Ph.D. Thesis

Ant Algorithms for Self-Organization in Social Networks

Conducted for the purpose of receiving the academic title
'Doktorin der technischen Wissenschaften'

Advisors

Prof. Gerti Kappel
Institute for Software Technology and Interactive Systems/E188
Faculty of Informatics
Vienna University of Technology

Prof. Wolfgang Nejdl
Distributed Systems Institute, Knowledge Based Systems
University of Hannover

Submitted to the
Vienna University of Technology
Faculty of Informatics

by

Elke Michlmayr
9402411
Schikanedergasse 6/6
1040 Wien

Vienna, 14th May 2007 _____

Abstract

Peer-to-peer networks and folksonomies are like living organisms, ever growing and changing as time goes on. This thesis addresses the applicability of algorithms derived from the self-organizing and emergent behavior observed from ant colonies to these complex networks for two specific purposes, namely (1) content-based search in unstructured peer-to-peer networks, and (2) the extraction of adaptive user profiles from folksonomies.

For search in unstructured peer-to-peer networks, the main goal is to find the shortest path from every querying peer to one or more answering peers that possess resources which are appropriate answers for the given query. The SEMANT algorithm, which is designed for this task, is based on *reputation learning*. In reputation learning, the information about the remote peers' resources is gained passively by observing the user queries and their answers that are forwarded through the local peer. Every successful query evokes small updates in the routing tables of those peers that are included in the path between the querying and the answering peer. The routing tables are used for subsequent queries to decide which link to follow in order to find appropriate resources. The SEMANT algorithm is compliant with the *Ant Colony Optimization* meta-heuristic, and it employs a probabilistic procedure to select outgoing links for query forwarding. This procedure combines an exploiting strategy, which selects those links currently known as the most appropriate ones, with an exploring strategy, which also follows links not currently known as the best ones with the aim of finding better paths not yet explored. A weight defines the ratio between the strategies.

Since the SEMANT algorithm is a content-based approach to peer-to-peer search, its performance depends on how the content is distributed in the network. The evaluation of the algorithm includes an investigation to which extent this is the case. Based on these results, we develop strategies for improvement. Under the assumption that the resources in the network are annotated according to a taxonomy, and that the query vocabulary is restricted to the leaf topics from the same taxonomy, it is possible to consider also the upper-level topics in the query routing procedure of the algorithm in order to increase its performance.

Ant algorithms include an *evaporation feature* for integrating a time factor when incrementally creating solutions. This feature is beneficial for the task of learning adaptive user profiles from tagging data. For this purpose we design the Add-A-Tag algorithm, which is based on a combination of an evaporation feature for adapting the user profile to trends over time, and the *co-occurrence technique* for determining the relationships between tags. The user profiles created with the Add-A-Tag algorithm are semantic networks derived from the structure of the tagging data, and they are adaptive in the sense that they change according to changes in a user's tagging behavior. In addition to the long-term interests of a user, also his or her short-term interests are included in the profile at any given point of time. We present a tool for visualizing the changes in the profile over time, and we show how to exploit the profile for personalized browsing of annotated data sources.

Kurzfassung

Peer-to-Peer Netzwerke und Folksonomies ähneln lebendigen Organismen, die stetig wachsen und sich kontinuierlich ändern. Ameisenalgorithmen sind dem emergenten und selbstorganisierenden Verhalten von Ameisenkolonien bei der Futtersuche nachempfunden. Die vorliegende Dissertation beschäftigt sich mit der Anwendbarkeit von Ameisenalgorithmen in diesen Netzwerken für (1) die semantische Suche in unstrukturierten Peer-to-Peer Netzwerken, und für (2) die Extrahierung von adaptiven Benutzungsprofilen von Folksonomies.

Das Hauptproblem bei der Suche in unstrukturierten Peer-to-Peer Netzwerken ist es, für jede Anfrage jene Peers zu finden, die dem anfragenden Peer topologisch nahe liegen und über Ressourcen verfügen, die als Antwort auf die Anfrage geeignet sind. Der im Rahmen dieser Dissertation für diesen Zweck erstellte Ameisenalgorithmus SEMANT verfolgt einen Ansatz, in dem die Informationen über die an den entfernten Peers verfügbaren Ressourcen auf passive Art und Weise, durch das Überwachen der am lokalen Peer empfangenen und weitergeleiteten Anfragen und Antworten, gewonnen werden. Jede erfolgreiche Anfrage hinterlässt eine Spur im Netzwerk. Für das Weiterleiten aktueller Anfragen wird ein probabilistisches Modell benutzt, das die Summe der bestehenden Spuren auswertet. Dieses Modell kombiniert eine verwertende mit einer explorierenden Strategie. In der verwertenden Strategie werden immer jene Spuren benutzt, die am besten geeignet sind. In der explorierenden Strategie werden auch schlechtere Spuren benutzt, um potentiell noch nicht bekannte bessere Wege zu finden und damit die Leistung des Gesamtsystems zu erhöhen.

Die Leistung von semantischen Ansätzen zur Peer-to-Peer Suche wird stark von der Verteilung der Inhalte im Netzwerk beeinflusst. Das gilt auch für den SEMANT Algorithmus. Unter der Annahme, dass alle Ressourcen im Netzwerk mit aus einer Taxonomie stammenden Metadaten ausgezeichnet sind und dass die Anfragen auf demselben Metadatenvokabular beruhen, ist es möglich, die hierarchische Information aus der Taxonomie in die Entscheidung mit einzubeziehen, zu welchem Nachbar-Peer eine Anfrage weitergeleitet werden soll. Damit wird die Leistung des Suchalgorithmus verbessert.

Ameisenalgorithmen inkludieren einen Evaporierungsfaktor zur Berücksichtigung von zeitlichen Aspekten bei der inkrementellen Generierung von Lösungen für Optimierungsprobleme. Der Add-A-Tag Algorithmus für die Extrahierung von adaptiven Benutzungsprofilen von Folksonomies nutzt diesen Evaporierungsfaktor in Kombination mit einem Co-occurrence Ansatz zur Anpassung von Benutzungsprofilen an die Änderungen im Tagging Verhalten eines Benutzers oder einer Benutzerin. Die mit Add-A-Tag erstellten Profile sind semantische Netzwerke, deren gewichtete Kanten die Stärke der Beziehungen zwischen den einzelnen Tags im Profil ausdrücken. Durch die Integration des Evaporierungsfaktors in den Algorithmus beinhalten die Profile sowohl die Langzeitinteressen als auch die Kurzzeitinteressen der BenutzerInnen zum Zeitpunkt der Profilberechnung. Die Profile können für das personalisierte Browsen von annotierten Datenquellen verwendet werden.

Acknowledgements

I am grateful to my advisors Prof. Gerti Kappel and Prof. Wolfgang Nejdl for their support. Moreover, I would like to thank my friends and colleagues who accompanied and helped me throughout my thesis work: Sabine Graf, Birgit Korherr, Marion Murzek, Andrea Schauerhuber, Veronika Stefanov, Nevena Stolba, Martina Umlauft, Stefanie Scherzinger, Doris Kastner, Sonja Willinger, Sonja Schindler, Daniela Knitel, Ulli Pastner, Beate List, Horst Eidenberger, Gerald Reif, Martin Bernauer, Gerhard Kramler, Michael Schadler, Monika Lanzenberger, Martina Seidl, Horst Kargl, Michael Strommer, Manuel Wimmer, George Metcalfe, Nysret Musliu, Agata Ciabattoni, Florian Ledermann, Karin Kosina, Michael Mehling, Wolf Siberski, Alexander Löser, Christoph Tempich, Christoph Schmitz, Christof Marti, Wai Gen Yee, Arne Handt, Daniel Gmach, Miriam Fernandez, Mischa Tuffield, Phil Moore, Tobias Bürger, Aldo Gangemi, Enrico Motta, Michele Pasin, Tom Heath, Steve Cayzer, Paul Shabajee, Carlo Torniai, Hui Wan, Andrew Byde, Paolo Castagna, Huw Jeffries, Daniel Drozdzewski, Serdar Cabuk, Johannes Kirschnick, Ilango Sriram, Andreia Estevez, Filipe Beato, Cressida Darwin, Christoph Trompler, Verena Hladik, Arno Pany, Christian Zech, Christine Kohlmayr, Peter Hochpöchler, Susanne Palmetzhofer, Susanna Hutsteiner, Martina Mayrhofer, Markus Koppenberger, Florian Hammer, and many more. Most of all, I thank my parents Christine and Leopold, my sister Birgit, and my brother Anton.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the motivation for the research conducted in this thesis. Next to providing the necessary background about the problems to be tackled, the contributions of the thesis are summarized and the organization of this document is presented.

1.1 Problem statement

The proliferation of the Internet has drastically enhanced the opportunities to access, publish, and manage information and resources. The *World Wide Web* provides for easily accessing and publishing information. *Peer-to-peer systems* allow users to make the resources stored at their personal computers available to the community. Current efforts – subsumed under the term *Web2.0* [138] – are targeted at making it even easier for end users to publish information, thus emphasizing the social network aspect of the Web, and at providing them with folksonomies as a simple means to annotate information. Since publishing is made so easy, an enormous amount of networked data is available. This raises challenges for the research community: How to support users in discovering what they are looking for?

Many approaches to searching or organising information in a network take a link-based approach. The most prominent example is the PageRank algorithm [22], which iteratively estimates the importance of a Web page by considering the number of links pointing to it in combination with the importance of the referring pages. In peer-to-peer search, the core of every search algorithm for unstructured networks is its method for evaluating a peer's outgoing links with respect to their desirability for forwarding a certain type of query. Folksonomies are tripartite graphs with hyperedges (that is, links which can connect more than two nodes) which allow to derive networks with weighted links that express the strength of the relationship between the nodes. Depending on which network is of interest, the nodes can either refer to tags, or to bookmarks, or to users.

This thesis investigates the use of ant algorithms for assessing the strengths between links in social networks. Ant algorithms mimick the pheromone-based, self-organizing behavior called trail-laying and trail-following that is used by natural ant colonies for finding the

shortest path between a nest and a food source. When moving, each ant drops a chemical substance called pheromone. In order to decide which way to follow, the ants sense their environment for existing pheromone trails and follow them with a probability that is proportional to the strength of the trail. If not reinforced by an ant that uses it, a pheromone trail will evaporate over time. The Ant Colony Optimization meta-heuristic (ACO, [45]) proposed by Di Caro and Dorigo defines a framework for applying trail-laying and trail-following techniques in combination with evaporation of trails to solving graph-based combinatorial optimization problems. Most ACO-compliant algorithms have been applied to centralized problems. However, since ant algorithms are exclusively based on modifications of pheromone trails and therefore need only local knowledge about the graph, they are also suitable for application in distributed environments without centralized control, i.e., in peer-to-peer networks. A dedicated subset of ant-based algorithms, subsumed under the term Ant Colony Routing [27], was specifically designed for managing routing tables in IP networks. A similar approach is used for routing tables in mobile ad-hoc networks [28].

1.2 Contributions

This thesis provides a twofold contribution. In the first part of the thesis, we investigate ant algorithms for content-based search in peer-to-peer networks. The three main contributions of this part of the thesis are the following:

- We propose a novel algorithm for content-based search in peer-to-peer networks compliant with the Ant Colony Optimization meta-heuristic.
- We assess the factors that have an impact on the performance on the algorithm (such as parameter settings, network topology, and content distribution in the network) and conduct a thorough evaluation to quantify their impact.
- We design an extension to the algorithm for taxonomy-based application scenarios that enhances its performance by additionally laying and following pheromone trails for higher-level topics.

In the second part of the thesis, we show how the evaporation factor from ant algorithms can be applied to the task of extracting adaptive user profiles from tagging data. The two main contributions of this part of the thesis are the following:

- We design the Add-A-Tag algorithm for adaptive user profile learning from tagging data, and we evaluate it (1) through statistical analysis and (2) with a small-scale user study, for which we implemented a tool for visualizing the profile changes over time.
- We show how to use the profiles created with Add-A-Tag for personalized browsing of annotated data sources.

1.3 Organization of the thesis

The thesis is organized as follows.

Part I presents the SEMANT algorithm for content-based search in unstructured peer-to-peer networks. This part is comprised of Chapter 2, Chapter 3, Chapter 4, and Chapter 5.

Chapter 2 provides an overview of peer-to-peer networks. It introduces the main characteristics of the peer-to-peer paradigm and the different approaches to structuring the topology of the overlay network. Since the SEMANT algorithm is designed for search within unstructured networks, the major part of the chapter is devoted to a review of related work on unstructured networks.

Chapter 3 introduces the basic principles of emergence and self-organization and gives a short summary of the relevant literature in this research area. The focus of this chapter is on ant-based methods. After giving a general overview of ant algorithms and their properties, three different ant algorithms that are relevant for the SEMANT algorithm are introduced and discussed. In particular, the applicability of the selected algorithms for search in peer-to-peer networks is evaluated. The outcome of this evaluation builds the basis for defining the SEMANT algorithm.

Chapter 4 is based on the findings of Chapter 3 and covers the specification of the proposed ant algorithm SEMANT for content-based search in peer-to-peer networks. After defining the problem and stating the assumptions which were made, what follows is a detailed description of all aspects related to the SEMANT algorithm. The remainder of the chapter is devoted to the evaluation of the algorithm's performance in various scenarios and settings, and to an overview of related work.

Chapter 5 serves two purposes. Firstly, it investigates the impact of different content distributions on the performance of the SEMANT algorithm. Secondly, it shows how taxonomies can be used for SEMANT in order to improve the performance. After providing the motivation for that work, it is shown how to extend the SEMANT algorithm to provide for the consideration of metadata from a taxonomy for routing. After that, the experimental results indicating the performance of the algorithm with respect to different content distributions – with or without using information from a taxonomy for routing – are presented, and related work on using taxonomies for routing in peer-to-peer networks is discussed.

Part II deals with tagging data and presents the Add-A-Tag algorithm for acquisition of dynamic user profiles from folksonomies. It is comprised of Chapter 6 and Chapter 7.

Chapter 6 delivers a case study on the properties of meta-data provided by a folksonomy (also referred to as *tagging data*). After providing the background about folksonomies and discussing to which extend the process of creating meta-data in a folksonomy is related to the idea of emergent semantics as defined by the IFIP 2.6 Working Group on Data Semantics, experiments are conducted in order to analyze the meta-data provided by the *del.icio.us* folksonomy. The aim is (1) to develop a method for selecting subsets of meta-data that adhere to the principle of interest-based locality, which was originally observed in peer-to-peer environments, and to (2) compare the data provided by the *del.icio.us* folksonomy to data provided by the DMOZ taxonomy.

This chapter provides the connection between the two parts of the thesis. Next to being an initial starting point for the more detailed investigation of folksonomies in Chapter 7, the work presented in this chapter was also carried out to determine if tagging data can be used as test data for a peer-to-peer simulation environment, and therefore contains backlinks to the work described in Part I of the thesis.

Chapter 7 This chapter presents the Add-A-Tag algorithm for profile construction from tagging data. Taking account of the structural and temporal nature of tagging data allows to create user profiles which (1) are represented as semantic networks, and (2) include both long-term and short-term interests of a user. In addition, this chapter explores ways of leveraging the user profiles. They can be used to guide a user's navigation, that is, to provide the user with personalized access to information resources.

Chapter 8 concludes and gives an overview of future work.

Part I

Content-based Search in Unstructured Peer-to-Peer Networks

Chapter 2

Search in peer-to-peer networks

This chapter presents an overview of peer-to-peer networks. It introduces the main characteristics of the peer-to-peer paradigm, and the different approaches to structuring the topology of the overlay network. Since the SEMANT algorithm is designed for search in unstructured networks, the major part of the chapter is devoted to a review of related work in this area.

Contents

2.1 Peer-to-peer networks

A peer-to-peer network is "a distributed system in which all nodes are completely equivalent in terms of functionality and tasks they perform" [10]. In contrast to the client/server model, every peer-to-peer node is both a client that consumes resources from other peers, and a server that provides resources to other peers. Another definition of peer-to-peer networks puts the focus on the implications of the fact that there is *no central coordination* in

the network: "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" [10]. This definition also stresses that each peer owns a portion of the resources (such as data or computing power) accessible in the network. Since each peer knows only about the peers it communicates with, it has a local view on the network. Risson and Moors's definition includes the *self-organizing* properties of peer-to-peer networks, that is, they "automatically adapt to the arrival, departure, and failure of nodes" [111].

Peer-to-peer networks are built at the application layer on top of a communication infrastructure at the physical layer, such as the Internet. They can be viewed as graphs in which the nodes are the peers and the edges are the links between these peers. Since (1) only a few of all nodes of the underlying infrastructure participate in the peer-to-peer network and (2) the links between these peers can be composed of a number of physical links, peer-to-peer networks are also referred to as *overlay networks*.

The peers are autonomous and can leave the network at any time. This must not cause any interruption of the network's services. Hence, peer-to-peer architectures must provide for *fault tolerance* by nature, which is one of their main characteristics. Another characteristic that is inherently present is *scalability*. Unlike in centralized systems where upgrading the server can be hard or impossible when the maximum number of users is reached, in a peer-to-peer network it must be considered already in the design phase of the system that any number of peers can participate. Other requirements of peer-to-peer networks include performance, fairness, and security.

For a more detailed general overview, see the work of Hauswirth et al. [68] and Milojicic et al. [94]. There are numerous research issues in the peer-to-peer area, e.g., discovery, content replication, schema mapping, caching, migration, access control, authentication, reputation, and others. One of the key challenges is search in peer-to-peer networks (also referred to as query routing or query processing), which is the focus of this part of the thesis.

2.2 Structured and unstructured networks

There are several possibilities for the structure (topology) of the overlay network, which will be described in the following.

2.2.1 Unstructured overlay networks

The most simple approach is to use the network structure the peers organize themselves into when joining the network. Each peer will connect to a set of other peers when joining the network, which will result in a random graph. Such overlay networks are called *unstructured overlay networks*. Depending on which search technique is used, the peers will either

(1) not manage a content index at all, or (2) manage an index of their own content, or (3) also include information about the content stored at their neighbor peers in their indexes. Each peer can send queries to its neighbors, and the peers collaborate to forward queries on each others behalf. Search techniques for unstructured overlays are reviewed in Section 2.3.

2.2.2 Super-peer architecture

In real-world settings, not all the peers will have the same capabilities in terms of bandwidth and amount of storage space. Instead, some of the peers will have a high amount of resources, whereas most peers will have an average amount of resources at their disposal. Therefore, it makes sense to impose a two-level hierarchy on the peers and to classify those peers with a high amount of resources as super-peers (sometimes also referred to as ultra-peers). The *super-peer architecture* has first been described by Yang and Garcia-Molina [146]. In this architecture, each peer connects to a super-peer when joining the network. Each super-peer (1) is connected to other super-peers and (2) serves a set of peers for which it manages an index of their content, and for which it forwards queries to other super-peers if the query can not be answered considering the content indexed at the super-peer itself. Additionally, the super-peers can organize themselves into a convenient topology that provides short paths for query routing, such as a hypercube as shown by Nejdl et al. [100].

2.2.3 Structured overlay networks

Similar to changing the overlay structure of the super-peer network, it is also possible to organize all the peers into regular topologies, such as rings [97], binary trees [2], multi-dimensional toruses [108], or others. Such overlay networks are called *structured overlay networks* or *distributed hash tables (DHT)* [15]. The regular topology allows for an efficient lookup procedure (proximity routing) for simple exact-match queries, also referred to as key lookup. However, in order to make use of the regular structure it is necessary to relocate the content in the network. This is also referred to as *publishing*:

- All the resources in the network are identified using unique numeric keys.
- According to the regular structure chosen, each peer is responsible for a certain range of the key space.
- To insert a resource into the system, it is necessary (1) to derive its key from the resource name using a hash function and to (2) copy the resource to the peer that is responsible for that key.

Structured overlay networks guarantee that each resource can be found (in the absence of network failures). In addition, they assure an upper bound for the number of hops neces-

sary for key lookup. Typically, the upper bound is $O(log(N))$, where N is the number of peers in the network.

2.2.4 Comparison

There is still a lot of debate within the peer-to-peer research community about which approach – unstructured overlays, super-peer architectures, or structured overlays – has the most benefits. It is generally agreed that key lookup with distributed hash tables is very efficient and can not be beaten by the other approaches. On the other hand, supporting complex queries is easier to implement with unstructured overlays or super-peer architectures (see [44]). However, it is also possible with structured overlays (see [65]). In addition, the fact that structured overlays require content publishing can not be neglected easily, since it (1) creates additional costs and also because it (2) is not possible in all application scenarios, e.g., due to user preferences or intellectual property (IP) issues.

Since most researchers are in favor of one method or the other, only a few direct comparisons between the different approaches can be found in literature. One of them is provided by Yang et al. [147], who compare the performance of full-text search in a structured, a super-peer, and an unstructured network. Their results show that the structured network provides a 30% better response time than the super-peer network, but needs six times as much network resources for publishing the content. The unstructured approach (with search implemented as random walks, see Section 2.3.3) needs no resources for publishing at all, but is much slower in responding to queries compared to the other approaches.

However, some related work on combining structured and unstructured approaches can be found. Castro et al. [29] show how to implement flooding (see Section 2.3) and random walks in a distributed hash table for better support of complex queries. Loo et al. [85] observe that unstructured networks (in their case Gnutella, which uses a simple super-peer architecture) are highly effective for locating popular files, whereas distributed hash tables are not. At the same time, unstructured networks are not effective for locating rare items, whereas distributed hash tables are. Hence, they suggest to use a hybrid infrastructure in which only rare items are published using a distributed hash table. The super-peers need to be organized into a regular structure, and they are responsible for identifying and publishing rare content from their client peers to the distributed hash table. Queries are then performed by using the methods used in unstructured networks. Only if there are not enough results, the query is issued to the distributed hash table.

2.3 Search in unstructured networks

This section provides a chronological overview of the different approaches to query routing in unstructured overlay networks. For a more detailed overview refer to [131, 111, 76, 10].

The idea of a peer-to-peer network as a network where nodes with equal properties collaborate is not a new one. It has already been applied for the *Usenet* [42] in 1979 and for the *FidoNet* [124] in 1984. The term *peer-to-peer* became famous with the advent of the distributed file-sharing application *Napster*. Napster, which was a very popular service between 1999 and 2001, relies on an index of all documents stored at central servers. The peers in the network register their contents at these servers, and use the indexes for querying. The content itself is exchanged using direct connections between the nodes. Peer-to-peer architectures which include centralized components are also referred to as *hybrid peer-to-peer systems*. The centralized component was the main drawback of Napster for two reasons. Firstly, it is a bottleneck because all users rely on the central indexes. Secondly, the owners of the central components can be made responsible for legal issues if the users of the system share copyrighted material. For this reason, Napster was shut down in 2002.

2.3.1 Breadth-first search and depth-first search

Soon after Napster's shutdown, similar file-sharing applications that did not rely on centralized components emerged. Their basic approach was not to use indexes at all, but to broadcast each query to the network with a limited scope. Broadcasting queries is also known as *flooding*. The queries are recursively forwarded to all reachable neighbor nodes. The scope of a broadcast, also referred to as time-to-live parameter (TTL), is defined by the number of hops that lie between the querying node and the most far-off node trying to answer the query. In breadth-first search, a query is recursively broadcast to all neighbors that can be reached within a defined number of hops. Less network resources than in breadth-first search are used in depth-first search, where nodes send a query to all of their neighbors sequentially, and stop after the query is satisfied. Breadth-first search (BFS) and depth-first search (DFS) are techniques known from Graph Theory [80]. Since they consume a high amount of network resource, the drawback of broadcasting techniques is that they provide only limited scalability [112, 90].

2.3.2 Modifications to breadth-first search

The drawbacks of flooding-based techniques with regards to scalability became evident soon, and a newly-formed research community began to work on improvements. Yang and Garcia-Molina [145] suggested *directed breadth-first search*, where the peers rely on statistical information they collect about their neighbors – such as the amount of results received in the past – together with simple heuristics for ranking them according to their performance in answering queries. Subsequent queries are sent to the peers that performed best. The directed breadth-first search approach considers only the first hop of each query, and uses flooding for the other hops. In addition, Yang and Garcia-Molina use dynamically increasing time-to-live parameters (*iterative deepening*). With iterative deepening, each query is

broadcast with a small TTL parameter first. If there is no answer, the TTL is increased. This process repeats until an answer can be found, or the maximum TTL is reached. A similar approach was suggested by Lv et al. [90] under the name *expanding ring*.

Kalogeraki et al. [78] proposed *modified breadth-first search*, where a search message is not forwarded to all neighbors but only to a fraction of them. For example, if the fraction parameter is set to 0.5, a peer will randomly select 50% of its neighbors for forwarding a query.

Menascé and Kanchanapalli's [92] approach also selects a subset of neighbors for forwarding. Instead of selecting a fraction of neighbors to according a certain probability, it employs a *broadcast probability* according to which each neighbor will be chosen. This means that it can select between 0 and n neighbors of a peer P_x, where n is the number of peer P_x's outgoing links.

2.3.3 Random walks

Another technique that does not rely on any indexes are *random walkers* as introduced by Lv et al. [90]. In the random walker approach, a query will be forwarded to a randomly chosen neighbor until an answer is found, or the maximum TTL is reached. Since using only one walker results in long response times for the user, k random walkers are employed to speed up the retrieval process. The authors suggest to use between 16 and 64 walkers for a query. Moreover, several possibilities to enhance random walks are proposed:

- Instead of using a TTL-based termination of the queries, it is also possible to use a technique named *checking*, where a walker periodically checks with the querying peer to find out if enough answers to the query were found already. If not, it continues walking.
- In *random walk with state*, each query has an unique ID. Each peer records the IDs of the queries it forwards, together with the identifier of the neighbor it sent it to. If another query with the same ID arrives, the peer will forward it to another neighbor.

In addition, the authors propose techniques for replicating the content in the network in order to improve recall.

2.3.4 Local indices

In the same paper [145] that presents directed breadth-first search, Yang and Garcia-Molina also propose another technique called *Local Indices*. In this case, each peer stores an index of the content of all its neighbors within a certain radius and can then process queries on behalf of its neighbors.

A refinement to local indices is defined by Crespo and Garcia-Molina [40], who are among the first to suggest a content-based approach. All content in the network is annotated with

meta-data, and the peers try to predict based on a query's keywords which remote peer is capable of answering it. Each peer maintains an index of content summaries of its neighbors. For selecting the most appropriate peer, 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 method of Crespo and Garcia-Molina is called *Routing Indices* (RIs). Both local indices and routing indices are actively created by exchange of content summaries between the peers.

2.3.5 Exploitation of network topology

As described in [148, 110, 116], most real-world peer-to-peer networks have power-law degree distributions. In such a network topology, the probability that a node has x outgoing links is proportional to $x^{-\tau}$, where $\tau > 0$ is referred to as the exponent of the distribution. Several approaches to search in peer-to-peer networks exploit the topology of the network for creating appropriate routing schemes.

Adamic et al. [7, 6] address the question of finding the shortest path between a source node and a target node in a power-law network. Their assumption is that each node knows about the degree of its neighbor nodes. The search messages are then always passed on to the neighbor with the highest degree which has not already been visited. Compared to random walks, a *high degree seeking* algorithm covers a greater portion of the network (that is, its well-connected part) more quickly. Using high degree seeking for keyword-based or filename-based search in peer-to-peer networks requires every peer to manage a local index of the resources of its neighbors one and two hops away.

A very sophisticated exploitation of the power-law degree distribution can be found in *percolation search* [117, 118]. Percolation search consists of three building blocks: content implantation, query implantation, and bond percolation. Content implantation means that every peer announces its content through a short random walk. Query implantation, which is executed each time a peer issues a query, means that the query will be "implanted" at a small number of peers through a short random walk. Content and query implantation ensure that content and queries are known by at least one high-degree node, since a random walk in a power-law network gravitates towards high-degree nodes because they have a higher number of incoming links. The search is executed in the bond percolation step. Each peer which has an implanted query will perform a probabilistic broadcast, in which it sends the query to each of its neighbors with probability q. Probability q is set to such a value that a query starting from a high-degree node will be broadcast to the so-called giant component of the network (which all high-degree nodes are part of with a high probability). Since content implantation ensures that each resource is known by at least one high-degree node, it will be found.

2.3.6 Reputation learning

Instead of creating local indices by actively exchanging information between the peers about their resources (cf. Section 2.3.4), it is also possible to gain information about the remote peers' resources passively by observing the user queries and their answers that are forwarded through the local peer. This has the benefit that no additional traffic is necessary for creating the local indices. The local indices are used to forward queries to those peers that are likely to have resources appropriate for answering it. Joseph and Hoshiai coined the term *reputation learning* for this approach [76]. The basic premise behind reputation learning is that peers which were able to satisfy previous queries are more likely candidates to answer a current query which is similar (see [37]).

In the *Intelligent Search Mechanism* [78], a local index is composed of pairs $(query, peer)$, where $peer$ is the ID of the peer that was able to answer a query and $query$ are the keywords of the query. Since an answer is transported back hop-by-hop to the querying peer, all peers that lie between a querying and an answering peer gain access to this information. To improve performance, each peer broadcasts the query and its own ID to its one hop away neighbors each time it successfully answers a query. There is a size limit on the local index, and the Least Recently Used policy is used for removing outdated pairs. To decide which peer a query should be forwarded to, the peers in the index are ranked according to the given query. The vector space model and cosine similarity are used to compute query similarities.

Tsoumakos and Roussopoulos [132] introduce a search algorithm called *APS (Adaptive Probabilistic Search)*, which is an extension of the k-random walker approach. Queries are issued for certain objects, and the local indices contain goodness values for every outgoing link and every object. Similar to the SEMANT algorithm, the goodness values are used by the walkers to make probabilistic decisions about which outgoing link to select for finding a certain object. The walkers choose between two modes of operation. In the optimistic approach, they assume that an object will be found along a certain path, and therefore increase its goodness value each time a link is used. In the pessimistic approach, they assume otherwise and decrease its goodness when using a link. If the assumption was wrong, the goodness values need to be corrected by the walker when traveling back to the querying peer. The decision between choosing the optimistic or the pessimistic approach is based on monitoring the success rate for every type of object request. In addition, the goodness values are increased in proportion to the distance between the peer and the object found.

InfoBeacons [39] implements a distributed search engine based on a peer-to-peer middleware system consisting of so-called *beacons*. Each beacon manages a small number of information sources and provides an interface for user queries. It is in charge of sending user queries to appropriate sources utilizing wrappers that are responsible for translating queries to the search interface locally exposed by the information source. If a user's query

can not be answered by any of the information sources the beacon is connected to, the query is forwarded to another beacon. All query results are cached at the beacons. The cached information consists of 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 for directing future queries to appropriate sources. For a query Q, comprised of a set of n keywords, ProbResults 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. If the count for a keywords is zero for a given source, a constant $0 < PW^{min} < 1$ is used to prevent that the outcome of ProbResults will be zero.

A heuristic called *experience weighting* provides for adaptation of cached information to changes at information sources. An experience factor $EF \geq 1$ is defined. After executing a query, each CW_i^s is multiplied by EF if source s returned any results, or divided by EF in case source s returned no results. The author discusses the possibility of using a forgetting factor μ from reinforcement learning (similar to the evaporation factor used in Ant Colony Optimization, see Section 3.2) instead of experience weighting.

2.3.7 Semantic overlay networks / Shortcut networks

All the approaches mentioned so far do not make any changes to the topology of the overlay network. There are several approaches that suggest to cluster semantically related nodes, either by creating shortcut links between peers with similar content, or by grouping all peers into an overlay network on top of the existing overlay. The basic premise behind that, namely *interest-based locality* [123], is that peers which possess the same or similar resources are more likely to be able to answer each other's queries.

Cohen et al. [37] suggest that all peers which share a certain resource should form a so-called *possession-rule sub-overlay*. Consequently, each peer is a member of multiple overlays. If it issues a query, the query is sent to a randomly selected overlay the peer is a member of.

Crespo and Garcia-Molina [41] propose *semantic overlay networks*. In this approach, the resources at each peer are classified according to the concepts of a classification hierarchy. There is a semantic overlay network for every concept, which a peer joins if it possesses a significant amount of resources related to that concept. The queries are also classified, and each query is sent to those semantic overlay networks that correspond to the concept(s) of the query. If they do not provide enough results, the query will additionally be sent to the semantic overlay networks corresponding to the super-concepts.

Cholvi et al.'s [34, 35] concept of *acquaintance links* uses the 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. It turns out that the peers tend to organize into a super-peer structure, where powerful peers that have a big amount of resources at their disposal are better connected than less powerful peers.

	gain info of remote content	informed (directed) search	probabilistic broadcast	changes to topology (shortcuts)	content-based routing
BFS/DFS	no	no	no	no	no
Directed BFS	no	first hop	no	no	no
Modified BFS	no	no	no	no	no
Random Walks	no	no	no	no	no
Local Indices	actively	no	no	no	yes
Routing Indices	actively	yes	no	no	yes
High-degree Seeking	actively	yes	no	no	no
Percolation Search	actively	yes	no	no	no
Intelligent Search	passively	yes	no	no	yes
Adaptive Probabilistic Search	passively	yes	yes	no	yes
InfoBeacons	passively	yes	no	no	yes
Semantic Overlay Networks	n/a	n/a	no	yes	yes
Acquaintance Links	passively	yes	no	yes	yes
Firework Model	actively	yes	no	yes	yes
REMINDIN'	passively	yes	no	yes	yes
SEMANT	passively	yes	yes	no	yes

Table 2.1: Summary of approaches

The *firework query model* [102] uses the vector space model for computing the similarity between the resource repositories of the participating peers. When a new peer joins the system, it creates so-called *attractive connections* to those peers that have a similar resource repository. Consequently, peers with similar content form clusters because they create attractive connections to each other. If the similarity between a query and the resource repository of a certain neighbor peer P lies below a pre-defined threshold value, the query is forwarded to this peer P using the attractive connection. Since the peers are clustered, it is likely that peer P has one or more attractive connections to peers with similar resource repositories. Hence, the query will be flooded to all the peers within the same cluster.

The *REMINDIN' algorithm* [130, 88] employs social metaphors for shortcut creation, that is, the way humans try to find answers for their question in a real-world scenario. The algorithm defines three different kinds of shortcuts. The first kind links to peers that possess resources which are suitable answers for a given query (content provider peers). The second kind links to peers that issued similar queries in the past (recommender peers). The third kind of shortcuts provides links to peers that have a high amount of knowledge about other peers (bootstrapping peers).

2.3.8 Comparison of approaches

Table 2.1 provides a summary of the approaches discussed in this section classified by their features. To enable comparisons, this table also includes the features of the SEMANT algorithm described in Chapter 4.

Chapter 3

Ant-inspired emergence and self-organization

This thesis is motivated by the evident similarities between the self-organizing behavior of ant colonies and self-organization in peer-to-peer networks [68]. This chapter introduces the basic principles of emergence and self-organization and gives a short summary of the relevant literature in this research area. After presenting a general overview of ant algorithms and their properties, three different ant algorithms that are relevant for the SEMANT algorithm are introduced and discussed. In particular, the applicability of the selected algorithms for search in peer-to-peer networks is evaluated.

Contents

3.1 Emergence and self-organization

Although often used as synonyms, the terms *self-organization* and *emergence* refer to different concepts that often – but not necessarily – occur in combination. De Wolf and Holvoet [140] provide a discussion of their differences and suggest the following working definitions:

- A system exhibits **emergence** when there are coherent emergents at the macro-level that *dynamically arise from the interactions between the parts at the micro-level*. Such emergents are novel w.r.t. the individual parts of the system.
- **Self-organization** is a dynamical and adaptive process where systems acquire and maintain structure themselves, *without external control*.

Emergent and self-organizing phenomena can be observed from nature, where they are ubiquitous. Since emergent systems build their results incrementally and dynamically, it is impossible to predict their behavior in advance. Although a high number of experiments empirically observe the appropriateness of emergent systems for different application scenarios, no formal proofs exist [73].

Researchers from many different areas such as physics, mathematics, engineering, biology, philosophy, and the social sciences are interested in emergent and self-organizing phenomena and the field has a long history. A historical overview about how the principles of emergence were discovered for the first time – a story that includes very famous people such as Alan Turing and Evelyn Fox Keller – can be found in [75].

In the 1940s, a group of scientists around Norbert Wiener [137] founded the research area of *cybernetics*, which is concerned with the investigation of control and communication in biological, mechanical, and social systems consisting of connected parts. In particular, cybernetics is interested in the information transfer and the feedback mechanisms occurring in the relationships between those parts. The system has the goal of maintaining some specific internal state, i.e., it behaves goal-oriented. If its current state is different from the goal state, the system will return from this state to the goal state.

The terms *bionics* and *biomimicry*, which were coined in the 1950s, refer to the principle of applying methods or systems found in nature to engineering. A distinction is made between rebuilding natural phenomena (biomimicry) and solving an actual problem in engineering using a bio-inspired model (bionics). In the domains of computer science and artificial intelligence, many existing techniques and research areas are inspired by bionics, such as swarm intelligence, neural networks, simulated annealing, evolutionary algorithms, and many others. Researchers in computer science tend to take the natural phenomena as an input and a metaphor, but many of them build their own extensions on top of the model of the natural phenomena. In addition, since most of the natural phenomena are not yet completely understood, the model of the natural phenomena can only be a simplification of it. Therefore, bionic systems are easier to achieve than biomimetic systems.

Since the 1980s, the natural phenomena that are mimicked are also called *complex systems* and their study is called *complexity science*. Examples for complex systems are ant colonies, organisms, minds, ecologies, and societies. Although cybernetics and complexity science are based on the same principles, there seems to be only little cooperation between those fields. Similar to a system in cybernetics, a complex system has a network-like structure; it consists of parts (agents) which are coupled. Complex systems dynamically change over time, and their present state is influenced by the former states. A complex system has the objective of preserving its collective functionality, e.g., adaptation to different environmental situations. The agents contribute to that goal by seeking to maximize their local view of the system according to some measure of goodness. The relationships between the agents are non-linear and may contain positive and negative feedback loops.

The term *complex adaptive systems* was coined by John H. Holland and others (see [84]). It refers to complex systems in which the agents are adaptive, that is, they can change their behavior in reaction to what they learn from interaction with their environment. Complex (adaptive) systems usually exhibit emergent and/or self-organizing behavior. This line of research puts the emphasis on the self-learning aspects of the systems, whereas cybernetics is mostly concerned with observing closed-loop control systems. Unlike in cybernetics, where research had to be conducted in an abstract way, the scientists working on complex adaptive systems have the necessary tools available for creating computer simulations of their models. Hence, the field of computer science contributes back by providing the necessary research tools for biologists, physicians, and sociologists who need to simulate and visualize the phenomena observed from nature.

In the field of artificial intelligence (AI), there are two opposing approaches to creating intelligent behavior: *Symbol processing systems* and *sub-symbolic systems*. In symbol processing systems, knowledge is stored as a set of symbols and reasoning is manipulation of these symbols. This is a top-down approach known as the classical approach, which is by far the dominant one. On the other hand, in *sub-symbolic processing* (which came up in the 1990s), knowledge is represented as a kind of network with interacting nodes, and reasoning is adjusting the weights on the network's nodes. Similar to complex adaptive systems, intelligent behavior emerges in a bottom-up manner from the interaction between the nodes. Since some proponents of sub-symbolic processing heavily attacked the physical symbol system hypothesis that the classical AI approach is based on and advocated the physical grounding hypothesis [23] instead, which demands the construction of physical agents, this line of research was not widely adopted. Other instances of sub-symbolic processing that do not rely on the physical grounding hypothesis are *neural networks* and *swarm intelligence* [19]. The term swarm intelligence emphasizes the fact that all the agents in the system have the same (simple) behavior and capabilities. Ant algorithms, which are described in the next section, can be classified as a swarm intelligence technique.

3.2 Ant-based methods

Ant algorithms are inspired by the collective foraging behavior of specific ant species. For different purposes, e.g., foraging, alarm behavior, or necrophoric behavior, ants indirectly communicate with the other members of their ant colony. The communication is based on modifying the local environment by dropping a chemical substance called *pheromone*. This was first observed by Grassé in 1959 who named the phenomena *stigmergy* [61]. For foraging, some species use the so-called *trail-laying and trail-following behavior* for finding the shortest path between a nest and a food source. The trail-laying and trail-following behavior consists of the following three basic principles:

1. Each time an ant moves, it lays a pheromone trail.
2. For finding its way, it senses its environment and
   a) follows existing trails, if there are any. The probability that the ant chooses a certain trail is proportional to the amount of pheromone on that trail.
   b) walks randomly, if no trails can be found.
3. Pheromone evaporates over time. If a trail is not used, it will vanish.

This is quite a simple behavior. However, it is sufficient for finding the shortest path – or a relatively short path – between two geographical locations. The ants travel from the nest to the food source, and back again. In the beginning there are no pheromone trails, so all the ants will walk randomly. The ant that randomly chose the shortest path will arrive first at the food source, and is also the first one to go back to the nest. For the way back, there is a high probability that it will use the pheromone trail it created by itself. If it does so, the strength of this trail will be doubled. When the other ants arrive at the food source, the trail of the first ant will be the strongest one. For this reason, the other ants will probably follow it as well. If so, the strength of the trail for the shortest path found will be increased further. Now its likely that the trail holds a significant concentration of pheromone. Consequently, all the other ants will follow it with a high probability. In summary, the trail-laying and trail-following behavior is another incidence of the "rich-get-richer" phenomena [16].

Ant colonies are multi-agent systems in the sense that each ant/agent has incomplete information or capabilities for solving the problem, that there is no global system control, that data is decentralized, and that computation is asynchronous [128]. In addition, ant colonies exhibit emergent behavior (cf. Section 3.1) because they build their results incrementally. Therefore, ant colonies are complex adaptive systems (cf. Section 3.1).

Several ant algorithms exist that model and exploit the trail-laying and trail-following behavior for solving graph-based NP-hard combinatorial optimization problems. They have been most extensively applied to the Traveling Salesman Problem (TSP, [83]), but to many others as well. Ant algorithms can be classified as a *Monte Carlo technique* [115], and there are similarities between *reinforcement learning* [127] (a technique known from machine learning) and ant algorithms. In particular, as stated in [54], a special form of reinforcement learning called *Q-learning* has many commonalities to ant algorithms.

The building blocks an ant algorithm should have in order to be justified as such are defined by Dorigo and Di Caro in the *Ant Colony Optimization* [45, 46] meta-heuristic. Dorigo and Stützle [48] claim that ACO can be applied to every discrete optimization problem for which some solution construction mechanism can be conceived. The meta-heuristic dictates three building blocks and their sequence of execution:

1. **Trail following (state transition rule).** For every iteration, each ant determines a solution to the optimization problem. In the Traveling Salesman Problem, this would be

a path through the graph from its (randomly selected) starting node to its destination node. A so-called *state transition rule* defines how the ants choose their path through the network. This rule derives which one of all possible next nodes to choose and is executed every time an ant has to select the next node of the path. This decision is based on two different types of information:

- The first type is the cost of the outgoing links (called *local information*). This information is problem-specific. In the Traveling Salesman Problem, the local information would be the geographical distance between the cities.
- The second type of information is the pheromone distribution of the outgoing links (called *global information*). This information is built incrementally by the ants.

2. **Trail laying (pheromone update rule).** When an ant arrives at the destination node, the total cost of the newly found path is calculated. The amount of newly dropped pheromone is inversely proportional to the cost of the found path. It is defined by the *pheromone trail update rule*. The same amount is used for each edge which is part of the found path. Depending on the algorithm, pheromone is either dropped to all the found paths, or only to the best one found in one iteration, or only to the best one found in one iteration if is the currently globally best one of all iterations.

3. **Evaporation (evaporation rule).** In each iteration, a certain percentage of pheromone evaporates according to an evaporation factor. Since each edge of the problem graph is initialized with a small amount of pheromone, the pheromone amount of an edge will always be a positive value greater than zero. Evaporation prevents unlimited increments in pheromone amount. In addition, it avoids that paths become too dominant: Since the pheromone amount is lowered by multiplication of the current value with a constant smaller than one, evaporation removes more pheromone from trails that store a high amount than of those that store a low amount.

Using this procedure, the ants will explore the search space and incrementally improve the solution by using the pheromone trails – which act as the memory of previous attempts – as a guidance for future iterations. Ant algorithms are known for building an acceptable solution rather quickly, but requiring a long time for reaching a converged stage where the solution is optimal. To some extent, convergence has been proven formally. Gutjahr [62] shows that for a particular *Ant Colony Optimization* instance called *Graph-based Ant System* and with certain constraints on the parameter settings, the probability that the current solution will converge to an optimal solution is 1 for $t \to \infty$. Stützle and Dorigo [125] show the same for the $\mathcal{MAX} - \mathcal{MIN}$ *Ant System* algorithm.

In this thesis, the focus is on the *Ant Colony System* [47] algorithm, which is described in the next section, the *AntNet* algorithm (see Section 3.3.1), and the *AntHocNet* algorithm

(see Section 3.3.2). The other existing instances – i.e, *Ant System* [38] and its variations, or $\mathcal{MAX} - \mathcal{MIN}$ *Ant System* [126] – are similar to *Ant Colony System* and vary mostly in their transition rules and pheromone trail update rules, or some other additional constraints or small add-ons. A very good overview about these algorithms and the differences between them is provided in [60].

3.2.1 *Ant Colony System*

The *Ant Colony System* [47] algorithm was published in 1997 as an improvement to *Ant system* [38]. It is the most prominent instance of *Ant Colony Optimization* and has been applied to the Traveling Salesman Problem [47], the Vehicle Routing Problem [56, 95], the Sequential Ordering Problem [55], the Bin Packing Problem [50], and many other combinatorial optimization problems. In the following, *Ant Colony System* will be described using the Traveling Salesman Problem as an example (Section 3.2.1.1) and after that its applicability for query routing in peer-to-peer networks will be discussed (Section 3.2.1.2).

3.2.1.1 Description of the algorithm

Ant Colony System implements the principles described in Section 3.2. The optimization problem to solve is represented as a graph. In this graph, each link (r, s) leads from node r to node s. Every link (r, s) stores a pheromone value $\tau(r, s)$, and has a link cost $\eta(r, s)$ which is the inverse value of the distance between city r and city s.

State transition rule. The algorithm employs a very sophisticated state transition rule which consists of two strategies together with a parameter $0 \leq q_0 \leq 1$ that defines the weight for each strategy. Let k be an ant positioned at node r. Ant k applies the state transition rule shown in Equation 3.1a to select the next node s. The ant uses the roulette wheel selection technique [58] to select one of the two the strategies. In the first step, it computes a random value $q \in [0, 1]$. If $q \leq q_0$, it will select the best link in terms of low link costs and a large amount of pheromone. In this strategy – which is called the exploiting strategy – the ant exploits the information collected by its predecessors by selecting the best possibility known so far.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\} & \text{if } q \leq q_0 \text{ (exploitation)} \\ S & \text{otherwise (exploration)} \end{cases} \tag{3.1a}$$

In Equation 3.1a, $J_k(r)$ is the set of all nodes ant k still has to visit. Weight $\beta > 0$ defines the influence of link costs. Variable S is derived according to Equation 3.1b, which comes to use if $q > q_0$. In this case, ant k considers the set of nodes $J_k(r)$ it did not visit yet and derives a probability distribution for all those nodes. Each probability $p_k(r, s)$ is proportional to the desirability of choosing (r, s) in terms of link costs and pheromone amount. Based on this probability distribution, the roulette wheel selection technique is used for choosing node s.

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,u)]\cdot[\eta(r,u)]^\beta}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^\beta} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{3.1b}$$

The state transition rule in Equation 3.1b is called random-proportional. In this strategy – which is called the exploring strategy – the ant uses the information collected by its predecessors by randomly selecting a link in proportion to its desirability.

Pheromone update rule and evaporation. *Ant Colony System* uses two pheromone update rules which complement each other. They are applied in different situations. Both of them use a combined formula for the pheromone updates and evaporation. The *global pheromone update rule* is shown in Equation 3.2. It is applied after all ants of an iteration have found a path.

$$\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s) + \alpha\cdot\Delta\tau(r,s) \tag{3.2}$$

In Equation 3.2, the amount $\Delta\tau(r,s)$ is the inverse of the length of the global best tour if link (r,s) belongs to the global best tour. If not, $\Delta\tau(r,s) = 0$. This means that additional pheromone is dropped only if an ant has found a path that is better than the currently best one. The constant $0 < \alpha < 1$ defines the weight between evaporation and updates.

Since the global rule puts a high emphasis on the best path found so far, the ants will stay in the neighborhood of this path. The *local pheromone update rule* shown in Equation 3.3 is designed for making the less-visited paths more attractive by removing a small amount of pheromone from an edge every time it is visited by an ant.

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \rho\cdot\Delta\tau(r,s) \tag{3.3}$$

In Equation 3.3, the amount $\Delta\tau(r,s) = \gamma\cdot max_{z\in J_k(s)}\tau(s,z)$, where γ is a constant $0 < \gamma < 1$. Therefore, the amount of additional pheromone is proportional to the pheromone amount of the outgoing link with the largest amount of pheromone. The constant $0 < \rho < 1$ defines the weight between evaporation and updates.

3.2.1.2 Discussion

Now the applicability of *Ant Colony System* for query routing in peer-to-peer networks is discussed. *Ant Colony System* is used for solving combinatorial optimization problems, and query routing in peer-to-peer networks can be seen as a special kind of combinatorial optimization problem. The main difference between this problem and the Traveling Salesman Problem is that in the latter the task is to find one solution (the shortest path that visits every city just once), whereas in the former there are lots of sub-solutions (every query is issued at a certain peer with certain keywords, and the shortest path to an appropriate peer offering answers to the query must be found). As mentioned above, *Ant Colony System* has

been applied to many different combinatorial optimization problems. This implies that the algorithm provides a high degree of adaptability.

Compared to the transition rules of the other algorithms reviewed later in this chapter, *Ant Colony System*'s transition rule is the most comprehensive and the most adaptable one. It will be selected for the SEMANT state transition rule as-is. In order to account for the fact that there are multiple possible sub-solutions, using the *Ant Colony System* state transition rule will be experimentally evaluated first. Based on the results, adaptations that encourage the ants to use multiple paths can be designed.

The combination of global and local pheromone update rule is tailored for the task of finding one solution, so it will not be adopted for SEMANT. Instead, only a global pheromone update rule is used. For defining the amount of new pheromone to add, the quality of a found path needs to be calculated. In the SEMANT scenario, there are two metrics that define the quality of a found path. These are the number of results found, and the number of hops that lie between the querying and the answering peer. Since *Ant Colony System* is usually used in a centralized environment, all the pheromone trails can be updated at the same time. This is not the case for SEMANT. The question of how to update pheromone trails in a distributed scenario is answered in the next section.

3.3 Distributed ant-based methods

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. Subsumed under the name *Ant Colony Routing* [21], a dedicated subset of ant algorithms was specifically designed for managing routing tables in telecommunication networks, in fixed networks based on the IP protocol, and in mobile ad-hoc networks. The three most prominent variants of ant algorithms for routing of data packets are:

- **Ant-based control (ABC).** The *ABC* algorithm was published in 1996 by Schoonderwoerd et al. [122]. It is designed for circuit-switched networks and its pheromone updating approach is appropriate for symmetric networks only. Hence, its application scenario is very different from the characteristics of peer-to-peer networks. For this reason, *ABC* is not evaluated further in this section.

- **AntNet.** This approach was published in 1998 by Di Caro and Dorigo [27]. It is designed for packet-switched networks and its pheromone updating approach is appropriate for both symmetric and asymmetric networks. *AntNet* is the most prominent distributed ant algorithm.

- **AntHocNet.** This algorithm was published in 2004 by Di Caro, Ducatelle, and Gambardella [28]. It is designed for mobile ad-hoc networks (MANETs).

Both *AntNet* and *AntHocNet* are designed for routing of data packets, that is, the process of selecting the next hop for an incoming data packet being forwarded based on information held in routing tables. The three main differences between routing of data packets and query routing are as follows.

- Firstly, when routing data packets, the destination address of a packet is known in advance. In query routing, it is not.
- Secondly, network routing does not account for the contents of data packets. On the contrary, in content-based approaches to query routing, the contents (keywords) of a query are considered in order to decide which node to send the query to.
- Thirdly, for routing of data packets, there is just one appropriate destination node. In query routing, there are possibly many of them.

However, despite these differences, the common ground lies within the distributed application area. In the following, the main building blocks and the distributed aspects of *AntNet* (see Section 3.3.1) and *AntHocNet* (see Section 3.3.2) are evaluated in order to identify those building blocks that can be transferred to query routing in peer-to-peer networks as well as those that can not.

3.3.1 *AntNet*

In *AntNet*, ants collaborate in building routing tables for routing of data packets in a communication network with the aim of optimizing the routing performance of the entire network. The algorithm is distributed and adaptive in the sense that it adapts the routing policy according to the current traffic conditions in the network.

In the next sections, first the *AntNet* algorithm (Section 3.3.1.1) and after that its strategy for preventing cycles (Section 3.3.1.2) are described in detail. The description is followed by a discussion of the adequacy of *AntNet*'s components for search in peer-to-peer networks (Section 3.3.1.3).

3.3.1.1 Description of the algorithm

The *AntNet* [27, 26] algorithm operates at the IP layer and does not offer management of error, flow, and congestion control. In addition, the algorithm does not provide support for link failures, node failures, or new nodes that join the network.

Data structures. The network that the algorithm should be deployed in is mapped on a directed weighted graph with N nodes. The edges of the graph are the links between

the nodes. The links are viewed as bit pipes with certain costs – bandwidth, measured in bits per second, and transmission delay, measured in seconds – that depend on the current load of the links. Each node n manages a routing table $\mathcal{T}_n$ 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. Each entry P_{nd} of these tables is a probabilistic value indicating the goodness of choosing the link to neighbor node n with respect to a given destination node d.

At startup, all entries in the routing table are initialized with equal values in such a way that the sum of all entries in a column is 1. Barán et al. [17] suggest an improvement of the initialization: Instead of using a uniform distribution, more weight should be put on those links that lead to neighboring nodes that simultaneously could be destination nodes.

Next to the routing table $\mathcal{T}_n$, each node n manages an array $\mathcal{M}_n$ which stores statistical information for each destination node d about the time it takes to travel from node n to node d. The information is comprised of (1) an estimated mean μ_d for the expected trip time, (2) the variance σ_d^2 expressing the stability of the expected trip time value, and (3) the shortest trip time W_{best_d} to node d as observed in a defined number of last samples.

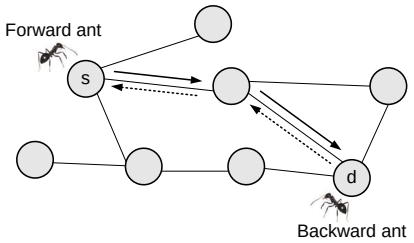

Figure 3.1: Forward and backward ants

Forward ants. Additional traffic – so-called forward ants – is created for keeping the information in $\mathcal{T}$ and $\mathcal{M}$ updated. At regular intervals, each node s generates a forward ant (see Figure 3.1) that builds a path to a randomly selected destination node d by applying the transition rule shown in Equation 3.4 each time it arrives at a node k and has to select an outgoing link. The transition rule decides based on the entry P_{nd} found in the routing table and includes the factor $l_n \in [0, 1]$. For each outgoing link n, factor l_n is proportional to the current length of the link queue to n in terms of bits waiting to be sent. The weight $\alpha \in [0, 1]$ defines the influence of the queue length factor. The authors suggest that the best value for α is between 0.2 and 0.5.

$$P'_{nd} = \frac{P_{nd} + \alpha l_n}{1 + \alpha(|neighbors(k)| - 1)} \tag{3.4}$$

When a forward ant has reached its destination node, it calculates the total trip time T of the path it used. The value T indicates the goodness of the path it found. Since each node

stores routing information locally, the forward ant cannot update the goodness values in the routing tables directly. Instead, it generates a backward ant that returns to the source node through the same path that was used by the forward ant. After creating the backward ant and passing it over a copy of the stack data it recorded while traveling to the destination node, the forward ant terminates.

Backward ants. The backward ant is responsible for updating the goodness values and the statistical information according to the information gathered by the forward ant by altering the data structures $\mathcal{T}$ and $\mathcal{M}$ at each visited node (see dotted arrows in Figure 3.1). 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. In *AntNet*, the sum of all probabilities which refer to a given destination node is always 1. This is achieved by increasing the probability value for the path that was used, and by decreasing the other values for destination node d accordingly, as shown in Equation 3.5a and Equation 3.5b. In these equations, the backward ant arrives from a node x at a node k and updates $\mathcal{T}_k$. The amount of pheromone is determined by the factor $r \in [0, 1]$ which is derived by comparing trip time T to the current best trip time W_{best_d} (see [27] for details). Since the goodness of a trip time is relative to the current load of the network, the comparison takes the mean and variance estimates into account as well.

$$P_{xd} \leftarrow P_{xd} + r(1 - P_{xd}) \tag{3.5a}$$

$$P_{nd} \leftarrow P_{nd} - rP_{nd}, n \in neighbors(k), n \neq x \tag{3.5b}$$

Next to updating $\mathcal{T}_k$, the backward ant also updates the data structure $\mathcal{M}_k$ with the recorded trip time from node k to node d. The mean and variance estimates are updated with the new values. If the new result is the best one out of the last w samples, it is stored in W_{best_d}.

Figure 3.2: Sub-paths of a path

Optimizing sub-paths. In the description above, the modifications of routing table entries corresponding to destination node d were specified. Consider a forward ant traveling from node a over nodes b and c to node d (see Figure 3.2). The corresponding backward ant travels from node d to node a. It updates the entries $\mathcal{M}_c(\mu_d, \sigma_d^2, W_{best_d})$, $\mathcal{M}_b(\mu_d, \sigma_d^2, W_{best_d})$, and $\mathcal{M}_a(\mu_d, \sigma_d^2, W_{best_d})$, and increments $\mathcal{T}_c(P_{cd})$, $\mathcal{T}_b(P_{bd})$, and $\mathcal{T}_a(P_{ad})$. Since the forward ant visits a couple of other nodes before arriving at the destination, the backward ant knows about the trip times to these nodes as well. Hence, the backward ant also performs the updates described above on the sub-paths (path ac, path ab, and path bc), but only if sub-paths are found that are significantly better than the current ones. If this is the case, the entries

$\mathcal{M}_a(\mu_c, \sigma_c^2, W_{best_c})$, $\mathcal{M}_a(\mu_b, \sigma_b^2, W_{best_b})$, $\mathcal{M}_b(\mu_c, \sigma_c^2, W_{best_c})$, $\mathcal{T}_a(P_{ac})$, $\mathcal{T}_a(P_{ab})$, and $\mathcal{T}_b(P_{bc})$ are updated.

3.3.1.2 Preventing cycles

The *AntNet* algorithm 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 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 v.

3.3.1.3 Discussion

Now the applicability of *AntNet* for search in peer-to-peer networks is discussed. One of the reasons which make it impossible to apply the *AntNet* algorithm without any changes to query routing in peer-to-peer networks is that all nodes that exist in the network must be known at each node in order to choose random destination nodes for forward ants. However, this requirement stems from data packet routing. It is not necessary to adopt this behavior for query routing.

The components of the algorithm that are most useful for the realm of peer-to-peer networks and thus will be adopted are those that are designed for coping with the distributed environment. These are:

1. **Data structures.** The distributed data structures used in *AntNet* are necessary in a peer-to-peer environment as well. However, in *AntNet* every column in the routing tables corresponds to a destination node in the network. For query routing, the columns must correspond to the keywords of the queries instead.

2. **Forward and backward ants.** The concept of forward and backward ants is absolutely necessary in a distributed environment. Some changes need to be made, because in *AntNet* the forward ant has a well-defined destination node, which is not the case for query routing in peer-to-peer networks. The drawback of this approach is that the backward ant visits every node on the path between the answering and the querying peer, although it would be possible to jump directly to the querying peer instead. However, in order to be able to use an ant algorithm, this can not be circumvented.

3. **Preventing cycles.** *AntNet*'s strategy for preventing cycles can be adopted for the SEMANT algorithm without any major changes.

Another idea from *AntNet* that can be easily transferred is the optimization of sub-paths. Since in SEMANT the pheromone trails correspond to the keywords of the queries instead of corresponding to destination nodes, sub-paths are automatically optimized.

The state transition rule used in *AntNet* (see Equation 3.4) puts high emphasis on the current loads of the links. Although considering the current load in the network is an aspect that could improve search performance in a peer-to-peer network, it is not the main goal of the thesis. Thus, adopting the state transition rule of *AntNet* as-is is not desirable. If those parts from the state transition rule that consider the current loads are removed, the remaining part is basically the same as the exploring strategy used in *Ant Colony System* (see Equation 3.1a). Therefore, it makes no sense to select *AntNet*'s state transition rule as the basis for adaptations.

The pheromone update rule used in *AntNet* enforces that the sum of all goodness values corresponding to a certain destination node is always 1. This makes the routing of data packets faster, because no further normalization of goodness values is necessary at runtime. Since query routing is less time-critical than routing of data packets, it is not necessary to transfer this feature. In addition, enforcing a sum of 1 makes it more difficult to include an evaporation feature.

3.3.2 *AntHocNet*

Next to their application in fixed networks, ant-based methods have also been employed for routing of data packets in mobile ad-hoc networks (MANETs, [91]). Mobile ad-hoc networks are networks in which the participating nodes are in motion while communicating, e.g., using handheld devices. The most prominent ant algorithm for this application scenario is *AntHocNet* [28, 51, 12] by Ducatelle, Di Caro, and Gambardella. In this section, *AntHocNet* is described (Section 3.3.2.1) and its applicability in peer-to-peer networks is discussed (Section 3.3.2.2).

3.3.2.1 Description of the algorithm

The *AntHocNet* algorithm is called a hybrid approach because it combines reactive and proactive components. Reactive behavior means that routing information is gathered in response to a certain event, whereas proactive behavior means that information is gathered beforehand, reckoning that exactly this information might be necessary soon. *AntHocNet* relies on an improved variant of *AntNet*'s concept of forward and backward ants. The algorithm uses two different kinds of forward ants for the reactive and for the proactive part, which are described in the following.

Reactive part. The reactive part of the algorithm is used for setting up a path between a source node s and a destination node d which want to establish a communication session. In

this case, a reactive forward ant F_d^s is launched at node s in order to build a path to node d. *AntHocNet* relies on a combination of unicast and broadcast forwarding:

- **Broadcast.** If there is no appropriate pheromone information available at a node, the reactive forward ant will proceed to all neighboring nodes. Since broadcast is used, multiple instances of the same forward ant can exist. All ants together that are generated by an initial forward ant starting at the source node form a so-called *ant generation*.
- **Unicast.** If a forward ant arrives at a node that stores pheromone information corresponding to node d, it will consult this information in order to decide which node to choose next. Each node stores a table $\mathcal{T}^i$ with entries $\mathcal{T}_{nd}^i$ that correspond to the goodness of choosing neighbor n to go to destination node d. Neighbor node n is chosen with the probability P_{nd}, which is computed as shown in Equation 3.6.

$$P_{nd} = \frac{\left(\mathcal{T}_{nd}^i\right)^{\beta_1}}{\sum_{j \in \mathcal{N}_d^i} \left(\mathcal{T}_{jd}^i\right)^{\beta_1}}, \beta_1 \geq 1 \tag{3.6}$$

In Equation 3.6, $\mathcal{N}_d^i$ is the set of neighbors of i which store pheromone information about their goodness when going to node d, and β_1 allows additional control of the reactive process. To ensure that only one path between s and d is set up, every ant that arrives at a node already visited by another ant of the same ant generation will be terminated. Similar to *AntNet*, the forward ant keeps a list of node it visited. Arriving at the destination node d, the forward ant will create a backward ant which goes back to the source node using the same path in the reverse direction. At each intermediate node x, the backward computes a local estimate of the time it takes to reach the neighbor it came from. These estimations are accumulated to compute the time $\hat{T}_d^x$ it takes to go from x to d. As shown in Equation 3.7, in addition to the estimated value, the algorithm also considers the number of hops between x and d for computing τ_d^x, which is the inverse value of the time it takes to go from x to d.

$$\tau_d^x = \left(\frac{\hat{T}_d^x + hT_{hop}}{2}\right)^{-1} \tag{3.7}$$

In Equation 3.7, h is the number of hops between x and d, and T_{hop} is a constant representing the time it takes to move one hop. The value $\mathcal{T}_{nd}^x$ is updated as shown in Equation 3.8.

$$\mathcal{T}_{nd}^x = \gamma T_{nd}^x + (1-\gamma)\tau_d^x, \gamma \in [0,1] \tag{3.8}$$

As soon as the backward ant arrives at the source node, the communication session between node s and node d is set up and data can be transmitted. The data packets are transmitted in the same way as the reactive forward ants (see Equation 3.6), but instead of β_1, a factor $\beta_2 \geq \beta_1$ is used.

Proactive part. The proactive part of *AntHocNet* is responsible for path maintenance and path improvement of already existing paths. Since mobile ad-hoc networks are comprised of nodes that move, the neighbor relationships between the nodes change rather often and path maintenance is an important issue. To keep track of the continuous changes in the neighbor relationships, *AntHocNet* uses *hello messages* which are broadcast by each node in defined regular intervals. If a node a knows about a neighbor node b but does not receive any hello messages from b within a defined period of time, node a will remove the pheromone information about node b from the routing tables of node a. This is *AntHocNet*'s local repair mechanism to cope with link failures. In addition, hello messages are used for discovering new neighbors. If a node a receives a hello message from a node c it did not yet communicate with, it adds node c as a neighbor.

Next to path maintenance, the hello messages are also used for the improvement of existing paths. Piggy-backed with the hello messages, each node sends excerpts of its pheromone tables to its direct neighbors. Imagine a node x broadcasting a hello message and a neighbor node y receiving it. Node y inspects the contents $\mathcal{T}^x_{xd}$ and adds the cost for sending packets from node y to node x, thus creating an estimate $\mathcal{B}^y_{xd}$ for sending an ant to node d over node x. If an entry $\mathcal{T}^y_{xd}$ for going to node d over node x exists, it will be updated using $\mathcal{B}^y_{xd}$. If not, the entry $\mathcal{B}^y_{xd}$ will not be stored in $\mathcal{T}^y$, but in a separate virtual pheromone table $\mathcal{V}^y$ to prevent broadcasting estimates based on non-local information (which would be propagated further, and thus be spread through the whole network quickly). The virtual pheromone tables are read only by those nodes that are source nodes of active communication sessions. At regular intervals, the entries in tables $\mathcal{T}$ and $\mathcal{V}$ corresponding to the destination nodes of the active communication sessions are compared. If the entry in $\mathcal{V}$ is significantly better, a so-called proactive forward ant is unicast to the destination node. It exploits the information from both virtual and regular pheromone tables, and creates a backward ant at the destination node.

3.3.2.2 Discussion

Ducatelle, Di Caro, and Gambardella [51] point out that ant-based methods are suitable for mobile ad-hoc networks because (1) they are adaptive to network changes, (2) robust to agent failures, and (3) can provide multi-path routing. They also state that the drawback of ant-based methods is the repeated path sampling (meaning that lots of ants are sent). Repeated path sampling needs to be dealt with carefully, because significant overhead in network traffic might be created. The same benefits and drawbacks as in mobile ad-hoc networks apply to peer-to-peer networks as well. The reactive part of *AntHocNet* is comprised of forward and backward ants as already known from *AntNet* together with the exploring strategy (without integrating link costs) already known from *Ant Colony System*. Hence, there is basically nothing new to learn here for the big picture. However, there are several

slight differences to the other algorithms:

1. What is different is the factor β_2 used in the transition rule (see Equation 3.6) for the routing packets, which was set to the high value of 20 in the experiments described in [51]. Using this factor has the effect that if one of the available outgoing links has a very high pheromone concentration compared to the others, its desirability for selection is even more emphasized. This could be useful in the SEMANT algorithm as well.

2. Another difference is that the pheromone trails are not initialized. Broadcast is used if no routing information is present for any of the outgoing nodes. This means that for every destination node that is requested for the first time, the whole network will be flooded. Consequently, there will be a high amount of traffic in the network, especially in the startup phase. If pheromone information exists only about a subset of the outgoing links, those links nothing is known about are omitted from the transition rule. Hence, it could happen that some outgoing links are never used, because they were not included in the initial broadcast for some reason. Given those two facts together, using initialization of the pheromone trails seems to be a better approach.

3. The third difference is the combination of unicast – i.e., directed forwarding to one neighbor – and broadcast techniques. Obviously, using broadcast improves the number of results for a query also in a peer-to-peer network, but at the cost of consuming a high amount of network resources. In the experiments described in [51], a maximum of 500 nodes was used, and broadcast seems to perform well in this scenario. In a peer-to-peer network, there are usually more than 500 nodes and broadcast is not scalable [112]. However, for the SEMANT algorithm it seems to be promising not always to rely on unicast forwarding, but to find an appropriate balance between unicast and broadcast by choosing a subset of the outgoing links to forward a query to if the pheromone trails indicate that this would be useful. In particular, choosing a subset seems to be of benefit if some of the outgoing links have the same or a very similar goodness values.

4. Unlike in *AntNet*, optimization of sub-paths is not used.

5. In a mobile ad-hoc network, the distances between the nodes change constantly and differ greatly over time. Therefore, using the *AntNet* approach for integrating link costs is not feasible in this scenario. Instead of incorporating link costs, *AntHocNet* uses a simpler version of local information about the network where the number of hops between the nodes is considered and the time it takes to move one hop is considered a constant factor (see Equation 3.7). The drawback of this approach is its assumption that every link has the same link cost, which is not the case in real-world

settings. Since the pheromone amounts correspond to inverted time values anyway, including the number of hops and estimating them with a fixed time value could lead to wrong results, especially if the links have very different link costs.

6. *AntHocNet* applies evaporation only if a backward ant visits a node. Although the pheromone update rule (see Equation 3.8) is basically the same as the global pheromone update rule of *Ant Colony System* (see Equation 3.2), the difference is that in *Ant Colony System* it is applied in every iteration for every node. This could be useful in peer-to-peer networks as well because in this case information is removed only if new information is available.

The proactive part of *AntHocNet* is relevant for peer-to-peer networks if the dynamic aspects – nodes leaving and joining – are considered. It does not apply to a static setting (as described in Chapters 4 to 5). Seen from the perspective of a neighbor peer, it does not matter if the peer moves away, as in a mobile ad-hoc network, or if it leaves, as in a peer-to-peer network. The effect – node is unreachable – is the same, and the consequences of the network topology changes depend on the actual rate in which these changes occur (churn rate and velocity of the nodes, respectively). However, it can be assumed that the change rate will be higher in a mobile ad-hoc network. What can be learned from *AntHocNet* is (1) the technique of using hello messages for keeping track of the state of the network and (2) the importance of separating local information which is approved by the local peer from locally stored information which is not approved by the local peer.

Chapter 4

The SEMANT algorithm

This chapter covers the specification of the proposed ant algorithm SEMANT for content-based search in peer-to-peer networks. After defining the problem and stating the assumptions which were made, what follows is a detailed description of all aspects related to the SEMANT algorithm. The remainder of the chapter is devoted to the evaluation of the algorithm's performance in various scenarios and settings, and to a discussion of related work on ant algorithms in peer-to-peer networks.

Contents

4.1 Problem description

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

The resources can be any kind of files that are annotated with metadata. The metadata are composed of name-value pairs, also called *elements*. These elements are used for specifying additional information about a certain resource. A meta-data schema defines the name and the meaning for each of the elements the schema is comprised of. Since the query routing procedure is the same no matter which element is employed, only one element is considered in the following. The values that can be used for annotation originate from a controlled vocabulary, e.g, the concepts of a taxonomy or an ontology.

The query vocabulary is the same as the metadata vocabulary used for annotating resources. The queries do not consider the actual content of a resource, but rather the metadata that describes its content.

The challenges for query routing are the following. Each peer is connected via outgoing links to some other peers which are called its neighbor peers. If a peer issues a query or receives a forwarded query from one of its neighbor peers, it has to decide based on its local information which neighbor peer to send the query to. The local information of a peer is gained by continuously observing the queries and answers that pass the local node and by recording which kind of queries its neighbor peers were able to answer in the past (cf. *reputation learning*, Section 2.3). The recorded data must be accumulated and stored in an appropriate way to support the selection process. Based on this information, the peer has to choose the neighbor peer which is (1) most likely to store results itself, or (2) has neighbor peers that are likely to store such resources. Approaches to query routing in which this decision depends on the keywords of the query are called *content-based* approaches. They are sometimes also referred to as *semantic query routing*. The goals for the query routing procedure are to maximize the number of query results and, at the same time, to minimize the usage of network resources. The best metric for measuring if those two goals are reached is to divide the total amount of network resources consumed by a query by the number of query results.

The objective of this work is to employ ant-based methods for the query routing procedure. Basically, the ant metaphor can be applied rather straightforward. The pheromone trails store the recorded data about the successful queries in the past. The queries themselves are represented as ants.

4.1.1 Assumptions

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

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

An additional assumption is that the network topology and the content distribution in the network are considered static.

4.2 Specification of the algorithm

This section specifies the components of the SEMANT algorithm. Section 4.2.1 documents the data structures that need to be stored at each peer. Section 4.2.2 describes the query routing procedure and all aspects related to facilitating ant algorithms in a peer-to-peer network. In Section 4.2.3, all issues related to the selection of outgoing links for query routing are covered. Section 4.2.4 explicates how routing tables are updated after successful queries. Section 4.2.5 describes the activities executed locally at the peers. Section 4.2.6 discusses how to improve the performance of the algorithm in the start-up phase. Finally, in Section 4.2.7, the algorithm flow is specified.

4.2.1 Data structures

The routing information is stored in two tables τ and η which are present at each peer P_i. Table τ maintains the pheromone trails. It is of size $C \times n$, where C is the size of the controlled vocabulary that defines the allowed keywords in a query and n is the number of peer P_i's outgoing links to neighbor peers. Each τ_{cu} stores the amount of pheromone corresponding to concept c dropped at the link from peer P_i to peer P_u, for each concept c and each neighbor peer P_u. The left side of Figure 4.1 shows an example. At startup, all entries in table τ are initialized with the same value $\tau_{init} = 0.009$. Initialization with a small value is necessary to prevent divisions by zero in the evaporation feature (see Section 4.2.5) and in link selection (see Section 4.2.3).

Next to table τ, each peer manages a table η that stores the link costs to its neighbor peers. This table consists of only one column (see right side of Figure 4.1). Each entry η_u is the

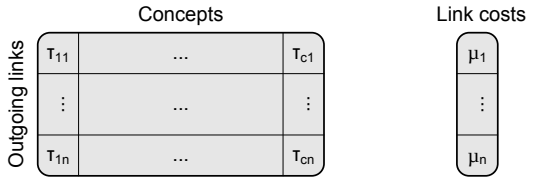

Figure 4.1: Format and size of the routing tables

inverse value $\frac{1}{lc_u}$ of the cost lc_u for sending an ant from peer P_i to peer P_u. The entries in this table are initialized with zero.

Note that these are the same data structures as used in *AntNet*. The difference is that in *AntNet* the columns of table τ refer to destination nodes, while in the SEMANT algorithm the columns of table τ refer to the keywords that can occur in a query. This implies that a different type of pheromone is used for every query keyword that can occur.

Allowing free-text search is not feasible in this setting. It would result in an unlimited number of pheromone types, and therefore in an unlimited number of columns in the routing tables.

4.2.2 Query routing

As discussed in Section 3.3.1.3, the concept of forward ants and backward ants from the *AntNet* algorithm (see Figure 3.1) is employed as the foundation of the query routing procedure. In addition, *AntNet*'s mechanism for preventing cycles is adopted. Whereas the latter can be adopted without any changes, the concept of forward ants and backward ants needs to be adapted to the application purpose of query routing. In the following, the necessary adaptations are discussed.

4.2.2.1 Queries

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

4.2.2.2 Time-to-live parameter

Next, it is necessary to define at which point in time the forward ant should stop its travel. In *AntNet*, forward ants terminate their travel through the network when they arrive at their well-defined destination peer. This behavior can not be transferred, since the forward ant's task is to find an unknown destination peer that in the worst case does not even exist. Instead, a stop point for forward ants must be defined to prevent forward ants from running infinitely if no results can be found. There are two possibilities for defining the stop point. The simplest solution is to use a time-to-live (TTL) parameter ttl_{max} like introduced in Gnutella (see [67]). Each time an ant travels one hop to reach another peer, it decrements ttl_{max} by one. The stop point is reached if $ttl_{max} = 0$.

Another possibility is to use a maximum lifetime t_{max} measured in seconds. This option is more complex, since it requires that either the clocks at every peer are synchronized, or that the ants carry a clock themselves. The benefit of this approach is that it creates an upper bound for the time it takes until the result arrives at the querying peer. If a forward ant arrives at a peer that stores results, it creates a backward ant. Consequently, if results were found and – as a consequence – backward ants were created, the upper bound for a backward ant to arrive at the querying peer is the time interval of $2 * t_{max}$. If no backward ants arrive within this time interval, the querying peer can deduce that no result could be found.

4.2.2.3 *MinResources* variation and *maxResults* variation

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

- ***MinResources* variation.** If the ants are terminated after they found the first result, the optimization goal is to use the minimum amount of network resources. Therefore, the backward ant strategy of stopping after the first result is found will be referenced to as the *minResources* variation of the SEMANT algorithm.

- ***MaxResults* variation.** In the other case, where the ants use the maximum time-to-live parameter, the ants use approximately the same amount of network resources for each query and the optimization goal is to maximize the number of results that are found for a query. In the remainder of this thesis, the strategy of using the maximum

time-to-live parameter will be referenced to as the *maxResults* variation of the SEMANT algorithm.

In practice, both of these variations are valid, because both of the optimization goals are desired at the same time. Using the *minResources* variation is of benefit for the performance of the entire network, since the algorithm saves as many network resources as possible. Using the *maxResults* variation is of benefit for the individual users of the network, since the algorithm tries to find as many results for a single query as possible.

The variations can be combined by using a weight that defines the ratio between using the *minResources* variation and using the *maxResults* variation. Based on this weight, each forward ant applies the *roulette wheel selection technique* [58] to select for one of the variations before it starts travelling.

4.2.2.4 Step-by-step description of the query routing procedure

Now the complete procedure for answering a query is laid out. Consider a query q issued at a peer P_q. The following seven steps are necessary for answering query q.

Step 1 Check the resource repository of peer P_q. If any results are found, present them to the user. If the number of results found is less than r_{max}, go to step 2. If the number of results found is greater than r_{max}, terminate the algorithm.

Step 2 Create a forward ant F_q with starting time t_{Fstart} and timeout t_{max} at peer P_q. Add the identifier of peer P_q to F_q's stack of already visited peers $s(F_q)$. Initialize the list $lc(F_q)$ that stores the link costs of all links used by F_q.

Step 3 Use the link selection procedure described in Section 4.2.3 for selecting the neighbor peer(s) P_{j_x} the forward ant F_q should choose ($x \in [1..n], n \in \mathbb{N}$). Send ant F_q to peer P_{j_1}. For every peer P_{j_x}, where $x \in [2..n]$, create a clone F_q^c of forward ant F_q and send ant F_q^c to peer P_{j_x}.

Step 4 For every forward ant F_q that arrives at a peer P_j, check if peer P_j was already visited by a clone F_q^c. If so, terminate ant F_q. Otherwise, check the resource repository of peer P_j for resources r that are results for query Q. If there are no results, continue at step 6. Otherwise, add the identifiers of all resources r to the set R.

Step 5 Generate a backward ant B_q. Pass it R, the identifier of the peer P_j that stores R, the stack of already visited peers $s(F_q)$, and the recorded link costs $lc(F_q)$. Send B_q back to the querying peer P_q using the procedure described in Section 4.2.4. In case the *minResources* variation is used, terminate the forward ant F_q. Otherwise, continue at step 6.

Step 6 Add the identifier of peer P_j to the stack of already visited peers $s(F_q)$. 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, terminate F_q.

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

The description above refers to using t_{max} as the time-to-live parameter. In case of using ttl_{max} instead (see Section 4.2.2.2), the expression in step 7 is changed to $ttl_{max} > 0$. Moreover, step 7 includes decreasing ttl_{max} by 1. Recording the starting time t_{Fstart} is not necessary in this case.

4.2.3 Link selection

Now the selection of outgoing links by the forward ants is described. In ant algorithms, this selection is made by applying a so-called transition rule. As justified in Section 3.2.1.2, the transition rule designed for the SEMANT algorithm is based on the transition rule from the *Ant Colony System* algorithm. The *Ant Colony System* transition rule consists of two strategies that supplement each other.

- In the exploiting strategy, the ant determines the quality of the links depending on the amounts of pheromone and 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_u for each neighbor peer P_u not already visited, and by applying the *roulette wheel selection technique* [58] to select one of the peers.

The strategies are described in Section 4.2.3.1 and Section 4.2.3.2. The question of which one of the two strategies to select according to its desirability in the current context of the ant is referred to as the *exploration-exploitation-dilemma* [70, 139]. The dilemma occurs not only in ant algorithms, but in reinforcement learning [127] in general. In *Ant Colony System*, the decision for one of the strategies is based on a defined weight $w_e \in [0, 1]$ which stays constant during the execution of the algorithm. Each ant individually decides for a strategy based on weight w_e and on the roulette wheel selection technique every time it has to decide for an outgoing link. The SEMANT algorithm also relies on weight w_e for strategy selection.

4.2.3.1 Exploiting strategy

In case the exploiting strategy is used, a forward ant F_q located at a certain peer P_i selects the neighbor peer P_j currently known as the most appropriate one by applying the transition rule shown in Equation 4.1.

$$j = arg\ max_{u \in U \wedge u \notin s(F_q)} \left([\tau_{cu}] \cdot [\eta_u]^{\beta} \right), \tag{4.1}$$

where τ_{cu} is the amount of pheromone for concept c on the path to peer P_u, η_u is the link cost of the path to peer P_u, β is a constant that defines the influence of link costs, U is the set of neighbor peers of peer P_i, and $s(F_q)$ is the set of peers already visited by F_q.

4.2.3.2 Exploring strategy

In case the exploring strategy is used, a forward ant F_q located at a certain peer P_i applies the transition rule shown in Equation 4.2a and Equation 4.2b. This rule is applied *for each* neighbor peer P_j in order to decide whether it should be selected. This is an adaptation of the roulette wheel selection technique: Each p_j is *separately* placed on the continuum between 0 and 1, and for each p_j a random value q is calculated for deciding if peer P_j should be selected. This mechanism allows more than one peer to be selected in order to account for the fact that there are multiple possible destination peers which contain answers for a query. To ensure that at least one peer will be selected, the algorithm falls back to the exploiting strategy in case applying the exploring strategy does result in not selecting any peer.

$$p_j = \frac{[\tau_{cj}] \cdot [\eta_j]^{\beta}}{\sum_{u \in U \wedge u \notin s(F_q)} \left([\tau_{cu}] \cdot [\eta_u]^{\beta} \right)} \tag{4.2a}$$

In Equation 4.2a, τ_{cj} is the amount of pheromone for concept c on the path to peer P_j, η_j is the link cost of the path to peer P_j, parameter β is a constant that defines the influence of link costs, U is the set of neighbor peers of peer P_i, $s(F_q)$ is the set of peers already visited by F_q, and

$$GO_j = \begin{cases} 1 \text{ if } q \leq p_j \wedge j \in U \wedge j \notin s(F_q) \\ 0 \text{ else} \end{cases} \tag{4.2b}$$

In Equation 4.2b, q is a random value, $q \in [0, 1]$, and the sum of all goodness values

$$\sum_{j \in U \wedge j \notin s(F_q)} p_j = 1.$$

If $GO_j = 1$, the forward ant F_q creates a clone F_q^c of itself and sends the clone F_q^c to peer P_j.

4.2.4 Routing table updates

A description of the mechanism used by the backward ants for updating the routing tables follows. A backward ant is created at a certain peer P_r storing a set of results R. Each backward ant B_q is in possession of a copy of the stack data recorded by its corresponding forward ant F_q. This stack contains all visited peers $s(F_q)$ and all recorded link costs $lc(F_q)$.

Depending on whether t_{max} or ttl_{max} is used for either measuring the time or measuring the number of hops consumed by the forward ant, the backward ant B_q either calculates the sum of all entries in $lc(F_q)$ to get the total link costs t_{qr} for the path from the querying peer P_q to the answering peer P_r, or it calculates the number of hops h_{qr} between peer P_q and peer P_r. After performing the calculation, the backward ant travels back hop-by-hop according to the information stored in $s(F_q)$ until it arrives at querying peer P_q. At each intermediate peer, the backward ant is responsible for two tasks:

- Firstly, it updates the link cost η_j according to the entry in $lc(F_q)$.
- Secondly, it drops pheromone by applying the pheromone trail update rule shown in Equation 4.3a to Equation 4.3c. In Equation 4.3a, the amount Z of newly added pheromone depends on the goodness of the found path. If t_{max} is used, amount Z is derived according to Equation 4.3b. If ttl_{max} is used, amount Z is derived according to Equation 4.3c.

The goodness of the found path is determined by comparing the number of resources found and the length of the path to pre-defined reference values. For the reference solution, the value for a path's total link costs is set to $\frac{1}{2} \cdot t_{max}$, and the number of resources is set to r_{max}.

$$\tau_{cj} \leftarrow \tau_{cj} + Z, \tag{4.3a}$$

where

$$Z = w_d \cdot \frac{|R|}{r_{max}} + (1 - w_d) \cdot \frac{t_{max}}{2 \cdot t_{qr}} \tag{4.3b}$$

or

$$Z = w_d \cdot \frac{|R|}{r_{max}} + (1 - w_d) \cdot \frac{ttl_{max}}{2 \cdot h_{qr}} \tag{4.3c}$$

In Equation 4.3b and Equation 4.3c, parameter w_d weights the influence of resource quantities and link costs.

4.2.5 Peer activity

Each peer performs management procedures on its local routing table. It is responsible for periodically applying an evaporation feature to the locally maintained pheromone trails. Evaporation is one of the constituents of pheromone management in ant algorithms (cf. Section 3.2). In a static scenario, the evaporation rule is useful for preventing that paths become dominant. In a dynamic setting, it can also help to remove outdated information.

$$\tau_{cu} \leftarrow (1 - \rho) \cdot \tau_{cu} \tag{4.4}$$

Each peer applies the evaporation rule shown in Equation 4.4 in predefined intervals t_e for each link to neighbor peer P_u and each concept c. The amount of pheromone that evaporates in every interval is controlled by parameter $\rho \in [0, 1]$. This rule is adopted from the evaporation part of *Ant Colony System*'s global pheromone update rule (see Section 3.2.1.1).

4.2.6 Bootstrapping

In the start-up phase, directly after the initialization of the routing tables, all pheromone trails are of equal strength. Therefore, the ants make their decisions completely randomly. This means that the performance values of the algorithm will be rather bad in the beginning. Informed decisions are made as soon as an appropriate number of ants found results, and updated the pheromone trails accordingly. After a certain time, the pheromone trails will be near to the optimum. When reaching this point, the performance values do not improve significantly any longer.

One possibility to speed up the time to reach this point is to send additional ants in the start-up phase. Since the exploring strategy already provides for selecting more than one outgoing link, sending additional ants is most useful in the exploiting strategy.

A simple criteria to detect if the algorithm is in the start-up phase is to check the amounts of pheromone on the outgoing links before link selection is performed in order to find out if all amounts for the keyword of the query are of equal size. If this is the case, one possibility would be to send an ant to every outgoing link, but this is equal to flooding. Another option would be to randomly choose two outgoing links, but initial experiments have shown that this still creates a high amount of network traffic. Instead, a parameter $p_{sendAdditional} \in [0, 1]$ is used. It defines the probability that two outgoing links are chosen in case (1) the pheromone trails on all outgoing links are equal and (2) the exploiting strategy is employed.

4.2.7 Program flow

Finally, the program flow of the SEMANT algorithm can be specified in pseudo-code. It is shown in Figure 4.2.

4.3 Simulation and Results

This section presents the experimental evaluation of the performance figures of the SEMANT algorithm as defined in Section 4.2. After describing the simulation setup in Section 4.3.1, Section 4.3.2 depicts the metrics used and discusses the rationale behind choosing them. The performance results are shown in Section 4.3.3. In Section 4.3.4, the impacts of the values chosen for the configurable parameters of SEMANT on its performance are investigated. Finally, Section 4.3.5 analyzes the influence of the network topology on the performance results.

```
t = CurrentTime;
t_end = EndTimeOfSimulation;

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

      # Concurrent activity at each peer
      in_parallel {
         if (Query Q) {
            checkLocalDocumentRepository();
            createForwardAnt(query_parameter);
         }

         foreach (ForwardAnt) {
            initializeParameters();
            while (Timeout_not_reached) {
               applyTransitionRule();
               t_a = CurrentTime;
               foreach (ForwardAnt) {
                  GoToNode(P_j);
                  t_b = CurrentTime;
                  checkLocalDocumentRepository();
                  if (DocumentsFound > 0) {
                     createBackwardAnt(stackData, P_j);
                     if (minResourceStrategy) {
                        terminate();
                     }
                  }
                  addDataToStack(P_j, t_b-t_a);
               }
            }
            terminate();
         }

         foreach (BackwardAnt) {
            calculateQuality();
            do {
               node = popStackData_Node();
               GoToNode(node);
               applyPheromoneTrailUpdateRule();
               updateListCosts(popStackData_Costs());
            } while (node <> source_node);
         }

         foreach (PeriodicalTimeInterval t_e) {
            applyEvaporationRule();
         }

      }
   }
}
```

Figure 4.2: The SEMANT algorithm in pseudo-code

4.3.1 Evaluation setup

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

The SEMANT algorithm is independent from a certain application scenario. It can be used in every situation where a group of individuals wants to share annotated resources. The application scenario chosen for this evaluation is to support cooperation between researchers in computer science by allowing them to share scientific articles. An overview of legal use cases for peer-to-peer-based search can be found in [87].

4.3.1.1 Network topology

Since the use case concerns a community of researchers, and therefore a social network, a *small world* [79] network topology is selected for the experiments in order to choose a realistic model for social networks. We use the Kleinberg small world generator provided by the JUNG framework [77], which models the network topology with a toroidal lattice. Each node has four local connections, one to each of its neighbors. In addition, each node has one long range connection to some node which is chosen randomly according to a probability proportional to $d^{-\alpha}$, where d is the number of hops between these nodes, and α is the clustering coefficient (which we set to 1). The size of the network is set to 1024 peers. As shown in [57], the underlying network topology has significant effects on the performance of the search algorithm. For this reason, the performance of the algorithm needs to be evaluated for other network topologies as well. This is the subject of Section 4.3.5.

The SEMANT algorithm provides support for taking different link costs into account. In a real-world setting, each link between two peers has a certain latency, a certain bandwidth, and a certain throughput (see, e.g., [81]) depending on the hardware of the connection. Unfortunately, there are two reasons that make the evaluation of this feature cumbersome. Firstly, there is no appropriate test data composed of a real-world network topology together with defined latency/bandwidth/throughput properties available. Secondly, even if such data would be available, there is no other approach to search in peer-to-peer networks that considers different link costs. This makes it impossible to compare the performance figures to reference values. For these reasons, it was decided not to evaluate this feature. The link costs for all links are set to 1.

4.3.1.2 Content distribution

The content is modeled with the ACM Computing Classification System (ACM CCS, [11]) as the underlying meta-data vocabulary. The ACM CCS is a taxonomy consisting of 1473

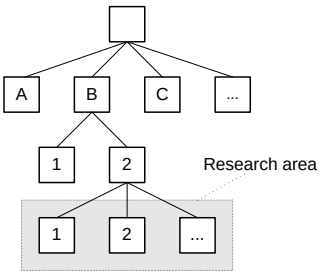

Figure 4.3: Research areas: Third-level topics and their leaf topics

topics for the domain of scientific literature in the field of computer science. Each resource is annotated with one keyword, i.e., each resource is an instance of one leaf topic from the taxonomy. In total, the ACM CCS taxonomy contains 910 leaf topics. In order to represent each research topic equally, the same number of resources is created for each leaf topic. Each peer stores on average 30 resources, and the total number of resources in the network is 30940 resources. This means that there are approximately 34 resources per topic. In real-world settings, the resources in a peer-to-peer network are not randomly distributed among the peers (see, e.g., [148]). Instead, patterns in the data can be observed because the interests of the peers are not uniformly distributed. Each peer focuses on one or a few topics it has special interests in. In the application scenario of researchers, the special interests are certain research areas. Modeling research areas with the leaf topics from the taxonomy would lead to very narrow interests. Therefore, the sub-topics of a third-level topic of the taxonomy (see Figure 4.3) are facilitated for modeling a research area. There are 177 third-level topics in the taxonomy. The assumption is that each peer is an expert on a certain research area and for this reason, on average 60% of the resources in his or her repository are instances of one particular research area. On average, another 20% of the resources are related to another research area. The remaining 20% are instances of random leaf topics.

Some remarks need to be made about the content distribution described above. Firstly, there is an additional assumption that the content distribution does not change during the simulation. In a real network, it would change since the peers retrieve resources from each other. This behavior is not integrated in the simulation. Secondly, the hierarchical relationships between the topics in the taxonomy are used only for modeling the research areas, but not for query routing. The exploitation of these relationships to improve the routing performance is subject of Chapter 5.

4.3.1.3 Query distribution

A uniform query distribution is used in the following. For uniformly distributing the queries within the network, a ticker clock at each peer is employed. The probability that

a peer issues a query within one time unit is set to 0.1. Each query consists of a randomly selected leaf topic from the ACM Computing Classification System taxonomy.

4.3.2 Metrics

Although many scientific articles about peer-to-peer systems use the metrics of precision and recall [14] known from information retrieval for their evaluation, the drawback of relying on these metrics is that they do not include the traffic created in the network. The aim of this work is to create an algorithm for query routing which has an optimal ratio between network traffic and quantity of results. Hence, 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 resources found for each query within a given period of time.
- *Efficiency* is the ratio of resource usage to hit rate. Dividing the number of links traveled by the number of resources found gives *the average number of links traveled to find one resource* as a result, which is the most practical metric.

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

4.3.3 Performance evaluation

This section describes the experimental evaluations conducted. Section 4.3.3.1 documents the parameter settings for the SEMANT algorithm. In Section 4.3.3.2, the SEMANT algorithm is compared against the *k-random walker* approach. In Section 4.3.3.3, it is shown that the simulation environment is implemented and set up correctly in such a way that it produces repeatable results. In Section 4.3.3.4, a comparison of the SEMANT *minResources* variation against the *maxResults* variation is performed. In Section 4.3.3.5, the bootstrapping mechanism is evaluated.

4.3.3.1 Parameter settings

The left side of Table 4.1 shows a listing of the configurable parameters of the SEMANT algorithm. The values used for these parameters at runtime have a considerable impact on the performance of the system, and choosing the most appropriate values is a non-trivial task. The parameter values chosen for the experiments described in the subsequent sections were

ρ	evaporation factor	0.07
ttl_{max}	timeout of forward ants	25
w_e	weight of exploiting vs. exploring strategy	0.85
r_{max}	maximum number of resources	10
w_d	weight of resource quantity vs. link costs	0.5
β	weight of link costs	1
$p_{sendAdditional}$	bootstrapping parameter	0.0

Table 4.1: Parameter values chosen for the SEMANT algorithm

defined after conducting initial experiments. These are not the optimal, but approximated values. The selected settings are shown in the right-most column of Table 4.1. Note that ttl_{max} (and not t_{max}) is used for defining the stop point (see Section 4.2.2.2). The effect that changing the values has on the overall performance of the algorithm is evaluated in Section 4.3.4.

4.3.3.2 Comparison of the *maxResults* variation against the *k-random walker* algorithm

Now the performance of the SEMANT algorithm is compared against that of the well-known *k-random walker* approach [90], which will be described below. The *maxResults* variation (see Section 4.2.2.3) and the parameter values shown in Table 4.1 are used. The time horizon for the experiment is set to 10000 time units. In order to provide for comparison fairness,

- both algorithms use the same setup as described in Section 4.3.1,
- the time-to-live parameters are set to an equal value, and
- the parameter settings for the algorithms are set in such a way that – in total – the agents of both algorithms are allowed to travel a comparable number of links.

Consequently, the parameters for the *k-random walker* algorithm are set to $k = 2$ and $TTL = 25$. This means that two walkers travel for 25 hops. A walker is similar to a forward ant, except from the fact that it does not rely on routing tables, but makes a random decision about which outgoing link to choose in the link selection procedure. If resources are found, the walker sends an information message back to the querying peer (similar to a backward ant), and walks on. After 25 hops, the walkers are terminated.

The results of the resource usage comparison between the SEMANT algorithm and the *k-random walker* algorithm are shown in Figure 4.4. Note that these numbers include both forward and backward ants/agents. The hit rate measurements are shown in Figure 4.5. 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

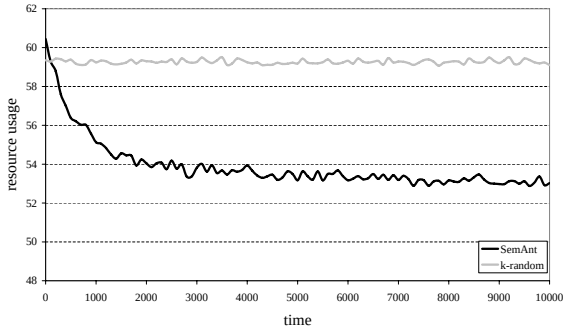

Figure 4.4: Resource usage comparison of the *k-random walker* algorithm and the SEMANT algorithm

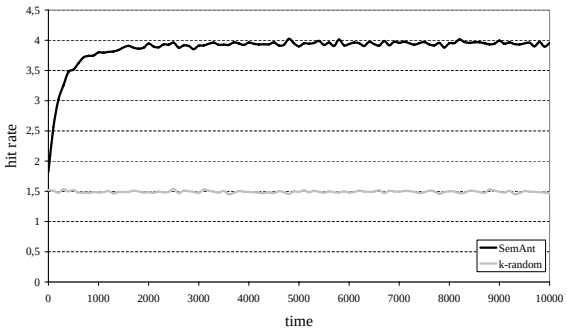

Figure 4.5: Hit rate comparison of the *k-random walker* algorithm and the SEMANT algorithm

optimization. On average, the agents travel 59.27 links for answering a query. The average number of results found for a query is 1.49 resources.

On the contrary, the SEMANT algorithm 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 ants have to travel 60.44 links in order to find 1.82 resources per query. After 1000 time units, these numbers improve to 55.13 links traveled for retrieving 3.8 resources 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 resources 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 resources 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 resources per query.

Note that the performance figures depicted in Figures 4.4 to 4.5 (and also all other results described in this section) are derived by building average values from all queries that occurred within 100 time units for the respective metric.

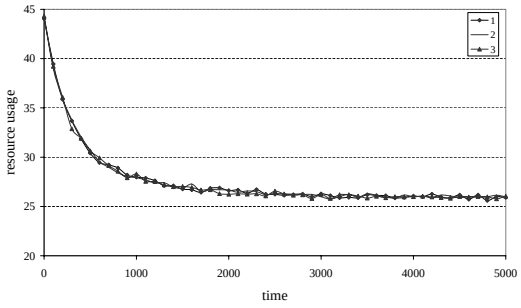

Figure 4.6: Resource usage comparison of 3 test runs

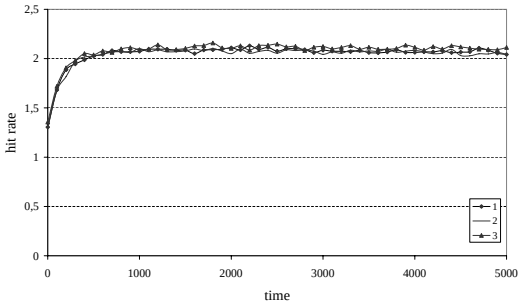

Figure 4.7: Hit rate comparison of 3 test runs

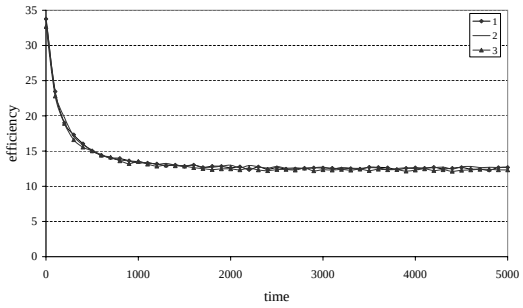

Figure 4.8: Efficiency comparison of 3 test runs

4.3.3.3 Correctness of the implementation

The design and implementation of the simulation environment used for the experiments is described in [104]. It is necessary to test if the simulation environment is implemented correctly, that is, to show that the results produced are deterministic and repeatable. In order to carry this out, the experiment described in Section 4.3.3.2 is repeated three times (using the *minResources* variation), and for each of these recurrences a newly generated network with the same content distribution parameters as described in Section 4.3.1 is used. This means that there are three different networks having the same properties. Since Figure 4.4 and Figure 4.5 already revealed that the algorithm's results do not significantly change between time unit 5000 and time unit 10000, in this experiment (and also in the subsequent ones) it is sufficient to use a time horizon of 5000 time units.

After that, the results are compared. The Figures 4.6 to 4.8 show the resource usage, hit rate, and efficiency figures for the three recurrences. What can be seen is that there are no significant differences between these results. Obviously, this is not a formal proof for the correctness of the simulation environment. However, it is sufficient for concluding that the evaluation method chosen is valid and that the results are reproducible.

4.3.3.4 Comparison of the *minResources* variation against the *maxResults* variation

Now the performance of the *minResources* variation of the SEMANT algorithm is compared against the performance of the *maxResults* variation. The parameter values shown in Table 4.1 are used. The results of the comparison are shown in Figures 4.9 to 4.12. The results for the *k-random walker* approach are included as a reference. The figures clearly indicate that, depending on the variation used, two different optimization problems are solved. In case of the *maxResults* variation, the hit rate is maximized. Figure 4.9 shows that after 5000 time units, on average 3.98 resources per query are found compared to 1.77 resources per query at time unit 0. However, the resource usage decreases only slightly. After 5000 time units, on average 53 hops are necessary for query routing (Figure 4.11) compared to on average 59 hops at time unit 0.

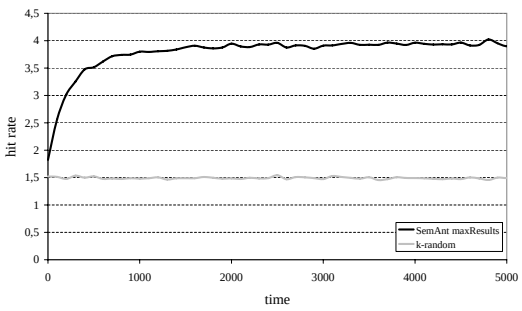

Figure 4.9: Hit rate evaluation of the *maxResults* variation of the SEMANT algorithm

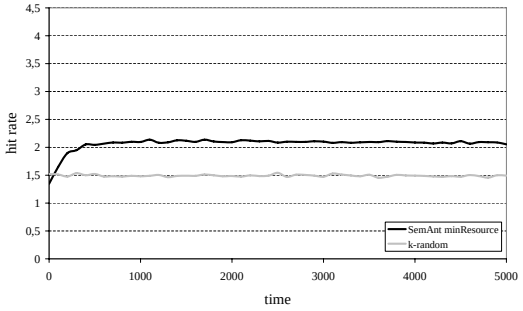

Figure 4.10: Hit rate evaluation of the *minResource* variation of the SEMANT algorithm

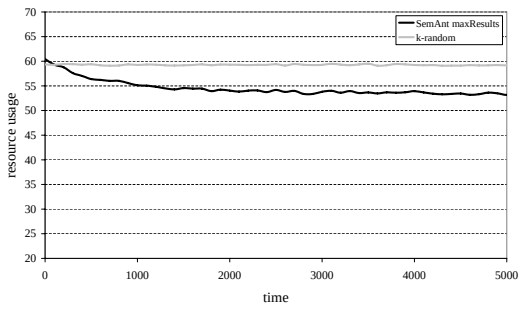

Figure 4.11: Resource usage evaluation of the *maxResults* variation of SEMANT

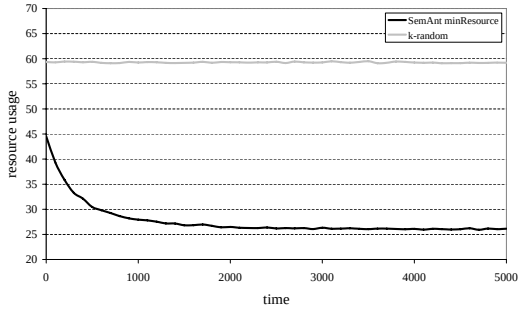

Figure 4.12: Resource usage evaluation of the *minResource* variation of SEMANT

On the other hand, employing the *minResources* variation minimizes the resource usage of the ants. In this case, after 5000 time units only 26 hops are needed on average for query routing (Figure 4.12), but on average only 2.1 resources per query are found (Figure 4.10). Since hit rate and resource usage depend on each other, the best way of comparing the vari-

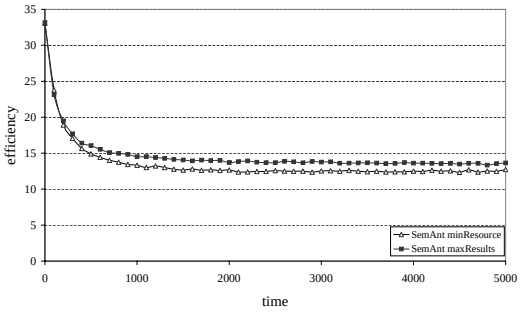

Figure 4.13: Efficiency of *minResources* variation and *maxResults* variation

ations against each other is to use the metric of efficiency. Figure 4.13 shows a comparison of the efficiency values for the two variations. It can be seen that, as soon as the converged phase is reached (after 500 time units), the *minResources* variation needs less hops to retrieve one resource than the *maxResults* variation. These results show that for the overall performance of the system, the *minResources* variation has a higher benefit in terms of a lower number of hops traveled for retrieving one resource. The difference after 5000 time units is approximately 0.9 hops per query (13.6 hops on average/12.7 hops on average). The reason for that is as follows. Recall that ants using the *maxResults* variation go on after they found the first appropriate peer. However, after finding it they tend to stay in the neighborhood of this peer, since the pheromone trails indicate that there is an appropriate peer nearby. In this case the trails are misleading, because forward ants can not visit the same peer twice. However, the performance difference between the variations is rather marginal. Therefore, employing the *maxResults* variation is reasonable as well, because in this case the number of resources found for a single query is higher. This is of benefit to the individual users of the network.

4.3.3.5 Evaluation of the bootstrapping mechanism

As the next step, the bootstrapping mechanism described in Section 4.2.6 is evaluated. The *minResources* variation together with the parameter values shown in Table 4.1 is used, and four test runs with parameter $p_{sendAdditional} = 0$, $p_{sendAdditional} = 0.1$, $p_{sendAdditional} = 0.2$, and $p_{sendAdditional} = 0.3$ are executed.

As can be seen in Figure 4.14, bootstrapping causes a high amount of network traffic in the start-up phase (between time unit 0 and time unit 100). The reason for that is that in this interval of time, all pheromone amounts are equal. Between time unit 100 and time unit 300, most of the pheromone trails are already initialized (that is, the amounts of pheromone are not equal any longer). Consequently, only slightly more network resources are consumed if $p_{sendAdditional} \neq 0$ than if otherwise. After time unit 300, and the bootstrapping mechanism

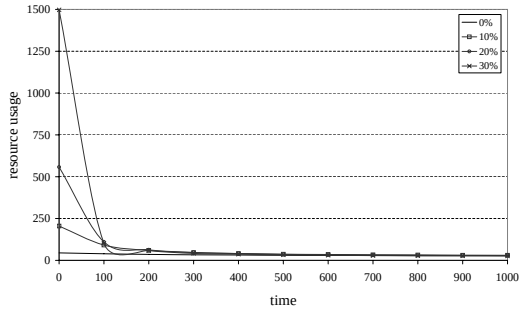

Figure 4.14: Resource usage evaluation of the bootstrapping mechanism

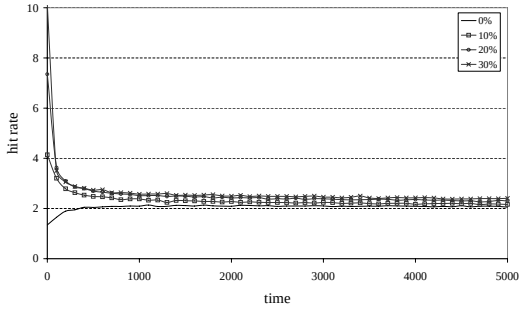

Figure 4.15: Hit rate evaluation of the bootstrapping mechanism

does not create any additional traffic. Figure 4.14 and Figure 4.16 reveal that the parameter $p_{sendAdditional}$ needs to be dealt with very carefully, because the amount of traffic in the network in the start-up phase is exponentially proportional to the value set for it. Using a value $p_{sendAdditional} > 0.3$ is not feasible.

Figure 4.15 shows that the higher the value for parameter $p_{sendAdditional}$, the higher the number of results in the start-up phase. But it can be observed from Figure 4.16 that these benefits are achieved in an inefficient way because the average number of hops necessary for answering a query is high.

However, as shown in Figure 4.17, using the bootstrapping mechanism for initializing the pheromone trails has positive effects on the performance which are permanent and still present after 5000 time units. Whereas for $p_{sendAdditional} = 0$ the efficiency result is 12.7 hops per query, for $p_{sendAdditional} = 0.1$ this value lowers to 12 hops, for $p_{sendAdditional} = 0.2$ to 10.9 hops, and for $p_{sendAdditional} = 0.3$ to 9.8 hops per query. Therefore, using the bootstrapping mechanism is of benefit.

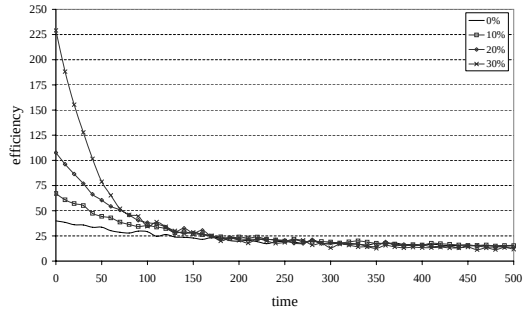

Figure 4.16: Efficiency evaluation of the bootstrapping mechanism (time unit 0 to 500)

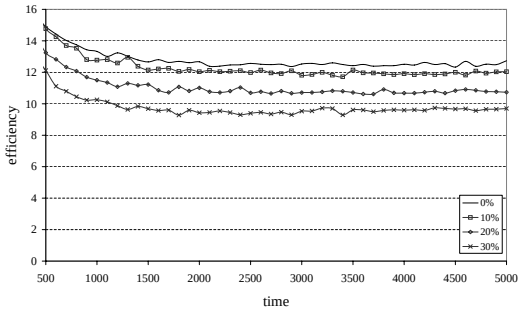

Figure 4.17: Efficiency evaluation of the bootstrapping mechanism (time unit 500 to 5000)

4.3.4 Influence of parameter settings on performance

One of the challenges when employing ant algorithms is that they include various configurable parameters for which appropriate value settings must be found. Ideally, it would be possible to automatically determine these parameter values at runtime, but this is not within the scope of this work. In the experiments conducted so far, fixed values were used for the configurable parameters of the SEMANT algorithm. In the following, the influence of different settings for the parameters on the performance of the algorithm is evaluated. Section 4.3.4.1 investigates the influence of the time-to-live parameter, Section 4.3.4.2 that of the weight between exploiting and exploring strategy, and Section 4.3.4.3 that of the evaporation factor. The other parameters of the SEMANT algorithm (r_{max}, w_d, β) do not have a significant effect on the performance values, and are therefore excluded from the evaluation.

4.3.4.1 Influence of time-to-live parameter ttl_{max}

The influence of the time-to-live parameter ttl_{max} is evaluated by acquiring the performance results for $ttl_{max} \in [5, 10, 15, 20, 25, 30]$ and comparing them. Since the assumption is that there is a dependency between variation and time-to-live parameter, the results for both the *minResources* variation and the *maxResults* variation are retrieved. The results of the evaluation are shown in Figure 4.18 to Figure 4.23.

The hit rate results for both variations (see Figure 4.18 and Figure 4.19) show that values lower than $ttl_{max} = 15$ have a negative impact on performance. The problem in these cases is that the forward ants are terminated before they found an appropriate peer. This effect itself is independent from the experimental setup, but the bound $ttl_{max} < 15$ is specific for the setup used in this experiment. The value for the bound is dependent on the network size and in the content distribution in the network.

Figure 4.18 reveals that in the *minResources* variation, the main effect of employing a higher value for parameter ttl_{max} is a faster convergence of the algorithm. For appropri-

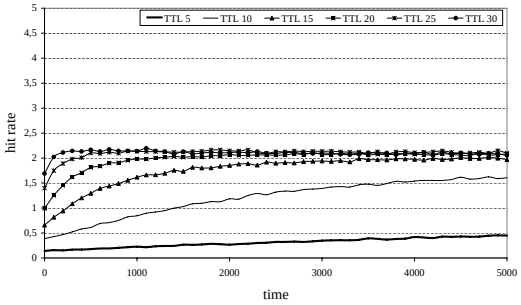

Figure 4.18: Hit rate evaluation of the influence of time-to-live parameter ttl_{max} in the *minResources* variation

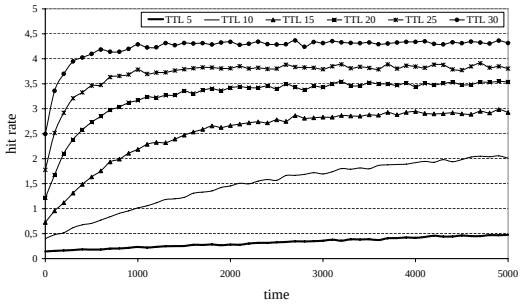

Figure 4.19: Hit rate evaluation of the influence of time-to-live parameter ttl_{max} in the *maxResults* variation

ate values of ttl_{max}, i.e., $ttl_{max} \in [15, 20, 25, 30]$, the differences concerning the absolute number of results are marginal after the converged phase is reached. The average values after 5000 time units are 2.05 for $ttl_{max} = 30$, 2.09 for $ttl_{max} = 25$, 2.08 for $ttl_{max} = 20$, and 1.97 for $ttl_{max} = 15$.

On the contrary, when employing the *maxResults* variation (see Figure 4.19), the absolute number of results increases significantly depending on the value used for parameter ttl_{max}. The average values after 5000 time units are 4.32 for $ttl_{max} = 30$, 3.80 for $ttl_{max} = 25$, 3.54 for $ttl_{max} = 20$, and 2.93 for $ttl_{max} = 15$.

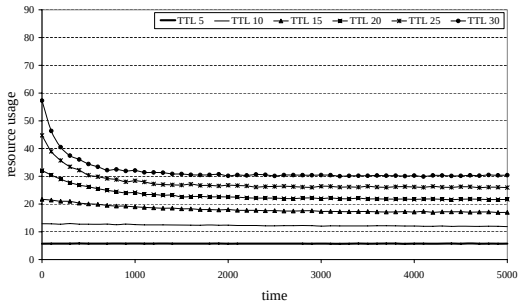

Figure 4.20: Resource usage evaluation of the influence of time-to-live parameter ttl_{max} in the *minResources* variation

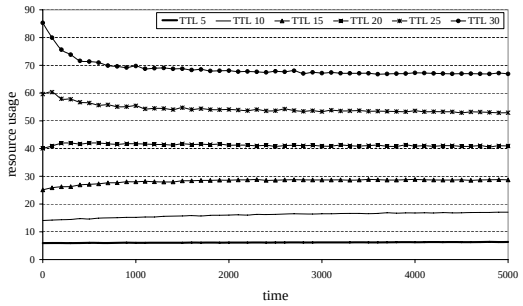

Figure 4.21: Resource usage evaluation of the influence of time-to-live parameter ttl_{max} in the *maxResults* variation

Figure 4.20 and Figure 4.21 show that the resource usage is directly dependent on the value used for parameter ttl_{max} for both variations of the algorithm. In the *minResources* variation (see Figure 4.20), the resource usage is *exponentially* dependent on the value used for parameter ttl_{max} between time unit 0 and time unit 1000 (average values at time unit 0: 30.4 for $ttl_{max} = 30$, 25.9 for $ttl_{max} = 25$, 21.7 for $ttl_{max} = 20$, 17.0 for $ttl_{max} = 15$) and decreases greatly within this time period. After the converged phase is reached (at time unit 1000), the resource usage is proportionally dependent on parameter ttl_{max}.

variation	5	10	15	20	25	30	mean	std. deviation
minResources	21.09	12.28	11.39	12.37	13.72	15.64	14.41	3.59
maxResults	21.73	12.91	12.51	13.49	15.33	16.85	15.47	3.48

Table 4.2: Average efficiency of the SEMANT algorithm when varying the value used for parameter ttl_{max}

When comparing the variations to each other, the resource usage is much higher in the *maxResults* variation (see Figure 4.21). This is obvious, since the forward ants do not stop after the first result in this case. In the *maxResults* variation, the resource usage in the start-up phase of the algorithm is very much biased by the high increase of the hit rate at the same time (see Figure 4.19). Therefore, interpreting this result is easier when utilizing the efficiency metric (see Figure 4.23).

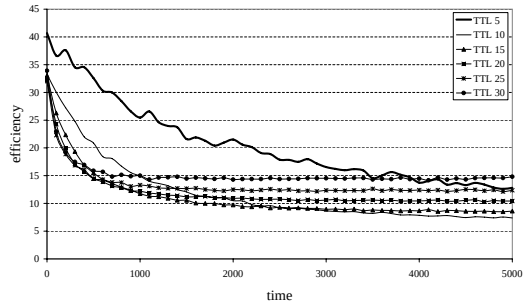

Figure 4.22: Efficiency evaluation of the influence of time-to-live parameter ttl_{max} in the *minResources* variation

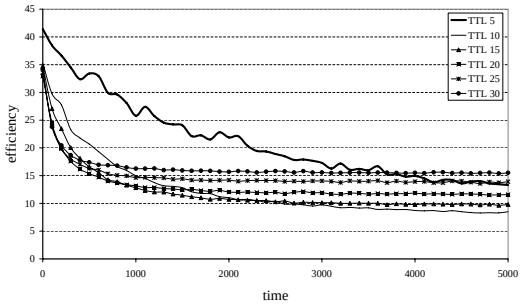

Figure 4.23: Efficiency evaluation of the influence of time-to-live parameter ttl_{max} in the *maxResults* variation

Figure 4.22 and Figure 4.23 show that, unlike it was assumed, there is no additional dependency between the value set for parameter ttl_{max} and the variation of the algorithm used. The only difference between the variations is the minor performance benefit of the *minResources* variation as already discussed in Section 4.3.3.4, which is observable for every setting of parameter ttl_{max} (see Table 4.2). It can seen in Figure 4.22 and Figure 4.23 that the value set for parameter ttl_{max} has a significant effect on the performance of the algorithm. In both variations, the best performance can be reached when setting $ttl_{max} = 15$. In this case, after 5000 time units on average 8.62 hops are necessary for retrieving one result in the *minResources* variation and on average 9.82 hops are necessary for retrieving one result in the *maxResults* variation. Again, these results are dependent on the network size and the content distribution in the network. According to which setup is used, there will be a different optimal value for parameter ttl_{max}.

Setting the value for parameter ttl_{max} higher than the optimal one has the effect that performance is lower. The average values after 5000 time units for the *minResources* variation are 10.42 for $ttl_{max} = 20$, 12.37 for $ttl_{max} = 25$, and 14.82 for $ttl_{max} = 30$. The average

values after 5000 time units for the *maxResults* variation are 11.58 for $ttl_{max} = 20$, 13.91 for $ttl_{max} = 25$, and 15.51 for $ttl_{max} = 30$.

Setting the value for parameter ttl_{max} lower than the optimal one results in low hit rates and is therefore not desirable. As discussed above, in case of $ttl_{max} = 5$, the reason for that is that the forward ants are terminated before they found an appropriate peer. An interesting result is that, in terms of efficiency, the case of $ttl_{max} = 10$ is actually the most efficient after 5000 time units. However, the algorithm does not improve the performance in the start-up phase as much as for settings where $ttl_{max} \geq 15$, and the performance in terms of hit rate is very low.

4.3.4.2 Influence of weight w_e

As discussed in Section 4.2.3, weight $w_e \in [0, 1]$ is a parameter that influences a basic decision each ant has to make before selecting an outgoing link. It can either

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

The exploitation-exploration dilemma has been discussed in several contexts, such as in the case of foraging bees [103], in economic systems [109], in software product development [71], and others. The exploitation-exploration dilemma is now addressed for the SEMANT algorithm by acquiring the simulation results for several different settings for parameter w_e. Since in the experiment described in Section 4.3.3.4 it turned out to perform better, the *minResources* variation is employed. In addition, parameter ttl_{max} is set to 15 according to the results gained in Section 4.3.4.1. For the parameters other than w_e and ttl_{max}, the settings given in Table 4.1 are used.

The results of the comparison are given in Figure 4.24 to Figure 4.26. Figure 4.24 shows that in the start-up phase, the resource usage is proportional to the ratio between exploring and exploiting strategy. The more forward ants employ the exploring strategy, the higher the amount of messages in the network. Consequently, for the overall load of the system it is better to employ a low rate of exploring forward ants. However, also a high rate, e.g., $w_e = 0.05$, is feasible. Although in this case the resource usage at time unit 0 is much higher (168.4 hops on average) than in case of a low rate (e.g., 16.7 hops on average for $w_e = 0.95$), still not even half as many resources are consumed as when using broadcast with a time-to-live parameter of 4.

The hit rate is dependent on parameter w_e not only in the start-up phase, but also in the converged phase (Figure 4.25). The more exploring forward ants in the network, the better the pheromone trails and, consequently, the higher the hit rate. The best result can

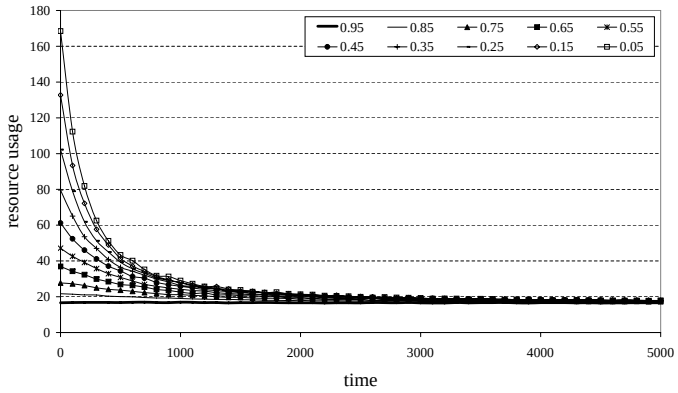

Figure 4.24: Resource usage of the SEMANT algorithm using the *minResources* variation when varying the value used for parameter w_e. Between time unit 0 and time unit 1500, resource usage is dependent on parameter w_e.

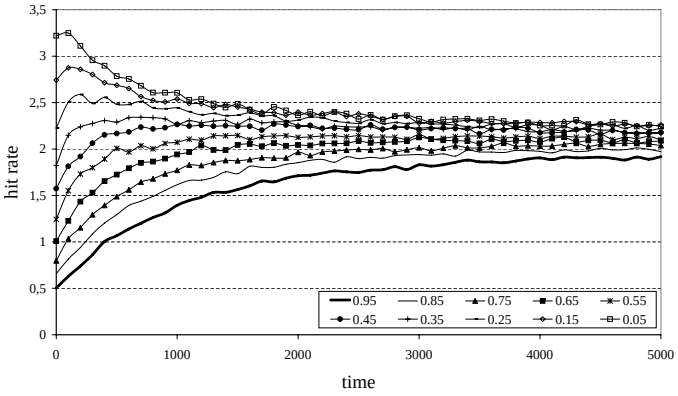

Figure 4.25: Hit rate of the SEMANT algorithm using the *minResources* variation when varying the value used for parameter w_e. The curves converge to the same limit.

be reached for setting $w_e = 0.05$. After 5000 time units, on average 2.24 resources are found in this case. The worst possible result (1.92 resources on average after 5000 time units) is obtained for setting $w_e = 0.95$. However, the difference in absolute numbers is marginal and the curves converge to the same limit over time.

Combining the two metrics shows that, independently from the ratio between exploring and exploiting strategy, the overall efficiency in terms of number of hops necessary for answering a query is similar for every setting of parameter w_e. Although a higher rate of exploring ants gives slightly less efficient results in the start-up phase, at the same time it moderately improves the performance in the converged phase. Figure 4.26 highlights this by depicting the efficiency results for $w_e = 0.05$ and $w_e = 0.95$ plotted in log-log scale. Table 4.3, showing the average efficiency over the complete time span for every setting of

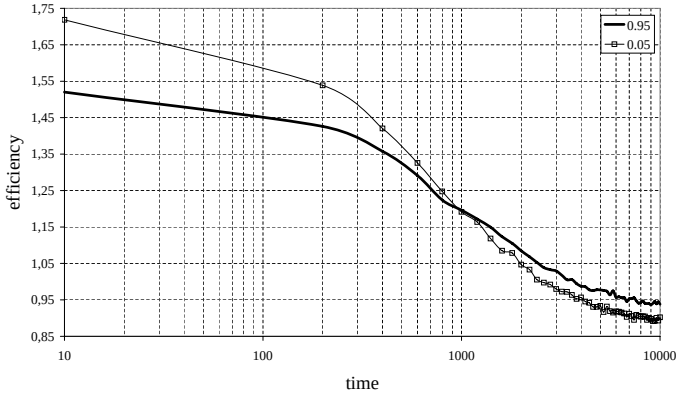

Figure 4.26: Log-log plot of the efficiency of the SEMANT algorithm using the *minResources* variation for parameter $w_e \in [0.05, 0.95]$. A higher rate of exploring ants results in slightly less efficient results in the start-up phase, but improves the performance in the converged phase. The curves for the other settings of parameter w_e lie in between those shown and are omitted for clarity. Note that the time axis is stretched by factor 2.

0.05	0.15	0.25	0.35	0.45	mean
11.43	11.31	11.33	11.45	11.25	11.38619209
0.55	0.65	0.75	0.85	0.95	standard deviation
11.35	11.37	11.42	11.38	11.57	0.087454343

Table 4.3: Average efficiency of the SEMANT algorithm using the *minResources* variation when varying the value used for parameter w_e

parameter w_e, reveals that those two effects are in balance with each other. The mean of all values is 11.39 hops per query, and the low standard deviation of 0.087 indicates that there are no significant differences in average efficiency when varying the ratio between exploring and exploiting strategy. The reason for this effect lies in the design of the exploring strategy. It can be explained by the fact that, although the forward ants choose outgoing links that are not currently known as the best ones, they still make this decision in proportion to their desirability.

4.3.4.3 Influence of evaporation factor ρ

In order to evaluate the influence of the value set for the evaporation factor ρ (see Section 4.2.5) on the performance of the algorithm, the results for $\rho \in [0.01, 0.03, 0.05, 0.07, 0.09]$ are acquired and compared. The *minResources* variation of the SEMANT algorithm is employed. Parameter ttl_{max} is set to 15. For the parameters other than ρ and ttl_{max}, the settings shown in Table 4.1 are used. Table 4.4, Figure 4.27, and Figure 4.28 show the results of the comparison. Table 4.4 shows that the performance values for the different settings of parameter ρ are very similar. The standard deviation of the average number of hops neces-

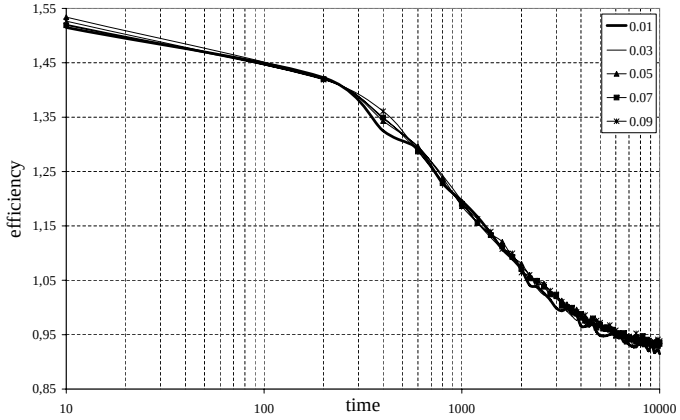

Figure 4.27: Efficiency of the SEMANT algorithm using the *minResources* variation when varying the value used for parameter ρ. Note that the time axis is stretched by factor 2.

sary for retrieving one result over the time span of 5000 time units is only 0.12. This means that the impact of the value used for the evaporation factor ρ on the performance of the algorithm is very small. For this reason, Figure 4.27 and Figure 4.28, which depict the results for the efficiency metric over time, are shown as log-log plots in order to highlight the minor

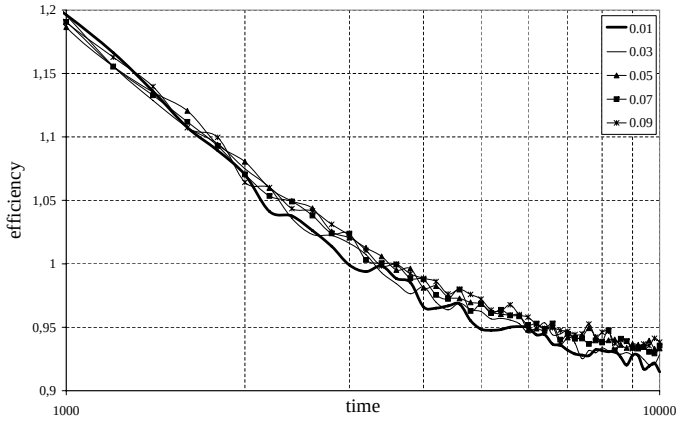

Figure 4.28: Zoom of the efficiency results between time unit 500 and time unit 5000 of the SEMANT algorithm using the *minResources* variation when varying the value used for parameter ρ. Note that the time axis is stretched by factor 2.

0.01	0.03	0.05	0.07	0.09	mean	standard deviation
11.18	11.30	11.44	11.39	11.47	11.35319617	0.116934088

Table 4.4: Average efficiency of the SEMANT algorithm using the *minResources* variation varying the value used for parameter ρ

differences between the curves. Figure 4.28 is a zoom of the behavior of the curves between time unit 500 and time unit 5000. It can be seen that the evaporation feature has a negative effect on the performance of the SEMANT algorithm. The best performance can be reached when setting parameter $\rho = 0.01$. The reason for that is that the content distribution in the network is static and no peers are leaving and joining during the simulation. Consequently, the information collected by the ants does not become outdated and therefore should not be removed by evaporation.

4.3.5 Influence of network topology on performance

The topology of the underlying network has a high impact on the performance of the search algorithm. The small world network (as described by Kleinberg [79] and Watts et al. [136]) with 1024 peers and 5120 links that was used in the experiments conducted in the previous sections is now used as a reference to compare the performance of the algorithm against the one that can be reached in a power-law network [7] with 1024 peers and 1000, 2000, 4000, 6000, 8000, and 10000 links between the peers. The *minResources* variation of the SEMANT algorithm is employed. Parameter ttl_{max} is set to 15, and all other parameters are set as shown in Table 4.1. The results of the comparison are depicted in Figure 4.29 to Figure 4.31. They reveal that the performance of SEMANT for the power-law network is better than in the small-world network, except from the case of 1000 links. However, this case can be neglected because the number of links is too low and, consequently, many of the nodes are of degree 0.

In case of 2000 links, the hit rate (see Figure 4.29) is slightly worse than in the small-world network with 5120 links, but at the same time the resource usage is significantly lower (see Figure 4.30). This leads to SEMANT's performance in the power-law network with 2000 links being more efficient (see Figure 4.31) than in the small-world network.

For 4000, 6000, 8000, and 10000 links, the hit rate is higher than in the small-world network (see Figure 4.29), but unaffected by the number of links (that is, nearly the same no

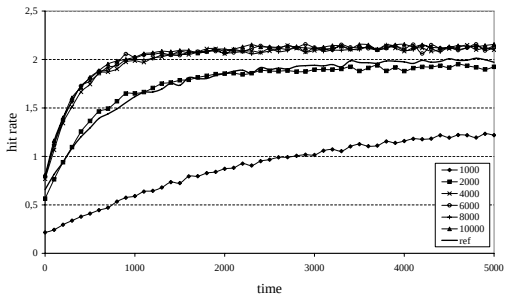

Figure 4.29: Hit rate comparison when using different network topologies

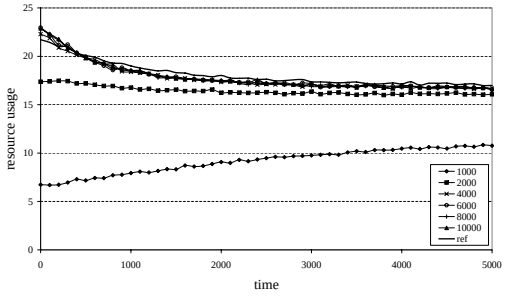

Figure 4.30: Resource usage comparison when using different network topologies

matter how many links are present in the network). The same behavior can be observed for resource usage (see Figure 4.30) and efficiency (see Figure 4.31). The fact that the hit rate is the same for all these cases and does not improve in proportion to the number of links indicates that the SEMANT algorithm can not take advantage of the additional connectivity in the network that should improve the search. The reason for that can be explained by the fact that SEMANT's mechanism of selecting outgoing links according to a probability distribution is the less effective the more outgoing links exist. In a power-law network, there are a few highly connected links that act as information hubs for the many nodes with low degree they are connected to. Since the SEMANT algorithm is the less effective the higher the degree of the node, it is not advisable for use in power-law networks.

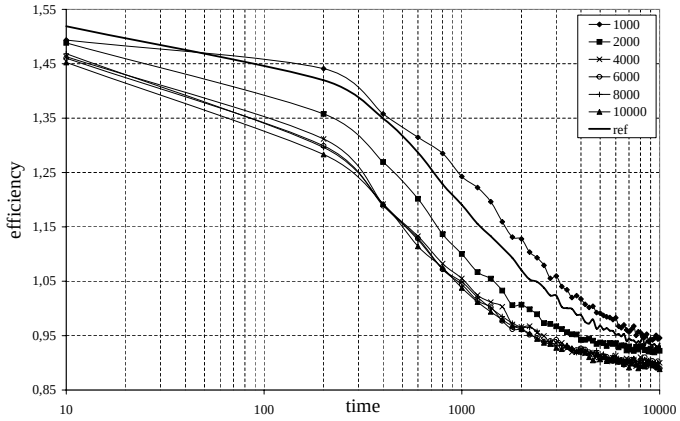

Figure 4.31: Efficiency of the SEMANT algorithm using the *minResources* variation when using different network topologies. Note that the time axis is stretched by factor 2.

4.4 Related work

This section reviews related work on ant algorithms in peer-to-peer networks. After presenting the directly related work that also focuses on peer-to-peer search in Section 4.4.1, Section 4.4.2 takes a broader view and presents an (incomplete) overview of ant-based approaches to peer-to-peer tasks other than search.

4.4.1 Ant algorithms for search in peer-to-peer networks

The most well-known practical approach towards employing ant algorithms in peer-to-peer networks is the Java-based open source framework *Anthill* [13] for the design, implementation, and evaluation of ant algorithms in peer-to-peer networks. An *Anthill* system is an overlay network of interconnected nests which correspond to peers. Nests provide services like document storage, routing table management, topology management, and generation of ants upon user requests. In addition, 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 them. The main difference between *Anthill* and SEMANT is that the *Anthill* project focuses on the development of the framework. It does not specify ant algorithms. This task is left to the users of the framework according to the application scenario. However, there are two applications based on the *Anthill* framework which were built by the *Anthill* developers for demonstration and evaluation purposes.

Gnutant [13] is a file-sharing application. In this application, each file is identified by an unique file identifier and associated with meta-data comprised of textual keywords. The query routing algorithm is based on the ant metaphor, but not on the principles defined by the Ant Colony Optimization meta-heuristic. Three different types of ants are responsible (1) for constructing a distributed index that contains URLs pointing to shared files and (2) for managing routing tables. If a user adds a new file to a nest, one *InsertAnt* is generated for each keyword of the file. InsertAnts propagate the presence of new files to the network by updating the distributed index. Gnutant utilizes hashed keyword routing based on the Secure Hash Algorithm (SHA). Each index entry contains the hash value of a keyword together with a set of nests that are likely to store files associated with the given keyword. *SearchAnts* are generated upon user queries and exploit the information stored in the routing tables in order to find files that match the queries' keywords. If no index entry that exactly matches the query's hash value exists, the SearchAnt selects the hash value that most closely matches the hash value of the query. If a SearchAnt localizes an appropriate file, it generates a *ReplyAnt* that immediately returns to the source nest and informs the user about the result of his or her query. The SearchAnt itself continues traveling until it reaches a defined time-to-live (TTL) parameter, that is, the maximum number of hops the ant is allowed to move. Once the TTL parameter is reached, it returns to its source nest via the same path it traveled before and updates both routing tables and distributed index.

The second application that was built using *Anthill* is called Messor [96]. It is not intended for search, but for load-balancing in peer-to-peer networks based on the necrophoric behavior of ants.

Schelthout et al. [119] evaluate whether the principle of synthetic pheromone can be employed for coordination in distributed agent-oriented environments. Their framework is based on the idea of objectspaces known from concurrent computing. Similar to *Gnutant*, Schelthout et al. create pheromone trails for each query and allow agents to follow trails that represent only one out of the multiple keywords in the query. Evaporation is used and the evaporation factor is set to 0.1. The test environment is too small (200 peers, one kind of resource) to be useful, but the experimental results show that pheromone trails enhance the performance of the network. The hit rate increases with the number of queries sent to the network (32% hit rate after 400 queries, 53% hit rate after 700 queries).

In addition to the directly related work, there are two areas of distantly related work. Firstly, there is work addressing the applicability of biological processes other than ant-based methods to search in peer-to-peer networks. For example, Babaoglu et al. [12] apply the principles of proliferation to search in unstructured overlay networks. Secondly, there are approaches to peer-to-peer which are inspired by stigmergy, but do not derive from the Ant Colony Optimization meta-heuristic. For instance, Handt [64] tackles the problem of search in peer-to-peer networks by inverting it. Instead of peers issuing queries, the annotated resources move through the network and lay/follow pheromone trails that guide them. In addition, the peers re-order according to their interests which are determined by creating node profiles out of the meta-data their locally stored resources are marked-up with.

4.4.2 Ant algorithms for other distributed tasks

The open-source project *MUTE* [114] 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.

Wu and Aberer [141] use swarm intelligence to create a model for the dynamic interactions between Web servers and users, which provide relevance feedback by browsing Web pages. This model is used for ranking Web documents. A swarm intelligent module is added to the Web server architecture. The trail laying feature used is composed of an accumulation feature that increases the pheromone amount when a user visits a page, and of a spreading feature in which pheromone is diffused to the pages that link to a certain page. The spreading feature does not comply with the Ant Colony Optimization meta-heuristic.

Labella et al. [82] present a foraging-inspired approach for division of labor between a group of robots performing object search and retrieval. The goal is to distributively deter-

mine which agent is suited best for carrying out a certain task using indirect communication only.

4.5 Summary

In this chapter, the effectiveness of the self-organizing and emergent behavior observed from natural ants for query routing in peer-to-peer networks was evaluated. Based on the findings of Chapter 3, two variations of the SEMANT algorithm for content-based query routing in peer-to-peer networks based on the *Ant Colony Optimization* meta-heuristic and on the *AntNet* algorithm were specified.

Because of the necessity to employ a different type of pheromone for every keyword that can occur in a query, allowing free-text search with the trail-laying and trail-following behavior is not feasible. As a consequence, the SEMANT algorithm was designed for metadata-based search. The use case of supporting researchers in computer science by allowing them to share scientific articles was selected, and the ACM Computing Classification System taxonomy was used as the underlying metadata vocabulary.

An experimental evaluation of the proposed algorithm was conducted in order to assess its performance. Analyzing the results gained allows to draw the conclusion that employing pheromone trails for optimizing query routes is a meaningful approach. Both variations of the SEMANT algorithm outperform the k-random walker algorithm, which was used as a reference, in terms of resource usage as well as in terms of hit rate. The *minResources* variation of the SEMANT algorithm was identified as the superior one by comparing the performance of the variations against each other.

The SEMANT algorithm converges fast and delivers stable results. Its performance in the start-up phase can be even more improved by integrating a bootstrapping mechanism that leads to a faster initialization of the pheromone trails.

The values set for the configurable parameters of the SEMANT algorithm and the properties of the peer-to-peer network in which it is used have impacts on the performance of the algorithm. Four main insights have been gained:

- The parameter with the highest influence on performance is the time-to-live parameter. Its optimal value is dependent on the network size and on the content distribution in the network.
- The parameter value chosen for the ratio between exploring and exploiting forwards ants (cf. exploration-exploitation dilemma) does not influence the overall efficiency of the algorithm's search process. The reason for this effect lies in the design of the exploring strategy. Although the forward ants choose outgoing links that are not currently known as the best ones, they still make this decision in proportion to their desirability.

- In a static setting, information will not become outdated. Therefore, using an evaporation factor has a small negative effect on performance.
- Since the SEMANT algorithm's link selection procedure is the less effective the higher the degree of the node, it is not advisable for use in power-law networks. It is indented for use in small-world networks.

In the work presented in this chapter, the namespace of the metadata vocabulary was considered flat. The next chapter presents an extension of the SEMANT algorithm, in which the hierarchical relationships between the topics in the taxonomy are used to improve query routing performance.

Chapter 5

Extensions to the SEMANT algorithm

This chapter (1) investigates the impact of different content distributions on the performance of the SEMANT algorithm and (2) shows how taxonomies can be used for SEMANT in order to improve the performance. After providing the motivation for that work, it is shown how to extend the SEMANT algorithm to provide for the consideration of metadata from a taxonomy for routing. Experimental results indicating the performance of the algorithm with respect to different content distributions – with or without using information from a taxonomy for routing – are presented, and related work on using taxonomies for routing in peer-to-peer networks is discussed.

Contents

5.1 Motivation

Since the keyword of a query governs the way an ant will take through the network, the SEMANT algorithm is a content-based approach. Therefore, as for every content-based approach, the content distribution in the network has an impact of its performance. As discussed in Section 2.3, several existing approaches are relying on assumptions about the distribution of content in the network, i.e., they take for granted that peers have special interests in specific topics and for this reason store a lot of their resources related to these

topics. In the evaluations presented in Chapter 4.3, this assumption was used as well. Remember that each peer was an expert within a certain research area and stored 60% of its resources about this area, 20% about another research area, and 20% of random topics. In this chapter, the goal is to find out which implications this assumption has, and therefore, to evaluate the influence of the content distribution in the network on the performance of the SEMANT algorithm.

In particular, the aim is to vary the degree of "coherence in the content", that is, the percentage of resources about a particular research area stored at the expert peers. Figure 5.1 shows an example of a coherent content distribution, where most of the resources about a certain topic are stored at one expert peer. Compared to Figure 5.1, the content distribution shown in Figure 5.2 is much more dispersed, because there is a higher number of peers storing resources about the given topic, but the number of stored resources is smaller.

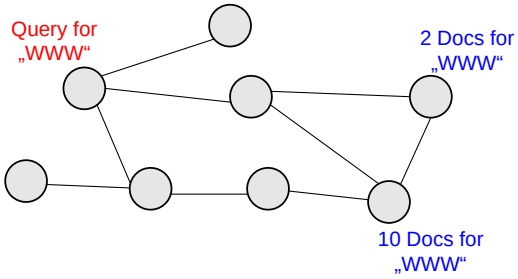

Figure 5.1: Coherent content distribution

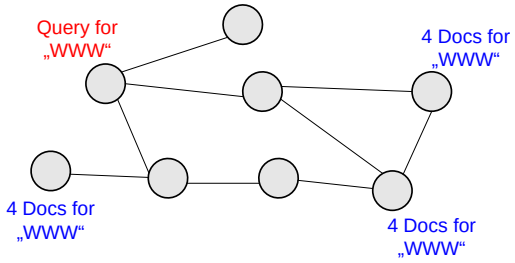

Figure 5.2: Dispersed content distribution

The expected result of the experiments conducted in this section is that the performance of SEMANT is dependent on the content distribution in the network, and that the algorithm's performance will degrade according to the degree of coherence in the content.

Therefore, the additional goal of this work is to develop strategies for enhancing the performance in case of dispersed content distributions. An obvious source of improvement is the underlying taxonomy that was used for creating the test data and modeling the research areas. Recall that research areas are modeled using the third-level topics of the taxonomy, but except from that the namespace was treated as flat and keywords were treated as independent from each other. For making it possible to use this information from the taxonomy, it must be known at every peer.

Basically, the idea now is to (1) lay and maintain pheromone trails not only for the leaf topics in the taxonomy, but also for the higher-level ones, and to (2) define the relationships between the keywords by the underlying taxonomy. The latter can then be used for improving the routing performance. Therefore, SEMANT's content-based approach to query routing will be extended to a *semantic query routing* approach in this chapter. Since the expected result is that the impact of higher-level topics on routing performance is even more

dependent on the content distribution in the network than that of the leaf topics, it is justifiable to discuss both these issues together.

5.2 Defining semantic relationships between pheromone trails

In this section, the extensions for integrating pheromone trails for all topics from the taxonomy are described. There are three parts of the algorithm that need to be extended. Changes need to be made to the data structures (see Section 5.2.1), to the trail following behavior (see Section 5.2.2), and to the trail laying behavior (see Section 5.2.3).

5.2.1 Data structures

As described in Section 4.2.1, in the original version of the SEMANT algorithm, the pheromone trails are maintained in a table τ of size $C \times n$ at each peer P_i, where C is the size of the controlled vocabulary. For using a taxonomy, these tables must be of size $T \times n$, where T is the number of topics in the taxonomy.

5.2.2 Trail following

Since it is executed much more often and most relies on the information already stored in the pheromone trails, considering the hierarchical relationships between the keywords is more useful in the exploiting strategy than in the exploring strategy. For this reason, the exploring strategy remains unchanged as described in Section 4.2.3.2.

In extended version of the exploiting strategy, a forward ant F_q located at a certain peer P_i selects a neighbor peer P_j as shown in Equation 5.1a. It considers the hierarchical structure of the pheromone trails. The strategy incorporates not only the pheromone trail for topic c, which is the keyword of query Q, but also the trails that correspond to the super-topics of topic c. To put more emphasis on those trails that directly match the query, a multiplication factor is used.

$$j = arg\ max_{u \in U \wedge u \notin S(F^Q)} \left(\sum_{i=0}^{n-1} \tau_{c_i u} \cdot \frac{1}{x^i} \right) \tag{5.1a}$$

In Equation 5.1a, U is the set of neighbor peers of peer P_i, $S(F^Q)$ is the set of peers already visited by forward ant F^Q, i is the distance between the super-topic c_i and topic c, $c_0 = c$, n is the distance between topic c and the top-level topic, $\tau_{c_i u}$ is the amount of pheromone for concept c_i on the path to peer P_u, and $x \in [2, 4, 8, 16, ...]$. Parameter n defines how many super-topics are considered and parameter x defines to which extent they are incorporated.

The appropriate values for n and x were determined using experimental evaluations. It turned out that the most appropriate value is to set $n = 2$. This means that only the direct super-topic – the one that corresponds to the research area – is specific enough for improving performance. The higher-level super-topics are not specific enough to bring improvements. If they are taken into account as well, performance is actually lowered.

To measure the usefulness of the underlying taxonomical information for the routing performance, the newly defined exploiting strategy as described above is compared to its original version (see Section 4.2.3.1), in which all pheromone trails for different topics are treated as being independent from each other. A forward ant F_q located at a certain peer P_i simply selects the link that stores the highest amount of pheromone for the given keyword as shown in Equation 5.1b.

$$j = arg\ max_{u \in U \wedge u \notin S(F^Q)} \tau_{cu} \tag{5.1b}$$

In Equation 5.1b, U is the set of neighbor peers of peer P_i, $S(F^Q)$ is the set of peers already visited by forward ant F^Q, and $\tau_{c_i u}$ is the amount of pheromone for concept c_i on the path to peer P_u.

Remember that the original version of the exploiting strategy (see Section 4.2.3.1) considered both the current amount of pheromone and the cost of a certain link for making a decision which outgoing link to follow. As discussed in Section 4.5, appropriate test data is missing for evaluation of this feature. For this reason, links costs were omitted in Equation 5.1a and Equation 5.1b.

5.2.3 Trail laying

In order to make it possible to consider them in the forward ants' exploiting strategy, certain amounts of pheromone need to be dropped by the backward ants not only to the topic itself, but also to the higher-level topics. Therefore, changes to the pheromone update rule as defined in Section 4.2.4 are necessary. The extended pheromone update rule is defined in Equation 5.2a to Equation 5.2c. Equal amounts of pheromone are dropped on the pheromone trail for topic c – which is the keyword of query Q – and on the pheromone trails corresponding to the super-topics c_i of topic c.

$$\tau_{c_i j} \leftarrow \tau_{c_i j} + Z, \tag{5.2a}$$

where

$$i = \{0, \ldots, n-1\}, c_0 = c \tag{5.2b}$$

and

$$Z = w_d \cdot \frac{|R|}{r_{max}} + (1 - w_d) \cdot \frac{ttl_{max}}{2 \cdot h_{qr}} \tag{5.2c}$$

ρ	evaporation factor	0.07
ttl_{max}	timeout of forward ants	25
w_s	weight of exploiting and exploring strategy	0.85
r_{max}	maximal number of resources	10
w_d	weight of resource quantity and path length	0.5
n	number of superconcept pheromone trails incorporated (including concept itself)	2

Table 5.1: Parameter values set for the SEMANT algorithm

As shown in Equation 5.2c, the values for the optimal solution are set to $\frac{1}{2} \cdot ttl_{max}$ for the path length and r_{max} for the number of resources. Parameter w_d weights the influence of resource quantities and path lengths.

5.3 Experimental results

The focus of this section is on evaluating the performance of the extensions to the algorithm. Section 5.3.1 describes the simulation setup. In Section 5.3.2, the impact of the number of experts in the network on the performance of the algorithm is determined. In Section 5.3.3, the impact of the degree of coherence in the content on performance is evaluated.

5.3.1 Simulation setup

The simulation setup used for the subsequent experiments is the same as already described in Section 4.3.1, and the same metrics as already defined in Section 4.3.2 are used. For distributing the resources across the network, it is assumed that each peer is an expert on a certain topic and therefore a certain percentage P_{expert} of its resources are instances of one particular research area. A research area is modeled by the third-level topics of the taxonomy and their leaf topics, as shown in Figure 4.3. The parameter values for the SEMANT algorithm are set as shown in Table 5.1.

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

5.3.2 Number of experts in the network

In the first experiment, the aim is to compare the effects different numbers of experts have on the overall performance. The setting $P_{expert} = 100\%$ is used for all test runs. What varies

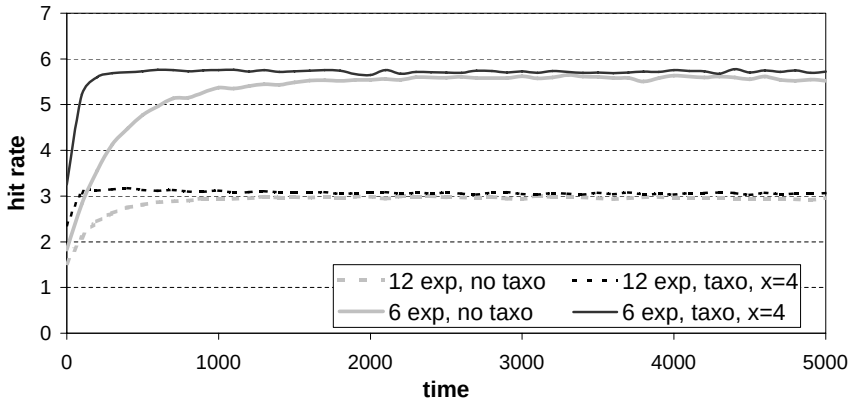

Figure 5.3: Hit rate evaluation of the impact of the number of experts (six and twelve) on performance in case of (1) using a flat namespace and in case of (2) using a hierarchic namespace

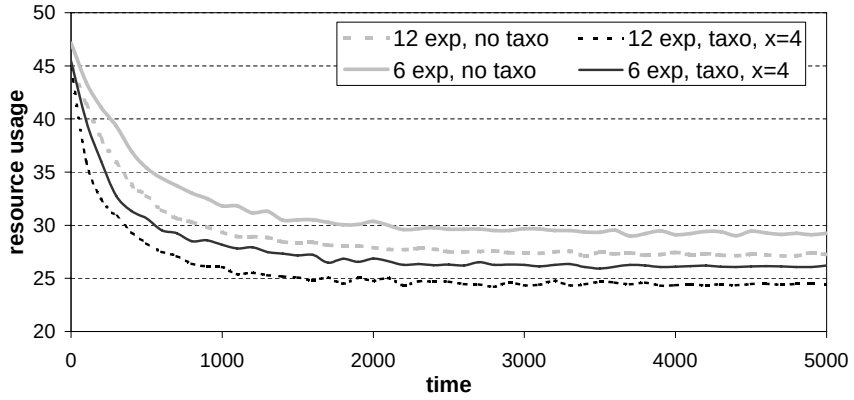

Figure 5.4: Resource usage evaluation of the impact of the number of experts (six and twelve) on performance in case of (1) using a flat namespace and in case of (2) using a hierarchic namespace

is the number of experts in the network. Two different cases are compared. In the first case, there are six experts per research area in the network. In the second case, there are twelve experts per research area. The number of resources per research area stays constant. Hence, in the latter case with twelve experts, each expert stores half as many resources as in case of six experts. Therefore, the content distribution is more dispersed in the case of twelve experts.

Because of $P_{expert} = 100\%$, the corresponding super-topic has the same significance for a query as the topic itself. Thus, it is irrelevant to which extend super-topics are considered, that is, which value is used for parameter x. It is set to $x = 4$.

The results of the experiment are shown in Figure 5.3 to Figure 5.5. Figure 5.3 shows the results for the hit rate. It reveals that the results are the better, the less experts exist. For

six experts (solid lines), the hit rate after 5000 time units is approximately 5.5 resources. For twelve experts (dashed lines), it is approximately 3 resources.

The reason for that is the following: the more experts exist, the less resources are stored at one expert peer. As soon as the converged phase is reached, the results for the hit rate (Figure 5.3) are nearly the same, no matter whether the super-topics are considered (black lines) or not (grey lines). The discrepancy is that, if the super-topic trails are included, the algorithm converges faster. The reason for that is that the super-topic has the same significance for a query as the topic itself. This helps to speed up the process, but not to improve the hit rate.

Figure 5.4 shows the results for the resource usage. As shown, the values for resource usage are proportionally lower if super-topic trails are included. The difference is rather small (approximately 3 messages in both cases) but permanent, because it is still present after 5000 time units. Note that the resources used by the backward ants are included in Figure 5.4. This is the reason why the resource usage is higher in case of six experts, because more resources are found. Therefore, more backward ants are created, which leads to more traffic in the network.

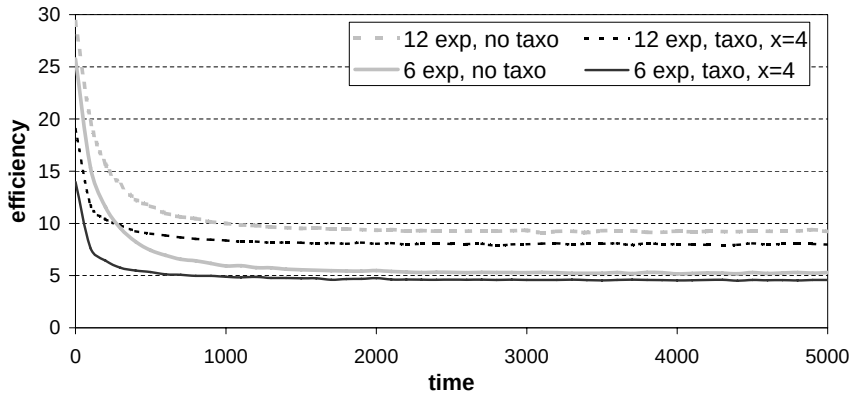

Figure 5.5: Efficiency evaluation of the impact of the number of experts (six and twelve) on performance in case of (1) using a flat namespace and in case of (2) using a hierarchic namespace

Since resource usage is dependent on hit rate as discussed above, the most meaningful metric for evaluation is the combined measure of efficiency. The results for efficiency are shown in Figure 5.5. The results show that the algorithm is very efficient in all four settings where $P_{expert} = 100\%$. For six experts without using the taxonomy, after 5000 time units approximately 5.30 hops are necessary in order to retrieve one resource. When using the taxonomy, approximately 4.58 hops are necessary. Hence, in this case the improvement is only slightly significant (0.72 hops). In case of 12 experts, the improvement after 5000 time units is 1.24 hops for retrieving one resource (9.22 without taxonomy, 7.98 with taxonomy). Again, this is only a slight improvement.

To summarize, in this experiment, using the super-topic trails for routing gives nearly the same results as not using them. The main and significant improvement that can be reached is that integrating super-topic trails into the routing decisions lowers the time it takes for the algorithm to reach the converged phase. Approximately 500 time units are necessary when creating hierarchical relationships between keywords in comparison to approximately 1000 time units that are necessary when treating the keyword namespace as flat.

5.3.3 Degree of coherence in the content

In the experiment described in the last section, all resources corresponding to a certain keyword were stored at one of the expert peers. This setting is the optimal one for using super-topics for routing. In this experiment, the effects of dispersed resource distributions are investigated. We use six expert peers for every test run, acquire the results for $P_{expert} = [0\%, 20\%, 40\%, 60\%, 80\%, 100\%]$, and compare them. The percentage of resources that is not stored at the expert peers is stored at randomly selected peers. Parameter x,

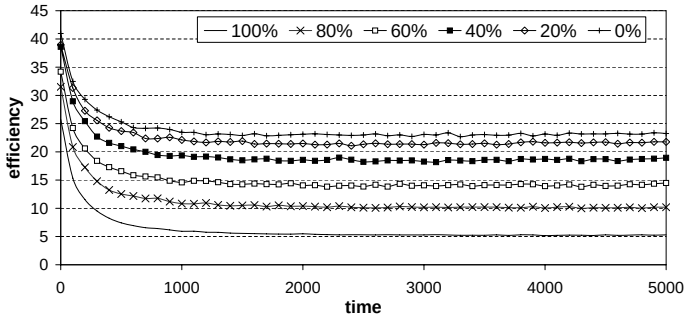

Figure 5.6: Efficiency evaluation of the impact of the degree of coherence in the content on performance in case of a flat namespace

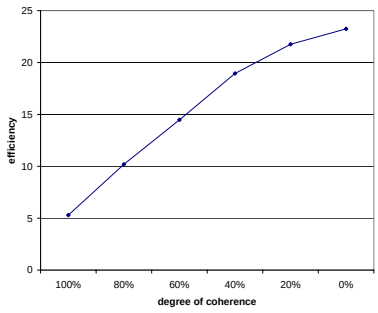

Figure 5.7: Comparison of the final results after 5000 time units of Figure 5.6

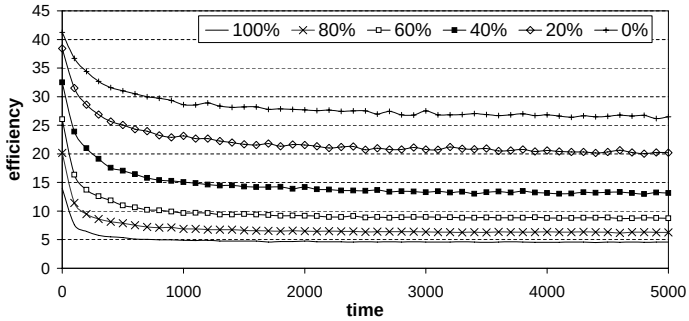

Figure 5.8: Efficiency evaluation of the impact of the degree of coherence in the content on performance in case of a hierarchic namespace (x=4)

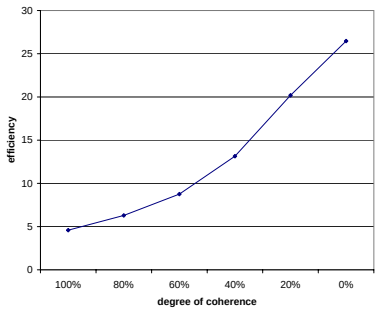

Figure 5.9: Comparison of the final results after 5000 time units of Figure 5.8

which defines the degree to which the super-topics are taken into account, is set to $x = 4$. This means that for a given query, the trail for the keyword of the query is considered to 100%, and the trail for the super-topic is considered to 25%. As experiments have shown, considering super-topics to an extent of more than 25% lowers performance. Since the optimal value is different for every content distribution, it would be necessary to determine it separately. However, in order to enable comparison between the test runs the same value $x = 4$, is used for all content distributions.

Now the results are presented and discussed. Since it turned out to be the most meaningful metric, only the results for efficiency are depicted. Figure 5.6 shows the values for efficiency when relying on the pheromone trails for topics only. As expected, the algorithms' performance is the weaker, the more disperse the content distribution is. For $P_{expert} = 100\%$, the algorithm needs on average 5 hops to retrieve one resource (cf. Section 5.3.2), which is a very efficient result. However, the results degrade proportionately to the coherence in the content. For $P_{expert} = 80\%$, $P_{expert} = 60\%$, and $P_{expert} = 40\%$, the degradation is linear (see Figure 5.7) and its amount rather high. For $P_{expert} = 20\%$ and $P_{expert} = 0\%$, the degree of degradation is lower but still immense. These results show that content-based approaches, such as SEMANT, are very inefficient for dispersed content distributions.

Coherence	Flat namespace	Taxonomy	Improvement
0%	23.25	26.47	worse
20%	21.76	20.19	7,2%
40%	18.95	13.15	30,6%
60%	14.48	8.76	39.5%
80%	10.20	6.29	38,2%
100%	5.30	4.58	13,5%

Table 5.2: Comparison of efficiency after 5000 time units when varying the degree of coherence in the content on performance (1) in case of a flat namespace and (2) in case of a hierarchic namespace

Now we look at the results when considering the hierarchical relationships between the keywords for routing. Figure 5.8 shows the results when the pheromone trails for super-topics are considered, Figure 5.9 shows the results after 5000 time units, and Table 5.2 shows a comparison of the results when using a flat namespace to those when using the information from the taxonomy. These results show that using super-topic pheromone trails for routing has a positive impact on performance as long as at least some patterns can be found in the content distribution. Solely in case of $P_{expert} = 0\%$, where all resources are individually stored at randomly chosen peers, considering the super-topic lowers performance. The reason for that is that if no coherence in the content can be found at all, the notion of research area that corresponds to the trails for the super-topics can not be found in the content distribution. Hence, the trails for the super-topics actually transport no information.

However, since there is always at least some coherence in the content in real-world settings, the results for $P_{expert} = 0\%$ can be neglected.

The most significant improvement in performance can be found in case of $P_{expert} = 60\%$, $P_{expert} = 80\%$, and $P_{expert} = 40\%$ (see Figure 5.9, and compare it to Figure 5.7). The degree of improvement is depicted in the right-most column of Table 5.2.

Another observation that can be made from these figures is that the lower the value for P_{expert}, the longer it takes until a converged phase is reached, and consequently, the more shallow the curve. For the case of $P_{expert} = 20\%$, not considering super-topics outperforms considering them until time unit 2500 is reached. However, after 5000 time units, a slight improvement of 7,2% can be found.

Note that the case of $P_{expert} = 100\%$ is not addressed here, because it was already discussed in Section 5.3.2. In some respect, the experiment described in this section can be compared to the one described in Section 5.3.2, because using the same number of resources together with a higher number of expert peers for a given topic has basically the same effects as spreading a percentage of the resources randomly. In both cases, the content distribution will be more dispersed. However, in the experiment described here the effects are easier to observe, because the degree of coherence is varied in a more fine-grained level of detail. In addition, since the randomly spread resources are stored individually at certain peers, the degree of dispersedness is higher in this case.

To summarize, the performance of the algorithm is highly dependent on the content distribution in the network. The lower the degree of coherence in the content, the worse the performance. Using the super-topic trails for routing significantly improves the performance in case of moderately coherent content distributions.

5.4 Related work

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.

It can be distinguished 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 [41] 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. Löser [86] investigates how a super-peer architecture and a distributed hash table can be combined to create a distributed catalog of peer descriptions. Schmitz [120] deals with a network in which each peer is characterized by one topic 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.

In the second category, next to the work of Tempich et al. (see Section 2.3), Pirredu and Nascimento [106] describe a peer-to-peer system in which all available content is annotated according to a balanced taxonomy. In a balanced taxonomy, all leaf topics can be found at the same level of hierarchy. The peers are responsible for two tasks. Firstly, they broadcast information about their contents. Secondly, they 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 [40]. Based on these indices, queries can be routed effectively. Similar to the work presented in this chapter, Pirredu and Nascimento use the ACM CCS taxonomy [11] for the design of the application scenario used in the experimental evaluation. Each resource is an instance of one leaf topic of the taxonomy and the allowed queries are limited to the leaf topics.

All the approaches mentioned above, and also the SEMANT algorithm, assume that the underlying taxonomy is known at every peer and that all the content is annotated according to that taxonomy. Edutella [99] provides a less restrictive infrastructure in which several different taxonomies can be used for annotation. Tzitzikas et al. [133] present a model in which every peer can employ its own taxonomy for resource annotation. For queries, mappings between the taxonomies are necessary.

5.5 Summary

Since SEMANT is a content-based approach to query routing in unstructured peer-to-peer network, its performance depends on how the content is distributed within the network.

The aim of this chapter was to investigate to which extent the performance of the algorithm is influenced by the content distribution in the network, and to determine the performance values under these different circumstances. The actual result values that were acquired and described in this chapter are valid for the SEMANT algorithm only, but the bigger picture applies to other content-based approaches as well. The results have shown that the performance of the algorithm degrades heavily if a dispersed content distribution is present in the network.

Therefore, the second main goal of this chapter was to develop strategies for improving the performance in these cases. Under the assumption that all the resources in the network are annotated according to a taxonomy that is known at every peer, an extended version of the SEMANT algorithm was presented that integrates these information into the routing decisions by treating the keywords as hierarchically related to each other, and by creating pheromone trails also for higher level topics. After empirically determining the extent to

which this information should be considered in order to actually improve the results, the effectiveness of the algorithm when using a flat namespace for keywords was compared to its effectiveness when using a hierarchical namespace. The results show that the performance of SEMANT can be significantly improved by considering super-topic pheromone trails for routing, particularly if the content distribution is moderately dispersed.

Part II

Acquisition of Dynamic User Profiles from Folksonomies

Chapter 6

A Case Study on Emergent Semantics in Communities

This chapter delivers a case study on the properties of metadata provided by a folksonomy (also referred to as *tagging data*). After providing the background about folksonomies and discussing to which extend the process of creating metadata in a folksonomy is related to the idea of emergent semantics as defined by the IFIP 2.6 Working Group on Data Semantics, experiments are conducted in order to analyze the metadata provided by the *del.icio.us* folksonomy. The aim is to (1) evaluate if tagging data adheres to the principle of interest-based locality, which was originally observed in peer-to-peer environments, and to (2) compare the data provided by the *del.icio.us* folksonomy to data provided by the DMOZ taxonomy.

This chapter provides the connection between the two parts of the thesis. Next to serving as an initial starting point for the more detailed investigation of folksonomies in Chapter 7, the work presented in this chapter has also been carried out to determine if tagging data can be used as test data for a peer-to-peer simulation environment, and therefore contains backlinks to the work described in Part I of the thesis.

Contents

The chapter is organized as follows. Section 6.1 provides the necessary background about folksonomies and discusses their strengths and weaknesses. Section 6.2 compares the behavior of the participants in a folksonomy to that of peers in a peer-to-peer network, and

examines the relationship between emergent semantics and folksonomies. Section 6.3 describes the experiments conducted on the provided metadata in order to analyze its properties, and reports on the results of these experiments. Section 6.4 concludes.

6.1 Folksonomies and their characteristics

The term *folksonomies* refers to a class of multi-user applications that provide a simple categorization system. These systems are used to organize items, e.g., bookmarks or images. Instead of managing them within the browser application or on the local hard disk, the items are sent to a central server and stored there, together with metadata authored by the user. The metadata are comprised of one or more keywords — so-called *tags* — which describe the item. The keywords can be chosen freely by the user. Unlike in other categorization systems, there is no controlled vocabulary that defines which terms can be used as keywords in the categorization process. Another difference to existing categorization systems is that all keywords lie within the same namespace. There is no intention and hence no possibility to build hierarchical relationships between different keywords. Folksonomies uses a very simple data model which is depicted in Figure 6.1.

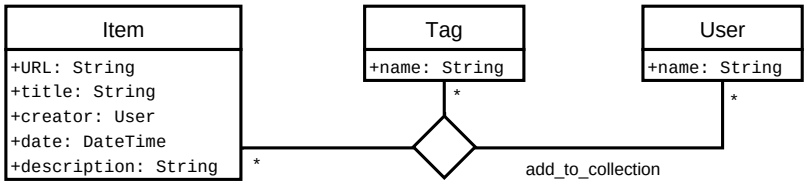

Figure 6.1: Simple data model

A folksonomy provides each participant with his or her own Web page that shows the participant's item collection which contains all items together with the corresponding tags. The items are sorted in chronological order, showing the latest entry first. It is possible to filter the list of items by tag names to show only items that are annotated with a certain tag A. It is also possible to use another tag B to filter this list and retrieve all items that were annotated with both A and B.

Folksonomies provide convenient interfaces to their participants, e.g. the possibility to add an item using a special link containing JavaScript code — a so-called bookmarklet — that transfers the URL to be added to the server. While typing in keywords, the participant is shown a list of the keywords he or she already used before. If the bookmark is already stored in the system, a list of popular keywords for that bookmark is shown as well. Those two lists are facilities that assist the participants in choosing appropriate keywords. After storing the bookmark, he or she is automatically redirected to the newly added Web page.

Folksonomies are of value to the individual for managing items, but more important is that all participants of a folksonomy cooperate by allowing public access to their item collection and the associated metadata. Public availability has two advantages:

- Firstly, all Web users have access to the annotated item collections.
- Secondly, since all metadata are stored at one single server, it is possible to analyze and aggregate it without having additional communication costs in terms of bandwidth.

Aggregations are performed for both items and tags. Information about items can be aggregated since the participants individually and independently create metadata about the same items. For each item, there exists a Web page in which all tags that an item was annotated with are shown, together with the total number of participants that used that tag to annotate the item (see Figure 6.2). In addition, the service lists all participants that included the item in their collection and shows the total number of these participants. The total number shows the popularity of a certain item. Aggregation for tags is possible because the participants use the same tags to annotate different items. For each tag, there exists a Web page with a list of all items at least once annotated with this tag.

The result of these aggregations is a network of related concepts. At the data level, folksonomies are undirected weighted graphs that can be seen from different perspectives. When viewing items as the nodes of the network, two nodes are connected if they are annotated with the same tags. These kinds of connections are weighted. The more often the same tags were used, the higher the weight of the edge. When viewing the participants as the nodes of the network, the edges are built by those items that are shared between the participants. These graphs are exploited as input for an algorithm that computes the relatedness of tags. Each Web page containing aggregated information about a certain tag also shows the tag names of related tags as computed by the algorithm. For example, the algorithm used by the photo sharing service *Flickr* [144] computes *tent*, *fire*, *hiking* as related tags for tag *camping*.

```
1 | 43 physics     15 mathematics  7 library      5 article          3 ai
2 | 41 science     10 articles     6 preprint     4 computer         3 study
3 | 27 research    10 journal      6 books        4 arxiv
4 | 23 math         9 archive      6 programming  4 literature
5 | 19 papers       8 biology      5 cs           4 toread
6 | 18 reference    7 eprint       5 academic     4 computerscience
```

Figure 6.2: Tag distribution of a sample del.icio.us item

This approach to categorization is different to the top-down approach that is employed in traditional categorization systems, e.g., taxonomies or ontologies. Ontologies provide domain-specific vocabularies that describe the conceptual elements of a domain and the

relationship between these elements, such as *is-a*-relationships or *part-of*-relationships. Creating an ontology requires careful analysis of what kind of objects and relations can exist in that domain [32]. This analysis is performed by domain experts together with information architects who need to reach consensus about the exact meaning of objects and relations. After the ontology has been developed and put in place, the data items are categorized according to the chosen categorization scheme. The same top-down approach is used in the database world. Before the actual data comes in, a schema is built that for that data. The schema defines which entities exists and how they are related. A third example is object-oriented modeling, where an instance can not exist without being a member of a class. The model defines the hierarchy of objects and their relationships.

In a folksonomy, each participant uses a certain tag with his or her personal meaning in mind. There is no controlled vocabulary or ontology that defines the meaning of tags. Everybody has the possibility to express his or her opinions about the categorization of a certain object. This freedom of choice induces all problems controlled vocabularies try to avoid:

- The participants can use either the singular or the plural form of a term. Hence, they create two different tags with exactly the same meaning. There is no synonym control.
- Different terms that refer to the same concept are in use.
- A special problem are keywords that consist of two terms: some participants create one tag by combining the terms with underscores or hyphens, or by creating a compound word with no separating character in between, while others create two tags, one for each term.

The bottom-up approach to categorization avoids the necessity to reach consensus about the most appropriate categorization of a certain object. Instead, semantic reconciliation is performed by the magnitude of participants that added metadata to the folksonomy. The tags that are used most often to annotate a certain item express the opinion of the majority. Thus, the utility of a folksonomy is directly proportional to its number of participants and the amount of metadata produced by them.

There are a number of existing services like *del.icio.us* [143] for bookmarks, *Flickr* [144] for images, *Connotea* [89] for references to scientific literature, and others. A review of the existing services is provided in [63]. Services for bookmarks are also called *social bookmarking services*. More information about the general ideas behind folksonomies can be found in [5] and [63]. In [36], the major differences between folksonomies and taxonomies are discussed and some statistical information about the tag distribution of the *del.icio.us* social bookmarking system is presented. A case study on the service the *Connotea* service, which provides a folksonomy for sharing metadata about scientific literature, can be found in [89].

6.2 Folksonomies and peer-to-peer environments

This section sheds some light on the relationship between folksonomies and peer-to-peer environments. Section 6.2.1 discusses if it is basically possible to use folksonomies for retrieving simulation data for peer-to-peer networks. Section 6.2.2 briefly introduces the idea of emergent semantics (which was brought up up within the area of peer-to-peer networks) and discusses its relationship to folksonomies.

6.2.1 Architecture, user behavior, and availability of data

Folksonomies are centralized services that heavily rely on the aggregation of metadata. These aggregations are possible because the metadata reside on a single server. For example, it is easy to determine all participants who share a certain information item. On the contrary, in a peer-to-peer environment there is no central server, and all peers store their information items at their local hard disk. Aggregating data consumes network resources. Knowing which peers share a certain information item is a non-trivial task for which it is necessary to track how the items are replicated within the network [20]. However, although the architectures of folksonomies (centralized) and peer-to-peer networks (distributed) are completely different, the important point is that the behavior of participants in a folksonomy is comparable to the behavior of peers in an unstructured peer-to-peer network:

- All participants act autonomously and there is no central authority coordinating them.
- All participants provide information items to others that can be browsed and retrieved.

Since the metadata produced by folksonomies are publicly available and can be easily retrieved from one central server, they provide an opportunity for retrieving test data for peer-to-peer applications. There is data available that can be used for modeling peers and their content distribution. However, the data about queries and query distribution is not provided. Moreover, the question is whether the content distribution in the tagging data has the same properties that are usually present in peer-to-peer content distributions, such as interest-based locality, or – more specifically – if a subset of the vast amount of tagging data can be selected that fulfills these properties. This is the subject of Section 6.3.2.

6.2.2 Do folksonomies provide emergent semantics?

The term *emergent semantics* was defined by Aberer et al. in [4]. In this work, the authors discuss semantic interoperability for loosely coupled information sources and observe that a-priori agreements on concepts, e.g., the use of ontologies, are not appropriate in ad-hoc

situations, because there is no possibility for the communicating peers to anticipate all interpretations. This is a major argument against using a taxonomy-based application scenario as the one described in Part I of the thesis.

Instead, the idea of *emergent semantics* suggests a semantic handshake protocol that allows negotiations between pairs of peers to reach an agreement over the meaning of models, e.g., by local schema mapping [3]. In order to save network resources, these negotiations are created out of local interactions whenever possible. Global agreements are obtained by aggregating local agreements. This means that semantic interoperability is constructed incrementally by lots of negotiations which are influenced by the context of existing global agreements. When letting aside the major differences that stem from the fact that folksonomies operate in a centralized environment, there are some ideas from emergent semantics that can be found in folksonomies. Both approaches

1. do not force their users to commit themselves to an existing ontology,

2. rely on lots of small interactions as well as on aggregations of the results of these interactions, and both

3. construct their global properties incrementally.

The main distinction is that, while the interactions in emergent semantics are initiated in order to reach consensus, in a folksonomy they are merely information-exchanging acts that to not lead to any definite agreement. The frequency distribution of the tags used for annotating a bookmark show the popularity of every tag, and its popularity might give some information about its accuracy. Still, there is no agreement. Tags with low popularity and tags with high popularity coexist.

Emergent semantics was proposed as an alternative to the use of top-down approaches to classification, such as the definition of ontologies or taxonomies. It was formulated as an abstract idea without providing an implementation. Social bookmarking services are an example of emergent semantics (with the restrictions discussed above). The research question addressed in Section 6.3.3 is whether the resulting metadata from the bottom-up classification process is similar to the resulting metadata when using a top-down approach. The comparison is performed on the data level.

6.3 Experiments and Results

This section reports on experiments conducted on data retrieved from a social bookmarking service. In Section 6.3.1, the experimental setup is described. The contents of Section 6.3.2 and Section 6.3.3 have been described above.

6.3.1 Experimental setup and test data

The test data used in the following experiments was gathered by downloading selected bookmarks from *del.icio.us* [143], which is one of the most successful social bookmarking services. The data was downloaded between June 21 and June 30, 2005. The downloading routines were implemented as Perl scripts. They have the following features:

- In order not to take up too many resources from the service, a delay of five seconds between each subsequent request was included.
- A list of all already retrieved URLs was kept to prevent multiple downloading of the same information.
- Error handling facilities were necessary since internal server errors of the service occurred frequently.

In order to include only those bookmarks with a sufficient amount of metadata attached, the tags and their frequency distribution for the bookmark were retrieved only if the routine encountered a bookmark that was included in the bookmark collection of more than hundred participants. If otherwise, the bookmark was not considered. All downloaded data was saved to text files with a simple format.

6.3.2 Data selection

The hypothesis of the experiment described in this section is that the principle of *interest-based locality* [123] can be observed in folksonomies. It has originally been observed in peer-to-peer environments, and it is defined as follows:

> If a participant A has a particular piece of content another participant B is interested in, it is likely the case that the other information items stored by participant A are also of interest to participant B.

The following algorithm is used for testing this hypothesis. Firstly, a random bookmark b is chosen as a starting point. In the second step, the names of all participants in the social bookmarking service that store bookmark b in their collection are retrieved. In the third step, the bookmark collections of all of these participants are downloaded. To keep the size of the test set within reasonable boundaries, only those fifty entries of each participant's bookmark collection which were added latest were included. This means that for participants storing less than fifty entries, the complete bookmark collection was considered.

The result of applying this algorithm is a test set which contains bookmark collections of a number of users. At least one of the bookmarks – bookmark b – is contained in every collection. If the principle of *interest-based locality* is present in the test set, the necessary

Test set Nr.	1	2	3	4
Number of participants	551	155	248	280
Number of items	17575	5709	8861	10237
Number of unique items	12855	4311	6045	6643
in % of all items	73,14%	75,51%	68,22%	64,89%
Number of popular items	5691	2393	3691	4207
Number of unique popular items	2301	1217	1479	1483
in % of all unique items	17,90%	28,23%	24,47%	22,23%
Average number of items per user (Max: 50)	31,9	36,83	35,73	36,56
Average number of popular items per user	10,32	14,79	13,61	13,50

Table 6.1: Properties of the test sets

condition is that also other bookmarks must be contained in all (or at least in the majority) of the bookmark collections.

Using the procedure described above, four test sets of different size (a big one, two medium-sized ones, and a small one) were collected. The four random bookmarks[1] were chosen to be bookmarks that refer to Web sites containing information about diverse topics.

The basic properties of the test sets are shown in Table 6.1. First of all, it can be seen that the total number of bookmarks of a test set is proportional to the number of participants that store bookmark b. Since the percentage of unique bookmarks in each test set ranges from 64,89% to 75,51%, the number of unique bookmarks in a test set is not proportional to the total number of bookmarks.

There are only small differences in the average number of bookmarks included in a participant's collection (Min: 31,9, Max: 36,83). The number of participants in a test set has a small impact on this number: In test set 1, which is by far the biggest of all sets, the average number of bookmarks per participant is higher than in the other test sets. The decision to consider only the first fifty entries of each bookmark collection was based on the assumption that a high percentage of all *del.icio.us* participants stores more than fifty entries. The retrieved data reveals that this is not the case. On average, only 1,15% of participants in each set own a collection which is comprised of fifty bookmarks or more.

A bookmark is defined to be a popular bookmark if it is included in the bookmark collection of more than hundred *del.icio.us* participants. On average, a third of all bookmarks fall in the category of popular bookmarks. As can be seen in test set 1 and test set 2, the smaller the test set in terms of number of participants, the higher the amount of popular bookmarks per user. This is partly due to bookmark b being popular.

[1] test set 1: b is `http://del.icio.us/url/06df5507a27ab5aa297fbb7748374df6`,
test set 2: b is `http://del.icio.us/url/463f3f6f9ce9471fef7f9edb881ad2d7`,
test set 3: b is `http://del.icio.us/url/245d7b2a49a80771da9a4d3a02d539c3`,

All items shared ...	by > 10	by 5 - 10	by 4	by 3	by 2	not shared
Test set 1	0,41%	1,84%	1,34%	2,76%	9,09%	84,56%
Test set 2	0,16%	1,83%	1,37%	2,85%	9,21%	84,57%
Test set 3	0,58%	2,45%	1,70%	2,96%	8,68%	83,62%
Test set 4	0,90%	2,60%	1,55%	3,10%	8,63%	83,22%
Average	0,51%	2,18%	1,49%	2,92%	8,90%	84,00%
Popular items shared ...	by > 10	by 5 - 10	by 4	by 3	by 2	not shared
Test set 1	1,91%	8,43%	6,04%	9,87%	23,55%	50,20%
Test set 2	0,49%	6,08%	3,94%	9,37%	20,46%	59,65%
Test set 3	2,50%	8,32%	3,92%	8,92%	17,58%	58,76%
Test set 4	3,84%	8,36%	4,18%	8,90%	18,07%	56,64%
Average	2,19%	7,80%	4,52%	9,27%	19,92%	56,30%

Table 6.2: Distribution of bookmarks if (a) considering all bookmarks (top), or (b) considering popular bookmarks only (bottom)

Next, the distribution of bookmarks in the test sets is analyzed for testing the hypothesis. For each bookmark, the total number of participants that store it in their collection was determined. The results are shown at the top of Table 6.2. It turns out that the bookmark distribution has equal properties for every test set: On average, only 0,51% of the bookmarks are stored by more than ten participants. 2,18% are stored by a group of five to ten participants. 1,49% are stored by four participants. 2,92% are stored by three participants. 8,90% are stored by two participants. These are very low values. On the contrary, the percentage of bookmarks that are not shared (but stored in only one participant's collection) is rather high. It is nearly equal in each test and it amounts to on average 84%. These figures show that the test sets do not conform to the principle of interest-based locality.

Even when considering only the popular bookmarks, interest-based locality is still not present. As can be seen at the bottom of Table 6.2, in this case the percentage of bookmarks that are present in only one collection lowers to 56,3%. On average, 19,92% of all popular bookmarks are shared by two participants, 9,27% are shared by three participants, 4,52% by four participants, 7,8% are shared by between five and ten participants, and 2,19% by more than ten participants.

Assuming that bookmarks that refer to Web sites with similar topics are annotated with the same tags, and that tags are less specific than bookmarks, now the tags associated with the popular bookmarks are considered for evaluating if interest-based locality is present in this case. For each bookmark, the top tag that was used by most participants was selected. The distributions of the top tags for all popular bookmarks are shown in Figure 6.3 to Figure 6.6. They reveal that the distribution curves for all four test sets have equal properties:

test set 4: b is `http://del.icio.us/url/c745432a483a84037c90e08d79f7c306`

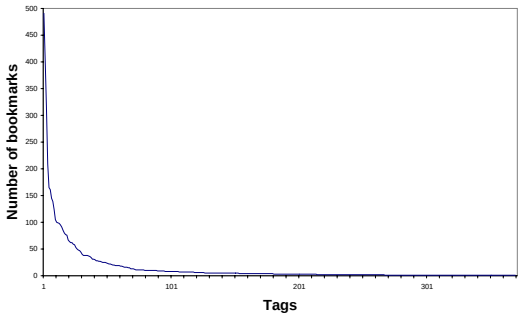

Figure 6.3: Top tag distribution (ranked plot, linear scale) in set 1

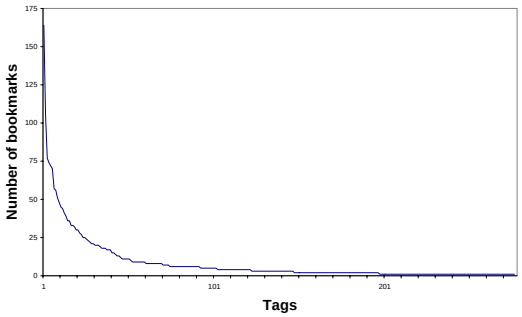

Figure 6.4: Top tag distribution (ranked plot, linear scale) in set 2

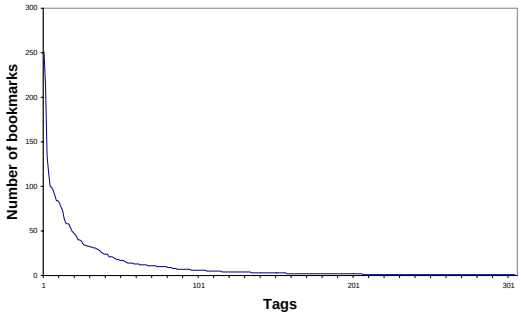

Figure 6.5: Top tag distribution (ranked plot, linear scale) in set 3

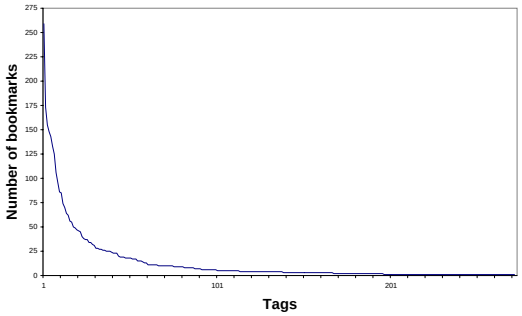

Figure 6.6: Top tag distribution (ranked plot, linear scale) in set 4

There is a long tail in each curve that reveals that there are many top tags that are included only once. The reason for that lies in the diversity of bookmark collections. Too many of them contain items about topics that are not related to the topics of the other items in the collection. This means that, even when considering the top tags of the popular bookmarks only, it is not possible to retrieve test sets that conform to the principle of interest-based locality without any further preparation of the data. The necessary preparations consist of removing all items that cause the tail of the top tag distribution.

6.3.3 Data semantics

In this experiment, the metadata provided by the *del.icio.us* folksonomy is compared to the metadata provided by the *DMOZ Project* [101]. The DMOZ project is a community effort to build a taxonomy for Web pages, and to categorize Web pages according to this taxonomy. It has been used for simulating user behavior in a peer-to-peer network [130].

For conducting this experiment, the RDF dump[2] of the structure and of the contents of the DMOZ directory was downloaded and stored in a relational database for performance

[2]available at `http://rdf.dmoz.org`

```
URL http://arxiv.org/
DMOZ Top/Science/Physics/Publications
DMOZ Top/Science/Math/Publications
DMOZ Top/Science/Math/Publications/Online_Texts/Collections
DMOZ Top/Science/Publications/Archives/Free_Access_Online_Archives
ID 19aa8ff1e9e2a06677ab34f3f2a5b0c8
TITLE arXiv.org e-Print archive
TAGS physics:43;science:41;research:27;math:23;papers:19;reference:18;mathematics:15;
journal:10;articles:10;archive:9;biology:8;eprint:7;library:7;preprint:6;books:6;
programming:6;cs:5;article:5;academic:5;computer:4;arxiv:4;literature:4;toread:4;
computerscience:4;ai:3;study:3;
```

Figure 6.7: A sample entry containing metadata from both sources

reasons. After that, a database lookup for each popular bookmark included in the test sets described in Section 6.3.2 was performed to check if the bookmark is included in the DMOZ contents. Each time the lookup routine encountered a hit, the bookmark and its metadata from both sources were appended to a text file with a very simple format (see Figure 6.7 for an example). If a bookmark was assigned more than one DMOZ topic, all of them were considered. Two observations were made when performing this task:

- The intersection of *del.icio.us* and the DMOZ directory is rather small. Although only popular bookmarks were used, only 25% of the bookmarks were also included in the contents of the DMOZ directory.
- Nearly 50% of those bookmarks that are present both in both sources are instances of subtopics of the DMOZ topic `Top/Computers`.

For the matching between tags and topic names, some preparations of the data were necessary:

- All DMOZ topic names were converted to lower-case characters.
- Underscores and hyphens were removed from both topic and tag names.
- To overcome the problem that singular and plural versions of a tag are in use, the last character of a tag or topic name was removed in case of it being the letter `s`. For example, `computers` was changed to `computer`.
- Some DMOZ topics have 26 subtopics for each letter from A to Z in order to categorize items by the first letter of their name. Such topics consisting of only one character were removed from the topic path.
- The topic `Top` was removed from each topic path.
- Since the leaf entry from the DMOZ topic path is the one that most exactly categorizes a bookmark, the topic paths were sorted to their reverse order. For example, the topic path shown in Figure 6.7 was converted to `publications physics science`.

	1st	2nd	3rd	4th	5th	6th	7th to 11th
Top tag	9,44%	15,94%	12,67%	4,72%	3,28%	1,72%	0,81%
Top 3 tags	20,37%	27,55%	21,58%	14,29%	12,23%	6,21%	2,30%
Top 5 tags	28,32%	34,81%	27,72%	19,75%	16,42%	11,03%	3,69%
Top 10 tags	37,38%	44,53%	35,94%	27,08%	25,91%	18,28%	6,25%
Top 15 tags	44,30%	52,45%	43,17%	34,16%	32,12%	26,55%	8,93%
All tags	52,99%	62,55%	52,48%	46,34%	44,34%	40,34%	14,73%

Table 6.3: Comparison of categorization data from both sources, considering 1, 3, 5, 10, 15, or all tags for a given bookmark.

- All DMOZ topics were considered except of the subtopics of topic `Top/World`, which is the branch in the DMOZ hierarchy that builds the top concept for multi-lingual categories not defined in the English language.

In total, the test data retrieved for this experiment consists of 788 bookmarks together with all corresponding DMOZ topics and all tags and their frequency distribution from *del.icio.us*. The average topic path length of all DMOZ topics prepared as described above is 4,67. The average number of tags per bookmark is 24,59.

The following method is employed for comparing topics to tags. The topics are used as a reference and the tags are compared to them. If more than one topic is assigned to a bookmark, a separate comparison for each topic is performed. One comparison consists of several lookups, one for each entry of a topic path, e.g., for `publications physics science` three lookups are performed. The result of a lookup is either true in case of a match, or false. The results of the matching are shown in Table 6.3. The cases of including the top 1, top 3, top 5, top 10, top 15, or all tags for a given bookmark are considered. The column headers refer to the position of the topic in the reversed DMOZ path. The values in the cells show the percentage of successful matches.

It turns out that the leaf entries of the topic path match more often than the top entries. This is not surprising, since the top entries are very general (e.g., `Computers`, `Arts`, or `Science`). Only a few *del.icio.us* participants use very general terms to describe their items. When taking into account only the top tag, which is the one that most participants used for annotating, the highest percentage of matches is 15,94% for the second entry of each topic path. The more top tags are taken into account, the better the results. The fairest comparison is that of the top five tags, since this is the average number of the topic path length. In this case, the highest percentage of matches is 34,81% for the second entry of each topic path.

However, it can be seen from these values that the terms used for categorization are very different in both sources. Even when comparing all tags to each topic path entry (and hence conducting on average 24 comparisons of tags to one single topic), there is no match in 37,45% to 85,27% of the cases.

6.4 Summary

This chapter introduced folksonomies and their basic properties. A statistical analysis of the metadata provided by the *del.icio.us* social bookmarking service to gain a better understanding of its structure. The analysis led to the following insights:

- The principle of interest-based locality could not be found in the test sets retrieved. This finding has implications on the applicability of tagging data as a source of test data for simulating peer-to-peer environments. It is not possible to use the tagging data for this purpose without further preparation, such as removing bookmarks that are only sparsely present.
- The data from the hierarchical classification system provided by the DMOZ Open Directory Project and the tagging data provided by the *del.icio.us* social bookmarking service, which are very different in structure, also have only few similarities on the data level. This means that the classification method used has an impact on the annotations created by using it.

Chapter 7

Creating and Visualizing User Profiles Over Time

In traditional approaches to information filtering, the user has to explicitly create his or her profile, and manually keep the profile up to date. Taking advantage of the popularity of collaborative tagging systems, the recorded tagging behavior can be used to construct user profiles. This chapter presents the Add-A-Tag algorithm for profile construction from data from social bookmarking systems. Taking account of the structural and temporal nature of tagging data allows to create user profiles which (1) are represented as semantic networks, and (2) include both long-term and short-term interests of a user. The profiles can be used to guide a user's navigation, that is, to provide the user with personalized access to information resources.

Contents

7.1 Introduction

Collaborative tagging systems, also called folksonomies or social bookmarking services, allow their users to manage bookmarks online and to annotate them with free-text keywords – so-called *tags* – for improving re-discovery of information. The most well-known social bookmarking service, del.icio.us [143], was started in 2003. Many others have followed since. Consequently, the amount of annotated bookmark data available is immense and a lot of effort has already been devoted to the analysis of these data (see, e.g., [30, 59, 93]).

Most of the existing work employs a bookmark-centric view by evaluating the properties of the tagging data related to certain bookmarks, or a tag-centric view in which the tagging data related to certain tags is analyzed. The work presented here relies on a user-centric perspective, meaning that the focus is on those tags which have been employed by a certain user. The user-centric perspective allows to treat tagging data as a continuous stream of information about a user's interests. Since many users of a folksonomy stick with the same bookmark collection for years, these data contain a high amount of fine-grained information that can be utilized for creating user profiles. Unlike many other profile learning mechanisms, which usually rely on relevance feedback from the user [31], no additional user input is required. Moreover, since tagging data is time-based, it is possible to create user profiles that dynamically adapt to drifts in users' interests.

Profile creation. This work describes how to utilize data from a social bookmarking service to create user profiles that can in turn be used for Information Filtering (IF). There are several challenges inherent to dynamic user profiles that need to be addressed. Firstly, the profile should represent the most important parts of a users' behavior (that is, some compression, clustering or summarization needs to be performed). Both persistent long-term interests and transient short-term interests should co-exist in the profile. Secondly, the algorithm for updating a profile should be performant and scalable, and it should be applicable for bookmark collections of various sizes.

Profile visualisation and usage. After creating them, the profiles need to be presented to the user in such a way that it serves him or her a useful purpose. One such purpose is being able to view the structure and the contents of the profile to get an overview of a user's interests. Aggregated information about a user's bookmark collection is usually represented as a tag cloud, in which (1) all tags a user has employed so far are listed alphabetically and (2) the font size of a tag is set according to how often it has been used so far. However, tag clouds fail to represent two properties of a user's bookmark collection:

- They do not represent the relationships between the tags, which can be derived by using co-occurrence techniques.
- They do not consider that tagging data is time-based in their weighting of the relative importance of a tag.

The aim of this work is to create a visualisation method for user profiles that includes those two properties. In addition, just as tag clouds (or user profiles) can be used as an interface to access a user's bookmark collection, aggregated information about tagging behavior can also be exploited as an user interface for browsing *some other source of data*.

This chapter is organized as follows. Section 7.2 discusses various possibilities for creating a user profile out of a tag collection. This section provides the design rationale for the Add-A-Tag algorithm, which is formally defined in Section 7.3. Section 7.4 presents the experiments conducted in order to assess and evaluate the user profiles created with the Add-A-Tag algorithm, and the results of the experiments. Section 7.5 describes a visualization tool for dynamic user profiles and discusses the results of a small-scale user study that was carried out to gain some insight about users' acceptance of the method. Section 7.6 describes how to use the profiles for personalized access to data sources. Section 7.7 gives an overview of related work. Section 7.8 concludes.

7.2 Profile construction

The data available for profile construction is the following. Consider a user's bookmark collection consisting of a user-defined number of bookmarks. Each bookmark in the collection is composed of a title, a description, a URL, a date, and a set of tags. For creating the profile, the tags and their temporal ordering by increasing date are used. This section presents three different methods for profile construction. Section 7.2.1 illustrates the naive approach, Section 7.2.2 the co-occurrence approach, and Section 7.2.3 the adaptive approach. Examples are used to illustrate the approaches, the sample data for which are shown in Figure 7.1.

```
1  datamining rdf tools web
2  algorithms design geo java library programming
3  danger security pc tools web
4  ais security research article
5  bbc media rss social syndication
6  blog flickr fun geo metadata social uk web
7  ai turing teaching
8  ajax eclipse programming jsp spring tools uml web
9  geo google gps javascript tools web web2.0
10 owl rdf semanticweb web2.0
11 ai teaching
12 ai teaching
13 teaching ai
14 ontology opensource research security
15 design research robot ai teaching
```

Figure 7.1: Sample data. A user stores a collection of 15 bookmarks. These bookmarks are annotated with the tags shown as space-separated lists. The lists are ordered according to the time the corresponding bookmarks were added to the bookmark collection. The oldest one is shown first (line 1). Note that this is a very small data sample, for explanatory purposes. The entire bookmark collection for this user contains many more bookmarks.

#Occ.	Tag
5	web, ai, teaching
4	tools
3	security, research, geo
2	web2.0, rdf, social, programming, design
1	semanticweb, danger, rss, turing, metadata, jsp, fun, library, owl, article, ontology, google, eclipse, ajax, syndication, ais, javascript, bbc, robot, media, pc, uml, flickr, blog, java, spring, datamining, gps, opensource, uk, algorithms

Table 7.1: List of tags ranked by their number of occurrence

7.2.1 Naive approach

To construct a user profile out of these data, the task is to aggregate it in such a way that the interests of the user are reflected according to their intensity. The more often a certain tag is used, the higher the interest of the user in the corresponding topic. Hence, the most simple method for creating aggregated data for a user's bookmark collection is to count the occurrence of tags. The result of this computation is a list of tags which is ranked according to tag popularity. For the sample data (see Figure 7.1), the ranked tag list is shown in Table 7.1. It reveals that most tags have been used only once, and that there are only a few tags which were used often. The user profile can then be created by selecting the top k most popular tags from the ranked list. When selecting the top 3 tags, for example, the resulting user profile consists of the tags `web ai teaching`.

The benefit of this method is that it is very simple, and hence fast. However, it has some drawbacks. One problem is that those tags which are most often used tend to be not very specific (e.g., the tag `web` is a very general one). Moreover, the resulting profile consists of unlinked tags. Although the tagging data includes information about the relationships between those tags, these relationships are not included in the user profile. The co-occurrence approach presented in the next section tackles both these drawbacks.

7.2.2 Co-occurrence approach

The resulting profile is more specific if the profile creating mechanism not only considers which tags have been used, but rather which tags have been used in combination. This can be achieved by relying on the co-occurrence technique known from Social Network Analysis [135]. If two tags are used in combination (*co-occur*) by a certain user for annotating a certain bookmark, there is some kind of semantic relationship between them. The more often two tags are used in combination, the more intense this relationship is.

This is represented by a graph with labeled nodes and undirected weighted edges. The nodes correspond to the tags, and the edges correspond to the relationship between the tags. The graph is created by parsing the tags for all items in the bookmark collection. Each time a new tag is used, a new node for this tag is added to the graph. Each time a new

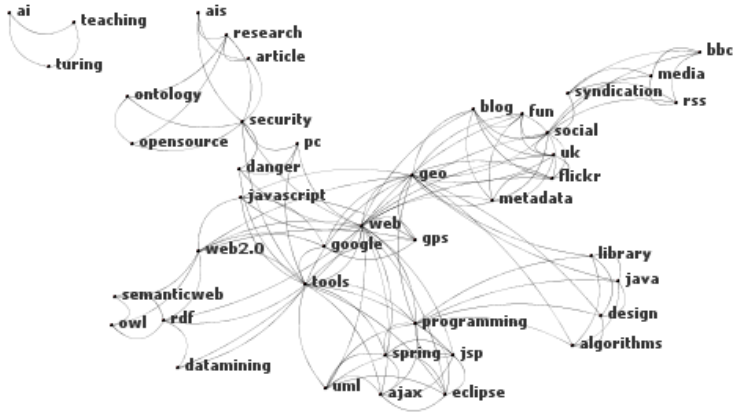

Figure 7.2: Co-occurrence network for the sample data shown in Figure 7.1. Two nodes are linked with an edge if the corresponding tags have been used in combination for annotating a bookmark. Edge weights are not shown. Note that although the amount of sample data is rather small, the resulting network is quite big.

combination of tags is used, a new edge with weight 1 between the corresponding nodes is created in the graph. If two tags co-occur again, the weight for the corresponding edge is increased by 1. Figure 7.2 shows the resulting graph when applying the co-occurrence approach to the sample data.

Co-occurrence techniques have been employed for diverse purposes. First and foremost, the folksonomy providers rely on it for computing related tags. Moreover, co-occurrence is also used for knowledge discovery from databases [33], for extracting light-weight ontologies from tagging data [93], or for tag recommendation [25, 142]. In this case, co-occurrence is used for one bookmark collection, only. The relationships between the tags are not the result of a community-driven process, but entirely created by one user instead. Hence, the relationships between the tags might not make sense to anyone except to the user who created them. However, in the case of user profile creation this is acceptable and even desirable, because for this task it is necessary to determine how the interests of a user are connected to each other, no matter how unorthodox these connections might be.

In the second step, a user profile is derived from the resulting graph by selecting the top k edges with the highest weights and their incident nodes. A ranked list of the weights of the resulting graph's edges for the sample data is shown in Table 7.2. Selecting the top 3 edges

Weight	Tag combination
4	ai - teaching, tools - web
2	geo - web, security - research

Table 7.2: Top 4 tag combinations ranked by their number of occurrence using the co-occurence technique (Section 7.2.2)

and their incident nodes for the user profile returns a graph with 5 nodes and the following edges: `ai-teaching tools-web geo-web`.

One drawback of the co-occurrence approach is that it does not include bookmarks that are annotated with a single tag. In order to overcome this issue, it would be necessary to combine it with the naive approach. The result would be a graph with weighted nodes and weighted edges. However, a decision was made against a combination of approaches, because the average percentage of bookmarks annotated with only one tag in the test set used for evaluation (see Section 7.4.1) is 8%. This can serve as an indicator that the average percentage of bookmarks annotated with only one tag on del.icio.us is small. Therefore, the loss of these data is acceptable in favor of a simpler method.

Another drawback of this approach is that the age of bookmarks and their temporal ordering is not considered. This issue is addressed by the adaptive approach, which is presented in the next section.

7.2.3 Adaptive approach

Since social bookmarking systems have been around for quite a while now, many of their users manage a rather big bookmark collection which they continuously have been adding items to for the time span of several months or even years. In the test set described in Section 7.4.1, the average lifetime of the bookmark collections is 607.7 days. Hence, the age information of the tagging data is important. It makes a difference if a user has used a certain tag and, therefore, specified a certain interest, one day or one year ago. In the co-occurrence approach, this information is not considered.

To include the age of the bookmarks in the user profile, the co-occurrence approach is extended with the evaporation technique known from ant algorithms [45]. Evaporation is a simple method to add time-based information to the weights of edges in a graph: Each time the profile graph is updated with tags from a newly added bookmark, the weight of each edge in the graph is decreased slightly by removing a small percentage of its current value.

Obviously, when creating the profile graph for the adaptive approach by parsing the tags for all items in the bookmark collection, it is necessary to start parsing from the oldest item and to process the items in the same temporal order as they were added to the bookmark collection. The nodes and edges in the resulting graph are the same as in the co-occurrence approach, but the weights of the edges are different. Since the user profile is created by extracting the top k edges with the highest weights and their incident nodes are from the profile graph, the resulting user profile will be different as well.

Applying the adaptive approach to the sample data apparently returns the same profile graph as before (Figure 7.2). However, the weights of the links in this graph are different. Table 7.3 lists the highest weighted edges in this graph. Selecting the top 3 edges and their incident nodes for the user profile returns a graph with 6 nodes and the following

Weight	Tag combination
3.83	ai - teaching
3.63	tools - web
1.89	security - research
1.85	geo - web

Table 7.3: Top 4 tag combinations for the adaptive approach with parameters $\alpha = 1.0, \beta = 1.0, \rho = 0.01$ (see Section 7.2.3 for details).

edges: `ai-teaching tools-web security-research`. The combinations `geo-web` and `security-research` occur the same number of times in the sample data. In the co-occurrence approach, the weight was the same for both combinations and therefore it was necessary to randomly select one of them for the profile. With the adaptive approach it is possible to detect that the latter combination has been used at a later point in time and can therefore be considered as currently more important to the user.

7.3 The Add-A-Tag algorithm

Now the adaptive algorithm that was described in Section 7.2.3 is formally defined. Section 7.3.1 describes how to create the profile graph. Section 7.3.2 defines how to extract the user profile from the profile graph.

7.3.1 Updating the graph

Consider a user u adding a bookmark item b tagged with tags $t_1, \ldots, t_n$ to his or her bookmark collection. The profile graph $G_u = (V, E)$ where $V = v_1, \ldots, v_n$ is the set of vertices (which correspond to tags) and $E = e_1, \ldots, e_n$ is the set of edges, is updated as follows.

Evaporation In the first step, the existing information in the graph is changed by applying the evaporation formula shown in Equation 7.1 to every edge $e_x \in E$

$$w_{e_x} \leftarrow w_{e_x} - \rho \cdot w_{e_x} \,, \tag{7.1}$$

where $\rho \in [0, 1]$ is a constant and w_{e_x} is the weight of edge e_x.

Reinforcement In the second step, the n new tags from bookmark b: $t_1, \ldots, t_n$ are added to the graph. For every combination $t_i t_j$ where $i, j \in 1, \ldots, n$ and $i < j$, the following procedure is executed:

1. Add a corresponding vertex v_x to graph G_u for every tag t_x ($x \in i, j$), if v_x does not exist.
2. Add an edge with weight α between vertex v_i and vertex v_j to graph G_u, if it does not yet exist. Constant α is a real number and $\alpha > 0$.

3. Otherwise, if an edge between vertex v_i and vertex v_j exists, increase its weight by β. Constant β is a real number and $\beta > 0$.

The procedure described above is executed each time the user adds an bookmark item to the bookmark collection.

7.3.2 Extracting the user profile

Extracting the user profile from the profile graph is defined as follows.

1. Create a ordered set E_s from $E = e_1, \ldots, e_n$. E_s contains all edges e_x ($x \in 1, \ldots, n$) from graph G_u ordered in decreasing order by their weights w_{e_x}.
2. Create set E_k by extracting the top k elements from set E_s, where k is a natural number and $k > 0$.
3. Create graph $G_{u'}$ which contains all edges from E_k and all vertices v_x from graph G_u which are incident to one of the edges in E_k.

The size of the user profile $G_{u'}$ is determined by the value chosen for parameter k.

7.4 Evaluation of profile adaptivity

Evaluating the adaptive aspects of the user profile creation mechanism is complicated because the amount of change in the user profile depends on the user's activity pattern as well as on the profile creation mechanism itself. In this section, an evaluation methodology is developed and applied to the co-occurrence approach and to the adaptive approach. The differences in the results demonstrate the effectiveness of the adaptive component in the Add-A-Tag algorithm. Section 7.4.1 describes the test data used. Section 7.4.2 introduces a metric for computing the distance between two versions of a user profile created at different points in time. Section 7.4.3 shows the results of applying the metric to the profiles created.

7.4.1 Test sets

For the experiments and the user study (see Section 7.5.1), the same test set consisting of six users' bookmark collections is used. Since this is user-related data, privacy concerns arise. Therefore, we refrain from retrieving a big test set from the del.icio.us service. Instead, we rely on a small but carefully selected test set which it is sufficient for the purposes described here. It consists of three small-sized (user 1, 2 and 3), two medium-sized (user 4 and 6) and one large bookmark collection (user 5). The owners of the bookmark collections included in the test set are personally known to at least one of the authors of this chapter, and they were explicitly asked for permission to retrieve and evaluate their personal tagging data.

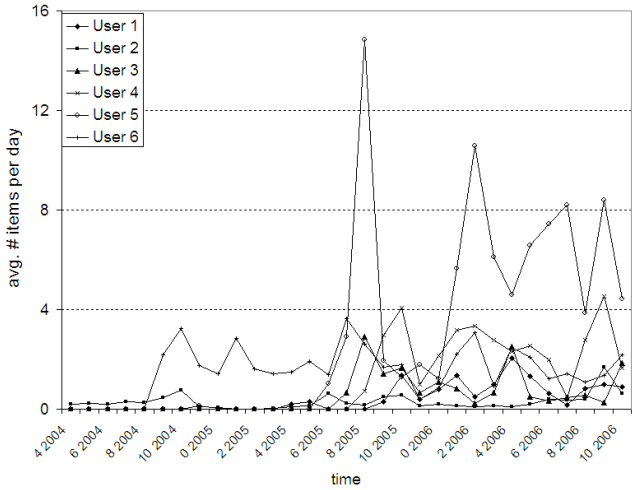

Figure 7.3: Average number of items added per month for the 6 sample users. It can be seen that the tagging activity depends on a user's mood and workload and does not follow any predictable patterns.

In addition, using a small test set also saves network resources and prevents overloading del.icio.us. Table 7.4 shows the basic properties of the six bookmark collections in the test set. It can be seen that the bookmark collections vary in number of days of use, number of items, number of tags, number of unique tags, and average number of tags per item. No proportional relationship between any of these figures can be found. In addition, as shown in Figure 7.3, the users' activity patterns are unpredictable. Some users maintain a reasonably constant level of activity, whereas others exhibit a bursty pattern depending on mood and workload.

7.4.2 Metrics

To evaluate the Add-A-Tag method it is necessary to determine the change of a profile over time. If the user profile of user u is computed at time t_1 and then again at time t_2, there needs to be a way to measure the difference (distance) between these two user profiles.

User	1	2	3	4	5	6
# of days in use	531	887	435	386	681	726
# of items	368	897	448	1112	2823	1362
# of tags	937	1331	2234	4703	16334	6343
# of unique tags	189	217	488	817	3451	1648
avg. # of tags per item	2.3	3.8	4.3	4.3	5.8	4.3
% of single tags	20%	13%	8%	1%	1%	2%

Table 7.4: Properties of the test set

Since measuring the distance between two graphs is a difficult and only partly solved issue [24], the problem can be simplified by mapping the graphs onto a simpler structure which only contains the information that is needed for the comparison. This structure is a set of edges ordered in decreasing weight order, because several methods for comparing ordered sets exist. A simple method for comparison would be to count the common members in both sets. However, counting common members does not consider the ranking of the set members. Therefore, the metric relies on the Kendall τ coefficient [1, 52] instead, which is a standard measure for comparing ordered sets that includes rank correlation.

The metric $dist(S_1, S_2)$ for the distance between two sets S_1 and S_2 based on the Kendall τ coefficient is defined as shown in Equations 7.2a to 7.2c. It obeys the rules for metrics (positiveness, reflexivity, symmetry, and triangle inequality; see [72]). The resulting values for $dist(S_1, S_2)$ are in the range between 0 and 1. The result is 0 if S_1 and S_2 are the same (that is, equally ranked). The result is 1 if S_1 and S_2 are in reverse order.

$$dist(S_1, S_2) = 1 - \frac{2 * \tau(S_1, S_2)}{n * (n-1)} \text{ , where} \tag{7.2a}$$

$$\tau(S_1, S_2) = \sum_{i,j \in P} \bar{\tau}_{i,j}(S_1, S_2) \text{ , and} \tag{7.2b}$$

$$\bar{\tau}_{i,j}(S_1, S_2) = \begin{cases} 0 & \text{if } i \text{ and } j \text{ are in same order in } S1 \text{ and } S2 \\ 1 & \text{otherwise} \end{cases} \tag{7.2c}$$

In Equation 7.2a, variable n is the size of the sets. In Equation 7.2b, P is the set of pairs of distinct elements in $S1$ and $S2$. The Kendall τ is applicable only for sets which have the same members and – consequently – are of same size. For this setting, this means that those set members that are present in only one of the sets need to be added to the other one. The missing set members are appended to the end of the set in order not to affect the ranking of the pairs.

7.4.3 Results

Now the user profiles for the six user's bookmark collections described in Section 7.4.1 are computed. The profile graph is created incrementally by adding the bookmark items in their temporal order, and – each time after adding all bookmarks that were created by the user within the time span of one week – by extracting the user profile from the profile graph. Using this procedure, a set of user profiles

$$\overline{G_{u'}} = \{G_{u'}^{t_x} | x = \{1, \ldots, n\}\}$$

for each user u and each week t_x is retrieved. In the next step, the metric $dist$ is applied to these data in order to assess the amount of change between the weekly snapshots of the

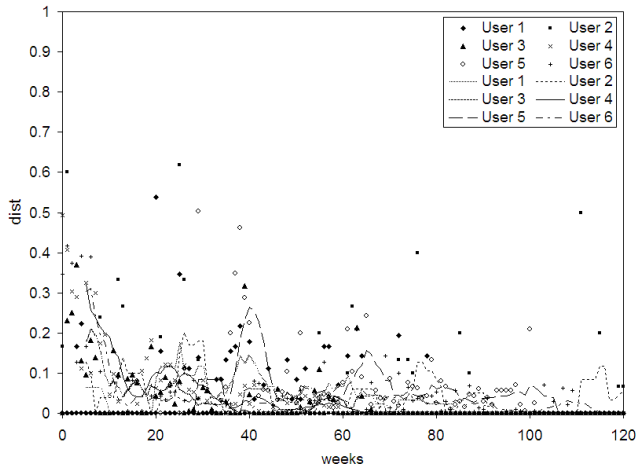

Figure 7.4: Co-occurrence approach ($\rho = 0$, $k = 20$)

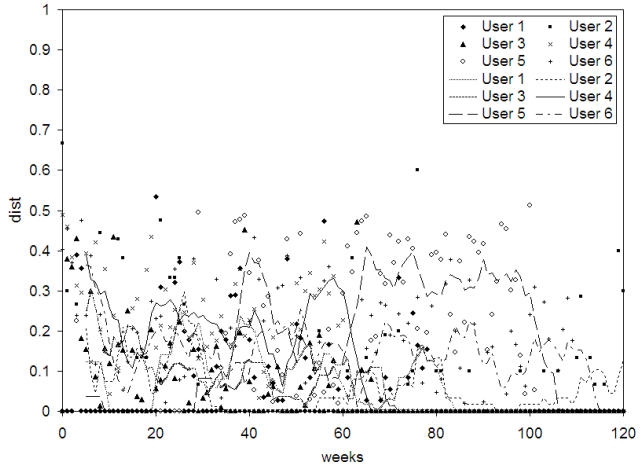

Figure 7.5: Add-A-Tag ($\alpha = 1.0$, $\beta = 1.0$, $\rho = 0.01$, $k = 20$)

profiles. For each user u, every user profile $G_{up}^{t_{x+1}}$ for week t_{x+1} is compared to the user profile $G_{up}^{t_x}$ for the previous week t_x. Figure 7.4 shows the results of this computation for the co-occurrence approach, with parameter k set to 20. Figure 7.5 shows the results for the Add-A-Tag approach, with parameter k set to 20, $\alpha = 1.0$, $\beta = 1.0$, and $\rho = 0.01$. In both figures, the data points show the metric values, and trend lines of type moving average with period 6 show the performance of the metric values over time for the different users.

In Figure 7.4, it can be seen that the degree of change in the user profiles decreases over time for the co-occurrence approach. Although the users are specifying new tag combina-

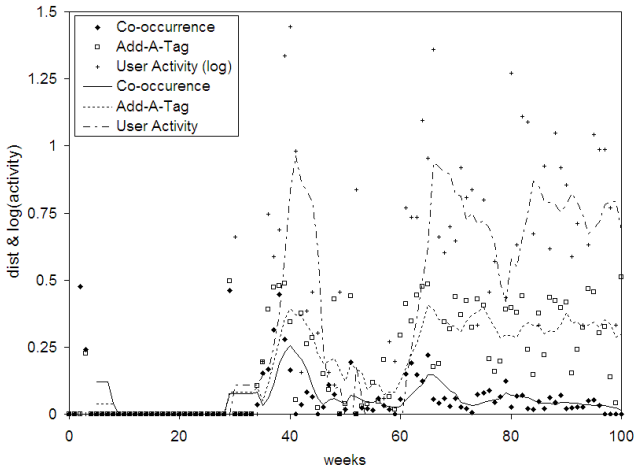

Figure 7.6: Direct comparison of approaches for user 5.

tions when adding bookmarks to their collections, the most often used tag combinations are dominant and tend to prevent newly arising tag combinations from being included in the profile. When comparing these results to those of the Add-A-Tag approach (Figure 7.5), one can see that (1) the degree of change in the user profile is higher in the Add-A-Tag approach, and that (2) when comparing the Add-A-Tag results for the 20-weeks time spans, the overall amount of change in the profiles over time is remarkably similar for every time span. This provides evidence that the Add-A-Tag approach meets its goal of adapting the profile to the interests of the user over time.

Figure 7.6 shows a direct comparison of the co-occurrence and the Add-A-Tag method for one user (user 5) together with the weekly activity of this user expressed as the logarithm of the average number of items added to the bookmark collection (cf. Figure 7.3). Again, trend lines of type moving average with period 6 are included. The dashed lines show the user's activity. It can be seen that both approaches exhibit a change pattern that is proportional to the user's activity pattern, but the Add-A-Tag approach's curve (1) shows a considerably higher amount of change and (2) fits better with the activity pattern. This is particularly the case for the results in the time span between week 80 and week 100, where the user's activity level is high and the Add-A-Tag approach appropriately reflects the activity pattern, but the co-occurrence approach puts too much emphasis on the most often used tag combinations and fails to adapt to the newly-used ones.

However, one drawback of the Add-A-Tag approach that became evident during the experiments is that the value for parameter ρ needs to be chosen very carefully. The higher the value for parameter ρ, the more emphasis is put on those items that were added to the bookmark collection recently. Choosing a value below 0.01 emphasizes often-used tag com-

binations, whereas values higher than 0.05 place the emphasis on newer tag combinations. Choosing a value in the range between 0.01 and 0.05 for ρ gives reasonable results with a balanced proportion between newly-used tag combinations and often-used tag combinations.

7.5 Profile visualization

This section presents a tool for visualizing the user profiles. It was developed for the user study (see Section 7.5.1) and allows a user to observe the changes in his or her user profile over time. The tool is implemented as a Java applet, and the graph visualization is based on the JUNG framework [77]. A screen shot of the visualization tool is shown in Figure 7.7.

The screen is divided into a main part and a control panel at the bottom of the screen. The control panel contains (1) radio buttons which allow the user to select one of the profile creation methods and (2) a button to start the visualization. The names of the profile creation methods were not mentioned in order not to influence the results of the user study. `Method A` refers to the naive approach, `Method B` to the Add-A-Tag approach, and `Method C` to the co-occurrence approach. After starting a visualization, the user profile is presented as an animation over time. The bottom panel shows a date, and the main part of the screen shows the state of the user profile at this date. A button allows the user to pause and resume the animation.

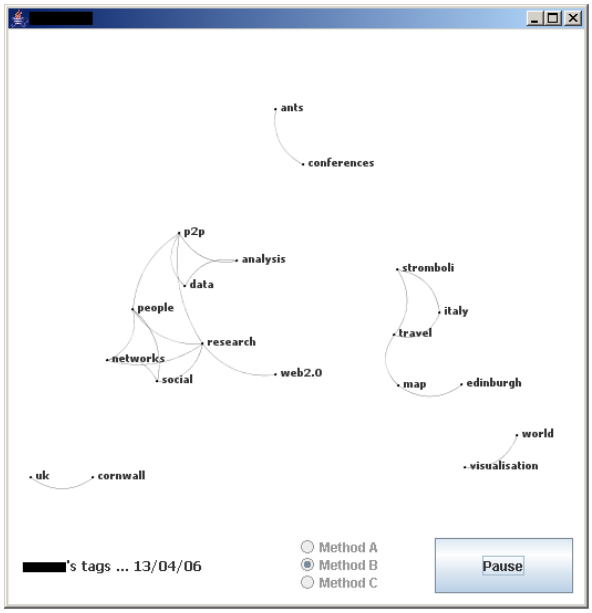

Figure 7.7: Visualization of a user profile

Visualizing the profile created with the naive approach is straightforward. To provide for intuitive observation of the dynamic changes, all nodes are moving using a "bubbling up" metaphor, which means that they enter the screen from the bottom and continuously move towards the top. If a tag is included in the user profile at one point of time, but not included in the next state, it vanishes from the screen. For the naive approach, this visualization method is sufficient. However, for the co-occurrence approach and for the Add-A-Tag approach, it is also necessary to visualize the edges between the nodes. The lengths of the edges between the nodes need to correspond to the edge's weights. The higher the weight, the shorter the length of the edge must be.

Basically, there are two approaches possible for visualizing these dynamic graphs. In the first approach, all nodes and edges that will be included in the profile at a certain point in time need to be known in advance. In the next step, a graph layout algorithm can be applied for calculating the positions of all the nodes and edges. During the animation, those nodes that are currently included in the profile are set to visible while all the others are set to invisible. The benefit of this approach is that the nodes do not move. However, the drawback is that the layout algorithm creates a visually pleasing layout for the complete graph, but the layouts of the different graph states shown over time are not optimized and tend to look quite ugly.

Therefore, it is necessary to adopt another approach by using an iteration-based graph visualization algorithm that incrementally optimizes the layout of the different graph states. To achieve this, the "bubbling up" metaphor is combined with the iteration-based spring embedder layout algorithm by Fruchterman and Reingold [53], in which the nodes repel or attract each other depending on the edges between them and on the weight of these edges. In addition, a minimum and a maximum length for the edges needs to be defined in order to prevent node labels being printed on top of each other and in order to avoid nodes being too far away from each other.

If a tag A that newly appears at the bottom of the screen has a connection to a tag B that is already shown on the screen, the spring embedder algorithm will cause tag B to move down on the screen and tag A to move up at the same time. Tag A and tag B will move towards each other until the edge between them has a length according to its weight. As a consequence, those components of the graph which evolve over time are vertically aligned in the center of the screen (e.g., the two components related to research and travel in Figure 7.7), because newly added tags make the older, related tags move down again. They refer to long- and mid-term interests of a user that are currently active.

On the contrary, those components that do not evolve but are included in the profile for the timespan of several weeks move to the top of the screen (e.g., `ants-conferences` in Figure 7.7). They refer to long-term interests of a user that are currently not active. The third category are those tags that move in from the bottom and vanish shortly after (e.g., `uk-cornwall` in Figure 7.7). They refer to short-term interests of a user.

7.5.1 User study

This section presents the results of a small user study conducted in order to get feedback about user's acceptance of the three different profile creation methods.

The same six users whose bookmark collections were used for the evaluation of profile adaptivity (see Section 7.4) were provided with the visualization tool described in the last section. They were asked to fill out a questionnaire in which they had to rate the different methods. The following scale was used for the rating: *Very good*, *Good*, *Fair*, *Poor*, *Very Poor*. In addition, the users were asked to rank the methods from 1 to 3 according to how much they liked them, and to justify both the choices for rating and ranking using free-form text. There was also some space for additional comments included. The application and the questionnaire were sent by email to the user, who also replied using email.

As an overall feedback, a *Wow!*-effect similar to the one described by Golder et al. [134] in their study of visualizing users' email archives could be observed. The users were generally pleased with the possibility of viewing aggregated information about their bookmark collection. Both being able to view the (1) relationships between the tags and the (2) trends over time were recognized and appreciated. In their feedback, many of the users mentioned that some tag combinations showed up in the profile at some point of time which they were able to track back to a specific event they could still remember. To cite one of the users: *"I kept having the feeling that by looking at the graph some sort of hidden meaning was coming out. The visualization style is definitely inspiring, for revealing non-obvious relations!"*

Although the participants in the user study were not provided with any information about the inner workings of the different methods, a majority of the users (4 of 6) were able to correctly identify and describe which kind of aggregation was performed for the different approaches, e.g, as one user expressed it: *"I guess Method 3 represents the average most used tags, while Method 2 the average most recently used tags."*

However, the users' preferences for the different methods turned out to be quite diverse. Two users ranked the co-occurrence approach first, two of them preferred the Add-A-Tag approach, and one of them ranked both of them equally. One of the users favored the naive approach. This may have been down to the visualization algorithm rather than the profile creation method: *"there was too much movement and too many changes on the screen, and the edges between them were detracting from the tags"*. The average rating for the naive approach was *Poor*. The average rating for both co-occurrence approach and the Add-A-Tag approach was *Good*. Several users mentioned that they perceived the difference between the co-occurrence approach and the Add-A-Tag approach as being rather small.

This leads to the conclusion that the preferred method of user profile creation is a very individual choice. For this reason, instead of creating a tool with a hard coded method, a preferable solution may be to allow the user to choose and configure his or her profile creation algorithm and visualization method. The popularity of the co-occurrence method

shows that users value the long term tag relationships in their profile; however they also appreciated that Add-A-Tag adapts better to recent changes. Allowing users to select the balance of long term and short term interests would provide control without over-burdening the user.

7.6 Profiles for personalized information access

In the following, the usage of the created profile for assisting users in navigating information resources is discussed. This section presents two example scenarios in which the created profile can be of benefit. Section 7.6.1 discusses the scenario of browsing the Web, Section 7.6.2 that of an annotated data source. Obviously, the profile can also be used for accessing a user's bookmark collection in the same manner as tag clouds are used for that task. Since visualising the relationships between the tags and the time-based aspects at the same time would cognitively overload users, in this section the focus is on visualising the relationships between the tags in the profile only.

7.6.1 Browsing the Web

If the person knows what he or she is looking for, e.g., when performing a search, knowing the user's additional interests other from the current one is of minor importance. On the contrary, knowing the user's interests is important if the person does not know what he or she is looking for, e.g, when browsing the Web for no specific purpose. In this case, the profile can be shown in the browser's sidebar or as a part of the Web page (similar to a navigation menu). When a tag occurs in the Web page the user is currently looking at, the tag can be highlighted in the profile, and clicking on it results in automatic scrolling to the position on the page on which the tag occurs. Another possibility is to highlight the terms in the Web page that are matched by tags in the profile (e.g., in the same manner as search strings are highlighted when viewing the Google cache of a search result). To improve the recall, string matching in combination with stemming can be used.

7.6.2 Browsing an annotated data source

The situation is more complex if a user wants to access a data source that is annotated with metadata. In this case, a matching needs to be performed. In general, matches are possible between (1) the profile and the content of the data source (as already discussed in the previous section), or between (2) the profile and the metadata of the data source as a description of the corresponding content. In the following the latter case is discussed, using the HP Technical Reports[1] as an example for such a data source. They comprise a document

[1] see `http://www.hpl.hp.com/techreports/`

collection annotated with metadata, such as title, author(s), date of publication, number of pages, abstract, and keywords. Only metadata that describe the contents of a resource can be used for the matching. Structural metadata (such as number of pages) is not helpful for matching, but can be exploited for additional navigational options in the interface.

In this scenario there are three possibilities for the matching between profile and data source. It can be matched (1) between tags and keywords, (2) between tags and abstracts, or/and (3) between tags and full text. If the tags in the profile are from very different domains than the domain of the data source, the matching may not be successful. However, at least a partial overlap between the user's overall interest and his or her current interests can be safely assumed. For the matching itself, string matching in combination with stemming is used. Since tags are most commonly in lower-case letters, whereas keywords are usually in capitalized letters, the matching needs must be performed in a case-insensitive way. The matching could be enhanced by backing the comparison algorithm with a thesaurus such as WordNet to link tags with synonym keywords. Another possibility would be to use clustering to find implicit relationships between tags or technical report keywords in order to allow a tag to be matched to a larger number of possible keywords.

A conceptual overview of the user interface layout is shown in Figure 7.8. The top left shows a representation of the profile. The user can select a tag from the profile to show only those resources in the main screen on the right that match with the selected tag. The bottom left shows additional navigation options (which are explained later).

The question of how to represent the profile needs to be addressed from two viewpoints. One of them is the *profile-centric viewpoint* which focuses on visualising the structure of the profile. For visualising the relationships between the tags in the profile, a spring embedder layout algorithm is used to position related tags next to each other. For showing the relative importance of a tag, the font sizes are set according to the relative importance of a tag, as in a tag cloud. However, since the profile will possibly contain tags for which no corresponding data can be found, it is also necessary to take a *data-centric viewpoint* by adapting the profile to the data that is available. Those tags for which no content exists are removed from the profile. For those tags for which corresponding resources exist, an optional possibility would be to print the number of resources that exist next to the tag name, as in faceted browsing. However, a decision was made against this option because combining font sizes (for relative importance of tags) and numbers (for number of resources) might be misleading to users.

The data source will contain content for which no corresponding tags are included in the profile. Therefore, using only the profile for navigation would make it impossible for the user to access that content. This can be avoided by offering additional navigation options to the user, such as a simple query interface. Moreover, providing additional context enables improved browsing of the data source. This is achieved using 2 navigation panels, shown in the bottom left of Figure 7.8. The first shows a list of keywords, each of which

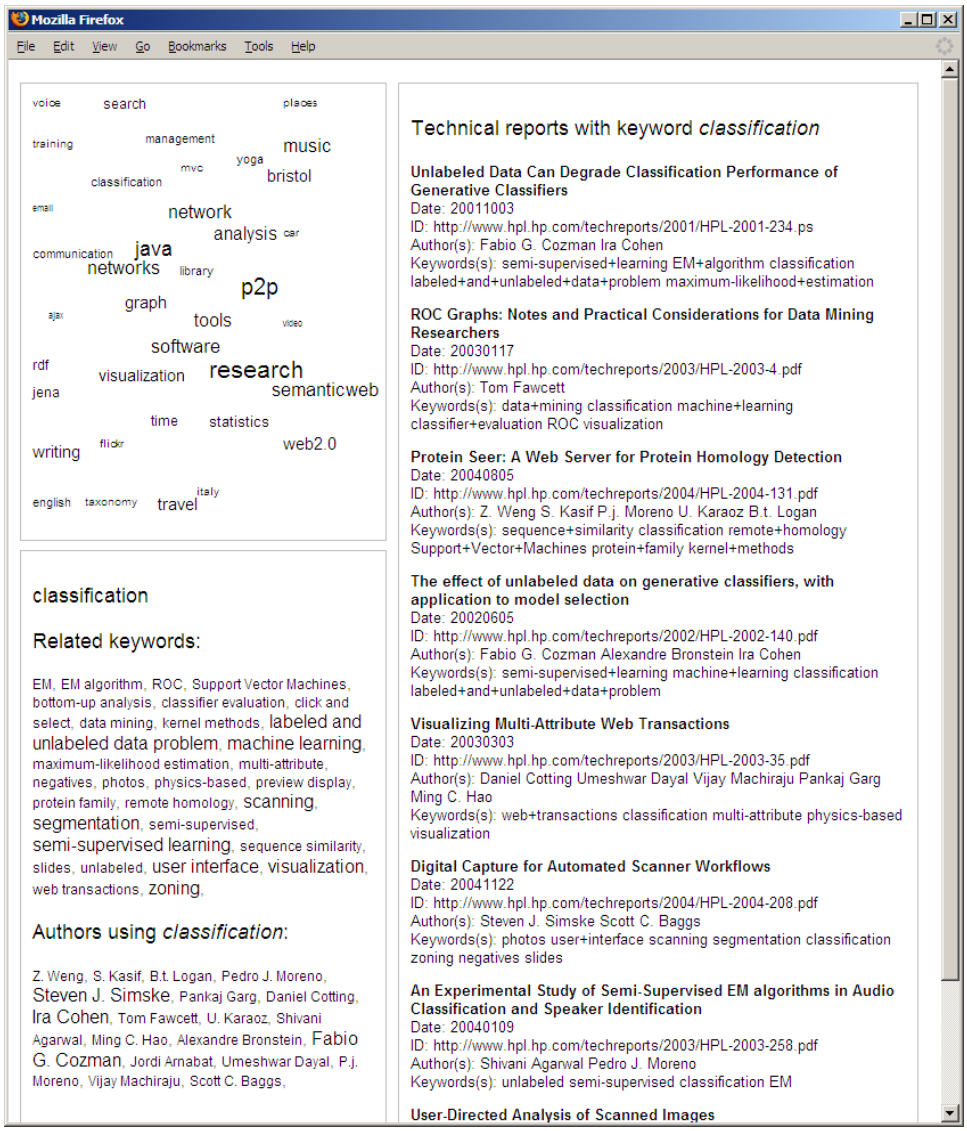

Figure 7.8: Interface layout. The top left shows the profile. The main screen (right) shows the resources that match with the tag from the profile selected by the user. The bottom left shows additional navigation options.

co-occurs with the selected keyword. Co-occurrence in this case means that the keywords in question are both attached to a single technical report. These related keywords are likely to cover between them many technical reports, including those which do not have any keyword matching a user's tags. The second navigation panel is similar, but this time shows all authors that have used the selected keyword to markup one or more of their technical reports. Again, the union of all technical reports authored by one of these people is likely

to include those that would not be covered by the profile alone. The layout of both these panels is similar. The font size represents the relative importance within the dataset (that is, the number of technical reports tagged with this person or keyword). Unlike the profile pane, co-occurrence patterns are not used to influence the relative positions.

We have also investigated the possibility of representing the user profile as a hierarchy. Such a structure would have advantages of simplicity and familiarity. Multiple inheritance issues (that is, a tag having 2 parents) do not preclude such a representation (a tag would just appear in 2 places in the hierarchy). We adopted an approach loosely similar to the one of Heymann and Garcia-Molina's [69], who use centrality measures to derive a taxonomy from tagging data based on the entirety of a folksonomy's tagging data. It consists of two steps which are executed for every subgraph. Firstly, the node with the highest betweenness centrality is determined as the root node of the tree. Secondly, Prim's algorithm [107] is used for computing the maximum spanning tree based on the weights. However, we have found that this approach is not well suited for profile representation of the type we are interested in. One problem is that the resulting tree can be quite unbalanced, which gives an unsatisfying browsing experience. In addition, nodes that frequently co-occur belong conceptually together and should exist at the same hierarchy level, e.g., the tags "semantic" and "web". The spanning tree approach forces these tags to exist at different levels, which is confusing for the user. For these reasons, we decided to go for the spring embedder layout style as described above.

7.7 Related work

There are several areas in which related work can be found. One of them is the analysis of tagging data. In this context, the work which is most directly related is ExpertRank [74] for measuring the expertise of a user in the context of a certain tag. ExpertRank can be viewed as an approach opposite to the Add-A-Tag approach. Instead of determining all areas of expertise for a given user, ExpertRank finds users that are knowledgeable in a certain area. Time-based aspects are not considered. Both [59] and [30] analyze tagging data by evaluating the data associated with a certain bookmark to show that the tag frequency distribution of the tags a certain bookmark is annotated with is stable over time, and that it can be modeled with stochastic processes. Mika [93] shows how to extract light-weight ontologies from tagging data, which is related to work on folksonomies and emergent semantics [4, 3]. Schmitz et al. [121] analyze the weight distributions of the co-occurrence networks and their connectivity for two different social bookmarking systems.

Creating (adaptive) user profiles is addressed in the area of (adaptive) information filtering. Allan [9] discusses incrementally applying relevance feedback to user profiles and how to cope with shifts in the user's interests, an issue he calls "query drift". Moreover, several nature-inspired approaches can be found in this research field. Nootropia [98] is a system

for adaptive information filtering based on an immune system inspired approach. Another approach is that of Tebri et al. [129], who use reinforcement learning for profile creation.

Several papers address visualization issues. TagLines [49] is a visualization of the most popular tags over time in Flickr [144]. It takes the entirety of Flickr tags into account. Since this is a huge amount of data, TagLines incorporates an efficient algorithm for computing the top k tags for time intervals of different sizes. Themail [134] is a system for visualizing a user's email archive. The focus is on visualizing the communication with one particular person over time. Two different timescales (yearly and monthly) are used. GUESS [8] supports visualization of dynamic graphs with the so-called tweening algorithm. Similar to the visualization tool presented earlier in this chapter, it creates an animation of the changes over time. This animation can be saved to QuickTime format. Hassan-Montero [66] suggests an improvement on tag cloud visualization by applying clustering algorithms to group related tags next to each other. Tag clustering is also addressed in [18].

Several applications for visualising a user's tag collection can be found on the Web. *Extisp.icio.us* [43], also described as "del.icio.us scattering" by its author, is a simple HTML-based visualisation that uses the size of a browser window. Just as for a tag cloud, the size of the tags depends on their popularity. The output looks similar to the one presented in Section 7.6.2, but unlike as in the Add-A-Tag approach, the tags are positioned randomly on the screen and the relationships between the tags are not taken into account. Since the tags are not filtered according to their popularity, the output is quite scattered. Some tags are printed in very small font size, and some on top of each other. *Revealicious* [105] provides three different ways for visualising a user's tag collection. One of them, called *SpaceNav*, is a method for graph exploration. Selecting a tag shows all its neighbors in a circle layout. Selecting a neighbor again brings up its neighborhood. The history of clicked tags is shown as a path. For selected tags, it is also shown how often it has been used and to how many other tags it is related. *TagsCloud* is an extended tag cloud in which hovering over a tag brings up related tags in color. *Grouper* does the same, but additionally groups all tag into the categories "most used", "commonly used", and "less used". *Delicious Soup* [149] shows all tags as dots in a two-dimensional grid. The size of a dot roughly corresponds to the number of times the tag has been used. Hovering over a dots shows textual information about how often and since when the tag has been used, together with the number of related tags. In addition, the related dots are highlighted. *Delicious Soup* could perhaps be improved by positioning related dots next to each other in the grid, and by including the tag for a related dot when highlighting it.

7.8 Summary

This chapter presented the Add-A-Tag algorithm for learning adaptive user profiles from bookmark collections, which is based on a combination of (1) the co-occurrence technique

for determining the relationships between tags and (2) an evaporation feature as known from ant algorithms for adapting the user profile to trends over time. The Add-A-Tag approach was evaluated in two ways. Firstly, by defining a metric appropriate for quantifying the amount of change over time, and by comparing the results of several methods for creating user profiles from bookmark collections. This has shown that the user profiles created with Add-A-Tag are adaptive in the sense that they change according to changes in tag usage in a continuous stream of tagging data. Secondly, a user study was conducted. For this purpose, a visualization method for dynamic graphs was designed and a prototype was implemented. Simply being able to view aggregated information about past tagging behavior was considered useful by the participants of the user study.

Chapter 8

Conclusion and future work

This thesis has dealt with the applicability of ant algorithms for two specific purposes: content-based search in unstructured peer-to-peer networks, and the extraction of adaptive user profiles from social bookmarking systems.

The SEMANT algorithm, which we have designed for search in unstructured peer-to-peer networks, consists of carefully selected constituents of the ant algorithms *Ant Colony System*, *AntNet*, and *AntHocNet*, which were combined and adapted to fit for the application purpose. We have shown through experimental evaluation that the SEMANT algorithm converges fast, delivers stable results, and outperforms the k-random walker reference algorithm in terms of resource usage as well as in terms of hit rate. For bootstrapping, probabilistic link selection provides for precise control of how much traffic is created in the network. However, the necessity to employ a different type of pheromone for every keyword that can occur in a query entails the management of rather large routing tables at each peer.

Moreover, as for every content-based approach to search, the algorithm's performance depends on how the content is distributed in the network. In this thesis we have investigated to which extent the performance is influenced by the content distribution, and whether it is possible to improve it by defining semantic relationships between the allowed keywords of a query. Under the assumption that all the resources in the network are annotated according to a taxonomy, we have presented an extended version of the SEMANT algorithm that integrates these information into the routing decisions. Our results have shown the dependency between content distribution and performance, and that the performance can be improved significantly by considering super-topic pheromone trails for routing. This is particularly the case if the content distribution is moderately dispersed.

Ant algorithms include an evaporation feature for integrating a time factor when incrementally creating solutions. Next to being useful for supporting the removal of outdated information from routing tables (see below), in this thesis we have identified another use case in which this feature is of great benefit: the learning of user profiles from tagging data. For this purpose we have designed the Add-A-Tag algorithm, which is based on a combination of an evaporation feature for adapting the user profile to trends over time and the co-occurrence technique for determining the relationships between tags. The resulting user

profiles are semantic networks derived from the structure of the tagging data, and they are adaptive in the sense that they change according to changes in tag usage in a continuous stream of tagging data.

Directions for future research

The contributions of this thesis may be further developed in several ways. Promising directions for future work are discussed in the following, separately for each part of the thesis.

Search in unstructured peer-to-peer networks. The experimental evaluations of the SEMANT algorithm were based on the assumption that the network topology, the query distribution, and the content distribution in the network are static. In this case, the information stored in the routing tables does not become outdated. However, real-world peer-to-peer systems are dynamic. The SEMANT algorithm is readily applicable for dynamic settings, because ant algorithms include an evaporation feature for removing outdated information. Therefore, one promising area of future work is to evaluate the performance of the SEMANT algorithm for dynamic scenarios, and to include further optimizations for dynamic settings based on the results of the evaluation. Real-world network traces that include data about the peers' behavior, such as churn rate and session time, and appropriate hardware resources are necessary to accomplish this task.

Extraction of adaptive user profiles from tagging data. For the Add-A-Tag algorithm, next to improving the quality of the user profiles, another area of future work is to investigate additional scenarios for profile usage. For the former, receiving feedback from a large user population would be beneficial. In case of the latter, up to now we have shown how to visualize the profiles' changes over time to the user, and we have investigated how the user profiles can provide personalized access to annotated textual information sources. This approach can be generalized to provide personalized access to tagged audiovisual sources, such photos in *Flickr*, videos in *YouTube*, or music files in *Last.fm*. Since the traditional search techniques from information retrieval can not be applied to non-textual content, providing alternative approaches to search based on tag matching and the relationship between tags is necessary. Other possibilities to make use of the network-like structure of the profile would be tag recommendation, or ranking of search results. No matter in which context a user employs a certain tag, as soon as it matches with a tag in the profile, those tags that have a relationship with the tag used can be employed for supporting the user's current task.

We strongly believe that emergent and self-organizing phenomena observed from nature are of great benefit for the field of 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 [113] to mark unrewarding paths. Augmenting ant algorithms with this feature would expand them with a negative feedback function.

Bibliography

[1] Hervé Abdi. *Encyclopedia of Measurement and Statistics*, chapter Kendall rank correlation, pages 508–510. Sage, 2007.

[2] Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. P-Grid: a self-organizing structured P2P system. *SIGMOD Record*, 32(3):29–33, 2003.

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

[4] Karl Aberer, Phillippe Cudre-Mauroux, Aris M. Ouksel, Tiziana Catarci, Mohand-Said Hacid, Arantza Illarramendi, Vipul Kashyap, Massimo Mecella, Eduardo Mena, Erich J. Neuhold, Olga De Troyer, Thomas Risse, Monica Scannapieco, Flix Saltor, Luca de Santis, Stefano Spaccapietra, Steffen Staab, and Rudi Studer. Emergent Semantics Principles and Issues. In *Procceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, March 2004.

[5] Adam Mathes. Folksonomies – Cooperative Classification and Communication Through Shared Metadata. `http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html`, December 2004.

[6] Lada A. Adamic, Rajan M. Lukose, and Bernardo A. Huberman. *Handbook of Graphs and Networks: From the Genome to the Internet*, chapter Local Search in Unstructured Networks. Wiley-VCH, 2003.

[7] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in Power-Law Networks. *Phys. Rev. E*, 64, 2001.

[8] Eytan Adar. GUESS: A Language and Interface for Graph Exploration. In *Proceedings of the International Conference on Conference on Human Factors in Computing Systems (CHI2006)*, April 2006.

[9] James Allan. Incremental Relevance Feedback for Information Filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in*

Information Retrieval (SIGIR '96), pages 270–278, New York, NY, USA, August 1996. ACM Press. ISBN 0-89791-792-8.

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

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

[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)*. IEEE, July 2002.

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

[15] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking Up Data in Peer-to-Peer Systems. *Communications of the ACM*, 46(2):43–48, 2003.

[16] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, October 1999.

[17] Benjamín Barán and Rubén Sosa. A New Approach for AntNet Routing. In *International Conference on Computer Communication and Networks (IEEE ICCCN-2000)*, October 2000.

[18] Grigory Begelman, Philipp Keller, and Frank Smadja. Automated Tag Clustering: Improving Search and Exploration in the Tag Space. In *Proceedings of the Collaborative Web Tagging Workshop, 15th International World Wide Web Conference (WWW 2006)*, May 2006.

[19] Gerardo Beni and Jing Wang. Swarm intelligence. In *Proceedings of the 7th Annual Meeting of the Robotics Society of Japan*, pages 425–428, 1989.

[20] Martin Bernauer, Gerti Kappel, and Elke Michlmayr. Traceable Document Flows. In *Proceedings of the 2nd International Workshop on Web Semantics (WebS), DEXA2004*, September 2004.

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

[22] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30(1-7):107–117, 1998.

[23] Rodney A. Brooks. Elephants Don't Play Chess. *Robotics and Autonomous Systems*, 6: 3–15, 1990.

[24] Fred Buckley and Frank Harary. *Distance in graphs*. Addison-Wesley, 1990.

[25] Andrew Byde, Hui Wan, and Steve Cayzer. Personalized Tag Recommendations via Social Network and Content-based Similarity Metrics. In *Proceedings of the International Conference on Conference on Weblogs and Social Media (ICWSM'06)*, March 2006.

[26] Gianni Di Caro and Marco Dorigo. AntNet: A Mobile Agents Approach to Adaptive Routing. Technical Report IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.

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

[28] 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*, volume 3242 of *Lecture Notes in Computer Science*. Springer, September 2004.

[29] Miguel Castro, Manuel Costa, and Antony Rowston. Debunking some Myths about Structured and Unstructured Overlays. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05)*, May 2005.

[30] Ciro Cattuto. Semiotic Dynamics in Online Social Communities. *Eur. Phys. J. C*, 46 (s02):33–37, 2006.

[31] Ugur Cetintemel, Michael J. Franklin, and C. Lee Giles. Self-adaptive user profiles for large-scale data delivery. In *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, page 622, Washington, DC, USA, February/March 2000. IEEE Computer Society. ISBN 0-7695-0506-6.

[32] B. Chandrasekaran, John R. Josephson, and Richard Benjamins. What are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems and Their Applications*, 14(1): 20–26, 1999.

[33] Hsinchun Chen and Kevin J. Lynch. Automatic Construction of Networks of Concepts Characterizing Document Databases. *IEEE Transactions On Systems, Man, and Cybernetics*, 22(5):885–902, September/October 1992.

[34] Vicent Cholvi, Pascal Felber, and Ernst Biersack. Efficient Search in Unstructured Peer-to-Peer Networks. *European Transactions on Telecommunications*, 15:535–548, 2004.

[35] Vicent Cholvi, Pascal Felber, and Ernst Biersack. Efficient Search in Unstructured Peer-to-Peer Networks. In *Proceedings of SPAA'04*, June 2004.

[36] Clay Shirky. Ontology is Overrated: Categories, Links, and Tags. `http://shirky.com/writings/ontology_overrated.html`, 2005.

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

[38] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed Optimization by Ant Colonies. In *Proceedings of the 1st European Conference on Artificial Life*, pages 134–142. Elsevier, December 1992.

[39] Brian F. Cooper. Guiding Queries to Information Sources with InfoBeacons. In *Middleware 2004, ACM/IFIP/USENIX International Middleware Conference*, volume 3231 of *Lecture Notes in Computer Science*, pages 59–78. Springer, October 2004. ISBN 3-540-23428-4.

[40] Arturo Crespo and Hector 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.

[41] Arturo Crespo and Hector Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical report, Computer Science Department, Stanford University, October 2002.

[42] Stephen Daniel, James Ellis, and Tom Truscott. USENET - A General Access UNIX Network. Technical report, Duke University, 1980.

[43] Kevan Davis. extisp.icio.us. `http://kevan.org/extispicious/`.

[44] Hadhami Dhraief, Alfons Kemper, Wolfgang Nejdl, and Christian Wiesner. *Databases, Information Systems, and Peer-to-Peer Computing*, volume 3367 of *Lecture Notes in Computer Science*, chapter Processing and Optimization of Complex Queries in Schema-Based P2P-Networks, pages 31–45. Springer, February 2005.

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

[46] Marco Dorigo, Gianni Di Caro, and Luca Maria Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(2):137–172, 1999.

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

[48] Marco Dorigo and Thomas Stützle. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management*, chapter The Ant Colony Optimization Metaheuristic: Algorithms, Applications and Advances, pages 251–285. Kluwer Academic Publishers, 2002.

[49] Micah Dubinko, Ravi Kumar, Joseph Magnani, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. Visualizing Tags over Time. In *Proceedings of the 15th International World Wide Web Conference (WWW'06)*, pages 193–202, New York, NY, USA, 2006. ACM Press.

[50] Frederick Ducatelle. Ant Colony Optimisation for Bin Packing and Cutting Stock Problems. Master's thesis, School of Artificial Intelligence, Division of Informatics, University of Edinburgh, 2001.

[51] Frederick Ducatelle, Gianni Di Caro, and Luca Maria Gambardella. Using Ant Agents to Combine Reactive and Proactive Strategies for Routing in Mobile Ad Hoc Networks. *International Journal on Computational Intelligence and Applications (IJCIA), Special Issue on Nature-Inspired Approaches to Networks and Telecommunications*, 2005.

[52] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing Top k Lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, October 2003.

[53] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[54] Luca Maria Gambardella and Marco Dorigo. Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. In *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, July 1995.

[55] Luca Maria Gambardella and Marco Dorigo. An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem. *INFORMS Journal on Computing*, 12(3):237–255, 2000.

[56] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. *New Ideas in Optimization*, chapter MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, pages 63–76. McGraw-Hill, 1999.

[57] Matthew E. Gaston and Marie desJardins. Social Network Structures and Their Impact on Multi-Agent System Dynamics. In *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference*, pages 32–37. AAAI Press, 2005.

[58] 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*, pages 69–93, July 1990.

[59] Scott A. Golder and Bernardo A. Huberman. The Structure of Collaborative Tagging Systems. *Journal of Information Science*, 32(2):198–208, April 2006.

[60] Sabine Graf. Auswahl und Implementierung eines Ameisenalgorithmus zur Steuerung von Patienten im Planspiel INVENT. Master's thesis, University of Vienna, March 2003.

[61] Pierre-Paul Grassé. La reconstruction du nid et les coordinations inter-individuelles chez bellicoitermes natalenis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insect Sociaux*, 6:41–81, 1959.

[62] Walter J. Gutjahr. ACO Algorithms with Guaranteed Convergence to the Optimal Solution. *Information Processing Letters*, 82:145–153, 2002.

[63] Tony Hammond, Timo Hannay, Ben Lund, and Joanna Scott. Social Bookmarking Tools (I): A General Review. *D-Lib Magazine*, 11(5), April 2005.

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

[65] Matthew Harren, Joseph M. Hellerstein, Ryan Huebsch, Boon T. Loo Loo, Scott Shenker, and Ion Stoica. Complex Queries in DHT-based Peer-to-Peer Networks. In *Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[66] Yusef Hassan-Montero and Victor Herrero-Solana. Improving Tag-Clouds as Visual Information Retrieval Interfaces. In *Proceedings of the International Conference on Multidisciplinary Information Sciences and Technologies (InSciT2006)*, October 2006.

[67] Manfred Hauswirth and Schahram Dustdar. *Software-Architekturen für Verteilte Systeme*, chapter Peer-to-Peer: Grundlagen und Architektur, pages 161–197. Springer, August 2003.

[68] Manfred Hauswirth and Schahram Dustdar. Peer-to-Peer: Grundlagen und Architektur. *Datenbank-Spektrum*, 13, 2005.

[69] Paul Heymann and Hector Garcia-Molina. Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Technical report, Computer Science Department, Stanford University, April 2006.

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

[71] Mikael Holmqvist. Experiential Learning Processes of Exploitation and Exploration Within and Between Organizations: An Empirical Study of Product Development. *Organization Science*, 15(1):70–81, 2004.

[72] Sergei Ivanov, Dmitri Burago, and Yuri Burago. *A Course in Metric Geometry*, volume 33 of *Graduate Studies in Mathematics*. American Mathematical Society, 2001.

[73] Mano Jean-Pierre, Bourjot Christine, Lopardo Gabriel, and Glize Pierre. Bio-inspired Mechanisms for Artificial Self-organised Systems. *Informatica*, 30(1):55–62, 2006.

[74] Ajita John and Doree Seligmann. Collaborative Tagging and Expertise in the Enterprise. In *Proceedings of the Collaborative Web Tagging Workshop, 15th International World Wide Web Conference (WWW 2006)*, May 2006.

[75] Steven Johnson. *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. Scribner, September 2002. ISBN 0684868768.

[76] Sam Joseph and Takashige Hoshiai. Decentralized Meta-Data Strategies: Effective Peer-to-Peer Search. *IEICE Transactions on Communications*, E86-B(6):1740–1753, June 2003.

[77] Joshua O'Madadhain and Danyel Fisher and Tom Nelson. JUNG: Java Universal Network/Graph Framework. `http://jung.sourceforge.net`.

[78] Vana Kalogeraki, Dimitrios 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.

[79] Jon M. Kleinberg. Navigation in a small world. *Nature*, 406:845, August 2000.

[80] William Kocay and Donald L. Kreher. *Graphs, Algorithms and Optimization*. Discrete Mathematics and its Applications. Chapman and Hall/CRC, 2005. ISBN 1-58488-396-0.

[81] Charles Kozierok. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005.

[82] Thomas Halva Labella, Marco Dorigo, and Jean-Luc Deneubourg. Division of labour in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):4–25, 2006.

[83] Eugene L. Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley Series in Discrete Mathematics & Optimization. John Wiley & Sons, September 1985. ISBN 0471904139.

[84] Roger Lewin. *Complexity: Life at the Edge of Chaos*. Macmillan, 1992.

[85] Boon Thau Loo, Ryan Huebsch, Ion Stoica, and Joseph M. Hellerstein. The Case for a Hybrid P2P Search Infrastructure. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS2004)*, February 2004.

[86] Alexander 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.

[87] Alexander Löser. *Adaptive Overlays in Peer-to-Peer Netzwerken*. PhD thesis, Technische Universität Berlin, 2005.

[88] Alexander Löser, Christoph Tempich, Bastian Quilitz, Steffen Staab, Wolf-Tilo Balke, and Wolfgang Nejdl. Searching Dynamic Communities with Personal Indexes. In *Proceedings of the 4th International Semantic Web Conference*, November 2005.

[89] Ben Lund, Tony Hammond, Martin Flack, and Timo Hannay. Social Bookmarking Tools (II): A Case Study – Connotea. *D-Lib Magazine*, 11(4), April 2005.

[90] 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*, pages 84–95, June 2002.

[91] Martin Mauve, Jörg Widmer, and Hannes Hartenstein. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. *IEEE Network Magazine*, 15(6):30–39, November 2001.

[92] Daniel A. Menascé and Lavanya Kanchanapalli. Probabilistic Scalable P2P Resource Location Services. *SIGMETRICS Performance Evaluation Review*, 30(2):48–58, 2002.

[93] Peter Mika. Ontologies are us: A unified model of social networks and semantics. In *Proceedings of the 4th International Semantic Web Conference (ISWC 2005)*, November 2005.

[94] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-Peer Computing. Technical report, HP Lab, 2002.

[95] Roberto Montemanni, Luca Maria Gambardella, Andrea E. Rizzoli, and Alberto V. Donati. Ant Colony System for a Dynamic Vehicle Routing Problem. *Journal of Combinatorial Optimization*, 10:327–343, December 2005.

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

[97] Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM 2001*, San Diego, CA, September 2001.

[98] Nikolaos Nanas, Anne de Roeck, and Victoria Uren. Immune-Inspired Adaptive Information Filtering. In *Proceedings of the 5th International Conference on Artificial Immune Systems (ICARIS 2006)*, September 2006.

[99] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA: a P2P networking infrastructure based on RDF. In *Proceedings of the 11th International World Wide Web Conference (WWW02)*, May 2002.

[100] Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario Schlosser, Ingo Bruckhorst, and Alexander Löser. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proceedings of the 12nd International World Wide Web Conference (WWW2003)*, May 2003.

[101] Netscape Communications Corporation. DMOZ Open Directory Project. `http://www.dmoz.org`, 2005.

[102] Cheuk Hang Ng, Ka Cheung Sia, and Chi Hang Chan. Advanced Peer Clustering and Firework Query Model in the Peer-to-Peer Network (Poster paper). In *Proceedings of the 12th World Wide Web Conference (WWW2003)*, May 2003.

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

[104] Arno Pany. Simulant - Design and Implementation of a Simulation Environment for Ant Algorithms in Peer-to-Peer Networks. Master's thesis, Vienna University of Technology, May 2006.

[105] Sebastien Pierre, Olivier Zitvogel, and Yann Klis. Revealicious. `http://www.ivy.fr/revealicious/`.

[106] Luca Pireddo and Mario 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.

[107] Robert C. Prim. Shortest connection networks and some generalisations. *Bell System Technical Journal*, 36, 1957.

[108] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-411-8.

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

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

[111] John Risson and Tim Moors. Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. *Computer Networks*, 50(17):3485–3521, December 2006.

[112] Jordan Ritter. Why gnutella can't scale. no, really. Technical report, Darkridge Security Solutions, February 2001.

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

[114] Jason Rohrer. MUTE: Simple, Anonymous File Sharing. `http://mute-net.sourceforge.net/`, 2005.

[115] Reuven Y. Rubinstein. *Simulation and the Monte Carlo Method*. Wiley, 1981. ISBN 0-471-08917-6.

[116] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of the Multimedia Computing and Networking Conference (MMCN)*, January 2002.

[117] Nima Sarshar, P. Oscar Boykin, and Vwani P. Roychowdhury. Percolation Search in Power Law Networks: Making Unstructured Peer-To-Peer Networks Scalable. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, August 2004.

[118] Nima Sarshar, P. Oscar Boykin, and Vwani P. Roychowdhury. Scalable Percolation Search in Power Law Networks. Technical report, Department of Electrical Engineering, University of California, Los Angeles, 2004.

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

[120] Christoph 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.

[121] Christoph Schmitz, Miranda Grahl, Andreas Hotho, Gerd Stumme, Ciro Cattuto, Andrea Baldassarri, Vittorio Loreto, and Vito D. P. Servedio. Network Properties of Folksonomies. In *Proceedings of the Workshop on Tagging and Metadata for Social Information Organization, 16th World Wide Web Conference (WWW2007)*, May 2007.

[122] Ruud Schoonderwoerd, Owen Holland, Janet Bruten, and Leon Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Journal on Adaptive Behavior*, 5:169–207, 1996.

[123] 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.

[124] FidoNet Coordinator Structure. FidoNet Policy Document. Technical report, FidoNet, 1989. http://www.fidonet.us/policy4.html.

[125] Thomas Stützle and Marco Dorigo. A short convergence proof for a class of ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 6(4):358–365, August 2002.

[126] Thomas Stützle and Holger Hoos. Improvements on the Ant-System: Introducing the MAX-MIN Ant System. In *Proceedings of the International Conference of Artificial Neural Networks and Genetic Algorithms*, pages 245–249. Springer, 1997.

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

[128] Katia P. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[129] Hamid Tebri, Mohand Boughanem, and Claude Chrisment. Incremental profile learning based on a reinforcement method. In *Proceedings of the ACM Symposium on Applied Computing (ACM SAC)*, March 2005.

[130] 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.

[131] Dimitrios Tsoumakos and Nick Roussopoulos. A Comparison of Peer-to-Peer Search Methods. In *Proceedings of the 6th International Workshop on the Web and Databases (WebDB)*, June 2003.

[132] Dimitrios Tsoumakos and Nick Roussopoulos. Adaptive Probabilistic Search for Peer-to-Peer Networks. In *3rd International Conference on Peer-to-Peer Computing (P2P'03)*, pages 102–109, September 2003.

[133] Yannis Tzitzikas, Carlo Meghini, and Nicolas Spyratos. Taxonomy-based conceptual modeling for peer-to-peer networks. In *Proceedings of the 22nd International Conference on Conceptual Modeling (ER'2003)*, October 2003.

[134] Fernanda Viégas, Scott Golder, and Judith Donath. Visualizing Email Content: Portraying Relationships from Conversational Histories. In *Proceedings of the International Conference on Conference on Human Factors in Computing Systems (CHI2006)*, April 2006.

[135] Stanley Wasserman and Katherine Faust. *Social Network Analysis*. Cambridge University Press, Cambridge, 1994.

[136] Duncan J. Watts and Steven Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[137] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. MIT Press, 1948.

[138] Wikipedia. Web 2.0 Introduction. `http://en.wikipedia.org/wiki/Web_2.0`, 2007.

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

[140] Tom De Wolf and Tom Holvoet. Emergence Versus Self-Organisation: Different Concepts but Promising When Combined. In *Proceedings of the 3rd International Workshop on Engineering Self-Organising Applications (ESOA'05), 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, pages 1–15, May 2005.

[141] Jie Wu and Karl Aberer. Swarm Intelligent Surfing in the Web. In *Proceedings of the 3rd International Conference on Web Engineering*, July 2003.

[142] Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the Semantic Web: Collaborative Tag Suggestions. In *Proceedings of the Collaborative Web Tagging Workshop, 15th International World Wide Web Conference (WWW 2006)*, May 2006.

[143] Yahoo! Inc. Del.icio.us Social Bookmarking Service. `http://del.icio.us`.

[144] Yahoo! Inc. Flickr Photosharing. `http://flickr.com`.

[145] Beverly Yang and Hector Garcia-Molina. Improving Search in Peer-to-Peer Systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, July 2002.

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

[147] Yong Yang, Rocky Dunlap, Michael Rexroad, and Brian F. Cooper. Performance of Full Text Search in Structured and Unstructured Peer-to-Peer Systems. In *Proceedings of IEEE INFOCOM06*, April 2006.

[148] Shanyu Zhao, Daniel Stutzbach, and Reza Rejaie. Characterizing Files in the Modern Gnutella Network: A Measurement Study. In *Proceedings of SPIE/ACM Multimedia Computing and Networking*, January 2006.

[149] Olivier Zitvogel. Delicious Soup. Available at `http://www.zitvogel.com/deliciousSoup/`.

Curriculum Vitae

Contact Information

Schikanedergasse 6/6
1040 Vienna
Austria

Voice: ++43 6769607788
E-mail: michlmayr@wit.tuwien.ac.at
Web: http://wit.tuwien.ac.at/people/michlmayr/

General Information

Date of Birth: 13/08/1976, Linz, Austria.
Languages: English (fluent), German (native).

Education

PhD studies in Computer Science 2003 - 2007
Vienna University of Technology, Austria
Advisors: Gerti Kappel, Wolfgang Nejdl (University of Hannover, Germany)

MSc Computer Science (Diplomingenieurin) 1996 - 2002
Vienna University of Technology, Austria
Advisors: Clemens Kerer, Mehdi Jazayeri
Thesis: Flexible Content Management for the LoL@ UMTS Application.
Graduated with honors

Student of Mathematics, Vienna University of Technology 1994 - 1996
Not finished

Secondary school, Bundesrealgymnasium Steyr 1986 - 1994
Final exam with honors

Experience

Intern September 2006 - March 2007
Hewlett-Packard Labs (Bristol, UK)
Java programming.

- Designed an algorithm for learning adaptive user profiles from tagging data based on co-occurrence of tags and a decay factor to consider time information in the weighting of the relative importance of a tag in the profile.
- Implemented a tool for visualizing the profile's changes over time, and an improved version of a tag cloud in which a spring embedder layout algorithm is used to position related tags next to each other on the screen. This representation of the user profile can be used for personalized browsing of annotated data sources.
- Advisors: Steve Cayzer, Paul Shabajee

Intern August 2005 - November 2005
IBM Research/OTI Labs - Eclipse (Zürich, CH)
Java programming, JUnit testing

- Implemented a tool for import and export of Bugzilla bugs for the Jazz platform (an extension of Eclipse for improved team collaboration). The tool includes a wizard for manual import and a headless UI for batch import. It is used as a test client for the platform's underlying framework.
- Advisors: Christof Marti, Erich Gamma

Software Engineer June 2001 - June 2003
ftw. Telecommunications Research Center Vienna (Vienna, AT)
Java programming (June 2001 - January 2003)

- Designed and implemented a HTTP/HTML-based content delivery system for a prototype of a location-based mobile tourist guide for UMTS networks. My part in this project was to integrate data from various sources (text, images, video, sound, location data) and to present it according to the client's device features and requirements, such as screen size.

Python programming (January 2003 - June 2003)

- Designed and implemented a Zope-based content management system

Trainer for Adult Education (Part time) May 2000 - May 2003
WIFI (Vienna, AT), telm@ (Groß Siegharts, AT)
Designed and taught introductory and advanced courses on

- Linux (installation, administration)
- Design and implementation of relational databases (with MySQL)
- Introduction to programming (with Perl)

Books (in German)

1. Horst Eidenberger, Elke Michlmayr, "Programmieren lernen und Probleme lösen mit Perl" (Programming and Problem Solving with Perl), ISBN 3-89864-320-4, 280 pp., dpunkt.Verlag, Heidelberg, June 2005.

Publications

2. Elke Michlmayr, "Self-Organization for Search in Peer-to-Peer Networks", Book chapter, Advances in biologically inspired information systems: models, methods, and tools (Springer Studies in Computational Intelligence), accepted for publication.
3. Elke Michlmayr, Arno Pany, Gerti Kappel: "Using Taxonomies for Content-based Routing with Ants,", Journal of Computer Networks, accepted for publication.
4. Elke Michlmayr, Steve Cayzer: "Learning User Profiles from Tagging Data and Leveraging them for Personal(ized) Information Access," Proceedings of the Workshop on Tagging and Metadata for Social Information Organization, 16th International World Wide Web Conference (WWW2007), Banff, Alberta, Canada, May 2007.
5. Elke Michlmayr, Steve Cayzer, Paul Shabajee: "Add-A-Tag: Learning Adaptive User Profiles from Bookmark Collections," Poster paper, International Conference on Weblogs and Social Media (ICWSM'06), Boulder, Colorado, USA, March 2007.
6. Elke Michlmayr, "Self-Organization for Search in Peer-to-Peer Networks: The Exploitation-Exploration Dilemma," Proceedings of the 1st International Conference on Bio-inspired Models of Network, Information and Computing Systems (BIONETICS 2006), Cavalese, Italy, December 2006.
7. Marion Murzek, Gerhard Kramler, Elke Michlmayr, "Structural Patterns for Business Process Models Transformation," Proceedings of the International Workshop on Models for Enterprise Computing, 10th IEEE International Enterprise Computing Conference (EDOC 2006), Hong Kong, October 2006.
8. Elke Michlmayr, Arno Pany, Sabine Graf: "Applying Ant-based Multi-Agent Systems to Query Routing in Distributed Environments," 3rd IEEE Conference On Intelligent Systems (IEEE IS06), London, UK, September 2006.
9. Elke Michlmayr, Arno Pany, Gerti Kappel, "Using Taxonomies for Content-based Routing with Ants," 2nd Workshop on Innovations in Web Infrastructure, 15th International World-Wide Web Conference (WWW2006), Edinburgh, UK, May 2006.
10. Elke Michlmayr: "Ant Algorithms for Search in Unstructured Peer-to-Peer Networks," Ph.D. Workshop, 22nd International Conference on Data Engineering (ICDE 2006), Atlanta, Georgia, USA, April 2006.
11. Elke Michlmayr, "A Case Study on Emergent Semantics in Communities," Workshop on Semantic Network Analysis, International Semantic Web Conference (ISWC2005),

Galway, Ireland, November 2005.

12. Elke Michlmayr, Sabine Graf, Wolf Siberski, Wolfgang Nejdl, "Query Routing with Ants," Workshop on Ontologies in Peer-to-Peer Communities, European Semantic Web Conference (ESWC2005), Heraklion, Greece, May 2005.
13. Martin Bernauer, Gerti Kappel, Elke Michlmayr, "Traceable Document Flows," 2nd Workshop on Web Semantics, 14th International Conference on Database and Expert Systems Applications (DEXA), Zaragoza, Spain, September 2004.
14. Gerti Kappel, Elke Michlmayr, Birgit Pröll, Siegfried Reich, Werner Retschitzegger, "Web Engineering - Old wine in new bottles?," 4th International Conference on Web Engineering (ICWE2004), Munich, Germany, July 2004.
15. Harald Kunczier, Elke Michlmayr, Günther Pospischil, Hermann Anegg, "LoL@ - A Prototype of a Network Independent Wireless Internet Service," 5th International Symposium on Advanced Radio Technologies, Boulder, Colorado, USA, March 2003.
16. Martina Umlauft, Günther Pospischil, Georg Niklfeld, Elke Michlmayr, "LoL@, a mobile tourist guide for UMTS," Journal of Information Technology & Tourism, Congnizant, USA, 2003.
17. Günther Pospischil, Martina Umlauft, Elke Michlmayr: "Designing LoL@, a mobile tourist guide for UMTS," 4th International Symposium on Human-Computer Interaction with Mobile Devices, Pisa, Italy, September 2002.
18. Martina Umlauft, Elke Michlmayr, Hermann Anegg, Harald Kunczier, Günther Pospischil, "LoL@: Ein Prototyp für einen UMTS-basierenden mobilen Stadtführer," H. Meuer and O. Spaniol (Ed.): Praxis der Informationsverarbeitung und Kommunikation, K. G. Saur, München, Germany, September 2002.
19. Hermann Anegg, Harald Kunczier, Elke Michlmayr, Günther Pospischil, Martina Umlauft, "LoL@: Designing a Location Based UMTS Application," ÖVE-Verbandszeitschrift e&i, Springer, Heidelberg, Germany, February 2002.

Teaching Experience

- Course on Web Engineering (Summer terms 2006, 2005, 2004)
- Course on Semantic Web (Summer term 2004)

Theses supervised

- Rene Koppensteiner: "Assisting Users in Storing the Results of their Information Search on the Web," February 2006.
- Arno Pany: "A Simulation Framework for Ant Algorithms in Peer-to-Peer Networks," December 2006.