Self-Organization for Search in Peer-to-Peer Networks: The Exploitation-Exploration Dilemma

Elke Michlmayr
Women's Postgraduate College for Internet Technologies (WIT)
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstrasse 9-11/E188, 1040 Vienna, Austria
Telephone: 0043 1 58801 18816
Email: michlmayr@wit.tuwien.ac.at

***Abstract*— This paper presents the design and evaluation of an ant-based approach to query routing in peer-to-peer networks. After pointing out how to employ the ant metaphor in peer-to-peer networks, we conduct a thorough evaluation of the impact of different settings for the configurable parameters present in ant algorithms on the performance values. In particular, the focus is on the effects of setting the ratio between (1) ants exploiting the option currently known as the best one and (2) ants exploring the search space with the aim of finding improved options. We show that the exploitation-exploration dilemma can be avoided by an adequate design of the exploring option.**

***Index Terms*— Self-Organization, Peer-to-Peer Networks, Ant Colony Optimization, Exploration, Exploitation, Distributed Artificial Intelligence, Multi-Agent Systems**

I. Introduction

The principles of self-organization and emergence have received a lot of interest in the research community recently. In particular, the trail-laying and trail-following behavior observed from foraging ants has been employed for solving diverse problems in computer science. Although the ant metaphor has been successfully applied to routing of data packets both in wireless networks [1] and fixed networks [2], little is yet known about its adequacy for the task of query routing in peer-to-peer networks. The challenges for the latter are the following: Each peer is connected via outgoing links to some other peers which are called its neighbor peers. If a peer issues a query or receives a forwarded query from one of its neighbor peers, it has to decide based on its *local knowledge* which neighbor peer to send the query to. Since ant-based methods rely on local knowledge and indirect communication only, they are suitable for this task.

In *reputation learning* approaches [3] to query routing, the local information of a peer is gained by (1) continuously observing the queries and answers that pass the local node and by (2) recording which kind of queries its neighbor peers are able to answer. The recorded data must be accumulated and stored in an appropriate way to support the neighbor selection process. Based on the recorded data, the peer chooses the neighbor peer which is most likely to store results itself, or has neighbor peers that are likely to store such resources.

One of the advantages of using the ant metaphor for reputation learning is that it is readily applicable. Pheromone trails store the accumulated data about the successful queries in the past, and queries are represented as ants. This paper is part of our ongoing efforts [4], [5] to design and evaluate SemAnt, a reputation learning-based algorithm for query routing in peer-to-peer networks compliant with the *Ant Colony Optimization* meta-heuristic [6]. One of the challenges of our work is that ant algorithms include various configurable parameters for which appropriate value settings must be found. Ideally, it would be possible to automatically determine these parameter values at runtime. In particular, there is a parameter that influences a basic decision each ant has to make before selecting an outgoing link. It can either

- *exploit* the best results known so far for path selection, or it can
- *explore* a path that is not currently known as the best one in order to possibly find an improved solution to the problem. If it succeeds, this will enhance the performance of the system.

The question of which one of the strategies to select according to its desirability in the current context of the ant is referred to as the *exploitation-exploration dilemma* [7], [8]. The dilemma occurs not only in ant algorithms, but in reinforcement learning [9] in general. In ant algorithms, the decision about which strategy to use is performed based on a parameter that specifies a pre-defined ratio between exploring and exploiting.

Contribution. The contribution of this paper is an evaluation of the impact of the ratio between exploring and exploiting strategy in the context of query routing in peer-to-peer networks. What we will show in the following is that our algorithm is self-configuring in the sense that the overall efficiency of the search process is the same no matter which ratio is employed, and also the performance of the system as perceived by individual users is not affected by the ratio.

Organization of the paper. This paper is organized as follows. Sec. II defines the problem and states the assumptions which were made. Sec. III contains a detailed description of all aspects related to the SemAnt algorithm. Sec. IV, next to showing the performance of the algorithm in comparison

This research has been funded by the Austrian Federal Ministry for Education, Science, and Culture (bm:bwk), and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

to other approaches, presents experimental results that prove the claims we made above. Sec. V provides a discussion of related work on ant algorithms in peer-to-peer networks.

II. Problem description

A peer-to-peer network is a network consisting of interconnected nodes in which each node manages an information repository containing a certain number of resources. Every peer (1) offers its resources to the other nodes in the network and (2) issues queries to the network. All nodes collaborate to answer the queries and therefore – as a whole – implement a distributed search engine. For each query, the shortest path through the network must be found that leads from the querying peer to one or more answering peers offering one or more resources that are appropriate for satisfying the query.

The resources present at each peer can be any kind of files that are annotated with metadata. The metadata are composed of name-value pairs, also called *elements*. These elements are used for specifying additional information about a certain resource. In general, a meta-data schema defines the name and the meaning for each of the elements the schema is comprised of. In the following a simplified view on the problem is used, where only one element is considered. In addition, there is a restriction on the allowed values. The values that can be used for annotation originate from a controlled vocabulary, e.g, the concepts of a taxonomy or an ontology.

The vocabulary for queries is the same as the metadata vocabulary used for annotating resources. This means that the queries do not consider the actual content of a resource, but rather the metadata that describes this content. A query Q consists of a concept c which will be referred to as the keyword of the query hereafter. The resource R will be referred to as the result of the query.

In order to make it possible to concentrate on the problem of query routing, the following assumptions are made about the application scenario:

1) Each peer has an unique address that can be used as an unique identifier.
2) The resources at each peer can be uniquely identified using an existing resource identifier (such as a filename) together with the peer identifier.
3) All links between peers are bi-directional, that is, can be used in both directions.
4) The network topology already exists. Each peer already knows which neighbor peers it is connected to. The problem of peer discovery is not within the scope of this paper.
5) The network topology and the content distribution in the network is considered static. The extension of the algorithm for a dynamic setting is subject of future work.

III. Specification of the algorithm

This section specifies the components of the SemAnt algorithm. Sec. III-A documents the data structures that need to be stored at each peer. Sec. III-B describes the query routing procedure and all aspects related to facilitating ant algorithms in a peer-to-peer network. In Sec. III-C, the mechanisms to select outgoing links in the query routing procedure are covered. Sec. III-D explicates how routing tables are updated after successful queries. Sec. III-E describes the activities executed locally at the peers.

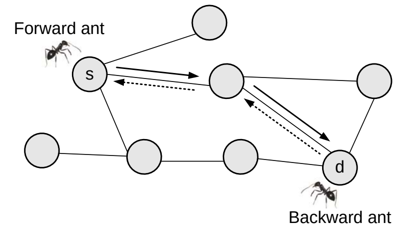

Fig. 1. The concept of forward and backward ants. Node s is the querying peer and node d is the answering peer. The solid lines show the path of the forward ant, the dotted lines that of the backward ant.

A. Data structures

The routing information is stored in a table τ which is present at each peer P_i. Table τ maintains the pheromone trails. It is of size $C \times n$, where C is the size of the controlled vocabulary that defines the allowed keywords in a query and n is the number of peer P_i's outgoing links to neighbor peers. Each τ_{cu} stores the amount of pheromone corresponding to concept c dropped at the link from peer P_i to peer P_u, for each concept c and each neighbor peer P_u. All entries in table τ are initialized with the same value $\tau_{init} = 0.009$. This is necessary to prevent divisions by zero in the evaporation feature (see Sec. III-E) and in link selection (see Sec. III-C).

B. Query routing

The concept of forward ants and backward ants (see Fig. 1) from the *AntNet* algorithm [2] is employed as the foundation of the query routing procedure. In addition, *AntNet*'s mechanism for preventing cycles is adopted. Whereas the latter can be adopted without any changes, the concept of forward ants and backward ants needs to be adapted to the application purpose of query routing. In the following, the adaptations necessary are discussed.

1) Queries: In the SemAnt algorithm, queries are represented as ants. This approach has two advantages. First, no additional traffic is created in the network. Second, representing queries as ants guarantees that the degree of optimization for certain query keywords directly depends on the popularity of a given keyword. The more often a query keyword is requested, the better its paths will be optimized in terms of indicating the way through the network to the most appropriate peers. Instead of sending out forward ants at regular intervals from random peers like in *AntNet*, a forward ant is created for each query that occurs in the network. This ant is created at the peer which issued the query, and it is responsible for answering it.

2) Link costs: Ant algorithms usually consider two different types of information in the link selection process. First, there is the pheromone distribution of the outgoing links. This information is built incrementally by the ants. Second, there is problem-specific information that defines the cost of every link. In the case of query routing, this would be the time it takes to travel from one peer to another. Since this feature is already integrated in ant algorithms, including support for different link costs in the SEMANT algorithm would be easily possible. It would also be useful because in a real-world setting, each link between two peers has a certain latency, a certain bandwidth, and a certain throughput depending on the hardware of the connection.

However, there are two reasons that make the evaluation of this feature very cumbersome. First, there is no appropriate test data composed of a real-world network topology together with defined latency/bandwidth/throughput properties available. Second, even if such data would be available, there is no other approach to search in peer-to-peer network that considers different link costs. This makes it impossible to conduct performance evaluation by comparison to reference values. For these reasons, the algorithm only considers the pheromone distribution of the outgoing links but no link costs.

3) Time-to-live parameter: Next, it is necessary to define at which point the forward ant should stop its travel. In *AntNet*, forward ants terminate their travel through the network when they arrive at their well-defined destination peer. This behavior can not be transferred, since the forward ant's task is to find an unknown destination peer that in the worst case does not even exist. Instead, a stop point for forward ants must be defined to prevent forward ants from running infinitely if no results can be found. The simplest solution is to use a time-to-live (TTL) parameter ttl_{max} like introduced in Gnutella (see [11]). Each time an ant travels one hop to reach another peer, it decrements ttl_{max} by one. The stop point is reached if $ttl_{max} = 0$.

4) MinResources variation and maxResults variation: After a forward ant has found a result and creates a backward ant, there are two possibilities for proceeding further. Either the forward ant (1) terminates its travel, or (2) it continues it until the maximum time-to-live parameter is reached. The idea behind the latter approach is that if forward ants are allowed to go on after they found the first peer that stores results, they can increase the absolute number of results found by detecting other appropriate peers. Keeping in mind that query routing in peer-to-peer networks is a special kind of optimization problem, the choice between these two options determines the optimization goal of the algorithm:

- ***MinResource* variation.** If the ants are terminated after they found the first result, the optimization goal is to use the minimum amount of network resources. Therefore, the forward ant strategy of stopping after the first result is found will be referenced to as the *minResource* variation of the SEMANT algorithm.
- ***MaxResults* variation.** In the other case, where the ants use the maximum time-to-live parameter, the ants use approximately the same amount of network resources for each query and the optimization goal is to maximize the number of results that are found for a query. The strategy of using the maximum time-to-live parameter will be referenced to as the *maxResults* variation of the SEMANT algorithm.

In practice, both of these variations are valid because both of the optimization goals are desired at the same time. Using the *minResource* variation is of benefit for the performance of the entire network, since the algorithm saves as much network resources as possible. Using the *maxResults* variation is of benefit for the individual users of the network, since the algorithm tries to find as many results for a single query as possible. It is possible to combine these variations by using a weight that defines the ratio between employing the *minResource* variation and employing the *maxResults* variation.

5) Step-by-step description of the query routing procedure: Now the complete procedure for answering a query is laid out. Consider a query q issued at a peer P_q. For simplicity, the assumption is that query q is a simple query containing exactly one keyword c. The extensions to the algorithm for supporting complex queries are described in [12]. The following seven steps are necessary for answering query q.

1) Check the resource repository of peer P_q. If any results are found, present them to the user. If the number of results found is less than r_{max}, go to step 2. If the number of results found is greater than r_{max}, terminate the algorithm.
2) Create a forward ant F_q with timeout ttl_{max} at peer P_q. Add the identifier of peer P_q to F_q's stack of already visited peers $s(F_q)$.
3) Use the link selection procedure described in Sec. III-C to select the neighbor peer(s) P_{j_x} $(x \in [1..n], n \in \mathbb{N})$ the forward ant F_q should choose. For every peer P_{j_x}, create a copy of forward ant F_q and send it to peer P_{j_x}.
4) For every ant F_q that arrives at a peer P_j, check if peer P_j was already visited by a copy of F_q. If so, terminate ant F_q. Otherwise, check the resource repository of peer P_j for resources r that are results for query Q. If there are no results, continue at step 6. Otherwise, add the identifiers of all resources r to the set R and continue at step 5.
5) Generate a backward ant B_q. Pass it R, the identifier of the peer P_j that stores R, and the stack of already visited peers $s(F_q)$. Send B_q back to the querying peer P_q using the procedure described in Sec. III-D. In case the *minResources* variation is used, terminate the forward ant F_q. Otherwise, continue at step 6.
6) Add the identifier of peer P_j to the stack of already visited peers $s(F_q)$.
7) If $ttl_{max} > 0$, let F_q continue at step 3. Otherwise, terminate F_q.

As soon as a backward ant B_q arrives at the querying peer P_q, the results $r \in R$ are presented to the user. In case the user decides to download a resource r, a direct connection between peer P_q and the peer P_j that stores r is established and resource r is retrieved from peer P_j.

C. Link selection

Now the selection of outgoing links by the forward ants is described. In ant algorithms, this selection is made by applying a so-called transition rule. The transition rule designed for the SEMANT algorithm is based on the transition rule from the *Ant Colony System* algorithm [13], which consists of two strategies that supplement each other. In the exploiting strategy, the ant determines the quality of the links depending on the amounts of pheromone and always selects the link with the highest quality. The exploring strategy encourages ants to discover new paths. This is achieved by deriving goodness values for the neighbor peers according to the amounts of pheromone on the links that lead to them, and by probabilistically selecting a subset of the peers in proportion to their goodness values.

As already discussed in Sec. I, the decision for one of the strategies is based on a parameter $w_e \in [0,1]$. For example, if parameter w_e is set to 0.85, the forward ants will employ the exploiting strategy in 85% of the cases. Each time a forward ant has to select an outgoing link, it individually decides for a strategy by applying the *roulette wheel selection technique* [14] together with parameter w_e as an input value. The strategies are defined as follows:

1) Exploiting strategy: In case the exploiting strategy is used, a forward ant F_q located at a certain peer P_i selects the neighbor peer P_j with the highest amount of pheromone for the keyword of the query (see Eq. 1).

$$j = arg\ max_{u \in U \wedge u \notin s(F_q)} \tau_{cu} \tag{1}$$

In Eq. 1, U is the set of neighbor peers of peer P_i, and $s(F_q)$ is the set of peers already visited by F_q.

2) Exploring strategy: In case a forward ant F_q utilizes the exploring strategy, the transition rule shown in Eq. 2a and Eq. 2b is applied *for each* neighbor peer P_j in order to decide whether peer P_j should be selected. Note that this is an adaptation of the roulette wheel selection technique: Each p_j is *separately* placed on the continuum between 0 and 1, and for each p_j a random value q is calculated for deciding whether peer P_j should be selected. This mechanism allows more than one peer to be selected in order to account for the fact that there are multiple possible destination peers which contain answers for a query. To ensure that at least one peer will be selected, the algorithm falls back to the exploiting strategy in case applying the exploring strategy does result in not selecting any peer.

$$p_j = \frac{\tau_{cj}}{\sum_{u \in U \wedge u \notin s(F_q)} \tau_{cu}} \tag{2a}$$

In Eq. 2a, the assumption is that a forward ant F_q is located at a certain peer P_i, U is the set of neighbor peers of peer P_i, and $s(F_q)$ is the set of peers already visited by F_q.

$$GO_j = \begin{cases} 1 \text{ if } q \leq p_j \wedge j \in U \wedge j \notin s(F_q) \\ 0 \text{ else} \end{cases} \tag{2b}$$

In Eq. 2b, q is a random value, $q \in [0,1]$, and the sum of all goodness values $\sum_{j \in U \wedge j \notin s(F_q)} p_j = 1$. If $GO_j = 1$, the forward ant F_q creates a copy of itself and sends it to peer P_j.

D. Routing table updates

A description of the mechanism used by the backward ants for updating the routing tables follows. A backward ant is created at a certain peer P_r storing a set of results R. Each backward ant B_q is in possession of a copy of the stack data containing a list of all visited peers $s(F_q)$ recorded by its corresponding forward ant F_q. Based on this information, the backward ant B_q calculates the number of hops h_{qr} between the querying peer P_q and the answering peer P_r. After that, it travels back hop-by-hop to the querying peer P_q according to the information stored in $s(F_q)$. At each intermediate peer, ant B_q is responsible for dropping pheromone by applying the pheromone trail update rule shown in Eq. 3a and Eq. 3b. The amount of newly added pheromone depends on the goodness of the found path, which is determined by comparing the number of resources found and the length of the path to predefined reference values. For the reference solution, the value for a path's total length is set to $\frac{1}{2} \cdot ttl_{max}$, and the number of resources is set to r_{max}.

$$\tau_{cj} \leftarrow \tau_{cj} + Z, \tag{3a}$$

where

$$Z = w_d \cdot \frac{|R|}{r_{max}} + (1 - w_d) \cdot \frac{ttl_{max}}{2 \cdot h_{qr}} \tag{3b}$$

In Eq. 3b, parameter w_d weights the influence of resource quantities and path length.

E. Peer activity

Each peer performs management procedures on its local routing table. It applies the evaporation rule shown in Eq. 4 in predefined intervals t_e for each link to neighbor peer P_u and each concept c. The amount of pheromone that evaporates in every interval is controlled by parameter $\rho \in [0,1]$.

$$\tau_{cu} \leftarrow (1 - \rho) \cdot \tau_{cu} \tag{4}$$

This rule is adopted from the evaporation part of *Ant Colony System*'s global pheromone update rule [13].

IV. Results

This section describes the experimental evaluation of the the SEMANT algorithm. After presenting the setup used in Sec. IV-A, the metrics and the rationale behind choosing them are discussed in Sec. IV-B. Sec. IV-C shows the performance in comparison, and Sec. IV-D discusses the exploitation-exploration dilemma in context.

A. Setup

For evaluating the design of the algorithm, a peer-to-peer system needs to be simulated. The SEMANT algorithm is independent from a certain application scenario. It can be used in every situation in which groups of individuals want to share annotated resources. The application scenario chosen for the evaluation is to support the cooperation between researchers in computer science by allowing them to share scientific articles.

The three most important characteristics of a peer-to-peer system that have an impact on the performance of the search algorithm used are the (1) network topology used, the (2) distribution of the content within the network, and (3) the distribution of queries within the network. The specifications for all three of them will be described below.

Network topology. Since the use case concerns a community of researchers and therefore a social network, a *small* *world* [15] network topology is selected for the experiments in order to choose a realistic model for social networks. The size of the network is set to 1024 peers, and the clustering coefficient of the network is set to 1.

Content distribution. The content is modeled by utilizing the ACM Computing Classification System [16] as the underlying meta-data vocabulary. The ACM Computing Classification System is a taxonomy consisting of 1473 concepts for the domain of scientific literature in the field of computer science. Each resource is annotated with one keyword, i.e., each resource is an instance of one leaf concept from the taxonomy. In total, the ACM Computing Classification System taxonomy contains 910 leaf concepts. In order to represent each research topic equally, the same number of resources is created for each leaf concept. Each peer stores on average 30 resources. In real-world settings, the available resources in a peer-to-peer network are not randomly distributed among the peers. Instead, patterns in the data can be observed since the interests of the peers are not uniform, but each peers concentrates on one or a few topics it has special interests in. In the application scenario of researchers, the special interests are certain research areas. Modeling research areas with the leaf concepts from the taxonomy would lead to very narrow interests. Therefore, the sub-concepts of a third-level concept of the taxonomy are facilitated for modeling a research area. There are 177 third-level concepts in the taxonomy. The assumption is that each peer is an expert on a certain research area and for this reason, on average 60% of the resources in his or her repository are instances of one particular research area. On average, another 20% of the resources are related to another research area. The remaining 20% of the resources are instances of random leaf concepts. Note that the hierarchical relationships between the concepts in the taxonomy are used only for modeling the research areas, but not for query routing. This topic has been addressed in [17], together with an evaluation of the impact of different content distributions on the performance of the algorithm.

Query distribution. In a real-world scenario, the query distribution would be power-law. For simplicity, in this experiment an uniform query distribution is assumed. A ticker clock at each peer is used, and the probability that a peer issues a query within one time unit is set to 0.1. Each query consists of a randomly selected leaf concept from the ACM Computing Classification System taxonomy.

B. Metrics

Several scientific articles about peer-to-peer systems use the metrics of precision and recall [18] known from information retrieval for their evaluation. The drawback of relying on these metrics is that they do not include the traffic created in the network. Our aim is to create an algorithm for query routing which has the property of a low ratio between network traffic and quantity of results. Therefore, we do not rely on precision and recall, but employ the following metrics instead:

- *Resource usage* is defined as the number of links traveled for each query within a given period of time.
- *Hit rate* is defined as the number of resources found for each query within a given period of time.
- *Efficiency* is the ratio of resource usage to hit rate. If we divide the number of links traveled by the number of resources found, we get the average number of links traveled to find one resource, which is the most practical metric.

Obviously, these metrics have the drawback that the recall – which measures the ratio between resources found and resources present in the network – is not known. The value for precision, which measures the ratio between correctly found resources and false positives, is always 100% since the SEMANT algorithm does not produce false positives.

ρ	evaporation factor	0.07
ttl_{max}	timeout of forward ants	15
w_e	weight of exploiting vs. exploring strategy	0.85
r_{max}	maximum number of resources	10
w_d	weight of resource quantity vs. link costs	0.5

TABLE I

PARAMETER VALUES CHOSEN FOR THE SEMANT ALGORITHM

C. Performance evaluation

For performance evaluation, the SEMANT algorithm is compared against the well-known *k-random walker* approach [19]. A random walker is similar to a forward ant, except from the fact that it does not rely on routing tables, but instead makes a random decision about which outgoing link to choose in the link selection procedure. If resources are found, the walker sends an information message back to the querying peer, similar to a backward ant, and walks on. There are two configurable parameters: k is the number of walkers per query, and a time-to-live parameter TTL defines the timeout for the walkers. The time horizon for the experiment is set to 5000 time units. The parameter values shown in Table I are used for the configurable parameters of the SEMANT algorithm. Both the *minResource* variation and the *maxResults* variation are evaluated. In order to provide for comparison fairness,

- both algorithms use the same setup as described in Sec. IV-A,
- the time-to-live parameters are set to an equal value, and
- the parameter settings for the algorithms are set in such a way that – in total – the agents of both algorithms are allowed to travel a comparable number of links.

Consequently, the parameters for the *k-random walker* algorithm are set to $k = 2$ and $TTL = 15$. The results of the comparison are shown in Fig. 2 to Fig. 4. Note that Fig. 2 includes the resource usage of both forward and backward ants/agents for both algorithms. An evaluation of the impact of varying the values used for the control parameters on the performance of the algorithm can be found in [20].

It can be seen that both variations of the SEMANT algorithm outperform the *k-random walker* approach. On average, the latter needs 29.7 hops for retrieving 0.79 results per query. This means that approximately 37.3 hops are necessary for retrieving one appropriate resource. Since the *k-random walker* approach does not include any kind of optimization, the performance values do not improve but stay nearly constant

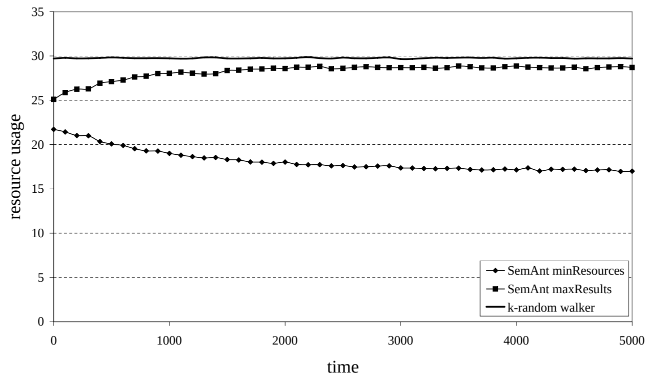

Fig. 2. Resource usage comparison between SEMANT using the *minResources* variation, SEMANT using the *maxResults* variation, and *k-random walker*

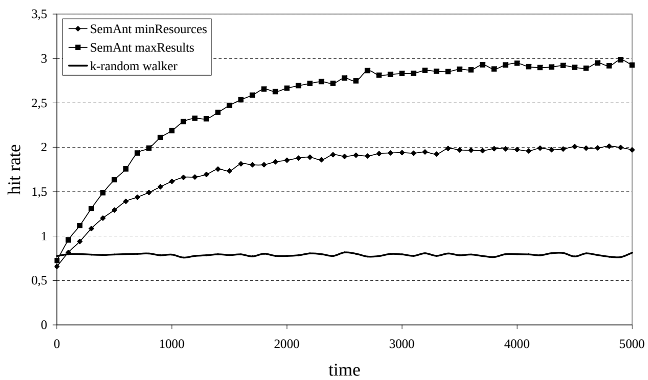

Fig. 3. Hit rate comparison between SEMANT using the *minResources* variation, SEMANT using the *maxResults* variation, and *k-random walker*

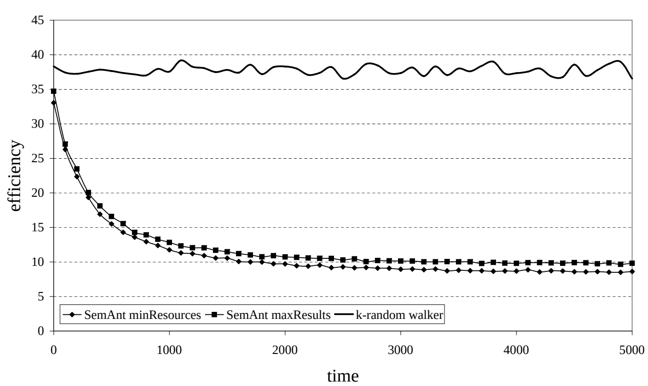

Fig. 4. Efficiency comparison between SEMANT using the *minResources* variation, SEMANT using the *maxResults* variation, and *k-random walker*

over time. On the contrary, the SEMANT algorithm – which in the start-up phase, where all the pheromone trails store the same amount of pheromone, shows only slightly better results as the *k-random walker* reference – optimizes its performance. Especially between time unit 0 and time unit 1000, the performance values improve significantly. About 1500 time units are necessary to reach a converged phase in which the results are nearly stable and show only slight improvements.

The figures clearly indicate that – depending on which variation of the SEMANT algorithm used – two different optimization problems are solved. In case of the *maxResults* variation, the hit rate is maximized. After 5000 time units, on average 2.9 resources per query are found (Fig. 3). However, the resource usage decreases only slightly in comparison to the *k-random walker* reference values. After 5000 time units, on average 28.7 hops are necessary for query routing (Fig. 2). On the other hand, employing the *minResource* variation minimizes the resource usage of the ants. In this case, after 5000 time units 16.9 hops are needed on average (Fig. 2) for finding on average 1.9 resources per query (Fig. 3). Note that in Fig. 2, the dotted curve for resource usage between time unit 0 and time unit 2500 in the *minResource* variation does not seem to decline as rapidly, but considering that (1) the hit rate increases at the same time and that (2) hit rate has an effect on resource usage, the decline of the curve is significant. The effects of minimizing resource usage and maximizing hit rate depending on the variation are even more present if a higher value for parameter ttl_{max} is used (see [12] for figures). However, as also shown in [12], employing a higher value for ttl_{max} has a negative effect on performance.

Since hit rate and resource usage depend on each other, the best way of comparing the variations against each other is to use the metric of efficiency (Fig. 4). It can be seen that in the *minResource* variation less hops are necessary to retrieve one resource in comparison to the *maxResults* variation. These results show that for the overall performance of the system, the *minResource* variation has a higher benefit in terms of a lower number of hops traveled for retrieving one resource. The average difference over the complete time span is is approximately 1.13 hops per query (12.51 hops on average/11.38 hops on average). The cause for this difference can be explained as follows. In the *maxResults* variation, when the forward ants continue searching after they found the first appropriate peer, they tend to stay in the neighborhood of the peer they already found and try to go back to it because the pheromone trails indicate that there is an appropriate peer nearby. However, this wastes resources since forwards ants are not allowed to visit the same peer twice. For this reason, the *minResource* variation of the SEMANT algorithm is more favorable than the *maxResults* variation.

D. Exploration versus Exploitation

Now we address the exploitation-exploration dilemma by acquiring the simulation results for several different settings for parameter w_e. Since in the previous experiment it turned out to perform better, the *minResource* variation is employed. For the parameters other than w_e, the settings described in Sec. IV-C are used. The results of the comparison are given in Fig. 5 to Fig. 7. Fig. 5 shows that in the start-up phase, the resource usage is proportional to the ratio between exploring and exploiting strategy. The more forward ants employ the exploring strategy, the higher the amount of messages in the network. Consequently, for the overall load of the system it is better to employ a low rate of exploring forward ants, but also a high rate, e.g., $w_e = 0.05$, is feasible. Although in this case the resource usage at time unit 0 is much higher (168.4 hops on average) than in case of a low rate (e.g., 16.7 hops on

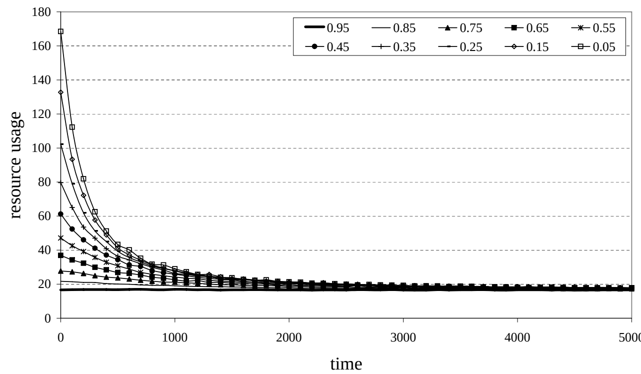

Fig. 5. Resource usage of SEMANT using the *minResources* variation when varying the value used for parameter w_e. Between time unit 0 and time unit 1500, resource usage is dependent on parameter w_e.

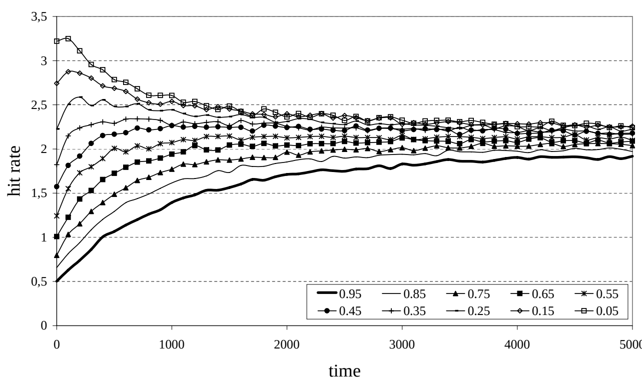

Fig. 6. Hit rate of SEMANT using the *minResources* variation when varying the value used for parameter w_e. The curves converge to the same limit.

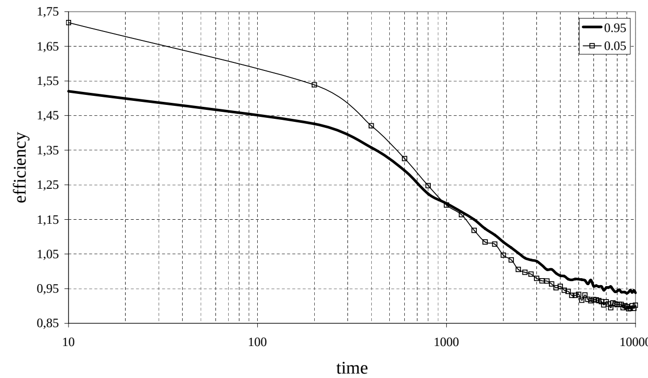

Fig. 7. Log-log plot of the efficiency of SEMANT using the *minResources* variation for parameter $w_e \in [0.05, 0.95]$. A higher rate of exploring ants results in slightly less efficient results in the start-up phase, but improves the performance in the converged phase. The curves for the other settings of parameter w_e lie in between those shown and are omitted for clarity. Note that the time axis is stretched by factor 2.

0.05	0.15	0.25	0.35	0.45	mean
11.43	11.31	11.33	11.45	11.25	$11,38619209$
0.55	0.65	0.75	0.85	0.95	standard deviation
11.35	11.37	11.42	11.38	11.57	$0,087454343$

TABLE II

AVERAGE EFFICIENCY OF SEMANT USING THE *minResources* VARIATION WHEN VARYING THE VALUE USED FOR PARAMETER w_e

average for $w_e = 0.95$), still not even half as many resources are consumed as when using broadcast with a time-to-live parameter of 4. The hit rate is dependent on parameter w_e not only in the start-up phase, but also in the converged phase (Fig. 6). The more exploring forward ants in the network, the better the pheromone trails and, consequently, the higher the hit rate. The best result can be reached for setting $w_e = 0.05$. After 5000 time units, 2.24 resources on average are found in this case. The worst possible result (1.92 resources on average after 5000 time units) is obtained for setting $w_e = 0.95$. However, the difference in absolute numbers is marginal and the curves converge to the same limit over time.

Combining the two metrics shows that, independently from the ratio between exploring and exploiting strategy, the overall efficiency in terms of hops necessary for answering a query is similar for every setting of parameter w_e. Although a higher rate of exploring ants gives slightly less efficient results in the start-up phase, at the same time it moderately improves the performance in the converged phase. Fig. 7 highlights this by depicting the efficiency results for $w_e = 0.05$ and $w_e = 0.95$ plotted in log-log scale. Table II – showing the average efficiency over the complete time span for every setting of parameter w_e – reveals that those two effects are in balance with each other. The mean of all values is 11.39 hops per query, and the low standard deviation of 0.087 indicates that there are no significant differences in average efficiency when varying the ratio between exploring and exploiting strategy. The reason for this effect lies in the design of the exploring strategy. It can be explained by the fact that, although the forward ants choose outgoing links that are not currently known as the best ones, they still make this decision in proportion to their desirability.

V. RELATED WORK

We identified three areas in which work related to ours can be found. First, the exploitation-exploration dilemma has been discussed in several contexts, such as in the case of foraging bees [21], in economic systems [22], in software product development [23], and others. Second, there is related work on ant algorithms in peer-to-peer networks.

Anthill [24] is a open source framework for the design, implementation, and evaluation of ant algorithms in peer-to-peer networks. There are two applications based on the Anthill framework. Gnutant [24] is a file-sharing application. In Gnutant, each file is identified by an unique file identifier and associated with meta-data comprised of textual keywords. Three different types of ants are responsible (1) for constructing a distributed index that contains URLs pointing to shared files and (2) for managing routing tables. If a user adds a new file to a nest, one *InsertAnt* is generated for each keyword of the file. InsertAnts propagate the presence of new files to the network by updating the distributed index. Gnutant utilizes hashed keyword routing based on the Secure Hash Algorithm (SHA). Each index entry contains the hash value of a keyword together with a set of nests that are likely to store files associated with the given keyword. *SearchAnts* are generated upon user queries and exploit the information stored in the routing tables in order to find files that match the queries' keywords. If no index entry that exactly matches the query's hash value exists, the SearchAnt selects the hash value that most closely matches the hash value of the query. If a SearchAnt localizes an appropriate file, it generates a *ReplyAnt* that immediately returns to the source nest and informs the user about the result of his or her query. Messor [25] implements load-balancing for peer-to-peer networks based on the necrophoric behavior of ants. Schelfthout and Holvoet [26] evaluate whether the principle of synthetic pheromone can be employed for coordination in distributed agent-oriented environments. Their framework is based on the idea of objectspaces known from concurrent computing. Similar to *Gnutant*, Schelthout et al. create pheromone trails for each query and allow agents to follow trails that represent one of the keywords in the query. In addition, an evaporation feature is integrated. Handt [27] tackles the problem of search in peer-to-peer networks by inverting it. Instead of peers issuing queries, the annotated resources move through the network and lay/follow pheromone trails that guide them. In addition, the peers re-order according to their interests which are determined by creating node profiles out of the meta-data their locally stored resources are marked-up with.

Third and finally, there is work addressing the applicability of biological processes other than ant-based methods to peer-to-peer networks, e.g., by Babaoglu et al. [28], who apply the principles of proliferation to search in unstructured overlay networks.

VI. Conclusion

In this paper we evaluated the effectiveness of the self-organizing and emergent behavior observed from natural ants for query routing in peer-to-peer networks. We specified two possible variations of the SEMANT algorithm based on the *Ant Colony Optimization* meta-heuristic and experimentally evaluated the performance of both variations in order to identify the superior one. After that, we addressed the exploration-exploitation-dilemma by showing that the overall efficiency of the SEMANT algorithm's search process is not influenced by the parameter value chosen for the ratio between exploring and exploiting forwards ants. This finding is important, because it simplifies the configuration of the algorithm and makes its behavior more easily predictable.

References

[1] Gianni Di Caro, Frederick Ducatelle, and Luca Maria Gambardella, "AntHocNet: An Ant-based Hybrid Routing Algorithm for Mobile and Ad Hoc Networks," in *Proceedings of Parallel Problem Solving from Nature*. September 2004, vol. 3242 of *Lecture Notes in Computer Science*, Springer.

[2] Gianni Di Caro and Marco Dorigo, "AntNet: Distributed Stigmergy Control for Communications Networks," *Journal of Artificial Intelligence Research (JAIR)*, vol. 9, pp. 317–365, July 1998.

[3] Sam Joseph and Takashige Hoshiai, "Decentralized Meta-Data Strategies: Effective Peer-to-Peer Search," *IEICE Transactions on Communications*, vol. E86-B, no. 6, pp. 1740–1753, June 2003.

[4] Elke Michlmayr, Arno Pany, and Sabine Graf, "Applying Ant-based Multi-Agent Systems to Query Routing in Distributed Environments," in *Proceedings of the 3rd IEEE Conference On Intelligent Systems (IEEE IS06)*, September 2006.

[5] Elke Michlmayr, "Ant Algorithms for Search in Unstructured Peer-to-Peer Networks," in *Proceedings of the Ph.D. Workshop, 22nd International Conference on Data Engineering (ICDE 2006)*, April 2006.

[6] Marco Dorigo and Gianni Di Caro, *New Ideas in Optimization*, chapter The Ant Colony Optimization Meta-Heuristic, pp. 11–32, McGraw-Hill, 1999.

[7] John H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.

[8] Stewart W. Wilson, "Explore/Exploit Strategies in Autonomy," in *From Animals to Animats 4: Proceedings of the 4th International Conference on the Simulation of Adaptive Behavior*, 1996, pp. 325–332.

[9] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[10] Matei Ripeanu, "Peer-to-Peer Architecture Case Study: Gnutella Network ," in *Proceedings of the 1st IEEE International Conference on Peer-to-Peer Computing (P2P2001)*, August 2001.

[11] Elke Michlmayr, "Specification of the SemAnt Algorithm," Tech. Rep., Vienna University of Technology, March 2006.

[12] Marco Dorigo and Luca Maria Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53–66, 1997.

[13] David E. Goldberg and Kalyanmoy Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in *Proceedings of the 1st Workshop on Foundations of Genetic Algorithms*, July 1990, pp. 69–93.

[14] Jon M. Kleinberg, "Navigation in a small world," *Nature*, vol. 406, pp. 845, August 2000.

[15] Association for Computing Machinery, "ACM Computing Classification System (ACM CCS)," 1998.

[16] Elke Michlmayr, Arno Pany, and Gerti Kappel, "Using Taxonomies for Content-based Routing with Ants," in *Proceedings of the Workshop on Innovations in Web Infrastructure, 15th International World Wide Web Conference (WWW2006)*, May 2006.

[17] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley Longman Publishing Co. Inc., 1999.

[18] Qin Lv, Pei Cao, Edith Cohen, and Scott Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," in *Proceedings of the 16th ACM Conference on Supercomputing*, June 2002, pp. 84–95.

[19] Elke Michlmayr, "Performance Evaluation of the SemAnt Algorithm," Tech. Rep., Vienna University of Technology, November 2006.

[20] Yael Niv, Daphna Joel, Isaac Meilijson, and Eytan Ruppin, "Evolution of Reinforcement Learning in Uncertain Environments: A Simple Explanation for Complex Foraging Behaviors," *Adaptive Behavior*, vol. 10, pp. 5–24, 2002.

[21] Lilia Rejeb and Zahia Guessoum, "The Exploration-Exploitation Dilemma for Adaptive Agents," in *Proceedings of the 5th European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS05)*, March 2005.

[22] Mikael Holmqvist, "Experiential Learning Processes of Exploitation and Exploration Within and Between Organizations: An Empirical Study of Product Development," *Organization Science*, vol. 15, no. 1, pp. 70–81, 2004.

[23] Ozalp Babaoglu, Hein Meling, and Alberto Montresor, "Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems," in *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 02)*. July 2002, IEEE.

[24] Alberto Montresor, Hein Meling, and Ozalp Babaoglu, "Messor: Load-Balancing through a Swarm of Autonomous Agents," in *Proceedings of the 1st International Workshop on Agents and Peer-to-Peer Computing*, July 2001.

[25] Kurt Schelfthout and Tom Holvoet, "A Pheromone-Based Coordination Mechanism Applied in Peer-to-Peer," in *2nd International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2003)*. July 2003, vol. 2872 of *Lecture Notes in Computer Science*, pp. 71–76, Springer.

[26] Arne Handt, "Self-Organizing Information Distribution in Peer-to-Peer Networks," in *Proceedings of New Trends in Network Architectures and Services: International Workshop on Self-Organizing Systems (IWSOS 2006)*, September 2006.

[27] Ozalp Babaoglu, Geoffrey Canright, Andreas Deutsch, Gianni Di Caro, Frederick Ducatelle, Luca Maria Gambardella, Niloy Ganguly, Márk Jelasity, Roberto Montemanni, and Alberto Montresor, "Design Patterns from Biology for Distributed Computing," in *Proceedings of the European Conference on Complex Systems (ECCS05)*, November 2005.