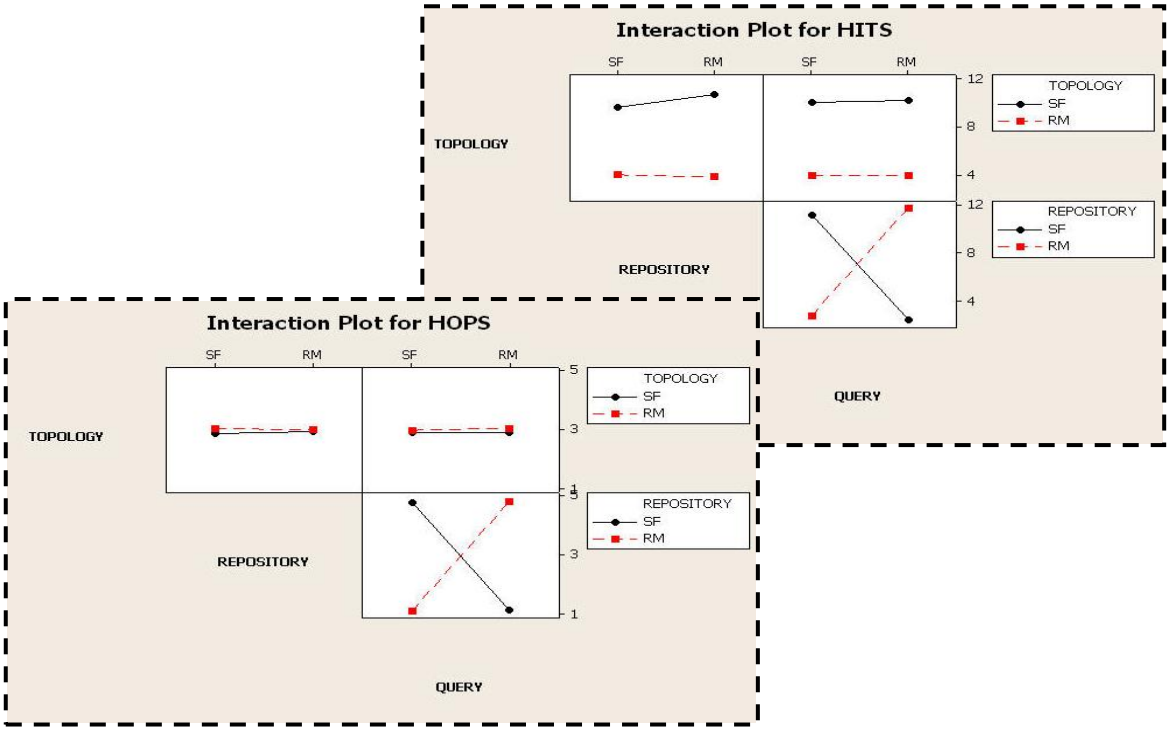


Figure 4.12. Resultados de las interacciones: *respuesta1* (HITS) y *respuesta2* (HOPS)

Los datos de la Tabla 4.5 muestran que el factor que tiene mayor significancia es la topología con una magnitud de -5.204, que indica que el cambio va de un nivel alto a un nivel bajo al cambiar de distribución de probabilidad de scale-free a aleatoria. Para los factores que están relacionados se puede observar que existe una fuerte relación entre los repositorios y las consultas. Además también hay una relación entre los tres factores. Esta información refuerza la información que dan las gráficas de las Figuras 4.11 y 4.12., y muestra que las relaciones entre las distribuciones de probabilidad de los tres factores son significantes para el rendimiento.

Tabla 4.5. Estimación de los efectos para la variable de respuesta de los éxitos

Experimento Factorial: Efecto de la topología, repositorios, consultas en los éxitos

Termino	**Efectos**
Topología	-5.204
Repositorios	0.296
Consultas	0.070
Topología*Repositorios	-0.296
Topología*Consultas	-0.154
Repositorios*Consultas	8.501
Topología*Repositorios*Consultas	-3.412

En este experimento una metodología basada en Diseño de Experimentos (DOE), fue presentada para contestar cuestiones de investigación sobre factores de desempeño. Los resultados obtenidos en la experimentación muestran que, la variabilidad en las distribuciones de probabilidad de las características de SQRP (topología, consultas y repositorio) afectan al rendimiento del algoritmo NAS.

Los resultados experimentales muestran que la distribución topológica tiene una gran influencia para las dos variables de respuesta: número de recursos encontrados para las consultas y número de pasos que hacen. Se encontró que las distribuciones de las consultas y los repositorios están fuertemente relacionadas en ambas variables de respuesta.

Con el análisis de resultados se identificaron los tipos de distribuciones de probabilidad que son más significantes para el rendimiento del algoritmo. En la topología, scale-free es resultante, en las consultas, la distribución aleatoria tiene un mayor impacto; y en los repositorios, scale-free es la más significantes.

4.3.2 Preparación del Algoritmo NAS, para la Selección de Parámetros Significantes

De los resultados obtenidos en la sección anterior se concluyó que la topología es considerada una de las principales características de SQRP. El algoritmo NAS usa una métrica para caracterizar la topología llamada DDC. En este experimento se analiza el rendimiento del algoritmo NAS con la métrica DDC y además se prueba con otra métrica topológica local

llamada *grado del nodo*. Las dos métricas fueron definida en la sección 1.2.1 y miden como es la vecindad del nodo en función de sus nodos vecinos.

Para llevar acabo el primer experimento se trabaja con la regla de transición de estado de NAS descrita detalladamente en las Ecuaciones (4.7) y (4.8), de la sección 4.2.2.

La regla de selección es:

$$s = \begin{cases} \arg\max_{L_n \forall n \in \{0,|L|\}} \left\{ \left[\tau_{r,L_n,t} \right] \cdot \left[DDC_{L_n} + ID\_HOP_{r,L_n,t} \right]^{\beta} \right\}, & \text{if } q \le q_0 \text{ (explotación)} \\ S, & \text{de otra forma (exploración),} \end{cases}$$

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

La función de la regla de transición es seleccionar el siguiente nodo después de haber hecho el análisis del nodo actual y haber aplicado el Lookahead. Como se puede observar en las dos ecuaciones citadas, la regla incluye actualmente la métrica topológica local DDC. El objetivo de este experimento será cambiar el DDC por el grado de un nodo, posteriormente comparar los resultados del rendimiento del algoritmo para seleccionar aquella que guíe mejor el algoritmo hacia nodos mejor conectados.

Desarrollo del experimento

Como se puede observar en las fórmulas anteriores, el DDC influye en la selección del estado siguiente. Al estar trabajando con el algoritmo se llegó a conocer mejor su comportamiento, por lo que surgió la siguiente pregunta: ¿Qué pasaría si se utiliza el grado del nodo en lugar de la métrica DDC?. Debido a esta interrogante y en la búsqueda de la respuesta a dicho cuestionamiento se planteó contrastar el rendimiento del algoritmo cuando utiliza DDC y cuando utiliza el grado del nodo.

Se utilizaron las instancias generadas en la sección 4.2.2. para los dos experimentos. Los resultados son el promedio de 30 ejecuciones del programa para algunos de los grupos de instancias de prueba.

Análisis Estadísticos de los Resultados

Una parte de los resultados obtenidos se presentan en la Tabla 4.6. Estos son resultados promedios de 30 ejecuciones del programa para algunos de los grupos de instancias de prueba, resultados similares fueron obtenidos con todas las instancias. La primera columna indica el nombre de la instancia, la segunda columna la tasa de aprendizaje (Hits/Hops) cuando se utiliza la métrica DDC, la tercera columna la tasa de aprendizaje utilizando el grado del nodo y la cuarta columna el porcentaje de mejora al utilizar el grado con respecto a utilizar el DDC.

La tasa de aprendizaje indica cuantos pasos en promedio tiene que dar el algoritmo para localizar un recurso, cuanto menor sea es mejor. Además se presentan las gráficas de estos resultados en las Figuras 4.13 y 4.14. El eje de las abscisas indica el nombre de la instancia y el eje de las ordenadas indica la taza de aprendizaje Hits/Hops (Éxitos/Saltos). Las Figuras 4.13 y 4.14 muestran los resultados gráficos de la Tabla 4.6 en dos partes.

Tabla 4.6. Resultados generados de las experimentaciones de DDC y el grado.

Instancia	hits/hops con DDC	hits/hops con Grado	% de mejora
BA2_0	1.681628	0.613971	63.489456
BA2_1	1.660164	0.579591	65.088340
BA2_2	6.914521	6.055680	12.420827
BA2_3	1.331784	1.251822	6.004124
BA2_4	1.832003	1.012265	44.745463
BA2_5	5.201389	4.829858	7.142923
BA2_6	4.398632	3.539422	19.533563
BA2_7	1.180062	1.160062	1.694826
BA2_8	2.389509	2.028176	15.121634
BA2_9	1.376024	1.305741	5.107753
BA3_0	1.742118	1.031781	40.774349
BA3_1	1.901270	1.075380	43.438857
BA3_2	7.704003	6.128471	20.450819
BA3_3	1.947650	1.737404	10.794858
BA3_4	5.696471	4.093567	28.138534
BA3_5	1.331592	1.203761	9.599883
BA3_6	2.823010	2.777373	1.616609
BA3_7	1.553711	1.367540	11.982387
BA3_8	4.532161	3.439857	24.101174
BA3_9	1.413565	1.374005	2.798616

Como se observa en las gráficas de las Figuras 4.13 y 4.14; el desempeño del algoritmo NAS, cuando utiliza el grado del nodo en lugar de la métrica DDC en la regla de selección, en general brinda mejores resultados ya que la cantidad de pasos necesaria para encontrar un recurso en la red es menor. Debido a estos resultados se rechaza la hipótesis nula y se acepta la hipótesis alternativa. Se puede observar en las gráfica que existen algunas instancias para las cuales el comportamiento del algoritmo es el mismo independientemente de la métrica que se este aplicando. Para estos casos se requiere un estudio profundo de las instancias determinar en que condiciones se presenta este fenómeno.

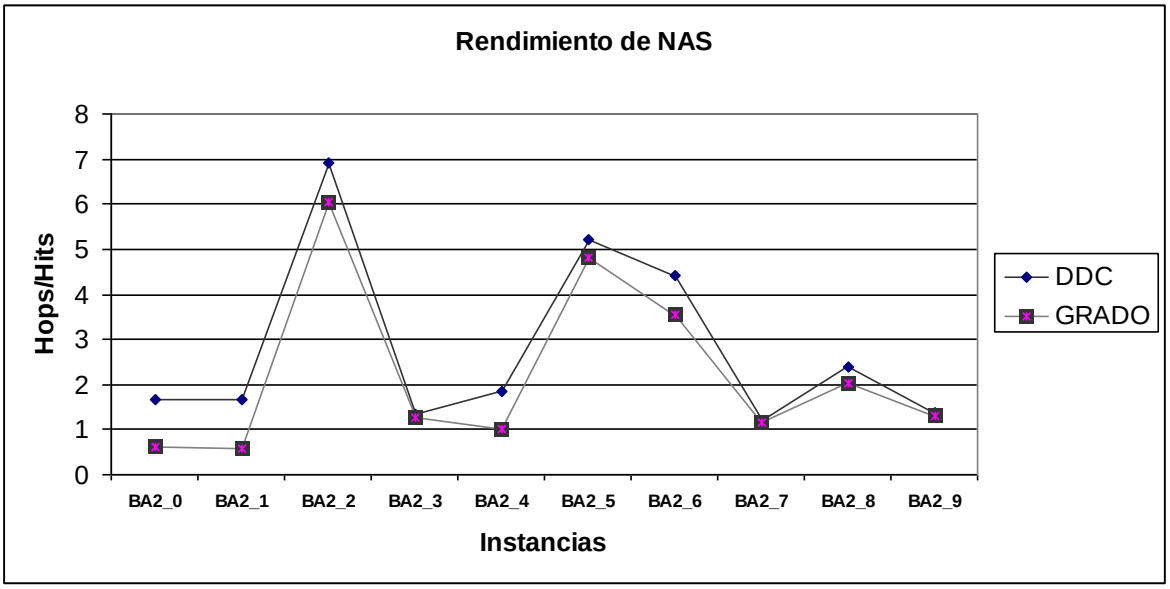


Figura 4.13. Resultados del rendimiento del algoritmo NAS para el DDC y el GRADO

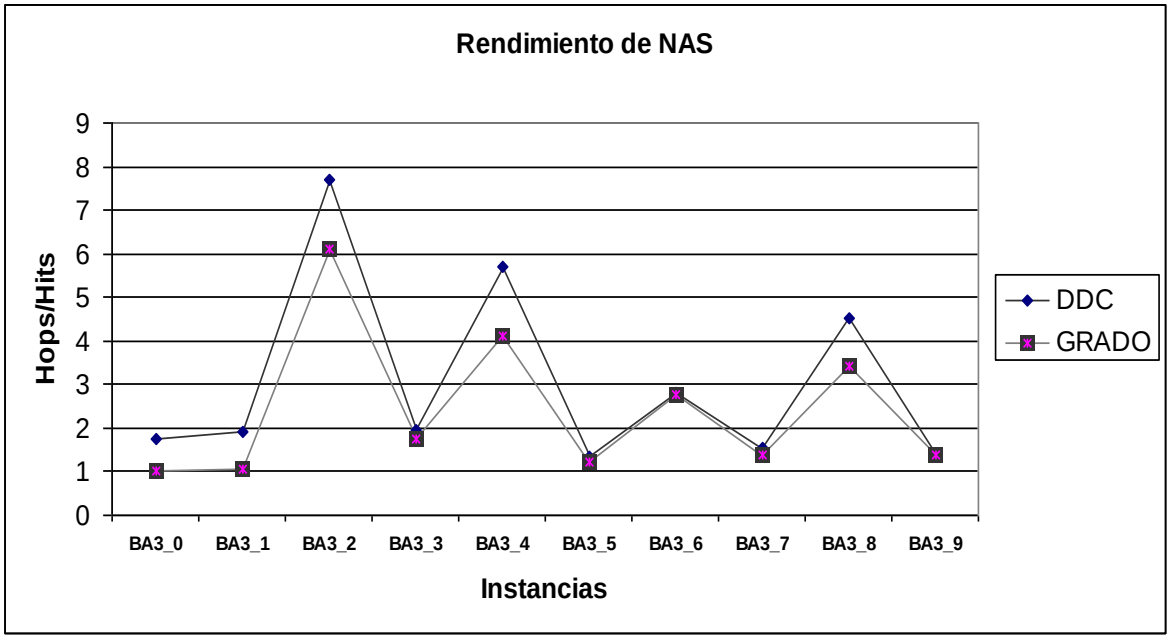


Figura 4.14. Resultados del rendimiento del algoritmo NAS para el DDC y el GRADO. Con instancias del grupo BA3.

La conclusión a la que se llega es que la métrica DDC, mide las diferencias entre el nodo que lanza la consulta y sus vecinos. Si el nodo que lanza la consulta esta muy bien conectado y este tiene un vecino con pocas conexiones entonces el DDC del vecino será alto y la métrica nos guiará hacia ese vecino. En otras palabras la métrica no siempre guía hacia vecinos bien conectados, sino hacia la mayor diferencia entre los grados de los nodos, mientras que con la del métrica grado siempre dice cual es el nodo con mayor grado de los vecinos.

Además de la selección de la métrica de caracterización de la topología, se realizo un análisis de las reglas de transición de estados y actualización de la feromona. Estas reglas están formadas por parámetros claves como son: la tabla de feromonas (τ), el grado (k_i), la tabla de experiencia que guarda las distancias hacia los recursos previamente encontrados ID_HOP. La regla de transición de estados es una de las más importantes debido a que a través de ella se van seleccionando los nodos de la ruta bajo ciertos criterios de conectividad, distancias, o valores de los vecinos. Debido a esto se establece el uso de parámetros de importancia a través de los cuales algunos términos de la ecuación pudieran ser cancelados o dar la misma importancia a todos los elementos de la regla de transición. Esta modificación es uno de los primeros pasos para definir la función de ajuste de control adaptativo del parámetro tiempo de vida.

Para solucionar este problema se agregaron parámetros de importancia y parámetros de equilibrio en la regla de transición. Los parámetros de importancia son usados para intensificar los valores en las ecuaciones cuando se tienen dos expresiones o más en una ecuación y es necesario dar mayor peso a una parte de la ecuación que a otra. Los parámetros de equilibrio son para evitar que el algoritmo se vuelva voraz, es decir que para la toma de decisiones no tome en cuenta las dos medidas locales (grado y distancia).

Los parámetros son: β_1 (intensificación de medidas locales), β_2 (intensificación de feromona), w_h (factor de equilibrio entre *hits* y *hops*), w_d (importancia del grado) y w_i (importancia de la distancia). La nueva propuesta de la Regla de transición se muestra en la Ecuaciones (4.12) y (4.13):

$$s = \begin{cases} \arg\max_{L_n \forall n \in \{0,|L|\}} \left\{ \left[\tau_{r,L_n,t} \right]^{\beta 2} \cdot \left[w_d(k_i) + w_i(ID\_HOP_{r,L_n,t}) \right]^{\beta 1} \right\}, & \text{if } q \le q_0; \text{explotación} \\ S, & \text{de otra forma (exploración),} \end{cases} \quad (4.12)$$

$$S = f(p_{r,u,t}), p_{r,u,t} = \frac{\left[\tau_{r,u,t} \right]^{\beta 2} \cdot \left[w_d(k_i) + w_i(ID\_HOP_{r,u,t}) \right]^{\beta 1}}{\sum_{\forall i \in \Gamma(r) \wedge i \notin V_k} \left[\tau_{r,u,t} \right]^{\beta 2} \cdot \left[w_d(k_i) + w_i(ID\_HOP_{r,u,t}) \right]^{\beta 1}} \quad (4.13)$$

Donde: *r* es el nodo actual en la ruta, V_k es la lista de nodos visitados, $\tau_{r,v,t}$ es la feromona de ir al nodo *v* a partir del *r* cuando se busca *t*, β_2 parámetro que define la intensificación de la feromona, w_d parámetro que define la importancia del grado, w_i parámetro que define la importancia de la distancia. $ID\_HOP_{r,u,t}$ es la longitud de la ruta más corta conocida del nodo *r* transitando por la arista (*r*,*v*). β_1 parámetro que define la intensificación de las medidas locales. *p* es un número aleatorio uniforme en [0,1]. k_i es el grado del nodo *i*.

Como se puede observar en la regla de transición al agregar β_2 que es el intensificador del rastro de la feromona y β_1 que es el intensificador de las métricas locales. Estos parámetros tendrán la función de guiar el algoritmo con las métricas locales cuando $\beta_2 = 0$, o por la feromona cuando $\beta_1 = 0$. En este experimento los valores iniciales son $\beta_1 = 2$ y $\beta_2 = 1$.

La regla de actualización de la feromona es mostrada en la Ecuación (4.10).

$$\tau_{r,s,t} \leftarrow (1-\alpha)\cdot\tau_{r,s,t} + \alpha\cdot\left[w_h HIT + (1-w_h) ITL\_HOP\right] \qquad (4.10)$$

en esta el parámetro w_h es el factor de equilibrio entre las funciones históricas HIT que esta relacionada con los éxitos pasados e ITL_HOP con la distancia usada para encontrar un recurso.

4.3.3 Selección de los Parámetros significantes del Algoritmo NAS

El algoritmo NAS tiene nueve parámetros que definen parte de su comportamiento algorítmico, dos de ellos: ID_HOP_0 y τ_0 son parámetros para inicializar estructuras de aprendizaje. Los valores de inicialización deben ser valores pequeños cercanos a cero, razón por lo cual no se considerarán estos parámetros en la búsqueda de la configuración inicial del algoritmo y se tomarán los valores de ellos como se recomienda por la literatura (ID_HOP_0=0.001 y τ_0=0.009).

Diseño Experimental

Siete ***factores*** o parámetros, fueron estudiados: *q* (parámetro que establece la relación entre la exploración y la explotación), ρ (factor de evaporación), β_1 (intensificación de medidas locales), β_2 (intensificación de feromona), w_h (factor de equilibrio entre *hits* y *hops*), w_d (importancia del grado) y w_i (importancia de la Distancia).

La variable de respuesta es el promedio del desempeño final del algoritmo NAS al ejecutarlo 30 veces con la misma configuración sobre el conjunto de instancias SQRP. El desempeño del algoritmo se mide a través de la tasa de aprendizaje *hits/hops*, que indica cuantos recursos encontró el algoritmo en cada nodo que visitó. El conjunto de instancias SQRP son las descritas en la sección 4.2.3, en la que se presenta información detallada sobre su generación. En total se utilizaron 90 instancias SQRP.

Desarrollo del experimento

Cada instancia del experimento está formada por una combinación de los valores de los parámetros y la variable respuesta que corresponde a dicha combinación. Los parámetros β_1 y β_2 toman valores enteros entre uno y cinco, el resto de los parámetros toman valores continuos entre cero y uno. En total 200 instancias fueron generadas.

Una vez obtenido el conjunto de instancias para el experimento, se buscó encontrar relaciones causa-efecto de los parámetros sobre el desempeño del algoritmo NAS, para lo cual es necesario hacer uso de un algoritmo para identificación de relaciones causales. El algoritmo PC es un algoritmo diseñado para encontrar relaciones causa-efecto entre variables en un conjunto de datos. Se utilizó implementación de dicho algoritmo del software Tetrad, para realizar el análisis sobre los datos.

La Figura 4.15 muestra un diagrama causal obtenido a través del software Tetrad, donde los parámetros se encuentran marcados con un rectángulo. Mientras que las mediciones se encuentran en óvalos con relleno gris. Las líneas indican las relaciones causa-efecto, el extremo de la línea sin flecha corresponde a la causa mientras que el otro extremo, el que tiene cabeza de flecha indica en cuál medición se produce el efecto. Para obtener dicho diagrama fue necesario indicar a Tetrad que utilizara el algoritmo PC con una prueba estadística de independencia z de Fisher con un nivel de significancia de 0.05.

En la Figura 4.15 se observa que los valores iniciales para los parámetros w_i, w_h y β_2 no influyen sobre el desempeño de NAS. El parámetro q influye tanto en el número de éxitos (*hits*) como en el tamaño de la ruta en la búsqueda (*hops*); mientras que los parámetros w_d, β_1 y ρ influyen directamente sobre la relación *Hits*/*Hops*. Valores iniciales adecuados para q, w_d, β_1 y ρ pueden producir un mejor comportamiento del algoritmo por lo que es importante encontrar un método para lograr un buen ajuste para dichos parámetros mientras que w_i, w_h y β_2 pueden tomar valores propuestos en la literatura.

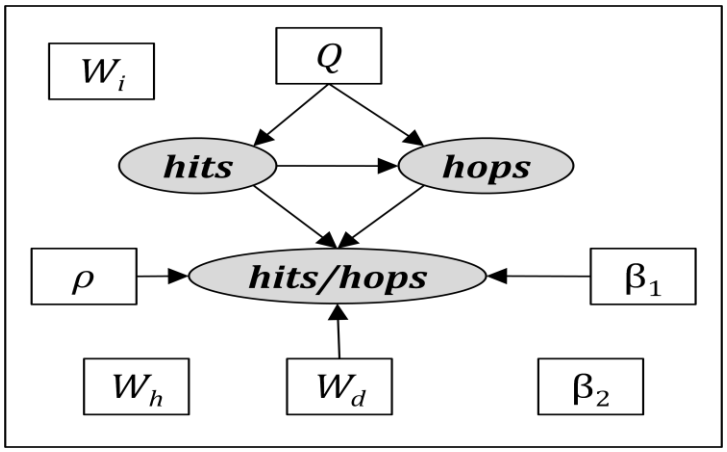


Figura 4.15. Diagrama de causalidad entre los parámetros y el desempeño de NAS.

4.3.4 Modelado de las Relaciones entre los Parámetros del Algoritmo NAS y las Características de SQRP

En la sección 4.3.2 se trabajó con el rediseño de una parte de la regla de selección intercambiando el grado k_i por el DDC e incluyendo parámetros que se han denominado de intensificación e importancia para algunos de los miembros de la Ecuaciones (4.12) y (4.13) como se ve a continuación:

$$s = \begin{cases} \arg\max_{L_n \forall n \in \{0,|L|\}} \left\{ \left[\tau_{r,L_n,t} \right]^{\beta 2} \cdot \left[w_d(k_i) + w_i(ID\_HOP_{r,L_n,t}) \right]^{\beta 1} \right\}, & \text{if } q \le q_0 \text{;explotación} \\ S, & \text{de otra forma (exploración),} \end{cases} \tag{4.12}$$

$$S = f(p_{r,u,t}),\ p_{r,u,t} = \frac{\left[\tau_{r,u,t}\right]^{\beta 2} \cdot \left[w_d(k_i) + w_i(ID\_HOP_{r,u,t})\right]^{\beta 1}}{\sum_{\forall i \in \Gamma(r) \wedge i \notin V_k} \left[\tau_{r,u,t}\right]^{\beta 2} \cdot \left[w_d(k_i) + w_i(ID\_HOP_{r,u,t})\right]^{\beta 1}} \quad (4.13)$$

En esta sección se trabaja con el modelado de las relaciones entre los parámetros del algoritmo NAS y las características de SQRP, usando también la Ecuaciones (4.12) y (4.13). Como se observa esta ecuación ya incluye información relacionada con los parámetros de NAS (ID_HOP y τ) y características de SQRP (k_i). En esta sección se trabaja con el parámetro ID_HOP a través del cual se rediseña su estructura para incluir información relacionada con repositorios y consultas. Para obtener ventaja de esta relación inicialmente se diseño una función denominada ID_HOP, que es la función de importancia de la distancia, la cual fue descrita en la sección 4.2.3. Siendo manejada a través de la Ecuación (4.3):

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

Además de esta función se creo la tabla de experiencia del nodo (Tabla 4.7), en la cual se almacena información importante acerca de las últimas cinco consultas que han transitado por el nodo, y a cada nodo se le asoció una tabla de experiencia.

Al hacer un análisis de la información de la tabla de experiencia y la información almacenada por la métrica ID_HOP, se propone modificar dicha función con la finalidad de aprovechar realmente la información obtenida de los repositorios y las consultas. Además de que este rediseño es uno de los primeros pasos para formar la versión adaptativa del algoritmo NAS, llamada AdaNAS.

Descripción del cambio de Estructura

La función de aprendizaje propuesta por NAS denominada Función de Importancia de la Distancia (ID_HOP) es usada por los agentes de búsqueda. El agente de búsqueda cuando avanza necesita hacerlo a través de la regla de transición, y dentro de dicha regla, ID_HOP es considerado como uno de los criterios para seleccionar el nodo siguiente en la trayectoria. La hormiga de retroceso actualiza el valor de ID_HOP según la cantidad de nodos recorridos. Rutas más largas son menos deseables que las rutas cortas. Esta estructura fue rediseñada para lograr fusionarla con la tabla de experiencia del nodo (Tabla 4.7). Para esta actividad se creó una tabla de registro para el ID_HOP y la tabla de experiencia del nodo.

Descripción de la Tabla de Experiencia

Debido a que las consultas generadas en la red siguen una distribución topológica de tipo scale-free se podría deducir que, aunque existen muchos tipos de recursos en la red, sólo unos pocos son altamente consultados. Partiendo de esta idea, se diseñó una primera estrategia que hace uso de las consultas anteriores para ajustar el valor de tiempo de vida.

A cada nodo de la red se agregó una estructura de datos como la presentada en la Tabla 4.7, la cual será llamada *Tabla de Experiencia del Nodo*. Dicha tabla tiene la finalidad de almacenar información importante acerca de las últimas cinco consultas que han transitado por el nodo. Ésta consta de seis columnas, la primera de ellas almacena cual es el recurso que se estaba consultando y las siguientes cinco columnas indican cuántos recursos se encontraron cuando la hormiga avanzó 1, 2, …, 5 pasos respectivamente.

Cuando una consulta es generada, el agente de búsqueda avanza por la red y los nodos por los cuáles transita van agregando nuevas filas en la tabla de experiencia. En caso de ya no haber espacio en la tabla, el registro correspondiente a la consulta más antigua es eliminado y sustituido por los datos de la nueva consulta.

Tabla 4.7. Tabla de experiencia del nodo

Recurso	Pasos				
	1	2	3	4	5

Cuando el agente encuentra un recurso lo regresa al nodo solicitante invirtiendo la ruta por la que ha transitado. Cuando el agente está enviando el recurso al nodo solicitante, las tablas de experiencia de los primeros cinco nodos en la ruta invertida son actualizadas. De esta forma la tabla de experiencia mantiene un registro de cuántos recursos se encontraron en los siguientes cinco pasos que avanzó la búsqueda.

Descripción de la nueva estructura del ID_HOP

La Figura 4.16. muestra la nueva estructura de la tabla ID_HOP, en la cual se optimizo el espacio, además de que permite conocer cuantos recursos se encuentran en el nodo más cercano que satisface la consulta. Dicha tabla va llenándose a medida que las consultas van ejecutándose en la red, de tal forma que los agentes del pasado heredan su conocimiento a futuros agentes que consulten lo mismo.

La tabla de la Figura 4.16 esta formada por las claves, el inverso de la distancia al primer recurso encontrado y la cantidad máxima de recursos encontrados en esa ruta. El funcionamiento de la tabla es idéntico al explicado para la tabla de experiencia, pero este formato cuenta con más elementos para la toma de decisiones de los agentes.

Los resultados experimentales de este cambio se verificarán en conjunto con los resultados de la versión final de AdaNAS.

2	3	4	9	10	CLAVES
3	5	4	3	2	DISTANCIA
5	3	3	1	1	No. RECURSOS

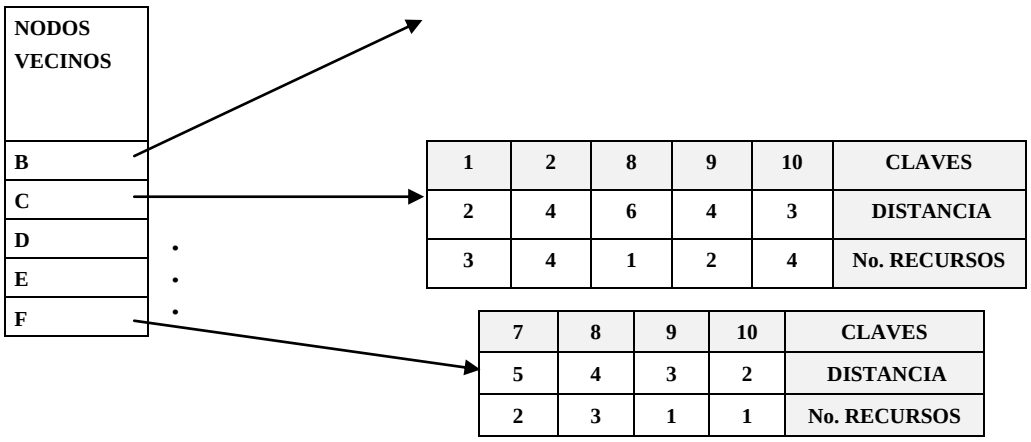


Figura 4.16. Estructura de la tabla ID_HOP

4.4 Desarrollo de un Sistema de Búsqueda Adaptativo

El algoritmo AdaNAS es un algoritmo metaheurístico que usa estrategias adaptavivas. Este algoritmo tiene como base el algoritmo NAS e incluye todas las sus características. El objetivo del rediseño del algoritmo NAS a AdaNAS es incluir una función de ajuste para el parámetros TTL en tiempo de ejecución del algoritmo.

Los mecanismos que pueden extender el tiempo de vida a una hormiga son llamados reglas de supervivencia. Esta incorpora información de consultas pasadas sobre estrategias de aprendizaje incluidas en AdaNAS, las características básicas de SQRP y un conjunto de parámetros que serán ajustados acorde con los resultados encontrados cuando se usa la regla de supervivencia. La regla evalúa la longitud de las rutas mas cortas conocidas que empiezan con la conexión (*i*,*j*) desde el nodo actual *i* a un nodo que contiene buenos resultados para una consulta *t*.

El ajuste por medio del *Control de Parámetros*, ofrece la ventaja de estar monitoreando los cambios en el ambiente al mismo tiempo que considera el estado actual del algoritmo. Mediante una configuración inicial a partir de técnicas de afinación y posteriormente ejercer control sobre los parámetros, esto brinda mejores resultados.

4.4.1 Diseño del Modelo Adaptativo Basado en Agentes

Un proceso adaptativo es caracterizado por el ambiente en el cual opera y el plan que el sistema sigue para reaccionar a los cambios percibidos en el ambiente. La división entre el sistema y el ambiente es a menudo implícita pero sencilla, aunque a veces puede ser una fuente de dificultades.

Se usa un modelo discreto de adaptación basado sobre la propuesta de Holland [101], en este se asume que el sistema realiza las acciones en pasos discretos $t = 1, 2, 3, \ldots$, esta suposición aplica prácticamente a todos los sistemas computacionales. El sistema es subdividido en cuatro partes: la estructura *A* para adaptar el ambiente (llamados agentes), los planes de adaptación *P*, la memoria *M* y los operadores *O*.

La estructura a adaptar *A* tiene varios estados alternativos A_1, A_2, A_3, ... dentro de los cuales uno será seleccionado por el sistema, acorde con las observaciones hechas por el ambiente. También *P* es un conjunto de reglas, una o mas de ellas serán aplicadas, estas reglas son aplicadas a las operaciones sobre el conjunto *O*. Un operador puede ser ambos una función determinista (A_i, P_j)$\rightarrow A_k$ o una función estocástica para una distribución de probabilidad sobre un conjunto de estados A_k. La memoria M permite que el sistema recolecte información sobre las condiciones del ambiente y del propio sistema para ser usada como una base para la toma de decisiones. Las observaciones del ambiente son modeladas como estímulos que activan los operadores.

Los objetos del proceso son:

- ε ; el medio ambiente del sistema bajo el proceso de adaptación.
- *S ;* el plan del sistema adaptativo el cual determina las modificaciones de la estructura en respuesta a el ambiente.

Para AdaNAS, el ambiente es una red P2P, la cual emite dos tipos de estímulos: la ocurrencia de los documentos a ser buscados (I_1) y el grado k_i del nodo *i* (I_2), el ambiente tiene un orden para enviar el estimulo cuando I_1 ocurre, siempre lo hace antes de I_2.

AdaNAS es un sistema de colonia de hormigas, donde cada hormiga es modelada como un agente. Este sistema maneja cuatro tipos de agentes: el *agente de consulta*, es responsable de atender las consultas del usuario y crear la *hormiga de búsqueda*, además de hacer la evaporación local de la tabla de feromona. Existe un agente de consulta para cada nodo de la red, y este permanece durante toda la corrida del algoritmo AdaNAS.

La *hormiga de búsqueda* usa las estrategias de aprendizaje para guiar la consulta y cuando encuentra recursos crea la *hormiga de recuperación*. La hormiga de búsqueda termina su vida cuando el valor de TTL=0 ó la cantidad de recursos encontrados (hits) son satisfechos. Cuando la hormiga de búsqueda finaliza su proceso, crea la *hormiga de actualización*.

La *hormiga de recuperación* es responsable de informar a la hormiga de consulta la cantidad de recursos encontrados en el nodo. Además de actualizar los valores de las estructuras de aprendizaje (*N, H, D*) que serán la base de la regla de supervivencia.

La *hormiga de actualización* actualiza la tabla de feromonas sobre los nodos de la ruta generada por la hormiga de búsqueda. La cantidad de feromona depositada depende de la cantidad de los recursos encontrados (*hits*) y el número de aristas viajadas (*hops*) por la hormiga de búsqueda.

El proceso de búsqueda implementado en la *hormiga de consulta* se requiere que sea adaptativo, de esta manera *A* es definida en función de este agente. Los posibles estados de *A* son cinco:

A_1: Cuando el agente no tiene una ruta y es un nodo inicial. La hormiga pude ser solamente activada cuando la hormiga de consulta le envía una consulta y puede

solamente recibir un estimulo a la vez.

A_2: Una ruta ha sido asignada y TTL no ha alcanzado cero.

A_3: *TTL* es cero.

A_4: La hormiga de consulta usa la regla de supervivencia para extender el TTL.

A_5: *X*: estado terminal es alcanzado por la hormiga de consulta.

La memoria *M* esta compuesta por cuatro estructuras que almacenan información acerca de consultas previas. La primera de esas estructuras es τ es la una tabla de feromonas tridimensional. El elemento $\tau_{i,j,t}$ es la preferencia para moverse de un nodo *i* a un nodo vecino *j* cuando se busca una palabra *t*. En este trabajo se asume que cada consulta contiene una palabra y el número total de palabras (o conceptos) conocidos para el sistema es denotado por *C*.

Los planes adaptativos *P* son:

$P_{1.}$ La *hormiga de recuperación*, es creada cuando un recurso es encontrado. Modifica M_2, M_3 y M_4 cuando nueva información es encontrada.

$P_{2.}$ La *hormiga de actualización*, modifica la tabla de feromona $M_1 = \tau$, cuando la hormiga de búsqueda alcanza cero y la regla de supervivencia podría no ser activada.

P_3. La *regla de transición*, selecciona el siguiente nodo aplicando el aprendizaje inherente almacenando en la feromona y la estructura de memoria $M_2 = D$.

$P_{4.}$ La *Regla de supervivencia*, prosigue cuando el aprendizaje almacenado en M_2,M_3 y M_4, permite extender TTL y determina en cuanto el TTL debe ser extendido.

$P_{5.}$ La *Regla de Transición modificada* es una variación de la regla de transición, en la cual se eliminan los efectos de la feromona y el grado.

Los operadores *O* de AdaNAS son los siguientes:

O_1: $(A_1, I_1) \rightarrow A_1$	Documentos fueron encontrados y una hormiga recuperación (P_1) actualiza la memoria, no cambia el estado del sistema.
O_2: $(A_1, I_2) \rightarrow A_2$	Una hormiga selecciona la ruta acorde con la regla de transición (P_3).
O_3: $(A_2, I_1) \rightarrow A_2$	Similar a O_1.
O_4: $(A_2, I_2) \rightarrow \{(p_{2,2,2}, A_2)$, $(p_{2,3,2}, A_3)\}$	La regla de transición (P_3) , ambas mantienen a la hormiga en el mismo estado con la probabilidad $P_{2,2,2}$ o se mueve al estado A_3 con una probabilidad $P_{2,3,2}$.
O_5: $(A_3, I_1) \rightarrow A_3$	Similar a O_1.
O_6: $(A_3, I_2) \rightarrow \{(p_{3,4,2}, A_4)$, $(p_{3,X,2}, X)\}$	La regla de supervivencia (P_4) y la hormiga de actualización (P_2) con probabilidad $P_{3,X,2}$ aplicando (P_2) la hormiga alcanza su estado terminal.
O_7: $(A_4, I_1) \rightarrow A_4$	Similar a O_1.
O_8: $(A_4, I_2) \rightarrow \{(p_{3,3,2}, A_3)$,	La regla de transición modificada (P_5) la hormiga puede tomar

$(p_{3,4,2}, A_4)\}$ cualquiera de los dos estados, permanecer en el mismo estado o moverse al estado A_3.

La tabla de feromonas $M_1 = \tau$ es separada en n-tablas dimensionadas $\tau_{j,t}$ una para cada nodo. Estas tablas solamente contienen las entradas $\tau_{i,j,t}$ para un nodo fijo i y por lo tanto tiene dimensiones C x $|\Gamma(i)|$. Las otras tres estructuras son también tablas de tres dimensiones, M_2 = D, $M_3 = N$ y $M_4 = H$. Cada división de la tabla local bi-dimensional n es idéntica a la explicada anteriormente. La información de esas estructuras es definida como: actualmente estando en el nodo i y están buscando t, hay una ruta de longitud $D_{i,j,t}$ empezando en el vecino j que tomará ventaja a un nodo identificado en $N_{i,j,t}$ que contiene $H_{i,j,t}$ éxitos o documentos que son iguales a lo buscado.

La Figura 4.17 muestra un diagrama que representa el modelo adaptativo de AdaNAS. El ambiente envía dos clases de estímulos: el número de documentos encontrados (línea punteada, I_1) y el grado del nodo (línea sólida, I_2). Un operador es seleccionado y la acción que se ejecutará dependerá de la señal de entrada. El estilo de la línea para el estado de transición es acorde con el estilo de la línea, la línea punteada corresponde con las transiciones desde I_1, y sólida para I_2.

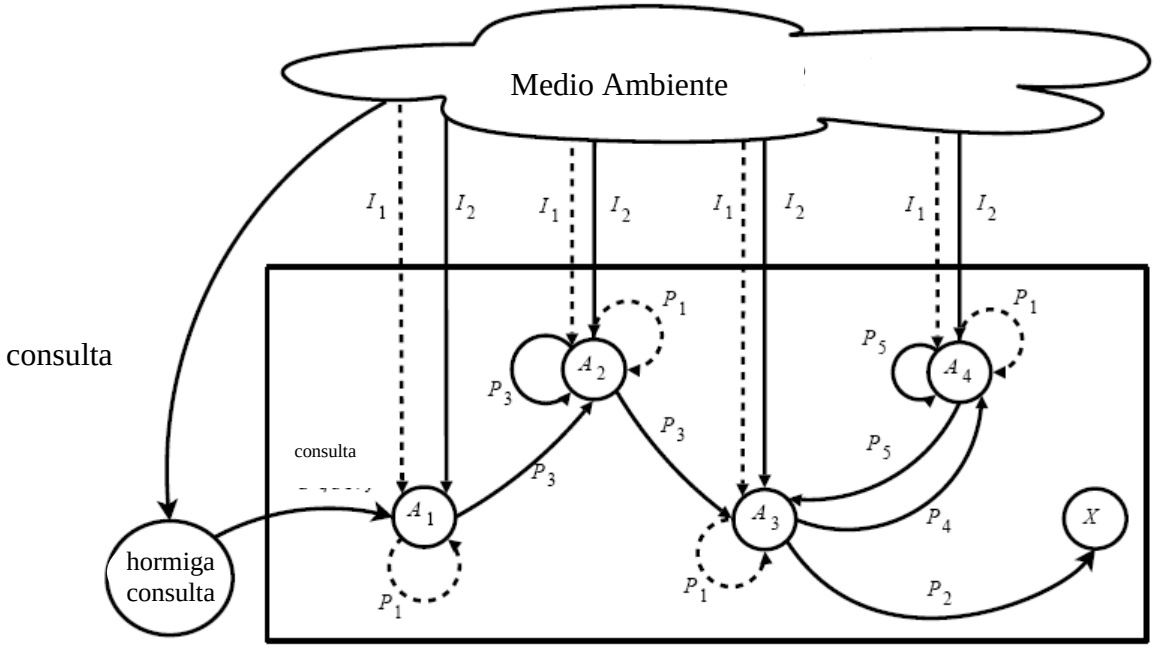


Figura 4.17. Modelo Adaptativo de AdaNAS

El modelo adaptativo detallado de AdaNAS con accesos a la memoria, se muestra en la Figura 4.18. Cuando una flecha apunta desde el plan adaptativo P_i a la estructura de memoria M_j indica que el plan adaptativo modifica a la memoria, pero si la flecha apunta de M_j a P_i, el plan adaptativo requiere información de la memoria para operar.

Las interacciones de este modelo se realizan como se describe a continuación: P_1 (hormiga de recuperación) actualiza las estructuras de memoria: N, D y H (letras a, b y d). Dichas estructuras son usadas para incrementar el tiempo de vida (letra c). Solamente cuando la hormiga de

búsqueda ha alcanzado TTL=0, crea una hormiga de actualización (P_2) la cual tiene como objetivo actualizar el rastro de la feromona (letra c). La regla de supervivencia (P_4) puede ser solamente aplicada cuando TTL=0 y se evaluará si el proceso del sistema termina o continua (letra c). La regla de transición (P_3) hace uso de dos estructuras para determinar el siguiente estado: τ y *D* (la feromona y la distancia entre el nodo actual y el nodo conocido mas cercano con documentos). La regla de transición modificada (P_5) se activa cuando la regla de supervivencia ha incrementado el tiempo de vida (letra c y d). P_5 obtiene la información de *D* (letra d). Además el sistema de adaptación depende de la información almacenada en las memorias, en *D*, *H* y *N*, con el objetivo de obtener mejores resultados de consultas pasadas.

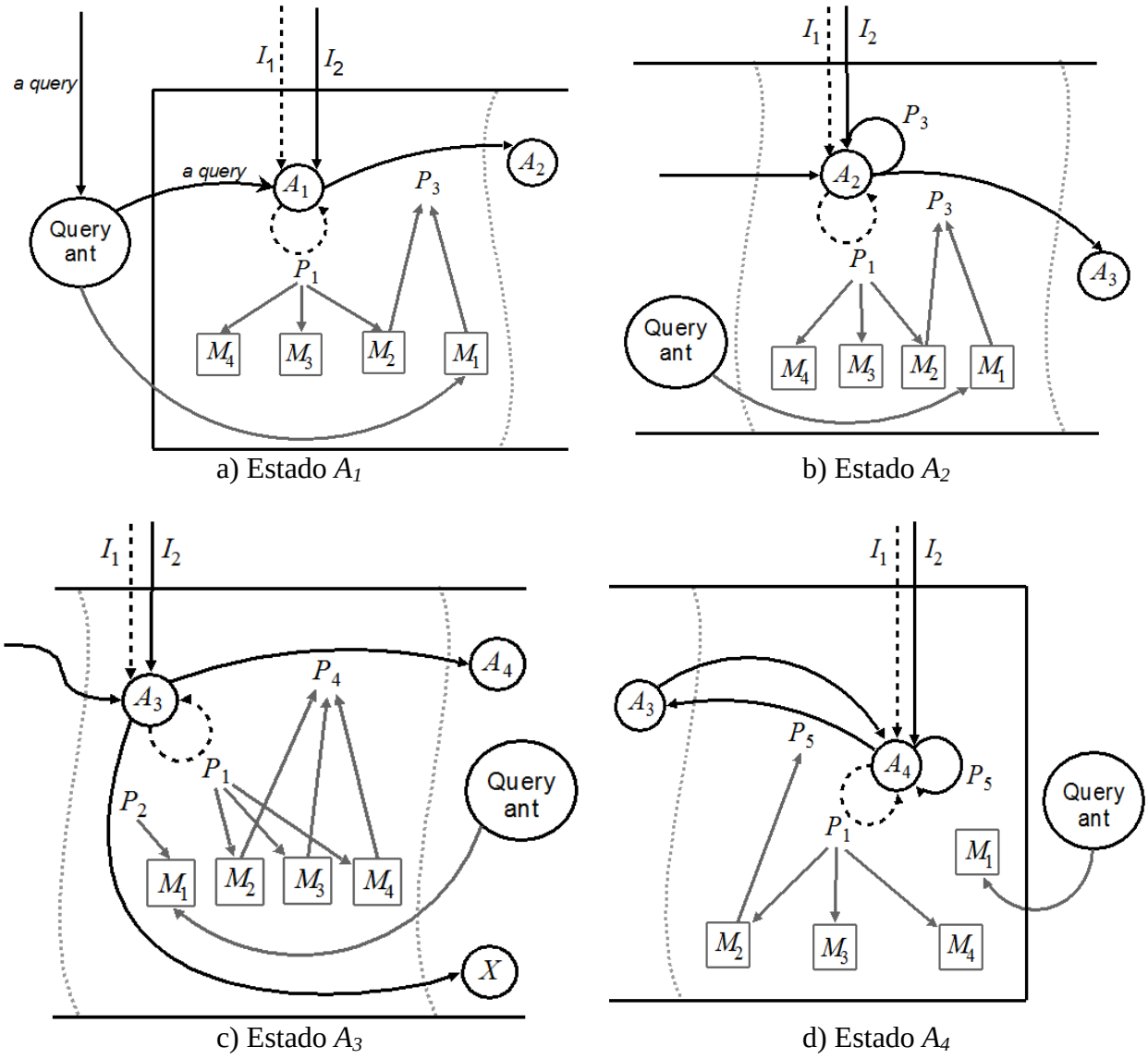


Figura 4.18. Modelo Adaptativo de AdaNAS.

4.4.2 Desarrollo del Algoritmo Adaptativo: AdaNAS

AdaNAS es un algoritmo metaheurístico, donde un conjunto de agentes independientes llamados hormigas cooperan independiente y esporádicamente para lograr una meta común. El algoritmo tiene dos objetivos: busca maximizar el número de recursos encontrados por las hormigas y minimizar el número de saltos dados por la hormiga. AdaNAS guía las consultas hacia nodos que tienen mejor conectividad usando la métrica estructural del grado del nodo (definido en el capítulo 1). Además el uso de la técnica Lookahead, la cual por medio de las estructuras permite conocer los repositorios de los nodos vecinos de un nodo especifico.

El algoritmo AdaNAS ejecuta en paralelo todas las consultas a través de agentes de consulta. Cada nodo tiene un agente de consulta, el cual crea una hormiga de búsqueda para atender cada una de las consultas de los usuarios, asignándoles la palabra buscada *t* a la hormiga de búsqueda. Además el agente de consulta realiza periódicamente la evaporación de la feromona local del nodo donde se encuentra localizado. El proceso realizado por el agente de consulta es representado en la Tabla 4.8 que corresponde al Algoritmo del agente de consulta.

Tabla 4.8. Algoritmo del Agente de consulta

```
     Algoritmo 1
1    En paralelo para cada agente de consulta w localizado en el nodo r
2    Mientras el sistema este corriendo hacer
3        Si el usuario realiza consultas para encontrar documentos con la palabra t hacer
4            Crear una hormiga de búsqueda x(r,t, maxResult)
5            Activar x
6        Fin
7        Aplicar evaporación de la feromona
8    Fin
9    Fin del paralelismo
```

En el Algoritmo 2 de la Tabla 4.9 se muestra el proceso realizado por la hormiga de búsqueda. Como se muestra todas las hormigas de búsqueda actúan en paralelo. En una fase inicial (líneas 4 - 8) la hormiga busca en el repositorio local, y si encuentra alguna coincidencia con el documento que esta buscando, entonces crea una hormiga de recuperación. Enseguida continua con su proceso de búsqueda (líneas 9 - 25) mientas no se le acabe el tiempo de vida (TTL > 0) y no haya encontrado el total de documento solicitados (maxResults).

El proceso de búsqueda esta dividido en tres secciones: evaluación de resultados, extensión de TTL y la selección del siguiente nodo. La **primera sección**, la evaluación de resultados (líneas 10 - 15) ejecuta la técnica clásica Lookahead, es decir, la hormiga *x* localizada en el nodo *r*, checa la estructura Lookahead, la cual indica cuantos documentos coincidentes están en cada uno de los nodos vecinos de *r*. Esta función necesita tres parámetros: el nodo actual (*r*), la palabra (*t*) y el conjunto de nodos conocidos (*conocidos*) por la hormiga. El conjunto de conocidos indica que nodos la función Lookahead deberá ignorar, debido a que sus documentos no coinciden con la palabra buscada y no serán tomados en cuenta. Si algún documento es encontrado, la hormiga de búsqueda crea una hormiga de recuperación y actualiza la cantidad de documentos coincidentes encontrados.

La **segunda sección** es la evaluación y aplicación de la extensión del TTL líneas 16 - 23), aquí es donde la hormiga verifica si el TTL ha llegado a cero, Si se cumple TTL = 0, la hormiga intentará

extender su tiempo de vida, en caso de que lo logre, la hormiga cambiará algunos parámetros de la regla de transición, creando una regla de transición modificada (línea 21).

En la **tercera sección** del proceso de búsqueda que es la selección del siguiente nodo, la regla de transición (original o modificada) será aplicada para la selección del siguiente nodo y algunas estructuras serán actualizadas. La etapa final ocurre cuando el proceso de búsqueda termina, entonces la hormiga de búsqueda crea una hormiga de actualización para llevar a cabo la actualización de la tabla de feromonas.

Tabla 4.9. Algoritmo de la hormiga de búsqueda

	Algoritmo 2
1	**En paralelo** para cada hormiga de recuperación *x*(*r*,*t*, *conocidos*)
2	**Inicializar**: *results* = 0, *TTL* = *maxTTL*, *hops* = 0, *la_results* = 0
3	**Inicializar:** *path* = {*r*}, Λ={*r*}, Conocido = {*r*}
4	*results* **=** obtener_documentos_locales(*r*)
5	**Si** *results* > 0 **entonces**
6	**Crear** una hormiga de recuperación y (*path, results, t*)
7	**Activar** *y*
8	**Fin**
9	**While** *TTL* < 0 y *results* < *maxResults* **do**
10	*la_result* = Lookahead (*r, t, conocido*)
11	**Si** *la_result* > 0 **entonces**
12	**Crea** hormiga de recuperación *y* (*path, results, t*)
13	**Activar** *y*
14	*results* = *results* + *la_ results*
15	**Fin**
16	**Si** *TTL* > 0 **entonces**
17	*TTL* = *TTL* **-** 1
18	**De lo contrario**
19	**Si** *results* < *maxResults* and ΔTTL (*k*, *results*, *hops*) > 0 **entonces**
20	*TTL* = *TTL* + Δ*TTL* (*k*, *results*, *hops*)
21	**Cambio de parametros**: q=1, w_d =0, β_2 =0
22	**Fin**
23	**Fin**
24	*hops* = *hops* + 1
25	*conocido* = *conocido* ∪ (*r* ∪ Γ(*r*))
26	Λ = Λ ∪ *r*
27	*r* = ℓ (*x*,*r*,*t*)
28	**Agregar** a la ruta(*r*)
29	**Fin**
30	**Crear** hormiga de actualización *z* (*path*,*t*)
31	**Activar** *z*
32	**Matar** *x*
33	**Fin del paralelismo**

El Algoritmo 3 mostrado en la Tabla 4.10 presenta el comportamiento paralelo para cada una de las hormigas de recuperación, las cuales recorrerán la ruta de la hormiga de búsqueda en sentido inverso. Y en cada nodo visitado actualizarán las estructuras *D*, *H* y *N* las cuales serán usadas por

las futuras consultas (líneas 7 - 11). Después de eso se le informará al agente de consulta ubicado en el nodo inicial cuantos documentos fueron encontrados por la hormiga de búsqueda y que ruta usó (línea 13).

La Tabla 4.11 muestra el Algoritmo 4. El cual presenta el comportamiento concurrente para cada una de las hormiga de actualización. Estas recorren de manera inversa la ruta marcada por la hormiga de búsqueda, y en cada nodo que visitaron actualizan el rastro de la feromona usando la Ecuación (5.20) de actualización global (línea 5).

Tabla 4.10. Algoritmo de la hormiga de recuperación

	Algoritmo 3
1	**Inicializar**: *hops* = 0
2	**En paralelo** para cada hormiga de recuperación *y* (*path, results, t*)
3	**Para** $i = \| path \| - 1$ **to** 1 **hacer**
4	$r = \text{path}_{(i-1)}$
5	$s = \text{path}_i$
6	$hops = hops + 1$
7	**Si** $D_{r,s,t} > hops$ entonces
8	$D_{r,s,t} = hops$
9	$H_{r,s,t} = results$
10	$N_{r,s,t} = \text{path}_{path}$
11	**Fin**
12	**Fin**
13	**Enviar** (*path, results*) para la hormiga de consulta en path_1
14	**Matar** *y*
15	**Fin del paralelismo**

Tabla 4.11. Algoritmo de la hormiga de actualización

	Algoritmo 4
1	**En paralelo** para cada hormiga de actualización *z* (*path,t,x*)
2	**Para** $i = \|path\| - 1$ **to** 1 **hacer**
3	$r = \text{path}_{(i-1)}$
4	$s = \text{path}_i$
5	$\tau_{r,s,t} = \tau_{r,s,t} + \Delta\tau_{r,s,t}(x)$
6	**Fin**
7	**Matar** *z*
8	**Fin del paralelismo**

a. Rediseño de las Estructura de Datos

Las estructuras de datos utilizadas en este sistema multiagente son esenciales para el funcionamiento del algoritmo ya que contienen toda la información necesaria relacionada con los nodos, sus vecinos y los agentes que se encargan de modificar el ambiente, así como las estructuras necesarias para representar el aprendizaje de consultas pasadas.

Existen dos elementos principales que definen el funcionamiento del algoritmo sobre una red compleja: la estructura del nodo y la de los agentes (hormigas), como se observa en la Figura 4.19. 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 y basados en las propuestas hechas en SemAnt y NAS la tabla de feromonas fue modificada con el propósito de tener diferente tipo de feromona, 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.

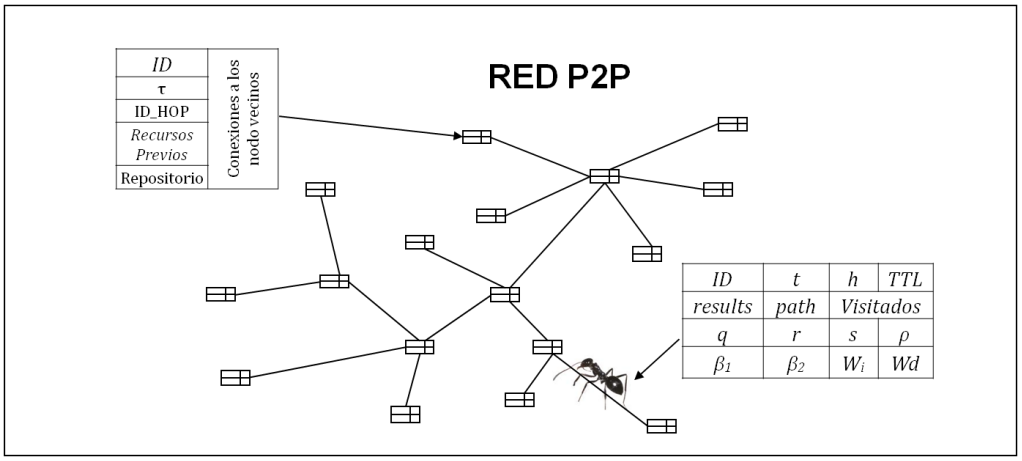


Figura 4.19**.** Estructuras principales del sistema multiagente.

Estructura del Nodo (*N*)

La *estructura de los nodos* (Figura 4.20) 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, 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. Las estructuras *D,H y N* que corresponde a 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, una tabla de *Recursos Previos* que indica cuantos recursos fueron encontrados en el pasado al seleccionar cierto nodo como siguiente en la ruta y el repositorio local de cada nodo, que comprende una tabla con los datos de cada uno de los recursos del repositorio local, así como la tabla de Lookahead que da información de los vecinos inmediatos y por último los enlaces hacia otros nodos.

La *estructura del agente* (hormiga, que representa al agente de búsqueda, actualización y de recuperación), mostrada en la Figura 4.21, está 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 (*Visitados*), el nodo que inicio la consulta (*r*), el nodo que está evaluando (*s*), los parámetros de configuración de las ecuaciones de selección (*q*, β_1,β_2) y de actualización (ρ, w_d, w_i) y la cantidad de recursos encontrados (*results*).

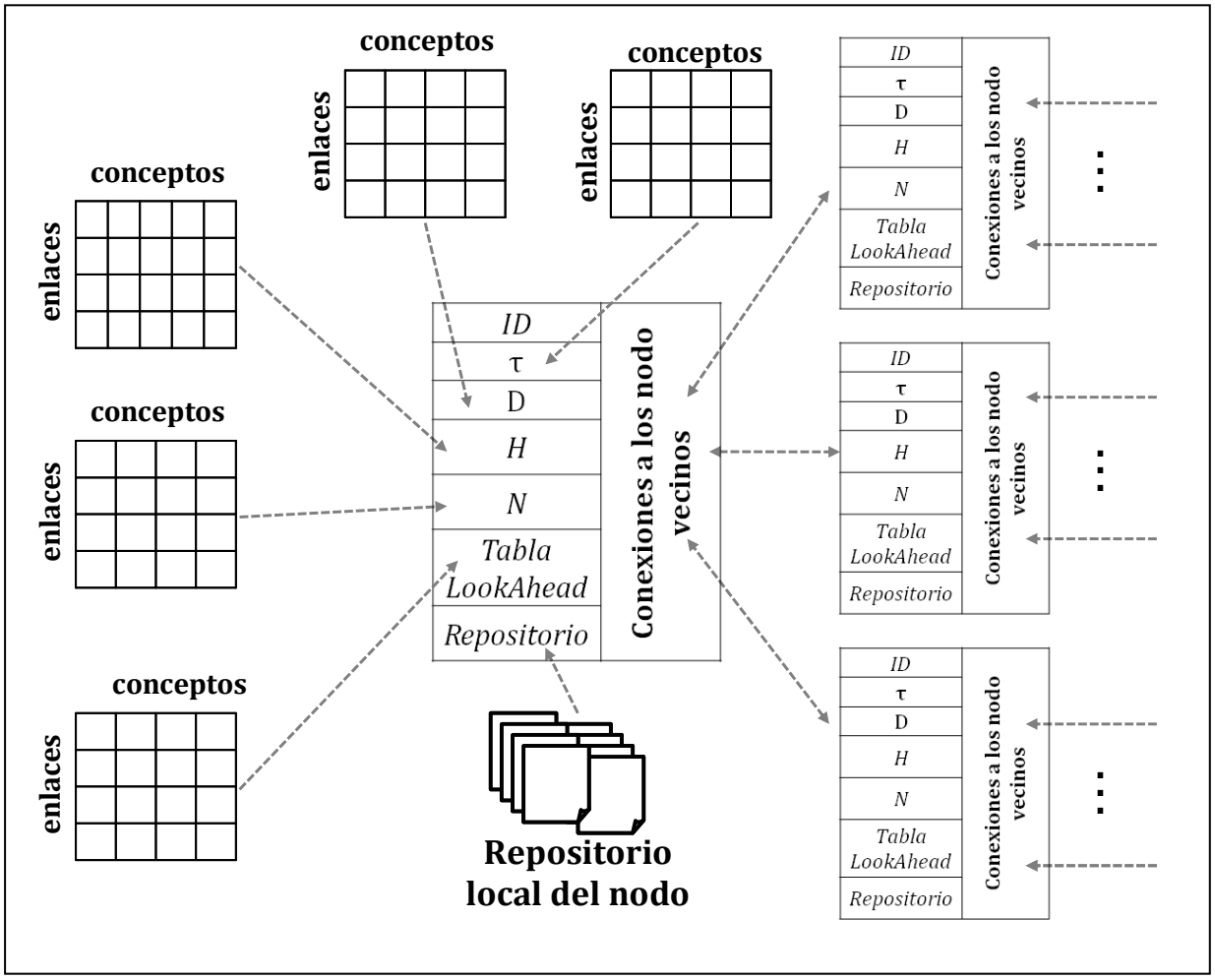


Figura 4.20. Estructura de un nodo

Estructura de la Tabla de Feromonas (τ)

La tabla de feromonas propuesta en el algoritmo NAS, se mejoró con el uso de listas de enlaces. El valor inicial de la feromona : $\tau_0 = 0.009$. Como se puede observar en la Tabla 4.12 (el cual es un caso hipotético, pero en naturaleza, muy probable que suceda debido al comportamiento del algoritmo), existen muchas claves donde el valor inicial de la feromona ($\tau_0 = 0.009$) se mantiene.

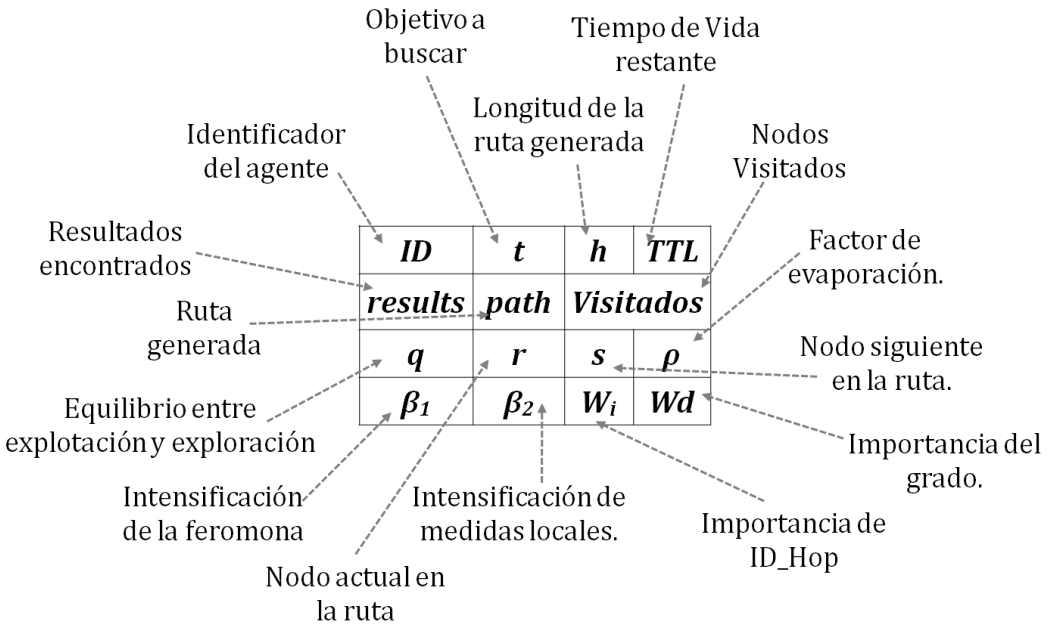


Figura 4.21**.** Estructura de un agente (hormiga)

Tabla 4.12. Tabla de feromonas del algoritmo NAS

NODOS VECINOS	CLAVES									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
B	0.009	0.5	0.3	0.3	0.009	0.009	0.009	0.009	0.1	0.1
C	0.1	0.1	0.009	0.009	0.009	0.009	0.009	0.2	0.1	0.1
D	0.009	0.009	0.009	0.009	0.009	0.1	0.1	0.1	0.1	0.1

Analizando esta estructura se encontró que dicha estrategia consume mucha memoria innecesaria, ya que hay celdas de la tabla de feromona que no cambian su valor durante la corrida del experimento. Además tomando en cuenta la distribución de la topología de la red y la cantidad total de nodos en el experimento, que son 1024, existen algunos nodos que llegan a tener hasta 50 o más vecinos. Mantener una estructura de este tipo ya que en ocasiones más del 50% de los nodos no modifica la feromona inicial.

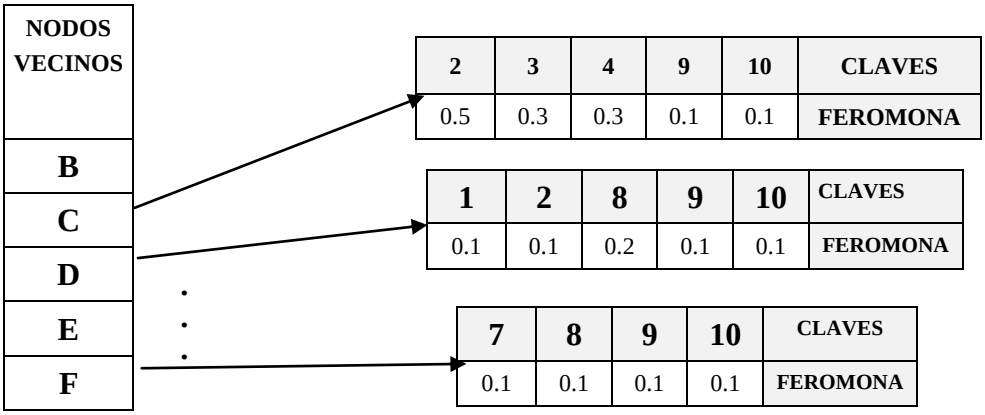


Figura 4.22. Estructura para la Tabla de Feromonas del algoritmo AdaNAS

Por lo observado anteriormente se implementó la estructura presentada en la Figura 4.22, y como se puede observar en la Figura 4.22, sólo se guardarán aquellas claves cuya feromona sea mayor al valor inicial. De esta manera la cantidad de espacio necesario para guardar la tabla de feromonas es menor.

Estructura de la Tabla Lookahead

La tabla de *Lookahead* del algoritmo NAS, permite a un nodo conocer la información almacenada en sus nodos vecinos. Por lo tanto, una técnica que permita almacenar dicha información de manera eficiente es crucial para el buen desempeña del método *lookahead*. En la Tabla 4.13 se observa la estructura de datos propuesta inicialmente para el algoritmo NAS. La tabla indica cuantos recursos del tipo "clave" guarda cada nodo vecino. Como se puede observar en la Tabla 4.13, existen muchos ceros. Con la finalidad de comprender la problemática se creó este ejemplo hipotético, sin embargo, este caso se presenta con mucha regularidad en las instancias de trabajo. La tabla de Lookahead es estática, una vez llena, el programa no modifica su valor durante toda su ejecución.

Tabla 4.13 Estructura de Datos para Lookahead del algoritmo NAS

NODOS VECINOS	CLAVES									
	1	2	3	4	5	6	7	8	9	10
B	0	5	3	3	0	0	0	0	1	1
C	1	1	0	0	0	0	0	2	1	1
D	0	0	0	0	0	1	1	1	1	1

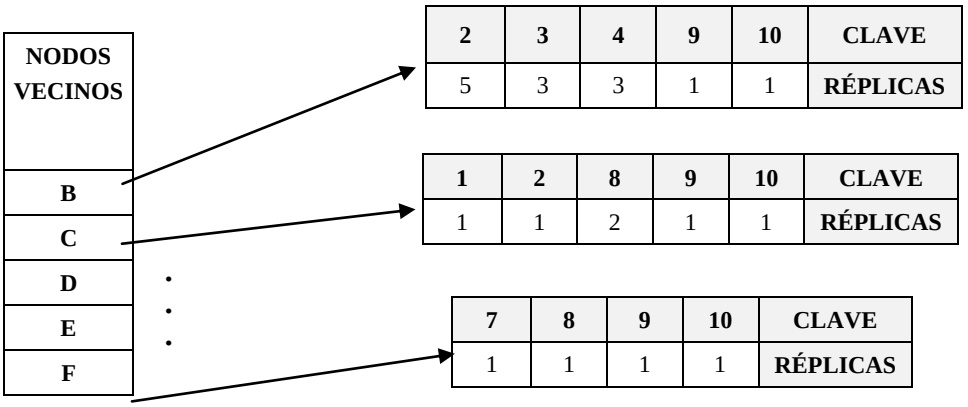


Figura 4.23. Estructura de Datos de la Tabla Lookahead para el algoritmo AdaNAS.

Analizando esta estructura se encontró que consume mucha memoria. Además tomando en cuenta la distribución de la topología de la red y la cantidad total de nodos en el experimento (1024), existen algunos nodos que llegan a tener hasta 50 o más vecinos. Mantener una estructura de este tipo donde en ocasiones más del 50% de los nodos mantienen muchos datos en 0 no es recomendable. Por lo observado anteriormente se planteó una estructura para la Tabla de Lookahead. Similar a la de la tabla de feromonas (Ver Figura 4.23).

Estructura de la Tabla *D* (ID_HOP) y Tabla de Éxitos (H)

En esta actividad se analizó la tabla *ID_HOP* del algoritmo NAS. Esta tabla se modificó nuevamente quedando una parte de la información en la tabla *D* y además creando la tabla *H* que es la tabla de los éxitos y su función es llevar el contador de los recursos encontrados. Dichas tablas se encuentran incluidas en la tabla *N* donde se encuentran los datos del Nodo. Como se puede observar en las Figuras 4.24 y 4.25. La estructura se mantiene como una lista enlazada.

Descripción de la nueva estructura *D*

La Figura 4.24. corresponde con la nueva estructura de la tabla D, y la Figura 4.25 corresponde con la tabla H. La tabla D contiene las distancias al primer recurso encontrado. El funcionamiento de la tabla es idéntico al explicado para la tabla de experiencia, pero este formato cuenta con más elementos para la toma de decisiones de los agentes.

La tabla *H* ésta permite conocer cuantos recursos se encuentran en el nodo más cercano que satisface la consulta. Las tablas se llenan a medida que las consultas se ejecutan en la red, de tal forma que los agentes del pasado heredan su conocimiento a futuros agentes que consulten lo mismo.

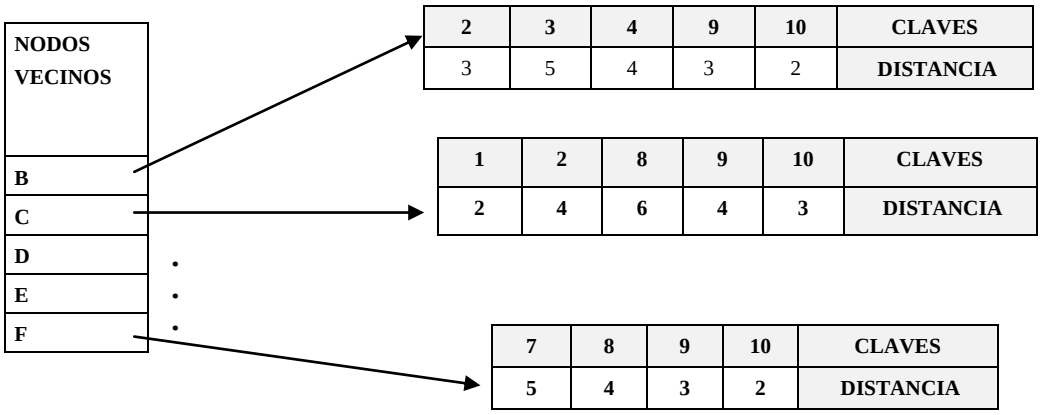


Figura 4.24. Estructura de la tabla *D*

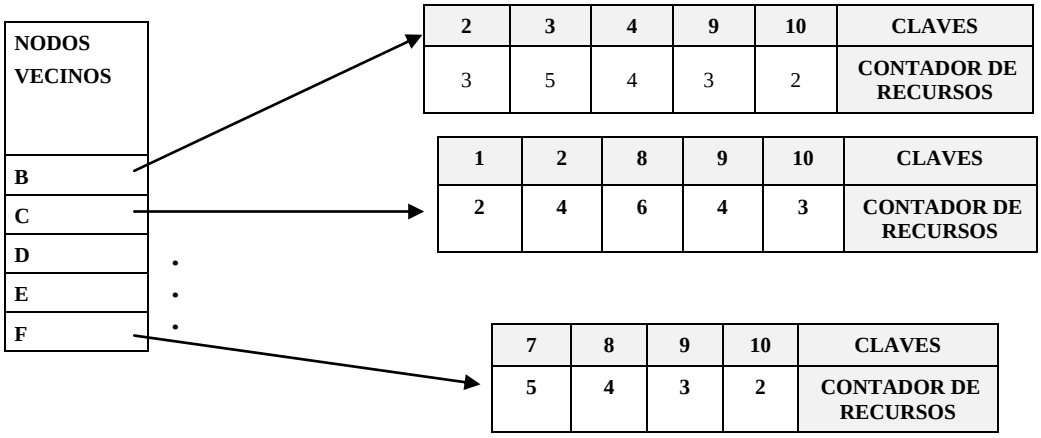


Figura 4.25. Estructura de la tabla *H*

4.4.3 Diseño de la Función de Ajuste de Control Adaptativo del Parámetro TTL

Esta sección describe las modificaciones hechas para las dos reglas de aprendizaje clásicas, originalmente propuestas por Dorigo [24] en el algoritmo ACS. La primera regla es llamada de transición de estado; con esta regla el siguiente nodo a ser visitado es seleccionado. La regla de transición usa dos estrategias: explotación y exploración. La segunda regla es llamada de actualización; con esta regla la hormiga la hormiga actualiza los nodos en la ruta viajada. La regla de actualización usa dos estrategias: actualización local y global de la feromona. Estas estrategias son usadas para retroalimentar el sistema sobre rutas exitosas.

La **regla de transición de estado** P_3 modificada para AdaNAS es formada por las Ecuaciones (4.14), (4.15), (4.17) y (4,18). Esta regla considera dos estructuras para determinar el siguiente estado: τ y D. La regla de transición para una hormiga x que esta buscando un una palabra clave t y se encuentra en el nodo r es:

$$\ell(x,r,t) = \begin{cases} \arg\max_{i\in(\Gamma(r)/\Lambda s)} \{\psi(r,i,t)\}, si \;\; p < q \\ \mathcal{L}(x,r,t) \qquad\qquad de\ otra\ forma; \end{cases} \tag{4.14}$$

Donde p es un número pseudo aleatorio, q es un parámetro del algoritmo que define la probabilidad de usar la técnica de exploración, Γ(r) es el conjunto de nodos vecinos de r, Λx es el conjunto de nodos previamente visitados por x, y

$$\psi(r,i,t) = (w_d \cdot \kappa(r,i) + w_i \cdot (D_{r,i,t})^{-1})^{\beta_1} \cdot (\tau_{r,i,t})^{\beta_2}, \tag{4.15}$$

Donde w_d es el parámetro que define la importancia del grado, w_i define la importancia de la distancia a través de los nodos mas cercanos ($D_{r,\ i,\ t}$). β_1 parámetro de intensificación de la contribución de medidas locales (grado y distancia), β_2 intensifica la contribución de la feromona ($\tau_{r,i,t}$), k (r,i) es la medida del grado normalizado expresado como se ve en la Ecuación (4.16):

$$\kappa_{r,i} = \frac{k_i}{\max_{j\in\Gamma(r)}\{k_j\}} \tag{4.16}$$

y $\mathcal{L}$ es la estrategia de exploración expresada como:

$$\mathcal{L}(x,r,t) = f(p_{x,r,i,t} \mid i \in \Gamma(r)), \tag{4.17}$$

donde $f(p_{x,r,i,t} \mid i \in \Gamma(r))$, es una función de selección pseudo-aleatoria basada sobre la bien conocida técnica de la ruleta [80] que selecciona un nodo i dependiendo de su probabilidad $p_{x,r,i,t}$ la cual indica la probabilidad que una hormiga x se mueva desde r hasta i buscando por una palabra t y es definida como:

$$p_{x,r,i,t} = \frac{\psi(r,i,t)}{\sum_{j\in(\Gamma(r)/\Lambda x)} \psi(r,i,t)} \tag{4.18}$$

La estrategia de exploración $\mathcal{L}$ de la Ecuación (4.17) es activada cuando $p \geq q$ y estimula a las hormigas a buscar por nuevas rutas. En caso que $p < q$, la estrategia de explotación es seleccionada: esta prefiere nodos que provee una gran cantidad de feromona y una mejor conectividad con un número pequeño de saltos hacia un recurso. Como es mostrado en la regla de transición Ecuación (4.15), β_2 es el intensificación del rastro de la feromona y β_1 es el intensificación de las métricas locales, esto significa que el algoritmo será guiado solamente por

las métricas locales cuando $\beta_2 = 0$ (anular la feromona) o por la feromona cuando $\beta_1 = 0$. En este trabajo los valores iniciales son $\beta_1 = 2$ y $\beta_2 = 1$.

Hay dos reglas de actualización básicas en un algoritmo de colonia de hormigas: la evaporación y el incremento de la feromona. La estrategia de evaporación de AdaNAS es basada sobre la estrategia usada en SemAnt [5] mientras que la estrategia de incremento de la feromona es basada sobre la propuesta en NAS; ambas reglas de actualización son descritas a continuación.

Regla de evaporación de la feromona, es una estrategia cuya finalidad es evitar que las aristas tomen valores demasiado grandes del rastro de la feromona causando un comportamiento voraz en el algoritmo. Cada unidad de tiempo la hormiga de consulta hace mas pequeño el rastro de la feromona del nodo donde esta ubicada, multiplicando el rastro por la razón de evaporación ρ, el cual es un número entre 0 y 1. Para evitar valores muy bajos en la feromona la regla incorpora un segundo termino que consiste del producto $\rho * \tau_0$, donde τ_0 es el valor inicial de la feromona. La Ecuación (4.19) expresa matemáticamente la regla de evaporación de la feromona.

$$\tau_{r,s,t} \leftarrow (1-\rho)\cdot\tau_{r,s,t} + \rho\cdot\tau_0 \tag{4.19}$$

Regla de incremento de la feromona, se ejecuta cuando una hormiga de búsqueda termina. La hormiga de búsqueda debe expresar su rendimiento en términos de feromona por medio de una hormiga de actualización, cuya función es incrementar la cantidad de feromona dependiendo de la cantidad de documentos encontrados y aristas recorridas por la hormiga de búsqueda. Esto se hace cada vez que una hormiga de actualización pasa por un nodo. Las Ecuaciones (4.20) y (4.21) describen la regla de incremento de la feromona:

$$\tau_{r,s,t} \leftarrow \tau_{r,s,t} + \Delta\tau_{r,s,t}(x) \tag{4.20}$$

donde $\tau_{r,s,t}$ es la preferencia de ir a *s* cuando la hormiga de búsqueda enta en *r* y buscando la palabra clave *t*, $\Delta\tau_{r,s,t}$ (*x*) es la cantidad de feromona soltada en $\tau_{r,s,t}$ por una hormiga de recuperación generada por la hormiga de búsqueda *x* y puede ser expresada como:

$$\Delta\tau_{r,s,t}(x) \leftarrow \left[w_h \frac{hits(x,s)}{maxResult} + (1-w_h)\frac{1}{hops(x,r)} \right] \tag{4.21}$$

donde *hits*(*x*,*s*) es la cantidad de documentos encontrados por la hormiga de búsqueda *x*, desde *s* hasta el final de su camino, y *hops*(*x*,*r*) es la longitud de la trayectoria recorrida por la hormiga de búsqueda *x* desde *r* hasta el nodo final de su ruta pasando por *s*.

Regla de transición modificada, es un caso especial de la regla de transición, donde $\beta_2 = 0$, $w_d = 0$ y $q = 1$. Esta regla es voraz y provoca la replicación de rutas generadas por otras hormigas. Esta regla se ejecuta cuando se selecciona la extensión de TTL por medio de la regla de supervivencia

de la Ecuación (4.2.7), cancelando la regla de transición normal. Esta regla se expresa en la Ecuaciones (4.22) y (4.23) .

$$\ell_m(x,r,t) = \left\{ \arg\max_{i \in (\Gamma(r)/\Lambda s)} \{\psi(r,i,t)\} \right. \tag{4.22}$$

Donde ℓ_m es la regla de transición modificada, r es el nodo actual en la ruta, t es la palabra clave buscada, Λx es el conjunto de nodos previamente visitados por la hormiga de búsqueda *x*, y

$$\psi(r,i,t) = (w_i \cdot (D_{r,i,t})^{-1})^{\beta_1} \tag{4.23}$$

Donde w_i define la importancia de la distancia a través de los nodos mas cercanos ($D_{r,\,i,\,t}$), que es la distancia necesaria para llegar al nodo mas cercano conocido con documentos que contienen la palabra clave *t*, desde *r* pasando por *i* y, β_1 es el parámetro de intensificación de la contribución de medidas locales (distancia).

Como puede verse en la Figura 4.18, p_1 (la hormiga de recuperación) actualiza la estructura de la memoria M_2 = D, $M_3 = N$ y $M_4 = H$. Estas estructuras son usadas en la regla de supervivencia P_4, para incrementar el tiempo de vida.

La regla de supervivencia puede aplicarse solamente cuando TTL = 0. La regla de supervivencia puede expresarse matemáticamente en términos de las estructuras *H*, *D* y *N* como es presentada en la Ecuación (4.27):

$$\Delta TTL(x,i,t) = \begin{cases} D_{i,\omega(x,i,t),t}; & si\ \Omega(x,i,t) > Z_x \\ 0; & en\ otro\ caso \end{cases} \tag{4.27}$$

donde ΔTTL(*x*,*i*,*t*) es el incremento asignado al TTL de la hormiga de búsqueda *x* (esto es, un número de saltos adicionales que se permitirán tomar a la hormiga) cuando se busca el recurso *t* estando actualmente en el nodo *i*. El número de pasos adicionales $D_{i,\omega(x,i,t),t}$ para llegar al nodo $\omega(x,i,t)$ es determinado por las rutas mas cortas generadas por las hormigas anteriores como se observa en la Ecuación (4.28) y se toman cuando su eficiencia asociada $\Omega(x,i,t)$ es mejor que Z_x la cual es una medida del rendimiento actual de la hormiga *x*, como se observa en la Ecuación (4.29). Las funciones auxiliares son:

$$\omega(x,i,t) = \arg\Omega(x,i,t), \tag{4.28}$$

$$\Omega(x,i,t) = \max_{j \in (\Gamma(i)/\Lambda_x)} \left\{ \frac{H_{i,j,t}}{D_{i,j,t}} \middle| N_{i,j,t} \notin \Lambda_x \right., \qquad (4.29)$$

donde Γ(i) es el conjunto de vecinos del nodo *i* y Λ*x* es el conjunto de nodos previos visitados por la hormiga de búsqueda *x*. La tablas de éxitos *H*, de distancias *D* y de nodos *N* fueron explicados en la secciones anteriores. La función $\omega(x,i,t)$ determina cuál nodo, que es vecino del actual nodo *i* y que aun no ha sido visitado, ha producido previamente la mejor eficiencia en servir una consulta acerca de la palabra *t*, donde la eficiencia se mide por $\Omega(x,i,t)$.

4.5 Evaluación de la Eficiencia del Algoritmo Adaptativo AdaNAS

Los últimos experimentos propuestos tienen el propósito de probar las estrategias propuestas para el algoritmo adaptativo AdaNAS. Además de comprobar el cumplimiento de los objetivos de SQRP que son: encontrar el conjunto de rutas entre los nodos que lanzan las consultas y los nodos que contienen los recursos, tal que la cantidad de recursos encontrados sea maximizada y la cantidad de pasos usados para encontrar los recursos sea minimizada.

Desarrollo del Experimento

Los algoritmos AdaNAS y NAS recibirán la misma información para las experimentaciones, obtendrán los resultados y se compararán para verificar y comparar la eficiencia de los dos algoritmos. Las Tablas 4.14 y 4.15 muestran los valores con los que serán configurados los parámetros de NAS y AdaNAS. Las condiciones del experimento son las mismas que se usaron en la sección 4.2.2, donde se detalla la información de las instancias de topología, repositorios y consultas.

Tabla 4.14. Configuración de los parámetros del algoritmo NAS

Parámetros	**Definiciones**
$\rho = 0.07$	Factor de evaporación local de la feromona
$\alpha = 0.07$	Factor de evaporación global de la feromona
$\beta = 2$	Parámetro que define la intensificación de la feromona
$\tau_0 = 0.009$	Valor de inicialización de la tabla de feromonas
$ID\_HOP_0 = 999$	Valor inicial de la función de importancia de la distancia
$q_0 = 0.9$	Importancia relativa entre el exploración y explotación
maxResults =10	Número máximo de resultados recuperados
maxTTL = 10	Máximo tiempo de vida del agente
wh = 0.5	Factor de peso que controla la importancia relativa entre los recursos encontrados (*HIT_RATE*) y el tiempo de vida

a. Análisis de los Resultados

En este experimento, el rendimiento del algoritmo AdaNAS y NAS son analizados experimentalmente para determinar el rendimiento de cada uno de ellos y así, verificar que la función de ajuste para el control dinámico local guía el algoritmo adaptativo para minimizar la cantidad de saltos y maximizar la cantidad de éxitos.

Tabla 4.15. Configuración de los parámetros del algoritmo AdaNAS

Parámetros	**Definiciones**
$\rho = 0.07$	Factor de evaporación local de la feromona
$\tau_0 = 0.009$	Valor de inicialización de la tabla de feromonas
$\beta_1 = 2$	Parámetro que define la intensificación de la feromona en la regla de transición
$q_0 = 0.9$	Importancia relativa entre el exploración y explotación
$D\_HOP_0 = 999$	Valor inicial de la función de importancia de la distancia
$\beta_2 = 1$	Parámetro que define la intensificación de las medidas locales en la regla de transición
maxResults =10	Número máximo de resultados recuperados
maxTTL = 10	Máximo tiempo de vida del agente
$w_h = 0.5$	Factor de peso que controla la importancia relativa entre los recursos encontrados (*HIT_RATE*) y el tiempo de vida
$W_i = 1$	Influencia de la distancia en la regla de transición
$w_d = 1$	Influencia del grado en la regla de transición

Los resultados mostrados en las Figuras 4.26, 4.27 y 4.28 son un ejemplo de una instancia para mostrar el comportamiento de los algoritmos NAS y AdaNAS para las tres variables de respuesta: promedio de éxitos, promedio de pasos y eficiencia promedio.

La Figura 4.26 muestra el *rendimiento promedio de los éxitos* durante un conjunto de consultas con NAS y AdaNAS para una instancias creadas bajo las mismas condiciones de configuración descritas en la sección 4.2.2. Como se observa en las gráficas en general la versión del algoritmo AdaNAS siempre se mantiene por encima de NAS, es decir siempre obtiene más éxitos que NAS.

El algoritmo NAS, empieza con 11 éxitos por consulta y después de 25000 consultas termina teniendo un promedio de 13.9 éxitos por consulta. En esa misma figura también se observa que el algoritmo AdaNAS inicia aproximadamente en 16.1 éxitos por consulta y después de haber ejecutado las 25000 consultas alcanza un rendimiento promedio del porcentaje de éxitos de 20 éxitos por consulta.

Por otro lado en la Figura 4.27 se muestra el *rendimiento del promedio de pasos* realizados durante un conjunto de consultas con NAS y AdaNAS para una instancias creadas bajo las mismas condiciones de configuración descritas en la sección 4.2.2. Como se observa en las gráficas, en general la versión del algoritmo AdaNAS se mantiene durante las 25000 consultas con una menor cantidad de pasos promedio en comparación con el algoritmo NAS.

Los resultados del rendimiento del promedio de pasos durante 25000 consultas muestran que el algoritmo NAS empieza con 8.5 pasos por consulta y al final del conjunto de consultas el promedio de pasos por consulta es de 8. Para el algoritmo AdaNAS el promedio de pasos por consulta inicia en 6.9 pasos por consulta y al final su valor se reduce levemente en al rededor de 6.6 pasos por consulta.

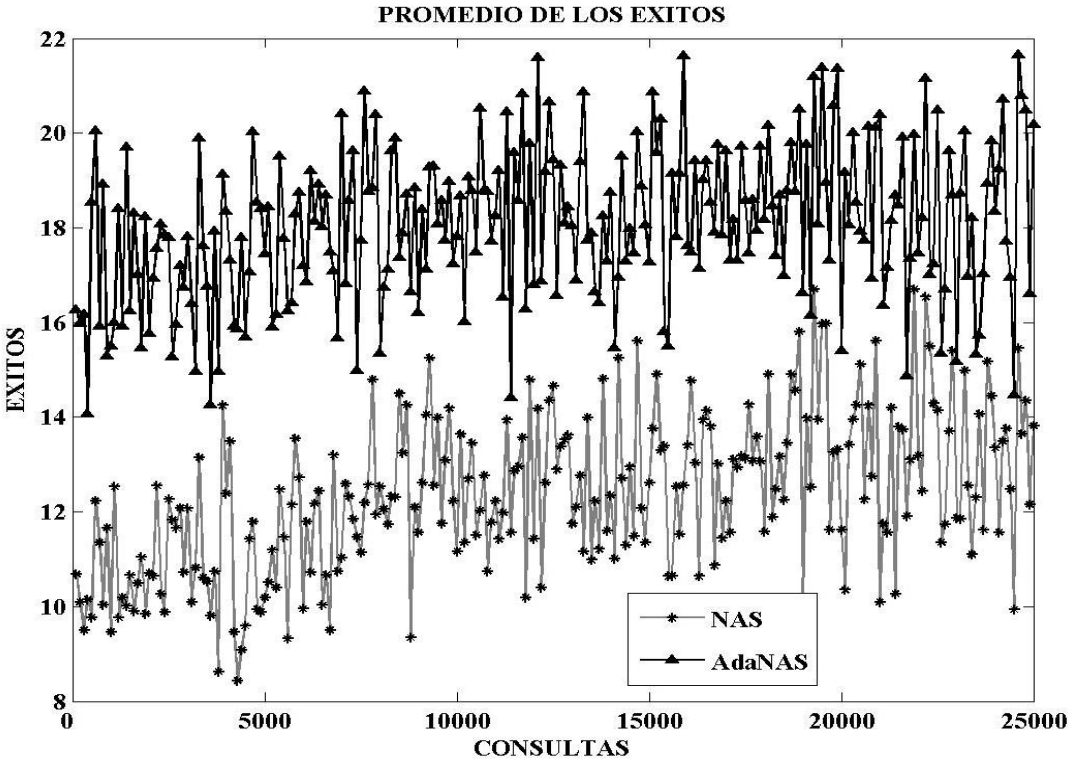


Figura 4.26 Comparación del promedio de éxitos de NAS contra AdaNAS.

Por otro lado, en la Figura 4.28 se muestra *la eficiencia promedio* durante un conjunto de conjunto de consultas con NAS y AdaNAS para una instancias creadas bajo las mismas condiciones de configuración descritas en la sección 4.2.2. Las gráficas muestran que en general el algoritmo adaptativo AdaNAS siempre mantiene en promedio una mejor eficiencia al ser comparado con el algoritmo NAS. Lo anterior se observa ya que el algoritmo NAS inicia con un rendimiento de 1.2 éxitos por paso y después de 25000 consultas termina con un rendimiento de 1.39 éxitos por paso. Los resultados para el algoritmo AdaNAS muestran que el algoritmo inicia con una eficiencia promedio de 2.1 éxitos por salto y termina con 2.6 éxitos por salto.

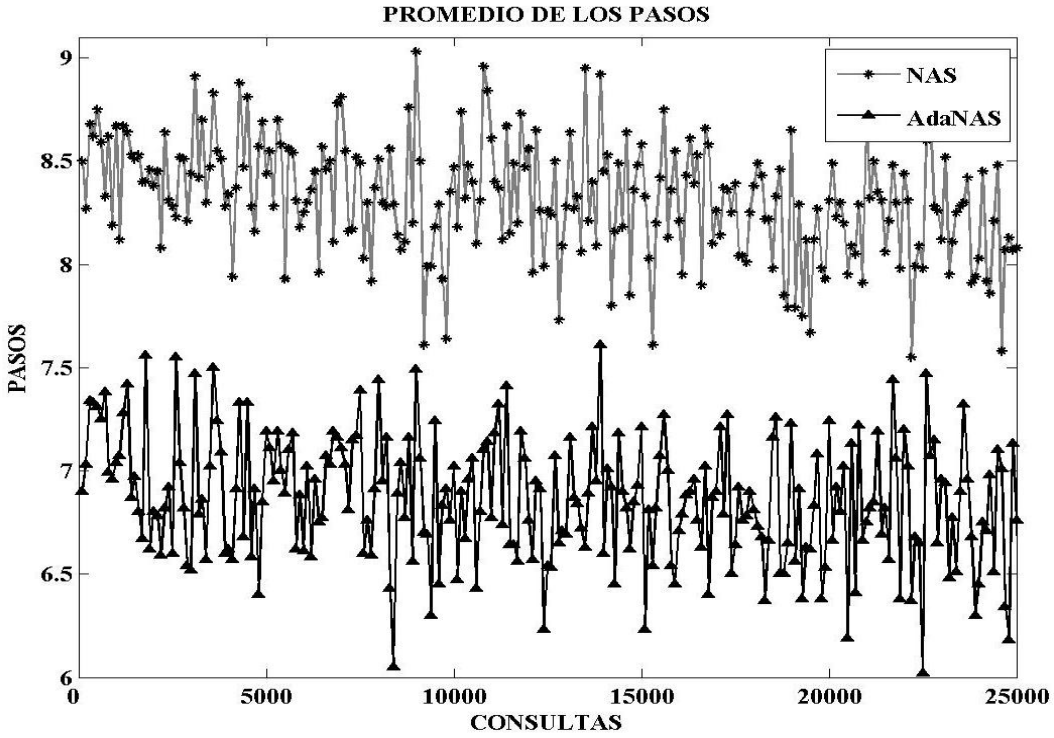


Figura 4.27 Comparación del promedio de los pasos de NAS contra AdaNAS.

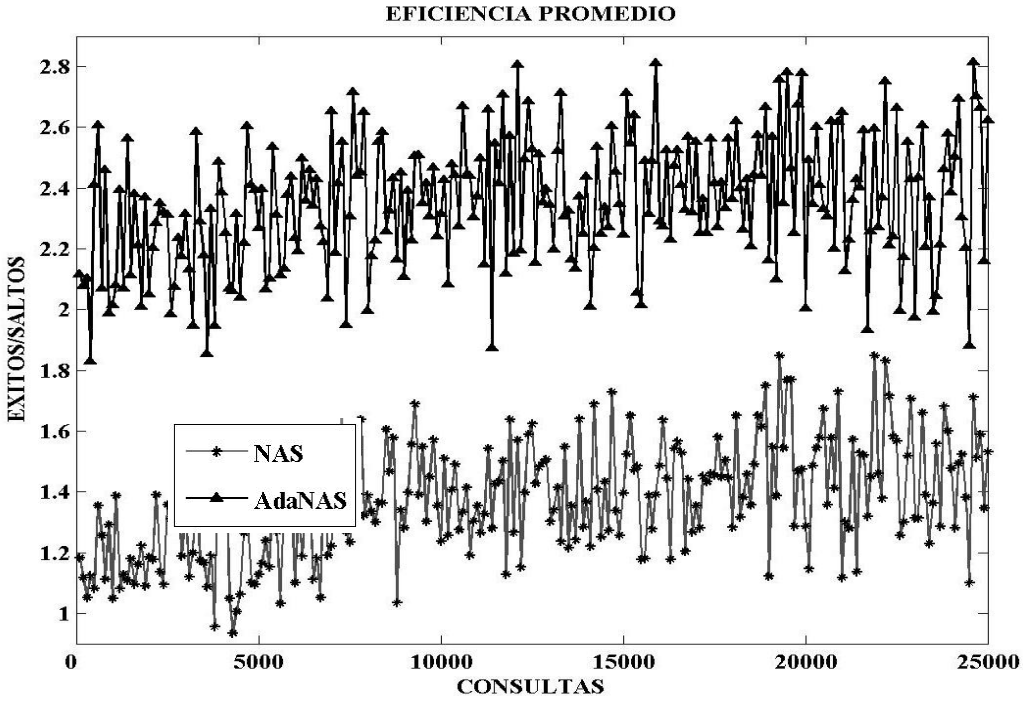


Figura 4.28 Resultados de la eficiencia Promedio para NAS y AdaNAS.

Los resultados revelan que se el algoritmo AdaNAS cumple con el objetivo planteado en la investigación. Ya que se reduce la cantidad de pasos o saltos e incrementa la cantidad de éxitos con respecto a NAS, y esto también se refleja en la eficiencia promedio del algoritmo AdaNAS.

La estrategia adaptativa AdaNAS muestra un incremento del 43.8% sobre los recursos encontrados con NAS, en la contribución en la reducción del número de pasos promedio en 21.2%, dando una eficiencia aproximada del 87% sobre la eficiencia promedio de NAS. Estas observaciones sugieren que el cambio de estrategia de ajuste manual a ajuste adaptativo en combinación con el Lookahead y el grado contribuyen a que el algoritmo adaptativo obtenga mejores resultados.

Tabla 4.16. Resultados del algoritmo NAS, aplicado a 10 instancias

Instancias	NAS			
	Exitos Promedio	**Saltos Promedio**	**Eficiencia Promedio**	**Varianza**
BA1_0	9.130	8.763	1.043	0.044
BA1_1	8.623	8.785	0.990	0.060
BA1_2	4.165	9.770	0.426	0.011
BA1_3	10.331	8.632	1.198	0.60
BA1_4	8.400	8.660	0.973	0.042
BA1_5	4.511	9.691	0.466	0.007
BA1_6	5.444	9.140	0.596	0.026
BA1_7	11.356	8.497	1.348	0.032
BA1_8	4.843	9.339	0.520	0.028
BA1_9	12.626	8.177	1.567	0.056

Tabla 4.17. Resultados del algoritmo AdaNAS, aplicado a 10 instancias

Instancias	AdaNAS			
	Exitos Promedio	**Saltos Promedio**	**Eficiencia Promedio**	**Varianza**
BA1_0	15.341	7.824	2.073	0.064
BA1_1	7.811	8.219	2.077	0.060
BA1_2	4.284	10.240	0.973	0.081
BA1_3	8.577	8.114	2.435	0.071
BA1_4	14.032	8.396	1.783	0.069
BA1_5	12.183	10.008	0.968	0.057
BA1_6	9.090	8.763	1.294	0.064
BA1_7	17.344	7.837	2.537	0.053
BA1_8	11.493	9.220	1.150	0.095
BA1_9	14.616	7.026	3.019	0.064

Después de haber mostrado los resultados en forma gráfica para una instancia. Se muestra una tabla con los resultados de 10 diferentes instancias SQRP son resueltas con NAS y AdaNAS sobre 20 corridas para cada instancia. La Tabla 4.16, muestra los resultados para el algoritmo NAS, la primera columna identifica la instancia, la segunda y tercera los éxitos y saltos promedio respectivamente. La cuarta muestra la eficiencia promedio y la última columna muestra la

varianza. Esta misma distribución se usa para mostrar los resultados del algoritmo AdaNAS, en la Tabla 4.17.

En las Tablas 4.16 y 4.17 se observa que la varianza de los dos algoritmos es pequeña, es decir, para NAS la varianza está entre 0.007 y 0.060 y para AdaNAS la varianza está entre 0.053 y 0.095. Estos resultados garantizan que los algoritmos obtienen valores cercanos a la eficiencia promedio.

Tabla 4.18. Resultados del % de mejora del promedio de la eficiencia, al comparar los algoritmos NAS y AdaNAS

Instancias	**Eficiencia Promedio de NAS**	**Eficiencia Promedio de AdaNAS**	**% de Mejora del Promedio de la Eficiencia**
BA1_0	1.043	2.073	50.313
BA1_1	0.990	2.077	47.664
BA1_2	0.426	0.973	43.782
BA1_3	1.198	2.435	49.199
BA1_4	0.973	1.783	54.570
BA1_5	0.466	0.968	48.140
BA1_6	0.596	1.294	46.058
BA1_7	1.348	2.537	53.133
BA1_8	0.520	1.150	45.217
BA1_9	1.567	3.019	51.904

La Tabla 4.18 muestra los resultados correspondientes a el porcentaje de mejora del promedio de la eficiencia, al comparar los algoritmos NAS y AdaNAS. la primera columna identifica la eficiencia promedio para NAS, la segunda columna identifica la eficiencia promedio para AdaNAS y en la última columna muestra el porcentaje de mejora del promedio de la eficiencia. Encontrándose que en todas las corridas el algoritmo AdaNAS supera a el algoritmo NAS. El porcentaje de mejora del algoritmo AdaNAS esta entre 43.7% y 54.5%.

Por último se muestra en la Figuras 4.29 y 4.30 los resultados de las 90 instancias ejecutadas, con las que se experimentaron. En estas gráficas se puede observar que en todos los casos el algoritmo AdaNAS supera al algoritmo NAS al comparar la eficiencia promedio de los dos algoritmo.

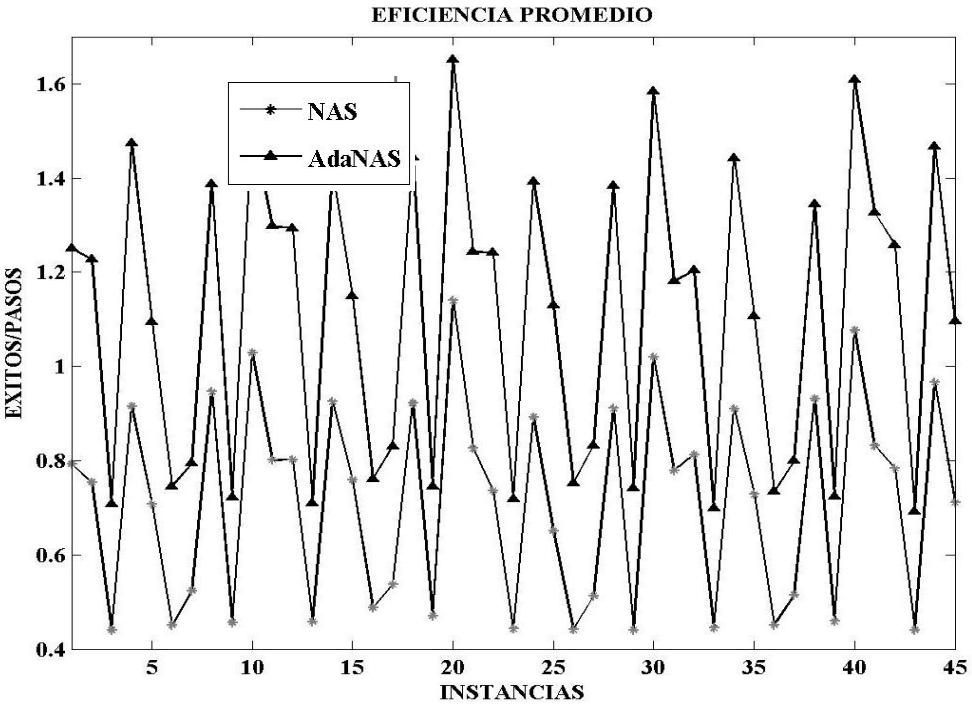


Figura 4.29 Resultado de la eficiencia promedio para NAS y AdaNAS, con 45 instancias.

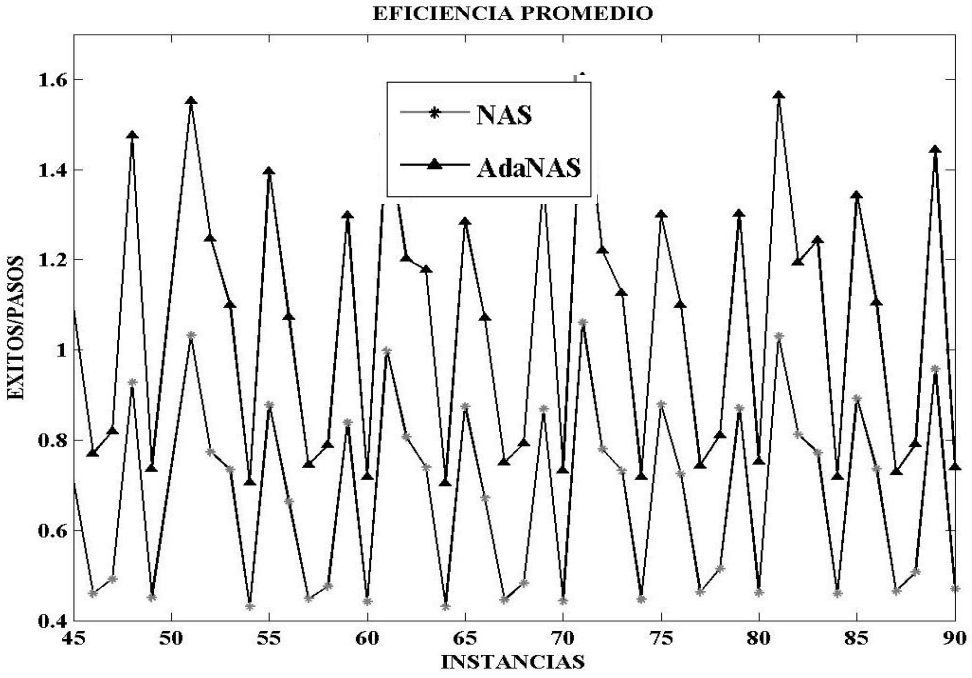


Figura 4.30 Resultado de la eficiencia Promedio para NAS y AdaNAS, con 45 instancias (45-90).

4.5.1 Diferencias Significativas: Prueba de Wilcoxon

Para comprobar que las diferencias de los resultados de los dos algoritmos NAS y AdaNAS, son significativas se aplicaron pruebas de hipótesis estadísticas no paramétricas de Wilcoxon. Las pruebas paramétricas necesitan cumplir con un conjunto de requisitos como son: la independencia de las muestras, la normalidad de los datos y la heterocedasticidad de las muestras es decir que las varianzas sean diferentes. Para el caso de las pruebas paramétricas esto no es necesario [102,103,104].

Se aplica la prueba de Wilcoxon a los datos correspondientes al rendimiento de los dos algoritmos NAS y AdaNAS, los cuales serán identificados en esta prueba como D_1 y D_2 respectivamente. Se desea saber si el rendimiento de D_1 es mejor que el rendimiento de D_2.

El procedimiento para aplicar la prueba de Wilcoxon con muestras correlacionadas es:

- Primero se calcula la diferencia entre las dos muestras, es decir $D_1 - D_2$
- Se calcula el valor absoluto de la diferencia obtenida en el paso anterior $|D_1 - D_2|$
- Se ordenan los datos con relación al valor absoluto
- Los valores en cero se eliminan
- Se ordenan las diferencias absolutas, de menor a mayor
- Se asigna un rango creciente. Si hay diferencias iguales se les asignará el valor promedio de sus diferencias
- Por último, a los rangos se les asigna el signo de la diferencia entre las muestras $D_1 - D_2$, y se realiza una suma de estos últimos valores.

S es la cantidad de registros con rango positivo y RS es la cantidad de registros con rango negativo.

Se establecen los supuestos bajo los cuales se harán las pruebas de significancia:

$H_{0:}\ \mu = \mu_0$ (*igual rendimiento*)
$H_1: \mu \neq \mu_0$ (*diferente rendimiento*)

Donde μ_0 es el valor promedio del rendimiento para los dos algoritmos se desea analizar.

Se determina S y RS para 40 instancias.

Como en todas las instancias el algoritmo AdaNAS fue mejor que el algoritmo NAS, los rangos positivos se desprecian y todas las diferencias son positivas, por lo que se obtiene el siguiente resultado.

|S - RS| = Suma de rangos positivos - Suma de rangos negativos = |0 - (-2047) | = 2047

Rechazar la hipótesis nula si (|S-RS| >= valor de tablas) = 2047 >= 264 Falso

El resultado anterior se compara contra el valor de tablas para el 95% de confiabilidad y n=40.

Se rechaza la hipótesis nula H_0 de que el rendimiento sea igual y **se acepta** hipótesis alternativa H_1 , hay diferencias significativas en el rendimiento, con una confiabilidad del 95%. Esto significa que las diferencias entre los promedios de la eficiencia son significativas.

CONCLUSIONES Y TRABAJOS FUTUROS

En esta sección se presentan las conclusiones de este trabajo, así como también sugerencias para el desarrollo del trabajo futuros en esta área de conocimiento.

Conclusiones

En Este trabajo se presenta un enfoque innovador en la solución del problema del ajuste de parámetros a través de estrategias de control adaptativas, en particular al aplicarlas al parámetro tiempo de vida (TTL) de los algoritmos de colonias de hormigas AdaNAS.

De acuerdo con la literatura especializada, se han propuesto algunas estrategias para el ajuste de parámetros. Estos han dividido el problema de ajuste de parámetros en estrategias: de afinación (global) y de control (local). Encontrándose con que se identificaron pocos trabajos que aborden el problema de ajuste de parámetros bajo las condiciones aquí presentadas. Específicamente la estrategia de ajuste de control adaptativo propuesta, muestra que ***es factible resolver de manera eficiente*** *el problema de direccionamiento de consultas semánticas (SQRP) en la Internet, modelado como un sistema complejo, controlando adaptivamente el parámetro TTL del algoritmo AdaNAS, mediante el uso de información* ***local*** *derivada de:*

- *La configuración inicial,*
- *El desempeño de consultas anteriores,*
- *Las condiciones topológicas heterogéneas.*

El algoritmo adaptativo AdaNAS al ser comparado con el algoritmo NAS que usa estrategias de ajuste global obtuvo una mejora de:

- Para la eficiencia promedio, la mejora se encuentra en un rango que va desde un 43.7% a un 54.5% éxitos por salto,
- Para el promedio de los éxitos por consulta, obtuvo una mejora del 43.8% en la cantidad de éxitos, y
- Para la cantidad de saltos, obtuvo una mejora del 21.2% saltos por consulta.

Concluyéndose que la versión adaptativa de AdaNAS alcanza mejores resultados que la versión con ajuste global NAS. Estos resultados fueron validados mediante pruebas estadísticas no paramétricas de Wilcoxon, con una confiabilidad del 95%.

Dentro de las versiones desarrolladas con ajuste global como son SemAnt, RandomWalk y NAS, al ser comparadas, se demostró que el algoritmo NAS es el que obtuvo un mayor eficiencia promedio, esto fue reportado en la publicación " A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks".

Aportaciones de la Investigación

Las principales contribuciones de la investigación al abordar el problema del ajuste de parámetros usando estrategias de control adaptativas con el parámetro TTL, en algoritmos de colonias de hormigas, para dar solución al problema de *direccionamiento de consultas semánticas* en la Internet, modelado como un sistema complejo son:

El desarrollo la demostración de NP-completitud de SQRP, demostrando que el problema de decisión es NP-completo, por reducibilidad desde el problema de la Mochila

Se analizó un conjunto de nueve métricas topológicas con el objetivo de encontrar el conjunto mínimo de métricas que caracterizaran las diferentes topologías de redes complejas de manera cuantitativa. Encontrándose que la métrica coeficiente de la distribución del grado obtuvo el 97.78% de efectividad para discriminar entre las diferentes topologías. Con la métrica Coeficiente de Distribución del Grado (DDC), se caracterizó la estructura de redes reales para identificando características que después se validarán en redes artificiales. Para el caso de las redes reales como la Internet, se encontró que la distribución del grado de la topología es scale-free y que se ha mantenido a través del tiempo. Además de que dicha estructura es no uniforme ya que cambia de un punto a otro punto de la red.

Se diseño y desarrolló la primera versión del algoritmo NAS que resuelve SQRP, basado en los metaheurísticos Ant Colony System y SemAnt. Usando estrategias de ajuste global de parámetros, mediante recomendaciones de la literatura, para controlar y extender el TTL de los agentes hormiga. Al algoritmo NAS con ajuste global de parámetros, *se le incorporó la métrica topológica DDC en la función de transición*, para que los agentes hormiga seleccionen el siguiente nodo y guiarlas hacia nodos mejor conectados. Además de la métrica topológica DDC *se le incorporó un mecanismo clásico de exploración local llamado Lookahead*, esté método proporciona información de los vecinos de primer nivel. Cabe señalar que las dos estrategias aportan información local a la búsqueda. Otra *innovación* incluida en la primera versión con ajuste global de NAS, son las tres *funciones de aprendizaje histórico* las cuales fueron creadas y desarrolladas para efectuar un registro del rendimiento histórico del algoritmo NAS en consultas pasadas a través de: la importancia de los éxitos (HIT), la importancia del tiempo de vida (ITL_HOP) y la importancia de la distancia (ID_HOP).

Con toda la información antes descrita se obtuvo una primera versión del algoritmo NAS que fue comparada con el algoritmo clásico de búsqueda Random-Walk y el algoritmo SemAnt. Encontrándose que esta primera versión supera a los otros dos algoritmos. Específicamente el algoritmo NAS muestra una mejoría de seis veces mas eficiencia, que el algoritmo SemAnt y mas de diez veces de mejoría que el algoritmo Random-Walk.

Se *analizaron las características de SQRP*, encontrándose que la distribución topológica tiene una gran influencia para las dos variables de respuesta: el promedio del número de recursos encontrados por cada agente hormiga de búsqueda, para las consultas y el promedio del número de pasos que es la cantidad de enlaces viajados por el agente hormiga de búsqueda, además se encontró que las distribuciones de probabilidad de las consultas, y los repositorios están fuertemente relacionadas y son significativas para ambas variables de respuesta (descritas en el párrafo anterior). Estos resultados sirvieron para seleccionar la distribución de probabilidad que tenga mayor efecto sobre el rendimiento del algoritmo NAS. Estableciéndose las condiciones para trabajo en: la topología con una distribución del grado power-law (red scale-free), para las consultas con una distribución del grado aleatoria y para los repositorios con una distribución del grado power-law.

Análisis de los parámetros significantes del algoritmo NAS, para esto se seleccionaron 9 parámetros, de los cuales los que tuvieron mayor efecto fueron: W_d(Influencia del grado en la regla de transición), β_1(define la intensificación de la feromona en la regla de transición), ρ(factor de evaporación local de la feromona) y q(importancia relativa entre exploración y explotación).

Para el modelado de las relaciones entre los parámetros del algoritmo NAS y las características de SQRP, se rediseñó la regla de transición al siguiente nodo. Encontrándose mediante experimentación que la métrica de caracterización local *grado del nodo* k_i, obtuvo en promedio una mejoría del 21.7% éxitos por paso al guiar él algoritmo NAS hacia nodos mejor conectados, es decir con un alto grado.

Se diseño y construyó la nueva versión adaptativa llamada AdaNAS, la cual incluye la función de ajuste para el control adaptativo para TTL.

Trabajos Futuros

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

a) Llevar a cabo el estudio de las características de SQRP con el objetivo de analizar la factibilidad de incrementar la cantidad de hormigas por consulta. Tomando en cuenta factores como son: el balanceo de cargas, la capacidad de los medios de transmisión, entre otros.

b) Implementar de manera dinámica: las conexiones y desconexiones de los nodos a la red, así como cambios en los datos de los repositorios. De manera que el modelo de consultas de SQRP se acerque mas a la realidad.

c) Diseñar una estrategia de administración de la memoria que permita evaluar instancias más grandes.

d) Usando el algoritmo AdaNAS se podrían analizar otros parámetros además de TTL, con el objetivo de mejorar el rendimiento del algoritmo AdaNAS. Caracterizando los casos en los cuales cada uno de los parámetros identificados afectan el rendimiento. La información integrarla en un hiperheurístico el cual pueda tener varias heurísticas para dar solución a los casos identificados.

e) Hacer un estudio para la selección de métricas que caractericen instancias de repositorios y consultas reales, con el objetivo de asegurar la validez y representatividad de instancias artificiales.

Anexo A

REDES COMPLEJAS

Uno de los factores importantes para tener éxito en la búsqueda de recursos en la Internet, es necesario tener un modelo que represente adecuadamente el ambiente donde se llevará acabo esta actividad. Este capítulo tiene como finalidad introducir las definiciones y terminología relacionada con los modelos de representación de la estructura o topología, la búsqueda de recursos, redes complejas y el problema del direccionamiento de consultas semánticas.

Definiciones de Grafos

Un *grafo* se define como $G = (N, E)$, donde *N* es el conjunto de nodos y *E* es el conjunto de aristas. Una representaciones muy usada es la *matriz de adyacencia M,* la cual usa una matriz cuadrada $n \times n$, donde *n* representa el número de nodos. Cada elemento de $M(i, j)$ representa la presencia o ausencia de una arista entre los nodos *i* y *j*. Si existe una arista entre los nodos *i* y *j* entonces $M(i, j) = 1$, en caso contrario $M(i, j) = 0$ [45,105].

El número de nodos en el conjunto *N* es denominado *orden de un grafo n =| N |*, y el número total de aristas en el conjunto *E* es llamando *tamaño del grafo e =* | *E* |. El tamaño del grafo puede ser como mínimo 0 y como máximo $\frac{n(n-1)}{2}$ [35,41,105].

Dos nodos son *adyacentes* si una arista los une. El número de aristas adyacentes a un nodo es el *grado* del nodo *i* y se define por la Ecuación (1):

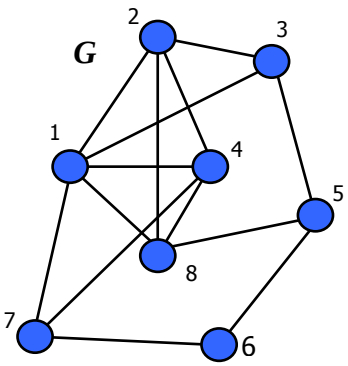
$$k_i = \sum_{j=1}^{n} M(i, j), \tag{1}$$

en un grafo *G* el *grado máximo* se expresa como $\Delta(G)$, el *grado mínimo* $\delta(G)$ y el *grado promedio* $\langle k \rangle$ es calculado con la Ecuación (2):

$$\langle k \rangle = \frac{2e}{n} \tag{2}$$

Los nodos adyacentes a un nodo *i* del grafo *G* se denominan *conjunto vecino* y se denota por $\Gamma(i)$ [41,45,55]. Si dos nodos *i, j* no son adyacentes, pueden estar conectados por una secuencia de *m* aristas, al conjunto de aristas que forma la secuencia se le denomina *ruta* entre *i* y *j* siendo *m* la longitud de la ruta [45].

Un *grafo* es una abstracción matemática de situaciones donde se pueden representar elementos y sus relaciones [50]. En la Figura 1 se muestra un ejemplo de un grafo, su matriz de adyacencia correspondiente, así como el calculo de sus propiedades.



M

	1	2	3	4	5	6	7	8
1	0	1	1	1	0	0	1	1
2	1	0	1	1	0	0	0	1
3	1	1	0	0	1	0	0	0
4	1	1	0	0	0	0	1	1
5	0	0	1	0	0	1	0	1
6	0	0	0	0	1	0	0	1
7	1	0	0	1	0	0	0	0
8	1	1	0	1	1	1	0	0

$V = \{1,2,3,4,5,6,7,8\}$

$E = \{(1,2),(1,3),(1,4),(1,7),(1,8),(2,3),(2,4),(2,8),(3,5),(4,7),(4,8),(5,6),(5,8),(6,7)\}$

$G(n) = 8$ $\quad G(e) = 14$ $\quad \langle k \rangle = (2*14)/8 = 3.5$

$k(1) = 5$ $\quad \Gamma(1) = \{2,3,4,7,8\}$

$\Delta(G) = 5$ $\quad \delta(G) = 2$

Figura 1. Ejemplo de grafo, matriz de adyacencia y propiedades.

Redes Complejas

Un *sistema* es un conjunto de componentes relacionados entre si con un objetivo común. Amaral en su artículo [10] propone una clasificación de los sistemas, clasificándolos en simples, complicados y complejos. Los *sistemas simples* están formados por un número pequeño de componentes, los cuales actúan de acuerdo a leyes bien comprendidas. Los *sistemas complicados* tienen un gran número de componentes los cuales tienen roles bien definidos y están gobernados por reglas bien comprendidas. Por otro lado los *sistemas complejos* tienen comúnmente 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, es decir la conectividad de los componentes y sus roles son variables [10,40].

En muchos estudios los sistemas complejos son modelados de manera abstracta como grafos [41] siendo así como nace el término de *red compleja* ó *no uniforme.* Las redes complejas comenzaron a ser estudiadas hacia finales de la década de los 50's principios de los 60's, mediante la teoría de grafos aleatorios, en los cuales las aristas se distribuyen aleatoriamente [46].

Después de estar estudiando las redes complejas se encontró que había características que la teoría de grafos no podía modelar, como por ejemplo la dinámica y la estabilidad de las grandes redes complejas. Debido a esto se incluyeron otras áreas como la física estadística, la cual trata de realizar predicciones estadísticas acerca del comportamiento colectivo de los componentes. Recientemente, se ha percibido que muchos sistemas formados por número muy grande de pequeños elementos, obedecen leyes universales independientemente de los detalles microscópicos [54].

Y es así como la teoría de grafos y la física estadística se combinan para proveer las bases necesarias que permitan el estudio de las redes complejas a través de la *Teoría de Redes Complejas* [47], la cual se enfoca en [105]:

1. Encontrar y destacar las propiedades estadísticas que caracterizan la estructura y el comportamiento de la red compleja, ofreciendo herramientas para medir sus propiedades.
2. Crear modelos de redes que ayuden a entender el significado de esas propiedades.
3. Predecir el comportamiento de la red compleja basándose en el comportamiento de las propiedades estadísticas.

Funciones de Caracterización: Local y Global

Las redes crecen y evolucionan por eventos locales, tales como la adición de nodos y enlaces ó la eliminación de enlaces entre nodos. Cada vez que se genera un evento local la estructura de la red cambia. Una forma de medir estos cambios es a través de las funciones de caracterización ya que tienen como objetivo proveer información de la red en términos cuantitativos. Y así de esta forma es posible caracterizar y analizar las redes complejas. La muestra una red *G* a la cual se le extraes sus características y son guardadas en un vector de características $\vec{f}_G$, para posteriormente llevar acabo el análisis, como se muestra en la Figura 2 [47].

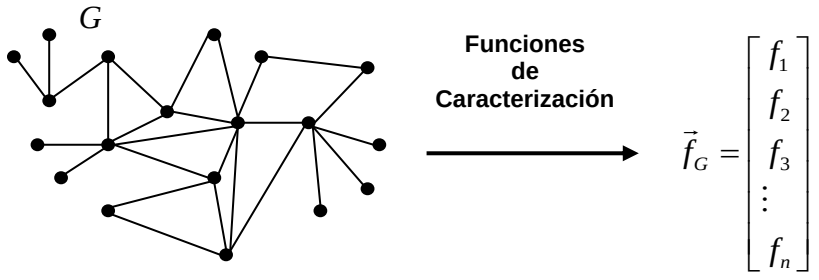


Figura 2. Extracción de características topológicas mediante funciones de caracterización

Las funciones de caracterización pueden clasificarse en dos tipos:

- Basadas en *información global*: requieren información de toda la red *G* para poder calcularlas.
- Basadas en *información local*: Considerando que cada nodo *i* en la red *G*, tiene asociado un subgrafo G_i, el cual esta formado por el nodo *i* su conjunto de vecinos $\Gamma(i)$. Una función de caracterización local es una métrica que se calcula para cada G_i en *G*.

Tipos de Redes Complejas

Para su estudio las redes complejas, se han caracterizado a través de sus propiedades dentro de las cuales se pueden mencionar: su gran *tamaño* ya que están formadas por un gran número de elementos. Cantidad de *aristas* (esparcidas), es decir tienen pocas aristas en comparación con el gran número de nodos *n*, en general el número de aristas es más cercano a *n* que al número máximo de aristas que pueden existir. Forman *grupos*, las aristas en el grafo no están distribuidas uniformemente; y tienen una secuencia de grado conocida como *distribución del grado* , esta propiedad describe el patrón de conexión de los nodos [50].

De acuerdo con la distribución del grado P(*k*), las redes complejas se clasifican en redes: Aleatorias, Scale-free (power-law) y Sigmoidales [5][23].

Redes Aleatorias

Dado que en una red aleatoria las aristas entre los nodos son establecidas aleatoriamente, la mayoría de los nodos tiene aproximadamente el mismo grado, cercano al grado promedio de la red $\langle k \rangle$.

Redes Power-Law

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 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 resto de los nodos tienen pocos enlaces (un grado pequeño) [41,42]. 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 [55].

Redes Sigmoidales

Este modelo fue propuesto en el 2009 por los investigadores Rogelio Ortega y Eustorgio Meza en [16] ellos proponen este modelo de crecimiento que incorpora la unión preferencial por los nodos de grado pequeño a través del parámetro α. Este modelo es una variación del modelo de Zonghua et. al [56] el cual exhibe una distribución del grado Sigmoidal, mostrada en la Ecuación (3):

$$\Pi(i) = \frac{\left[(1-p)k_i^{\alpha} + p\right]}{\sum_{\forall j \in N}\left[(1-p)k_j^{\alpha} + p\right]} \tag{3}$$

donde k_i es el grado del nodo i ya presente en la red $G = (N,E)$, p es la probabilidad de que un nodo se conecte aleatoriamente, N es el conjunto de nodos en G, k_j es el grado del nodo $j \in N$ y α=1 ó -1 es un parámetro que permite elegir entre la unión preferencial por los nodos de mayor grado (proporcional a k_i) o la unión preferencial por los nodos de menor grado (inversamente proporcional a k_i).

Si α=1 y $0 < p < 1$, la Ecuación (4) se reduce a la ecuación propuesta por Zonghua et. al. [56].

$$\Pi(i) = \frac{\left[(1-p)k_i + p\right]}{\sum_{\forall j \in N}\left[(1-p)k_j + p\right]} \tag{4}$$

Cuando en la Ecuación (A.4) , $0 < p < 1$, la unión preferencial es considerada entre preferencial por los nodos de mayor grado y aleatoria, produciendo redes con una distribución de grado entre power-law y exponencial [56].

Si α=1 y p =0, la Ecuación (5), se reduce a la ecuación propuesta por Barabási y Albert [46] que fomenta la unión preferencial por los nodos de mayor grado,

$$\Pi(i) = \frac{k_i}{\sum_{\forall j \in N} k_j} \tag{5}$$

produciendo redes con una distribución del grado power-law.

Si α={1, -1} y p = 1, la Ecuación (6) se reduce a la ecuación :

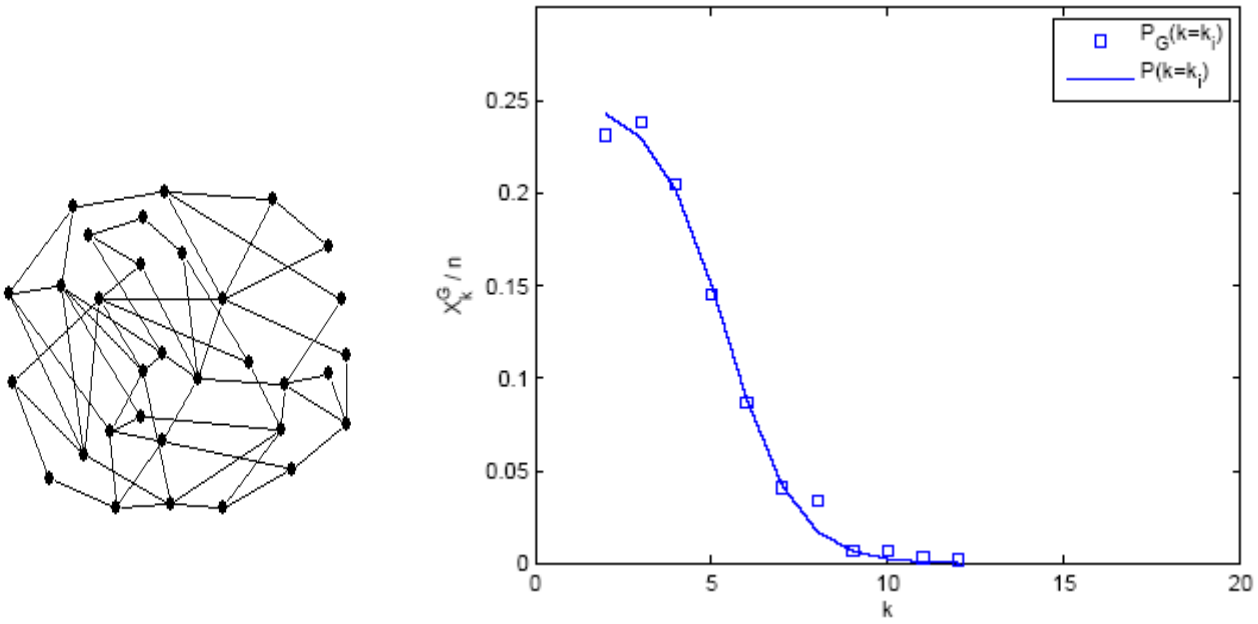
$$\Pi(i) = \frac{1}{\sum_{\forall j \in N} 1} = \frac{1}{n} \tag{6}$$

que fomenta la preferencia por la unión aleatoria, donde un nodo nuevo se conecta a m nodos elegidos aleatoriamente [56]. La unión aleatoria produce redes con una distribución del grado exponencial.

Finalmente cuando α= -1 y p = 0, la Ecuación (7) se reduce a la unión preferencial por los nodos de menor grado,

$$\Pi(i) = \frac{\frac{1}{k_i}}{\sum_{\forall j \in N} \frac{1}{k_j}} \tag{7}$$

donde un nodo nuevo se conecta a *m* nodos con grado pequeño, cuando $0 < p < 1$ la unión preferencial es considerada entre aleatoria y con preferencia por los nodos de grado pequeño. La distribución del grado de las redes creadas con esta regla esta entre exponencial y lo que es claramente una nueva distribución del grado, como se puede ver en la Figura (3).



a) b)

Figura 3. Red sigmoidal: a) estructura topológica, b) gráfica de la distribución del grado.

Modelos de Generación

Son una herramienta importante a través de los cuales se reproducen las características topológicas de las redes del mundo real. El estudio de estos modelos nace con la finalidad de analizar y comprender funciones y fenómenos que se llevan a cabo en las redes complejas especialmente las redes naturales. Estos modelos son los correspondientes a los tipos de redes descritas en la sección 1.2.1.

Modelo de Erdös y Rényi (MEyR)

El modelo de Erdös y Rényi [41,55] es un modelo sin crecimiento. Esto significa que el número de nodos *n* permanece constante durante la construcción del grafo, y la agregación de aristas entre cualquier par de nodos sigue una distribución de probabilidad.

La construcción de grafos aleatorios es llamada: evolución. Comenzando con un conjunto de *n* vértices aislados, el grafo se desarrolla agregando sucesivamente aristas aleatorias (ver Figura 4). Los grafos obtenidos en las diferentes etapas de este proceso corresponden a la mayor probabilidad de conexión *p,* eventualmente obtendremos un grafo totalmente conexo para *p=1* [35].

Por tanto el número total de enlaces es un variable aleatoria con un valor esperado de $p[n(n-1)/2]$. Su diámetro de la red y coeficiente de agrupación son muy pequeños y cada nodo se relaciona con aproximadamente la misma cantidad de nodos, de tal manera que si se elimina algún nodo de la red el daño es mínimo.

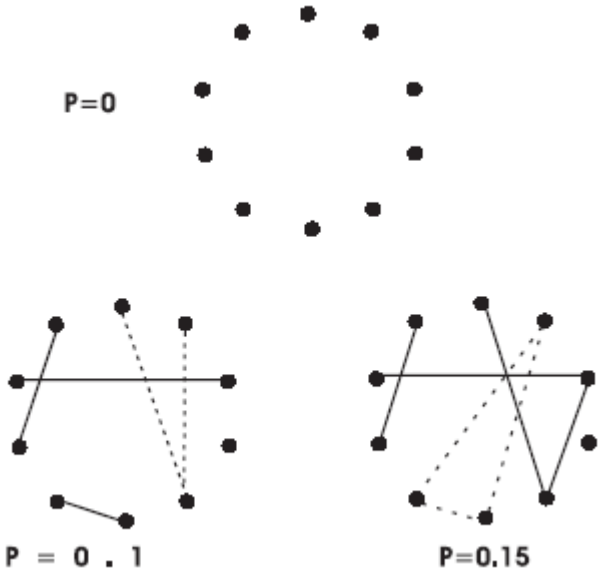


Figura 4. Proceso de evolución de grafos con el modelo Erdos-Rényi. Con 2 diferentes estados del desarrollo de los grafos, correspondientes a *p=0.1* y *p=0.15*.

Modelo Barabási-Albert (MBA)

Después de haber estudiado el comportamiento de muchas redes reales hacia 1999, se introduce el modelo Barabási-Albert [46] inspirado en el crecimiento y el enlace preferencial, este fue el primer modelo que reprodujo redes con una distribución del grado Power-Law. El modelo considera dos elementos:

- Crecimiento: comienza, con un pequeño número de nodos m_0, a cada paso *t* se añade un nuevo nodo con $m \leq m_0$ aristas que enlacen al nuevo nodo con m diferentes nodos ya existentes en el sistema.
- Enlace preferencial: cuando se selecciona los nodos a los cuales un nuevo nodo se va a conectar, se asume que la probabilidad Π de que un nuevo nodo sea conectado al nodo *i* depende del grado k_i del nodo *i.* Mostrado en la Ecuación (8) :

$$\Pi(k_i) = \frac{k_i}{\sum_{j \in N} k_j} \qquad (8)$$

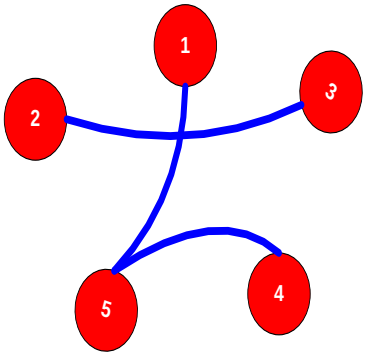
donde *n* es el conjunto de nodos en la red. Después de *t* lapsos de tiempo, el procedimiento resulta en una red con $n = t + m_0$ nodos con *mt* aristas. Ejemplo de generación de un grafo usando el modelo de Barabási-Albert. Para *mo* = 5, *m* = 2, *t* = 2.

1. Para *t* = 0 el modelo inicia con m_0 nodos iniciales, cada nodo con un enlace por lo menos (configuración inicial) representado en b) de la Figura 5. En el inciso a) Cálculo del grado de cada nodo, b) Configuración inicial modelo Barabási, c) Cálculo de la lista de probabilidades ordenada, para el enlace preferencial, d) Sumatoria de los grados.

a)

Nodo *i*	1	2	3	4	5
Grado k_i	1	1	1	1	2

b)



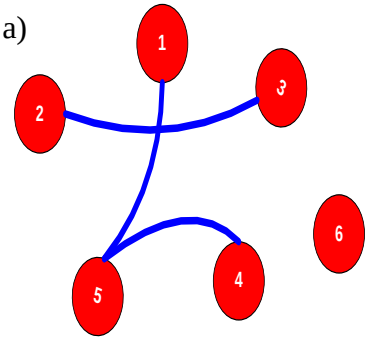
c)

Nodo *i*	1	2	3	4	5
Grado Π_i	0.166	0.332	0.498	0.664	1

d) $\sum k_i = 6$

2. Para *t*=1, se agrega un nodo nuevo a la red. Mostrado en a) la Figura 6, para determinar los enlaces del nuevo nodo con los nodos ya existentes en la red, se tiene que proceder por construir la lista de probabilidades para este tiempo (Figura 6 b).).

a)



b)

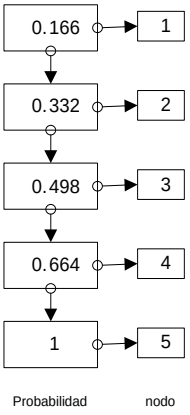


Figura 6.a) Grafo con el nuevo nodo agregado 6,b) Lista de probabilidades del grafo.

3. Para poder conectar el nodo 6 con algún nodo ya existente en el grafo, se procede a generar un número aleatorio, en este caso *Na*=0.331, es decir el *m* enlace del nodo 6 es con el nodo que entre en el rango del 0.167 al 0.332, por lo tanto el 6 se conecta con el nodo 2 del grafo. Esto se muestra en la Figura 7.

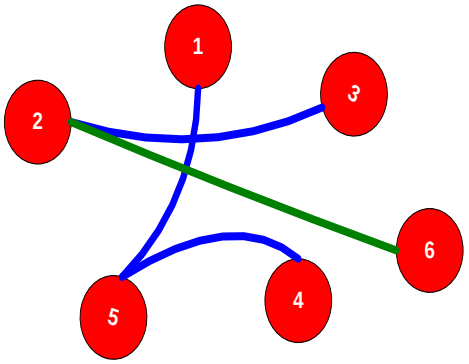


Figura 7. Conexión del nodo 6 con 2, mediante el enlace preferencial.

4. Para cada tiempo *t* se tienen que cumplir con *m* enlaces, es decir al nodo 6 le hace falta conectarse con otro nodo ya existente en la red, utilizando nuevamente la lista de probabilidades en b) de la Figura 6. Para esto se tiene que generar otro numero aleatorio *Na* = 0.099, el cual el rango probabilidad esta de 0.166 hasta 1, es decir el segundo y último enlace en este tiempo es el nodo 6 se conecta con el nodo 1, el grafo conectado se muestra en la Figura 8.

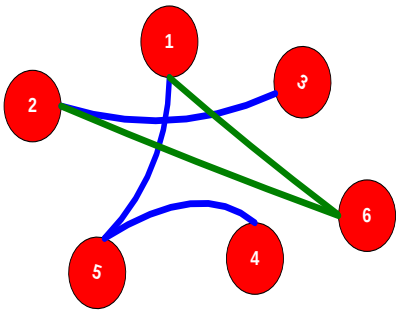


Figura 8. Conexión del nodo 6 con 1, mediante el enlace preferencial.

5. Para *t*=2, se tiene que calcular nuevamente el grado de cada nodo, ya que la red cambio con respecto a su topología (Figura 9)

b)

Nodo *i*	1	2	3	4	5	6
Grado k_i	2	2	1	1	2	2

Nodo *i*	3	4	1	2	5	6
Grado Π_i	0.1	0.2	0.4	0.6	0.8	1

c) $\sum k_i = 10$

Figura 9. a) Cálculo del grado del nodo para el *t=2* , b) Cálculo de la probabilidad de cada nodo,

6. Para *t*=2, se agrega un nodo nuevo a la red, en este caso es el nodo 7. Para determinar los enlaces del nuevo nodo con los nodos ya existentes en el grafo, se tiene que proceder por construir la lista de probabilidades para este tiempo (Figura 10) con su correspondiente grafo (Figura 10 a)).

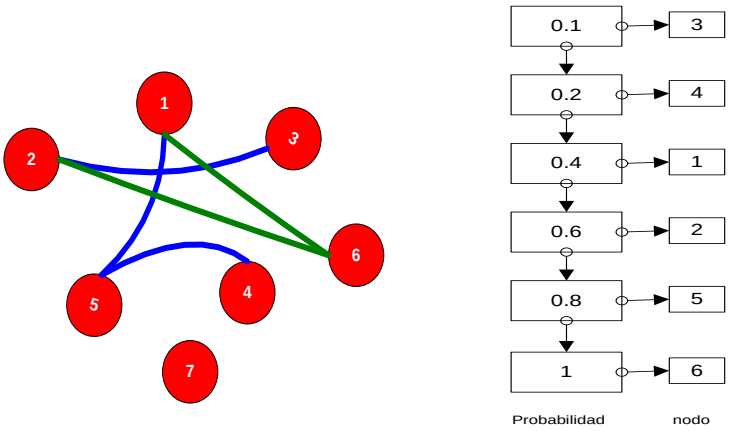


Figura 10. a) Grafo con el nuevo nodo agregado, b) Lista de probabilidades del grafo.

7. Para poder conectar el nodo 7 con algún nodo ya existente en el grafo, se procede a generar un número aleatorio, en este caso *Na*=0.25, es decir el *m* enlace del nodo 6 es con el nodo que entre en el rango del 0.21 al 0.4, por lo tanto el 7 se conecta con el nodo 4 del grafo. Esto se muestra en la Figura 11.

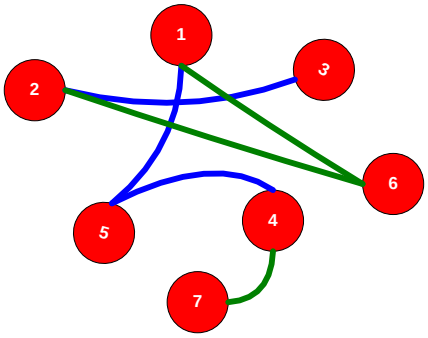


Figura 11. Conexión del nodo 7 con 4, mediante el enlace preferencial.

8. Como ya se sabe al nodo nuevo 7 le falta todavía una conexión mas con algún otro nodo existente en el grafo, utilizando nuevamente la lista de probabilidades en b) de la Figura 14. Para esto se tiene que generar otro numero aleatorio *Na*=0.66, el cual el rango probabilidad esta de 0.6 hasta 0.79, es decir el segundo y ultimo enlace en este tiempo es el nodo 7 se conecta con el nodo 2, el grafo conectado se muestra en la Figura 12.

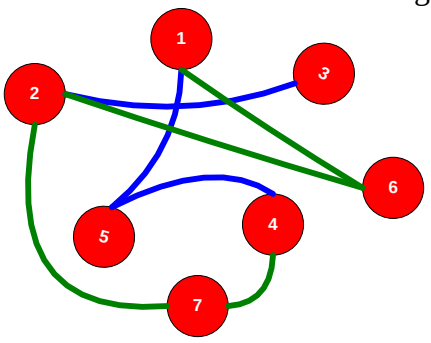


Figura 12. Conexión del nodo 7 con 2, mediante el enlace preferencial.

9. Como ya se han cumplido los *t* tiempos de construcción con sus *m* enlaces correspondientes, se concluye que el proceso de generación se ha terminado.

Modelo Ortega-Meza (MO-M)

El modelo Ortega-Meza inspirado en el crecimiento donde un nodo nuevo se conecta a *m* nodos con grado pequeño. Para aplicar el proceso de crecimiento, se inicializa los parámetros *p*,α, *m*, y *t*. Donde *m* es el número de lados que un nodo nuevo establece con otros nodos cuando se agrega a la red. El parámetro *t* representa unidades discretas de tiempo en cada una de las cuales se agrega un nodo con *m* lados a la red. Inicialmente, en el tiempo $t = 0$ comenzamos con un grafo completo con *m* nodos y en $t > 0$ un nuevo nodo es agregado con *m* lados. La selección de cada uno de los m vecinos se realiza con la Ecuación (3). En el tiempo *t*, la red contiene $n = m + t$ nodos y $e = (m(m-1))/2 + mt$ lados, sin ciclos ni lados paralelos.

Redes Punto-a-Punto (P2P) Complejas

Nuevos modelos de comunicación se han desarrollado sobre la Internet, esto debido a la gran cantidad de información que es necesario administrar. Los modelos pretenden ofrecer ventajas significantes a los usuarios, administrando la información de manera distribuida. Un modelo de comunicación que ha tomando gran relevancia son los sistemas conocidos como redes punto-a-punto sin estructura. En una red P2P, un conjunto de nodos forman conexiones entre ellos mismos y ofrecen sus recursos a otros puntos de la red. Los sistemas P2P junto con la capa de comunicaciones de red (típicamente la Internet) forma un sistema complejo que requiere operación autónoma a través de mecanismos de búsqueda inteligentes [5].

Métodos de Búsqueda en Redes P2P Complejas

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. 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 [38].

Anexo B

DEMOSTRACION DE LA COMPLEJIDAD DE SQRP

Conceptos de complejidad computacional

Este anexo presenta la descripción de la demostración de la complejidad del problema de direccionamiento de consultas semánticas (SQRP), se dará inicio con una breve descripción de la teoría de la complejidad computacional incluyendo los pasos necesarios para desarrollar la demostración de complejidad. Los conceptos de esta sección fueron seleccionados de [26, 27,28,34].

Teoría de Complejidad Computacional

Matemáticamente, los problemas pueden caracterizarse atendiendo su dificultad de solución por una computadora. Se han definido varias clases de problemas, entre las que destacan las clases *P*, *NP* y *NP−completo*. Como se observa en la Figura 1.

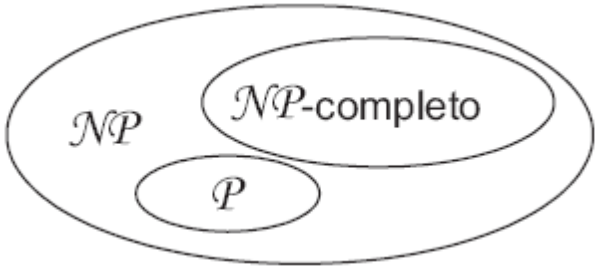


Figura 1. Relación entre los problemas P, NP y NP-completo.

Un problema se puede resolver en un *tiempo polinomial* cuando el tiempo de ejecución del algoritmo que lo resuelve se puede relacionar con el tamaño de la entrada con una formula polinomial. Los problemas para los que existe un *algoritmo polinomial* se denominan *P*. Se considera que los problemas *P* se pueden resolver en un tiempo de ejecución razonable para la informática actual.

Una gran cantidad de problemas útiles en el ámbito de la optimización combinatoria son difíciles de resolver, es decir no se pueden resolver en un tiempo de ejecución razonable. Hay problemas

que, pese a que no se haya encontrado un algoritmo polinomial que los resuelva, sí se puede saber en tiempo polinomial si un valor corresponde a la solución del problema. A este tipo de problemas, se les denomina *NP*. A simple vista, puede parecer que la gran mayoría de los problemas son *NP*, ya que intuitivamente parece que comprobar una solución es bastante más sencillo que calcularla. Sin embargo, para los problemas de optimización, comprobar si esos valores corresponden a la solución óptima no es nada sencillo. De hecho, existen problemas que requieren ejecutar el mismo algoritmo para buscar soluciones al problema tanto como para comprobar que unos valores son solución.

Los problemas *P* también son problemas NP, ya que siempre es posible comprobar que un valor es solución al problema en tiempo polinomial. Si no se encuentra una forma más sencilla de hacerlo, siempre se puede ejecutar el propio algoritmo polinomial que lo resuelve y comparar el resultado con el valor obtenido.

Además de las clases *P* y *NP* existe otra clase de problemas que es conocida como la clase *NP−completos* estos problemas que no tienen un algoritmo en tiempo polinómial que los resuelva. No se ha podido demostrar formalmente que no exista, pero los matemáticos creen que realmente no existe. Este tipo de problemas son un subconjunto de los problemas *NP*, es decir, que existe un algoritmo polinómial que puede determinar si un valor es solución al problema.

La teoría de la *NP*-completitud se aplica únicamente a problemas de decisión, aquéllos que tienen una solución "sí" o "no", en virtud de que su solución se define en función de la aceptación del lenguaje de casos codificados, por una máquina determinista de Turing (MDT). Como las máquinas de Turing sólo tienen la capacidad de aceptar o no una cadena, los únicos problemas que pueden resolver son los problemas de decisión. Algunas definiciones importantes de la teoría de la NP-completitud se enuncian a continuación.

Problema de decisión y algoritmo

Un problema de decisión se asocia con un lenguaje formal y un algoritmo con una máquina de Turing. Un problema de decisión π consta de un conjunto de casos *D*, obtenidos a partir de un caso genérico, el cual se especifica en términos de componentes: como conjuntos, grafos, funciones y números. El conjunto *D* contiene un subconjunto de casos-sí $Y \subseteq D$. Un caso pertenece a *Y* si y sólo si, la respuesta para ese caso es "sí".

Clase *P*
Es el conjunto de todos los problemas de decisión que pueden ser resueltos en tiempo polinomial por una MDT. Los problemas que pertenecen a esta clase se les denomina tratables.

Clase *NP*
Es el conjunto de todos los problemas de decisión que se resuelven en tiempo polinomial con una máquina no determinista de Turing (MNDT).

Relación entre *P* y *NP*
Como toda máquina determinista es una máquina no determinista, se tiene entonces que $P \subseteq NP$.

Transformación polinomial

Se dice que un problema $\pi_1=(D_1, Y_1)$ se puede transformar polinomialmente al problema $\pi_2=(D_2, Y_2)$, si existe una función $f: D_1 \rightarrow D_2$ que satisface las siguientes dos condiciones:

1. f es computable por un algoritmo determinista de tiempo polinomial.
2. Para todo caso $I \in D_1$, $I \in Y_1$ si y sólo si $f(I) \in Y_2$.

En tal caso se dice que hay una transformación polinomial de π_1 a π_2, y se escribe $\pi_1 \leq_P \pi_2$.

Problemas *NP*-completos

Un problema π es *NP*-completo si y sólo si,

1. $\pi \in NP$ y
2. si $\pi_1 \in NP$-completo implica que $\pi_1 \leq_P \pi$.

Esto es, que un problema *NP*-completo conocido se pueda transformar polinomialmente al problema de interés.

Demostración de la NP-completitud del Problema de Direccionamiento de Consultas Semánticas (SQRP) por reducibilidad desde el problema de la Mochila

La descripción del problema inicia con la definición de cada uno de los elementos de SQRP que son: grafo, repositorios y consultas. Para continuar con la definición formal de SQRP en su versión de optimización que es la que se aborda en la tesis. Después seguirá la versión de decisión de SQRP La segunda formulación es para la demostración de la complejidad que es un caso especial del primero en su versión de decisión.

Descripción General de SQRP

Una red se representa como un grafo $G = (V,E)$. El conjunto $V = \{v_1, v_2, v_3, ..., v_n\}$, contiene todos los *nodos* de la red y $E = \{e_1, e_2, ..., e_m\}$, es el conjunto de *aristas* o conexiones de la red. Se asume que una arista es un par ordenado de nodos distintos denotado simplemente por $e_i = \{u, v\}$; donde $u,v \in V$ y $u \neq v$. Dos nodos son adyacentes cuando están unidos por una arista y a cada arista se le asigna un costo dado por $cc(u,v)$, que representa el costo de transmitir la información de u a v.

Los usuarios de la red *consultan documentos a través de palabras claves* las cuales están registradas en un ***diccionario***. El diccionario es el conjunto $L = \{l_1, l_2 , l_3 , . . . , l_z \}$, que contiene todas las palabras claves asociadas a los documentos que pueden ser consultados en la red. Cada palabra clave l_i puede ser asociada a uno o mas documentos e identifican la naturaleza semántica del documento.

En una red cada nodo v_i contiene un conjunto de documentos a compartir que se encuentran depositados en un repositorio r_i llamado ***repositorio local***. El conjunto de todos los repositorios locales de la red se denota como $R=\{r_1, r_2, r_3,..., r_n\}$ y es denominado *repositorio de la red.* El repositorio local del nodo i es r_i y esta formado por un conjunto de documentos $r_i = \{d_{i1}, d_{i2},$

d_{i3}, ... , d_{iz_i}}, donde d_{ij} representa j-ésimo documento almacenado en el nodo i y z_i es la cantidad máxima de documentos contenidos en r_i. El contenido de un documento es tipificado mediante un conjunto de palabras claves $L_s \subseteq L$, las cuales indican la naturaleza semántica de la información almacenada en el documento, entonces $r_i = \{d_{i1}(L_1), d_{i2}(L_2), d_{i3}(L_3), ..., d_{iz_i}(L_s)\}$, donde $L_i \subseteq L$.

Debido a que el principal objetivo de una red P2P es el de compartir información, es frecuente que los usuarios utilicen la red con el propósito de encontrar recursos que contengan información específica. Para esto es necesario identificar cuáles nodos de la red tienen en sus repositorios documentos que puedan satisfacer las consultas generadas por los usuarios. La *consulta* es la forma en cómo el usuario "solicita" información en la red.

Una *consulta* es una tupla $c_i = \{v, k, tt_i\}$, donde v es el nodo desde el cual el usuario solicita la consulta, $k \in L$ es la palabra clave asociada a los documentos requeridos y tt_i es el instante en el cual se solicitó la consulta. En cada tt_i pueden ser creadas dos o mas consultas, generadas en paralelo asumiendo un tiempo de reloj de unidades fijas. Se considera $C = \{c_1, c_2, c_3, ..., c_m\}$ como el conjunto de m consultas generadas por los usuarios de la red en un intervalo de tiempo determinado.

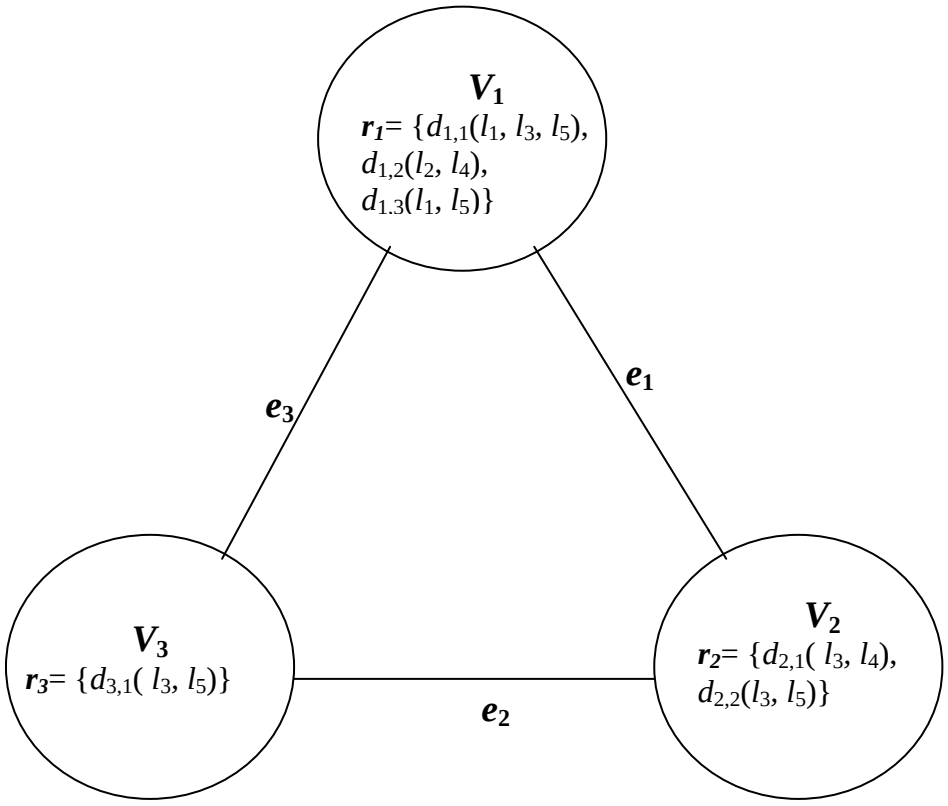
Una ***ruta* o *trayectoria*** es una secuencia de nodos $T = (t_1, t_2, ..., t_{|T|})$ donde cada nodo t_i es adyacente al nodo t_{i+1} para toda $i \in [1, |T|-1]$. La longitud de una ruta es el total de aristas $[t_i, t_{i+1}]$, que indican la cantidad de aristas que se deben recorrer para llegar desde el nodo t_1 hasta $t_{|T|}$ usando la trayectoria T.

La Figura 2 se muestra un ejemplo de los conceptos anteriores.

Cada consulta emitida c_i genera un costo de transmisión al pasar del nodo u al nodo v y se le denota por $cc(u,v)$. El costo total generado por una consulta c_i a través de una trayectoria T esta dado por: $Costo(c_i) = \sum_{u,v \in T} cc(u,v)$.

Como los resultados esperados al procesar la consulta son documentos que fueron seleccionados mediante palabras claves, a la consulta se le denomina *Consulta Semántica*.

Siempre que en un nodo v se emite una consulta $c_i = \{v, k, tt_i\}$, donde $v \in V$ y $k \in L$. Se selecciona un nodo adyacente al cual se reenviará la consulta llegando en el tiempo $tt_i + 1$. El nodo que recibe la consulta también la reenviará a uno de sus nodos adyacentes llegando en el tiempo $tt_i + 2$. Para el tiempo $tt_i + j$ la consulta ha sido reenviada y ha llegado j veces, creándose una ruta $T = (t_1, t_2, ..., t_{|T|})$, donde $t_1 = v_i \wedge [t_j, t_{j+1}] \in E$; por la cual ha transitado la consulta; a este proceso se le llama *enrutamiento*.



G = (V,E)
V = {v_1, v_2, v_3}
E = {e_1, e_2, e_3}
L = {l_1, l_2, l_3, l_4, l_5}
R = {r_1, r_2, r_3}
C = {c_1, c_2, c_3, c_4, c_5}

C	**{ *v, k, tt*$_i$ }**
c_1	v_2, l_3, 0
c_2	v_3, l_4, 1
c_3	v_1, l_5, 1

Para c_1
T={v_1, v_2, v_3}
CC = {10,5}

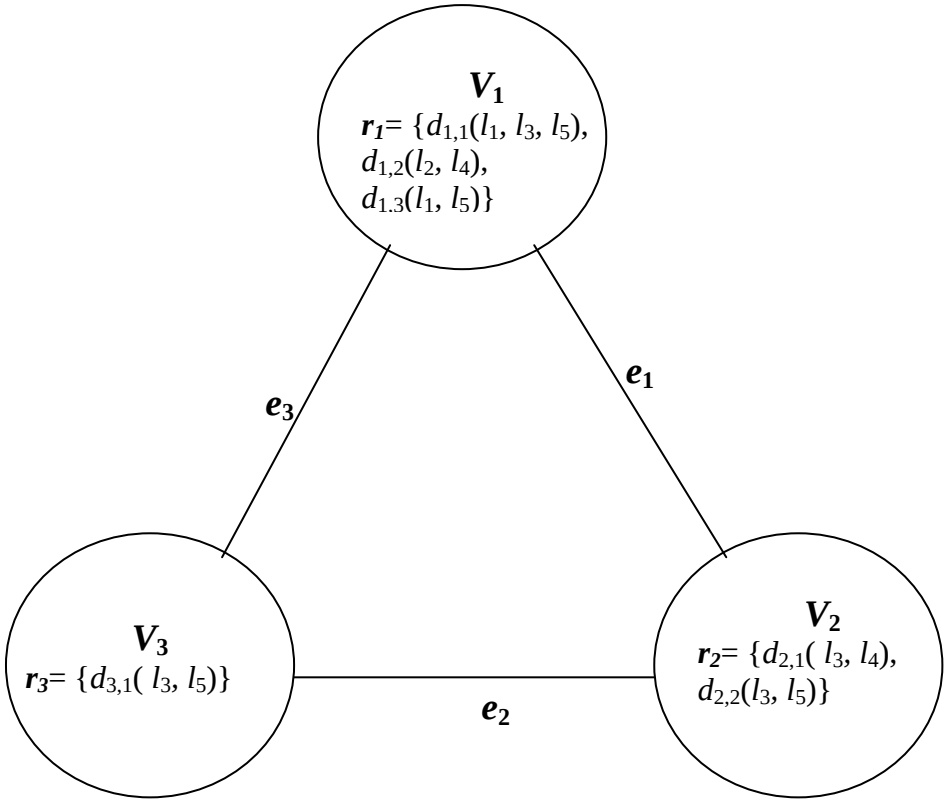
Figura 2. Grafo que representa la información definida anteriormente.

Cada vez que una consulta c_i = {v, k, tt_i} se encuentra en un nodo diferente u con su correspondiente repositorio r_u, el nodo u es revisado para conocer si existen documentos que corresponden con la palabra clave k, que el usuario está solicitando. En caso de existir, se informa al nodo solicitante v la cantidad de documentos existentes en el nodo u. Como se muestra en la Figura 3.

La cantidad de documentos localizados en los nodos de la trayectoria T iniciada en v, se calcula de la siguiente forma:

$$NumDoc(c_i, T) = \sum_{v \in T} \sum_{d_v \in R_v} f(c_i, d_v);$$

$$f(c_i, d_v) = \begin{cases} 0, si: k \notin d_v \\ 1, si: k \in d_v \end{cases}$$



G = (V,E)
V = {v_1, v_2, v_3}
E = {e_1, e_2, e_3}
L = {l_1, l_2, l_3, l_4, l_5}
R = {r_1, r_2, r_3}
C = {c_1, c_2, c_3, c_4, c_5}

C	{ v, k, tt_0 }
c_1	V_2, l_3, 0
c_2	V_3, l_4, 1
c_3	v_1, l_5, 1

Para c_1
T={v_1, v_2, v_3}
CC = {20,10,30//10,20,5//30,5,20}
Costo(c_1)=15
NumDoc(c_1, T) = 3

Figura 3. Grafo que representa la información definida anteriormente.

Definición Formal

El problema de SQRP es definido formalmente como:

Dados

G = (V,E)	la red P2P modelada,
CC	la matriz de costos de transmisión,
L = {l_1, l_2 , l_3 , . . . , l_z }	el diccionario de palabras clave,

$R = \{r_1, r_2, r_3, \ldots, r_n\}$	los repositorios locales de la red, donde $r_i = \{d_{i1}(L_1), d_{i2}(L_2), d_{i3}(L_3), \ldots, d_{iz_i}(L_s)\}$, $L_i \subseteq L$ y d_{ij} representa el j-ésimo documento almacenado en el nodo i y z_i es la cantidad máxima de documentos contenidos en r_i,
$C = \{c_1, c_2, c_3, \ldots, c_m\}$	las consultas emitidas por los usuarios, donde $\{v, k, tt_i\}$ con $v \in V$, $k \in L$, y tt_i es el instante en el cual se solicitó la consulta.
maxResult	cantidad mínima de documentos para satisfacer una consulta, y
TTL	costo máximo permitido en una consulta.

El problema SQRP consiste en determinar una trayectoria $T \subset G$, que maximice la relación:

$$\varepsilon = \frac{\sum_{c_i \in C} NumDoc(c_i, T)}{\sum_{c_i \in C} Costos(c_i)}$$

sujeta a las siguientes restricciones:

a) $NumDoc(c_i,T) > 0; \forall c_i \in C$ (Se satisfacen todas las consultas C).
b) $NumDoc(c_i, T) \geq maxResult; \forall c_i \in C$
c) $Costo(c_i) \leq TTL; \forall c_i \in C$

Para probar que SQRP es NP_duro basta probar que una versión simplificada del problema de decisión asociado a SQRP es NP_completo.

Definición Formal de la Versión de Decisión

La versión de decisión del problema de SQRP es definido formalmente como:

Dados

$G = (V,E)$	la red P2P modelada,
CC	la matriz de costos de transmisión,
$L = \{l_1, l_2, l_3, \ldots, l_z\}$	el diccionario de palabras clave,
$R = \{r_1, r_2, r_3, \ldots, r_n\}$	los repositorios locales de la red, donde $r_i = \{d_{i1}(L_1), d_{i2}(L_2), d_{i3}(L_3), \ldots, d_{iz_i}(L_s)\}$, $L_i \subseteq L$ y d_{ij} representa el j-ésimo documento almacenado en el nodo i y z_i es la cantidad máxima de documentos contenidos en r_i,
$C = \{c_1, c_2, c_3, \ldots, c_m\}$	las consultas emitidas por los usuarios, donde $\{v, k, tt_i\}$ con $v \in V$, $k \in L$, y tt_i es el instante en el cual se solicitó la consulta.
maxResult	cantidad mínima de documentos para satisfacer una consulta
TTL	costo máximo permitido en una consulta

El problema SQRP consiste en determinar si existe una trayectoria T $\subset$ G, tal que satisfaga las siguientes restricciones:

a) $\varepsilon \geq$ *maxResult / TTL*; que la eficiencia sea mayor igual que el cociente de los límites.
b) $NumDoc(c_i,T) > 0; \forall\ c_i \in C$ (Se satisfacen todas las consultas C).
c) $NumDoc(c_i,T) \geq maxResult; \forall\ c_i \in C$
d) $Costo(c_i) \leq TTL; \forall\ c_i \in C$

Para probar que la versión de decisión de SQRP es NP_completo basta probar que una versión simplificada del problema es NP_completo.

Consideraciones para la simplificación del Problema del Enrutado de Consultas Semánticas Restringido

Para simplificar el problema SQRP se harán las siguientes simplificaciones:

- $C = \{c_1\}$, lo cual significa que en un intervalo de tiempo determinado solo se podrá emitir una consulta en el sistema $c_1 = \{v, l_i, 1\}$ donde $v \in V$.

A la versión simplificada de SQRP se denominará SQRP_R. Se observa que éste problema se formulará también como un problema de decisión. SQRP_R se obtiene a partir de SQRP, aplicando las simplificaciones descritas anteriormente.

Después de las consideraciones de simplificación se describe el problema π_1 que es SQRP_R, para ser formulado como un problema de decisión.

Dados

$G = (V,E)$	la red P2P modelada,
CC	la matriz de costos de transmisión,
$L = \{l_1, l_2, \ldots, l_z\}$	el diccionario de palabras clave,
$R = \{r_1, r_2, r_3, \ldots, r_n\}$	los repositorios locales de la red, donde $r_i = \{d_{i1}(L_1), d_{i2}(L_2), d_{i3}(L_3), \ldots, d_{iz_i}(L_s)\}$, $L_i \subseteq L$ y d_{ij} representa el j-ésimo documento almacenado en el nodo i y z_i es la cantidad máxima de documentos contenidos en r_i,
$C = \{c_1\}$	las consultas emitidas por los usuarios, donde $\{v, k, 1\}$ con $v \in V$ y $k \in L$.
maxResult	cantidad mínima de documentos para satisfacer una consulta
TTL	costo máximo permitido para una consulta

El problema SQRP_R consiste en determinar si existe una trayectoria T ⊂ G, tal que satisfaga las siguientes restricciones:

a) ε ≥ *maxResult / TTL*; que la eficiencia sea mayor igual que el cociente de los límites.
b) *NumDoc*(c_1,*T*) > 0,
c) *NumDoc*(c_1 ,*T*) ≥ *maxResult,* y
d) C*osto*(c_1) ≤ *TTL*.

Demostrar que SQRP es NP

El primer paso de la demostración es demostrar que existe un algoritmo no-determinista que encuentra una trayectoria T en tiempo polinomial y satisface las restricciones.

a) ε ≥ *maxResult / TTL,*
b) *NumDoc*(c_1,*T*) > 0,
c) *NumDoc*(c_1 ,*T*) ≥ *maxResult,* y
d) C*osto*(c_1) ≤ *TTL*.

Etapa adivinadora de una solución candidata
// Adivinar la longitud de la ruta
Length = random (n)
// Adivinar los nodos de la ruta
Para *i* = 1 hasta *length*+1
    T[*i*] = random (*v*)

} $O(n)$

Entrada: *TTL, maxResult, T*
Salida: Indica si T es una trayectoria valida que cumple con las restricciones a,b,c y d.

Etapa de verificación: la solución candidata es una T

// Verificar que no existen vértices repetidos en *T* y que t[*i*] y t[*i+1*] son adyacentes.
Para i = 1 hasta length -1
    Para j = i+1 hasta length
        Si (t[*i*] ≠ t[*j*]) j = j + 1;
        Si no respuesta = no;
    Fin_Para
    Si ((T(t[*i*] ,t[i+1]) == 1) entonces i = i + 1;
    De lo contrario respuesta = no;
Fin_Para

} $O(n^2)$

```
// Verificar límites : TTL y maxResult
NumDoc = 0; Costo = 0;
Para i = 1 hasta length
      Costo_c1 = costo(t[i] ,t[i+1]) + Costo_c1;
       Para j = 1 hasta length(rj)
          if (i ≠ j) entonces
              if ( k == (t[i].d[j]) ) doc = doc + 1;
       NumDoc = NumDoc + doc;
Fin_Para
ε = NumDoc / Costo_c1;
Si ((NumDoc > 0) ^ (Costo_c1 ≤ TTL)
     ^ (NumDoc ≥ maxResult) ^ (ε ≥ (maxResult / TTL)))
    Entonces respuesta = si
    De lo contrario respuesta = no
Regresar respuesta
```

$O(n^2)$

Dado que: Si, es posible construir una MTND de tiempo polinomial

se concluye que SQRP_R $\in$ NP

Transformación Polinomial

La función de transformación polinomial es de suma importancia, ya que a través de esta se transformara la instancia de un problema π_1 a su correspondiente instancia en de un problema π_2 el cual ya es NP-completo, haciéndose corresponder las dos instancias de los problemas.

Transformación:

$V = \{v_i \mid 1 \le i \le n\} \cup v_0$; donde el elemento v_i esta asociado con el objeto $o_i \in O$
$E_1 = \{(v_i, v_j, cc_j) \mid i \ne j ; \forall v_i , \forall v_j \in V ; cc_i + cc_j \le P\}$
$E_2 = \{(v_j, v_i, cc_j) \mid i \ne j ; \forall v_i , \forall v_j \in V ; cc_j + cc_i \le P\}$
$E_3 = \{(v_0, v_i, cc_i) \mid \forall v_i \in V ; cc_i \le P\}$
$E = E_1 \cup E_2 \cup E_3$
$L = \{li \mid 1 \le i \le n\}$; donde el elemento li esta asociado con el objeto $o_i \in O$
$r_i = \{ d_{i,1}(li), d_{i,2}(li), ... ,d_{i,bi}(li)\}\ 1 \le i \le n ; r_0 = \{\ \}$
$R = \{r_i \mid 1 \le i \le n\} \cup r_0$; donde el elemento v_i esta asociado con el objeto $o_i \in O$
TTL = P
maxResult = G_M

Ejemplo de una transformación:

INSTANCIA MOCHILA 0-1	***INSTANCIA SQRP_R***
O={o_1, o_2, o_3, o_4}	V={v_1, v_2, v_3, v_4} L ={1, 2, 3, 4} R={ r_1, r_2, r_3, r_4}
C={75, 150, 100, 50}	E={(v_1, v_2, 150), (v_1, v_3, 100), (v_1, v_4, 50), (v_2, v_1, 75), (v_2, v_3, 100), (v_2, v_4, 50), (v_3, v_1, 75), (v_3, v_2, 150), (v_3, v_4, 50), (v_4, v_1, 75), (v_4, v_2, 150), (v_4, v_3,100) }
B={200, 100, 250, 150}	r_1 = { $d_{1,1}$(1), $d_{1,2}$(1), ... ,$d_{1,200}$(1)} r_2 = { $d_{2,1}$(2), $d_{2,2}$(2), ... ,$d_{2,100}$(2)} r_3 = { $d_{3,1}$(3), $d_{3,2}$(3), ... ,$d_{3,250}$(3)} r_4 = { $d_{4,1}$(4), $d_{4,2}$(4), ... ,$d_{4,150}$(4)}
P = 300 G_M = 500	TTL = 300 NumDoc = 500
	Datos Adicionales: E_3={(v_0,v_1,75),(v_0,v_2,150),(v_0,v_3,100),(v_0,v_4,50)} r_0 = {} c_1 = (v_0) C = {c_1}

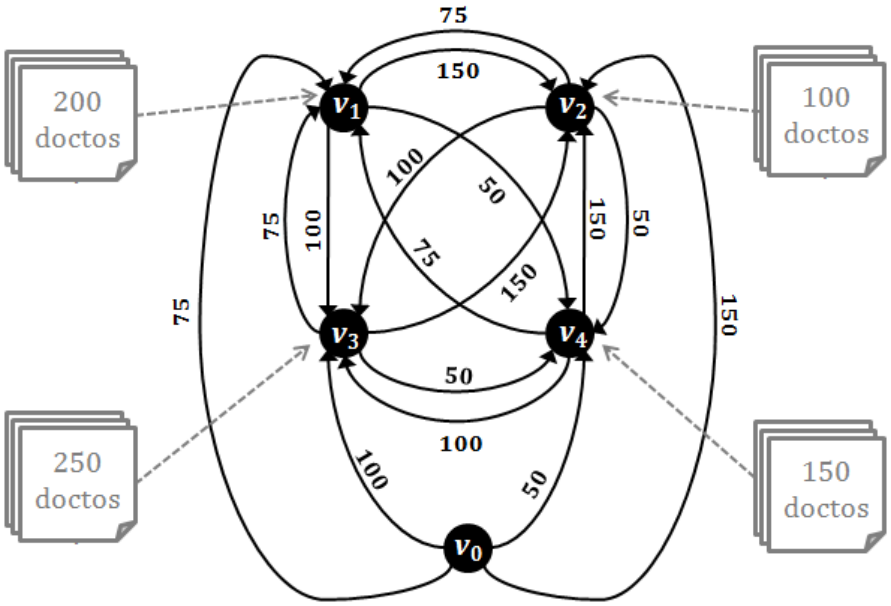
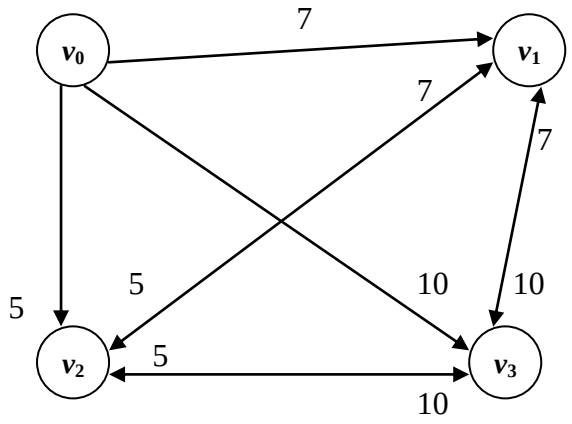


Figura 4. Grafo resultado de la transformación.

INSTANCIA MOCHILA 0-1	***INSTANCIA SQRP_R***
Si $U = \{o_2, o_3, o_4\}$, entonces: $Costos(U) = \sum_{o_i \in U} c(o_i) \leq P$ Costos(U) = (150 + 100 + 50) ≤ 300 $Beneficio(U) = \sum_{o_i \in U} b(o_i) \geq G_M$ Beneficios(U) = (100 + 250 + 150) ≥ 500 Todas las restricciones fueron satisfechas para el empaquetamiento U. Por lo tanto U es un empaquetamiento válido.	Si T = { v_0, v_2, v_3, v_4}, entonces: $NumDoc(c_i, T) = \sum_{v \in T} \sum_{d_v \in R_v} f(c_i, d_v) \geq maxResult$ $NumDoc(c_1, T) > 0$ $NumDoc(c_1, T) = 100 + 250 + 150 \geq 500$ $Costos(c_i) = \sum_{u,v \in T} cc_i(u,v) \leq TTL$ Costos(c_i) =150 + 100 + 50 ≤ 300 $\varepsilon = \frac{\sum_{c_i \in C} NumDoc(c_i, T)}{\sum_{c_i \in C} Costos(c_i)}$ ε = 500 / 300. ε ≥ *maxResult / TTL* ε ≥ 500 / 300. Todas las restricciones fueron satisfechas para la trayectoria T. Por lo tanto T es una trayectoria válida.

Ejemplo de una transformación:

INSTANCIA SQRP_R	***INSTANCIA MOCHILA 0-1***
$V=\{v_1, v_3, v_2\}$ $L=\{1, 3, 2\}$ $R=\{r_1, r_3, r_2\}$	O={o_1, o_2, o_3}
$E=\{(v_1, v_2, 5), (v_1, v_3, 10),$ $(v_3, v_1, 7), (v_3, v_2, 5),$ $(v_2, v_1, 7), (v_2, v_3, 10)\}$	C={7, 5, 10}
$r_1 = \{d_{1,1}(1), d_{1,2}(1), \ldots, d_{1,20}(1)\}$ $r_2 = \{d_{2,1}(2), d_{2,2}(2), \ldots, d_{2,10}(2)\}$ $r_3 = \{d_{3,1}(3), d_{3,2}(3), \ldots, d_{3,25}(3)\}$	B={20, 10, 25}
TTL = 30 NumDoc = 50	P = 30 G_M = 50
Datos Adicionales: $E_3=\{(v_0,v_1,7),(v_0,v_2,5),(v_0,v_3,10)\}$ $r_0 = \{\}$ $c_1 = (v_0)$ $C = \{c_1\}$	



INSTANCIA SQRP_R	***INSTANCIA MOCHILA 0-1***
Si T ={ v_0, v_2, v_3, v_1}, entonces: $NumDoc(c_i, T) = \sum_{v \in T} \sum_{d_v \in R_v} f(c_i, d_v) \geq maxResult$ $NumDoc(c_1, T) > 0$ $NumDoc(c_1, T) = 10 + 25 + 20 \geq 50$ $Costos(c_i) = \sum_{u,v \in T} cc(u, v) \leq TTL$ Costos(c_i) = 5 + 10 + 7 ≤ 30 $\varepsilon = \frac{\sum_{c_i \in C} NumDoc(c_i, T)}{\sum_{c_i \in C} Costos(c_i)}$ ε = 50 / 22 = 2.72 ε ≥ *maxResult / TTL* ε ≥ 50 / 30 = 1.6 2.72 ≥ 1.6 Todas las restricciones fueron satisfechas para la trayectoria T. Por lo tanto T es una trayectoria válida.	Si U = {o_2, o_3, o_1}, entonces: $Beneficio(U) = \sum_{o_i \in U} b(o_i) \geq G_M$ Beneficios(U) = (10 + 25 + 20) ≥ 50 $Costos(U) = \sum_{o_i \in U} c(o_i) \leq P$ Costos(U) = (5 + 10 + 7) ≤ 30 Todas las restricciones fueron satisfechas para el empaquetamiento U. Por lo tanto U es un empaquetamiento válido.

Demostrar SQRP_R es NPC

Para demostrar que SQRP_R es NPC es necesario seguir los siguientes pasos:

1. Seleccionar un problema Π_2 perteneciente a NPC
2. Definir una función de transformación polinomial de Mochila 0-1 a SQRP_R
3. Demostrar que toda instancia de Π_1 es una instancia-si, si y solo si la instancia reducida también es una instancia-si de Π_2.

1. Seleccionar un problema Π_2 perteneciente a NPC

El problema de Mochila 0-1 es un problema NPC [Gary 1979]

Sea:

Un conjunto de objetos O =$\{ o_0, o_1, o_2, \ldots , o_n \}$, donde *n* es la cantidad de objetos. Para cada objeto le corresponde un peso el cual se guarda en el conjunto C=$\{ c_0, c_1, c_2, \ldots , c_n\}$; así mismo se tiene un beneficio de cada uno de los objetos de O, el cual se guarda en el conjunto B = $\{ b_0, b_1, b_2, \ldots , b_n\}$. Un número entero *P* que indica la capacidad limite de la mochila.

Pregunta:

¿ Existe un conjunto U de objetos, $(\cos to(U) = \sum_{i=1}^{m} c[u_i] \leq P) \wedge (Beneficio(U) = \sum_{i=1}^{m} B[u_i] \geq G_M)$?

Dados

O =$\{o_1, \ldots , o_n \}$	objetos,
C =$\{c_1, \ldots , c_n\}$	costos de los objetos,
B = $\{b_1, \ldots , b_n\}$	beneficio de los objetos,
G_M	beneficio mínimo que deben aportar la suma de los objetos, y
P	capacidad limite de la mochila.

El problema Mochila 0-1 consiste en determinar si existe un conjunto de objetos U, tal que satisfaga las siguientes restricciones:

a) Beneficios (U) $\geq G_M$
b) C*osto*(U) ≤ P.

π_2 es el problema de la Mochila 0-1

Definir una función de transformación polinomial de Mochila 0-1 a SQRP_R

Transformación:

$V = \{v_i \mid 1 \leq i \leq n\} \cup v_0$; donde el elemento v_i esta asociado con el objeto $o_i \in O$
$E_1 = \{(v_i, v_j, cc_j) \mid i \neq j \; ; \forall v_i , \forall v_j \in V \; ; cc_i + cc_j \leq P\}$
$E_2 = \{(v_j, v_i, cc_j) \mid i \neq j \; ; \forall v_i , \forall v_j \in V \; ; cc_j + cc_i \leq P\}$
$E_3 = \{(v_0, v_i, cc_i) \mid \forall v_i \in V \; ; \quad cc_i \leq P\}$

E = $E_1 \cup E_2 \cup E_3$
L = {li | $1 \le i \le$ n }; donde el elemento li esta asociado con el objeto $o_i \in$ O
r_i = { $d_{i,1}$(li), $d_{i,2}$(li), ... ,$d_{i,bi}$(li)} $1 \le i \le$ n ; r_0 ={ }
R = {r_i | $1 \le i \le$ n}$\cup r_0$; donde el elemento v_i esta asociado con el objeto $o_i \in O$
TTL = P
maxResult = G_M

Demostrar que la función de reducción es polinomial:

1. Crear vértices, repositorios y diccionario. O(n)
2. Crear documentos y asignarlos a los repositorios O(n^2)
3. Crear aristas con costos O(n^2)
4. TTL = P y maxResult = G_M O(2).

Algoritmo de Reducción Polinomial

```
Crear  v0
V = V ∪ v0 // Considerar que el costo de v0= 0
Para ∀oi ∈ O                                          ┐
      Crear  vi                                       │
      Crear  ri                                       │
      Crear li                                        │
      V = V ∪ vi                                      │  O(n)
      R = R ∪ ri                                      │
      L = L ∪ li                                      │
Fin_Para                                              ┘
r0 = 0;
o0= 0;
Para ∀oi ∈ O                                          ┐
    Para j = 1 hasta bi                               │  O(n2)
        ri = ri + dj (li);                            │
Fin_Para                                              ┘

Desde i = 1 hasta i = |V|                             ┐
   Desde j = 1+i hasta j = |V|                        │
      Si ( C[oi] + C[oj] ≤ P )                        │
         Entonces  crear e1 = {vi, vj, c[oj]}         │
       Si ( C[oj] + C[oi] ≤ P )                       │  O(n2)
          Entonces  crear e2 = {vj, vi, c[oj]}        │
       Si ( C[o0] + C[oi] ≤ P )                       │
          Entonces  crear e3 = {v0, vi, c[oi]}        │
 E = E ∪e1∪e2∪e3                                      │
      Fin_Si                                          │
   Fin_Para                                           │
Fin_Para                                              ┘
TTL = P
maxResult = GM
```

Demostrar que toda instancia de Π_1 es una instancia-si, si y solo si la instancia reducida también es una instancia-si de Π_2

Una instancia-si de Π_1 corresponde con una instancia-si de Π_2 bajo la transformación polinomial.

Supongamos que *U* es un empaquetamiento valido en la mochila 0-1, se puede obtener *T* que es un recorrido de SQRP_R como sigue:

$t_0 = v_0$, $t_i = v_i$ tal que v_i esta asociado con $o_i \in O \quad \forall i$.

Como es un empaquetamiento se sabe que la suma de los pesos de los objetos es a lo mas *P* (Costo(*U*) ≤ *P*), por construcción el peso de cualquier objeto $o_i \in O$ se asigna al costo de cualquier arco que incide en $v_i \in$ T ; y como *TTL = P*, se sigue que la suma de los costos de los arcos de *T* es a lo mas TTL, esto significa que ($\sum_{u,v \in T} cc(u,v) \le TTL$). Por otra parte se tiene que como consecuencia de la transformación, los costos de los objetos corresponden con los costos de los arcos que conectan a los nodos en la trayectoria T, entonces

$Costos(c_i) = \sum_{u,v \in T} cc(u,v) \le TTL$. Por lo tanto se satisface la restricción d.

Por otro lado, dado que *U* es un empaquetamiento, se sabe que la suma de los beneficios es al menos G_M (Beneficios (U) ≥ G_M), por construcción el beneficio de cualquier objeto $o_i \in O$ se asigna al número de documentos del vértice $v_i \in T$ y como $maxResult = G_M$, se sigue que la suma del número de documentos almacenados en los vértices de *T* es al menos *maxResult, esto significa que* ($\sum_{v \in T} \sum_{d_v \in R_v} f(c_i, d_v) \ge maxResult$). Por otra parte al aplicar la transformación, el beneficio de los objetos corresponde con la cantidad de documentos que hay en cada nodo en la trayectoria T, entonces $NumDoc(c_i, T) = \sum_{v \in T} \sum_{d_v \in R_v} f(c_i, d_v) \ge maxResult$. Por lo tanto se satisface la restricción b,c.

Como consecuencia de estas dos condiciones se tiene que ε < maxResult / TTL. Por lo tanto se concluye que la instancia asociada por la transformación a la instancia-si del problema de empaquetamiento le corresponde una instancia-si del problema SQRP_R.

Ahora se probará que **existe una trayectoria T ⊂ G**, y satisface las siguientes restricciones: ε ≥ *maxResult / TTL, NumDoc*(c_1,*T*) > 0, *NumDoc*(c_1 ,*T*) ≥ *maxResult*, y C*osto*(c_1) ≤ *TTL;* entonces **existe un conjunto U de objetos**, que satisfacen las siguientes restricciones: el Beneficio(U) ≥ G_M y el C*osto*(U) ≤ P.

Supongamos que T es una trayectoria valida en SQRP, se puede obtener un empaquetamiento U válido en la mochila 0-1 como sigue:

$t_i = v_i$ tal que o_i esta asoiado con $t_i \in$ T $\forall i$.

Como es una trayectoria se sabe que la suma de los costos de los arcos es a lo mas TTL ($cc(u,v) \leq TTL$), por construcción el costo de cualquier arco $t_i \in T$ se asigna al peso de cualquier objeto $o_i \in O$; y como $P = TTL$, se sigue que la suma de los pesos de los objetos de U es a lo mas P, esto significa que ($\sum_{o_i \in U} c(o_i) \leq P$). Por otra parte se tiene que como consecuencia de la transformación, los costos de los arcos corresponden con los costos de los objetos seleccionados para el empaquetamiento U, entonces

$Costos(U) = \sum_{o_i \in U} c(o_i) \leq P$. Por lo tanto se satisface la restricción b.

Por otro lado, dado que T es una trayectoria, se sabe que la suma de los documentos es al menos *maxResult*, por construcción el número de documentos de cualquier vértice $t_i \in$ T se asigna al beneficio de cualquier objeto $o_i \in O$ y como G_M = *maxResult*, se sigue que la suma de los beneficios de los objetos almacenados en U es al menos G_M, *esto significa que* ($\sum_{o_i \in U} b(o_i) \geq G_M$).

Por otra parte al aplicar la transformación, la cantidad de documentos corresponde con el beneficio de los objetos que hay en el empaquetamiento U, entonces $Beneficio(U) = \sum_{o_i \in U} b(o_i) \geq G_M$. Por lo tanto se satisface la restricción a.

Por lo tanto se concluye que la instancia asociada por la transformación a la si-instancia del problema de empaquetamiento le corresponde una si-instancia del problema SQRP_R.

Por lo tanto se concluye que SQRP_R es NP_completo

Anexo C

SISTEMA MULTIAGENTE

El estudio de la inteligencia que surge de las interacciones entre muchos agentes, es conocido como *sistemas multiagente*. Dichos sistemas son complejos por naturaleza y se caracterizan por un gran número de partes que tienen muchas interacciones. Además esta complejidad no es accidental, es una propiedad innata de los tipos de tareas para las cuales estos sistemas son utilizados [66].

Los agentes son los elementos principales que conforman un sistema multiagente. Entonces 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 [68].

Los desarrollos de aplicaciones sobre redes de gran escala de telecomunicaciones y sistemas distribuidos ha incrementado su interés con el uso de sistemas multiagente [66]. La propia definición de agentes de software, indica que los agentes tienen características parecidas a la de los sistemas P2P, como son su funcionamiento de forma distribuida y autónoma. El trabajo con sistemas multiagente se facilita cuando dichos agentes trabajan con servicios P2P estándares [69].

Los desarrollos sobre redes de gran escala de telecomunicaciones y sistemas distribuidos han visto incrementado su interés con los sistemas multi-agente [66]. 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 inter-operar con servicios P2P estándares [69].

Los agentes son los elementos principales que conforman un sistema multi-agente. Podemos definir un agente como: “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” [68]. 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 [69]:

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.

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 debido a que pueden son modelados como un sistema multiagente y comparten muchas de las características importantes como la emergencia y autoadaptación que son necesarias para que el algoritmo de búsqueda, pueda aprender y adaptarse al ambiente de búsqueda [65].

Dentro de los algoritmos metaheurísticos existe una clase de algoritmos poblacionales denominados algoritmos de colonias de hormigas, los cuales son modelados como un sistema multiagente, cada hormiga agente contribuye a lograr un objetivo global, los datos son descentralizados y la computación es asíncrona. Las colonias de hormigas exhiben un comportamiento emergente debido a que construyen sus resultados de manera incremental por lo que son consideradas como sistemas autoadaptativos complejos [64].

a) Análisis del Desempeño de algoritmos Metaheurísticos

El desempeño de un algoritmo basado en metaheurísticas esta determinado por su eficiencia y su efectividad. La *efectividad* de un algoritmo se refiere a la calidad de la solución encontrada o su confiabilidad en la tarea de encontrar soluciones adecuadas, es decir son altamente dependiente de la estructura del problema y esto tiene que ver directamente con las instancias del problema que se analiza. Por otro lado la eficiencia, esta muy relacionada con el conocimiento que se tenga del dominio del problema [Rosas 2007],[Landero 2008].

Sin embargo, este análisis no siempre puede ser aplicable a todos los algoritmos. Existen algunos algoritmos como los metaheurísticos que son considerados muchas veces como cajas negras cuyo funcionamiento interno no es conocido, otro aspecto a considerar pudiera ser que la complejidad del problema a resolver dificultará la estimación de la eficiencia. Para el análisis del desempeño de los algoritmos metaheurísticos, la comunidad científica muestra mayor interés en estudiar los aspectos relacionados con su efectividad [Merz 1998]. Este aspecto es el más complicado de modelar debido a que interviene de manera directa la naturaleza del problema.

McGeoch [McGeoch 2000] y Barr [Barr 1995] sugieren la existencia de tres categorías principales de factores que afectan el desempeño algorítmico: problema, algoritmo y ambiente. Factores relacionados con el *problema*: dimensión, distribución de los parámetros que lo definen, estructura del espacio de solución entre otros. Factores relacionados con el *algoritmo*: estrategias heurísticas seleccionadas (procesos de construcción de solución inicial y parámetros de búsqueda asociados), códigos de computadoras empleados, configuración de control interno del algoritmo y comportamiento en la ejecución entre otros. Factores relacionados con el *ambiente de desarrollo*: estos factores se refieren al ambiente físico en el que serán ejecutados los algoritmos como los son el software (sistema operativo, compilador) y hardware (velocidad del procesador, memoria). Así como también aquellos relacionados con el programador: pericia, habilidad de afinación, lógica.

Los criterios para medir el desempeño de los algoritmos de aproximación dependen de los métodos elegidos para su caracterización, que pueden ser teóricos ó experimentales. En los teóricos, para cada algoritmo, se determina matemáticamente la cantidad de recursos necesarios como función del tamaño del caso considerado mejor, peor o promedio. En el caso de usar métodos experimentales se basan en la experimentación para realizar la caracterización y a diferencia de los anteriores permiten describir el comportamiento de casos específicos. En los algoritmos metaheurísticos raramente se estudia su efectividad de manera teórica, esto debido principalmente a la complejidad de los problemas de combinatoria. Asumiendo que no hay algoritmos que resuelvan en tiempo polinomial los problemas, la única opción disponible es analizar experimentalmente la efectividad de los algoritmos metaheurísticos.

b) Métodos Basados en Colonias de Hormigas

Los algoritmos de colonias de hormigas están inspirados en el comportamiento colectivo de la búsqueda de comida, en donde las hormigas se comunican de manera indirecta con otros miembros de su colonia. 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" con el objetivo de encontrar la ruta más corta entre el nido y la fuente de comida [Grassé 1959].

Las colonias de hormigas artificiales son representadas como sistemas multiagente. Es decir cada hormiga-agente tiene información o capacidades parciales para resolver una parte del problema. La información se va registrando en la tabla de feromonas, la cual es administrada de manera local, 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 simuladas como sistemas adaptativos complejos.

Ant Colony Optimization, es una metodología basada en el comportamiento de las hormigas, y comprende diferentes tipos de sistemas de hormigas [Dorigo 2004]. Los primeros algoritmos de hormigas fueron diseñados para resolver problemas de optimización basados en grafos, y con una tabla de feromonas global. En la actualidad se han diversificado la metodología original con el objetivo de atacar problemas con estructuras distribuidas tales como el enrutamiento de paquetes en redes que usan tablas de feromonas distribuidas.

c) Algoritmos de Hormigas

El primer algoritmo usado fue Ant Colony System [Dorigo 1997], que es uno de los algoritmos de hormigas con mejor rendimiento y más referenciados [Asmar 2005], el otro algoritmo es SemAnt que es una propuesta de solución para el problema del direccionamiento de consultas semánticas.

Referencias

[1] Adenso-Díaz, Belardino; Laguna, Manuel : "*Fine-Tuning of Algorithms Using Fractional Experimental Desings and Local Seach"*. Instituto de Investigación de Operaciones Institute y Ciencias de la Administración, pp. 99-114, 2004.

[2] Birattari Mauro: "The Problem of Tuning Metaheuristics as seen from a machine learning perspective", Tesis Doctoral, Universidad libre de Bruxelles. 2004.

[3] Barr, R.; Golden, J.; Kelly, M.: "Designing and Reporting Computational Experiments with Heuristics Methods", Journal of Heuristics, Vol. 1, pp. 9-32. 1995.

[4] Pantrigo F., Juan J.: "Resolución de Problemas de Optimización Dinámica Mediante la Hibridación entre Filtros de Partículas y Metaheurísticas Poblacionales". Tesis Doctoral, Universidad Rey Juan Carlos, 2005.

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

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

[12] 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.

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

[14] 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.

[15] Zhou Y., Croft W., Levine B., "Content-based search in peer-to-peer networks", Reporte Técnico IR-367, Universidad de Masachusets, E.U., 2004.

[16] Ortega R., "Estudio de las Propiedades Topológicas en Redes Complejas con Diferente Distribución del Grado y su Aplicación en la Búsqueda de Recursos Distribuidos", Tesis Doctoral, p.151, 2009.

[17] Gómez Santillán C., Turrubiates López T., Cruz Reyes L., Meza Conde E., Aguirre Lam M.,: Statistical Methodology of Topological Feature Selection in Complex Network to Improve Distributed Resource Search, 15th International Congress on Computer Science Research October 29-31, Aguascalientes México, 2008.

[18] Ridge, Enda: "Design of Expirements for the Tuning of Optimization Algorithms", Tesis Doctoral. Universidad de York, 2007.

[19] Gómez Santillán C.G., Cruz-Reyes L., Meza Conde E., Aguirre Lam M.A..“ Performance Analysis of the Neighboring-Ant Search Algorithm through Design of Experiment ”, Lectures Notes in Computer Science, Springer-Verlag, Berli✓n Heidelberg New York, 2009.

[20] Michalewicz, Z., Fogel, D.B., "How to Solve It: Modern Heuristics", Springer-Verlag., 2nd ed., 554 p., 2004.

[21] Constantinou Antoniou : " On–line Calibration for Dynamic Traffic Assignment", Tesis Doctoral, Massachusetts Institute of Technology, 2004.

[22] Meyer-Nieberg S. and H.-G. Beyer. "On the analysis of self-adaptive recombination strategies: First results". In B. McKay et al., editors, Proc. 2005 Congress on Evolutionary Computation (CEC’05), Edinburgh, UK, pages 2341–2348, Piscataway NJ, 2005.

[23] Wu, Chi-Jen, Yang; Kai-Hsiang; Ho, Jan-Ming, “AntSearch: An Ant Search Algorithm in Unstructured Peer-to-Peer Networks”, *IEICE Transactions on Communications*, vol. 89, No. 9, pp. 2300~2308, 2006.

[24] 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.

[25] Glover, F.: "Tabu Search - Part I, 1.st comprehensive description of tabu search". ORSA Journal on Computing,Vol.1,No. 3, pp. 190-206, 1989.

[26] Lawler, E., "Combinatorial Optimization: Networks and Matroids", Dover Publications, Inc. New York., 372p.

[27] Papadimitriou, C., Steiglitz, K.: "Combinatorial Optimization: Algorithms and Complexity", Dover Publications 1998.

[28] Garey M.R., Johnson D.S.: "Computer and Intractability: a Guide to the Theory of NP-Completeness ", WH Freeman, New York, 1979.

[29] Feo T., resende M. :"Greedy Randomized Adaptative Search Procedures ", Journal Global Optimization, No.6, págs. 109-133, 1995.

[30] Resende M.G.C., González Velarde J.L.: "GRASP: Procedimientos de Búsqueda Miopes Aleatorizados y Adapatativos". Inteligencia Artificial, No.19, págs. 61-76, 2003.

[31] Glover,F.,Laguna,M.:"Tabu Search". Kluwer Academic Publishers, 1986.

[32] Díaz F. Glover, M. H. Ghaziri, J. L. González, M. Laguna, and F. T. TTseng. "Optimización Heurística y Redes Neuronales", Paraninfo, Madrid, 1996.

[33] Barr R.S., Golden B.L. Kelly J., Steward W.R., Resende M.: "Guidelines for Designing and Reporting on Computational Experiments with Heuristics Methods", Proceedings of International Conference on Metaheuristics for Optimizations. Kluwer Publishing, Norwell MA, pp. 1-17, 2001.

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

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

[36] 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.

[37] 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.

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

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

[40] Mindlin, Gabriel: "Causas y Azares: La historia del caos y de los sistemas complejos", ISBN 978-987-629-037-1. Buenos Aires, Argentina, 2008.

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

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

[43] 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.

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

[45] Latora, V., Marchiori, M.: "Efficient Behavior of Small World Networks", Physical Review Letters. Vol. 87, No.19, 2001.

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

[47] Costa, L., Rodrigues, F.A., Travieso, G., Villas, P.R.: "Characterization of Complex Networks: A Survey of Measurements", arXiv:cond-mat/05050485v5, 2007.

[48] Cruz Reyes L, Meza Conde E., Gómez Santillán C., Turrubiates López T., Ortega I.: "Statistical selection of relevant features to classify random, scale free and exponential networks", Innovations in Hybrid Intelligence System (HAIS 2007), Vol. 44/2008, pp.454-461. 2007.

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

[50] Hayes, B.: "Graph Theory in Practice: Part I", American Scientist Vol. 88 No. 1,págs. 9 – 13, 2000.

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

[52] Cruz Reyes L., Meza Conde E., Turrubiates López T., Gómez Santillán C., Ortega Izaguirre R.: "Experimental design for selection of characterization functions that allow discriminate among random, scale free and exponential networks" , *Polish Journal of Environmental Studies*. ISSN 1230-1485, Indexed by SCI-Extended, Vol.16, N.5B. Miedzyzdroje, Poland. pp. 67- 71, 2007.

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

[54] 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.

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

[56] Liu, Z., Lai, Y., Ye, N., Dasgupta, P.: Connectivity Distribution and Attack Tolerance of General Networks with Both Preferential and Random Attachments. Physics Letters A. Vol. 303, Issue. 337 – 344. 2003.

[57] Zipf G.K., "Human Behavior and the principle of least effort". Adisson-Wesley Publishing, 1949.

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

[59] 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.

[60] 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.

[61] 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.

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

[63] Dorigo M. y Stützle T., "Ant Colony Optimization″ MIT Press, 2004.

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

[65] 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.

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

[67] Russell S., Norving P., " Inteligencia Artificial / Un enfoque moderno ", Ed. Prentice Hall Hispanoamericana, S.A, México, 960 p., 1996.

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

[69] 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.

[70] Babaöglu O., Meling H. y Montresor A., "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.

[71] 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.

[72] 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.

[73] Eiben, Ángoston Endre; Hinterding, Robert; Michalewicz, Zbigniew: "Parameter Control in Evolutionary Algorithms", IEEE Transactions on Evolutionary Computation, Vol. 3 pp. 124-141, 1999.

[74] Angeline, Peter: "Adaptative and Self-Adaptative Evolutionary Computations", IEEE, Computational Intelligence, pp. 152-163, 1995.

[75] Hinterding R., Michalewicz Z., and A. E. Eiben, "Adaptation in evolutionary computation: A survey," in *Proc. 4th IEEE Conf. Evolutionary Computation*. Piscataway, NJ: IEEE Press, pp. 65–69, 1997.

[76] Smith J.: "Self adaptation in evolutionary algorithms," Ph.D. dissertation, University of the West of England, Bristol, 1997.

[77] Kramer O.: "Self-Adaptive Heuristics for Evolutionary Computation", Springer Verlag, 182 pp. 2008.

[78] Sutton R. y Barto A., "Reinforcement Learning: An Introduction", MIT Press, Cambridge, Massachusetts pp. 300-350, 1998.

[79] Manson, R.L., Gunst R.F., James L.H.: "Statistical Design and Analysis of Experiments with Applications to Engineering and Science", Second Edtion. Wiley – Interscience, 2003.

[80] Montgomery, Douglas. Diseño y Análisis de Experimentos. Limusa Willey. Segunda Edición, 2004 .

[81] Salazar, A.M.A.: "Pronóstico de Demanda por Medio de Redes Neuronales Artificiales (RNA's) en la Industria de Telecomunicaciones", Tesis de Maestría. Universidad Autonoma de Nuevo León. Facultad de Ingeniería Mecánica y Eléctrica. San Nicolás de los Garza, Nuevo León, México, 2005.

[82] Schmidt, S.R., Launsby, R.S.: "Understanding Industrial Designed Experiments", 3rd Edition. Air Academy Press, 1991.

[83] Cohen, P.: "Empirical Methods for Artificial Intelligence", The MIT Press Cambridge, Massachusetts, pp. 4-5, London, England, 1995.

[84] Lemeire, J., Dirkx, E.: "Causal Models for Performance Analysis", 4th PA3CT Symposium, Edegem, Belgica, 2004.

[85] Spirtes, P., Glymour, C., Scheines, R.: "Causation, Prediction, and Search", MIT Press, 2nd edition 2001.

[86] Spirtes, P.: An Anytime Algorithm for Causal Inference. citeseer.ist.psu.edu/385601.html, 2000.

[87] Pérez R., V.: "Modelado Causal del Desempeño de Algoritmos Metaheurísticos en Problemas de Distribución de Objetos". Tesis de Maestría, ITCM, División de Estudios de Posgrado e Investigación, Maestría en Ciencias e Ciencias de la Computación. 2007.

[88] Escobar L., Sánchez D., Vila A.: Relaciones Causales en Reglas de Asociación. XII Congreso Español sobre Tecnologías y Lógica Fuzzi, Universidad de Jaén, 2004.

[89] Carnegie Mellon's University. Open Learning Initiative (OLI). http://www.cmu.edu/oli/.

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

[91] Morales E., " Aprendizaje Reforzado", Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), México., 2009.

[92] Cooperative Association for Internet Data Analysis.: http://www.caida.org/home/

[93] Turrubiates López T., Goméz Santillan C., Cruz Reyes L., Ortega Izaguirre R., Meza Conde E.: " Diseño Experimental para la Selección de Funciones de Caracterización que Permitan Discriminar entre Redes Aleatorias, de Libre Escala y Exponenciales", *In Proceedings of the 14th International Congress on Computer Science Research*, Orizaba, México. 2007.

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

[95] 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.

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

[97] Lu J. y Callan J., "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.

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

[99] 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.

[100] 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.

[101] John H. Holland. Adaptation in Natural and Artificial Systems. MIT Press, 1992.

[102] García S., Molina D., Lozano F., Herrera F., *A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real ParameterOptimization*, Journal of Heurisatics, Springer Netherlands, Netherlands 2008.

[103] Lowry Richard, *The Wilcoxon Signed-Rank Test*, Vassar Stast: Web site for statistical computation, N.Y. USA, 2009.

[104] Chris Wild and George Seber, *Wilcoxon Rank-Sum Test*, Published by John Wiley & Sons, (http:// www. stat. auckland.ac.nz/ ~wild/ChanceEnc/ Ch10. wilcoxon.pdf), New York, 2009.

[105] Novaes De Santana, C.: "Analise da Pluviometria do Nordeste Brasileiro Segundo Modelagem em Redes", Monografía, 2005.

Glosario de Términos

A

ACO	Por sus siglas en inglés Ant Colony Optimization. Metodología basada en el comportamiento de las hormigas.
ACS	Por sus siglas en inglés Ant Colony System. Algoritmo metaheurístico para solución de problemas de direccionamiento.
AdaNAS	Por sus siglas en inglés, Adaptative Neighboring Ant Search. Algoritmo metaheurístico de búsqueda adaptativa.
Afinación de parámetros	Ofrecer una configuración inicial estática durante toda la ejecución del algoritmo.
Afinación manual	Esta se lleva acabo a prueba y error. Seleccionando aquella que haya dado mejores resultados.
Afinación por meta-evolución	Es una de las técnicas menos usadas ya que trata de afinar un algoritmo a través de otro algoritmo metaheurístico.
Agente	Son los elementos principales del sistema multiagente.
Agente inteligente	Proceso computacional capaz de realizar tareas de forma autónoma y que se comunica con otros agentes para resolver problemas.
Algoritmo	Lista bien definida, ordenada y finita de operaciones que permite hallar la solución a un problema. Dado un estado inicial y una entrada, a través de pasos sucesivos y bien definidos se llega a un estado final, obteniendo una solución.
Algoritmo determinístico	Algoritmo completamente predictivo. Dicho de otra forma, si se conocen las entradas del algoritmo siempre producirá la misma salida, y la máquina interna pasará por la misma secuencia de estados.
Algoritmo de colonia de hormiga	Técnica probabilística utilizada para solucionar problemas de cómputo. Este algoritmo está inspirado en el comportamiento que presentan las hormigas para encontrar las trayectorias desde la colonia hasta el alimento.
Algoritmos de búsqueda	Están diseñados para localizar un elemento concreto dentro de una estructura de datos. Consiste en solucionar un problema de existencia o no de un elemento determinado en un conjunto finito de elementos, es decir, si el elemento en cuestión pertenece o no a dicho conjunto, además de su localización dentro de éste.

Algoritmo Random Walk	Es una técnica de búsqueda ciega donde los nodos de la red no poseen información sobre la localización o contenido de los recursos requeridos, a menos de que el recurso se encuentre en el mismo nodo.
Algoritmo Flooding	Es una técnica de búsqueda clásica en redes punto a punto, en la cual el nodo que emite la búsqueda lo hace hacia todos sus vecinos al mismo tiempo, y los vecinos hacia todos sus vecinos, y asi sucesivamente hasta cumplir con la condición de paro.
Algoritmos metaheurísticos	Son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Éstas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos.
Algoritmos heurísticos	Procedimientos simples a menudo basados en el sentido común que se supone que obtendrán una buena solución (no necesariamente óptima) a problemas difíciles de un modo sencillo y rápido.
Algoritmo de optimización	Son los algoritmos que usan las estrategias de solución aplicadas a problemas combinatorios.
Ajuste de parámetros	Selección de los valores de los parámetros que regulan el comportamiento de los algoritmos.
Ajuste global	Este se lleva a cabo cuando los valores de los parámetros son calculados usando promedios. Evalúa el desempeño general del algoritmo, y no garantiza que los valores propuestos sean los mejores a lo largo de toda la ejecución.
Ajuste local	Este se lleva a cabo cuando los valores de los parámetros son calculados usando solo el dato y sus vecinos inmediatos.
Algoritmo genético	Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una Selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.
AntSearch	Algoritmo metaheurístico con el objetivo de mejorar el proceso de búsqueda en términos del tráfico generado en la red.

Aprendizaje	Proceso que implica un cambio duradero en la conducta, o en la capacidad para comportarse de una determinada manera, que se produce como resultado de la práctica o de otras formas de experiencia.
Aprendizaje reforzado	Define la manera de comportarse de un agente a un tiempo dado en un tiempo exacto. Puede verse como un mapeo entre los estados del ambiente que el agente percibe y las acciones que toma, cuando se encuentra en esos estados.

B

Búsquedas ciegas	En estas se propagan la consulta a un número suficiente de nodos para satisfacer la consulta y no se cuenta con información que ayude a guiar la consulta.
Búsquedas informadas	En esta técnica se toma información acerca de la ubicación de documentos la cual ayuda a guiar la consulta.
Búsqueda en amplitud inteligente	Es un algoritmo de búsqueda informada en el cual los nodos almacenan consultas exitosas para guiar la consulta. Para ejecutar la consulta inunda por niveles hasta encontrar la información.
Búsqueda adaptativa probabilística	Usa la retroalimentación de búsquedas anteriores para realizar búsquedas futuras eficientes. Cada par 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.

C

Causalidad	Es el término que de manera general relaciona las acciones con sus consecuencias, o las causas y los efectos.
Consulta	Son los cuestionamientos que lanzan las personas en búsqueda de información, estos son hechos a través de palabras clave.
Configuración factible	Es aquella que cumple con todas las restricciones o condiciones del problema y genera el mejor valor de la función objetivo.
Control adaptativo	Se realiza cuando existe alguna forma de retroalimentación del pasado que determina un cambio en sentido y magnitud del parámetro.

Control auto-adaptativo	Cuando los parámetros son codificados dentro de la estructura del algoritmo metaheurístico, de esta forma las mejores soluciones se encuentran asociadas a una configuración paramétrica la cual también irá evolucionando así como la solución.
Calibra	Software para ajuste global de parámetros, solo puede ajustar máximo 5 parámetros, a través de experimentos factoriales.
Congestionamiento	El congestionamiento se da cuando los medios de comunicación están saturados de mensajes que envían o reciben los nodos.
Configuración inicial	Son los valores iniciales que toman los parámetros para iniciar un proceso.
Comunidades virtuales	Son grupos que comparten intereses comunes, los cuales se conectan mediante algún software de comunicación, con el objetivo de compartir información, música, fotos, etc.
Coeficiente de dispersión del grado	Por sus siglas en inglés, DDC. Mide la dispersión entre el grado de un nodo *i* y sus vecinos.
Control de parámetros	Ésta actividad monitorea los cambios en el ambiente al mismo tiempo que considera el estado actual del algoritmo.
Configuraciones factibles	Son las que cumplen con todas las restricciones, y generan el mejor valor de la función objetivo.
Control Determinístico	Toma lugar cuando se asignan valores a través de una regla determinista que modifica su valor sin considerar ninguna retroalimentación.

D

Distribución de probabilidad	Es conocida también como distribución del grado.
Distribución del grado	Es la distribución de probabilidad del grado de una red y esta basada en información global.
Definición formal	Es la definición del problema de optimización, el cual esta sujeto a un conjunto de restricciones.
Desempeño del algoritmo	Este puede ser parcial o total del algoritmo. El desempeño parcial se calcula en cada iteración del algoritmo y el final es cuando se cumplen las condiciones de paro.

Distribución Zipf	Es una ley empírica formulada usando estadística matemática y esta relacionada con el hecho de que muchos tipos de datos estudiados en la física y las ciencias sociales pueden ser aproximados con una distribución zipfiana.
Diseño de experimentos	Por sus siglas en inglés DOE, son un conjunto de herramientas estadísticas a través de las cuales se analizan los datos con el objetivo de determinar el ajuste óptimo global. Es una estrategia de afinación de parámetros global.
Diseño factorial	Es una estrategia de afinación de parámetros global. Estudia los efectos o la influencia de dos o más factores.

E

Efecto de un factor	Se conoce como el cambio en la respuesta producida por un cambio en el nivel del factor
Ecuación estructural	Es un conjunto de ecuaciones lineales de la forma $x_i = f_i(pa_i, u_i)\ i = 1,...,n$ donde cada pa_i son los padres de x_i para un conjunto de variables X y donde el u_i representa el error (o "perturbaciones") debido a factores omitidos.

F

Funciones de caracterización	Tienen el objetivo de encontrar y destacar las propiedades que caracterizan la estructura y el comportamiento de la red compleja, ofreciendo herramientas para medir propiedades; y así crear modelos de redes que ayuden a entender el significado de esas propiedades.
Función objetivo	Representa o mide la calidad de las decisiones (usualmente números enteros o reales) puede ser de maximización o minimización.
Filtro de Kalman	Es un conjunto de ecuaciones matemáticas que proveen una solución recursiva eficiente del método de mínimos cuadrados.

G

Grado de un vértice	Es el número de conexiones asociadas a un vértice.
Grafos	Es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

H

Heterogeneidad topológica	Diferencias en el grado entre los nodos. Por ejemplo: redes scale-free, redes exponenciales, entre otras.

I

Información global	Es toda la información de la red.
Información local	Es la información relacionada con un punto de la red, y el conjunto de sus vecinos.
Internet	Es un conjunto descentralizado de redes de comunicación interconectadas, que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.
Instancias reales	Son un conjunto de datos tomados de comunidades reales que se encuentran en la Internet.
Instancias artificiales	Son un conjunto de datos generados que representan información que puede ser topológica, de repositorios o de consultas. Estas instancias son creadas a partir de características de las instancias reales.
Inundación de mensajes	Es la saturación de los medios de comunicación con información enviada en forma de mensajes a otros puntos de la red.
Inundaciones controladas	Es una forma de controlar o evitar la saturación de los medios de comunicación con información enviada en forma de mensajes a otros puntos de la red.

L

Lookahead	Método de exploración clásico, donde se explora a los vecinos inmediatos para obtener información rápidamente.

M

Métricas topológicas	Son medidas que obtienen información de la topología de la red, para evaluarla en una instancia de tiempo.
Modelado Causal	Es una representación generalizada del conocimiento, el cual es obtenido al encontrar dependencias en los datos, que impliquen relaciones de causa y efecto.

N

Nodo	Es un punto o vértice de la red.
NAS	Neighboring Ant-Search es un algoritmo de colonia de hormigas que fue diseñado para resolver SQRP usando estrategias de ajuste global.
Nivel de un factor	Son valores fijos que se quieren estudiar en un factor. Un factor puede tener dos o mas niveles.

O

Optimización combinatoria	Es el estudio matemático del acomodo de un grupo, el ordenamiento o la selección de objetos discretos, usualmente finito en número cuando las combinaciones posibles son demasiadas para ser analizadas una por una.

P

Parámetros	Es el conjunto de variables que representan las características del problema.
P2P	Por sus siglas en inglés peer-to-peer. Es un sistema distribuido donde todos los nodos son iguales en términos de funcionalidad y las tareas que desarrollan en el sistema
Problema NP-duro	Es un problema difícil de resolver ya que no existe un algoritmo que lo resuelva en tiempo polinomial.
Problemas de optimización combinatoria	Son aquellos que consisten en encontrar la mejor solución dentro de un gran número de soluciones potenciales, para conseguir ciertos objetivos.
Proceso de decisión de Markov	La caracterización de la problemática de encontrar el balance entre exploración y explotación está dada por procesos de decisión de Markov.

R

Redes complejas	Son conjuntos de muchos nodos conectados que interactúan de alguna forma. También es conocida como red no uniforme.
Repositorio	Son bancos de datos que están situados en las máquinas, pueden ser discos duros o memorias.
Restricciones	Son las condiciones que se tienen que cumplir para lograr un objetivo.

Redes aleatorias	En estas las aristas entre los nodos son establecidas aleatoriamente y la mayoría de los nodos tiene aproximadamente el mismo grado.
Redes Scale-free	Presentan la característica invariante de que un número reducido de nodos que conforman la red tienen grado muy alto y una gran cantidad de nodos tienen un grado pequeño.
Replicas	Repetición del experimento básico. Permite calcular el error experimental.

S

SQRP	Problema del direccionamiento de consultas semánticas. Es el problema de búscar información textual a través de palabras claves en la Internet.
Sistema	Conjunto de elementos relacionados entre si con un objetivo común.
Sistemas complejos	Tienen comúnmente un gran número de elementos, los cuales actúan de acuerdo a reglas que pueden cambiar a través del tiempo, es decir la conectividad de los elementos y sus roles son variables.
SemAnt	Algoritmo metaheurístico basado en colonias de hormigas, para solucionar problemas de búsqueda de contenido.
Sistema multiagente	El estudio de la inteligencia que surge de las interacciones entre muchos agentes.

T

Topología	Se refiere al patrón de conexiones que forman los nodos de la red.
Trayectoria	Es un unión de aritas o lados, los cuales están unidos y forman una trayectoria.
Tablas de feromonas	Es una memoria donde las hormigas van depositando sus rastros de las mejores rutas encontradas.
Time-To-Live	Por sus siglas en inglés TTL. es el tiempo de vida de cada uno de los agentes del algoritmo de búsqueda.
Teoría de la complejidad computacional	Es la parte de la teoría de la computación que estudia los recursos requeridos durante el cálculo para resolver un problema.

TETRAD	Es un software gratuito desarrollado con el objetivo de crear, simular datos, estimar, probar, predecir y buscar modelos estadísticamente causales.

V

Valor óptimo	Es el mejor valor encontrado para un parámetro.
Variables exógenas	Son aquellas cuyos valores son determinados por factores fuera del modelo.
Variables endógenas	Tienen valores descritos por un modelo de ecuaciones estructurales.