Applying Ant-based Multi-Agent Systems to Query Routing in Distributed Environments

Elke Michlmayr, Arno Pany, Sabine Graf

Abstract— **This paper presents *SemAnt*, a novel ant-based multi-agent system designed for distributed query routing. While the ant metaphor has been successfully applied to network routing both in wireless and fixed networks, little is yet known about its applicability to the task of query routing in distributed environments. We point out the similarities and dissimilarities between routing of data packets and routing of queries, and we present the design of *SemAnt*, which is based on the Ant Colony Optimization meta-heuristic. For experimental evaluation, we deploy the algorithm in a peer-to-peer environment with a real-world application scenario and compare its performance against the well-known k-random walker approach. As we will show, the benefits of *SemAnt* are that the routes for queries are optimized according to their popularity, and that the algorithm is highly suitable for volatile environments.**

Index Terms— **Cooperative Artificial Intelligence Systems, Distributed Artificial Intelligence, Multi-Agent Systems, Peer-to-Peer Networks, Ant Colony Optimization**

I. Introduction

Network routing refers to the process of selecting the next hop for an incoming data packet being forwarded based on information held in routing tables. Ant-based trail-laying and trail-following algorithms have successfully been applied to routing in distributed systems. Based on the Ant Colony Optimization (ACO) meta-heuristic [1] proposed by Di Caro and Dorigo in 1999, a dedicated subset of ant algorithms was specifically designed for managing routing tables in telecommunication networks [2] and in mobile ad-hoc networks [3]. Query routing in distributed environments is the task of finding one or more appropriate destinations for a given query. Each node in the network manages an information repository that can be accessed by the other nodes. Each node issues queries to the network. All nodes collaborate to answer the queries. The aim is to maximize the number and the quality of query results while minimizing the overhead necessary for management of routing the queries.

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.

Elke Michlmayr is with the Women's Postgraduate College for Internet Technologies (WIT), Institute for 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).

Arno Pany is currently a master's student at the Institute for Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11/E188, 1040 Vienna, Austria (email: e9425000@stud1.tuwien.ac.at).

Sabine Graf is with the Women's Postgraduate College for Internet Technologies (WIT), Institute for Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11/E188, 1040 Vienna, Austria (telephone: 0043 1 58801 18817, email: graf@wit.tuwien.ac.at).

Different approaches for query routing in distributed environments exist, ranging from simple broadcasting techniques to sophisticated methods that exploit stored information about user-generated queries in the past in order to predict – based on a given query's keywords – which node is capable of answering it. Ant-based methods are based on the same basic principle. They have proven to perform well for network routing [4] but have not yet been applied to the task of query routing. The differences between routing of data packets and query routing are as follows. First, when routing data packets, the destination address of a packet is known in advance. In query routing, it is not. Second, network routing does not account for the contents of data packets. In query routing, it is of benefit to consider the keywords of a query when deciding which node to send the query to. Despite these differences, we assume ant-based routing techniques to be suitable for query routing in distributed environments for the reasons of *decentralization* and *adaptivity*. Ants are autonomous agents that travel through a network and spread pheromone on their paths. At each node, they examine the pheromone amounts already spread on the available outgoing links in order to decide which link to follow. Communication among ants is indirect and exclusively based on modifications of pheromone trails. Hence, ant algorithms do not require any global knowledge about the network and can therefore be applied in peer-to-peer environments. Since strategies for reacting to network topology changes exist, ant algorithms are suitable for volatile networks, where nodes can fail/leave or join at any time. The strategies to cope with these dynamics consist of adapting the pheromone trails in the network according to changes in topology.

The paper is organized as follows. Section II describes ant-based methods for routing. Section III discusses previous research on employing ant-based methods for query routing. Section IV gives a detailed description of *SemAnt*, the proposed ant algorithm. Section V presents experimental results that show the performance of the algorithm.

II. Ant-based algorithms for Routing

In this section we describe ant algorithms, which are inspired by the collective foraging behavior of specific ant species. Summarized by a meta-heuristic called *Ant Colony Optimization* [1], several algorithms exist that model and exploit this behavior for solving graph-based NP-hard combinatorial optimization problems. In these algorithms, after initializing each edge of the problem graph with a very small amount of pheromone and defining each ant's starting node, a small number of ants runs for a large number of

iterations. For every iteration, each ant determines a path through the graph from its starting to its destination node by applying a so-called *random proportional transition rule* at each decision point. This rule derives which of all possible next nodes to choose, based on (1) the specific edge's amount of pheromone and (2) its costs. When the ant arrives at the destination node, the total costs of the newly found solution are calculated. After all ants have found a solution, the *pheromone trail update rule* is applied for each edge which is part of the solution. The amount of newly dropped pheromone depends on the quality of the solution. In each iteration, some pheromone evaporates according to an evaporation factor.

The most prominent variant of ant algorithms for routing is *AntNet* [4] by Di Caro and Dorigo. AntNet is designed for packet-switched networks and its pheromone updating approach is appropriate for both symmetric and asymmetric networks. In AntNet, ants collaborate in building routing tables that adapt to current traffic in the network with the aim of optimizing the performance of the entire network. The network is mapped on a directed weighted graph with N nodes. Each node manages a routing table that stores information about the outgoing links and their amount of pheromone. The edges of the graph are the links between nodes and are viewed as bit pipes having a certain cost – bandwidth and transmission delay – that depends on the current load of this link. The routing tables are matrices of size $N \times l$, where l is the number of outgoing links. At startup, all routing tables are initialized with a uniform distribution of all reachable nodes. At regular intervals, each node generates a so-called forward ant that builds a path to a randomly selected destination node. Again, the decision about which node to choose next is based on (1) the link costs of the outgoing links and (2) the amount of pheromone already dropped on these link in previous iterations. When a forward ant F_{sd} launched at a source node N_s has reached its randomly selected destination node N_d and calculates the total costs of the solution, it cannot update the pheromone trails directly. Since the problem is distributed, it has to generate a backward ant B_{ds} that will return to node N_s through the same path that was used by the forward ant. The backward ant is responsible for updating the pheromone trail according to the information gathered by the forward ant by altering the routing table of each visited node. The backward ants update all entries corresponding to destination node N_d. For preventing cycles, each forward ant F_{sd} manages a stack of nodes already visited. 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, F_{sd} is terminated. If it is less, F_{sd} removes all nodes that are part of the cycle from its stack and continues traveling at node N_v.

The main reason that makes it impossible to apply AntNet to query routing in peer-to-peer environments is that it needs information about all existing nodes in the network for choosing destination nodes.

III. Related work

In the following we give an overview of existing techniques for query routing in peer-to-peer networks. In addition, we discuss the relationship between multi-agent systems and peer-to-peer networks, and we evaluate previous attempts to apply ant-based methods for query routing in peer-to-peer networks.

Peer-to-peer systems are "distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content [...] without requiring the intermediation or support of a global centralized server or authority" [5]. Different approaches for query routing in peer-to-peer networks exist (see [6] for a survey). The most basic technique is to use a central index of all documents. The drawback of this approach is that all nodes in the network have to register their contents at a central node. Another basic technique is not to use indexes at all, but to broadcast each query to the network within a limited scope. The scope of the broadcast is defined by the number of hops that lie between the querying and the answering node. Both techniques provide only limited scalability.

Sophisticated approaches employ indexes that are distributed among the nodes to allow for fault tolerance and scalability. The first category of these approaches relies on distributed hash tables (DHT) for creating structured overlay networks. The idea is to assign keys to all content items and to distribute the ownership of the key space among all the nodes in the network. According to its key, each content item is copied to the node that is responsible for the key. Hence, content lookup by key is efficient. The shortcoming is that content needs to be relocated in order to be found. The second category of approaches relies on the unstructured overlay network that nodes organize into when joining the system, and does not relocate the content provided. Instead, the user-generated queries and answers that are sent across the network are observed and this information is stored in local indexes. The predictions about which node is capable of answering a certain query are based on functions that rank the information in the indexes according to their relevance to the query in order to select the node that is most likely to contain content that satisfies the query. The ranking functions are based on a stochastic models [7], social metaphors [8], or on assumptions about the content distribution in the network [9]. Our approach falls within this category as well. It is a stochastic model based on the ant metaphor, and hence a multi-agent system. The characteristics of multi-agent systems are that each agent has incomplete information or capabilities for solving the problem, that there is no system global control, that data are decentralized and that computation is asynchronous [10]. Koubarakis points out that the basic concepts underlying multi-agent systems and peer-to-peer networks are very similar [11], and that peer-to-peer networks can provide the infrastructure for deploying multi-agent systems. Babaoglu et al. [12] argue for the applicability of certain biological processes to distributed environments, particularly to unstructured overlay networks.

The project most closely related to our work is Anthill [13], a Java-based open source framework for the design, implementation, and evaluation of ant algorithms in peer-to-peer networks. An Anthill system is an overlay network of interconnected nests (peers). The Anthill API provides a basic set of actions for ants that enables them to travel from nest to nest, and to interact with the services provided by nests. The ant algorithm is not specified by Anthill, but must be designed by the user of the framework according to the application scenario.

IV. Description of the SemAnt Algorithm

In this section, the proposed ant-based algorithm *SemAnt* is described. Section IV-A presents its application scenario, and Section IV-B gives a high-level overview of the algorithm. In Section IV-C, we describe the data that needs to be stored at each node. In Section IV-D, we justify the design rationale of the algorithm's routing strategy. In Section IV-E, we provide a step-by-step description of the algorithm. Finally, Section IV-F illustrates the evaporation rule used.

A. Application scenario

The application scenario for the algorithm is that of a distributed search engine where each node (1) manages a repository of documents and (2) offers the content of its repository to other nodes. In addition to offering documents, nodes pose queries to the network. The system provides for keyword-based search based on meta-data [14].

As described in Section II, foraging ants find the shortest path to one kind of resource, and all pheromone refer to that kind of resource. In our application scenario, each keyword refers to a different kind of resource. Consequently, the network must be independently optimized for all possible keywords. This goal can be reached by employing multiple pheromone types as introduced by Sim et al. [15]. Each keyword is represented by a corresponding type of pheromone. To restrict the maximum number of pheromone types, we constrain the document meta-data vocabulary to a controlled vocabulary, e.g, the concepts of a taxonomy or an ontology. Each document can be an instance of one or more concepts. A query can consist of one or more keywords $c_1, ..., c_n$. As described above, the set of allowed keywords is limited to the concepts of the controlled vocabulary. Multiple keywords are connected using the boolean operator OR. A document is an appropriate result for a given query Q if it is classified to be an instance of one of the concepts that are keywords of the query Q.

B. Overview of the algorithm

For each query, the shortest path through the network must be found that leads from the querying node to a destination node offering information matching the query. The more often a query is requested, the stronger its paths should be optimized. Thus, it is reasonable to represent queries as ants. This guarantees that the degree of optimization for a certain query directly depends on its popularity. If a query is common, its pheromone trails will converge and lead to the nodes that offer the most results for the given query.

We adopt the AntNet strategy for supporting distributed problems by forward ants and backward ants as well as AntNet's strategy for preventing cycles. In AntNet, forward ants terminate their travel through the network when they have reached their well-defined destination node. This behavior can not be transferred to *SemAnt*, since in our scenario the forward ant's task is to find an unknown destination node that in the worst case does not even exist. To specify the point of time at which a forward ant terminates its travel, we define a timeout parameter T_{max} that is the maximum lifetime of a forward ant.

C. Data structures

At each node N_i, pheromone trails are maintained in a table τ of size $C \times n$, where C is the size of the controlled vocabulary and n is the number of node N_i's outgoing links to neighbor nodes. Each τ_{cu} stores the amount of pheromone type c dropped at the link from node N_i to node N_{n_u}, for each concept c and each neighbor node N_{n_u}. At startup, all table entries are initialized with the same small value $\tau_{init} = 0.009$. In addition, each node manages a table η that stores link costs to neighbor nodes N_{n_u}. Each parameter η_u is the inverse value $\frac{1}{lc_u}$ of the cost lc_u for sending an ant from node N_i to node N_{n_u}.

D. Routing strategy

The algorithm's routing strategy is defined in the transition rule. After a thorough evaluation of existing approaches, we decided against using AntNet's transition rule, since it aims to optimize the path to one specific destination node only. We also decided against AntHocNet's [3] broadcasting strategy, since broadcast consumes an excessive amount of network resources. Instead, we select *Ant Colony System*'s transition rule [16] and adapt it to our application scenario as described below.

The Ant Colony System transition rule consists of two strategies that supplement each other. Based on probability w_e, each ant decides whether it applies an exploiting or an exploring strategy. In the *exploiting strategy*, the ant determines the quality of the links depending on the amounts of pheromone and the link costs and always selects the link with the highest quality. The *exploring strategy* encourages ants to discover new paths. This is achieved by deriving a goodness value p_{n_u} for each neighbor node N_{n_u} not already visited, and by applying the roulette wheel selection technique [17] to select one of the nodes. Both strategies are described in Section IV-E.

We utilize an adaptation of the roulette wheel selection technique: Each p_{n_u} is *separately* placed on the continuum between 0 and 1, and for each p_{n_u} a random value q is calculated for deciding whether N_{n_u} should be selected. This strategy allows more than one node to be selected to account for the fact that there are multiple possible destination nodes which contain answers for a query. To ensure that at least one node will be selected, we fall back to the

exploiting strategy in case the exploring strategy does not to select any node.

E. Step-by-step description of the algorithm

A step-by-step description of the algorithm follows. Consider a query Q containing a keyword c issued at a node N^Q. The following seven steps are necessary for answering query Q:

Step 1 Check the document repository of node N^Q. If any results are found, present them to the user. If the number of documents found is less than D_{max}, go to step 2. If the number of documents found is greater than D_{max}, terminate the algorithm.

Step 2 At node N^Q, create a forward ant F^Q with starting time T_{Fstart} and timeout T_{max} responsible for retrieving results for query Q. Add node N^Q to forward ant F^Q's stack of already visited nodes $S(F^Q)$, and initialize the list $LC(F^Q)$ that stores the link costs of all links used by forward ant F^Q.

Step 3 Apply the transition rule in order to decide which outgoing link(s) the forward ant F^Q should choose. As described in Section IV-D, the decision about which strategy to used is based on probability w_e. In case the exploiting strategy is used, the forward ant F^Q applies the transition rule shown in (1) to select the best neighbor node N_j.

$$j = arg\ max_{u \in U \wedge u \notin S(F^Q)} \left([\tau_{cu}] \cdot [\eta_u]^\beta \right), \tag{1}$$

where β is a constant, U is the set of neighbor nodes of N_i, and $S(F^Q)$ is the set of nodes already visited.

In case the forward ant F^Q utilizes the exploring strategy, the transition rule shown in (2) and (3) is applied for each neighbor node N_j in order to decide whether node N_j should be selected.

$$p_j = \frac{[\tau_{cj}] \cdot [\eta_j]^\beta}{\sum_{u \in U \wedge u \notin S(F^Q)} \left([\tau_{cu}] \cdot [\eta_u]^\beta \right)} \tag{2}$$

and

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

where q is a random value, $q \in [0, 1]$, and

$$\sum_{j \in U \wedge j \notin S(F^Q)} p_j = 1 \tag{4}$$

In (3), if $GO_j = 1$, the forward ant F^Q creates a clone F_c^Q of itself and sends the clone F_c^Q to node N_j.

Step 4 Upon arrival at node N_j, check if this node was already visited by a clone F_c^Q. If so, terminate forward ant F^Q. Otherwise, check the document repository of node N_j for documents d that are results for query Q and add the identifiers of all matching documents to the set D. If there are no results, continue at step 6.

Step 5 If $D \neq \{\}$, generate a backward ant B^Q and pass it D and N^D, which is the node that stores D. In order to find its way back to the querying node N^Q, the backward ant B^Q needs a copy the stack data recorded by forward ant F^Q. This stack contains all visited nodes $S(F^Q)$ and all recorded link costs $LC(F^Q)$. In the first step, the backward ant B^Q calculates the sum of all entries in $LC(F^Q)$ to get the total link costs T_D for the path from node N^Q to node N^D. After that, the backward ant B^Q travels back hop-by-hop according to the information stored in $S(F^Q)$ until it arrives at querying node N^Q.

At each intermediate node, backward ant B^Q is responsible for two different tasks. First, it updates the link cost η_j to the entry in $LC(F^Q)$. Second, it drops pheromone by applying the pheromone trail update rule. The pheromone trail update rule is adopted from [16] and defined as given in (5) and (6).

$$\tau_{cj} \leftarrow \tau_{cj} + Z, \tag{5}$$

where

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

In (6), the amount Z of pheromone depends on the quality of the solution, that is, the number of documents found and the total link costs. w_d weights the influence of document quantities and link costs. Amount Z is derived by comparing the goodness of the found solution to an optimal one. For the optimal solution, we set the optimal value for a path's total link costs to $\frac{1}{2} \cdot T_{max}$, and the optimal number of documents to D_{max}.

Step 6 Add node N_i to forward ant F^Q's stack of already visited nodes $S(F^Q)$ and add the cost of the last used link to list $LC(F^Q)$.

Step 7 If $T_{Fstart} + T_{max} < CurrentTime$, continue at step 3. Otherwise, if forward ant F^Q reached its maximum lifetime T_{max}, terminate it.

Note that unlike in AntNet, a forward ant is not terminated directly after it created a backward ant. Instead, it continues its travel until the maximum lifetime is reached, Hence, a single forward ant can generate multiple backward ants. Since the maximum lifetime of a forward ant is T_{max}, all backward ants will arrive within a time interval of $2 * T_{max}$. As soon as a backward ant arrives at the querying node N^Q, the result documents D are presented to the user. If the user decides to download a document d, a direct connection between node N^Q and node N^D that stores d is established and the document is retrieved from node N^D.

F. Evaporation

Each node applies the evaporation rule shown in (7) in a predefined interval t_e for each link to neighbor node N_{n_u} and each concept c, where the amount of evaporating pheromone is controlled by parameter $\rho \in [0, 1]$.

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

This rule is adopted from [16] without any modifications. Note that we do not apply a local pheromone trail update rule as proposed in [16], since not only the ant that

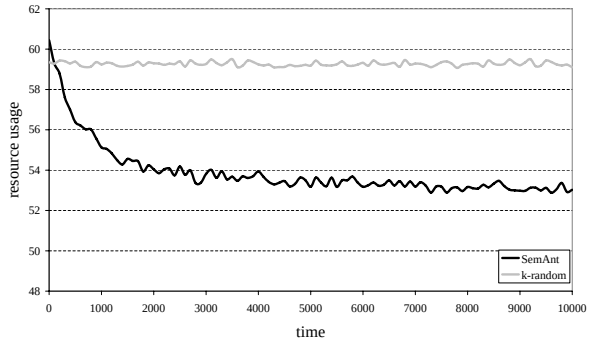

Fig. 1. Resource usage (y-axis) over time (x-axis). k-random walker with k=2 and TTL=25, *SemAnt* with settings shown in Table I

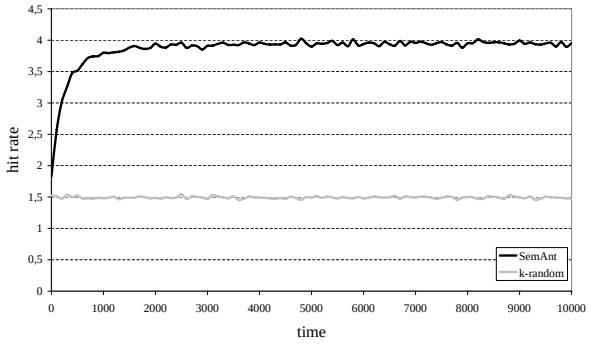

Fig. 2. Hit rate (y-axis) over time (x-axis). k-random walker with k=2 and TTL=25, *SemAnt* with settings shown in Table I

found the best solution of an iteration but all backward ants spread pheromone.

V. Simulation and Results

We evaluate the design of the *SemAnt* algorithm by simulating a peer-to-peer system with real-world settings. The system aims to support cooperation between researchers in computer science by allowing to share documents.

The most important characteristics of a peer-to-peer system are the network topology used, the distribution of content, and the distribution of queries within the network. The underlying network topology has significant effects on the performance of the search algorithm. Since we deal with a community of researchers, we select a *small world* [18] network topology for the experiment in order to choose a realistic model for social networks. The size of the network is set to 1024 nodes, and the clustering coefficient is set to 2. The link costs for all links are set to 1. The content is modeled by utilizing the ACM Computing Classification Systemas the underlying meta-data vocabulary. Each document is annotated with one keyword, i.e., each document is an instance of one leaf concept from the taxonomy. We do not assign documents to nodes randomly, but employ a more realistic model similar to [8]. In this model, each node is an expert on a certain research area and therefore on average 60% of the documents in his or her repository are instances of one particular research area, and on average 20% of the documents are related to another research area. The remaining documents are instances of random leaf concepts. For modeling a research area we facilitate the 177 third-level concepts of the taxonomy, together with their sub-concepts. For the distribution of the concepts within the documents we use uniform distribution in order to represent each research area equally. The distribution of documents among the nodes follows a bell curve with a mean of 29.6 and a standard deviation of 12.15. In this setting, we assume that the network topology and the document distribution do not change during the simulation. For uniform distribution of queries within the network, we use a ticker clock at each node. The probability that a node issues a query within one time unit is set to 0.1. A query consists of a randomly selected leaf concept.

TABLE I
Parameters of *SemAnt*

ρ	evaporation factor	0.07
T_{max}	timeout of forward ants	25
w_e	weight of exploiting vs. exploring strategy	0.85
D_{max}	maximum number of documents	10
w_d	weight of document quantity vs. link costs	0.5
β	weight of link costs	1

For evaluation, we compare *SemAnt* against the well-known k-random walker [19] algorithm described below. The following metrics are used for evaluation. *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. Initial experiments were performed to analyze the performance of *SemAnt* with various parameter settings. These experiments showed that resource usage is highly dependent on the weight w_e. The more ants employ the exploring strategy, the more traffic occurs in the network, not necessarily leading to proportionally better hit rates. The best trade-off between resource usage and hit rate is given when setting parameter w_e to 0.85. The parameter values chosen are shown in Table I. In order to provide for comparison fairness, (1) both algorithms use the same setup as described in Section V, (2) the time-to-live parameters are set to an equal value, and (3) 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 $TTL = 25$. This means that two walkers travel for 25 hops. At each node, they make a random decision about which node to choose next. If documents are found, the walker sends an information message back to the querying node, similar to a backward ant, and walks on. After 25 hops, the walkers are terminated.

We let the experiment run for 10000 time units. The results of the resource usage comparison between *SemAnt*

and k-random walker are shown in Fig. 1. Note that these numbers include both forward and backward ants/agents. The hit rate measurements are shown in Fig. 2. What can be seen from these figures is that the performance of the k-random walker algorithm stays constant. The reason for that is that k-random walker does not include any kind of optimization. On average, the agents travel 59.27 links for answering a query. The average number of result documents found for a query is 1.49 documents. On the contrary, *SemAnt* steadily increases its performance by storing information about queries and results in the past in pheromone trails and by exploiting this information for the routing decisions. In the starting phase of the experiment, the agents have to travel 60.44 links in order to find 1.82 documents per query. After 1000 time units, these numbers improve to 55.13 links traveled for retrieving 3.8 documents per query. The algorithm converges quite fast and reaches its optimal performance after 2000 time units. The average hit rate goes up to 3.95 documents per query, and resource usage decreases to 54.04 links traveled. Between time unit 2000 and time unit 10000, the results are robust. The number of documents per query stays nearly constant. Resource usage decreases slightly. After 10000 time units, the agents need to travel 53.02 links to find 3.95 documents per query.

VI. Conclusion

In this paper we proposed *SemAnt*, a novel algorithm based on the Ant Colony Optimization meta-heuristic. The algorithm was specifically designed for the task of query routing in distributed environments, such as peer-to-peer networks. We conducted an exhaustive study of the constituents of ant-based methods to identify the ones that are best qualified for deployment in peer-to-peer networks. Those selected were then combined and customized to fit our purposes. We conducted an experimental evaluation of the proposed algorithm. Analyzing the results gained, we can draw the conclusion that employing pheromone trails for optimizing query routes is a meaningful approach: *SemAnt* outperforms k-random walker both in terms of resource usage and hit rate. In addition, the algorithm converges fast and shows robust results. However, more work remains to be done, since our investigations were based on the assumption that the network does not change over time. Currently, we are working on an extension of the algorithm that adds support for the inherent dynamics of peer-to-peer networks.

We strongly believe that emergent and self-organizing phenomena observed from nature are of great benefit for the fields of artificial intelligence and computer science. Most of these natural phenomena are not completely understood by now. Recently, biologists found empirical evidence that ants do not only rely on pheromone for indicating the shortest way to a food source, but also use a special kind of stop-pheromone [20] to mark unrewarding paths. We reckon that integrating stop-pheromone in ant-based routing methods can improve performance.

References

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

[2] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz, "Inspiration for Optimization from Social Insect Behaviour," *Nature*, vol. 406, pp. 39–42, July 2000.

[3] 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.

[4] 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.

[5] Stephanos Androutsellis-Theotokis and Diomidis Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, December 2004.

[6] John Risson and Tim Moors, "Survey of research towards robust peer-to-peer networks: Search methods," Tech. Rep., University of New South Wales, September 2004.

[7] Brian F. Cooper, "Guiding Queries to Information Sources with InfoBeacons," in *Middleware 2004, ACM/IFIP/USENIX International Middleware Conference*. October 2004, vol. 3231 of *Lecture Notes in Computer Science*, pp. 59–78, Springer.

[8] Christoph Tempich, Steffen Staab, and Adrian Wranik, "REMINDIN': Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors," in *Proceedings of the 13nd International World Wide Web Conference (WWW2004)*, May 2004.

[9] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," in *Proceedings of IEEE INFOCOM 2003*, April 2003.

[10] Katia P. Sycara, "Multiagent systems," *AI Magazine*, vol. 19, no. 2, pp. 79–92, 1998.

[11] Manolis Koubarakis, "Multi-Agent Systems and P2P Computing: Methods, Systems and Challenges (Invited talk)," in *Proceedings of the 7th International Workshop on Cooperative Information Agents (CIA 2003)*. August 2003, vol. 2782 of *Lecture Notes in Artificial Intelligence*, pp. 46–61, Springer.

[12] 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.

[13] 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.

[14] 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.

[15] Kwang Mong Sim and Weng Hong Sun, "Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 5, pp. 560–572, 2003.

[16] 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.

[17] 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.

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

[19] 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.

[20] Elva J. H. Robinson, Duncan E. Jackson, Mike Holcombe, and Francis L. W. Ratnieks, "Insect communication: 'no entry' signal in ant foraging," *Nature*, vol. 438, pp. 442, November 2005.