Ant Algorithms for Search in Unstructured Peer-to-Peer Networks*

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

Abstract

Although the ant metaphor has been successfully applied to routing of data packets both in wireless and fixed networks, little is yet known about its applicability to the task of query routing in peer-to-peer environments. This work presents SemAnt, an algorithm for distributed query routing based on the Ant Colony Optimization meta-heuristic. The experimental results show that the algorithm produces robust results and converges fast. Based on the results gained so far, the goal for the Ph.D. thesis is to extend the algorithm to include strategies for self-adaptation to volatile networks where nodes may leave or join at any time.

1 Introduction

Ant algorithms are inspired by the collective foraging behavior of specific ant species which use a chemical substance called pheromone for indirect communication. Summarized by a meta-heuristic called *Ant Colony Optimization* [8], several algorithms exist that model and exploit this behavior for solving graph-based NP-hard combinatorial optimization problems, e.g., the traveling salesman problem. Similarities between the self-organizing behavior of ant colonies and self-organization in peer-to-peer networks have already been observed in [3]. Since they do not require any global knowledge about the network, ant algorithms are qualified for peer-to-peer networks.

The research question to be answered in this thesis is: *"To which extent are ant algorithms feasible for the task of content-based query routing in peer-to-peer networks?"* In particular, one aspect that did not yet receive a lot of attention in the research community is that of the inherent dynamics of peer-to-peer networks: changes in the content repositories occur frequently, and peers can join or leave at any time (churn). The goal of the thesis is to deliver an algorithm that includes strategies to cope with these dynamics.

The paper is organized as follows: Section 1.1 introduces ant-based methods, with a focus on those for routing. Related work is described in Section 1.2. Section 2 specifies *SemAnt*, the proposed ant algorithm, and Section 3 shows experimental results indicating its performance. Section 4 presents plans for future work.

1.1 Ant-based algorithms for routing

In ant algorithms, after initializing each edge of the problem graph with a 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 decides, based on the specific edge's amount of pheromone and cost, which one of all possible next nodes to choose. 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. Additionally, some pheromone evaporates in each iteration according to an evaporation factor.

A dedicated subset of ant-based algorithms was specifically designed for managing routing tables in IP networks. The most prominent variant of ant algorithms for routing is *AntNet* [5]. 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

*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.

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 by applying a transition rule at each node in order to decide – based on link costs and pheromone amounts – which outgoing link to follow. When a forward ant F_{sd} launched at a source node N_s has reached its destination node N_d and calculates the total costs of the solution, it cannot update the pheromone trails directly. Since each node stores routing information locally, 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. An evaporation feature is not included in the algorithm.

1.2 Related work

Since ant-based methods are not applicable to structured networks, the focus of the literature review lies on unstructured networks. It includes previous attempts to apply ant-based methods to query routing in peer-to-peer networks.

Many approaches to query routing in unstructured peer-to-peer networks [1] exist. The simplest technique is broadcasting, where a query is recursively forwarded to all reachable neighbor nodes. Since broadcasting consumes a high amount of resources, Yang and Garcia-Molina [20] were among the first to suggest that a peer should rely on statistical information about its neighbors – such as the number of results received in the past – together with simple heuristics to make an intelligent guess which one is the best to send a query to. Their approach considers only the first hop of each query, and uses broadcast for the other hops. Other methods consider all hops and try to predict based on a query's keywords which node is capable of answering it, again by exploiting locally indexed information about user queries in the past. Joseph and Hoshiai coin the term *reputation learning* for these techniques [12]. The basic premise behind reputation learning, which was first published by Cohen et al. [6], 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, namely *interest-based locality* [18] is that peers which share a certain resource are more likely to be able to answer each other's queries because they have at least one common interest. For peer selection, functions that rank the information in the index according to its relevance to the query are used in order to select the node that is most likely to contain content that satisfies the query. The ranking functions are based on probabilistic models like in *Infobeacons* [7], social metaphors like in *REMINDIN'* [19], or on information retrieval techniques like in [13].

The most comprehensive source of information about the applicability of biological processes to distributed environments is provided by Babaoglu et al. in [3], 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. *Anthill* [4] is a Java-based open source 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 not on the algorithm itself.

2 Description of the SemAnt algorithm

The application scenario for the algorithm is that of a distributed search engine where each peer (1) manages a repository of documents and offers its content to the other peers, and (2) each peer performs keyword-based searches based on meta-data [12]. The network must be independently optimized for each possible keyword by employing multiple pheromone types [17]. To restrict the maximum number of pheromone types, the document meta-data vocabulary is constrained 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 Q can consist of one or more keywords $c_1, ..., c_n$. Multiple keywords are connected using the boolean operator `OR`. A document is an appropriate result for a given query Q if it is an instance of one of the concepts $c_1, ..., c_n$.

At each peer P_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 peer P_i's outgoing links to neighbor peers P_u where $u \in \{1, ..., n\}$. Each τ_{cu} stores the amount of pheromone type c dropped at the link from peer P_i to peer P_u, for each concept c and each neighbor peer P_u. At startup, all table entries are initialized with the same small value τ_{init}.

SemAnt adopts the *AntNet* strategy of forward ants and backward ants for supporting distributed problems, and 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. A time-to-live (TTL) parameter T_{max} is used to prevent forward ants from running infinitely. If a forward ant arrives at a peer that stores documents satisfying the query, it creates a backward ant. 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. After the first backward ant is created, there are two possibilities: Either the forward ant terminates its travel, or it continues it until the maximum TTL is reached. The idea behind the latter approach is that if forward ants are allowed to go on after they found the first result node, they can increase the absolute number of documents found by detecting other result nodes and generating multiple backward ants. For describing the implications of these two possibilities, we introduce the following metrics: *Resource usage* is defined as the number of links traveled for each query, and *hit rate* is defined as the number of documents found for each query. If the ants are terminated after they found the first result, the resource usage is minimized, but the hit rate increases only slightly. In the other case, where the ants use the maximum TTL, the hit rate is maximized, but the resource usage decreases only slightly. Our experiments show that terminating the forward ant after it has found the first result leads to a lower number of hops traveled for retrieving one document and thus increases the overall performance of the network. The reason for that is that if the the ants go on, they tend to stay in the neighborhood of the node they already found, since the pheromone trails indicate that there is an appropriate node nearby. Nevertheless, it is also reasonable to let the ants use the maximum TTL, because in this case the number of documents found for a single query is higher. This impact is of benefit for the individual users of the system.

As described above, *AntNet* natively accounts for different links costs in terms of a link's bandwidth and latency. This feature could be easily transferred to *SemAnt*. However, the difficulty lies in evaluating its usefulness. The yet unsolved problems are (1) to find adequate network test data that includes the cost of all links and (2) to find an appropriate method for comparison, since we are not aware of any other approach that takes link costs into account.

The ant algorithm's routing strategy is defined by its transition rule. An adapted version of *Ant Colony System*'s transition rule [9] is used. This rule consists of two strategies that complement each other. Based on probability w_s, each forward ant F^Q decides if it applies the *exploiting* or the *exploring* strategy. In the exploiting strategy, the link to the best neighbor peer P_j with the highest quality is selected by applying the following formula:

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

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

The exploring strategy encourages the forward ants to discover new paths. In case F^Q utilizes the exploring strategy, the transition rule shown in (2) and (3) is applied for each neighbor peer P_j. The first step is to derive a goodness value p_j for each neighbor peer P_j not already visited. In the second step, an adapted version of the roulette wheel selection is applied technique [10] 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. To ensure that at least one peer will be selected, *SemAnt* falls back to the exploiting strategy in case the exploring strategy does not to select any peer.

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

and

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

where q is a random value, $q \in [0,1]$, and $\sum p_j = 1$. In case of $GOTO_j = 1$, F^Q sends a clone of itself to peer P_j.

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 $S(F^Q)$. 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. At each intermediate peer, the backward ant drops pheromone on the link previously selected by the forward ant. The pheromone update rule is adopted from [9] and defined as shown in (4) and (5). 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. As shown in (5), 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.

$$\tau_{cj} \leftarrow \tau_{cj} + Z, \tag{4}$$

where

$$Z = w_d \cdot \frac{|D|}{D_{opt}} + (1 - w_d) \cdot \frac{T_{max}}{2 \cdot T_D} \tag{5}$$

Each peer applies the evaporation rule shown in (6) in a predefined interval t_e for each link to neighbor peer P_u and each concept c, where the amount of evaporating pheromone is controlled by parameter $\rho \in [0,1]$.

$$\tau_{cu} \leftarrow (1 - \rho) \cdot \tau_{cu} \tag{6}$$

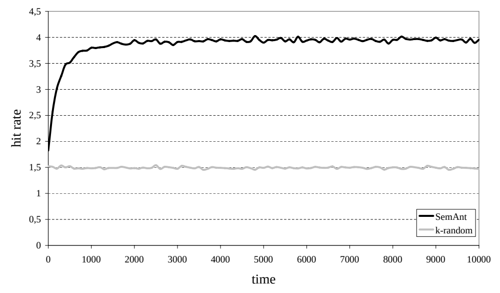

Figure 1. Hit rate

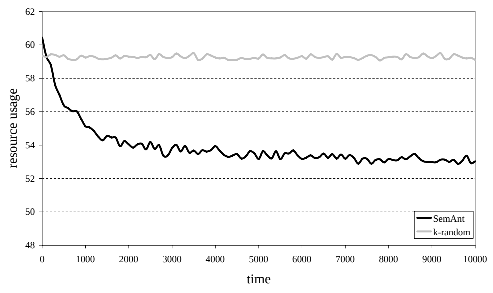

Figure 2. Resource usage

3 Simulation and results

SemAnt is evaluated by simulating a peer-to-peer system with 1024 peers and by comparing its performance against that of the well-known k-random walker [15] approach. In this experiment, 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 [14] with a clustering coefficient of 2 is used. Content is modeled with the ACM Computing Classification System [2] 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. This is a rather small setup, but sufficient for our purposes. For the document distribution, a model similar to [19] is used. Each peer is an expert on a certain topic and therefore 60% of its documents are instances of one particular research area. 20% of its documents are related to another research area. The remaining documents are instances of random leaf concepts. Research areas are modeled by the third-level concepts of the taxonomy. The distribution of documents among the peers follows a bell curve with a mean of 29.6 and a standard deviation of 12.15. 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. A query consists of a randomly selected leaf concept.

The optimal values for the configurable parameters of *SemAnt* are mostly dependent on the network topology used and on the content distribution within the network. A first study of the impact of different content distributions on the overall performance of the system can be found in [16]. It is planned to investigate this issue in more detail with the goal of finding a method for determining the optimal parameter settings at runtime. The experiments conducted so far revealed that resource usage is highly dependent on weight w_s. The more ants employ the exploring strategy, the more traffic occurs in the network, but not necessarily leading to proportionally better hit rates. In the experiment described here, the best trade-off between resource usage and hit rate can be observed when setting parameter w_s = 0.85. The other parameter values were chosen as follows: ρ = 0.07, T_{max} = 25, D_{max} = 10, w_d = 0.5. To provide for comparison fairness, the time-to-live parameter of k-random walker is also set to 25, and k is set to 2.

The metrics used for evaluation are *resource usage* defined as the number of links traveled for each query, and *hit rate* defined as the number of documents found for each query. In this experiment, the forward ants use the maximum TTL, thus maximizing hit rate (see Fig. 1). The results of the resource usage comparison between *SemAnt* and k-random walker are shown in Fig. 2. These numbers include both forward and backward ants/agents. The performance of k-random walker stays constant with on avg. 59.27 links traveled by the agents for finding on avg. 1.49 result documents. On the contrary, *SemAnt* increases its performance. The algorithm converges quite fast and reaches its optimal performance after 2000 time units. The hit rate increases to 3.95 documents per query, and resource usage decreases to 54.04 links traveled. From time unit 2000 to 10000, the hit rate stays nearly constant while the resource usage decreases slightly and declines to 53.02 links per query at time unit 10000.

4 Future work

In the experimental evaluation described above, a static network topology and document distribution was assumed. In the next step, we want to evaluate and improve the performance of the algorithm in a dynamic setting in which new peers are joining the network, peers are leaving, and peers add or remove documents from their repositories. In these cases, it is necessary to adapt the pheromone trails in order to reflect these changes. To correct the amount of pheromone, the η-strategy proposed by Guntsch and Middendorf [11] will be investigated. According to this strategy, the closer a link is to a joining or leaving node, the higher the amount of pheromone which is removed from it. This increases the influence of new links, and thus encourages the ants to use the new paths. In the original η-strategy, the measurement of closeness is based on the link costs. We will test a simplified version and measure closeness in number of hops. Next to the η-strategy, other possibilities for bootstrapping the pheromone trails of newly joined peers are either to copy pheromone information from neighbor peers, or to create artificial queries at the newly joined peer.

5 Conclusion

This paper proposed *SemAnt*, a novel algorithm based on the Ant Colony Optimization meta-heuristic designed for the task of query routing in peer-to-peer networks. In *SemAnt*, ants cooperate in creating pheromone trails – probabilistic overlay networks – that indicate the most promising path for a given query. The more popular a query, the better its trail is optimized. Our first experimental results show that the algorithm converges fast, and that its performance is acceptable with the great benefit of being stable and robust.

The vision for the Ph.D. thesis is to build a self-repairing network that can adjust itself to network churn. There are several reasons to be confident that our approach is suitable for the inherent dynamics of peer-to-peer network. First, the evaporation feature is built-in by nature and automatically helps to remove outdated trails. Second, since content is not replicated or relocated in our approach, it is comparatively easier to design a strategy for coping with churn. Third and finally, the pheromone trails build a global map of the content distribution in the network which can be adapted according to changes in network topology and content distribution.

Acknowledgements

I would like to thank the anonymous reviewers for their valuable comments and I am grateful for the opportunity to discuss and get feedback at the ICDE Ph.D. Workshop.

References

[1] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] B. F. Cooper. Guiding Queries to Information Sources with InfoBeacons. In *Middleware 2004, ACM/IFIP/USENIX International Middleware Conference*, volume 3231 of *LNCS*, pages 59–78. Springer, 2004.

[8] M. Dorigo and G. D. Caro. *New Ideas in Optimization*, chapter The Ant Colony Optimization Meta-Heuristic, pages 11–32. McGraw-Hill, 1999.

[9] 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.

[10] 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.

[11] M. Guntsch and M. Middendorf. Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. In *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, pages 213–222. Springer, 2001.

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

[13] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*, pages 300–307, November 2002.

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

[15] 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.

[16] E. Michlmayr, A. Pany, and G. Kappel. Using Taxonomies for Content-based Routing with Ants. Technical report, Vienna University of Technology, February 2006.

[17] K. M. Sim and W. H. 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*, 33(5):560–572, 2003.

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

[19] 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.

[20] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, July 2002.