A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks

Sistema de Colonia de Hormigas Autoadaptativo para el Problema de Direccionamiento de Consultas Semánticas en Redes P2P

Claudia Gómez Santillán[1,2], **Laura Cruz Reyes**[2], **Eustorgio Meza Conde**[1], **Elisa Schaeffer**[3] **and Guadalupe Castilla Valdez**[2]

[1]Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada del Instituto Politécnico Nacional (CICATA- IPN). Carretera Tampico-Puerto Industrial Altamira, Km.14.5. Altamira,Tamps.,México.

[2]Instituto Tecnológico de Ciudad Madero (ITCM). 1ro. de Mayo y Sor Juana I. de la Cruz s/n CP. 89440, Tamaulipas, México.

[3] Facultad de Ingeniería Mecánica y Eléctrica (FIME-UANL), Avenida Universidad s/n. Cd. Universitaria, CP. 66450, San Nicolás de los Garza, N.L. México.

cggs71@hotmail.com, lcruzreyes@prodigy.net.mx, emezac@ipn.mx, elisa@yalma.fime.uanl.mx, gpe_cas@yaoo.com.mx.

Article received on July 17, 2009; accepted on November 05, 2009

Abstract

In this paper, we present a new algorithm to route text queries within a P2P network, called Neighboring-Ant Search (NAS) algorithm. The algorithm is based on the Ant Colony System metaheuristic and the SemAnt algorithm. More so, NAS is hybridized with local environment strategies of learning, characterization, and exploration. Two Learning Rules (LR) are used to learn from past performance, these rules are modified by three new Learning Functions (LF). A Degree-Dispersion-Coefficient (DDC) as a local topological metric is used for the structural characterization. A variant of the well-known one-step Lookahead exploration is used to search the nearby environment. These local strategies make NAS self-adaptive and improve the performance of the distributed search. Our results show the contribution of each proposed strategy to the performance of the NAS algorithm. The results reveal that NAS algorithm outperforms methods proposed in the literature, such as Random-Walk and SemAnt.

Keywords: Search Process, Internet, Complex Network, Ant Colony System, Local Environment, Neighbor.

Resumen

En este documento, proponemos un nuevo algoritmo para ruteo de consultas textuales dentro de una red P2P, llamado Neighboring-Ant Search (NAS). El algoritmo está basado en la metaheurística Ant Colony System (ACS) y el algoritmo SemAnt. Además, NAS está hibridizado con estrategias del ambiente local de aprendizaje, caracterización y exploración. Dos reglas de aprendizaje (LR) son usadas para aprender del rendimiento pasado, esas reglas son modificadas por tres Funciones de Aprendizaje (LF). Un Coeficiente de Dispersión del Grado (DDC) es usado 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 una capacidad auto-adaptativa que mejora el rendimiento de la búsqueda distribuida. Los resultados experimentales muestran la contribución de cada estrategia propuesta para el rendimiento del algoritmo NAS. Estos resultados revelan que el algoritmo NAS obtiene mejores resultados que los algoritmos propuestos en la literatura existente tales como Random-Walk y SemAnt.

Palabras Clave: Proceso de Búsqueda, Internet, Redes Complejas, Sistema de Colonia de Hormigas, Ambiente Local, Vecindad.

1 Introduction

The popularity of peer-to-peer (P2P) systems is motivated by the benefits offered to the end user. In contrast to the traditional Web, a P2P system does not need to rely on any dedicated centralized servers, which makes P2P networks reliable and fault tolerant. Hence a user can easily join a network and leave when necessary, giving rise to

unstructured self-organizing networks. Due to the unstructured nature, these applications often employ a flooding-based data search mechanism, which generates severe communication overhead and limits the growth of P2P systems. These systems together with the underlying Internet are considered complex dynamic distributed networks for their size and constantly evolving interconnectivity. In complex dynamic distributed networks, global knowledge collection is not a feasible approach to handle queries on shared resources. In these circumstances, each query needs to determine locally its behavior, without resorting to a global control mechanism.

Digital technologies and new standards make it possible to produce music, movies, pictures, images, and textual information in a digital form with reasonable quality. Internet is the essential, cheapest and most convenient way to manage digital files, to sell, buy, and share digital content. Such a popular application like the World Wide Web (WWW) has not been convenient enough to share files. Sharing content on the WWW requires infrastructure (a HTTP server) and makes it difficult for individual users to share their files in an easy and independent way. The files published on the WWW are available for search only after their respective sites are crawled and indexed by existing centralized search engines. Since such operations may take a significant amount of time, users have no direct control over the published files to make them available for immediate search. These disadvantages make the use of traditional applications for file sharing complicated [13].

In 1999 the P2P systems arose as a response to the increased demand for file sharing. These systems are formed by interconnected peers that offer their resources to other peers within the network. The participants connect and disconnect constantly, producing changes in the structure of the network. Due to the unstructured nature, applications mainly employ flooding-based data search mechanisms. Flooding-based search generates vast amounts of Internet traffic that limits the growth of peer-to-peer systems. The obvious problems that appeared with the growing popularity of peer-to-peer file sharing systems are two: a) the poor accuracy of the information search and b) the traffic caused by the flooding-based search. Measurements have shown that peer-to-peer systems are the main source of Internet traffic [13],[10],[11], making the development of new approaches to avoid flooding an important research challenge.

The Semantic Query Routing Problem consists in each peer deciding, based on a keyword in the query, to which neighboring peer to resend the text query. To avoid flooding, the goal is to maximize the number and the quality of query results, while minimizing the use of the resources of the network. Existing approaches for query routing in P2P networks range from simple broadcasting techniques to sophisticated methods [13],[10],[11]. Due to the fact that P2P networks are based on non-central authorities and high-growing dimension, the challenge for query routing is the development of methods that adapt themselves to dynamic environments. Such intelligent adaptation must be based only on the local knowledge of each peer. Among the intelligent mechanisms successfully applied to several problems in distributed systems, lie the ant-colony methods. The metaheuristic of Ant Colony System, proposed by Dorigo [12], solves optimization problems based on graphs. Many ant algorithms have been specifically designed for handling routing tables in telecommunications. However, there are very few ant algorithms for handling routing tables in the Semantic Query Routing Problem [10].

In this paper, we present a novel algorithm for distributed text query routing. The algorithm, called the Neighboring-Ant Search (NAS), is based on two well-known ant algorithms: the Ant Colony System [7] and the SemAnt [10]. Additionally, NAS is hybridized with local strategies of learning, characterization, and exploration. Three functions are employed to learn from past performance. The first function is used to evaluate the NAS performance based on the found and expected results. The second function qualifies the NAS performance based on the available time for the searching. The third function qualifies each peer depending on the distance towards a previously found resource. A topological metric based on the number of connections of each peer is used for the structural characterization. An adaptation of the well-known one-step Lookahead search is used to explore the neighbor peers of the nearby environment [11]. These three strategies contribute with the main goal of the application which is to find a greater amount of resources in the least amount of time.

2 Background

In order to place the research in context, this section is divided in four parts. The first part defines, explains, and models the peer-to-peer complex networks. The second part explains and formally defines the Semantic Query Routing Problem. The third part explains and defines the Lookahead exploration and the Degree-Dispersion-Coefficient local metric. The last part refers to the Random-Walk algorithm.

2.1 Peer-to-Peer Complex Networks

P2P systems are formed by interconnected peers that offer their resources to other peers within the network. Hence a P2P network is a distributed system that can be modeled as a *graph*. Each peer in the network is represented by a *node* (also called a *vertex*) of the graph. The interactions among the peers are represented by the *connections* (also called the *edges*) of the graph. In P2P networks, the nodes are capable of self-organization for the purpose of sharing resources, without requiring the mediation or support of a server or centralized authority [2].

More so a P2P system, together with the underlying communication network (typically Internet), forms a complex system that requires autonomous operation through mechanisms of intelligent search [10]. Amaral [1] published a classification for different types of systems and categorized them into simple, complicated, and complex systems. *Complex systems* are those systems that have (typically) a very large number of components, the connections among them may evolve over time, and the roles of the components may vary. In many studies, complex systems are modeled as networks, giving rise to the concept of *complex networks* and, within this context, the term P2P Complex Networks.

One of the main motivations for modeling systems as P2P complex networks is the flexibility and generality of the abstract representation that allows handling properties such as dynamic topology in a natural way. A recent methodology for modeling complex systems, called Autonomous Oriented Computing (AOC), was proposed by Liu [12]. AOC consists in the formulation of a tuple that represents the general model of the system, i.e.: $<\{e_1, e_2,\ldots,e_i,\ldots,e_N\}, E,\Phi\}>$, where $\{e_1,e_2,\ldots,e_i,\ldots,e_N\}$ is a subset of size N of autonomous *entities*, E is the *environment* in which the entities reside, and Φ is the *objective function* of the global system. Each entity is a basic element with a well-defined goal within the complex system. To achieve its goal, it has attributes that describe its behavior rules, current state, and an evaluation function. We use the AOC notation to model P2P systems as follows: *i*) the entities are the *agents* that surf in the network with the objective of finding resources; *ii*) the environment is the P2P network and *iii*) the objective function is to find the maximal set of resources in the shortest possible time. A more detailed description of our querying system based on intelligent agents is given in Section 4.

2.2 The Semantic Query Routing Problem

The problem of locating textual information in a P2P network over the Internet is known as *Semantic Query Routing Problem* (SQRP). The goal of SQRP is to determine the shortest paths from a node that issues a query to nodes that can appropriately answer it (by providing the requested information). The query traverses the network moving from the initiating node to a neighboring node and then to a neighbor of the neighbor and so forth until it locates the requested resource (or gives up in its absence). This type of propagation is known as *flooding* and it is the most common search strategy in P2P networks. Algorithms for SQRP must consider several factors, ranging from hardware and software characteristics to user behavior. Due to its complexity [10],[1],[12],[6], solutions proposed to SQRP typically limit to special cases. Yang et al. [6] propose AntSearch that controls the quantity of flooding using a simple learning technique, whereas Michlmayr [10] proposes the SemAnt algorithm for learning from user behavior.

Formally, SQRP is defined with the description of an *Instance* and an *Objective* that must be satisfied by a solution algorithm such as the ant-based algorithm proposed in this work. ***Instance***: given a P2P network represented by a graph T, a set of *contents* distributed in the nodes called repositories R, and a set of semantic *queries* Q launched by the nodes. Each query can be launched from any node in the time T_0, $\forall T_0 \in \mathbf{Z}$, assuming a discrete-time process. The node that originally launches a query (or receives a query from other node) in time T_0+i, $\forall i \in \mathbf{Z}^+ \cup \{0\}$, can locally process the query and/or forward a copy of the query to a set of nearby nodes at time $T_0+(i+1)$. The query processing finishes when a stop condition has been satisfied, whether either the maximal quantity of

resources has been found or th time-to-live value specified for the query is reached. ***Objective***: find a set of paths among the nodes launching the queries and the nodes containing the resources, such that the quantity of found resources is maximized and the quantity of steps given to find the resources is minimized.

2.3 Lookahead and the Degree-Dispersion-Coefficient

A node *i* is a *neighbor* of a node *j* if the two nodes *i* and *j* are connected by an edge (*i*, *j*) in the graph that models the system. The set of all neighbors of a node *i* is denoted by $\Gamma(i)$. In an undirected simple graph, that is, a graph in which the edges are considered bidirectional communication channels and each pair of nodes may be connected by at most one edge, the *degree* of a node is the number of neighbors it has.

A well-known strategy based on local information is the one-step *Lookahead* exploration method [11]. Lookahead is employed in algorithms to examine neighboring resources up to a certain level before deciding how to proceed with the search. In this work, we assume that each node knows the resources of the first-level neighbors.

In order to locally focus the exploration strategy we use the *Degree-Dispersion-Coefficient* (DDC) function [14]. DDC is based on local information that through the dispersion of the degree of a node measures the differences between the degree of a node and the degrees of its neighbors, this is:

$$DDC(i) = \frac{\sigma(i)}{\mu(i)}, \qquad (1)$$

Where the degree variation of the set $\{i \cup \Gamma(i)\}$ $\sigma(i) = \sqrt{\frac{\sum_{j \in \Gamma(i)} \left[k_j - \mu_i\right]^2 + \left[k_i - \mu_i\right]^2}{N_i}}$, the average degree found in the set $\{i \cup \Gamma(i)\}$ $\mu(i) = \frac{\sum_{j \in \Gamma(i)} \left[k_j\right] + k_i}{N_i}$, N_i is the number of nodes in $\{i \cup \Gamma(i)\}$, and, k_i and k_j are the degree of nodes *i* and *j* respectively.

2.4 Random-Walk Algorithm

The Random-Walk (RW) search algorithm is a *blind* search technique where the nodes of the network possess no information on the location or contents of the requested resource, unless the resource resides in the node itself . Let *G* be a graph that models the network and *v* a node in *G*. A *T-hop Random-Walk* from *v* in *G* is a sequence of dependent random variables $X_0,\ldots, X_T$ defined as follows: $X_0 = v$ with probability 1 and for each $i = 1,\ldots,T$, the value for X_i is selected uniformly at random among the nodes in $\Gamma(X_{i-1})$, that is, among the neighbors of the node of the preceding step. In others words, a Random-Walk begins at a node and on each step moves to a neighbor of the current node, until it arrives to a node that meets the goal. In our P2P model, the goal is met when a node contains the requested resource [3]. Optionally, one could include the DCC function into the Random-Walk algorithm. A simple modification to include such a structural preferentiality is to choose uniformly at random two neighbors, calculate their DCC values, and move on to the neighbor with higher DCC.

3 Related Work

The success of SQR algorithms in P2P file-sharing networks lies on search mechanisms that have received special attention [10][6]. We summarize here some of the most relevant proposals for semantic query routing using Ant-Colony Algorithms. These algorithms are based mainly on a learning structure named *pheromone*. This structure is used for establishing indirect communication between ants about their past performance. The pheromone table (τ) is used as a query routing table.

Michlmayr [10] proposes a distributed SQRP algorithm for P2P networks called SemAnt and includes an evaluation of the parameter configuration that affects the performance of the algorithm. Michlmayr aims for an

optimal ratio between network traffic and quantity of results and compares the performance of SemAnt with the Random-Walk algorithm using the following metrics: *i*) *Resource Usage* define as the number of links traveled for each query within a given period of time, *ii*) *Hit Rate* defines as the number of resource found for each query within a given period of time, and *iii*) *Efficiency* defined as the ratio of resource usage to hit rate. Dividing the number of links traveled by the number of resources found, gives the average number of links traveled to find one resource.

Yang et al. [6] propose an algorithm called *AntSearch* for non-structured P2P networks. In AntSearch, each pair of nodes stores information on the level of success of past queries as well as on the pheromone levels of the immediate neighbors. The work of Yang et al. was motivated by the need to improve search performance in terms of the traffic in the network and the level of information recovery. Yang et al. use three metrics to measure the performance of the AntSearch. One is the *number of searched files* for a query with a required number of results, given *N as*: a good search algorithm should retrieve the number of results over but close to *N*. The second one is the *per result costs* that defines the total amount of query messages divided by the number of searched results; this metric measures how many average query messages are generated to gain a result. Finally, *search latency* is defined as the total time taken by the algorithm.

Di Caro and Dorigo [4] propose *AntNet* that is designed for packet-switched networks. The ants collaborate in building routing tables that adapt to current traffic in the network, with the aim of optimizing the performance of the entire network. AntNet uses global information on the nodes of the network in order to choose the destination nodes. Di Caro and Dorigo focused on standard metrics for performance evaluation, considering only sessions with equal costs, benefits and priority and without the possibility of requests for special services like real-time. In AntNet framework, the main measures are: *i*) *throughput* correctly delivered bits/sec, *ii*) *delay distribution for data packets* (sec), and *iii*) *network capacity usage* for data and routing packets, expressed as the sum of the used link capacities divided by the total available link capacity.

Most relevant aspects of former works have been incorporated into the proposed NAS algorithm. The framework of AntNet algorithm is modified to correspond to the problem conditions: in AntNet the final addresses are known, while NAS algorithm does not know a priori the nodes where the resources are located. On the other hand and different to AntSearch, the SemAnt algorithm and NAS are focused on the same problem conditions, and both use algorithms based on AntNet algorithm. However, the difference between the SemAnt and NAS is that SemAnt only learns from past experience, whereas NAS takes advantage of the local environment. This means that the search in NAS takes place in terms of the classic local exploration method of Lookahead [11], the local structural metric DDC, and three local functions of the past algorithm performance. These three performance functions are described in the following section.

4 Classical Learning Rules Modified by the Proposed Learning Functions

The classic ACS algorithm is formed by two rules – selection and update – that allow the convergence of the system towards better results. Modifications of these rules were made with the goal of adapting the ACS algorithm to the SQRP. Also, new functions were added such as the DDC topological metric, the Lookahead method, and three Learning Functions: hit importance, time-to-live of the agent, and distance towards a resource that improve the performance of the system in terms of the objective of the problem. The additions of these functions improved the system performance and are explained in detail in this section.

4.1 Learning Functions

This section describes the new proposed Learning Functions (LF) and so is divided into three parts. The first function defines the hit importance, the second function defines the time-to-live importance, and the last function defines the distance importance. With these three LF, introduced in the classical learning rules, the agents search the resources.

Hit Importance Function (HIT), shown in Eq. (2), qualifies the performance of the search agent *k* (that is, an ant that represents a query), based on the found result ($results_k$), and the expected result (*maxResults*):

$$HIT = \frac{results_k}{maxResults}, \quad (2)$$

where $results_k$ represents the amount of results found by the ant k and *maxResults* is the amount of results requested. The value of the *HIT* function is applied in the global updating function, Eq. (8), with the purpose of guiding the queries toward routes that provide the greatest possible amount of found resources. This *HIT* value is registered in each node of the path traversed by the search ant, as shown in Figure 1.

Importance of Time-to-live Function (ITL_HOP), qualifies the performance of the search agent *k*, based on the time-to-live used to find one resource (TTL_k) and the maximum time-to-live originally assigned:

$$ITL\_HOP = \frac{maxTTL}{2 \times TTL_k}, \quad (3)$$

where TTL_k is the partial time-to-live of the ant k until the moment, and *maxTTL* is the maximum time-to-live assigned to an ant for a query. These time measures are given in terms of the number of hops and correspond respectively with the given steps and the maximum steps allowed to each ant. The result is applied into the global updating rule, Eq. (8), with the overall goal of decreasing the time-to-live necessary to find a set of resources. The *ITL_HOP* value is registered in each node of the path traversed by the ant until the node with the last resource is found, as shown in Figure 2.

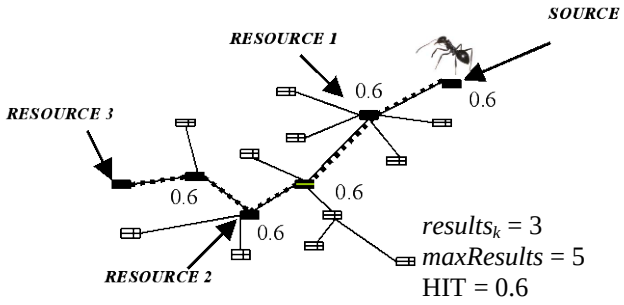


Fig. 1. Example of the calculation of the HIT function

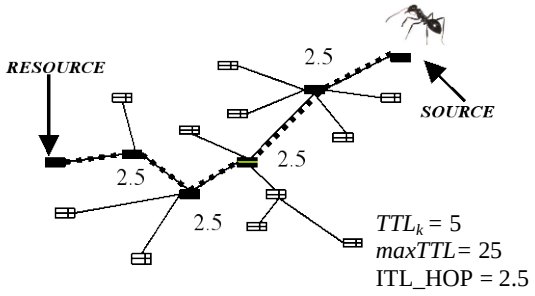


Fig. 2. Example of the calculation of the ITL_HOP function

Importance of Distance Function (ID_HOP), for the agent *k*, shown in Eq. (4), qualifies each node *s*, which is a neighbor of the current node *r*, depending on the distance towards a previously found resource *t*:

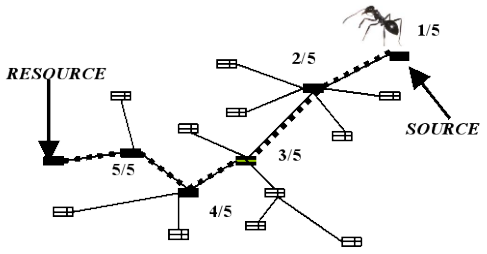
$$ID\_HOP_{r,s,t} = \left(\frac{h_k}{h_{r,s,t}} \right)^{-1} \quad \forall r, s \in route \text{ of } k \quad (4)$$

when r is the current node, s is the evaluated node, t is the found resource, k is the ant agent, h_k is the total number of steps taken by the agent, and $h_{r,s,t}$ is the number of hops from the source to an evaluated resource node. Once an ant k generates a route to a resource t, the function $ID\_HOP_{r,s,t}$ is applied to each node belonging to this route to increase its importance in terms of the distance to a found resource node with respect to the length of the route. This function is obtained through the inverse of the total number of hops h_k made by the ant on the route to the resource found,

divided by the relative number of hops $h_{r,s,t}$ from the source to the node evaluated *s*, which is a neighbor of the current node *r*, as is shown in Figure 3.

4.2 Classical Learning Rules (LR)

This section describes the modifications made to the two classical learning rules, originally proposed by Dorigo [7] in the ACS algorithm. The first rule is called *state transition;* with this rule the next node to be visited is selected. The transition rule uses two strategies: *exploitation* and *exploration*. The second rule is called *updating*; with this rule the ant updates the nodes in the traversed path. The updating rule uses two strategies: local and global updating of the pheromone. All strategies are used to feedback the system on successful routes.



$h_k = 5$
$h_{r,s,t}$= 1, for *r*=1 and *s* =2
$h_{r,s,t}$= 3, for *r*=3 and *s* =4
ID_HOP$_{r,s,t}$ = 1/5, for *r*=1 and *s* =2
ID_HOP$_{r,s,t}$ = 3/5, for *r*=3 and *s* =4

Fig. 3. Example of the calculation of the *ID_HOP* function

The modified *state transition* rule of NAS is formulated by Eqs. (5) and (6). This:

$$s = \begin{cases} \arg\max_{L_n \forall n \in \{0,|L|\}} \left\{ \left[\tau_{r,L_n,t} \right] \cdot \left[DDC_{L_n} + ID\_HOP_{r,L_n,t} \right]^{\beta} \right\}, & \text{if } q \le q_0 \text{ (exploitation)} \\ S, & \text{otherwise (biased exploration),} \end{cases} \quad (5)$$

where *r* is the current node where the ant *k* is located, *u* belongs to the set of neighboring nodes of *r*, V_k is the set of nodes visited by the ant *k*, τ is the pheromone table, *t* is the searched resource, β is the parameter that determines the relative importance between the pheromone and the DDC with *ID_HOP*, *q* is a random number, and q_0 determines the relative importance of exploitation versus exploration. In case that $q \le q_0$, the exploitation strategy is selected: it selects a node that provides a greater amount of pheromone and better connectivity with smaller numbers of hops toward a resource. Otherwise the exploration strategy, Eq. (6), is selected:

$$S = f(p_{r,u,t}), \quad p_{r,u,t} = \frac{\left[\tau_{r,u,t} \right] \cdot \left[DDC_u + ID\_HOP_{r,u,t} \right]^{\beta}}{\sum_{\forall i \in \Gamma(r) \wedge i \notin V_k} \left[\tau_{r,i,t} \right] \cdot \left[DDC_i + ID\_HOP_{r,i,t} \right]^{\beta}}, \quad (6)$$

where *S* selects a node applying a random selection function *f* based on the well-known roulette-wheel selection to favor nodes with higher connectivity, stronger pheromone trail, and a shorter distance to a requested resource. This exploration strategy stimulates the ants to search for new paths. Note that the pseudorandom variable is modified by the DDC topological metric and the *ID_HOP* learning function.

The ACS *updating* rule is composed of both local and global updating. Hence the modified *updating* rule in this work is given in Eqs. (7) and (8). The *local update* strategy of NAS is formulated by:

$$\tau_{r,s,t} \leftarrow (1-\rho) \cdot \tau_{r,s,t} + \rho \cdot \tau_0, \quad (7)$$

where *r* is the current node where the ant *k* is located, *s* is the current neighboring node where the ant is going to move, *t* is the searched resource, τ is the pheromone table where the ant does the local updating, τ_0 is the initialization value of the pheromone, and ρ is the local evaporation factor of the pheromone. Each time an ant decides to move towards a node by the state transition rule, some pheromone is deposited at each node that has been visited to establish a trend towards the most frequently visited nodes.
On the other hand, the modified *global update* strategy, given by Eq. (8), is computed each time that a resource is found and is applied to each node belonging to the route that lead to the discovery of the resource:

$$\tau_{r,s,t} \leftarrow (1-\alpha)\cdot\tau_{r,s,t} + \alpha\cdot\left[w\cdot HIT + (1-w)\cdot ITL\_HOP\right] \quad \forall r,s \in path \text{ of } k, \tag{8}$$

where α is the global evaporation factor of the pheromone, and *w* is a weight factor that controls the relative importance between the resource found (*HIT* function) and the time-to-live (*ITL_HOP* function). The amount of pheromone deposited depends on the quality of the solution obtained, the amount of resources found, and the time-to-live of the ant at time of discovery.

5 Multi-agent Architecture and Parallel Pseudocode

In this section we describe the NAS algorithm for the Semantic Querying Routing Problem. We first present an agent-based architecture and then a parallel pseudocode.

Multi-Agent System Architecture. The overall system architecture is shown in Figure 4. It comprises two elements: *i*) the *environment* (*E*) which is a static P2P complex network, with a probability distribution for the network topology (*T*), understood as a set of linked nodes with local information about their neighboring nodes, *ii*) the *agents* {{e_1}, {e_2}, {e_3}} that can be of three kinds, depending on their role: query, search, and retrieval. In the NAS algorithm the agents are represented as *ants*, each agent is either an ant that carries a query (*Q*) or a node of the network that launches the query; each node also has its own repository (*R*).

The *query agents* {e_1} represent stationary ants located in the nodes that launch queries and their role is to create *search agents*. The *search agents* {e_2} represent ants born in a node that launches a query. Their role is to move through network following a set of rules. These agents operate through the state transition that chooses between exploratory or exploitatory movements through Eqs. (5) and (6). Each time that an action is activated the local update module conducts local evaporation from the pheromones table through Eq. (7). In order to evaluate its performance, an agent records the routes that have been selected, and each time it finds a resource, it creates a *retrieval agent*. The Lookahead strategy verifies if a resource exists in the current node or in its neighborhood. The time-to-live of a search agent is set at *maxTTL*, but the agent could also ceases to operate upon reaching the expected amount of results *maxResult*. All modules rely on control parameters that must be configured properly to ensure good performance of the system. The *retrieval agents* {e_3} are ant created whenever a result is found in a node. These agents are responsible for evaluating the performance of the search agents and updating with feedback the pheromone table. With this feedback, done through Eq. 8, the route that was traversed by the search agent up the resource is updated on the pheromone table and returned to the end user.

NAS Algorithm Parallel Pseudocode, is a metaheuristic algorithm, where a set of independent agents called ants cooperate indirectly and sporadically to achieve a common goal. The algorithm has two objectives: it seeks to maximize the number of resources found by the ants and to minimize the number of steps taken by the ants. NAS guides the queries toward nodes that have better connectivity using the local structural metric DDC [14]. Since the DDC, in order to minimize the hop count, measures the differences between the degree of a node and the degrees of its neighbors, the more frequent that a query is carried towards a resource a better path is selected. This is, the rate of optimization of a query depends directly on its popularity.

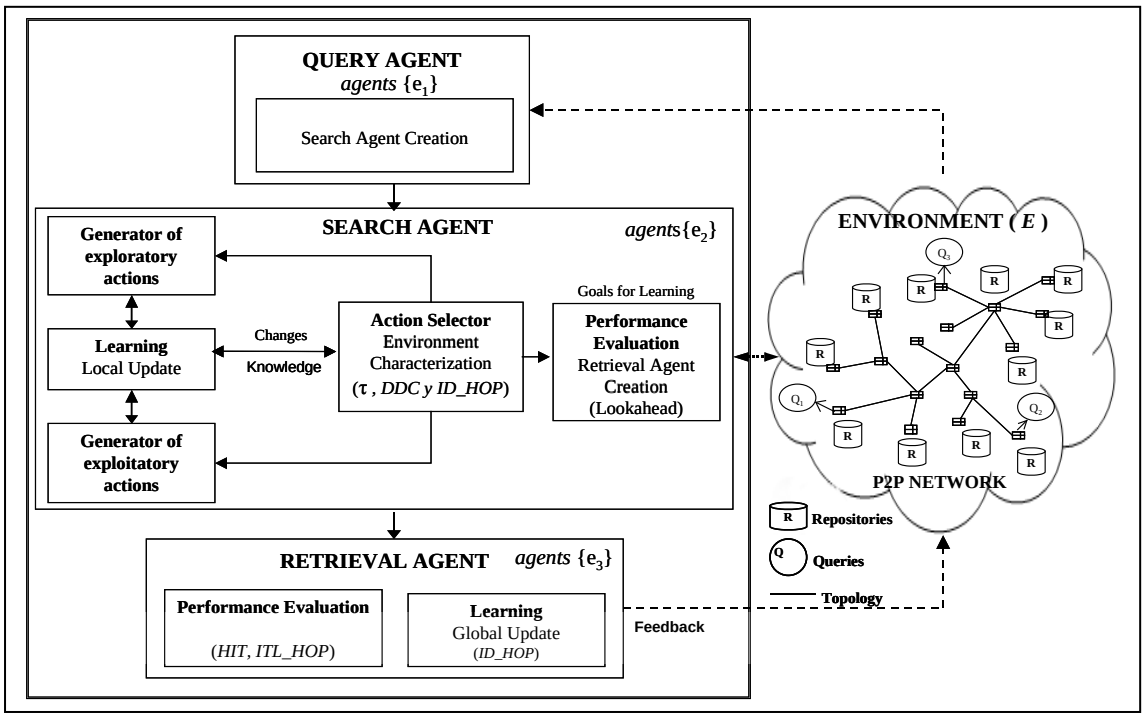


Fig. 4. NAS Architecture

Table 1. NAS algorithm pseudo code

01	**parallel** // Concurrent activity of **query agents**
02	**for each** query in r_k create a search agent k with $TTL_k = maxTTL$ and $Hits_k=0$
03	**while** $Hits_k < maxResults$ and $TTL_k > 0$ // Concurrent activity of **search agents**
04	// Phase 1: The evaluation of results
05	**if** the unvisited $s_k \in \{ r_k \cup \Gamma(r_k)\}$ has the searched resource // Lookahead strategy
06	r_k = append s_k to $path_k$
07	$Hits_k = Hits_k + 1$
09	Local Pheromone Update (Eq. 7)
10	Global Pheromone Update (Eq. 8) // Concurrent activity of **retrieval agents**
11	**else** // Phase 2: The state transition
12	**if** r_k is a leaf node or does not have an unvisited neighbor ,
14	remove the last node from $path_k$
15	**else**
	s_k = apply the *transition* rule with the DDC function (Eqs. 5 and 6)
16	r_k = append s_k to $path_k$
18	Local Pheromone Update (Eq. 7)
	$TTL_k = TTL_k - 1$
19	kill the search agent

The NAS algorithm performs in parallel all the queries using **query agents**. The role of each query agent is to create a **search agent** when a query is launched in a corresponding node. The activity of each search agent consists of two main phases. The first phase, the **evaluation of results** (lines 04-10 of the pseudocode in Table 1) implements the classical Lookahead technique. That is, an ant k, called search agent and located in a node r_k, verifies if the

resource exists in an unvisited node s_k that belongs to its neighborhood, including itself. If the resource is found, the ant adds the node s_k to its path, updates the number of occurrences of the queried resource $Hits_k$, reduces the ant time TTL_k by one hop, performs the local pheromone update according to Eq. (7), and performs the global pheromone update according to Eq. (8). The global updating of the pheromone is a concurrent activity of the ants called **retrieval agents**; the route to the resource is updated on the pheromone table and returned to the end user. In the case that the evaluation phase fails, a second phase, the **state transition** (lines 11-18 of the pseudocode in Table 1) is carried out. This phase selects through random number q, Eq. (5), a neighbor node s. In the case that there is no new node towards which to move, that is to say, the node is a leaf or all neighbor nodes have been visited, a hop backwards is carried out on the path. Otherwise the ant adds the selected node s_k to its path, updates locally the pheromone, Eq. (7), and reduces TTL_k by one hop. The query process ends when the expected number of results has been found or TTL_k reaches zero. In both cases the search agent is killed, indicating the end of the query.

6 Experimental Analysis of the NAS Algorithm

In this section, we describe two experiments carried out on the NAS algorithm. The objective of the first experiment is study the performance of NAS in comparison with algorithms proposed in the literature, SemAnt and Random-Walk. The objective of the second experiment is to examine the contribution of each local environment strategy to the performance of the NAS algorithm.

6.1 Experiments Setup

The NAS algorithm was implemented to solve SQRP instances. The application of the NAS algorithm requires the specification of the problem instance to solve and the definition of the control parameters of the algorithm. In our implementation, an SQRP instance is determinated by three separate files: topology, repositories, and queries. We generated the experimental instances as follows.

The generation of the *topology* (*T*) is based on the method of Barabási et al. [4] to create a non-uniform network with a *scale-free distribution*. In the scale-free or *power law* distribution, a reduced set of nodes has a very high degree and the rest of the nodes have a small degree. All networks generated have 1,024 nodes and bi-directional edges. The number of nodes was selected based on recommendations by Michlmayr [10] and Di Caro [6].

In the P2P model, each peer manages a *local repository* (*R*) of resources and offers its resources to other peers. We generated these repositories using "topics" obtained from ACM Computing Classification System taxonomy (ACMCCS). This database contains a total of 910 distinct topics. Also the content distribution is a power law: few nodes contain many topics in their repositories and the rest of the nodes contain few topics.

For the generation of the *queries* (*Q*), each node was assigned a list of possible topics to search. This list is limited by the total amount of topics of the ACMCCS. During each step of the experiment, each node has a probability of 0.1 to launch a query, selecting the topic uniformly at random within the list of possible topics of the node. The probability distribution of *Q* determines how often the query will be repeated in the network. When the distribution is uniform, each query is duplicated 100 times on average.

The topology and the repositories were created static, whereas the queries were launched randomly during the simulation. Each simulation was run for 20,000 time units (queries). The average performance was studied by computing three performance measures each 100 units of time:
Average hops, defined as the average amount of links travelled by a search agent until its death (that is, reaching either *maxResults* = 10 or *maxTTL* = 25). *Average hit-rate*, defined as the average number of resources found by each search agent until its death and *Average efficiency*, defined as the *average hit-rate* divided by the *average hops*.

The control parameters of the algorithm are specified in a file containing a *global static configuration* of the NAS algorithm parameters. The configuration of the NAS algorithm used in the experimentation is shown in Table 2. In the first column is the parameter value and the in second column is given a description of the parameter. These parameter values were based on recommendations done by Michlmayr [10] and Dorigo [7].

6.2 Comparative study of the NAS algorithm

In this experiment, the performance of the NAS algorithm is compared against the SemAnt [10] and Random-Walk [3] algorithms. For the experimentation with three algorithms we use the general specifications described in Section 6.1.

To carry out the experiment with the SemAnt algorithm with the same conditions, we only changed the parameter *number of links*. For SemAnt [10], the number of connections range between 4,000 and 10,000 which does not affect the performance of the algorithm. Hence, we fixed the number of the links to 7,000 links, choosing an intermediate value in that range.

Table 2. Configuration parameter of the NAS algorithm

Parameter	Definition
ρ= 0.07	Local pheromone evaporation factor
α= 0.07	Global pheromone evaporation factor
τ_0 = 0.009	Pheromone table initialization
ID_HOP_0 = 0.001	Initial value of the table of distances to previously encountered resources
β = 2	Relative importance of DDC and *ID_HOP* with respect to the pheromone
q_0 = 0.9	Relative importance between exploration and exploitation
maxResults =10	Maximum number of results to retrieve
maxTTL = 15	Time-to-live of the search agents
w = 0.5	Relative importance of the resources found and the time-to-live

The experimental values for the SemAnt algorithm were obtained from [10]. For the *average hit-rate* variable the SemAnt algorithm shows values from 0.8 to 2.1 hits per query. However, for the NAS algorithm, the average hit-rate ranges from 9.5 to 10 hits per query. On the other hand, the *average hop* count of the SemAnt algorithm starts out at 23 and lowers to 16 hops per query during the operation of the algorithm, whereas the NAS algorithm, starts at the average hop of 12.5 and then diminishing to 12 hops per query. Finally, the *average efficiency* of the SemAnt algorithm improves from 0.034 to 0.13 hits per hop, while for the NAS algorithm, as shown in Figure 5, it increases from an initial value of 0.76 up to 0.84 hits per hop.

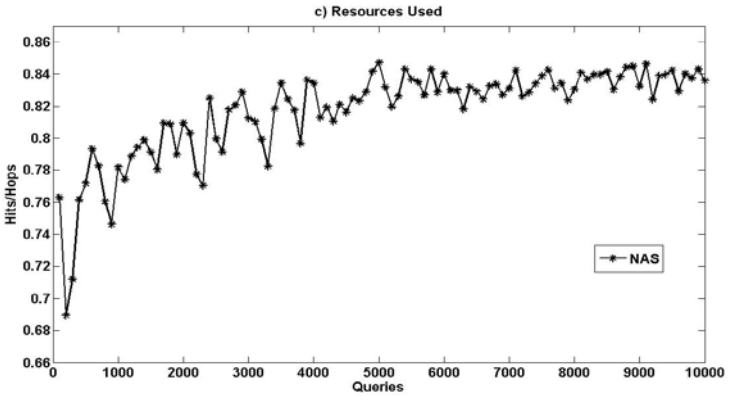


Fig. 5. Performance of NAS Algorithm *average efficiency*

For the experiment with the Random-Walk (RW) algorithm, we also used the general specifications described in Section 6.1. Figure 6(a) shows the *average hit-rate* for both RW and NAS algorithms. The behavior changes slightly over time. That is, the average hit-rate in RW varies between 1 to 0.5 hits per query, while in NAS, the average hit-rate varies between 9.6 to 10 hits per query. Similarly Figure 6(b) shows the *average hop* count for both RW and

NAS algorithms. The behavior of the RW does not evolve; the average hop count keeps around 15 hops per query, from the beginning to the end. However in NAS the behavior evolves, starting at 12.5 and lowering down to 12 hops on average. Finally, Figure 6(c) shows the *average efficiency* for both RW and NAS algorithms. In the RW algorithm, the behavior does not evolve; the average efficiency keeps around 0.5 hits per hop. For NAS the behavior does evolve; so that the average efficiency increases from 0.76 to 0.84 hits per hop.

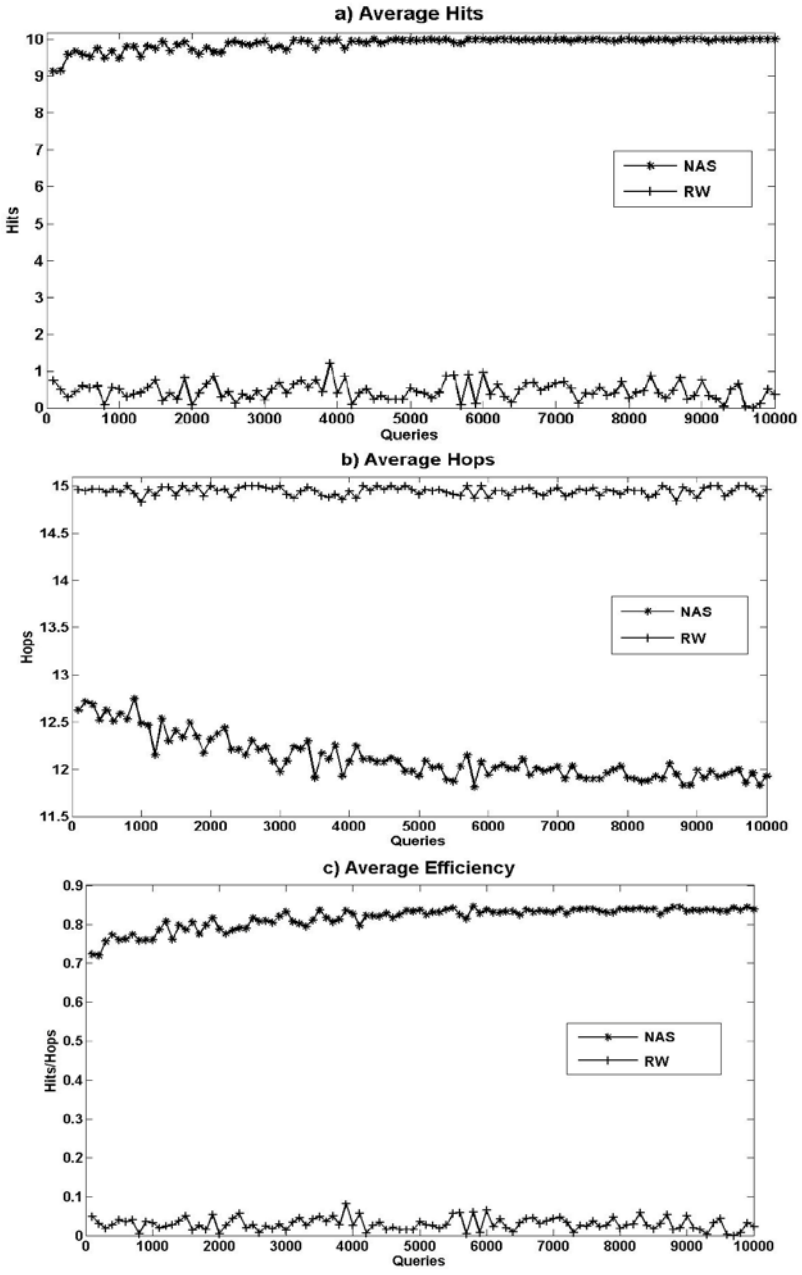


Fig. 6. Comparison of NAS Algorithm against Random-Walk Algorithm. a) *average hits-rate*, b) *Average hops* and c) *average efficiency*

6.3 Experiments for the Analysis of the Local Environment Strategies of NAS Algorithm

In this experiment, the performance of the NAS algorithm is analyzed experimentally in order to determine the contribution of each of the three used local strategies – modified LR, DDC, and Lookahead – to increase the performance of the NAS algorithm.

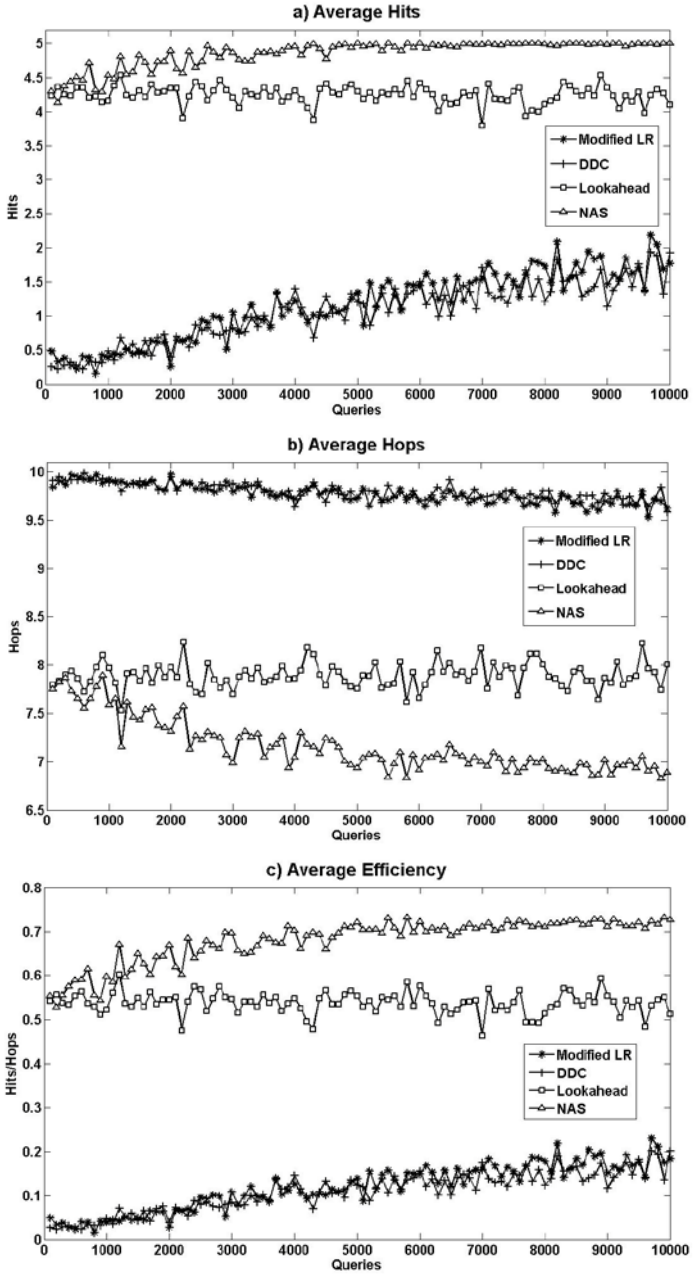


Fig. 7. Comparisons of NAS Local Environment Strategies. a) Average HITS, b) Average HOPS and c) Average Efficiency

For evaluating the contribution of modified LR, DDC and Lookahead strategies were eliminated. The contribution of DDC was evaluated eliminating only Lookahead; modified LR was kept because DDC is included in it. Similarly, Lookahead was evaluated without modified LR and DDC. For the experiment with the NAS algorithm, we used the general specification described in Section 6.1, with the exception of three parameters: *maxResult* =5, *maxTTL*=10, and $q_0 = 0.90$.

Figure 7(a) shows the *average hit-rate* performed during a set of queries with NAS and three different configurations of NAS: modified LR, DDC and Lookahead. For the modified LR and DDC, the algorithms start approximately at 0.5 hits per query; at the end, the average hit-rate increases to 2 hits per query. For Lookahead and NAS the average hit-rate starts at 4.3 and after 1,000 queries the behavior changes; for Lookahead the average hit-rate ends at 4 and for NAS ends at 5. On the other hand, Figure 7(b) shows the *average hops* performed during in a set of queries with NAS and three different configurations. For modified LR and DDC, the behavior is approximately the same; the average hop count starts at 10 and ends at 9.5 hops per query. However, the Lookahead and NAS start at 7.8 and after 1500 queries, the behavior changes; for Lookahead the average hop count ends at 8 and for NAS ends at 6.9 hops per query. Finally, Figure 7(c) shows the *average efficiency*. For the modified LR and DDC, the behavior is approximately the same; at the beginning the efficiency is around 0.05, at the end the efficiency increases to 0.2 hits per hop. However, for the Lookahead and NAS, the behavior evolves such that the average efficiency stars at 0.55 and after 1,500 queries, the behavior changes; for Lookahead, the average efficiency ends at 0.5, for NAS, the average efficiency ends at 0.7. Hence, the best performance was obtained with the combination of these three strategies.

Besides, the results reveal that for the specified configuration, the Lookahead method shown the biggest contribution to the final performance of NAS, giving an efficiency of 0.5 hits per hop. While the DDC and modified LR had a similar impact of 0.2 hits per hop. In experimentations with other configurations [13], the analyzed strategies have shown different contributions. Due to this result, it becomes relevant to study further the relations that exist between the problem characteristic and the algorithm parameter configuration in order to yield a bigger benefit of each one local strategy proposed for NAS.

7 Conclusions and future work

For the solution of the Semantic Query Routing Problem (SQRP), we proposed a novel algorithm called NAS that is based on existing ant colony algorithms but incorporating local environment strategies: modified LR, DDC, and Lookahead. Three functions are used to learn from past performance: importance of hits (HIT), importance of time-to-live (ITL_HOP), and importance of distance (ID_HOP). This combination results in a lower hop count and an upper hit count, outperforming two algorithms proposed in related work, Random-Walk and SemAnt.

Our analysis and simulations confirm that the proposed techniques are more effective at improving search efficiency. Specifically the NAS algorithm in the efficiency shows six times better performance efficiency, than the SemAnt, and seventeen times better performance than the Random-Walk. We observe that upon including learning and characterization with modified LR and DDC respectively, the algorithm evolves to reach an average of 2 hits with 9.5 hops per query. Adding only exploration with Lookahead, the algorithm keeps a constant performance of 4 hits with 8 hops. Combining all the three strategies the NAS algorithm evolves to yield 5 hits per 6.9 hops. As observed, the best results were obtained in the combination of proposed strategies.

We plan to study more profoundly the relation among SQRP characteristics, the configuration of NAS algorithm and the local environment strategies employed in the learning curve of ant-colony algorithms, as well as their effect on the performance of hop and hit count measures.

References

1. **Amaral, L. A. N., & Ottino, J. M. (2004).** Complex Systems and Networks: Challenges and Opportunities for Chemical and Biological Engineers, *Chemical Engineering Scientist*, 59(1), 1653–1666.

2. **Androutsellis-Theotokis Stephanos & Diomidis Spinellis (2004).** A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4), 335–371.
3. **Arora, S., & Barak, B. (2009).** *Complexity Theory: A Modern Approach*. New York: Cambridge University Press.
4. **Barabasi, A., Albert & R., Jeong, H. (1999).** Mean-Field theory for Scale-free Random Networks. *Physical A*, 272(1), 173–189.
5. **Cruz-Reyes, L., Gómez S. C., Aguirre L. M., Schaeffer E., Turrubiates L.T., Ortega I. R., & Fraire H. H. (2008).** NAS Algorithm for Semantic Query Routing System for Complex Network. *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008), Advances in Soft Computing*, 50, 284-292.
6. **Di Caro, G. & Dorigo, M. (1998).** AntNet: Distributed Stigmergy Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9(1), 317-365.
7. **Dorigo, M. & Gambardella, L. M. (1997).** Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, 1(1), 53-66.
8. **Erdős, P. & Rényi, A. (1960).** On the Evolution of Random Graphs. *Publications of the Mathematical Institute of the Hungarian Academic of Sciences.* 5(1), 17-61.
9. **Liu, J., XiaoLong, J. & Kwok, C.T. (2005).** *Autonomy Oriented Computing /From Problem Solving to Complex System Modeling*. New York: Kluwer Academic Publisher.
10. **Michlmayr, E. (2007).** *Ant Algorithms for Self-Organization in Social Networks*. PhD Thesis, Vienna University of Technology. Austria, Vienna.
11. **Mihail, M., Saberi A. & Tetali P. (2006).** Random Walks with Lookahead in Power Law Random graphs. *Internet Mathematics*, 3(2), 147-152.
12. **Ortega R., Meza E., Gómez C., Cruz L., & Turrubiates T. (2007).** Impact of Dynamic Growing on the Internet Degree Distribution. *Polish Journal of Environmental Studies*, 16(1), 117-120.
13. **Sakaryan G. (2004).** *A Content-Oriented Approach to Topology Evolution and Search in Peer-to-Peer Systems*, PhD Thesis, University of Rostock. Rostock, Germany.
14. **Yang, K. & Wu, C., Ho, J. (2006).** AntSearch: An Ant Search Algorithm in Unstructured Peer-to-Peer Networks. *IEICE Transactions on Communications*, 89(9), 2300-2308.

Claudia Gómez S*. was born in Mexico in 1971. She is a doctoral student at National Polytechnic Institute, Mexico. She received her MS degree in Computer Science from the Leon Institute of Technology, Mexico, in 2000. Her research interests are optimization Techniques, complex network and autonomous agents.*

Laura Cruz-Reyes *was born in Mexico in 1959. She received the PhD (Computer Science) degree from National Center of Research and Technological Development, Mexico, in 2004. She is a professor at Madero City Institute of Technology, Mexico. Her research interests include optimization techniques, complex networks, autonomous agents and algorithm performance explanation.*

Eustorgio Meza *received the PhD (Oceanic Engineering) deegree from Texas A&M University, College Station, U.S.A. He received the MS degree in Computer Science (AI) from Institute of Technology and Advanced Studies of Monterrey, Mexico. He is a Professor at Research Center in Applied Science and Advanced Technology from National Polytechnic Institute. His research interests are oceanology, complex Network.*

Elisa Schaeffer *was born in Finland in 1976. She received the PhD (Science in Technology) degree from Helsinki University of Technology, Finland, in 2006. She is a teaching researcher at the Graduate Program in Systems Engineering (PISIS), at the Faculty of Mechanical and Electrical Engineering (FIME), Universidad Autonoma de Nuevo Leon, Mexico. Her research interests include nonuniform networks, graph clustering and optimization of network operation*

Guadalupe Castilla*. is a doctoral student at Madero Institute of Technology, Mexico. She received her MS degree in Computer Science from the Leon Institute of Technology, Mexico, in 2002. Her research interests are optimization Techniques.*