UNIVERSITY OF TWENTE.

Faculty EECMS - Computer Science
Database Group

Master of Science Thesis

Towards Distributed Information Retrieval based on Economic Models

Author:
E. Eerenberg

Graduation Committee:
Dr. Ir. D. Hiemstra
K.T.T.E. Tjin-kam-jet, MSc.
Dr. Ir. M. van Keulen

January 3, 2011

Acknowledgments

In February of 2010 I started to look for a challenging and fun final project, which I hoped to find by wandering around the building of Computer Sciences and talk to many different professors. I already had my own idea of using economic models in computer science problems that I came up with during a study tour I had organized. During this study tour, I witnessed a motivating presentation of a professor at Harvard about how he had built a successful spam-filter using an economic model instead of using traditional computer science techniques. During the final assignment of my Bachelor curriculum, I had already applied economic models on planning problems and that had been a fun and challenging assignment.

When I walked into Djoerd his office in February, there was a lucky coincidence. Djoerd turned out to be researching the use of online advertisement models (economic models) as a means to organize distributed information retrieval. I had found my assignment! Djoerd introduced me to Kien and Almer as members of the committee, and the assignment was started. Sadly, Almer left the committee after a few weeks, but I would still like to thank him for his support with the system design.

Being a politician for the city council and running again for office, the political campaign had started as well in February, intensively. So I started simultaneously with my thesis assignment and the political campaign, which was followed by coalition negotiations. Maybe I should have waited with the start of my thesis, as I was spotted phoning a lot in the staircase. Nonetheless, I want to thank Djoerd and Kien sincerely for their trust in letting me combine these two hectic happenings.

During the past 9 months, Djoerd, Kien and I had interesting, motivating and chattering meetings about the project. I would like to thank Kien for his sharp eye, showing me the large amount of tiny mistakes that I made while typing this report. Furthermore, Kien really helped me with his critical questions on the simulation setup, guiding me into the right direction. Djoerd I would like to thank for his quick mind; always rapidly grasping the essence of a problem and steering me into the right direction with only a few remarks. Especially the one time when Djoerd walked into the computer room and asked a question about what was missing in my homebrewed corpus, making me realize that the corpus was unusable and we had to skip that part of the research. Despite of this single question making a week of work useless, it also sharpened my view on the problem and I learned that it is sometimes good to rethink your approach drastically.

In addition, I want to thank Djoerd and Kien for the trust they had in me in freely exploring the subject and extending the research with a real-world test,

knowing the risks of doing so.

Maurice was asked to join the committee for the last few weeks. I would like to thank him for the support he gave me when discussing the problems I faced with the model checker. It forced me to rethink everything, and to finally come up with a useful way of applying it.

Finally, I want to think my friends, family and girlfriend. They participated in the tests (if qualified), and debated the subject with me extensively.

I had a great time, and I will certainly miss the research environment where one is free to investigate if ideas will work without being in the pressure cooker of a company.

Eelco Eerenberg

Abstract

Introduction

With the ever increasing amount of data on the Internet, there is an increasing need to search this information in new and more efficient ways. A part of the data on the Internet are not accessible to traditional search engines, as these data can only be accessed by a form for example. With distributed information retrieval systems however, these types of data can be accessed. In these systems there is a central broker with multiple servers, and the broker redirects queries to the servers. Each server fetches results from its own database and returns this to the broker.

We are interested if this architecture can be built using an economic model, in which servers need to pay for the right to return results. We have seen from previous research that the use of an economic model might yield good results, as a successful spam filter based on an economic model has already been built.

The aim of this research is to build a successful distributed information retrieval system based on an economic model, allowing servers to open up their part of the deep web.

Methodology

This research consists of three parts: 1) selecting suitable economic models, 2) simulating these models, and 3) performing a real-world test.

We selected the economic models starting with a review of the current literature on economic models. With the obtained information we performed a multi-criteria analysis, a model checking phase, and a test on economic properties to select suitable models.

The remaining models were simulated in custom-built simulation software, in which multiple variables were modified in different runs in order to observe their effects.

The most suitable economic model was implemented in a real-world test, in which users valued the results of the system based on an economic model as well as a traditional search engine.

Results

We found the models of Vickrey auction and bond redistribution to be the most suitable ones. These models behaved well in our simulation and both outperformed a naive comparison model. The Vickrey auction model performed best in a scenario that mostly resembles the Internet. On average 69% of all models with a strong correlation between the economic outcomes and the performance of information retrieval (Kendall's-$\tau > 0.6$) is a Vickrey auction model. In the real-world test we show that users appreci-

ate both the use and administration of an information retrieval system based on an economic model. Furthermore, if we apply a perfect categorization, the economic model outperforms the comparison engine with a 66% increase in performance.

Discussion

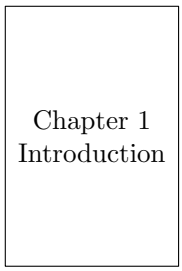
We conclude that it is possible to build a distributed information retrieval system based on an economic model. It performs better than a naive system and also in a real-world test it outperforms a traditional engine. However, non-human categorization of the queries negatively influenced the performance of the models, which shows the need for better categorization algorithm. Exposing the deep web with the use of an economic model is feasible and might even introduce new business models for servers and brokers by earning money with search results.

Contents

Chapter 1

Introduction

As the Internet is expanding rapidly, many new sources of information erupt every day. In order for users to find the information that they specifically need, users need ways of searching the Internet. This searching is getting harder, with a wide variety of types of sources. Hence, there is a need for new, more efficient ways of searching the Internet. In this chapter we will introduce our research problem and research questions.



1.1 Traditional Search Engines

Traditional web search engines execute three phases to enable Internet search: 1) crawling, 2) indexing and 3) searching.

The process of *crawling* is in essence an automated manner of browsing the web. This process starts with a set of Internet pages, called a *seed*. For every page in the seed, all links are identified and added to a list of pages to visit. This list is then visited by the crawler, where the process is repeated for every page it visits. The crawler creates a copy of every page it visits for the next phase; indexing.

It is impossible to run every search query over every page that has been copied by the crawler, as this takes too long. This is the problem which *indexing* solves. Indexing parses the pages and extracts useful terms. For example, all occurrences of terms might be counted in a page and stored in a so-called *inverted index* together with an identifier for the document. When a term is searched, the document with the highest occurrence of the term can easily be found and the document retrieved by following the identifier.

The processes of crawling and indexing are both needed and designed for the actual *searching*. The user enters a *query*, which describes the information the user is searching for. These queries are often unstructured and contain ambiguous terms (e.g., apple could refer to fruit or to a computer vendor). The query is therefore processed by the search engine, using a combination of technologies like language processing or query expansion [24].

The processed query is finally run on the index, where matching terms are related back to the original document and the Uniform Resource Identifier (URI) that points to the document.

1.2 The Deep Web

The process described previously s suitable and well-developed for finding parts of the Internet that are known as the visible web. The *visible web* are those parts of the Internet that are published and accessible by following links.

Apart from the visible web, there is a large collection of data on the Internet that is part of the *invisible web* or *deep web* [6]. This type of data is not able to be crawled for a variety of reasons; the data might not be accessible using an URI, there might be no links pointing to the resource, or the owner of data might have excluded the resource explicitly.

The biggest part of the deep web resides in all sorts of databases and information systems. In most cases the data is actually presented to the user, but only after some sort of operation in an information system. E.g., a train schedule database is not directly visible on the web, but users can see the information it contains by scheduling a trip on the train company's website. As crawlers are not able to crawl these databases, this type of data is not directly indexed and searchable by the search engine.

1.2.1 Searching the Deep Web

In order to make the deep web accessible for search engines there are two approaches: 1) top-down or 2) bottom-up.

The top-down approach is closest to the traditional search engine process as described previously. The crawler is extended with a technique referred to as *sampling* [9]. Whenever the crawler finds a database that is accessible by a form, the crawler will fill out the form with randomized patterns and the result pages are processed (i.e., establishing which data are retrieved from the web form and how) and indexed.

The bottom-up approach requires a completely different architecture. Databases should proactively make themselves known to the search engine, together with details on how to query the database. The search engine can then relay queries to each database and merge the results. This distributed architecture introduces new problems, as we will discuss in the next section.

1.3 Distributed Information Retrieval

The field of Distributed Information Retrieval (DIR) [8] studies the distributed bottom-up architecture as we described above and the problems and challenges it brings. In the remainder of this report we will refer to the central search engine in the distributed architecture (i.e., where the user types in queries) as the *broker*, whereas a *servers* is defined as an entity that hosts a collection of searchable data. A server is connected to a broker, allowing the broker to send queries to the server and process the results. This is the general naming convention used in many DIR research papers and books [2, 33].

Both servers and brokers operate in their own *domain*. Following the definition of Wieringa [45] a domain is a part of the world with his own real-world entities, events, and messages between these entities. For example, the domain of an personnel information system consists of employees and events like hiring or firing them. These events are subject to norms like labor laws and company policies. In order for servers and brokers to communicate about their domains, they must have a shared *domain knowledge*. For example, the categorization of queries is domain dependent. In order to communicate with each other about categorization the servers and brokers should share the same categorization.

The challenges that come with the distributed approach are well defined in the literature [8, 38]; 1) resource selection, 2) resource description, and 3) results merging.

The problem that is referred to as *resource selection* is one that intuitively follows from the distributed approach. When a large number of servers are known to a broker, it is inefficient to relay the query to each and every one of these servers as only a few of them will contain relevant information to the query. Furthermore, it will cost a lot of bandwidth and computational power. For the end-user the results will be a long wait, as every server has a different response time. Showing the results to the user as they come will yield in usability problems for the user. In short, the problem of resource selection is to know to which subset of the servers to send the query beforehand.

Related to the problem of resource selection is the problem of *resource description*: how to formally describe the data that a server hosts. One could select the servers based on this description if one knows which kind of data is searched for with a given query.

Once the broker has received the results of every server to which the query was sent, the problem of *results merging* arises. Some of the servers will return

an ordered list of results, with the top one being the most important result. Other servers will provide such a list, together with a relevance score for each result, and yet other servers will just return unordered results. However, the end-user wants to see one ordered list of results. Results merging is the problem on how to make one correct list out of all the list, differing tremendously in style and content, as delivered by the servers.

1.4 Motivation

The motivation of our research is twofold. First, we are motivated by the research done by Loder et al. on economic models solving the problem of unsolicited email (spam) [29].

The research on economic models to solve spam focuses on introducing an economic model to email. This means that both sender and receiver will have a (marginal) cost to send or read an email. In this model, there are for example emails with a high value to the sender but low value to the receiver. These emails will not be read by the receiver.

In the actual world, however, receivers cannot tell for sure beforehand what the value of an email will be. So there is a gray area where some sort of economic mechanism should cover this problem. Loder et al. researched different solutions, among others introducing a bond which is returned to the sender if the receiver values the email. The outcomes of this study show that introducing a bond for email messages eliminates the spam problem, without costly filtering techniques. Currently, there is a patent being reviewed on how to incorporate this mechanism in email protocols [30]. A detailed explanation of this research is presented in Section 2.2.

In general, their research learned us that incorporating non-computer science solutions into computer science solutions might yield surprising results.

Second, our research group is currently performing research on keyword auctions for distributed information retrieval [19]. These keyword auctioning is motivated by the success of search engine advertisements, where search engines are able to place relevant advertisements next to search results based on an economic model (i.e., an auction). There is an economic driver for search engines to place relevant advertisements for the user, as they earn money if a user clicks on an advertisement and users will do so more often if the advertisement is relevant to them. If it is possible to show relevant advertisements based on this auctioning system, the same method could be of interest for gathering relevant search results in a distributed scenario.

Both the research by Loder et al. and the research on keyword auctioning motivated us to further investigate the possibilities of economic models in DIR.

1.5 Problem Description

As we described in the section on Distributed Information Retrieval, there are two main problems with a distributed architecture: 1) it is hard to determine the servers that should participate in a given query (*resource selection*) and 2) merging the results of participating servers is a challenge due to different or absent rankings (*results merging*).

Furthermore, the use of economic models is a central part of our research and hence part of the problem description. We have been inspired by Loder et al. and the success of search engine advertisement to investigate economic models, but there is no knowledge on how to apply economic models on distributed information retrieval systems.

We will not cover the problem of *resource description*. In fact, we will test if an economic model is suitable for selecting the servers and hence there is no further formal description of the content of a server needed.

Our problem for this research can be summarized as the lack of knowledge and experience with economic models solving the problem of resource selection and results merging.

1.6 Research Questions

The problem description that we stated above allows many viewpoints and possible solutions. However we will solely focus on three research questions that cover the two aspects (i.e., resource selection & results merging, and the use of economic models) of the research problem.

R1 Which economic models are able to contribute to the solution of the two problems with distributed information retrieval?

R2 Which type of economic model(s) yields the best results with regard to the first research question?

R3 Is a distributed information retrieval system based on an economic model feasible to use in a real world scenario?

The first two questions (R1, R2) are directly related to the problems that we described and cover the use of economic models for the problems of resource selection and results merging. Question R3 is related to the lack of (practical) knowledge with economic models for distributed information retrieval. By researching economic model feasibility in a real world scenario we will introduce humans and their behavior in our tests, which allows us to draw lessons about both the practical implications of such a system and the effect of human behavior on the system.

1.7 Hypotheses

Given the research questions that we described and explained above, we set our hypotheses on what we expect to find and achieve within our research. In the remainder of this report, we will connect our findings to these hypotheses.

With regard to research question R1, we introduce the following hypotheses covering the problems with distributed information retrieval.

H1 Well-performing servers in terms of information retrieval (i.e. servers with high precision) are rewarded by the economic model and end up high in the merged result list. Thereby, making an economic model suitable to select the best performing servers for information retrieval purposes.

H2 Merging the results from participating servers based on the economic value of their results enables efficient results merging.

H3 Selecting the servers based on economic values enables efficient resource selection.

The hypotheses that follow from the research question about the exact economic model with the best results (R2) are described below.

H4 Auction models are most suitable for distributed information retrieval contexts if there is shared knowledge about the domain between all servers and the broker.

H5 Bond models are most suitable for distributed information retrieval contexts if knowledge about the domain is not shared between servers and the broker.

Finally, we want to test if distributed information retrieval based on an economic model is feasible in practical situations(R3), hence the last set of hypotheses.

H6 Search engine users will favor a system built on economic models, compared to a centralized engine.

H7 Administrators will rate a distributed information retrieval system based on economic models as easy to implement and maintain.

1.8 Organization and Methodology

Our research is organized as shown in Figure 1.1. We start with an exploration of the current literature on distributed information retrieval, economic models and the application of economic models on computer science problems. Our research splits in two branches from this point, where one branch focuses on the development selection of suitable economic models, shown in Figure 1.1 as the upper branch. The other branch focuses on the design of an actual distributed information retrieval system, shown as the lower branch in the figure. The two branches combine at the point where we will simulate a distributed information retrieval system based on economic models. Finally, we will test the system in a real world scenario and draw conclusions.

During the steps of *economic model selection* we will start with any economic model that will qualify according to criteria that we will set, in the *economic model simulation* and *distributed IR simulation* steps we will drop models that do not score good enough on criteria which we will cover in the remainder of the chapters.

For the remainder of this report we will follow the organization that closely resembles our research organization. In Figure 1.2 we show the contents of this report and the relation to the steps in our research as we described above. In this figure the gray boxes are the research steps that we performed and the white boxes are the chapters were we will present the process and results of each step.

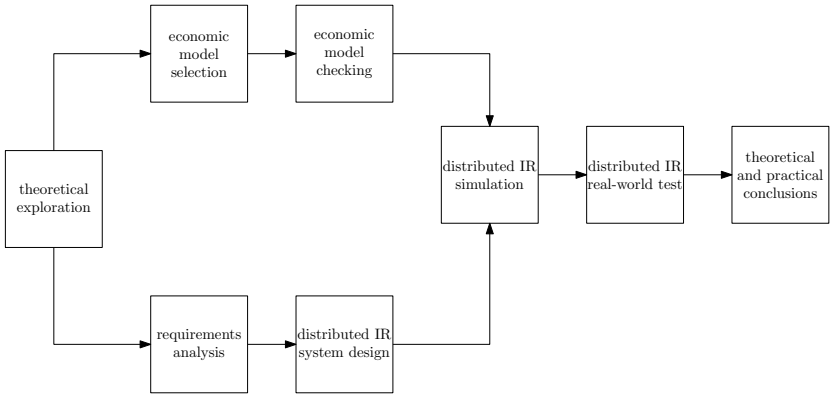


Figure 1.1: research organization

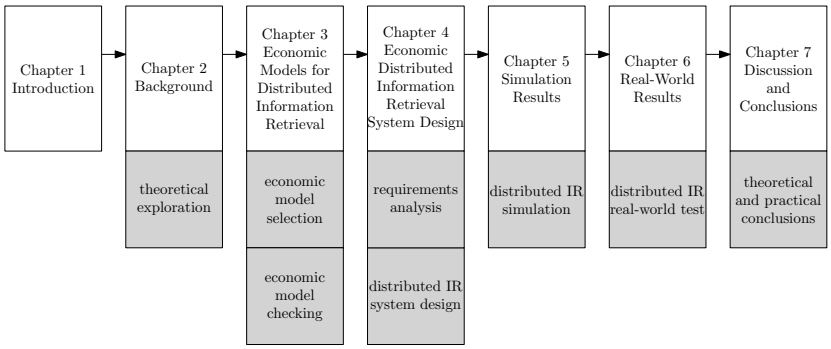


Figure 1.2: report organization

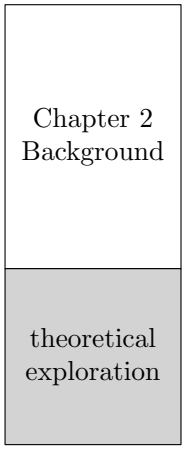
1.9 Chapter Summary

In this chapter we introduced the problem of searching the deep web; where information is not accessible by tradition search engines. This type of information can be searched however, if servers proactively open this information to a system. We propose such a system based on economic models, where servers need to pay for the right to return a result set. We believe that such a system is effective in selecting the right servers for a query and merging the results that servers send back. Furthermore we believe that the system is suitable to work in a real-world test. We want to research this claims in the rest of this report, as well as finding out which economic models are suitable for use in a distributed information retrieval system.

Chapter 2

Background

In the remainder of this report we will use the terminology and knowledge from economic modeling (especially the work on fighting spam with economic models by Loder et al. [29]), and information retrieval. The goal of this chapter is therefore to give an overview of the relevant scientific fields, their state of art, and to provide background information to this report. In this chapter the outcomes of the research step *theoretical exploration* are presented.



2.1 Economic Models

The range of economic models and theories is very broad. We can divide these economic theories into two large groups: *macroeconomic theories* and *microeconomic theories* [14].

Macroeconomics examine the world economy as one big system and study or model outcomes of such a system such as gross national income or inflation. This part of economic theory and models is of no interest for our research, as our system is not considered related to the world economy and properties like inflation are of no usage for our research.

Microeconomics however study the individual parts of the big system, especially those parts where goods or services are sold and bought. These models are more bottom-up oriented, in which global behavior is mainly defined by the individual parts. This class of economic models is what we will be considering for our research. We will cover two aspects on microeconomics in more detail: 1) agent-based computational economics; and 2) the Pareto efficiency.

2.1.1 Agent-Based Computational Economics

The type of economic models that we will use as a foundation for our research is in scientific literature denoted as *agent-based computational economics* [44]. Basically, these economic models can be seen as bottom-up models. Small micro systems define the behavior of the final macro system.

Economies are decentralized with a number of economic agents (e.g. humans, companies, artificial agents) which are involved in many local distributed interactions. From this distributed state, many global behaviors erupt such as trading protocols and socially acceptable prices. These global behaviors influence the local transactions, which lead to new or updated global behaviors. These bottom-up arisen feedback loops are the key distinguishable concepts of models from the field of agent-based computational economics. Traditional quantitative economic models do not model these feedback loops, as agents are believed to behave according to top-down rules.

Researchers who use agent-based computational economics focus on modeling the micro level, and study the behavior of the macro system over time. This matches our approach to the previously covered problems with distributed information retrieval: we will model individual servers and study the overall information retrieval system that erupts.

Real or Simulated Economy

Economic models for agent-based economies can either be simulated or real. In the case of simulated models, there are no actual monetary transactions between agents in the model. However, the agent will behave as if there are real monetary transaction (i.e., the decisions that an agent will make are not dependent on whether or not a real transaction occurs). In the case of real models, there are actual monetary transactions and hence connections to the real economy.

For our research we will not consider connections to the real economy, but assume that every party in the microeconomic system behaves as if the economy is real. This is a common assumption in agent-based computation economics

research [44]. Therefore we will use the generic term *credits* as the currency for a server or broker in our models.

2.1.2 Pareto Efficiency

Vilfredo Pareto was an Italian economist who performed many studies on income distributions in the 19^{th} century[18][1]. His basic idea is that there is more to an economy than producers and consumers and finding the optimal production strategy to satisfy the highest number of consumers. Pareto stated that a economy can be improved if one of the parties can be better off, without making one of the others worse. Hence if every party is better off under one policy compared to another, the former is preferable due to a higher Pareto efficiency.

Intuitively, fining parties within an economy is not Pareto efficient because paying a fine lowers the wealth of one party. However, in the definition of Pareto this is not necessarily the case. A classic example is if a monopolist is fined for being a monopolist and forced to behave less like one. The monopolist will however be compensated, as the economy will be more flexible due to this fine. This fine is therefore Pareto efficient, as everybody in the system gains [20].

Generally, a microeconomic can be Pareto efficient if the losers from a policy are compensated by the winners of the same policy. This has been formalized by Pareto, but we will not cover the formal definitions. In layman terms, Pareto introduced an economically sound definition of an honest situation between all participants in an economic system.

Pareto is a widely used validation measure for microeconomic models, as it has been shown that a good microeconomic model should be Pareto efficient [32]. We will hence use Pareto efficiency in our research to validate possible economic models.

2.2 Modeling Email Value and Spam

As we stated in the introduction of this report, we have been inspired by the work of Loder et al. They introduced an economic model as a means to solve the spam problem. In this section we will further explain their approach and results.

2.2.1 Motivation

The problem of spam is not limited to nuisances for email users, but has a worldwide impact on all sorts of organizations.

Adding up the computational power and wasted labor time, the yearly economic damage done by spam in the United States is in the range of $42 billion [22] and $87 billion [10]. The same studies show that more than 60% of the email nowadays is spam, hence a large percentage of network traffic and resource usage.

In a report from 2009 on spam and environmental impact [40], researchers from the McAfee cooperation calculated that the worldwide energy usage of

[1]Pareto also introduced the 80-20 rule when he noticed that 20% of the Italian population contained 80% of the total wealth. This rule has been applied to many fields afterwards, including the Zipf distribution which is used in information retrieval

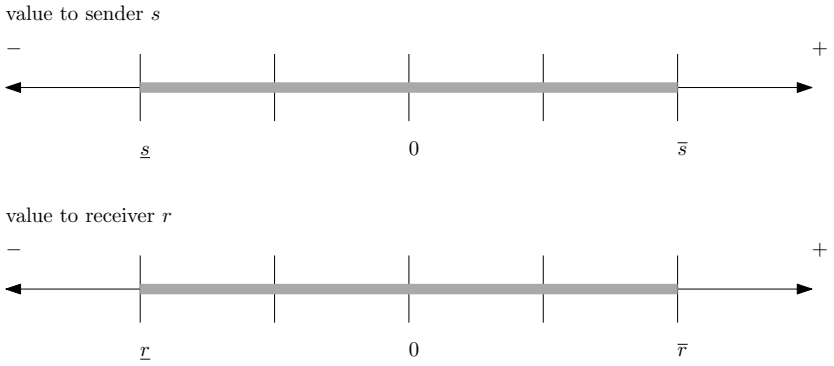


Figure 2.1: message value modeled

spam solutions totals to 33 billion kilowatt-hours (kWh): the equivalence of 2.4 million households.

Current spam solutions can be divided in two groups: legislative and technological solutions, both are not successful in fighting the spam problem. The legislative solutions are dependent on definitions (i.e., which messages are considered to be spam and which are not) that are very hard to agree upon and partly contradict free speech. Furthermore, the costs of controlling with police and to adjudicate offenders are high and do not guarantee that spam will stop spreading around.

Technological solutions mainly focus on filtering. As with any filtering technique this yields false positives (i.e., non-spam classified as spam) and false negatives (i.e., SPAM is not classified as spam), both unwanted effects for a spam solution. Filtering does not decrease spam traffic on the network, as filtering is primarily done at the receiver.

Loder et al. [29] are therefore motivated to solve the spam problem, improving the existing technological solutions.

2.2.2 Generic Economic Model

The actual solution that Loder et al. introduce is to model an economy for sending and receiving email messages. Opposed to legislative and technical solutions, their economic model solves the spam problem, by having an economic model. In this subsection we will explain the actual model that inspired us, in order to make our own motivation clear.

The main driver of the model is that every *email* has a party-dependent *value*. There are two parties in the model; a *sender* and a *receiver*. Hence every email has a value s to the sender and a value r to the receiver. The value of r and s is limited on two sides (i.e. a negative limit and a positive limit). The limits are denoted as $\underline{r}$, $\overline{r}$,$\underline{s}$, and $\overline{s}$. In terms of the *entity-attribute-value* model [11], an email message is the entity and the economic value an attribute of this entity which has value r or s depending on the party. In Figure 2.1 we show the modeling of message value.

Besides the values of the messages there are actual costs introduced in the model. Costs c_s for sending a message and costs c_r for receiving a message. The modeled costs are not meant to be interpreted as people paying for reading email,

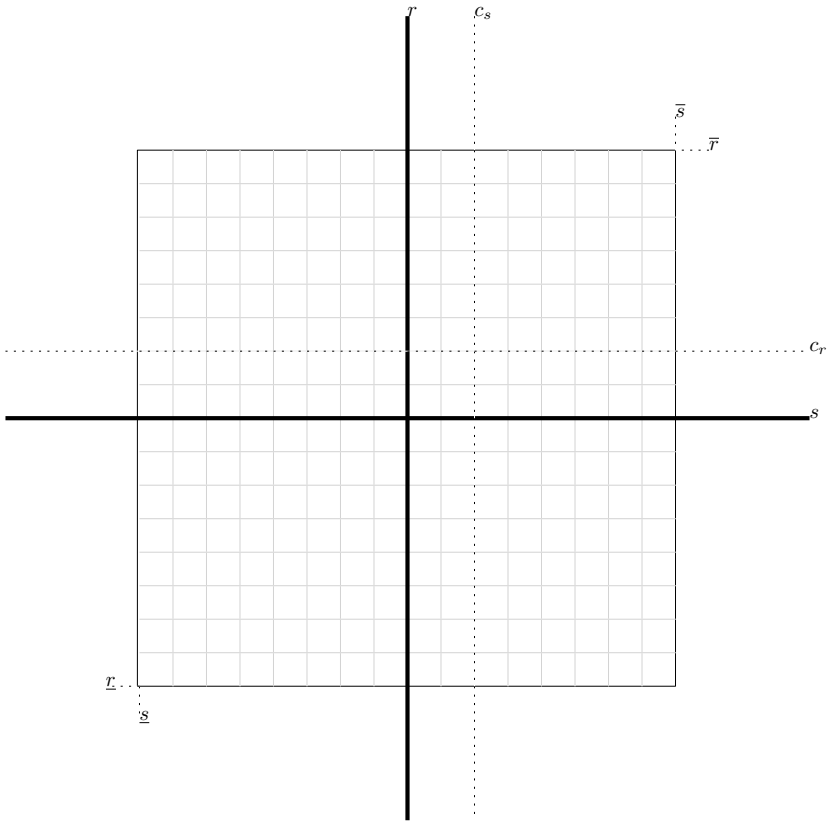


Figure 2.2: economic model

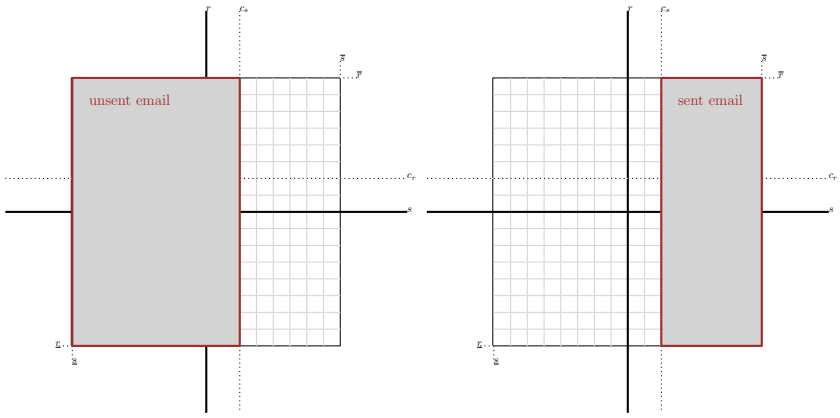
but are an economic measure of for example lost time by reading the email. As with the values, the costs are limited and can be both positive and negative. For example, a message with negative costs for receiving c_r will theoretically allow the receiver to earn money by receiving the message.

In the model, the sender knows the value s of the message he wants to send as well as the actual costs c_s of sending it. A sender will not send a message if $s \leq c_s$. Two additional rules are set in the model. First, the sender does not know the value r of the message to the receiver. Second, a receiver knows his value r *only after* reading the message and incurring cost c_r.

When all concepts that we depicted above are combined, we can draw an overview of the model as seen in Figure 2.2. On the two axes we represent the values, s and r, of a message. The dotted lines are the costs of sending and receiving, which are set at a small positive value for our explanation.

In the figure there are two categories of messages, those that will be sent ($s > c_s$) and those who are not going to be sent ($s \leq c_s$) as depicted in Figure 2.3. As stated before this division is logic, as messages which are more costly to send compared to the value are not sent.

Within these two categories we can further differentiate based on the receiver. Receivers want to read email that has a higher value than the actual costs of reading ($r > c_r$), and do not want to read email that is not worth the



(a) unsent email (b) sent email

Figure 2.3: economic model with unsent and sent email

investment ($r \leq c_r$). This creates four categories as we show in Figure 2.4:

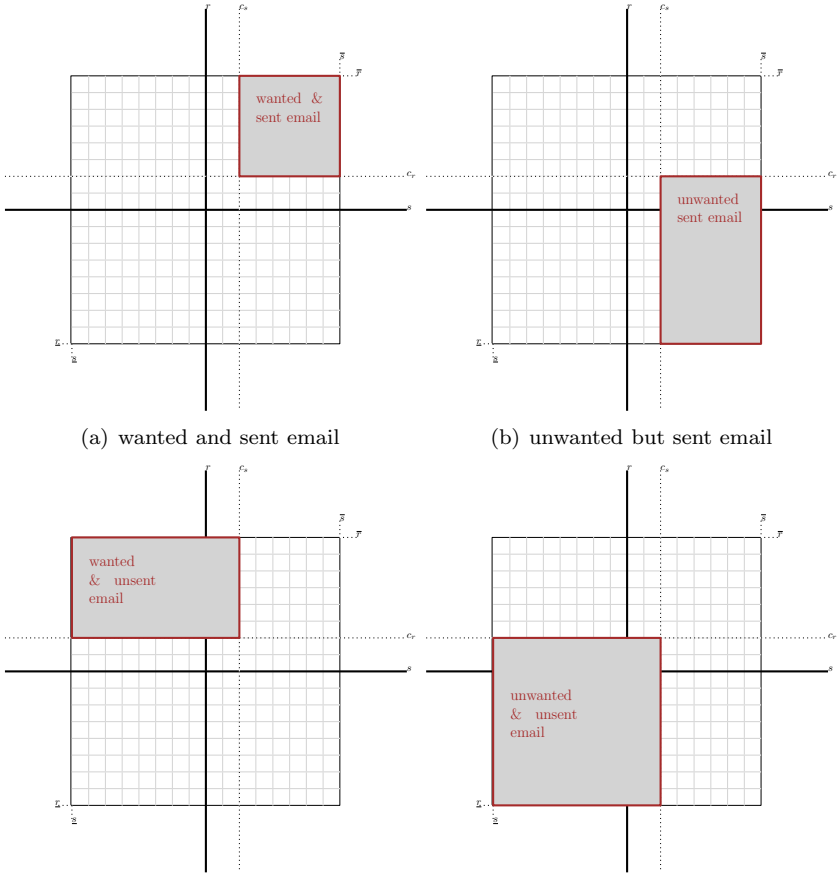
- *Wanted* and *sent* email (E.g., an invitation for a birthday party of a close friend).
- *Unwanted* but *sent* email (E.g., a spam message on counterfeited medicine).
- *Wanted* but *unsent* email (E.g., a labor-intensive personalized email with customized information from different sources).
- *Unwanted* and *unsent* email (E.g., an outdated notification on changes in some regulation).

The description and explanation that we provided in this section was on a *generic* economic model for email value. Within this model, one can model different economic models and maximize the size of the wanted emails category. Furthermore, existing solutions can be modeled within the same model [29, 28].

2.2.3 Attention Bond Mechanism

In their paper, Loder et al. model a perfect filter in the economic model. A perfect filter is defined as a technological filter which operates without any cost, makes no mistakes, knows the preferences of every receiver and eliminates all email messages that are not worth reading ($r < c_r$) prior to receipt. The perfect filter is introduced as comparison case for the model that Loder et al. introduce themselves: the Attention Bond Mechanism (ABM). Loder et al. prove that their ABM is better than the perfect (technological) filter.

A *bond* is formally described as "a contingent liability with an expiration date" [28]. In more describing terms, a bond is a sum of money which the sending party sets aside by a third party before a transaction occurs, as a sign of good faith. If the receiving party is not content with the delivered service, it requests the bond from the third party. In the case that the receiving party actually is content, the third party repays the bond back to the sending party.



(a) wanted and sent email

(b) unwanted but sent email

(c) wanted but unsent email

(d) unwanted and unsent email

Figure 2.4: economic model with four email categories

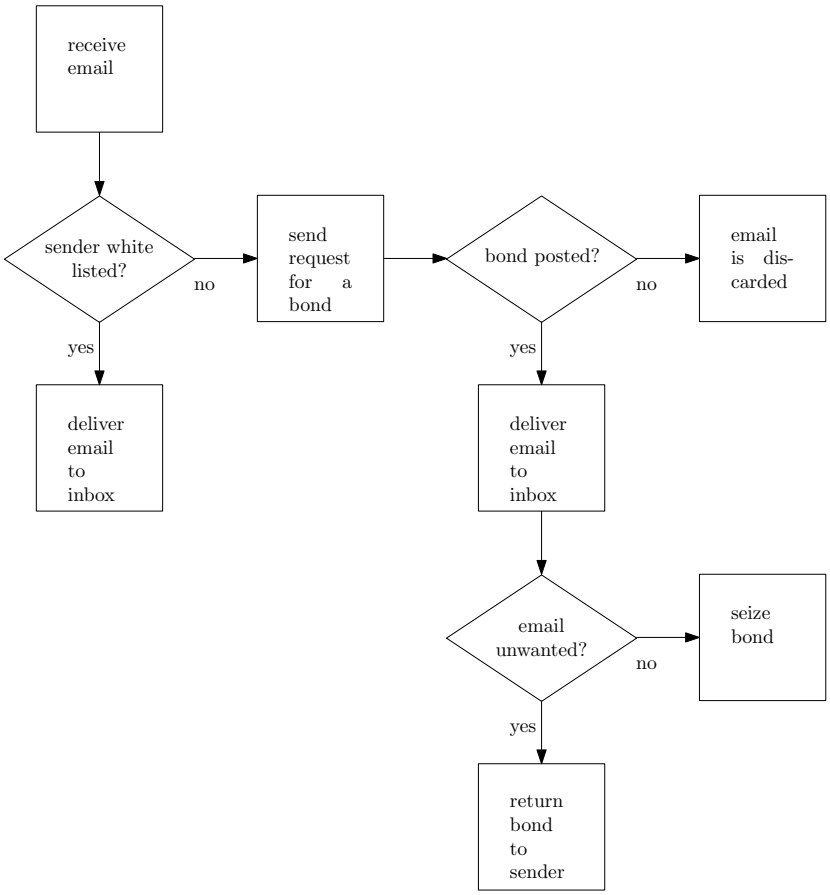


Figure 2.5: Attention Bond Mechanism flowchart

The ABM works with pre-approval using a white list. A receiver can white list certain senders (e.g., close friends). Email sent from a sender on the white list will be received by the receiver, without further interference. If the sender is not on a receivers white list, the sender is required to post a small bond in order for the email to be delivered. Economically, the sender therefore guarantees the content of the email to be useful in his opinion. The client reads the email and might either decide to seize the bond if the email was unwanted or time consuming (e.g., interesting direct marketing) or decide to not seize the bond if the email was a welcome one. The size of the bond is set by the receiver, when an email is not white listed the size of the bond is posted back. Obviously, people who want to be sure that they are not bothered with unwanted email set a high bond size, whilst people who are interested in new information might decide to set it relatively low. In Figure 2.5 we summarized this description of the ABM.

When the ABM is modeled in the economic model that we described, Loder et al. model the bond ϕ and the probability of seizure as p. Hence the sender will not send emails for which $s > (c_s + p\phi)$ is false. This is different compared to the perfect filter, as some receivers will positively value certain types of junk

mail which was filtered by the perfect filter. Therefore the surplus for the sender is shown to be higher compared to the perfect filter, as more messages will be read. Furthermore, the surplus for the receiver is also positive, because of the possibility to make money with (not) reading email, which was not possible with the perfect filter.

The whole system of senders and receivers is performing better with an ABM compared to the perfect filter, as the *social welfare* is proven to increase [29]. Social welfare is the sum of every income of each party in a system, in the email case the surplus of the sender and the receiver. The social welfare for the ABM is also proven to be higher compared to the perfect filter, hence every party in the system benefits most from this model.

Social Benefits

The findings of Loder et al. show that within the economic model which they defined, their Attention Bond Mechanism is the best solution to fight spam. However, this is an economic and theoretical conclusion. Therefore, Loder et al. also conclude on what they define as *social benefits*: practical benefits of the ABM for users.

First, the amount of spam will drop significantly because there will be no senders willing to warrant their message (i.e., post a bond) of which they know to be spam. This decreases the nuisance of receiving spam for users.

Second, the ABM will create a new market. There will be people who are interested in direct marketing. They will seize the bond, but Loder et al. show that the total costs of sending marketing with their solution are lower when compared to traditional direct marketing techniques (e.g., sending paper advertisements to unknown addresses).

Third, the ABM uses *ex post* verification where filters use *ex ante* verification to value an email. Ex post verification takes place after the message has been delivered, whilst ex ante verification takes place before the message is delivered. It is impossible to fool an ex post verification, as it is the receiver himself who decides on the quality of an email. Filters can be fooled (ex ante), by clever email writing (e.g. putting spaces in keywords which normally trigger the filter). The problem of false negatives and false positives is therefore solved by the ABM, as these problems are non-existing with ex post verification.

2.2.4 Summarizing Conclusions

The final conclusion by Loder et al. is that for a variety of reasons their proposed Attention Bond Mechanism dominates other systems. The introduction of a bond improves the welfare of both senders and receivers, as senders are forced to act on their private knowledge of the email value. Email of low value to the receiver, the ABM compensates with a small amount of money which flows from sender to receiver. On the other hand, email which is of high value to the receiver, will not bother the receiver with obligatory monetary transactions.

As an added benefit, direct marketing can take place for those consumers who actually want to read the marketing messages whilst being compensated for their time.

2.3 Query Categorization

One of the problems in distributed information retrieval is to know which types of queries need to be sent to which peers, known as resource selection. Solving this issue centrally requires knowledge about both the query and the information that servers are able to deliver (i.e., resource description). In our approach, the broker will not keep track of the knowledge within each server, as the economic models should force servers to solve the resource selection issue by themselves.

The category of the query is still needed for some of our models, see Chapter 3. The broker will categorize the query and will send his category together with the query to each server. This central approach ensures a leveled playing field, as there will be no errors due to different classifications. Misclassification might still occur, but at least every server has the same effect of the error introduced by the classification of the broker.

In the field of information retrieval the process of reasoning about queries and categorizing them is referred to as query categorization [2].

Query categorization uses the text of the actual query and a knowledge source to determine the category of the query. In general there are two steps within a query categorization algorithm.

First, the text of the query is analyzed and processed. Common words such as "the" might be deleted, nouns and names might be extracted, or synonyms for nouns might be added (i.e., a process called query expansion). The type of processing and analysis is algorithm dependent.

Second, the processed query is run through a knowledge source. Again, there are different strategies to this step. Some algorithm use formal taxonomies (e.g., WordNet, a lexical database of the English language [54]) to map queries to categories, others might use a search engine to fetch results for the query and analyze them (e.g., perform a word count on the results and expand the query with common found words).

2.4 OpenSearch

We will be using search engines in our research extensively as both the servers and the broker are search engines. Each of the servers in our setup will have to answer to the broker in the same format, to prevent mapping issues at the broker. Hence, we will use a standardized form of communication between broker and servers.

We will use the latest OpenSearch standard for all communication to and from search engines. The OpenSearch standard is an Extensible Markup Language (XML) standard for search engines en clients [49]. There are two main architectural components to OpenSearch.

1 The *OpenSearch Description Document*. This is an XML file on the search engine's host which describes the search engine and mainly how to query this engine.

2 The *OpenSearch Response Elements*. Search engines should return results in existing formats like RSS or Atom. These existing formats should be extended with the response elements from OpenSearch. For example, a

search engine adds to its Atom-list with results an OpenSearch-element that contains the original query.

The connection between these two components is the URI where a client can post a query. This URI is defined in the OpenSearch Description Document, as well as how to properly use it. After submitting a query to this URI, results are returned back in a format that is extended with the response elements.

OpenSearch is an open format that allows for extensions, using the default XML extension methodology with name spaces. Multiple extensions already exist, such as possibilities to send search suggestions to the client while the client is typing the query.

As OpenSearch allows for easy extension of standardized messages between search engines, we will use the OpenSearch standard for the communication between the servers and brokers that we will build and cover in the rest of this report. We will extend the messages with specific elements that will contain information about the economic model.

2.5 Information Retrieval Measures

Throughout this report we will use the *precision* of search results, as a measure of how well our solution performs. Precision is, together with *recall*, a widely used measure in the field of information retrieval [8]. In this section we will explain the two measures and why we do not use recall as a measure.

2.5.1 Precision

For a given query an information retrieval system will return a number of documents d from its corpus. From these d documents, only r documents are relevant to the search query . Precision is denoted as $P = \frac{r}{d}$ and measures the fraction of the retrieved documents that is relevant to all the retrieved documents. Precision can be measured for the complete set of returned results, or only for the top-n documents. In the latter case the precision for these n retrieved documents is denoted as $P@n$.

We will use precision as it measures how well our information retrieval system is in returning a set of relevant results to the user.

2.5.2 Recall

As with the precision definition, an information retrieval system returns d documents of which r are relevant to the query. However, in the corpus is a total number of $R \geq r$ documents being relevant to the query. Recall measures how good a system is in retrieving these relevant documents from the complete corpus and is calculated as $\frac{r}{R}$.

We will not use recall to measure how well our system behaves, as we are not interested in finding all relevant documents but only interested whether or not the documents found are relevant.

2.6 Related Work

In this section we will present related work in three areas. The areas that cover the two initial problems of our research: 1) resource selection and 2) results merging. The third area that we will cover is about the application of (micro)economic models to computer science problems.

2.6.1 Resource Selection

Research has been conducted on the problem of selecting the right servers for a query. Most of these solutions assume there is a description of the contents of each server. The most common solution is the CORI algorithm, where the *tf-idf* measure is used. Tf-idf stands for *term frequency-inverse document frequency* and is a statistical measure calculating how important a term is within a document from a corpus (i.e., the set of all documents that a server hosts). The *term frequency* calculates the number of times a term exists in a given document, whereas the *inverse document frequency* calculates the importance of the same term in the complete corpus. A document is deemed relevant if the *term frequency* times the *inverse document frequency* is high, which is true if a document has many occurrences of the term while being among few other documents in the corpus containing the same term [17].

In a distributed scenario where CORI is used, the *term frequency* for every document is calculated at every server, whereas for the *inverse document frequency* the total number of documents from all servers is used. This distributed solution is shown to have a 100% recall when 60% of servers has been searched, and an average precision of 0.4 in different distributed experiments [8].

Another proposed solution is based on language theory, where a description of each server's content resource description is harvested by the broker. The broker queries the server with generated queries and analyzes the results, which is referred to as *query-based sampling*. The results are analyzed and a language model is built. A language model is basically a set of probabilities for sequences of words from a document. For example the sequence "car is stolen" might have a probability of 0.7 to occur in document, whereas the sequence "car stolen is" has a probability of 0.2 Whenever servers need to be selected for a given query, the probability on the sequence of words from the query is determined for each language model (i.e. for each server). The servers with the highest probabilities are then selected to receive and answer the query [47, 36].

2.6.2 Results Merging

Many results merging solutions are related to the scores that each server sends together with each result. Merging the results from all results is then somehow based on these scores and referred to as Raw Score Merging [38]. Within these algorithms, there are multiple variations. One could weight each score from each server with a value that is related to the corpus of each server. For example, some systems multiply each result score with the *inverse document frequency* value of the server [38].

Another solution that has been proposed is to let the broker download a preset number of the top-documents of the result list from every server, and let the broker create an index for these documents. Using this index the traditional

information retrieval measures can be used to rank the results, such as counting search terms in every document and ranking the documents based on this counting [31].

2.6.3 Economic Models

We already extensively covered the work of Loder et al. with regard to spam problems [29] in Section 2.2. Their conclusion is that the use of an Attention Bond Mechanism (i.e., an economic model) is more effective in solving the spam problem when compared to existing techniques.

Buyya et al. [7] show that resource management in grid computing can be efficiently performed with economic models. They examined, amongst others, posted price and auction models to allocate resources for those who are demanding them. Their simulation only covered a model which there is a fixed price for a period of resource use. Their results show that it is possible to use economic models for a robust system.

From our literature review on the application of economic models in computer science, we conclude that the majority of research in this topic is applied on social sciences. A group of social scientists are interested in how people behave when resources have to be shared. This can be tested with real-world field experiments involving real people, but also with simulated economic agents. In the latter case, the economic models are programmable but the framework is designed by the researchers. [21, 43, 46]

2.7 Chapter Summary

We explained in this chapter what agent-based computational economies are and that we are conducting research within that field, where economies are made out of agents which make their own decisions. We also introduced the Pareto efficiency, an economic description of honesty that should be fulfilled by good microeconomic models. We explained the model of Loder et al. [29] in detail, who proved how to build an economic spam filter that works better than traditional spam filters. We also covered other types of related work, in which we described how the problems of resource selection and result merging are solved by different approaches. The most well-known solution that solves both problems is the CORI-algorithm.

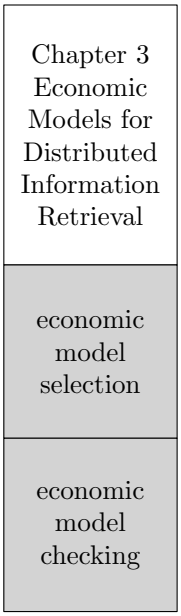
Chapter 3

Economic Models for Distributed Information Retrieval

In this chapter we present the results and general process of two of our research steps: *economic model selection* and *economic model simulation*.

The idea behind all economic models in this chapter is that the transaction of information from server to broker is of *value* to the server (e.g. generating traffic for the provider), the broker (e.g., better search experience for the user), and the user (e.g. good results are of high value). When we consider the transaction between a server and broker, it is hard to estimate the value of the transaction beforehand. Every server will value a transaction differently, based on the expected *revenue* of the transaction to the server.

In our research we will solely focus on modeling the servers (i.e., as a sender of information) and the broker (i.e., as a receiver of information). The user does not participate in the economic process, but is represented by the broker who wants to achieve the best result for the user.



3.1 Generic Model

We will use the email economic model [29] as a template for our own Distributed Information Retrieval (DIR) model. In our model there is always one broker and there can be any number of servers. The unit of trade that we will use is a *result set*, where in Loder et al. this unit is an email message. A result set is a fixed number of results for a given query. Users will type in a query at the broker, and participating servers might decide to submit a result set for the query.

We assume that servers want to attract visitors to their websites, either for commercial reasons (e.g., selling products) or for social reasons (e.g., providing users with correct information). Therefore, submitting a result set to a query is of value to a server, modeled as s. There are also costs c_s for the server, like bandwidth usage. We will introduce different types of additional costs for the server in the models that are being investigated in the remainder of this report. Those model-specific costs are not part of c_s.

The broker wants to achieve the best service for its users. Therefore, the received result set of a server has a value b to the broker. Analogously to servers we will model the costs of the result sets for the broker c_b. These are real costs like computational power and data storage.

The model that we introduced is similar to the generic model of Loder et al. For example, figure 2.2 can easily be imagined with the two variables that we introduced above.

We will not model the user in this model, although one can argue that a result set has a value to the user . We will indirectly model user value though, but for the model the value to the user is represented by the value to the broker.

3.2 Economic Model Selection

There are many different economic models, each with their own established purposes and subject domain. To select the most suitable economic model(s) for our research problem, we start with a literature review on economic models. From this review we will select economic models which will be further investigated in four steps:

1 **Multi criteria analysis on economic models**. In the next section we will state criteria an economic model should fulfill in order to be of interest to our research goals. This step of our research is covered in Section 3.2.2.

2 **Formal modeling of economic models**. We will model the remaining economic models in a modeling checking tool, and check properties (e.g., will there be no deadlock situation). This step of our research is covered in Section 3.2.3.

3 **Check for Pareto efficiency**. As a good microeconomic model should be Pareto efficient, we will check if the economic models are Pareto efficient and when this is not the case how to change them to become Pareto efficient. This part of our research is covered in Section 3.2.4.

4 **Simulation of a distributed information retrieval system with economic models**. We will finally build a distributed information re-

trieval that actually runs on the remaining models. This is extensively covered in the Chapter 5.

Our goal is to end up with one or two models that are suitable for the two scenarios (i.e., a scenario where broker and server share the same domain knowledge and a scenario where there is no shared domain knowledge) that we described in Chapter 1, by decreasing the number of suitable economic models in each of the steps described above.

3.2.1 Literature review

The term "economic model" is very broad. It includes economic models that describe how the world economy behaves, models that predict stock values, models that connect educational systems to a country's welfare, and so on and so forth [3].

We want to model the situation as described in Section 3.1: servers want to provide users with relevant results and draw traffic to their websites. Therefore, the right to send results to the broker is of value to the server. In short, the model should allow for one broker and multiple servers. The object of value is the right to send a result set, servers are buyers and the broker is the seller of this right.

According to economics, we are therefore interested in *supply-demand models* or *market models*, a subcategory of microeconomic models [16]. In these models there is a market function as there are parties interested in a certain commodity which is offered by other parties. Depending on the availability of the commodity, the value is estimated by the economic model and transactions between supplier and consumer will occur. Within the supply-demand models there are multiple models that describe how the value of a commodity is determined.

The following type of models are defined within the *supply-demand models* and possibly of interest to our research. This classification is not standard in literature, but our own combination of economic models from different scientific sources [3, 26, 29, 25, 7].

1 *Bidding* or *Tendering*. Within these models, the consumer of a commodity asks suppliers to make a bid and state their conditions. The consumer chooses the best bid (lowest price with the best conditions). This model is suitable if there is plenty of the commodity available, and there is time to make elaborate decisions.

2 *Commodity Market*. Within these models more complex financial products are used to trade. Examples are bonds and future contracts. This model is suitable for trading large quantities of commodities in short amounts of time.

3 *Auctions*. Within these models a consumer sells the commodity to multiple consumers who will bid against each other. The highest bidder receives the commodity. This model is suitable if there is a low availability of the commodity and a huge demand.

4 *Bartering*. Within these models, there is a physical swap between supplier and consumer. This means that the supplier will deliver the commodity

to the consumer, if the consumer returns a different commodity to the supplier. These models are suitable if both parties have commodities that are of interest to the other party.

5 *Fixed Price.* Within these models, a central organization sets the price of a commodity. Suppliers and the consumer will trade according to this fixed price. These models are suitable when there is a need to fully control the market as a third party.

These five types of economic models will be further investigated, starting with a multi criteria analysis in the next section.

3.2.2 Multi-Criteria Analysis

In the previous section we have shown that there are five categories of economic models that might be of interest to our research. Criteria will be used to select the categories of models (or specific models from the categories) which are suitable for further analysis, and described and listed below.

- The model should allow for multiple consumers (servers) of the same commodity (the right to publish queries).
- The model should allow for easy addition of new consumers, as new server might decide to join the system.
- The model should allow for quick transactions; time-consuming transactions which ask consumers and suppliers to communicate often are of no interest as servers should answer queries directly.

We will cover each model and its relationship to the criteria in the next subsections.

Bidding

In the general model that we depicted previously, we have one supplier and many consumers. With bidding models there is the opposite situation: there is one consumer with many suppliers. The criteria analysis for this model is listed below.

- *multiple consumers.* As the bidding model requires one consumer, this criterion is not met by this model. We could model a broker as a consumer (of search results) with many suppliers (servers), and where suppliers will place bids to be the preferred supplier. As we want our general model to be valid, the unit of transaction is the right to submit results. Turning the model around would make the unit of transaction the result set and invalidate our general model.
- *easy consumer addition.* With only one consumer in the model, it is not easy to add additional consumers. This criterion is therefore not met either.

- *quick transactions.* Transfer of the commodity can take place quickly with the bidding model. Each supplier places one bid and the consumer chooses the lowest bidder. Hence there are only two messages between supplier and consumer, which is considered quick and fulfills the criterion. Models which allow for multiple bids are considered auctions and do not fall within this type of economic models.

Commodity Market

There are many different types of commodity markets, the most simple on is where suppliers sell their commodity to consumers for a price that is set by the supplier. On the other end of the spectrum are complex financial products, where the right to buy commodities at a certain point in time for a certain price are considered commodities themselves. We will not consider the complex types of commodity markets, as the complexity will make it hard to allow easy consumer addition and quick transactions.

We will consider the elemental commodity market, in which the supplier sells commodity for a price and a bond model in which bonds are used to pay on a later moment under certain conditions. Loder et al. [29] also use a model based on bonds, which is why we will continue investigating bond models. The criteria analysis for this model is listed below.

- *multiple consumers.* Commodity markets allow for multiple suppliers and multiple consumers to trade their commodities on the market. Therefore, we can model our situation with one supplier and multiple consumers using a commodity market.

- *easy consumer addition.* As we only consider an elemental commodity market and a bond model, consumers can easily be added. In the first case, consumers can join just by buying the commodity (if available), and in the second case the model involves placing a bond. The size of the bond is known beforehand, so a consumer can join by paying the bond to receive the commodity (under the bond conditions).

- *quick transactions.* Transactions are simple in the two commodity markets that we will consider. In the case of consumers buying the commodity from the supplier, there is one message: the supplier stating the price. Bond models require more messages, but the first step only involves placing the bond. The actual calculation of the bonds happens after the conditions are met or broken. For our situation, this means that bonds are still of interest, only if queries do not have to wait for the final bond calculations. In the next step of our investigation we will investigate the bond models further.

Auctions

There are many different types of auctions, but four main categories of auctions [34]:

1 *first-price sealed bid auctions.* Consumers place their bids, but they cannot see bids of other consumers. The highest bidder wins the auction and pays his bid.

2 *second-price sealed bid auctions*. Consumers place their bids, but they cannot see bids of other consumers. The highest bidder wins the auction and pays the bid of the second-placed consumer.

3 *English auctions*. Consumers place their bids, but they can see bids of other consumers. They are allowed to place multiple bids, and react on each other, thereby raising the price. The highest bidder wins the auction and pays his bid.

4 *Dutch auctions*. There is a central price which is visible to all consumers. This price drops over time and the first consumer to accept the price pays this price for the commodity.

Besides these four main categories there is also the *all-pay auction*, which is used in some Internet systems like auctioning jobs for contractors [13]. This auction is a *first-price sealed bid auction*, but every consumer pays his bid regardless if the auction has been won or not.

We will not consider *Dutch auctions*. The continuously dropping value in reversed auctions will create many messages between parties and requires all servers to be on-line once an auction has started. This will by definition lead to a slow and unfair system and fails the third criterion.

Furthermore, we will not consider *English auctions* as they allow multiple bids (bidding against each other) and will slow down the process, failing the third criterion as well.

The other three auction types (i.e. *first-price sealed bid auctions*, *second-price sealed bid auctions*, *all-pay auctions*) are of interest for our research. The criteria analysis for this model is listed below.

- *multiple consumers*. Auctions are designed to sell a commodity from a supplier among different consumers and therefore compliant with our criterion and generic economic model.
- *easy consumer addition*. Consumers can place their bids according to the known auction rules. After placing the bids (once per auction, as we ruled out auctions with multiple bids), the highest bidder is selected. It does not matter how many consumers participate in this process, allowing easy addition of new consumers.
- *quick transactions*. As with the bidding models, auctioning involves two messages which we consider to be quick. One message is needed for placing a bid and one message is needed to communicate the winner of the auction.

Bartering

In essence, bartering is about swapping commodities. The supplier delivers the commodity to the consumer, only if the consumer delivers a commodity in return. This returning commodity must be wanted by the supplier and valued equally to the original commodity of trade. The criteria analysis for this model is listed below.

- *multiple consumers*. The bartering model allows for multiple consumers; there is no constraint stating that if the supplier has a number of commodities available all of these commodities must be bartered with one and the same consumer. Hence, the criterion is met by the model.

economic model	multiple consumers	easy consumer addition	quick transactions
bidding	✗	✗	✓
commodity market	✓	✓	✓
auctions	✓	✓	✓
bartering	✓	✗	✗
fixed price	✓	✓	✓

Table 3.1: multi-criteria analysis of economic models

- *easy consumer addition.* It is hard to add consumers to a bartering model, as the supplier does not request an unlimited return of commodities of a certain type. Hence, new consumers should bring unique new commodities that are demanded. In our model with only one supplier and many consumers, it is expected that after a given amount of consumers, new consumers will not bring valuable commodities to the supplier. This criterion is therefore not met by the bartering model.
- *quick transactions.* Transactions are slow with the bartering model. There is a negotiation phase about the commodities that the supplier wants in return. In case the consumer does not have any demanded commodity in return, the transaction even halts. This criterion is therefore not met by the bartering model.

Fixed Price

The fixed price model meets all of our criteria, because of its simple form. The costs of a commodity is preset by some party in the system and every other parties agrees to pay this fixed price. The criteria analysis for this model is listed below.

- *multiple consumers.* There is no reason why there could not be more then one consumer in a model where the price of a commodity is fixed. Therefore, this criterion is met.
- *easy consumer addition.* Joining this model as a consumer is very easy; the consumer knows what the price of a commodity is and can buy it directly.
- *quick transactions.* As there are no negotiations or biddings in this model, transactions are kept to the minimum: only paying the price that is set beforehand.

Model Selection

In Table 3.1 we show the results of the multi criteria analysis that we covered in the previous sections. We will select those models that score on every criterion and drop the other models for further research. This means that we will continue researching commodity market models, auction models and fixed price models.

As stated previously, these categories of economic models are broad and not every model from the selected categories is suitable for our research. Therefore, we select from the commodity market models the *simple market* and *bond* models. From the auction models, we select the *first-price sealed bid* auction, the *second-price sealed bid* auction and the *all pay* auction. The *fixed price* model is a model on itself. The individual reasons for selecting these models are covered in the individual sections on each model category.

3.2.3 Model Checking

From the criteria analysis step we ended with six models: 1) simple market, 2) bond, 3) first-price sealed bid, 4) second-price sealed bid, 5) all-pay auction, and 6) fixed price. These models are worthwhile to investigate, but before we will actually build a system and incorporate them, we will model them in a model checker. There are two reasons to model check our economic models: 1) it allows us to check if a model respects certain properties (e.g. no deadlock), and 2) it allows us to study the model without having to build a complete information retrieval system. The latter reason is very helpful as we will check if we implemented the model correctly. Mistakes made in this phase are more easily to correct compared to correcting mistakes in a more complete software system.

We will use *UPPAAL 4* [52] as our modeling tool. UPPAAL is designed to model networks of *timed automata*; a network of entities which can change their state either based on signals (as with traditional automata) or based on a change in time [4]. This is helpful for our situation, as we can model that servers are not allowed to bid after a certain moment in time has passed. Furthermore, UPPAAL has the ability to visualize the system, which allows rapid development. UPPAAL allows to check global properties of a system, expressed in a notation that allows to express time and system states; called *temporal logic* [5].

Each of our models is made out of three types of entities; a broker, and two types of servers: 1) good servers and 2) bad servers. Good servers behave like they have relevant information for queries and will act accordingly in the model. Bad servers are modeled to behave like they have no relevant results at all and hence to not participate in the economic model. We abstract from the actual information retrieval part of the servers, they simply return relevant (good server) or irrelevant (bad servers) results. This is an extreme situation, but as it is the same for all the six models we can still draw conclusions.

We created a general model that follows a basic cycle of steps (i.e. a path in UPPAAL terms): 1) a query is send to each server, 2) a server either sends back results or sends back a no-results-message, 3) the broker waits for a result-message or no-results-message from each server and starts the process over. There is also a *game over* path, in which a server remains if there are no more credits available. This general model is depicted as a flow graph in Figure 3.1. The upper part resembles the general model for the broker whereas the lower part resembles the general model for a server. De dashed arrows resemble communication between broker and server.

We checked two global properties on this general model with UPPAAL. First, we checked if the combination of a broker and any number of servers is deadlock free. In temporal logic this is stated as $\forall\Box\neg deadlock$, meaning that in every possible state of the system ($\forall\Box$) there is another reachable state and hence

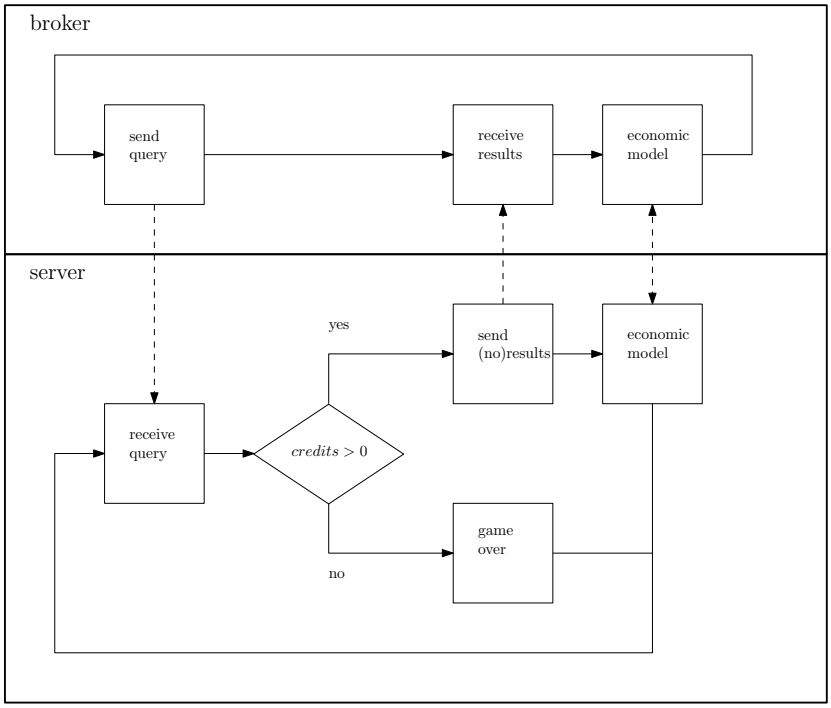


Figure 3.1: generic UPPAAL model

no deadlock ($\neg deadlock$). Second, we checked if the generic model rewards good behavior and punishes bad behavior, by checking if servers submitting bad results will eventually end up in the *game over* state. We added servers to the system which submit only bad results and checked if they always end up in the game over state: $\forall\Diamond server.gameover$. Where $\forall\Diamond$ means that in every state a future property will hold (i.e., $server.gameover$). Both properties are successfully with the general model.

We implemented the six remaining economic models using this generic model but in a very elemental way: we did not model the value of submitting results to a server (i.e. the increase in credits if a server is selected by the economic model) and we did not model the possibility that multiple server might win one round of the model. As every model is modeled with these restrictions, they are still comparable to each other.

With each of the six models implemented by extending the general model and hence respecting the checked global properties, we observed the time it takes for a model to stabilize. Our goal of this step is to further decrease the amount of economic models that we will further investigate, as we will only select the strongest models. I.e., models needing many cycles to punish bad servers and reward good servers are dropped and stronger ones are selected. We measure this with two measures. First, the difference between the amount of credits a good server has and a bad server has, once the model stabilized. We introduce two variables in order to express this difference: the starting amount of credits for a server S, and the credits a server had at a given point in time c. In the next sections we will cover the outcomes of the assessment of each

model. The second measure is the percentage of good servers being the winner from 100 executed queries, the more often a good server wins a query the better the model is for DIR usage.

Simple Market

The simple market has the downside that both good and bad servers pay a price for returning result sets, there is no distinction in behavior. The price that a server pays differs per query and per server, but finally all servers will end up with a deficit of credits. Hence, the difference between good servers ($c_g = 0$) and bad servers ($c_b = 0$) is $0 - 0 = 0$. From the queries that we executed, on average 45% was won by good servers.

Bond

The bond model only discredits the bad servers, as their bond is seized. Good servers are neither discredited or rewarded, but they will never end up in the game over state. Good servers will end up with $c_g = S$ credits and bad servers will end up with $c_b = 0$ credits, making the difference between good and bad servers $S - 0 = S$, the highest difference possible. From all queries that we executed, 100% was won by good servers.

First-Price Sealed Bid

The bid that a server places differs per server per query. The highest bidder receives the right to publish a result set and pays his bid. The height of a bid is dependent on how good a server thinks he can answer a query, so the good servers are modeled to place higher bids compared to the bad servers. It takes a relatively large amount of queries as only one server pays for every query, but finally the good servers will run out of credits and the bad servers will have all of their credits after which all the bad servers will be the winners of the auction. In the end, no server has credits left and the difference between good and bad servers is $0 - 0 = 0$. The percentage of queries that is won by good servers is in our setup 50%.

Second-Price Sealed Bid

As with the first-price sealed bid auction, this auction lets servers place their bid and only one wins. However, the server has to pay the second highest bid instead of his own bid. Because of the analogy with the first-price sealed bid auction, the difference between good servers and bad servers is again $0 - 0 = 0$. However, the percentage of queries won by good servers is higher as their average amount of credits to pay is lower: 65% of the queries is won by good servers.

All-Pay Auction

The all-pay auction makes no distinction between in behavior of good and bad servers; every server has to pay their bid. Every server pays their bid, making c differ per server, but in the end all servers will end up with no credits. The difference between good and bad servers is $0 - 0 = 0$. As with the two auctions depicted above, the good servers will place higher bids and will win the auctions

economic model	difference between good and bad servers	percentage of queries won by good servers
simple market	0	45%
bond	S (maximum)	100%
first-price sealed bid	0	50%
second-price sealed bid	0	65%
all-pay auction	0	45%
fixed price	0	50%

Table 3.2: results of model checking step

until they are out of credits. Because every server pays its bid, the good servers are out of credits earlier (having higher bids) compared to the bad servers. Only 45% of the queries are won by good servers, as the model behaves comparable to the simple market model.

Fixed Price

This model has the same effects as the all-pay auction. Good servers endure the same effects as bad servers and might end up in the game over state. It depends on the amount of the fixed price, but finally all servers will end up with no credits and the difference is again $0 - 0 = 0$. As every server pays the same price, the server that won is randomly selected by UPPAAL. Hence, 50% of the queries is won by good servers.

Model Selection

In Table 3.2 we summarize the results that we covered in the above subsections on model checking the remaining six economic models.

We select two models for further research. First, we select the *bond model*, as it has the highest difference in credits between good servers and bad servers, making it the strongest model to differ between good and bad behavior. Furthermore, the bond model has 100% of its queries executed by good servers in our setup, making it the best model to select the good servers. The second model that we select is the *second-price sealed bid* auction as it has the second-highest percentage of queries won by good servers.

3.2.4 Pareto Efficiency

As covered in Section 2.1.2, the Pareto efficiency is a validation measure for economic models. For second-price sealed bid auctions, it has been shown that this type of auction model is Pareto efficient by Sakurai et al. [41].

The bond model that Loder et al. [29] developed has also been shown to be Pareto efficient, but this cannot be translated to our situation. In the case of Loder et al. the bond posted for a particular email to a particular receiver can be seized by the receiver. The user is the only one who is influenced negatively by spam email (e.g., it takes time to clean up a mailbox), and gets compensated by the seized bond. This mechanism is Pareto efficient, as there is a justified

compensation and every party is better off. In our case there is no relation between server and user, but the broker is representing every user for every query. Hence, all seized bonds will end up at the broker. Furthermore, servers with relevant result sets are also negatively influenced by servers who submit irrelevant result sets. They should therefore also be compensated, as bad servers might block the results of good servers.

Given that both the broker will end up with all seized bonds and that there are servers who are entitled to receive compensation, the bond model is not Pareto efficient. We introduce a variant called a *bond redistribution* model, where both the broker and servers who need to receive compensation will receive part of the bond. In the next section we will cover this bond redistribution model in more detail.

Model Selection

From the two previously selected economic models, the second-price sealed bid auction is selected for further research as it is Pareto efficient. The bond model is not Pareto efficient, and is refined into a bond redistribution model where both the broker and servers will receive a part of seized bonds. We will further research this bond redistribution model instead of the normal bond model.

3.2.5 Model Selection Summary

The steps that we performed in this section, the models and the findings about these models are summarized in Figure 3.2. In this figure, models or model categories are shown as a rectangle, a bold horizontal line represents that a model is dropped. The steps performed are denoted as thick black boxes. The two economic models selected (bond and second-price sealed bid) are covered in more detail in the next section.

3.3 Selected Economic Models

The previously described steps ended in two models: 1) bond redistribution, and 2) second-price sealed bid auction. The second-price sealed bid auction is also known as *Vickrey auction*, which is how second-price sealed bid auctions will be referred to hereafter. In this section we will explain these two models in more detail and also in relation to DIR.

In Section 3.1 we introduced a generic model, analogously to Loder et al. [29] which we will repeat here briefly for readability purposes.

A result set for a given query is of value s to a server, and has costs c_s to return to the broker. For example, the advertisement incomes of a site visit might be calculated in s, knowing there is a probability that the result set will draw visitors. The costs to send the result set c_s are dependent on bandwidth costs for example.

The broker has similar variables as a result set is of value b to the broker and costs c_b to receive and process the result sets. The value b models how search engines (i.e. brokers) want to attract visitors by delivering good results and earn money by displaying advertisements. The costs for the broker c_b are for example power and storage costs.

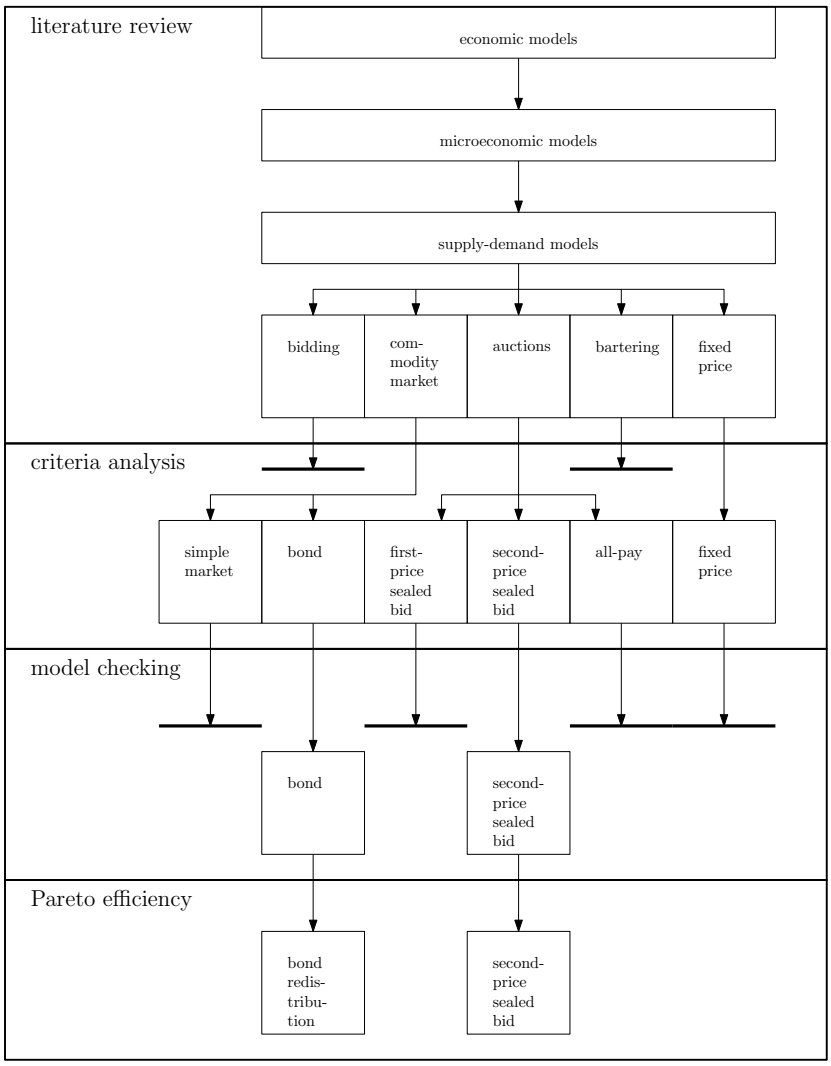


Figure 3.2: selection of economic models

3.3.1 Bond Redistribution Model

We previously introduced a stronger version of the bond model, where the broker does not end up with lots of credits. In this model there is a bond of size ϕ, which is placed by the server based on the server's belief on how good its result set is. Hence, servers who have a high trust in their result set are willing to place a high ϕ.

From all servers that participate in a given query, there are good servers which will return result sets containing relevant information and bad servers with result sets containing only irrelevant results. The set of good servers is denoted as G and the set of bad servers as B. As every server can only reside in one of the two sets, the two sets are disjoint and $G \cap B = \varnothing$.

A server will place a bond of size ϕ (based on its belief on how good the result set is) if they want to participate in a query. As all the bonds of the bad servers are seized, the broker will seize a total of $|B|\phi$ credits. From this amount of credits, the broker will subtract a fee f_b that covers the costs ($f_b \geq c_b$). The remaining credits $((|B|\phi) - f_b)$ are then distributed over the good servers. Every server that is in G will hence receive $\frac{(|B|\phi)-f_b}{|G|}$ credits.

The surplus that the broker expects for a query is then defined as $b - c_b + f_b$, the value of the result sets minus the costs to process them and plus the fee that the broker subtracts from all seized bonds.

The surplus for the server depends on the set that the server belongs to. If a server belongs to B, the surplus is $s - c_s - \phi$: the value of the result set minus the costs and bond. If a server belongs to G the surplus is $s - c_s + \frac{(|B|\phi)-f_b}{|G|}$.

3.3.2 Vickrey Auction

As with every auction, the broker initiates the auction by announcing the start of an auction of the right to return result sets. Every server which wants to participate replies by placing a bid φ. A Vickrey auction is sealed by definition, which means that servers cannot observe each other's bids and change their behavior in response.

When server i places a bid φ_i, there is a server j which has a lower or equal bid φ_j. Note that it is possible that only one server participates in a Vickrey auction. In the latter case the definition of bids still holds as $i = j$ and $\varphi_i = \varphi_j$. The probability that the bid φ_i is a winning bid is defined as p. Hence, with a probability of p, server i needs to pay φ_j, resulting in an expected value $p\varphi_j$ for server i.

The broker will earn the second highest bid (or in case of only one bidder, the only bid) for a given query, making the surplus for the broker $b - c_b + \varphi_j$.

The surplus for the server i is $s - c_s - p\varphi_j$, the value of the result set minus the costs and minus the expected value of the auction.

3.3.3 Summary

We summarize our formal description of the three final selected models in Table 3.3.

economic model	broker surplus	good server surplus	bad server surplus
bond redistribution	$b - c_b + f_b$	$s - c_s + \frac{(\lvert B \rvert \phi) - f_b}{\lvert G \rvert}$	$s - c_s - \phi$
Vickrey auction	$b - c_b + \varphi$	$s - c_s - p\varphi_j$	$s - c_s - p\varphi_j$

Table 3.3: selected economic models

3.4 Chapter Summary

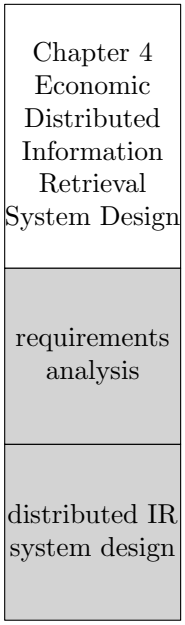
In this chapter the selection of economic models being suitable for our research is described. We started with the broad category of economic models, which we narrowed down to the category of supply-demand models. Subsequently, we performed a multi-criteria analysis on five economic model categories from the supply-demand models: bidding, commodity market, auctions, bartering, and fixed price. From the multi-criteria analysis the following economic models remained for further investigation: simple market, bond, first-price sealed bid, second-price sealed bid (i.e., Vickrey auction), all-pay auction, and fixed price. These models were model checked for a number of properties and only the bond and Vickrey auction model made it to the last step. In this last step we checked the two remaining models for Pareto-efficiency. The Vickrey auction turned out to be Pareto-efficient, and for the bond model we introduced the stronger and Pareto-efficient bond redistribution model. These steps are depicted in Figure 3.2.

These last two models have been described in detail in the last section of this chapter.

Chapter 4

Economic Distributed Information Retrieval System Design

In this chapter we cover the design steps of our distributed information retrieval system. We present the requirements that an economic distributed information retrieval system should fulfill and the general system design that we created. After reading this chapter, one could start building an economic distributed information retrieval system that follows our design and understand the remainder of this report.



4.1 Requirements Analysis

The process of coming to a set of requirements consists of three steps: 1) eliciting requirements, 2) analyzing requirements, and 3) recording requirements [35].

The first step involves interviews with stakeholders and creating agreement between different stakeholders on the desired requirements. As we will use our system for research purposes and not as a real-world search engine we perform this first step in a different manner. However, we will both run lab experiments as real-world tests with users. So, our system should be able to run both with and without users. We will state system goals as our requirements, and define constraints based on literature research.

The goals of our system are part of our research problem and hypotheses (see Chapter1) and can be stated as:

- Selecting the most relevant servers for a given query from the set of participating servers in the system, without centrally analyzing each server's corpus.
- Merging the results from selected server in such a manner that the most relevant results are ranked higher compared to less relevant results.
- Enabling servers to earn money with good behavior, hence allowing for new business models.
- Enabling system administrators to easily configure and maintain servers for the distributed system.

In order to use our system for research purposes we will add another system goal:

- Enabling researchers to use the system with or without users and calculating measures on how well the system performs.

The following requirements are actually constraints on the system, and follow from literature on distributed information retrieval [1]. These requirements should always be implemented in order to have a successful distributed information retrieval system.

- The system should pose no requirements on the type of database the server makes available to the system. I.e., every type of data could be made available to the system by a server.
- The system should relay queries in real-time, there will be no wait due to asynchronous steps in the algorithm.
- Every server should communicate with the broker using the same protocol, being an open standard to allow for easy usage.

4.2 System Design

We follow the methodology of Wieringa as described in his book on *design methods* [45]. Wieringa introduces a new method, called *Not Yet Another Method*

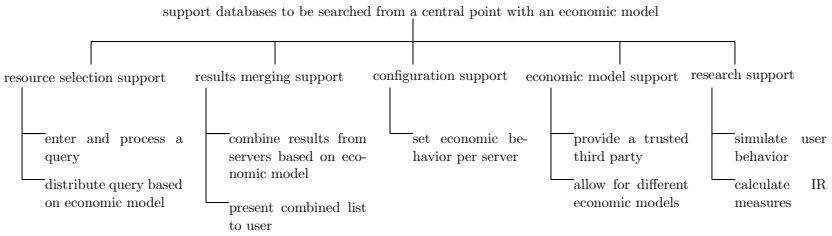


Figure 4.1: function refinement tree

that basically combines well known design and analysis methods (e.g. domain analysis, functional properties analysis) into one set of techniques.

One of the advantages of Wieringa's method is that it lists existing techniques, places those techniques in four categories and allows for *light-weight* or *heavy-weight* execution of the techniques. A *light-weight* execution basically means that from every category of techniques only one technique is executed, whereas with *heavy-weight* every technique from each category is executed.

We choose to execute the *light-weight* set of techniques, as it is not in our research interests to come up with a detailed system design. Wieringa has shown that the *light-weight* set of techniques is enough to give an understanding of the system to be built.

In the remainder of this section we will cover the most prominent technique from each of the four categories as defined by Wieringa, per category.

4.2.1 Functions

This category of system design techniques is closest to the requirements, as it will state the functions of the system that we are designing. We present a *function refinement tree*, a diagram that states the system's functionality in increasing detail. This diagram is shown in Figure 4.1.

The top node of the tree contains the general mission of our system, and relates directly to one of our research questions: finding out if economic models are suitable for distributed information retrieval. Hence, a system which allows distributed search with an economic model is to be build.

The mission is divided into five functions: 1) resource selection, 2) results merging, 3) configuration, 4) economic model and 5) research support. The first three main functions relate to the first three requirements. The fourth requirement, stating that server should be able to earn money with good behavior is divided over the three main functions being an integral part of the system as well as a separate function to allow for economic facilities. The last function (i.e., research support) is a special function as it is not necessary if one would build a search engine for a real-world environment, but only enables to research the system. This means that one can leave out this fifth function and still have a requirements fulfilling system.

At the bottom of the tree are the different services that the system must provide for different stakeholders of the system. Each function has multiple services. For readability we will not cover all of them, but present two examples. The first example is the service to type in a query. This service is provided to the end-user and allows this user to type in a query that will be processed by the

system. The second example is the *simulate user* service, which is represented by the research functions and allows us to run the system in a lab setting without having real users controlling the system.

4.2.2 Behavior

The second category of system design methodologies involves modeling the behavior of the system. The most elemental form is to provide a list of events and their desired effects. This list is provided below in Figure 4.2. This list is self-explaining. However, the last event covering researchers who are starting a simulation is only for the lab environment and can be left out if one is building a search engine system that is meant to work outside a laboratory environment.

4.2.3 Communication

The third category involves the design of the communication between different system parts. This step focuses on the communication between system parts are not on how to specify the system parts. The *Data Flow Diagram* is a suitable technique for a *light-weight* execution. In Figure 4.3 we show this diagram.

The key aspects of the diagram are the arrows, stating which types of data flow from which entity to another entity. The diagram states that there are three types of users in the system: 1) (normal) users, 2) researchers and 3) administrators (all denoted by a stick figure).

A *user* enters a query, hence there is query-data flowing from the user to the broker. The broker will add a category to the query and sends it to the economic model. The economic model is denoted as a data store, as it has a state that can be saved (e.g. if a server loses money, the economic model is changed and saved). From the economic model an offer is passed to a server, containing information about the query and the model. The server processes this offer and creates an acceptance (this might be a negative acceptance). In the next section we will cover offers and acceptances more extensively. With an accepted offer, the corpus of a server is used to return results. The broker then merges results and sends them back to the user. This whole process can also be executed by the *researcher*, but he will start one simulation which will send multiple queries. The simulation will collect and analyze the results, and provides the researcher with information retrieval results: the precision of the results.

The last flow in the diagram is from administrator to the server. The administrator can submit settings about the server (e.g. how much risk a server should take in the economic model). As these settings are not directly readable by the server and might differ per type of server, there is a process (denoted as a circle) that transforms the settings into economic model settings. These are send to the server and confirmed if valid.

4.2.4 Decomposition

The fourth and final category of design methods covers the global decomposition of the system into components. In Figure 4.4 we present a high-level decomposition, which is sufficient for our *light-weight* execution.

event: a user types in a query and presses enter

desired effect:

- the query is categorized if needed by the economic model
- those servers with the best economic values for the query or category are selected
- the query is send to the selected servers

event: a server submits his results

desired effect:

- the server is administered as having responded
- `if` all selected servers have responded
  - the results of all servers are merged, based on the economic model
  - the merged results are shown to the user

event: a server submits a new value according to the economic model

desired effect:

- check if the value is valid for the current economic model
- administer the value and confirm to the server

event: an administrator submits the configuration for a server

desired effect:

- check if the configuration is valid
- administrate the new configuration

event: a researcher starts a simulation

desired effect:

- configuration for the servers is created from a simulation setup
- queries are send to the system by the simulation
- results are measured by the simulation

Figure 4.2: event list

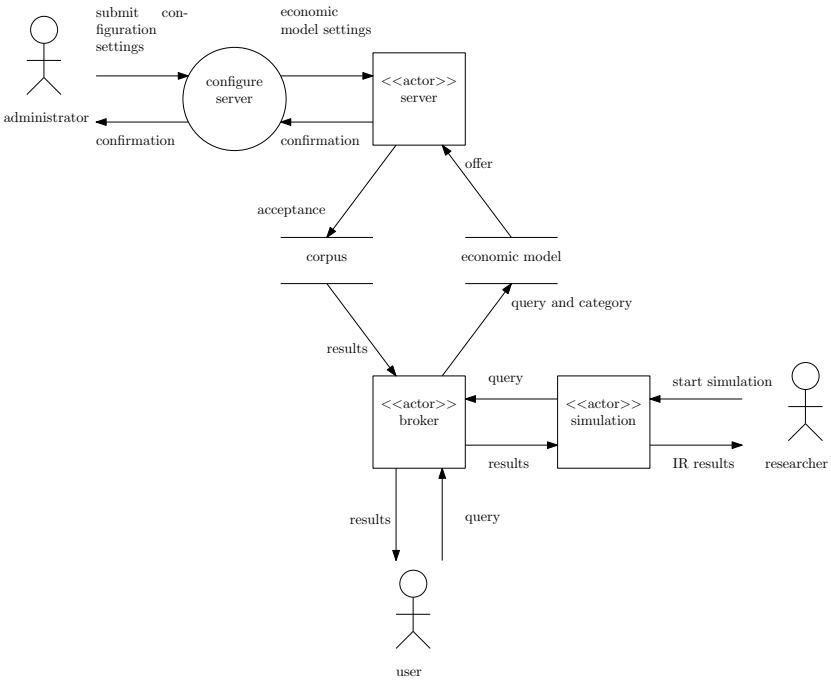


Figure 4.3: data flow diagram

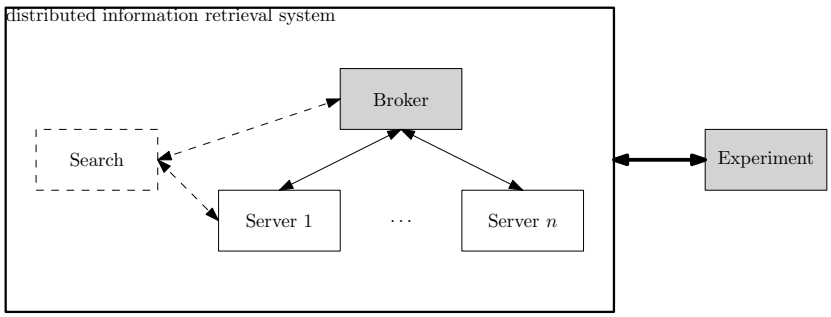


Figure 4.4: general architecture

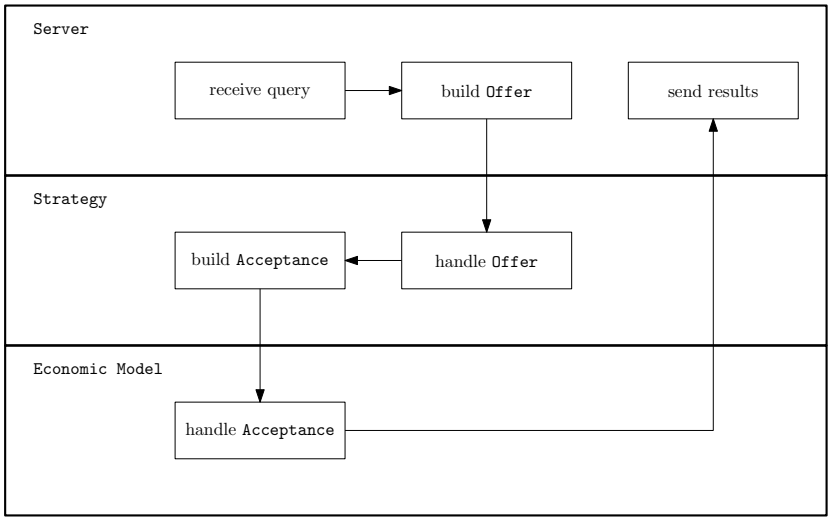


Figure 4.5: generic flow chart

The decomposition is made up out of a `Broker` component, a `Server` component and a `Search` component. The `Broker` receives queries from the end-user and redistributes them over participating servers. In case the server has relevant answers to the query, it will return results to the broker. For the communication between server and broker a `Search` component is used by both parties. This component contains all the shared functionalities that both parties use, such as the economic model and protocol.

As the economic models are a key part of our system, we will further decompose the parts of the server that are responsible for making economic decisions.

When a query has been received, an `Offer` instance is build which contains the query, the category, the current balance and the chance on relevant results for the category.

An `Offer` is first processed by the `Strategy` of the server. The `Strategy` contains the risk behavior of the server. Based on the chance that is stored in the `Offer` instance, a strategy might decide whether or not to proceed with the remainder of the economic model. The result of this step is an `Acceptance`, which states if an offer is accepted by the server and at which price.

As a last step the acceptance is handled by a specific economic model, which will handle the monetary effects of the decision made to accept the offer. Finally, the `Server` instance will return the results to the broker if the offer has been accepted by its strategy. In Figure 4.5 we show the relation between `Server`, `Strategy` and `Economic Model` in a simplified flow chart showing the components that we described as well as the data flow.

The two economic models that we will be investigating have different processes between query and results. In the next section we will further explain the differences and how they are finally implemented.

4.3 Economic Model Implementation

We built a local simulation program in Java which follows the design as described above. The only difference is that we left out the user, as queries will be send to the broker by the simulation self.

We used a *client-server architecture*, where each server (playing the role of client in the architecture) actually communicates with the broker (playing the role of server) over network hardware. In our setup, the servers and the broker reside on the same machine, but communicate by sending network messages to each other.

The Java program consists of four packages: `Server`, `Broker`, `Experiment` and `Search`. Each package contains a number of Java classes, which we will not cover separately in this report.

The broker can be run standalone, or an experiment from the `Experiment` package can be run. In that case, the servers and the broker are monitored by the experiment and predefined runs will take place. The results of each run are measured with regard to their precision.

The two models that we implemented in our simulation both behave differently. With the bond redistribution model a server reacts on every query, whereas with the Vickrey auction a server reacts on categories. In this section we will explain the implementation of the two models in more detail.

4.3.1 Generic Implementation

Some aspects of the implementation of both economic models are shared and seen as a generic part of the implementation. This subsection will cover those shared aspects.

Each server starts with a *category distribution*; a randomized distribution of documents over the categories that are used. From the category distribution, a server can fetch the amount of relevant documents that it contains for a given category. In the next section we will cover the different types of category distribution in more detail.

In the real-world, not every query in a category will lead to the exact same number of results as found in category distribution. Therefore, we introduced an error rate e, which is either 0, 5, 15 or 30%. This error rate introduces *false negatives*, meaning that in e% of the relevant results, the result is sent back as irrelevant. As there is no standard error rate for information retrieval systems, the four values allow us to study the behavior of the system under increasing error rates.

4.3.2 Bond Redistribution Model

Within the bond redistribution model servers make their decisions once the actual queries are received. Hence, the flow of this model starts with the broker who creates a random $\langle \text{query}, \text{category} \rangle$ tuple and broadcasts it to every server. The flow of this model is shown in Figure 4.6. This figure contains gray steps, which denote specific steps in order to use the system for research purposes. The gray bottom-right part of the flow chart denoted steps that we introduced to generate results, in a real-world situation some databases will be searched here.

Last, the gray box that states *create query and category* is in the real-world done by the user, who enters a query and the broker who categorizes the query.

Once the server receives the tuple it will check his balance. If the server has no credits left it will not participate in the remainder of the process, if there are credits available it will fetch the probability p on relevant documents for the query's category from his category distribution. If p is below or equal to a given threshold the server will return a message stating that it will not return results, otherwise it will fetch results and send them back. This threshold is dependent on the strategy that the server employs. With an aggressive strategy the threshold value is set to 0.3, with a normal strategy 0.6 and with the passive strategy 0.8.

If the threshold is met, the server will fetch ten or less results, which will be sent back to the broker as a result set.

The broker receives the result sets of each server and merges them round-robin, starting with the server which placed the highest bond. This means that the first result of the server with the highest bond is followed by the first result of the server with the second highest bond. If all first results are merged, the second results are merged and so on. Because we are only interested in the final top 10, we will stop the round-robin if we have a total of ten results.

From the merged results we calculate the information retrieval measures and the broker will send a new $\langle \text{query}, \text{category} \rangle$ tuple to all servers.

4.3.3 Vickrey Auction

The Vickrey auction has two phases. The first phase takes place before the broker starts sending queries to the server and is called the *bidding phase*. The second phase is the *query-result phase*. The complete flowchart is shown in Figure 4.7. As with the previous flowchart, the gray boxes represent steps that are specific to research usage.

Within the *bidding phase* every server decides on which categories to bid and how much. This phase is shown in the top of Figure 4.7. When the probability of relevant documents (retrieved from the category distribution) is higher than a given threshold, the server will place a bid on this category. This threshold is dependent on the strategy, with an aggressive strategy the threshold value is 0.3, with a normal strategy 0.6 and with the passive strategy 0.8.

According to the Vickrey auction theory [39], the best strategy is to bid only once and directly bid the amount that the object being auctioned is worth to you. We assume that the value of answering a query is related to the probability of returning relevant documents in a category as defined by the category distribution. Hence, we introduce a function where $bid = 10p + \varrho$ with p being the probability on relevant documents in a category and ϱ being a random number within that is within $[-p, p]$. This random number ϱ models that each server will value a query category differently, but still based on the query distribution.

A server sends his bids for the categories that have a probability (from the category distribution) above the threshold to the broker. The broker stores all bids from all servers in order for the broker to retrieve the servers that placed bids for a give category.

Once every server has placed his bids (which can be none at all), the *query-result phase* starts, depicted in the middle and lower box in Figure 4.7. This phase is similar to the bond redistribution model, with two differences: only

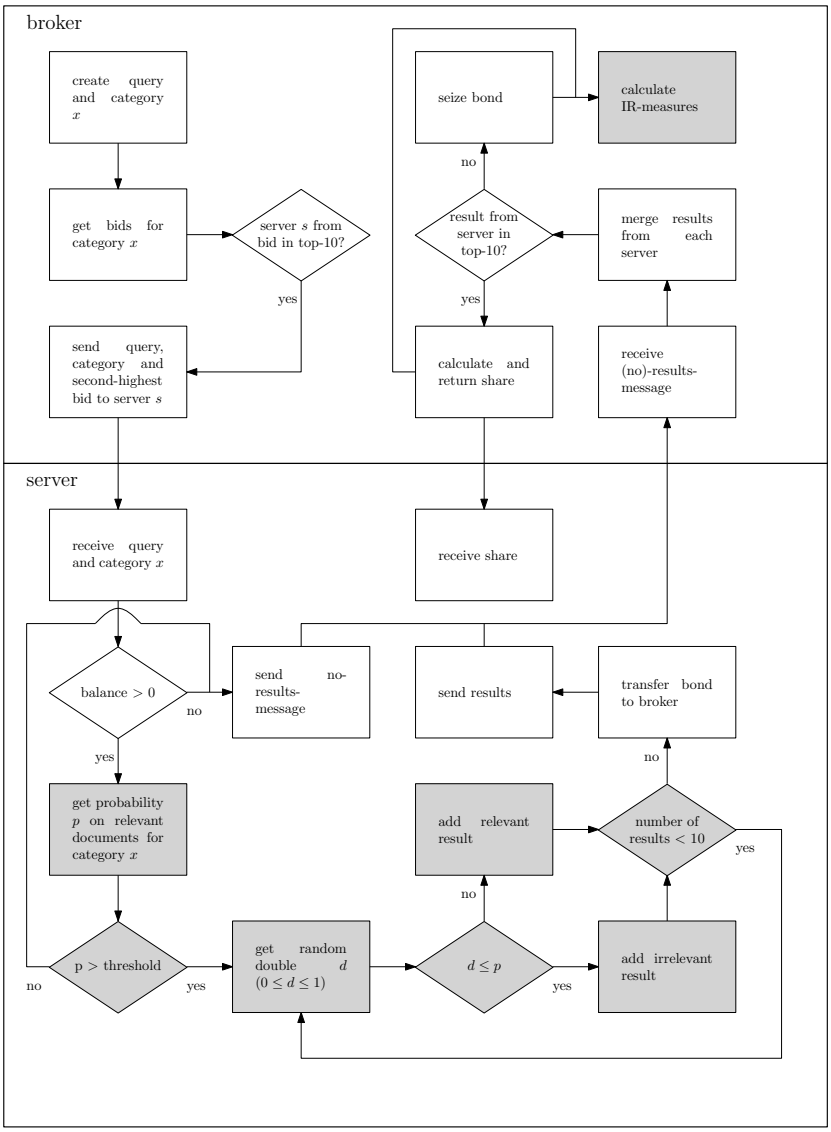


Figure 4.6: bond redistribution flowchart

servers that placed the highest bids for a category will receive the queries in that category and results merging is based on the height of the bids.

Once the broker has created a $\langle \text{query}, \text{category} \rangle$ tuple, it looks for the 10 highest bidders of the category. Those ten servers (or less if there are lesser than ten server who placed bids) will receive the query and the category. In case no servers have placed a bid for the category, we conclude that there are no results for this query.

When a server receives the query, it will always return a result set it as it has already determined that it will do so by placing a bid. Only if the server is out of credits, it will not send back result sets.

The broker receives all results and merges them with the round-robin algorithm. However, the highest bidder will be on top of the list followed by the second highest bidder and so on. This means that the first result of the highest bidder is followed by the first result of the second highest bidder and if all first results are merged the same order will be used for the second results. The round-robin algorithm will stop if there are 10 results and for these ten results the information retrieval measures are calculated. After this a new $\langle \text{query}, \text{category} \rangle$ tuple is created and distributed, with the same bids from the servers on the categories being applicable.

4.4 Chapter Summary

This chapter described the design of a distributed information retrieval system based on economic models. We began with defining the requirements of the system using Wieringa's method [45]. Next, the functions, behavior, communication and decomposition of the system were discussed. Subsequently, we introduced a separate branch of design decisions of research functions, in order to use the system design in a lab system as well as a real-world system. Finally, we covered a precise implementation of the Vickrey auction and bond redistribution model.

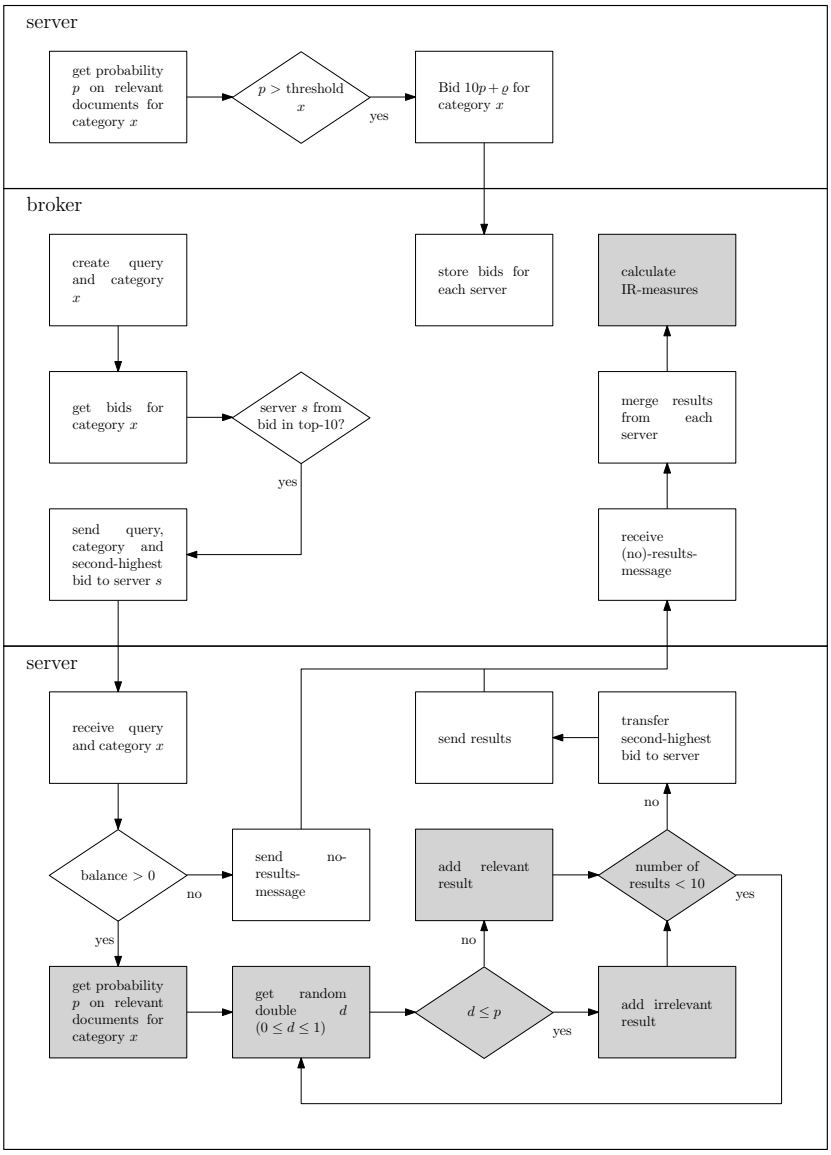
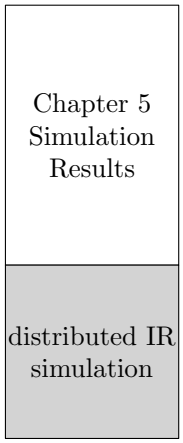


Figure 4.7: Vickrey auction flowchart

Chapter 5

Simulation Results

This chapter will present the results of the simulation of two economic models: 1) Vickrey auction, and 2) bond redistribution. We will cover which variables we measured and which ones were fixed. Furthermore, the results of several runs with our system will be reported.



5.1 Simulation Setup

In our setup there are a large number of variables. In this section we will cover every variable in the system and the values that each of the variables can take. For manageability reasons, we will fix as many variables as possible but without negatively influencing our research setup.

5.1.1 Fixed Variables

The *number of brokers* is fixed to always be one. Situations in which brokers might behave as a server to another broker are not considered in our research. However, this will not influence our findings, as we are solely interested in the mechanism of servers submitting results to a broker based on an economic model. How a server actually returns results does not influence the mechanism itself and it might even be the case that a server retrieves its results by distributing the query itself. As this does not influence the economic model, we will not consider multiple brokers for the sake of simplicity.

Every server will start with an amount of *credits* in order to participate in the economic model. We fix this amount at 1000 credits. Hence, the only way to obtain more credits is to actually behave well in the economic model and not because of a higher starting point.

In the two economic models there is a fee for the broker to cover the costs c_b, which is previously defined as f_b. In our simulation we choose to fix $p_b = 1$, as 1 is the smallest amount of credits that we can use, as we do not allow for floating numbers.

The *number of results* that a server will return is fixed at a maximum of 10. If a server has less than ten results, it is allowed to send back less results. The reason for this fixation is that research has shown that the top-10 results contain the most relevant results [23], and that results further down the list are of little relevance and almost never considered by the user as relevant documents.

The *algorithm to merge the results* of multiple servers into one result list at the broker is fixed to be round-robin (as explained in the previous chapter). In case the Vickrey auction is used, the first result of the highest bidder is followed by the first result of the second-highest bidder. When the first results are all merged, the same order is used for the second result of every server and so on. In case the bond redistribution model is used, the first server to send back his results is on top of the list (analogous to being the highest bidder).

In our simulation there is no usage of the actual *query*. The broker does send a $\langle \text{query}, \text{category} \rangle$ tuple to the servers, but the query is never used. The servers decide what to do based on the category of the query and the probability distribution, as we will describe in the next section.

5.1.2 Free Variables

There are five variables that are free in our simulation runs. The two most important ones are the economic model and the strategy. We also vary the number of servers to draw conclusions, the error rate and the way documents are split over different servers. For all free variables holds that they cannot change during a run.

The number of servers can vary between 20, 50, 100, 250 and 500. With more than 500 servers the simulation becomes unstable, due to the amount of local connections that are allowed by our operating system. Less than 20 servers is considered to be to less of a challenge, as there is a very low need to select servers in that case to come to a top-10.

With regard to *strategy* that a server can have, there are two possibilities: 1) either all servers have the same strategy, or 2) every server picks a random strategy from all available strategies. If we consider the latter case (i.e., random strategy) as a separate strategy that a server can choose from. Hence the strategy for every server is *aggressive*, *normal*, *passive*, or *random*. Every server keeps the same strategy during any given run. The strategy is responsible for two things: 1) deciding if the server should proceed with a query based on the probability on relevant answers, and 2) deciding the height of the value that a server assigns to the result set that he wants to return. There is no general optimal strategy for servers as it is highly corpus dependent. Servers with lots of relevant information on many subjects will flourish with an aggressive strategy, as it will answer many queries correctly. Servers with a silo corpus (i.e., many relevant documents on a few subjects) will be better off with a passive strategy: only answering queries for which the probability of a relevant answer is high.

The *economic model* is another free variable. Every server will hence have the same economic model in a given run. The fixation of the economic model per server makes sense, as it is impossible for the broker to work with servers with different models.

There is a probability that the categories of the results of a query do not match the categorization of a server. There are two reasons that an error might occur with respect to categorization. First, the server might have categorized documents in the wrong category and second, the categorization is very broad: on sub categorizations the server might have different relevance numbers (e.g., 60% of the documents related to food queries are relevant, but this is made out of sushi with a 90% relevance and schnitzel with a 30% relevance). To model the categorization error, we introduce an *error rate*, which is either 0, 5, 15 or 30% and hence a free variable. This error rate only introduces *false negatives*, meaning that the error rate is the percentage of relevant results being send back as irrelevant.

The *category distribution* of a server is randomly generated, but based on four different scenarios:

1 *silo scenario.* Each server is specialized in only one category, with a randomly generated number of relevant results which is high (0.8 or higher).

2 *equal scenario.* Each server brings his own new corpus which is randomly generated as the corpus from the equal scenario. The probability on a given relevance percentage is Gaussian distributed, with a mean of 0.4 (40%), as research has shown that this is a real-world probability on relevant document [42].

3 *mixed scenario.* Half of the servers have a category distribution based on the silo scenario and the other based on the equal scenario.

4 *split equal scenario.* Similar to the equal scenario, but there is one corpus for the complete system, with a random number of relevant and irrelevant documents for every category. This corpus is equally split over all servers.

variable name	fixed / free	values
number of brokers	fixed	1
number of credits	fixed	1000
broker fee	fixed	1
number of results	fixed	10
merging algorithm	fixed	round-robin
number of categories	fixed	15
number of servers	free	20, 50, 100, 250, 500
strategy	free	aggressive, normal, passive, random
economic model	free	Vickrey auction, bond redistribution
error rate	free	0, 5, 15, 30%
distribution scenario	free	silo, equal, mixed, split equal

Table 5.1: variables in our simulation

The category distribution contains 15 categories, using the same categories as the Open Directory Project (ODP) [48] does (see Appendix A). For every category a randomized number is drawn ranging between 0.0 and 1.0, based on the scenarios that we depicted above. This number represents the percentage of relevant documents in that category, where a score of 1 represents that every document in that category is relevant. Because we use a pseudo-random generator with a seed, the numbers drawn are the same in every run for every scenario and the servers are therefore identically configured in every run.

5.1.3 Summary

The variables that we covered above are summarized in Table 5.1, together with whether or not they are fixed and the different values for free variables.

5.2 Results

As we summarized in Table 5.1, our simulation five free variables: 1) number of servers (5 values), 2) economic model (2 values), 3) error rate (4 values), 4) economic strategy (4 values), 5) distribution scenario (4 values). This means that we have a total of $5 \times 2 \times 4 \times 4 \times 4 = 640$ different situations covered in our simulation.

We introduced a *naive standard*, where there is no strategy or economic model. In this standard, every server will receive the query and category, and result sets which are sent back are merged using the round-robin algorithm. We will be comparing the results of our two models with the naive standard.

The introduction of the naive standard, introduces $5 \times 4 \times 4 = 80$ new situations as there is no economic model or strategy used for the naive standard. We ran every situation three times, so a total of $3 \times (512 + 80) = 1776$ runs have been conducted.

For every run we calculated the precisions of every result list (i.e., the top-10) that the broker created and averaged this precision over every result list. We use two measures, the precision at five ($P@5$), and the precision at ten ($P@10$,

i.e., the complete list). Both measures denote how well the system managed to present relevant results in the first 5 results or the complete list. We use both precision measures, to study how relevant results are distributed over the final ten results; e.g., if $P@5$ is high and $P@10$ is low, the most relevant results are within the first five results.

In the next subsections we will cover the results of each model. In these sections we will consider results as significant if the difference between the result and the comparison case is 10% or more.

5.2.1 Bond Redistribution Model

In this subsection we will cover the influence of each of the four remaining free variables (the economic model is a free variable on its own) on the two outcome measures (i.e., $P@5$ and $P@10$) for the bond redistribution model. We will start with describing differences between the two outcome measures and end with a comparison between the naive standard and the bond redistribution model.

Outcome Variables

For all of the four remaining free variables, we did not found significant differences between the precision of five results and ten results. The highest difference is -0.04 between P@5 and P@10, which is not considered to be a significant difference ($< 10\%$). Hence, the relevant results are equally distributed over the top-10 results when using a bond redistribution model. In the next subsections addressing the four free variables, we will no longer cover $P@10$ and $P@5$ separately, as we have just shown that there is no significant difference between the two.

Number of Servers

The number of servers does not influence the outcomes for any bond redistribution model run that was executed. As can be seen in Figure 5.1 and Figure 5.2, each line is fairly flat. For individual runs, there are some small significant differences, but when we averaged the differences between each value (e.g., between 20 and 50 servers) for every run, there is no trend as the average difference stays below the 10%. Hence, the number of servers makes no difference for the performance of the bond redistribution model.

Strategy

The strategy significantly influences the results. As can be seen in Figure 5.1, the relative increase in performance for different economic strategies is roughly the same over increasing number of servers. This is also true for other variables. The passive strategy scores ≈ 0.12 higher than the normal strategy, which scores ≈ 0.1 higher than the random strategy run, which scores another ≈ 0.1 higher than the aggressive strategy. This effect turns out to be the same for every error rate; the size of the differences is constant across the different error rates.

Error Rate

We show the $P@10$ results for the silo distribution with different error rates in Figure 5.1, each sub-figure shows the differences between economic strategies. Both the type of outcome variable and the distribution scenario do not influence these results, so we can draw conclusions using solely the silo distribution and the $P@10$ measure.

As can been seen in Figure 5.1, the error rate has a big influence on the precision. In our results, we found that for every 1% increase in the error rate the results became 1% worse. Hence, the system reacts linear on the error induction.

Distribution Scenario

In Figure 5.2 we show four figures of bond redistribution runs, where the only changing variable is the distribution scenario. Even in extreme situations where servers contain only relevant documents in one category (silo distribution), there is no influence on the precision. For all other runs we see the same trend; the distribution scenario does not influence the quality of the results. With regard to the distribution scenario we conclude that there are no relevant results using the bond redistribution model.

Comparison with Naive Model

The naive baseline performs significantly worse compared to the worst strategy (≈ 0.18). For every run executed, this difference is the same and the bond redistribution outperforms the naive model.

5.2.2 Vickrey Auction

Vickrey auctions show some identical result trends when compared to the bond redistribution models, but also different results on certain variables. In the next subsections we will cover the differences in the outcome variables, the four remaining free variables and the comparison with the naive model.

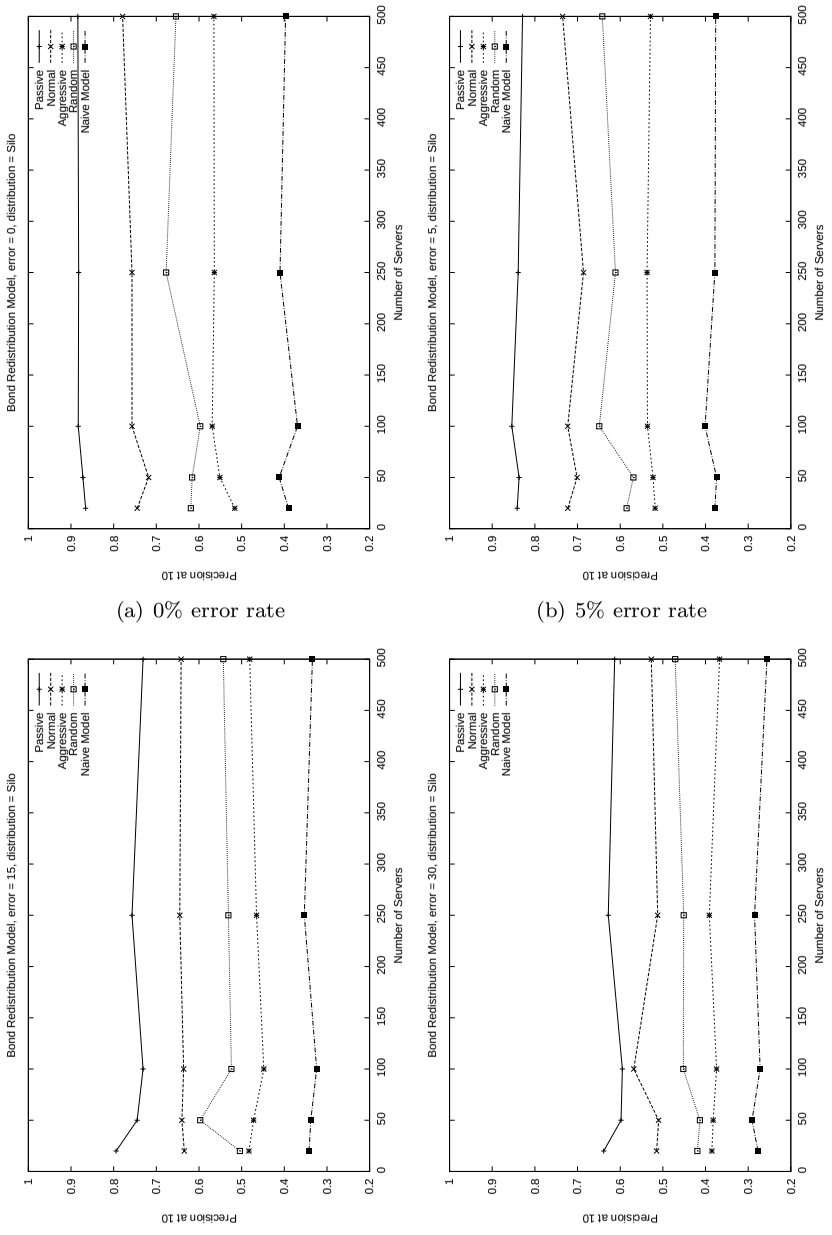
Outcome Variables

As with the bond redistribution model, there is no significant difference between $P@5$ and $P@10$. The relevant results are equally distributed over the top-10 results with the Vickrey auction.

Number of Servers

Vickrey auctions show different results with different numbers of servers and different distribution scenarios as we show in Figure 5.3. If servers behave like silos, the precision of the results is constant, no matter the number of servers that participate. The differences between the economic strategies are very low in this case, and not significant between the normal and aggressive model. Only the passive model scores significantly higher compared to the other strategies.

For the remaining three distribution scenarios there is a significant positive trend between the number of servers and the precision.



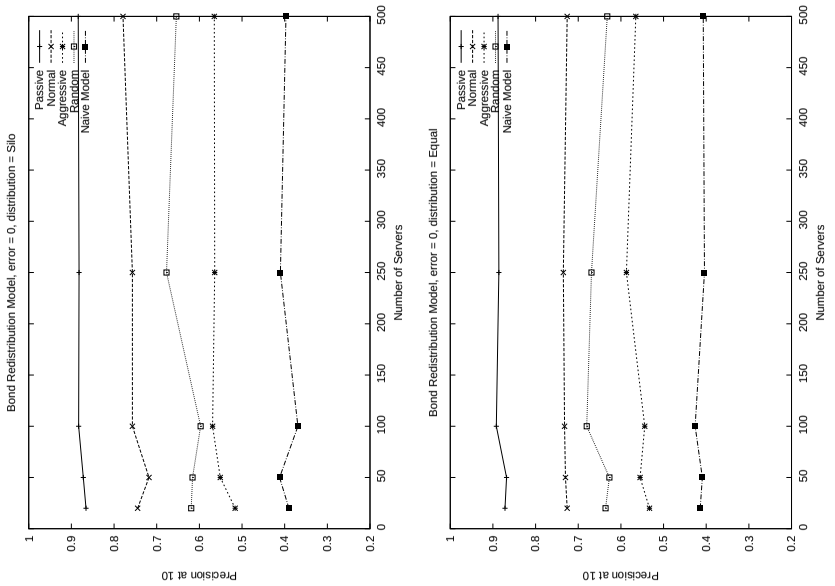
(a) 0% error rate

(b) 5% error rate

(c) 15% error rate

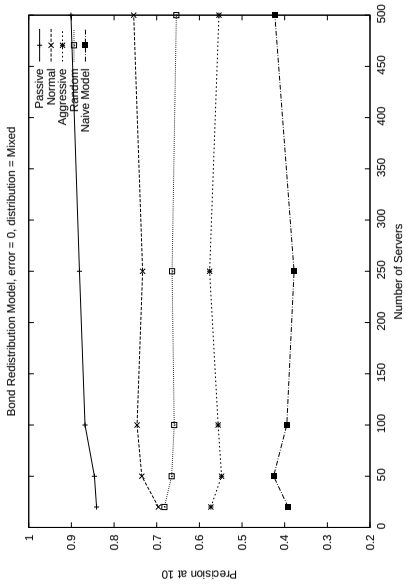
(d) 30% error rate

Figure 5.1: bond redistribution with different error rates

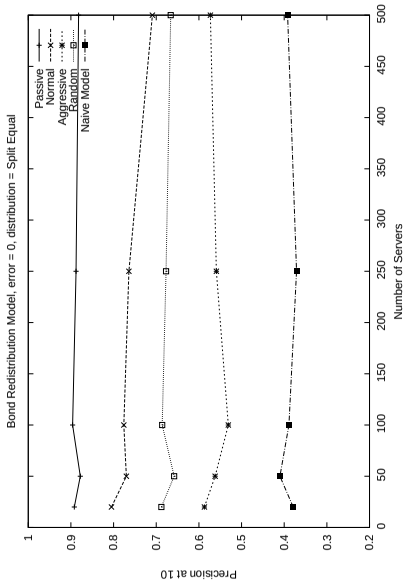


(a) silo scenario

(b) equal scenario

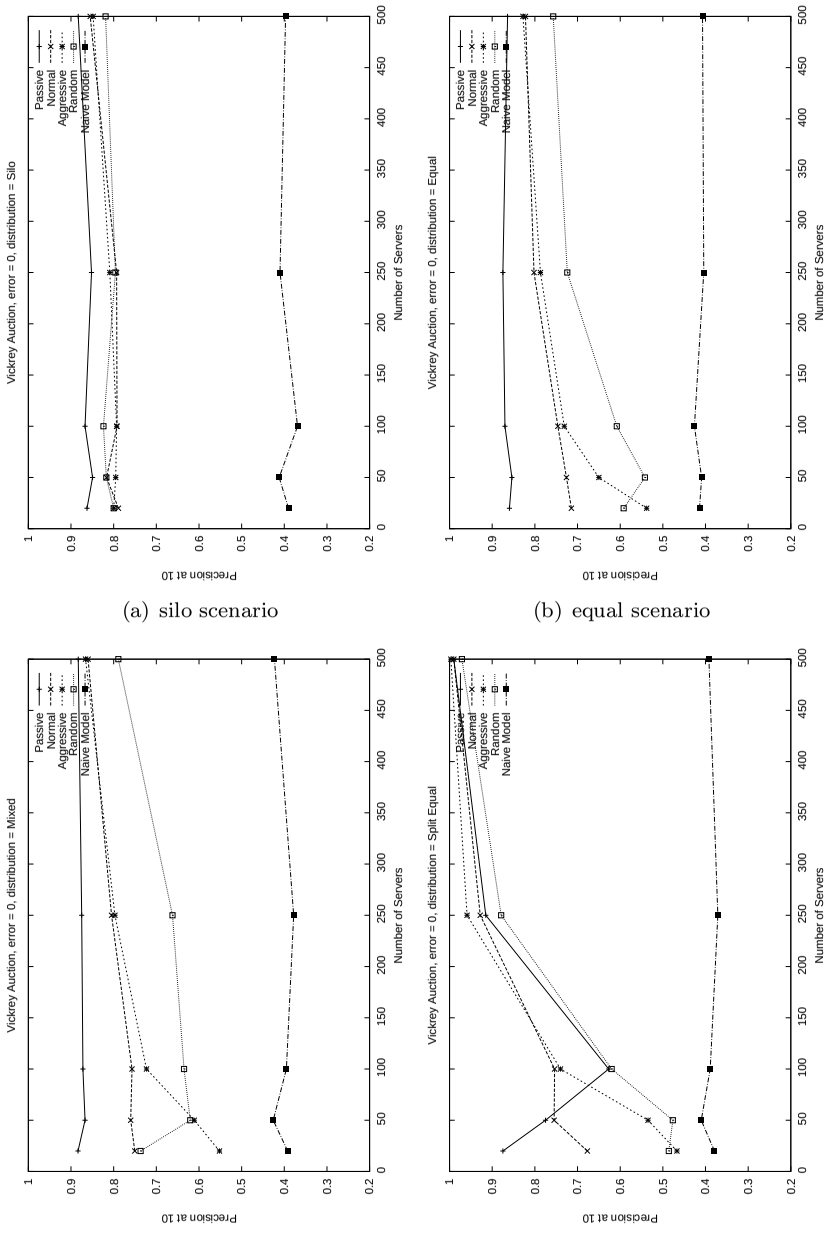


(c) mixed scenario



(d) split equal scenario

Figure 5.2: bond redistribution with different distribution scenarios



(a) silo scenario

(b) equal scenario

(c) mixed scenario

(d) split equal scenario

Figure 5.3: Vickrey auction with different distribution scenarios

Strategy

The influence of the strategy differs per distribution scenario of the Vickrey auction.

For the silo scenario, there is no significant relation between strategy and precision.

For the mixed scenario and the equal scenario, the passive strategy behaves better than the normal strategy, which behaves better than an aggressive strategy and the random strategy behaves worst. However, when the number of servers increases, the precision increases noticeably for the random, aggressive and normal strategy. They grow towards each other (to a $P@10$ of ≈ 0.8) and there is almost no significant difference.

Finally, we consider the split equal scenario, where a big corpus is split over every server and hence the number of relevant results per server per category decreases if the number of servers increases. Independent of the other free variables we see the same trend: the aggressive and normal strategy grow to the same point when the number of servers grows to 100, whereas the passive strategy declines and the random strategy grows to this same declined value. When the number of servers increases to 500, all the four model grow to the same high score (with no significant differences), which is close to perfect in the runs with no error rate.

Error Rate

The influence of the error rate is linear and similar to the bond redistribution model; increasing the error rate decreases the precision with a constant factor per added percentage of errors.

Distribution Scenario

There is no significant difference between the mixed distribution scenario and the equal scenario. The relation between all other variables is the same for these two scenarios.

The split equal scenario and the silo scenario yield different results. We covered most of the results already in the subsections on the strategy and the number of servers, but in general a silo scenario leads to good precision results, irrespective of any other variable. The split equal scenario has the strongest relationship between the number of servers and the precision, which approaches to perfect, but at the same time this scenario has to most unstable performance with a smaller amount of servers.

Comparison with Naive Model

As with the bond redistribution model, the Vickrey auction performs better in all cases when we compare it to the naive baseline. The difference in precision differs between the number of servers, but on average the Vickrey auction performs 70% better compared to the naive model.

5.2.3 Separate Results Merging

In order to test the effects of results merging and resource selection separately, we ran all runs again but with a default round-robin merging algorithm instead of an economic merging algorithm. We measured the difference in precisions between a run with economic results merging and without economic results merging. On average there is almost no difference between the two runs (a difference of 6×10^{-4}), which holds for most of the runs.

However, for Vickrey auction runs with 250 or 500 servers and an aggressive strategy there is a sharp increase in precision by economic merging, the precision increases with $\approx 15\%$. This effect is independent of the distribution scenario that is chosen. For all other runs the measured difference is not significant.

5.2.4 Kendall's τ Test

One of our hypotheses (H1) that we tested in this research is that well behaving servers are rewarded for their good behavior. This means that servers which send back many relevant results to the broker are among the richest participants of the economic model.

To test this hypothesis, our simulation created two lists after every run. The first list is a ranked list of the number of relevant results returned by each server, with the server returning the most relevant results on top. The second list is a ranked list of every server's amount of credits, with the richest server on top. These two lists are compared using the *normalized Kendall's* τ measure. This measures how similar two ranked lists are, where a value of 1.0 means that both lists are ranked identically and 0.0 means that there is absolutely no correlation between the two lists.

For every run we calculated the normalized Kendall's τ (denoted as τ in the remainder of this section) between the list of servers ranked by the number of relevant results that they submitted and the final amount of credits of each server. A high τ can be reached in two ways: 1) a server returns many irrelevant results and ends up with little or no credits, or 2) a server returns many relevant results and ends up with many credits.

We added the τ for each of the executed 1776 runs to a list. As every run is executed three times, we averaged the τ for each run, ending up with 576 values on the list. The list ranges from a τ of 0.93 (an aggressive Vickrey auction with 50 servers and 5% errors, with a mixed scenario) to a τ of 0.03 (a random Vickrey auction with 500 servers, 0% errors and a corpus that is split over the servers).

We will categorize the τ scores in three categories: a strong correlation ($\tau > 0.6$), a medium correlation ($0.3 \leq \tau \leq 0.6$) and a weak correlation ($\tau < 0.3$). When analyzing the list using these categories we conclude the following:

- There are 128 runs in the strong relation category, 67% of these are Vickrey auction runs, 30% are bond redistribution runs and the remaining 3% are naive runs.
- There are 289 runs in the medium relation category, 13% are Vickrey auction runs, 66% are bond redistribution runs and 21% are naive runs.
- There are 159 runs in the weak relation category, 83% are Vickrey auction runs and 17% are bond redistribution runs.

variable name	weak correlation	medium correlation	strong correlation
number of servers			
20	24%	16%	29%
50	12%	23%	31%
100	13%	35%	18%
250	20%	21%	11%
500	30%	16%	10%
strategy			
aggressive	12%	26%	42%
normal	2%	19%	42%
passive	50%	27%	2%
random	36%	28%	14%
economic model			
Vickrey auction	84%	16%	69%
bond redistribution	16%	84%	24%
error rate			
0%	25%	24%	27%
5%	25%	25%	26%
15%	25%	26%	23%
30%	26%	25%	24%
distribution scenario			
silo	22%	26%	26%
equal	26%	19%	34%
mixed	23%	25%	27%
split equal	28%	29%	13%

Table 5.2: Kendall's-τ overview

- Aggressive and normal strategies can be found more often in the strong relation category (81% of this category are runs with these strategies), whereas the random and passive strategies are more often found in the weak relation category (69% of this category are runs with these strategies).

- With regard to the other free variables (error rate, distribution scenario, number of servers) there is no conclusion to draw as they are roughly equally distributed over the three correlation categories.

Apart from the results we listed above, we present in Table 5.2 all percentages of Vickrey auction and bond redistribution runs related to the correlation categories (due to rounding errors, the sums might not be 100%).

variable	effect	
	Vickrey auction	bond redistribution
number of servers	no	yes
strategy	yes	yes
error rate	yes	yes
distribution scenario	yes	yes

Table 5.3: summary of experiment results

5.2.5 Chapter Summary

We presented three different results in this chapter; 1) the results of different variables on the precision outcomes of the model, 2) the results of a separate experiment on resource selection, and 3) the results of Kendall's-τ test.

The first type of outcomes showed that there is no difference between the $P@10$ and $P@5$ outcome measures. Free variables influencing the precision of the model are summarized in Table 5.3. Because of the complexity of some of these relations (which might differ for different values of free variables), we will not summarize these here.

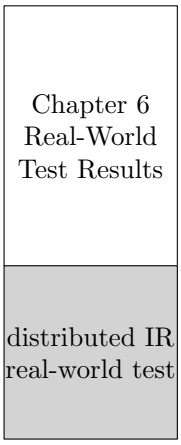
The second outcome measures showed that economic results merging makes no difference compared to non-economic results merging, except for Vickrey auctions with a large amount of servers and an aggressive strategy.

From the Kendall's-*tau* test we learned that the Vickrey auction has the strongest relation between information retrieval performance and economic performance. Therefore, we will continue the real-world test in the next chapter with the Vickrey auction as economic model.

Chapter 6

Real-World Test Results

To verify our simulation results as we presented in the previous chapter, we executed experiments with users in a real-world environment. In this chapter we will present the experimental design and its results. We will use the Vickrey auction as economic model, as it turned out in the previous chapter to have the strongest relationship between information retrieval results and economic results, as well as the best performance in the mixed distribution scenario. The latter scenario is the one used in our real-world test, as we will show in this chapter.



6.1 Experiment Design

Our main goal of these experiments is to examine to which degree real users value the results and manageability of an economical distributed information retrieval system. For our experiment we need to add or change certain parts of our simulation software:

- Add a search *web interface*, so users can type in queries and rate the results.
- Change servers in order to fetch and return real data from *real servers*.
- Add a *human provided* economic model *configuration* for each server.
- Change the generated queries to *real queries* that are from the same domain as the servers.
- Change the generated categorization by 1) categorization from a *categorization engine* and 2) a human categorization.
- Add a *comparison* case, in which users can enter the same queries and get results from the same corpus.

In the next sections we will cover the general design on each of these parts.

6.1.1 Web Interface

The first change, a search interface, was realized by building the distributed system again in a *PHP* [50] version. This version runs on a web server, and is accessible to users. It has no connection to our Java implementation, but is a standalone implementation of the system as designed in Chapter 4.

Our interface allowed users to enter a query and presented them with results in a two-column layout. The left column showed the results from a centralized search engine and the right column our economic distributed system. Users were asked to select relevant results and submit them back. From this we can calculate the precision of our results, but also compare our results to those of the centralized search engine.

6.1.2 Real Servers

We added real servers by using the Wikia [53] *wiki farm*. A wiki farm is a set of wiki websites. On a normal wiki website, a subject (e.g., music) has its own page and any sub-subject (e.g., blues music) is a separate page where the former page (i.e., music) refers to. In a wiki farm, the music page is a complete music wiki with a page on blues music. From now on we will refer to these individual wiki pages from the Wikia wiki farm as *wikia*s.

We manually selected 100 wikias, as we found out that the results from our simulation stabilizes from 100 servers (see the figures and explanation in Chapter 5). We used the 15 top-level categories from the Open Directory Project (ODP) [48] to ensure wikias from every category were included. Some wikias that we selected were silo servers, with only knowledge on one category, whereas other wikias had knowledge of multiple categories. Hence, this real-world test has a mixed distribution scenario.

6.1.3 Human Provided Configuration

The human provided configuration for the economic model is generated in a separate experiment that we conducted. We asked 35 users with an IT background to assess the 100 wikias. Users were asked to study a wikia and rate the contents of the wikia. For each of the 15 categories, users were asked to answer the question "how good is this wikia in answering queries from this category?" with a number between 1 and 10. The result is that for every server we know how well this server is in answering queries from each category.

Furthermore, we asked users to imagine the wikia that they assessed is able to earn money by correctly answering queries. We asked them to assign an economic strategy to the wikia page: answering many queries in many categories (aggressive strategy), answering less queries in categories that are certain to return relevant answers (passive strategy), or a strategy residing in between (normal strategy). This is a high abstraction of our economic model, but is chosen so to test if our hypothesis about the easiness of configuration (H7) is true. This can only be tested by using an easy and high-level abstraction to our economic model.

From the category distribution and the economic strategy assessed by users, we have a human provided configuration for every server.

6.1.4 Queries

We changed the generated queries from our simulation into queries that we harvested from the selected wikias. For each of the 100 wikias, we selected three random pages. We extracted the titles of these pages and used these as queries (300). As we have found out that our simulation stabilized after 50 queries, we will use a random set of 50 queries from these 300 queries.

The random selection of 50 queries from 300 queries, resembles both our simulation as it introduces some variance in the number of queries from a given category (i.e., the 50 queries will not be evenly distributed over the 15 categories). Furthermore we believe that in real-world usage of search engines, this variance is also present and might even be stronger.

6.1.5 Categorization Engine

To allow the Vickrey auction to work, we need the queries to be categorized, as the servers place their bids on the categories beforehand. We used the ODP as categorization engine to do this. One can query the ODP and obtain a ranked list of the most relevant categories for the query. As we will only use top-level categories, we extracted these from the first five relevant categories returned by the ODP. The top-level category which occurred most in the five returned categories is selected as the query category.

We manually checked the categorization returned by the ODP and found that from our 300 queries, only 76 got categorized correctly. This means that $\approx 75\%$ of the categories are incorrectly classified by the ODP.

Therefore, we also performed a human classification of the same query set to the same top-level categories of the ODP. For each of the 50 selected queries we created a *query description*, that shortly described the purpose of the query.

Based on both the query and a query description that we provided, one human accessor placed each of the 50 queries in a category.

6.1.6 Comparison

As stated above, we show the results of the Wikia centralized search engine in the left-column of the web interface. We submitted the query the user provides to the wikia search engine and processed the results, deleting results coming from wikias which are not in our set of 100 wikias. This means that we use the same corpus for both the centralized search engine as our economic distributed information retrieval system.

Users were asked to rate the relevance of both the centralized as the distributed results, and the relevance for the central engine is used as comparison case to the results of our system.

6.2 Results

The experiments that we conducted resulted in two categories of results. First, we asked the users who configured the 100 servers to rate the easiness of the configuration task. Second, the results with regard to the precision of our search engine and compared to the centralized search engine from Wikia.

6.2.1 Administrative Usability Results

The 35 unique users that configured the economic model by judging the 100 wikias, also filled in a short survey on how they perceived the task of configuring the economic model. We asked them how easy they perceived the configuration task and how helpful the notion of an economic model is for this configuration task. Furthermore, we asked the users how experienced they are within the IT-field. The results are shown in Figure 6.1, Figure 6.2, and Figure 6.3.

Although we asked people with an IT-background; 11 respondents think of themselves as inexperienced or very inexperienced. After asking some of the respondents why they might rate themselves as inexperienced, it appeared that people interpreted the question as being asked about their experience in advanced information retrieval knowledge.

With regard to the easiness of the configuration task, the majority of the respondents judge the task as normal, easy or very easy (80%). The helpfulness is also rated slightly positive or normal (71%). On average, the respondents rated both the easiness and helpfulness with a positive answer.

6.2.2 Information Retrieval Results

We asked 10 experienced IT-users to execute five queries from our set of 50 queries in two experiments. In the first experiment, the queries were categorized by the ODP and in the second experiment by our human accessor. The users rated the results of both the centralized wikia.com search engine and our economic distributed information retrieval system on relevance (i.e. they submitted which results are relevant and which are not).

From the returned relevance count of each query, we calculated the precision at ten ($P@10$) for the two search engines. If a search engine submitted $x < 10$

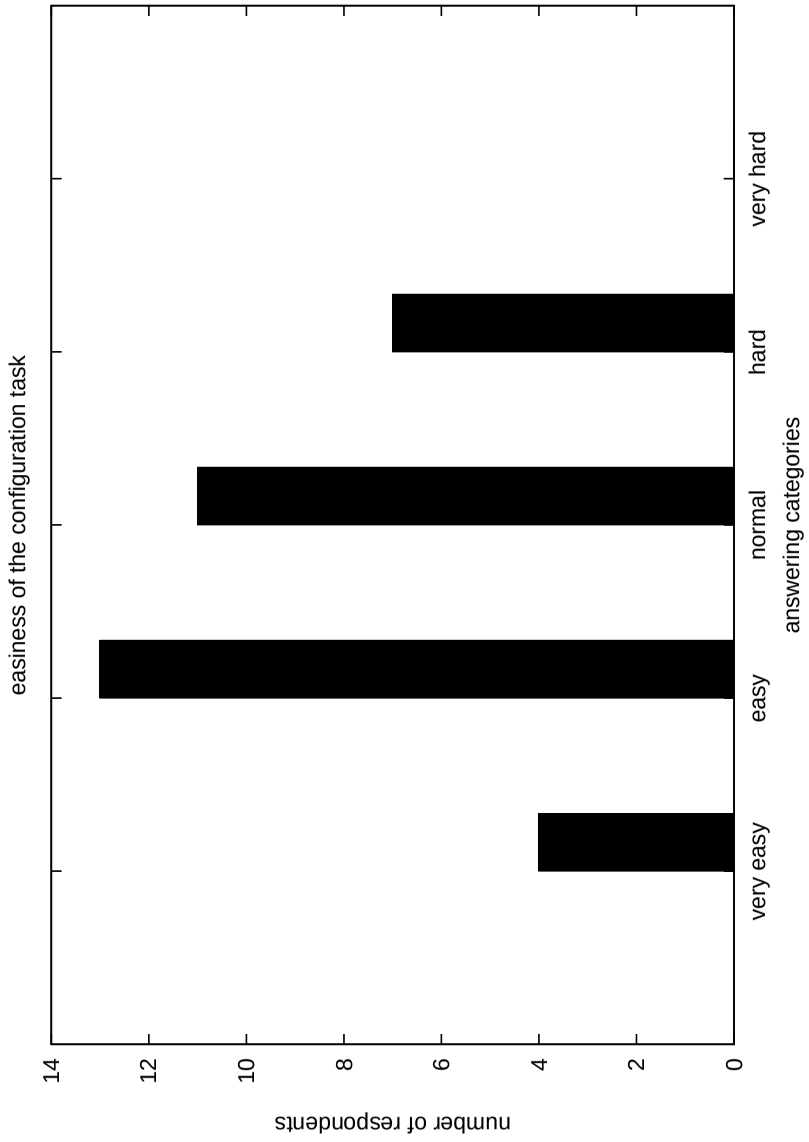


Figure 6.1: easiness of the configuration task

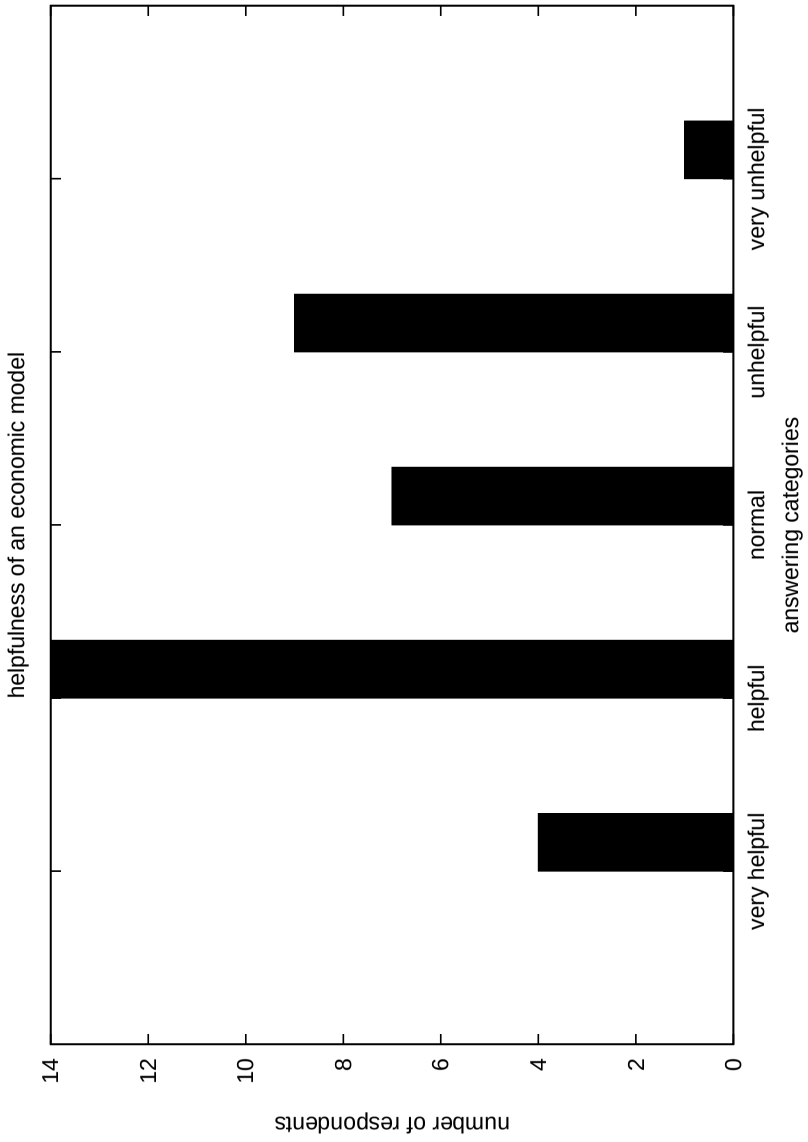


Figure 6.2: helpfulness of an economic model

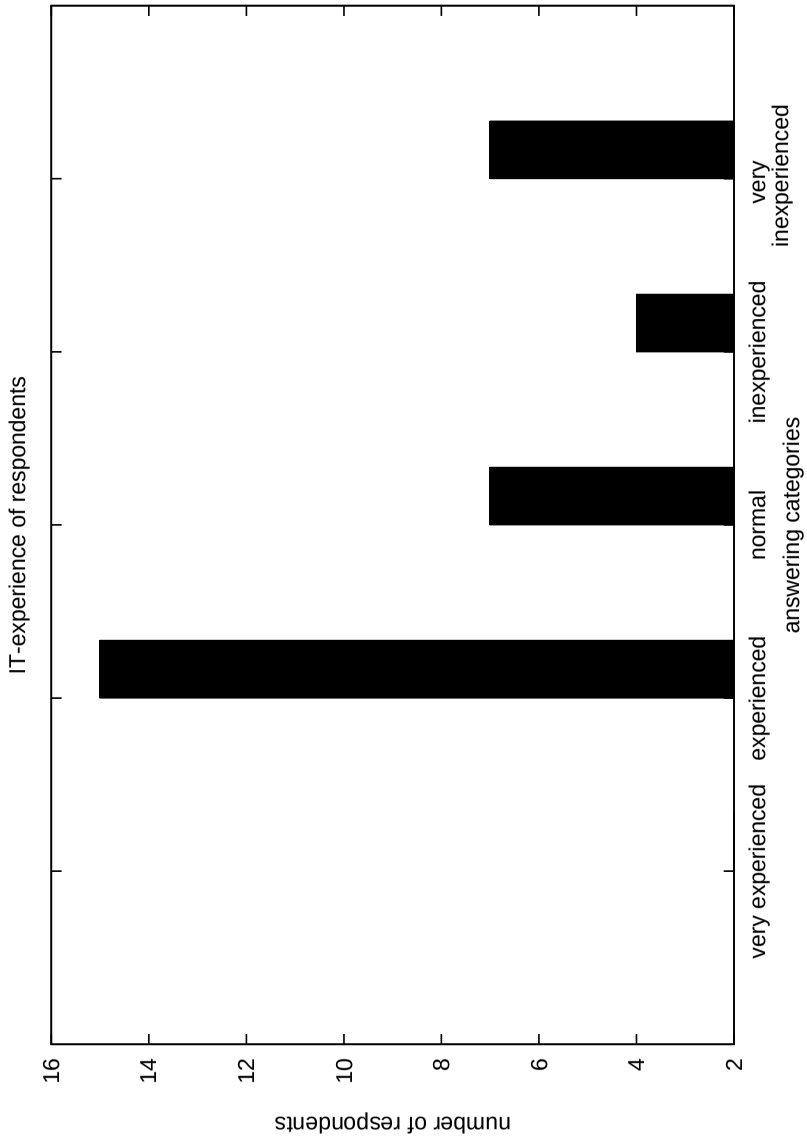


Figure 6.3: IT-experience of respondents

query categorization engine	average precision of centralized engine	average precision of economic information retrieval system
ODP	0.3	0.1
human accessor	0.3	0.5

Table 6.1: information retrieval results

results, we calculated the $P@x$ instead of the $P@10$. This means that if one of the search engines submits only three relevant results, the precision is considered to be 1 ($= \frac{3}{3}$). This is a valid choice, as it would be unfair to give a server with three relevant results out of three total results the same precision as a server returning three relevant results within ten results. Finally, we averaged the precision scores from the 50 queries, which are shown in Table 6.1.

From these results we can see that the ODP categorization engine decreases the precision tremendously, to a very low score of 0.1. However, when we use our perfect categorization, the economic model outperforms the central Wikia search engine. In a real-world situation, we cannot achieve the perfect classification, but as better categorization algorithms exist [27, 37], it is plausible that the economic model scores at least as good as a centralized engine.

6.3 Chapter Summary

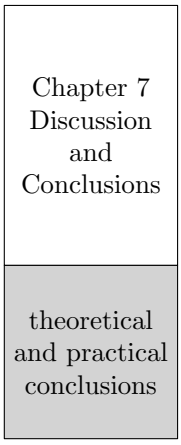
We executed a real-world test which we presented in this chapter. We used real queries and servers, both created from the Wikia wikifarm. We asked users to configure the servers, and to rate the difficulty of this task. On average, the users found it is easy to administer a system based on an economic model and they also found the notion of the economic model helpful.

We asked users to execute two sets of 50 queries on our search engine. The first set was categorized by the Open Directory Project (ODP) categorization engine and the second set by a human accessor. Users viewed the results from both the central Wikia search engine and from the economic distributed information retrieval system with a Vickrey auction. Subsequently, users stated which results were relevant to the query. The ODP engine has a misclassification rate of 75%, and users rated the precision of the results for the queries categorized by ODP as low as 0.1, whereas the results of the central engine was rated as 0.3. When we used the human accessed categorization, the economic model outperformed the central engine with a precision of 0.5 against 0.3.

Chapter 7

Discussion and Conclusions

In this chapter we will summarize our findings and relate them to our research problems and hypotheses, thereby drawing conclusions. Furthermore, we will discuss possible drawbacks of our approach and introduce suggestions for future research.



7.1 Hypotheses Testing

In this section we will cover the hypotheses outlined in the introduction of our research. For each hypothesis we will cover whether or not the hypothesis turned out to be true, and how we conclude this.

H1 Well-performing servers in terms of information retrieval (i.e. servers with high precision) are rewarded by the economic model and end up high in the merged result list. Thereby, making an economic model suitable to select the best performing servers for information retrieval purposes.

As covered in Section 5.2.4, there have been experiments in which the correlation between economic outcomes and the behavior in terms of returning relevant results is strong (i.e., a Kendall's τ value higher than 0.6). We have also seen that within the category of strong correlation, Vickrey auctions with an aggressive or normal strategy comprises the majority of the runs that we conducted. Hence, we can conclude that a distributed information retrieval system with Vickrey auction as an economic model and with an aggressive or normal strategy does reward the right behavior.

H2 Merging the results from participating servers based on the economic value of their results enables efficient results merging.

We executed a separate experiment, where we did not merge the results based on the economic model. From this experiment we learned that only Vickrey auctions, with an aggressive strategy and 250 or 500 servers have a significant increase in precision due to results merging based on the economic model. All other runs have no significant increase. This shows that results merging based on an economic value is an effective merging algorithm in only a few cases. Therefore we can conclude that this hypothesis is only partly true.

H3 Selecting the servers based on economic values enables efficient resource selection.

The same conclusion as with the previous hypothesis holds for whether or not an economic model is more efficient. We have shown that it is effective and efficient by building two systems that perform well in terms of precision, as compared to a centralized search engine. However, we cannot conclude that it is more efficient compared to existing solutions.

H4 Auction models are most suitable for distributed information retrieval contexts if there is shared knowledge about the domain between all servers and the broker.

In our simulation we had agreement on the domain, as both the broker as well as the servers used the same categories. We found out that the Vickrey auction performed better in terms of precision and in terms of relation between economic outcomes and the information retrieval. Therefore, we can conclude that this hypothesis is true. Furthermore, the Vickrey auction performs best with the mixed distribution scenario, which resemble the Internet and deep web situation closest of all distribution scenarios.

hypothesis	true/false
H1	partly true
H2	partly true
H3	partly true
H4	true
H5	partly true
H6	true
H7	partly true

Table 7.1: conclusions about our hypotheses

H5 Bond models are most suitable for distributed information retrieval contexts if knowledge about the domain is not shared between servers and the broker.

If there is no shared knowledge on the domain, it is impossible to make a central categorization of a query that is understood by all servers. Auction models require a categorization of the query, as it is not feasible to place bids on queries requiring too much time to execute a query. Hence, auction models can by definition not work in the situation in this hypothesis. As all other possible models have been concluded as not suitable in Chapter 3, this hypothesis is theoretically true having the bond redistribution model as only remaining model. However, we did not test this with our experiments, as we ran the bond redistribution with categories. Hence, we will conclude that this hypothesis is partly true, as we do not know how it behaves when simulated or used in a real-world test.

H6 Search engine users will favor a system built on economic models, compared to a centralized engine.

The conclusions about this hypothesis follow from Chapter 6, where we presented the results of a real-world experiment. Users indeed favored a system built upon economic models, as shown by a higher rated precision of the economic system compared to a central system. Therefore, we conclude that this hypothesis is true. However, this is dependent on the quality of the categorization engine, as we will cover in the discussion section of this chapter.

H7 Administrators will rate a distributed information retrieval system based on economic models as easy to implement and maintain.

As we have presented in Chapter 6, users value the configuration of the economic model in majority as easy or very easy. The specific notion of an economic models, was reported as helpful or very helpful by a majority of users. However, we did not test if these users have an opinion about other distributed information retrieval systems and how the actual implementation of the system itself is valued. Therefore, this hypothesis is partly true.

The findings and conclusions about the hypotheses are summarized in Table 7.1.

7.1.1 Summarizing Conclusions

With the conclusions drawn per hypothesis in the previous section, we are able to draw conclusions per research questions as stated in the introduction and which are directly related to the problem that we are trying to solve. In this section we will answer the research questions as a summary of the conclusions that we have drawn above.

R1 Which economic models are able to contribute to the solution of the two problems with distributed information retrieval?

R2 Which type of economic model(s) yields the best results with regard to the first research question?

As we have shown in Chapter 3 and summarized in Figure 3.2, there are two models that are suitable: Vickrey auction and bond redistribution. The Vickrey auction models yield better results on all aspects, but require categorization. For categorization to work, all servers and the broker need to have the same understanding about the domain (i.e., use the same categorization). If it is not feasible to use a shared categorization, the bond redistribution model is the best model.

R3 Is a distributed information retrieval system based on an economic model feasible to use in a real world scenario?

As reported in Chapter 6, administrators rate the configuration of a search engine based on an economic model on average as easy and the understanding of an economic model as helpful. The real-world experiment further showed that a real-world implementation of our system is highly dependent on the quality of the categorization. With a good (i.e., human) categorization algorithm, the economic distributed information retrieval system outperforms a central engine on the same corpus, but fails to do so with a normal categorization engine. Given that there exists categorization which performs better compared to the Open Directory Project [48] we used, we conclude that a real-world application of our system is feasible.

With regard to the general research problem, we addressed both an economic distributed information retrieval system design and implementation(Chapter 4) and its results (Chapter 5 and 6). We believe that our system performed reasonably as a distributed information retrieval system. Where reasonable performing is defined as performing at least better compared to naive runs in the simulation and being at least equally good as a the centralized search engine from the real-world experiment. As we have seen in the corresponding chapters, both criteria are met and we conclude that a distributed information retrieval system based on an economic model is feasible.

Finally, we have showed that it is possible to make the deep web accessible, using economic models for an information retrieval system because of three reasons. First, system administrators rate the usage and configuration of such a system as easy and helpful, making it plausible that they would want to use such as system to configure their deep web. Second, we showed that a system based on economic models yields good search results, making it also useful for searching deep web applications. Thirdly, servers can make money by disclosing their deep web, making it worth the effort.

7.2 Discussion

The conclusions that we presented in the previous section have some drawbacks, due to either our research scope or insights that we gathered during our research.

Categorization turned out to be a very important factor in the success of economic distributed information retrieval. The Vickrey auction is dependent on a categorization performed by the search engine and understood by the servers as well. In our real-world test we used the categorization engine as offered by the Open Directory Project, which has a misclassification rate of about 75%. Except from an expected small number of errors by the servers, three out of four queries are unanswerable by the system beforehand. This is a big dependence, making the whole system vulnerable to the quality of the categorization by introducing a single point of failure.

We did not know how to perform a simulation with a standardized test collection. The TREC [51] test sets for example, allow researchers to build their own search engine, use the corpus and queries from TREC and calculate the number of relevant documents that are retrieved (TREC supplies whether or not a document is relevant for a query). As we are dependent on categorization we could not use existing and comparable data sets, as there are no sets that have categorized documents, categorized queries and the relevance of each document to each category. This means that we could not compare our system with other systems under the exact same conditions.

In addition to the lack of comparability between standard systems and our system, we have extensively used probabilities at the server. Both the amount of relevant documents, the distribution of documents over categories and the amount of errors of a server are not based on real distributions. We minimized the effect by varying these probabilities in different scenarios (e.g., an error rate of 0%, 5%, 15% or 30%), allowing us to declare that values in a real-world environment would actually lays in between these scenarios. However, the conclusions would be stronger if we would change the probabilities into either a real corpus or probability distributions based on analysis of real data.

One of the advantages of an economic model is that there is the possibility to actually earn money as a broker, analogous to online advertisements. Due to scope and time limitations we could not perform an analysis of the financial effects to a server in a real-world scenario. Although it works from an information retrieval viewpoint, we cannot state that it would also work financially for search engines.

Finally, we did not test a few scenarios that would have increased the insight in the behavior of the models. Our simulation did not allow for running queries without categories. Therefore, we could not test how well a bond redistribution scenario would behave in a situation where the only thing available is queries. We do believe that this would increase complexity for system administrators: it might however yield more precise results.

7.3 Contribution

Our research has added scientific knowledge to the field of information retrieval on multiple topics.

First, we proved that distributed information retrieval based on an economic

model is feasible. Our review of different economic models and different steps to select the suitable modes, introduces an overview of the possibilities and limitation of economic models which has not been done before. The system design that we presented gives an overview on how to start building an economic distributed information retrieval system. Finally, our results showed that our economic models are suitable to use in terms of information retrieval performance which has not been proved before.

Second, we showed that an economic distributed information retrieval system is appreciated by users. We showed that system administrators value the idea of an economic model when configuring servers for distributed information retrieval purposes. Furthermore we showed that real users rate the results of our system as of higher quality compared to a central search engine.

From a more practical viewpoint, we contributed knowledge that might lead to new business models. Companies can make their deep web accessible, and earn money with doing so. Search engines might research new ways of retrieving search results from distributed search engines and earn money by doing so as well.

7.4 Future Research

We will finish our report with suggestion for further research in the field of economic models applied to distributed information retrieval.

We assumed that our models need categories to work properly. For Vickrey auction this is the case, but the bond redistribution might work without categories; in which case servers decide per query what to do. It would be interesting to repeat our experiments with this model, as it eliminates the need to have a categorization engine and hence reduces bias.

A better comparison of an economic model with an existing Distributed Information Retrieval (DIR) solution might strengthen the conclusions that we presented in this report. For example, one could implement a distributed CORI next to the economic distributed information retrieval system. Furthermore, when different types of databases are connected the actual situation of the deep web is more closely met.

Furthermore, we would suggest a more economic and business administration oriented study on new business models that proceed from economic distributed information retrieval. It is very well possible that new constraints erupt from this business branch which influence the development of an economic distributed information retrieval system. A comparison to online advertisement models would make sense in such a study.

Finally, we suggest to start building a test collection that is suitable for research on economic distributed information retrieval systems. This collection should consists of categorized documents, categorized queries and the relevance of a document for a query (category). With such a test collection it is easier to compare different DIR-systems.

Bibliography

[1] R Ahmed and R Boutaba. A survey of distributed search techniques in large scale distributed systems. *Communications Surveys Tutorials, IEEE*, PP(c99):1 –18, May 2010.

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. Addison-Wesley Reading, MA, 1999.

[3] D. Basu. *Economic models: methods, theory and applications*. World Scientific Pub Co Inc, 2009.

[4] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

[5] M. Ben-Ari. *Principles of Concurrent and Distributed Programming (2nd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[6] M.K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1):01–07, 2001.

[7] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and computation: practice and experience*, 14(13-15):1507–1542, 2002.

[8] Jamie Callan. Distributed information retrieval. In W. Bruce Croft, editor, *Advances in Information Retrieval*, volume 7 of *The Information Retrieval Series*, pages 127–150. Springer US, 2000.

[9] Jamie Callan and Margaret Connel. Query-based sampling of text databases. *ACM Trans. Inf. Syst.*, 19:97–130, April 2001.

[10] Ian Campbell. Spam, the repeat offender. Technical report, Nucleus Research, april 2007.

[11] Peter Pin-Shan Chen. The entit-relationship model towards a unified view of data. *ACM Trans. Database Syst.*, 1:9–36, March 1976.

[12] Eustache Diemert and Gilles Vandelle. Unsupervised query categorization using automatically-built concept graphs. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 461–470, New York, NY, USA, 2009. ACM.

[13] Dominic DiPalantino and Milan Vojnovic. Crowdsourcing and all-pay auctions. In *Proceedings of the tenth ACM conference on Electronic commerce*, EC '09, pages 119–128, New York, NY, USA, 2009. ACM.

[14] K. Dopfer, J. Foster, and J. Potts. Micro-meso-macro. *Journal of Evolutionary Economics*, 14(3):263–279, 2004.

[15] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *The American Economic Review*, 97:242–259(18), March 2007.

[16] Mark B. Garman. Market microstructure. *Journal of Financial Economics*, 3(3):257 – 275, 1976.

[17] A. Goker and J. Davies. *Information Retrieval: Searching in the 21st Century*. Wiley, 2009.

[18] David R. Henderson. Vilfredo pareto: The concise encyclopedia of economics.
`http://www.econlib.org/library/Enc/bios/Pareto.html`, December 2007.

[19] D. Hiemstra. Distributed information retrieval using keyword auctions. Technical Report TR-CTIT-08-55, Centre for Telematics and Information Technology University of Twente, Enschede, September 2008.

[20] H.M. Hochman and J.D. Rodgers. Redistribution and the Pareto criterion. *The American Economic Review*, pages 752–757, 1974.

[21] W. Jager, MA Janssen, HJM De Vries, J. De Greef, and CAJ Vlek. Behaviour in commons dilemmas: Homo economicus and Homo psychologicus in an ecological-economic model. *Ecological economics*, 35(3):357–379, 2000.

[22] Richi Jennings. Cost of spam is flattening our 2009 predictions.
`http://bit.ly/cDLh26`, January 2009.

[23] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 154–161, New York, NY, USA, 2005. ACM.

[24] Jaana Kekäläinen and Kalervo Järvelin. The impact of query structure and query expansion on retrieval performance. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 130–137, New York, NY, USA, 1998. ACM.

[25] N. Khanna. Optimal bidding for tender offers. *Journal of Financial Research*, 20:323–342, 1997.

[26] V. Krishna. *Auction theory*. Academic press, 2009.

[27] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228, 2000. 10.1023/A:1007608224229.

[28] Thede Loder, Marshall Van Alstyne, Rick Wash, and Mark Benerorfe. The spam and attention bond mechanism faq. Technical report, University of Michigan, 2004.

[29] Thede Loder, Marshall Van Alstyne, and Rick Wash. An economic answer to unsolicited communication. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 40–50, New York, NY, USA, 2004. ACM.

[30] Theodore C. Loder, Marshall Van Alstyne, and Richard L. Wash. Economic solution to the spam problem. US Patent Number 20090319290, December 2009.

[31] Yiyao Lu, Weiyi Meng, Liangcai Shu, Clement Yu, and King lup Liu. Evaluation of result merging strategies for metasearch engines. In *WISE Conference*, pages 53–66, 2005.

[32] D. G. Luenberger. New optimality principles for economic efficiency and equilibrium. *Journal of Optimization Theory and Applications*, 75:221–264, 1992. 10.1007/BF00941466.

[33] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July 2008.

[34] R. Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25(2):699–738, 1987.

[35] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 35–46, New York, NY, USA, 2000. ACM.

[36] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM.

[37] Hsiao-Tieh Pu, Shui-Lung Chuang, and Chyan Yang. Subject categorization of query terms for exploring web users' search interests. *J. Am. Soc. Inf. Sci. Technol.*, 53(8):617–630, 2002.

[38] Yves Rasolofo, Faïza Abbaci, and Jacques Savoy. Approaches to collection selection and results merging for distributed information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 191–198, New York, NY, USA, 2001. ACM.

[39] John G. Riley and William F. Samuelson. Optimal auctions. *The American Economic Review*, 71(3):pp. 381–392, 1981.

[40] Tracy Ross. The carbon footprint of email spam report. Technical report, McAfee, april 2009.

[41] Yuko Sakurai, Makoto Yokoo, and Shigeo Matsubara. A limitation of the generalized vickrey auction in electronic commerce: robustness against false-name bids. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, AAAI '99/IAAI '99, pages 86–92, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[42] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 298–305, New York, NY, USA, 2003. ACM.

[43] P. Terna. Simulation Tools for Social Scientists: Building Agent Based Models with SWARM. *Journal of Artificial Societies and Social Simulation*, 1, 1998.

[44] L. Tesfatsion. Agent-based computational economics: A constructive approach to economic theory. *Handbook of computational economics*, 2:831–880, 2006.

[45] R. J. Wieringa. *Design Methods for Reactive Systems: Yourdon, Statemate, and the UML*. Morgan Kaufmann, January 2003.

[46] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The michigan internet auctionbot: a configurable auction server for human and software agents. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 301–308, New York, NY, USA, 1998. ACM.

[47] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 254–261, New York, NY, USA, 1999. ACM.

[48] Odp - open directory project.
`http://www.dmoz.org`.

[49] OpenSearch 1.1 Specification (draft 4).
`http://www.opensearch.org/Specifications/OpenSearch/1.1`.

[50] Php: Hypertext preprocessor.
`http://www.php.net`.

[51] Text REtrieval Conference (TREC) Data.
`http://trec.nist.gov/data.html`.

[52] Uppaal.
`http://www.uppaal.org`.

[53] Wiki communities for everyone - wikia.com.
http://www.wikia.com.

[54] WordNet – A lexical database for English.
http://wordnet.princeton.edu.

Appendix A

Open Directory Project Categories

- Arts
- Business
- Computers
- Games
- Health
- Home
- Kids and Teens
- News
- Recreation
- Reference
- Regional
- Science
- Shopping
- Society
- Sports

Glossary

agent-based computational economics is a field of study in economics where economies are modeled as decentralized systems with agents that make their own decisions leading to global system properties.

bond is a sum of money which one party sets aside by a third party, before a transaction occurs, as a sign of good faith.

bond redistribution is a model in which a server places a bond in order to guarantee the quality of a result set. The bond is seized from bad servers and redistributed among good servers.

broker is the central entity in a Distributed Information Retrieval architecture, receiving queries from users, distributing these among servers and merging the results returned by servers.

domain is a part of the world with its own taxonomy, messages, real-world entities, jargon, norms and standards.

OpenSearch is a standardized XML format for search engines allowing to share search results with each other.

precision is the fraction of retrieved relevant information to all retrieved information.

recall is the fraction of retrieved relevant information to all relevant information.

resource description is a formal description of data a server hosts.

resource selection is the process of selecting the needed servers to answer a given query from a set of servers.

result set is a fixed number of results for a given query.

results merging is the process of integrating the result lists returned by each server into a single coherent list.

server is an entity in a Distributed Information Retrieval architecture, hosting a collection of data that is searchable.

Vickrey auction is a second-price sealed bid auction, where the auction winner pays the second-highest bid instead of its own bid.

Acronyms

ABM Attention Bond Mechanism.

DIR Distributed Information Retrieval.

ODP Open Directory Project.

URI Uniform Resource Identifier.

XML Extensible Markup Language.