Using Taxonomies for Content-based Routing with Ants*

Elke Michlmayr
Women's Postgraduate College for Internet Technologies, Institute for Software Technology and Interactive Systems, Vienna University of Technology
michlmayr@wit.tuwien.ac.at

Arno Pany
Institute for Software Technology and Interactive Systems, Vienna University of Technology

Gerti Kappel
Business Informatics Group, Institute for Software Technology and Interactive Systems, Vienna University of Technology

ABSTRACT

Although the ant metaphor has been successfully applied to routing of data packets both in wireless and fixed networks, little is known yet about its appropriateness for search in peer-to-peer environments. This paper presents *SemAnt*, a distributed content-based routing algorithm based on the Ant Colony Optimization meta-heuristic and adapted for deployment in peer-to-peer networks. Under the assumption that content is annotated according to a taxonomy, it is possible to determine the hierarchical relationships between queries, and to exploit this information to improve the routing process. Our results show that using taxonomies enhances search performance in peer-to-peer networks. The degree of enhancement is highly dependent on the content distribution in the network.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing Protocols*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Information networks*; I.2.8 [**Artificial Intelligence**]: Problem solving, control methods and search backtracking—*Heuristics*

General Terms

Algorithms, Design, Experimentation

Keywords

Taxonomies, Metadata, Peer-to-Peer, Self-Organization, Ant Colony Optimization, Distributed Agents

1. INTRODUCTION

As advances in the research areas of information retrieval and search engines have shown, providing full-text search within centralized environments is feasible and shows good performance. On the contrary, full-text search in distributed environments is still a vision. A good overview of the state of the art in full-text search in peer-to-peer networks is provided by Zhong et al. [29]. The core of this research area is the design of full-text indexes for the resources of each peer, while the routing schemes used are usually quite simple.

The complement of distributed full-text search is distributed key lookup, where indexing is avoided by assigning an unique key to every resource and by providing lookup by key only using distributed hashtables [6]. The essence of this line of research lies in the design of sophisticated routing schemes that allow for efficient lookup of keys. As shown in [20], distributed hashtables can provide the basis for distributed full-text search.

Metadata-based search is the compromise between full-text search and key lookup. If the metadata the resources are annotated with are taken into account instead of their actual content, indexing a peer's content is simple. Hence, the major effort in this field is devoted to the design of sophisticated routing schemes that allow for keyword-based search. Most of the approaches for content-based or semantic routing fall in this category, since the main problem all content-based approaches have to face is the necessity to consider every query that can occur in the network, and to treat it based on its syntax and semantics. Thus, it is desirable to have a finite number of possible queries and it would be even better to know about the relationships between query keywords in order to simplify routing tables. The relationships between concepts/keywords can be defined with ontologies and/or taxonomies. In this paper, we investigate how and to which extent taxonomies can be applied for improving content-based routing. The investigation is carried out within the frame of our current work [22, 21] on applying ant algorithms to content-based search in peer-to-peer networks.

Contributions. The contribution of this paper is twofold. First, we show how to extend *SemAnt* – an ant algorithm for content-based search in peer-to-peer networks – in order to allow for integration of information from a taxonomy for routing. Second, we present experimental results indicating the performance of the algorithm in different application scenarios, with or without using information from a taxonomy for routing. In addition, we deliver an in-depth study of the impacts of different content distributions on the performance of the algorithm.

Paper overview. Section 2 provides a review of related work. Section 3 specifies *SemAnt*, the proposed ant algorithm. The experimental results are presented in Section 4.

*This research has partly 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.

WWW2006, May 22–26, 2006, Edinburgh, UK.
.

2. RELATED WORK

The work related to our approach can be grouped into three areas. In Section 2.1 we give an overview of approaches based on exploitation of query patterns occurring in the network. In Section 2.2 we provide a survey on how and for which purposes taxonomies have been used so far in peer-to-peer networks. Section 2.3 discusses previous attempts to apply self-organized behavior to peer-to-peer environments.

2.1 Query-pattern-based approaches

The most basic approaches to query routing in unstructured peer-to-peer networks are simple broadcasting techniques. Sophisticated methods exploit locally indexed information about user queries in the past, in order to predict which node is capable of answering a given query based on its keywords. Joseph and Hoshiai coin the term *reputation learning* for these techniques [15].

The basic premise behind reputation learning, which was first published by Cohen et al. [9], is that peers that would have been able to satisfy previous queries are more likely candidates to answer a current query which is similar. Another premise is that peers which share certain resources are more likely to be able to answer each other's queries because they have at least one common interest.

Sripanidkulchai et al. [26] use these premises to create shortcut links between peers with similar interests. Queries are then sent to the shortcut peers first. Only if there is no result, the query is flooded to the overlay network.

Cohen et al. [9] criticize that shortcuts focus on the first hop of query propagation only, and suggest that all peers which share a certain resources should form a so-called *possession-rule sub-overlay*. Each peer is a member of multiple overlays, and if it issues a query, the query is sent to one randomly selected overlay the peer is a member of.

Cholvi et al.'s [8] concept of *acquaintance links* uses query patterns occurring in the network to dynamically adapt the topology of the overlay network in order to create communities of peers that share similar interests in a self-organized manner. It turns out that the peers tend to organize into a super-peer structure [28], where powerful peers that have a big amount of resources at their disposal are better connected than less powerful peers.

Our approach is based on the exploitation of query patterns in the past as well, but does neither rely on shortcuts nor on topology adaption. Instead, we build a probabilistic overlay network for each query that occurs in the network. The approach that resembles our work most closely is the *InfoBeacons* system by Cooper [10], which is a middleware system designed for connecting uncooperative Web information sources by so-called beacons. Each beacon manages a small number of information sources and is in charge of sending user queries to appropriate sources. All query results are cached at the beacon as a set of pairs (W_i, CW_i^s), where W_i is a keyword and CW_i^s is the count of this word at source s. These statistics about past queries are exploited by the *ProbResults* ranking function to direct future queries to appropriate sources: For a query Q comprised of a set of n keywords, the function is computed as the product of $\prod_{i=1}^{n} CW_i^s/k_s$, where k_s is the number of queries previously sent to s. In addition, a heuristic called *experience weighting* provides for adaptation of cached information to changes at information sources. Although the beacons are very efficient in managing the information sources they are responsible for, the main drawback of this approach is that the beacons themselves do not cooperate. If a query can not be satisfied with the resources that are managed by one beacon, it is simply forwarded to another beacon without any attempts to make an intelligent guess which beacon is most likely to contain an answer.

2.2 Taxonomy- or ontology-based approaches

The main advantage of using taxonomies or ontologies for routing by content is that they provide a way to extract the relationship between different resources or queries and hence allow to define similarity measures. We can distinguish between approaches that require a summary of a peer's expertise in order to classify the peer, and more fine-grained approaches that rely on the classification of the peer's resources rather than on summaries.

In the first category, Crespo and Garcia-Molina [12] were the first to propose that the expertise of a certain peer should be classified according to a taxonomy, and that peers storing similar contents should be clustered into *semantic overlay networks*. Some others based their work on this idea, e.g., Löser [17], who investigated how a super-peer architecture and a distributed hashtable can be combined to create a distributed catalog of peer descriptions, or Schmitz [25], whose work deals with a network in which each peer is characterized by one concept or instance from a certain ontology and shows how the peers can organize themselves into a network structure that resembles the structure of the ontology.

The most notable piece of work in the second category is that of Tempich et al. [27] which was later refined and extended in [18]. Both of their approaches, namely *INGA* and *REMINDIN'*, are based on shortcut creation and borrow ideas from social networks to rank peers based on the information about queries in the past according to their likelihood to be able to answer a certain query and to select the best ranked node for each query. The shortcuts are stored in local indices. The size of a local index is limited and a combination of replacement strategies is used to keep the most useful shortcuts while removing the less valuable ones.

Pirredu and Nascimento [23] describe a peer-to-peer system in which all available content is annotated according to a balanced taxonomy. In a balanced taxonomy, all leaf concepts can be found at the same level of hierarchy. The peers (1) broadcast information about their contents and (2) manage a local index that stores aggregated information about the contents of its n-hops-neighborhood – where n is the height of the taxonomy – in different granularities depending on how many hops the peer is away. This approach is similar to *hop-count routing indices* as proposed by Crespo and Garcia-Molina [11]. Based on these indices, queries can be routed effectively.

2.3 Emergent behavior and peer-to-peer networks

The most comprehensive source of information about the applicability of biological processes to distributed environments is provided by Babaoglu et al. in [4], including a discussion of ant-based methods in context and an attempt to apply the biological process of proliferation to search in unstructured overlay networks.

An abstract formulation of the basic ideas behind emergent behavior and self-organization can be found in *emergent semantics* by Aberer et al. [2], a well-known theoreti-

cal framework for semantic interoperability between loosely coupled information sources. They suggest to use a semantic handshake protocol to allow for negotiations between pairs of peers to reach an agreement over the meaning of models, e.g., by local schema mapping [1]. Similar to the ant metaphor, (1) negotiations are local interactions whenever possible, (2) global agreements are obtained by aggregating local agreements, and (3) the negotiations are influenced by the context of the existing global agreements.

The most well-known practical approach is the *Anthill* [5] framework for the design, implementation, and evaluation of ant algorithms in peer-to-peer networks. Although – for demonstration purposes – Anthill was used to build a file-sharing application called *Gnutant*, the projects focuses on the development of the framework and does not design algorithms. This task is left to the users of the framework.

The open-source project *MUTE* [24] implements a peer-to-peer system that relies on the ant metaphor for user discovery. Ants are used to track which neighbor connections are associated with particular sender addresses. Distributed search is based on controlled flooding to locate files by name based on free-form query strings.

3. ALGORITHM

In this section we describe the *SemAnt* algorithm. While Section 3.1 describes the building blocks of distributed ant algorithms, Section 3.2 shows how to adapt them to peer-to-peer environments. Section 3.3 explains how taxonomies are used for content-based routing. Section 3.4 specifies the *SemAnt* routing procedure. Section 3.5 points out how the pheromone trails are updated when documents are found. Section 3.6 explains how the evaporation feature is used.

3.1 Distributed ant-based methods

Ant algorithms are inspired by the collective foraging behavior of specific ant species which use a chemical substance called *pheromone* for communication. Since pheromone-based communication is indirect, there is no need for global knowledge about the network. This qualifies ants algorithms for application in peer-to-peer networks.

Although the primary application area for Ant Colony Optimization is in solving graph-based optimization problems in a centralized manner, a considerable amount of work has been devoted to solving distributed problems with the ant metaphor. The most prominent distributed ant algorithm, *AntNet* [7] by Di Caro and Dorigo, is designed for routing of data packets in IP networks. In the following we provide an introduction to the basic principles of *AntNet*. Consider a directed weighted graph having N nodes. The edges of the graph are the links between nodes and are viewed as bit pipes with a certain cost (bandwidth and transmission delay) that depends on the current load of this link. Each node manages a routing table that stores information about the outgoing links and their amount of pheromone. The routing tables are matrices of size $N \times l$, where N is the number of nodes in the network and l is the number of outgoing links. The routing tables entries for all reachable nodes are initialized with equal values. At regular intervals, each node P_S generates a so-called forward ant (see Figure 1) that builds a path to a randomly selected destination node P_D by applying a so-called *transition rule* at each node. The transition rule decides which outgoing link is to follow based on link costs and pheromone amounts. The better the link

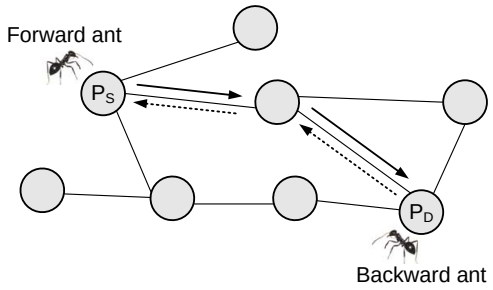

Figure 1: Forward and backward ants

in terms of low link costs and a high amount of pheromone, the higher the probability that it will be selected. When a forward ant has reached its destination node, it calculates the total link costs of the path it used. The lower the total link cost, the more additional amount of pheromone should be dropped to the path. Since each node stores routing information locally, the forward ant cannot update the pheromone trails directly. Instead, it generates a backward ant that returns to the source node through the same path that was used by the forward ant. The backward ant is responsible for updating the pheromone trail (see dotted arrows in Figure 1) according to the information gathered by the forward ant by altering the routing table of each visited node. The backward ant updates only those entries that refer to the destination node. Subsequent forward ants make their routing decisions based on the altered values.

AntNet includes a strategy for preventing cycles. Each forward ant manages a stack of nodes already visited. Each time it has to decide which node to visit next, it excludes all already visited nodes. If it detects a cycle because it is forced to go to an already visited node N_v since all possible next nodes were already visited, it calculates the time span t it spent inside the circle. If t is greater than 50% of the ant's total lifetime, it is terminated. Otherwise, the ant removes all nodes that are part of the cycle from its stack and continues traveling at node N_v.

3.2 Adapting the ant metaphor to peer-to-peer networks

The application scenario for the *SemAnt* algorithm is that of a distributed search engine where each peer manages a repository of documents, offers its content to the other peers, and issues queries to the network. For these purposes, we adopt (1) *AntNet*'s concept of supporting distributed problems with forward ants and backward ants and (2) its strategy for preventing cycles. The difference is that – instead of sending out forward ants at regular intervals – we create a forward ant for each query that occurs in the network. This ant is responsible for answering the query. Figure 2 shows the *SemAnt* algorithm in pseudo-code. If a forward ant arrives at a peer that stores documents satisfying the query, it creates a backward ant and terminates its travel. In addition, we define a time-to-live (TTL) parameter T_{max} to prevent forward ants from running infinitely. Consequently, if a backward ant was created, it will arrive at the querying peer within a time interval of $2 * T_{max}$. If it does not, then there is no result for the query.

In a previous version of the algorithm [22], we used a different approach where an ant always used its maximum TTL

```
# Initialization
t = CurrentTime;
t_end = EndTimeOfSimulation;

# Concurrent activity
foreach (Node) {
    initializePheromoneTables();
    while (t <= t_end) {

        # Concurrent activity at each node
        in_parallel {
            if (Query Q) {
                checkLocalDocumentRepository();
                if (DocumentsFound == 0) {
                    createForwardAnt(query_parameter);
                }
            }

            foreach (ForwardAnt) {
                while (Timeout_not_reached) {
                    applyTransitionRule();
                    foreach (ForwardAnt) {
                        GoToNode(N_j);
                        checkLocalDocumentRepository();
                        if (DocumentsFound > 0) {
                            createBackwardAnt(Stack, N_j);
                            terminate();
                        }
                        addDataToStack(N_j);
                    }
                }
            }

            foreach (BackwardAnt) {
                do {
                    node = popStack_Node();
                    GoToNode(node);
                    applyPheromoneTrailUpdateRule();
                } while (node <> source_node);
            }

            foreach (PeriodicalTimeInterval t_e) {
                applyEvaporationRule();
            }
        }
    }
}
```

Figure 2: *SemAnt* algorithm in pseudo-code

and continued its search after finding the first result node. The assumption behind this approach was that if forward ants are allowed to go on after the first result node, they can find other result nodes and generate multiple backward ants. Actually, our experiments showed that ants tend to stay in the neighborhood of the node they already found, since the pheromone trails indicate that there is an appropriate node nearby. For this reason, letting ants go on after they found the first result does not enhance performance.

3.3 Taxonomies for routing by content

Unlike in routing of data packets, where the destination node is known and the content of the packets is irrelevant, in query routing it is essential to consider the content of a query – its syntax and semantics – to find an appropriate destination node. Similar to the data structures in *AntNet*, we employ a two-dimensional data structure for routing tables, but for our purposes each row of the matrix corresponds to a certain query that can occur in the network, rather than corresponding to a specific destination node.

We rely on taxonomies for restricting the maximum size of the routing tables. We assume that the contents of the document repositories are annotated according to the concepts of a taxonomy. Each document can be an instance of one or more concepts. Query keywords are restricted to the concepts of a taxonomy as well. For simplicity, we assume that a query Q consists of just one concept c. A document is an appropriate result for a given query Q if it is an instance of concept c. Supporting queries with multiple keywords would be possible by building average values of the corresponding pheromone amounts.

At each peer P_i, pheromone trails are maintained in a table τ of size $C \times n$, where C is the number of concepts in the taxonomy 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_{n_u}, for each concept c and each neighbor peer P_{n_u}. At startup, all table entries are initialized with the same value $\tau_{init} = 0.009$. Initialization with a small value prevents divisions by zero and is necessary for the evaporation feature (see Section 3.6).

3.4 Routing strategies

An ant algorithm's routing strategy is defined by its transition rule. The transition rule used in the *SemAnt* algorithm is an adapted version of one used in *Ant Colony System* [13]. It consists of two strategies that complement each other. Based on probability $w_s \in [0, 1]$ and a random value $q \in [0, 1]$ that is calculated each time a forward ant has to select an outgoing link, either an exploiting strategy (see Section 3.4.1) or an exploring strategy (see Section 3.4.2) is applied. While the exploiting strategy weighted with w_s solely utilizes the results gained so far, the exploring strategy weighted with $1 - w_s$ provides for mutability and randomness. Obviously, more weight should be put on the exploiting strategy, i.e., $w_s \in [0.5, 1]$.

3.4.1 Exploiting strategy

The exploiting strategy selects the link to the neighbor peer P_j with the highest quality. To measure the usefulness of the underlying taxonomical information for routing, we use two different variants of this strategy.

In the first variant, which is shown in (1a), we consider the hierarchical structure of the pheromone trails. The strategy incorporates not only the pheromone trail for concept c, which is the keyword of query Q, but also those trails that correspond to the super-concepts of concept c. To put more emphasis on the trails that directly match the query, a multiplication factor is used.

$$j = arg\ max_{u \in U \wedge u \notin S(F^Q)} \left(\sum_{i=0}^{n-1} \tau_{c_i u} \cdot \frac{1}{x^i} \right) \qquad (1a)$$

In (1a), U is the set of neighbor peers of P_i, and $S(F^Q)$ is the set of peers already visited by forward ant F^Q, i is the distance between the super-concept c_i and concept c, $c_0 = c$, n is the distance between concept c and the top-level concept, and $x \in [2, 4, 8, 16, ...]$. Parameter n defines how many superconcepts are considered and parameter x to which extent they are incorporated. Appropriate values for n and x must be found out in the experiments.

In the second variant, we treat all pheromone trails for different concepts as being independent from each other. The ants simply select the link that stores the highest amount of pheromone for the given keyword as shown in (1b).

$$j = arg\ max_{u \in U \wedge u \notin S(F^Q)} \tau_{cu} \quad (1b)$$

In (1b), U is the set of neighbor peers of P_i, and $S(F^Q)$ is the set of peers already visited by F^Q.

3.4.2 Exploring strategy

The exploring strategy encourages the forward ants to discover new paths by taking not only the best but all neighbors into account. It consists of two steps. The first step, which is shown in (2), is to derive a goodness value p_j for each neighbor peer P_j not already visited.

$$p_j = \frac{\tau_{cj}}{\sum_{u \in U \wedge u \notin S(F^Q)} \tau_{cu}} \quad (2)$$

where the sum of all goodness values $\sum p_j = 1$, U is the set of neighbor peers of P_i, and $S(F^Q)$ is the set of peers already visited by F^Q.

In the second step, which is shown in (3), an adapted version of the roulette wheel selection technique [14] is applied for selecting peers: 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 P_j should be selected. This strategy allows more than one peer to be selected to account for the fact that there are multiple possible destination peers which contain answers for a query.

$$GOTO_j = \begin{cases} 1 \text{ if } q \leq p_j \wedge j \in U \wedge j \notin S(F^Q) \\ 0 \text{ else} \end{cases} \quad (3)$$

where q is a random value and $q \in [0, 1]$. If $GOTO_j = 1$, the forward ant sends a clone of itself to peer P_j.

To ensure that at least one peer will be selected, the algorithm falls back to the exploiting strategy if applying the exploring strategy would result in not selecting any peer.

Since we allow ants to duplicate themselves, it can happen that two ants representing the same query sequentially arrive at the same peer. Each peer keeps track of which ants it was already visited by, and terminates duplicated ants.

3.5 Pheromone updates

If a forward ant arrives at a certain peer P^D storing appropriate result documents D, it generates a backward ant and supplies it with D and with a copy of the stack that contains all visited peers $S(F^Q)$. After generating the backward ant, the forward ant terminates. The backward ant calculates the sum of all entries in $S(F^Q)$ to get the total number of hops T_D for the path from the querying peer P^Q to peer P^D, and travels back hop-by-hop according to the information stored in $S(F^Q)$ until it arrives at peer P^Q. If this is not possible because of a topology change, the backward ant terminates and the result will not be transported to the querying peer.

At each intermediate peer, the backward ant drops pheromone on the link previously selected by the forward ant. The amount of pheromone depends on the goodness of the path. The goodness is determined by comparing the number of documents found and the length of the path to optimal values. The pheromone update rule is adopted from [13] and defined as shown in (4), (5), and (6). As shown in (4) and (5), equal amounts of pheromone are dropped on the pheromone trail for concept c – which is the keyword of query Q – and on the pheromone trails corresponding to the super-concepts c_i of concept c.

$$\tau_{c_i j} \leftarrow \tau_{c_i j} + Z, \quad (4)$$

where

$$i = \{0, \ldots, n-1\},\ c_0 = c \quad (5)$$

and

$$Z = w_d \cdot \frac{|D|}{D_{opt}} + (1 - w_d) \cdot \frac{T_{max}}{2 \cdot T_D} \quad (6)$$

As shown in (6), the values for the optimal solution are set to $\frac{1}{2} \cdot T_{max}$ for the path length and D_{opt} for the number of documents. Parameter w_d weights the influence of document quantities and path lengths.

3.6 Evaporation

The evaporation feature, after taking pheromone amounts into account for the routing strategy and updating them according to the goodness of result paths, is the third constituent of pheromone management in ant algorithms. To avoid unlimited increments in pheromone amount, a certain percentage of pheromone vanishes over time. In addition, evaporation prevents that paths become too dominant: Since the pheromone amount is lowered by multiplication of the current value with a constant, evaporation removes more pheromone from trails that store a high amount than of those that store a low amount.

$$\tau_{cu} \leftarrow (1 - \rho) \cdot \tau_{cu} \quad (7)$$

In *SemAnt*, each peer applies the evaporation rule shown in (7) in a predefined interval 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 $\rho \in [0, 1]$.

4. SIMULATION AND RESULTS

In this section we present the setup and the results of the experimental evaluation of using taxonomies in combination with the *SemAnt* algorithm. We focus on comparing the performance of the algorithm with regard to different content distributions, and evaluate the impact of using taxonomical relationships between keywords for routing. Section 4.1 depicts the metrics and the parameter values that were used. Section 4.2 describes the simulation setup and shows the results.

4.1 Metrics and parameter settings

We have already shown in [21] that *SemAnt* exhibits good performance in comparison to standard approaches, i.e., the *k-random walker* algorithm [19]. In this paper, we focus on evaluating the performance of the algorithm in two different settings, and on determining the discrepancy between considering taxonomical relationships between concepts (cf. Equation (1a) in Section 3.4.1) and treating concepts as unrelated keywords (cf. Equation (1b) in Section 3.4.1) for each of these settings. Initial experiments were performed to determine parameter n, which is the optimal number of superconcepts that should be taken into account for routing. It turned out that the best value is to set $n = 2$. This means that only the direct superconcept is specific enough

for improving performance. If higher-level superconcepts are taken into account as well, performance is actually lowered. The metrics used for performance evaluation are the following:

- *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 documents 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 documents found, we get the average number of links traveled to find one document, which is the most practical metric.

Table 1 shows the values chosen for the input parameters of *SemAnt*. To make comparisons possible, we use the same parameter values for all different content distributions. As we also found out in our initial experiments, resource usage is highly dependent on weight w_s. This parameter defines the ratio between ants using the exploring and ants using the exploiting strategy. The more ants employ the exploring strategy, the more traffic occurs in the network, but not necessarily leading to proportionally better hit rates. The best trade-off between resource usage and hit rate is obtained when setting parameter $w_s = 0.85$.

4.2 Setup and experimental results

We simulate a peer-to-peer network with 1024 peers. A static network topology and document distribution is assumed. The system supports cooperation between computer scientists by sharing documents. In order to choose a realistic model for social networks, a small world network [16] with a clustering coefficient of 2 is used. Similar to [23], we model the content with the ACM Computing Classification System [3] taxonomy. Each document is an instance of a certain leaf concept. In total, ACM CCS contains 910 leaf concepts. To represent each research topic equally, we create the same number of documents for each leaf concept. We create 34 documents per concept and thus get 30940 documents in total. For uniform distribution of queries within the network, a ticker clock at each peer is used. The probability that a peer issues a query within one time unit is set to 0.1. The keyword of the query consists of a randomly selected leaf concept from the ACM CCS taxonomy. For distributing the documents across the network, we adopt the assumption that each peer is an expert on a certain topic and therefore a certain percentage P_{expert} of its documents are instances of one particular research area. A research

ρ	evaporation factor	0.07
T_{max}	timeout of forward ants	25
w_s	weight of exploiting and exploring strategy	0.85
D_{opt}	optimal number of documents	10
w_d	weight of document quantity and path length	0.5
n	number of superconcept pheromone trails incorporated (including concept itself)	2

Table 1: Parameters of *SemAnt*

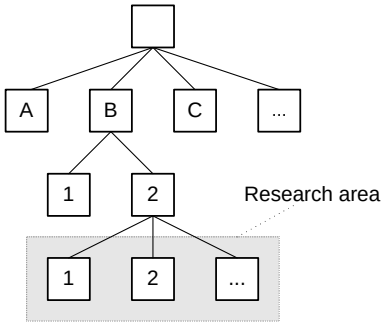

Figure 3: Third-level concepts and their leaf concepts model research areas

area is modeled by the third-level concepts of the taxonomy and their leaf concepts, as shown in Figure 3.

If $P_{expert} = 100\%$, all the documents stored at a certain peer belong to the same research area. If $P_{expert} < 100\%$, those documents that are not stored at an expert peer are randomly and individually spread in the network. Consequently, in case of $P_{expert} = 0\%$ all documents are spread randomly across the network.

Scenario 1 In the first scenario, we set $P_{expert} = 100\%$ and compare the effects different numbers of experts have on the overall performance. In the first case, there are 6 experts per research area in the network, while in the second case, there are 12 experts. Figure 4 shows the results for the hit rate in scenario 1, Figure 5 those for the resource usage, and Figure 6 the combined measure of efficiency. Because of $P_{expert} = 100\%$, the corresponding superconcept has the same significance for a query as the concept itself. Thus, it is irrelevant to which extend superconcepts are considered, that is, which value is used for parameter x. The figures show the results for setting $x = 4$.

What can be seen is that the results are the better the less experts exist. For 6 experts, the hit rate is about 5.5 documents. For 12 experts, it is about 3 documents. This is obvious, since the more experts exist, the less documents are stored at one expert peer. As soon as the converged phase is reached, the results for the hit rate (Figure 4) are nearly the same, no matter whether the superconcept is considered or not. The discrepancy is that the algorithm converges faster if the superconcept trails are included, and that the performance gain is higher in case of more experts.

As shown in Figure 5, the values for resource usage are proportionally lower if superconcept trails are included. The difference is rather small (approx. 3 messages in both cases), but permanent, because it is still present after 5000 time units. The resources used by the backward ants are included in these figures. This is the reason why the resource usage is higher in case of 6 experts, because more documents are found and thus more backward ants are created.

To summarize, the algorithm is very efficient in all four settings (Figure 6) where $P_{expert} = 100\%$. Using the superconcept trails for routing gives nearly the same results as not using them, but it additionally lowers the time it takes to reach the converged phase (approx. 500 time units when using superconcepts, approx. 1000 time units without).

Scenario 2 In the scenario described above, all documents were stored at expert peers. This is the optimal setting for using superconcepts for routing. In this scenario, we

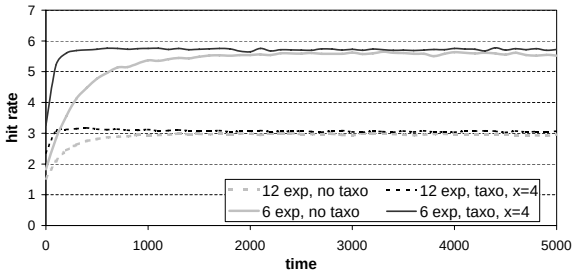

Figure 4: Scenario 1, hit rate

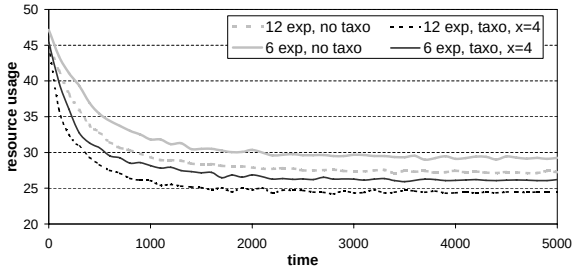

Figure 5: Scenario 1, resource usage

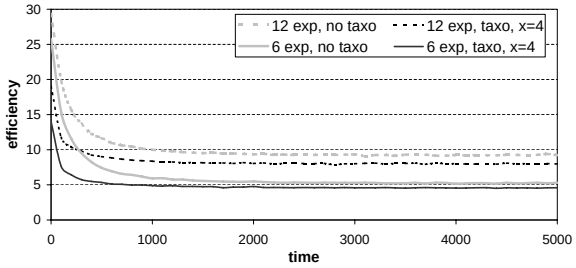

Figure 6: Scenario 1, efficiency

want to investigate the effects that dispersed document distributions have on the performance. We acquire the results for $P_{expert} = [0\%, 20\%, 40\%, 60\%, 80\%, 100\%]$ and compare them. Figure 7 shows the values for efficiency when relying on the pheromone trails for concepts only. As expected, the algorithms' performance is the weaker, the more disperse the document distribution is. This is a problem that every content-based approach has to face.

Figure 8 shows the results when the pheromone trails for superconcepts are considered. We use $x = 4$ for all simulations to ease comparison. It turns out that using superconcept pheromone trails for routing has a positive impact on performance as long as at least some patterns can be found in the document distribution. Solely in case of $P_{expert} = 0\%$, where all documents are individually stored at randomly chosen peers, considering the superconcept lowers performance. The most significant improvement can be found in case of $P_{expert} = 60\%$, where the efficiency value is 39.5% better after 5000 time units. Second best is $P_{expert} = 80\%$, where improvement is 38,2%. For $P_{expert} = 40\%$, it is still significant with 30,6%. The improvement for $P_{expert} = 100\%$ is 13,5%, and 7,2% for $P_{expert} = 20\%$. Note that the lower the value for P_{expert}, the longer it takes until a converged phase is reached. Hence, for $P_{expert} = 20\%$, not considering superconcepts outperforms considering them until time unit 2500 is reached.

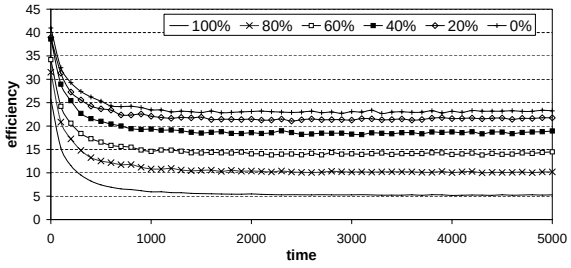

Figure 7: Scenario 2, not using superconcepts

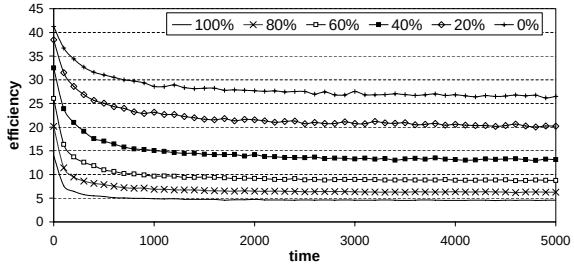

Figure 8: Scenario 2, using superconcepts (x=4)

5. CONCLUSION AND FUTURE WORK

In this paper we presented an improved and extended version of the *SemAnt* algorithm. *SemAnt* uses a probabilistic model, in which every query leaves a small trace in the overlay network, and the summation of all traces can be used for subsequent queries as a hint which link to follow in order to find appropriate results. The focus of this paper was to investigate how hierarchical relationships between query keywords can be employed for upgrading routing performance. We compared the effectiveness of the algorithm when using a flat namespace for keywords to its effectiveness when using a hierarchic namespace. The results show that *SemAnt* exhibits stable and robust results and that its performance can be significantly improved by considering superconcept pheromone trails for routing. Since the performance of content-based approaches to routing heavily depends on how the content is distributed within the network, we used different settings for the content distribution to find out how much coherence in the content is necessary.

Our plans for the future are to conduct additional comparisons to other state-of-the-art approaches than k-random walker. Moreover, we will consider different query distributions within the network. Finally, we will publish the performance figures when using other than the optimal values for the parameters of *SemAnt*.

6. ACKNOWLEDGEMENTS

We are grateful to Sabine Graf for sharing her expertise in ant algorithms with us. We thank Veronika Stefanov and Martina Umlauft for helpful comments. We thank Horst Eidenberger and Martina Umlauft for letting us use their hardware resources for the experiments.

7. REFERENCES

[1] K. Aberer, P. Cudre-Mauroux, and M. Hauswirth. The Chatty Web: Emergent Semantics Through Gossiping. In *Procceedings of the 12th International World Wide Web Conference (WWW 2003)*, May 2003.

[2] K. Aberer, P. Cudre-Mauroux, A. M. Ouksel, T. Catarci, M.-S. Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, O. D. Troyer, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, and R. Studer. Emergent Semantics Principles and Issues. In *Procceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, March 2004.

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

[4] O. Babaoglu, G. Canright, A. Deutsch, G. Di Caro, F. Ducatelle, L. M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, and A. Montresor. Design Patterns from Biology for Distributed Computing. In *Proceedings of the European Conference on Complex Systems*, 2005.

[5] O. Babaoglu, H. Meling, and A. 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*, 2002.

[6] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking Up Data in Peer-to-Peer Systems. *Communications of the ACM*, 46(2):43–48, 2003.

[7] G. D. Caro and M. Dorigo. AntNet: Distributed Stigmergy Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[8] V. Cholvi, P. Felber, and E. Biersack. Efficient Search in unstructured peer-to-peer networks. *European Transactions on Telecommunications*, 15:535–548, 2004.

[9] E. Cohen, A. Fiat, and H. Kaplan. A Case for Associative Peer to Peer Overlays. *ACM SIGCOMM Computer Communication Review*, 33(1):95–100, January 2003.

[10] B. F. Cooper. Guiding Queries to Information Sources with InfoBeacons. In *Middleware 2004, ACM/IFIP/USENIX International Middleware Conference*, 2004.

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

[12] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Stanford University, November 2002.

[13] M. Dorigo and L. M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66, 1997.

[14] D. E. Goldberg and K. Deb. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In *Proceedings of the 1st Workshop on Foundations of Genetic Algorithms*, pages 69–93, 1990.

[15] S. Joseph and T. Hoshiai. Decentralized meta-data strategies: Effective peer-to-peer search. *IEICE Transactions on Communications*, E86-B(6):1740–1753, 2003.

[16] J. M. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.

[17] A. Löser. Towards Taxonomy-based Routing in P2P Networks. In *Proceedings of the 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing, 13th International World Wide Web Conference (WWW2004)*, May 2004.

[18] A. Löser, C. Tempich, B. Quilitz, S. Staab, W.-T. Balke, and W. Nejdl. Searching Dynamic Communities with Personal Indexes. In *Proceedings of the 4th International Semantic Web Conference*, 2005.

[19] Q. Lv, P. Cao, E. Cohen, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th ACM Conference on Supercomputing*, 2002.

[20] S. Michel, P. Triantafillou, and G. Weikum. Minerva∞: A scalable efficient peer-to-peer search engine. In *Middleware 2005, ACM/IFIP/USENIX, 6th International Middleware Conference*, November 2005.

[21] E. Michlmayr. Ant Algorithms for Search in Unstructured Peer-to-Peer Networks. In *Proceedings of the Ph.D. Workshop, 22nd International Conference on Data Engineering (ICDE2006)*, April 2006.

[22] E. Michlmayr, S. Graf, W. Siberski, and W. Nejdl. Query Routing with Ants. In *Proceedings of the 1st Workshop on Ontologies in P2P Communities, 2nd European Semantic Web Conference 2005 (ESWC2005)*, May 2005.

[23] L. Pireddo and M. A. Nascimento. Taxonomy-Based Routing Indices for Peer-to-Peer Networks. In *Proceedings of the Workshop on Peer-to-Peer Information Retrieval, 27th International Annual ACM SIGIR Conference*, July 2004.

[24] J. Rohrer. Mute: Simple, anonymous file sharing. `http://mute-net.sourceforge.net/`, 2005.

[25] C. Schmitz. Self-Organization of a Small World by Topic. In *Proceedings of the 1st International Workshop on Peer-To-Peer Knowledge Management, MobiQuitous 2004*, August 2004.

[26] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *Proceedings of IEEE INFOCOM*, April 2003.

[27] C. Tempich, S. Staab, and A. Wranik. REMINDIN': Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors. In *Proceedings of the 13nd International World Wide Web Conference*, 2004.

[28] B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of the 19th Internation Conference on Data Engineering (ICDE2003)*, March 2003.

[29] M. Zhong, J. Moore, K. Shen, and A. Murphy. An Evaluation and Comparison of Current Peer-to-Peer Full-Text Keyword Search Techniques. In *Proceedings of the 8th International Workshop on the Web & Databases (WebDB2005), ACM SIGMOD/PODS 2005 Conference*, June 2005.