A KNOWLEDGE BASE ON NP-COMPLETE DECISION PROBLEMS AND ITS APPLICATION IN BIBLIOGRAPHIC SEARCH

Harald Sack, Uwe Krüger*, Michael Dom
Institut für Informatik
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2
D-07743 Jena
{sack, ukrueger, dom}@informatik.uni-jena.de

Abstract: NP-complete decision problems are one of the most important subjects in computer science. We have developed a knowledge base of NP-complete decision problems that reflects the conceptual knowledge and the relationships of more than 350 decision problems including additional information resources such as bibliographic references or author information. The knowledge base is applied by the web based bibliographic search tool *NPBibSearch*. By providing an intuitive graphical user interface for navigation within the domain of NP-complete decision problems and by integrating full text search capabilities with conceptual knowledge, *NPBibSearch* offers the user an ontology augmented bibliographic search.

1 Introduction

NP-complete decision problems are one of the most important topics in computer science and mathematics. In 2000 the Clay Mathematical Institute of Cambridge, Massachusetts (CMI) presented the so called "Millennium Problems" focusing on important classic questions that have resisted solution over the years. The solution of the question whether NP-complete problems can be decided efficiently or not is subsumed to the equivalence of the two complexity classes P and NP [GJ79]. In computational complexity theory P summarizes all decision problems that can be computed efficiently, i.e. they can be solved deterministically in polynomial time w.r.t. the length of their input. In contrast, NP describes the set of decision problems solvable in polynomial time on a *nondeterministic* Turing machine, among them the class of NP-complete problems, which are considered to be very hard to solve and thus, are still subject of ongoing research [Joh05]. To the solution of this famous "P=NP"-question the board of directors of CMI has designated a $1 million prize [Cla].

Our goal was to support ongoing research on NP-complete decision problems by creating a knowledge base of this important class of problems and to make it available to every researcher interested in the subject. The knowledge base has been populated with more than 350 decision problems including information about their relationships with each other and it is continuously growing.

*Supported by the Deutsche Forschungsgemeinschaft, Research group PEAL (Parameterized complexity and exact algorithms), NI 369/1.

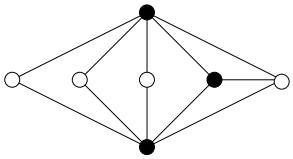

Figure 1: Example for VERTEX COVER. The black vertices form a vertex cover of size three: Every edge has at least one black endpoint.

Among the billions of WWW documents, there is also an increasing number of scientific papers, often organised in electronic journals or bibliographic databases. The *Electronic Colloquium on Computational Complexity* (ECCC) [ECC] is an electronic journal dedicated to the field of computational complexity. It provides a bibliographic database including more than 1.000 ECCC papers related to NP-complete decision problems (for a detailed description see [BMS98]). We improved the regular, keyword-based bibliographic search of ECCC by adding a new ontology enhanced search facility (*NPBibSearch*) based on our ontology representation of the knowledge base of NP-complete decision problems. By providing a graphical user interface *NPBibSearch* enables the user to navigate within the search domain, facilitates the resolution of ambiguities among query keywords, and allows cross-referencing between related search results. Additionally, *NPBibSearch* connects regular keyword-based full-text retrieval of ECCC documents via Google and Yahoo Web APIs Services [Goo, Yah] to information provided by the ECCC database for realising improved search results. *NPBibSearch* enables the user an intuitive navigation within the domain of NP-complete decision problems. It points out how particular decision problems are related to other decision problems, adds supplementary information to the decision problem being in current focus, and is able to search for references of that particular decision problems within scientific papers. In this paper we describe how to design and to implement a knowledge base of NP-complete decision problems and show, how to apply this knowledge base to enhance bibliographic search.

To this end, the paper is structured as follows: Section 2 starts with a concise introduction in NP-complete problems. In Section 3, the design and the implementation of the knowledge base of NP-complete problems is discussed, while Section 4 presents *NPBibSearch* – an application of this knowledge base in bibliographic search. Section 5 concludes the paper and gives an outlook on ongoing and future work.

2 NP-complete Decision Problems and their Relevance

The computational problems we are dealing with are *decision problems*, i.e., problems where the task is to answer a question with *yes* or *no*. For an example, we consider the problem VERTEX COVER, which is defined as follows:

> **Input:** An undirected graph $G = (V, E)$, a positive integer k.
> **Question:** Is there a subset $C \subseteq V$ of at most k vertices such that each edge in E has at least one endpoint in C?

The set C is called a vertex cover for the graph G (see Fig. 1). A problem instance — i.e., a concrete input[1] — is called a yes-*instance* if its solution is *yes* and a no-*instance* otherwise.

[1] In complexity theory the term *problem instance* denotes a concrete input for a problem, e.g. a concrete pair (G, k) in the case of VERTEX COVER. Don't mistake it with the *instance of a class* in the object oriented context as it occurs in Sect. 3.

Problems can be classified by considering the running times that are needed to solve them. To this end, an abstract computer model is used which is called *(deterministic) Turing machine* and whose computational power is identical to that of most relevant real world computer programs.

The running time needed to solve a problem X is measured in terms of how many steps a Turing machine has to perform to solve the problem. It is given as a function $t_X(n)$ on the size n of the input (e.g., on the number of vertices and edges in the case of VERTEX COVER). More exactly, the function $t_X(n)$ always expresses the so-called worst-case running time, i.e., $t_X(n)$ is the maximum running time needed for solving the problem X, taken over all problem instances of size n.

The class P contains all decision problems X that can be solved within a running time $t_X(n)$ where t_X is a polynomial. For most practical applications such a polynomial running time means that the problem is solvable within a reasonable amount of time.

For defining the problem class NP a modified computer model called *nondeterministic Turing machine* is introduced: In contrast to deterministic Turing machines, a nondeterministic Turing machine has the freedom to *guess* in each step. By definition, it solves a problem if for every *no*-instance it always answers correctly with *no*, and if for every *yes*-instance it answers correctly with *yes* if it has guessed right in every step. The class NP is defined as the class of all decision problems X that can be solved nondeterministically in time $t_X(n)$, where t_X is a polynomial.

Due to the power of guessing and to the definition of "solving a problem", the class NP contains an enormous number of problems, many of them of substantial practical relevance. In particular, all problems that belong to P are also contained in NP (which directly follows from the definitions). However, there are a lot of important problems in NP that seem not to be solvable deterministically in polynomial time: Replacing the powerful guessing by deterministic steps often results in exponential running time (the so-called "combinatorial explosion"). In order to combine such "difficult" problems into a class of their own, the concept of the *reduction* is introduced: A problem X is *(polynomial-time) reducible* to a problem Y (denoted with $X \leq Y$) if there is a function f that maps every problem instance x of X to a problem instance $y = f(x)$ of Y such that y is a *yes*-instance of Y if and only if x is a *yes*-instance of X. Moreover, there must be a polynomial t_f such that the time for computing $f(x)$ does not exceed $t_f(|x|)$. Intuitively speaking, the problem Y is "as least as hard" as the problem X, because a problem instance x of X can be solved using an algorithm for problem Y: First, compute the problem instance $y = f(x)$ of Y and then solve y using the algorithm for Y.

The concept of reducing a problem can be illustrated by the very simple reduction of the problem INDEPENDENT SET to VERTEX COVER: The problem INDEPENDENT SET asks, given a graph $G = (V, E)$ and a positive integer k, if G has an independent set $C \subseteq V$ of at least k vertices. (An independent set is a set of vertices that are pairwise not connected by edges.) INDEPENDENT SET can be reduced to VERTEX COVER by mapping each problem instance (G, k) of INDEPENDENT SET to a problem instance $(G, |V|-k)$ of VERTEX COVER. If $(G, |V|-k)$ is a *yes*-instance of VERTEX COVER, then G has a vertex cover C of size at most $|V| - k$ and, therefore, (G, k) is a *yes*-instance of INDEPENDENT SET: The vertices not belonging to C are not connected by edges and form an independent set of size at least k (to see this, consider the white vertices in Fig 1).

The definition of the polynomial-time reduction directly implies that if $X \leq Y$ and $Y \in$ P, then also $X \in$ P. The other way round, if $X \leq Y$ and if X is one of the "difficult" problems, it is unlikely that Y can be solved in polynomial time (because otherwise the reduction would constitute a polynomial-time algorithm for X). A problem Y is called NP-*hard*, if every problem $X \in$ NP can be reduced to Y. If, in addition, $Y \in$ NP, the problem Y is called NP-*complete* (see also [Pap94]).

Thus, NP-complete problems are the "most difficult" problems of the class NP, and no algorithms are known that solve these problems efficiently.

There are thousands of NP-complete problems, and they arise in all areas of life [Pap97]. The problem VERTEX COVER for example is NP-complete, and we will only mention two applications of VERTEX COVER:

1. Finding the cheapest placement of monitoring devices in a communication network: If each device can monitor all links incident to the site where it is located, the problem of monitoring the whole network with a given number of devices is equivalent to VERTEX COVER.
2. Resolving conflicts: If in a series of experiments there are pairwise conflicting results (i.e., at least one must be faulty), one can try to resolve all conflicts by deleting a small number of results to make the remaining ones conflict-free.

Due to the universality of NP-complete problems and to their inherent difficulty it is of great interest to gather knowledge about these problems and to understand the relations (such as reducibility) between them. For this reason, we have developed a knowledge base representing conceptual knowledge about NP-complete decision problems and populated it with the most prevalent decision problems.

3 A Knowledge Base on NP-complete Decision Problems

The knowledge base on NP-complete decision problems was implemented with the *Web Ontology Language* (OWL) [MvH04]. We restricted to OWL LT, because it is the most simple OWL language version that enables the definition of disjoint classes and 0/1 cardinalities, which is required for our knowledge base. For modeling and populating the knowledge base, we used the Protégé [NSD+01] OWL Plugin. Below, an outline of the knowledge base of NP-complete decision problems is given by a description of the class hierarchy and the class relationships followed by an example of a class instance.

3.1 Classes and Class Properties

Following the classification of Garey and Johnson [GJ79], we have structured the domain of decision problems into different subsets such as graph problems, logic problems, set problems, sequence and scheduling problems, etc. (see Fig. 2 for a simplified outline of the classes and class relationships of the ontology of NP-complete decision problems). The basic class of the ontology is the *decision problem*, which is a special case of the class *problem* restricted to those problems that can only be answered with *yes* or *no*. Each problem is characterised by its *input* and its *output*. The type of the problem's input depends on the problem subset to which it belongs to, e.g. for a graph problem there is always a graph among its input. For decision problems, the output can be restricted to a boolean variable representing *yes* or *no*. For each decision problem the following information is provided:

- According to the resources (running time or space) required to decide a decision problem it is member of a particular *complexity class* (e.g. P, NP, NP-complete, PSPACE, etc.). For modeling the complexity classes we provided miscellaneous complexity measures (e.g. *logarithmic space*, *polynomial time*, etc.) and complexity bounds.
- The *algorithm* class describes algorithms for deciding a particular decision problem. Reductions between decision problems are considered to be a special case

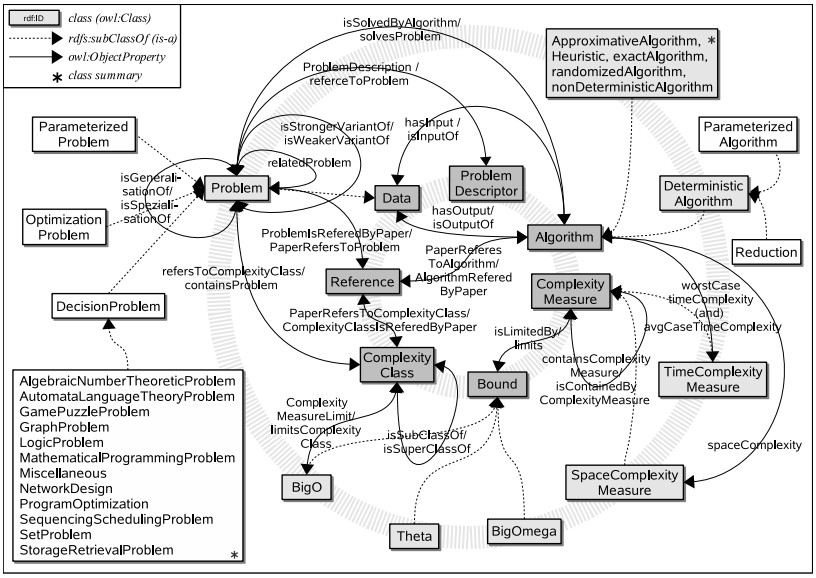

Figure 2: Simplified Ontology for NP-complete Problems.

of algorithms. We distinguish deterministic, nondeterministic, and randomized algorithms.

- Each decision problem is further characterised with a *problem descriptor*. The problem descriptor comprises detailed information about the decision problem and facilitates disambiguation by providing synonyms, alternative spellings, and possible abbreviations for the name of the decision problem.
- Wherever possible, a primary bibliographic reference as well for decision problems as for algorithms and reductions is provided. Bibliographic references are described by suitable Dublin Core metadata elements [Dub].

For decision problems there are several possible relationships to consider:

- By changing a single parameter of a decision problem, it may change its membership to some complexity class. E.g. the 3-SAT decision problem testing the satisfiability of a Boolean formula in conjunctive normal form with at most three variables per clause is an NP-complete decision problem, while 2-SAT, where only clauses with at most two variables are considered, belongs to P. In this way, decision problems can be related to *stronger* or *weaker* decision problems of the same kind that belong to different complexity classes.
- Introducing a new parameter or fixing a given parameter of a decision problem can result in a special case of that particular decision problem. Thus, decision problems can be related to each other by *generalisation* or *specialisation*, e.g. SAT is a generalisation of 3-SAT or 2-SAT, while 2-SAT is a special case of SAT.
- NP-complete decision problems are also related to each other by *reductions*, e.g. SAT can be reduced to 3-SAT in linear running time simply by using auxiliary variables.

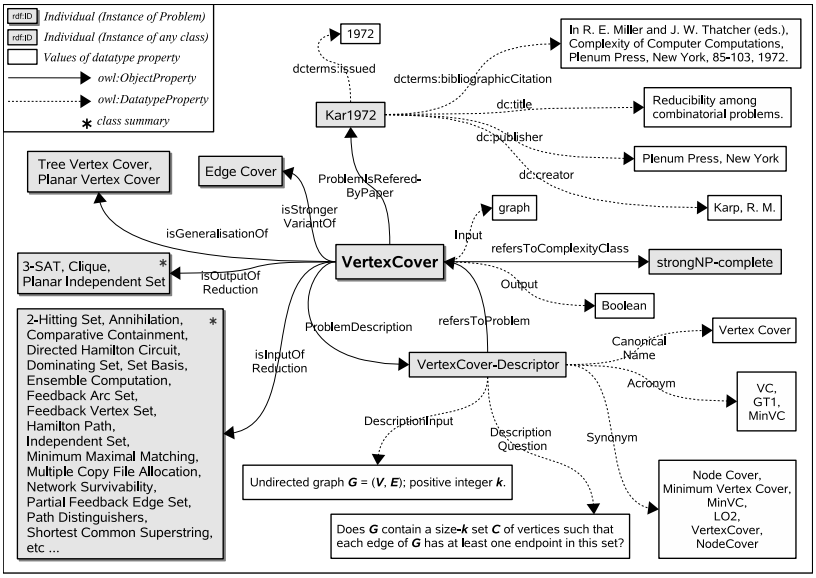

Figure 3: The VERTEX COVER Decision Problem and its Relationships.

3.2 Instances

To further illustrate the structure of the knowledge base we examine the VERTEX COVER decision problem (see Fig. 3 for a graphical outline of the VERTEX COVER problem and its related information):

- As being a graph problem, among the input of VERTEX COVER there must be a graph. It is also a decision problem, therefore its output is boolean.
- VERTEX COVER is related to the *weaker* EDGE COVER problem.
- Special cases of VERTEX COVER are PLANAR VERTEX COVER and TREE VERTEX COVER, where the input graph is restricted only to planar graphs and trees.
- There are possible reductions from 3-SAT, CLIQUE, and PLANAR INDEPENDENT SET to VERTEX COVER. Additionally, there are also reductions from VERTEX COVER to several other decision problems such as e.g. 2-HITTING SET, ANNIHILATION, or DOMINATING SET.
- VERTEX COVER is referred to by several synonyms such as e.g. NODE COVER and MINIMUM VERTEX COVER. Sometimes the abbreviation VC or the identifier LO2 introduced by Garey and Johnson is used.
- The primary bibliographic reference for VERTEX COVER is [Kar72].

Up to now, we have included the entire collection of the NP-complete decision problems accumulated in [GJ79] together with many other related decision problems. The knowledge base comprises more than 350 decision problems and is constantly growing. Furthermore, we added more than 160 primary bibliographic references and included 323 particular reductions between the decision problems. The knowledge base on NP-complete decision problems is available at [SVNK].

4 Ontology Enhanced Bibliographic Search - NPBibSearch

The knowledge base of NP-complete decision problems is applied to a web based bibliographic search tool — *NPBibSearch* — with its focus on scientific papers related to computational complexity. Therefore, we chose the repository of the *Electronic Colloquium on Computational Complexity* (ECCC) [ECC] as our primary bibliographic database. ECCC provides a large database of more than 1.000 scientific papers related to computational complexity. *NPBibSearch* extends a conventional keyword-based search request on ECCC by combining additional conceptual knowledge of NP-complete decision problems with a restricted search on a generic search engine such as Google or Yahoo. Moreover, *NPBibSearch* provides query string evaluation, query string expansion, and navigation aid within the domain of NP-complete decision problems. Further information about the query subject and the referenced papers is supplied by external resources, such as the *Digital Bibliography and Library Project* (DBLP) [Ley95], the *Scientific Literature Digital Library* (CiteSeer.IST) [GBL98], and HomePageSearch [HM01].

4.1 NPBibSearch Architecture

NPBibSearch is a web based server application and, therefore, its implementation is based on the Java Servlet Technology. Its design follows the *model-view-controller* (MVC) paradigm [Ree79, BMR+96]: The control logic is processed in a central *controller* servlet component, which handles the request parameters and forwards the control to the respective Java Server Pages (JSP). The data (*model*) that is bound to the user session is implemented as Java Beans. The *view* component is represented by JSPs that provide a suitable user interface. The OWL encoded ontology representing the knowledge base on NP-complete decision problems is processed using the Jena Semantic Web Framework [McB02]. The search results provided by NPBibSearch are achieved by utilising Google's and Yahoo's web services interface [Goo, Yah] via SOAP messages over HTTP. See Fig. 4 for an overview of the NPBibSearch architecture. NPBibSearch can be accessed via [SK].

4.2 NPBibSearch Use Case

In the following we give a short overview of the functions of *NPBibSearch* by describing a brief use case. Let's assume the user starts a bibliographic search about the VERTEX COVER problem by typing the query string "*vertex*".

Query string evaluation. The string *vertex* occurs to be among the synonyms of VERTEX COVER, but it is also found for TREE VERTEX COVER, FEEDBACK VERTEX SET, and other decision problems. A list of possible alternatives is displayed to the user, who has to decide which of the referred decision problems to select. After deciding on VERTEX COVER, a keyword-based full text search in the ECCC database is performed. Additional information about the chosen decision problem, as e.g. problem description, problem input, and primal bibliographic reference, is displayed above the search results.

Searching. ECCC only allows full text search in titles, keywords, or abstracts, but not inside the text body. For ECCC resources being also available as static HTML files, they are on-hand for web robots of general search engines. Therefore, we are able to conduct a keyword-based full text search via Google or Yahoo restricted only to the ECCC web site with the search string *Vertex Cover*. It is possible to switch between the original Google or Yahoo results, where only the paper's title and a short text snippet is displayed, and the ECCC results that provide additional useful information. For each

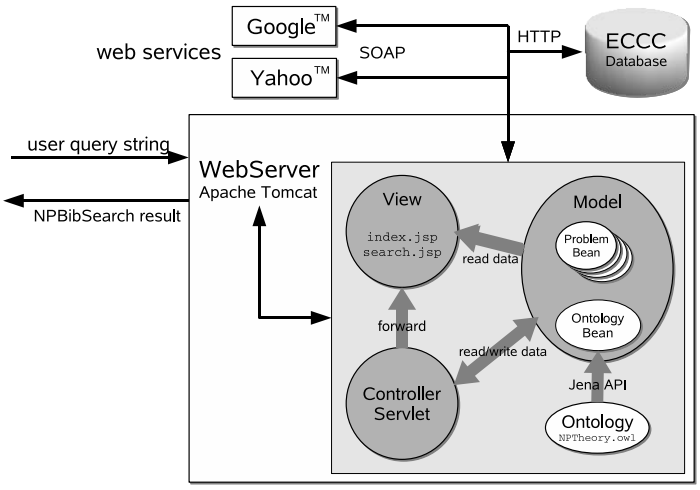

Figure 4: Schematic Overview of the NPBibSearch Architecture.

search result the following information is provided:

- the authors of the paper linked to the author's homepage or linked to a search request for the author's homepage at HomePageSearch
- the abstract of the paper
- the list of keywords provided by the authors
- highlighted occurrences of the query string among the results
- a link to the ECCC cover page related to that paper.

Navigation. The ontology of NP-complete decision problems can be utilised for domain navigation. Cross references between individual decision problems represented within the ontology are displayed in an extra navigation interface on top right of the search results (see Fig. 5). According to the three defined relationships among decision problems (stronger/weaker, generalization/specialization, reduced to/from) we charted three axes for navigating the ontology domain. With the help of this very simple navigation tool the user can choose to switch to a related decision problem that is transferred to the query string and a new search can be performed.

Query string refinement. To refine or to broaden the current search, the user has the possibility to add additional terms to the search string, either by typing a supplementary query string, or by choosing one or more entries from the list of synonyms and alternative names (displayed below the query string on top of the search results). The result page of a search on VERTEX COVER is shown in Fig. 5.

Additionally, the user can extend his search to other databases and services. For each document among the search results a link to CiteSeer.IST and to DBLP is provided. Thus, the user is able to gather more information about the specific document and where it is referenced.

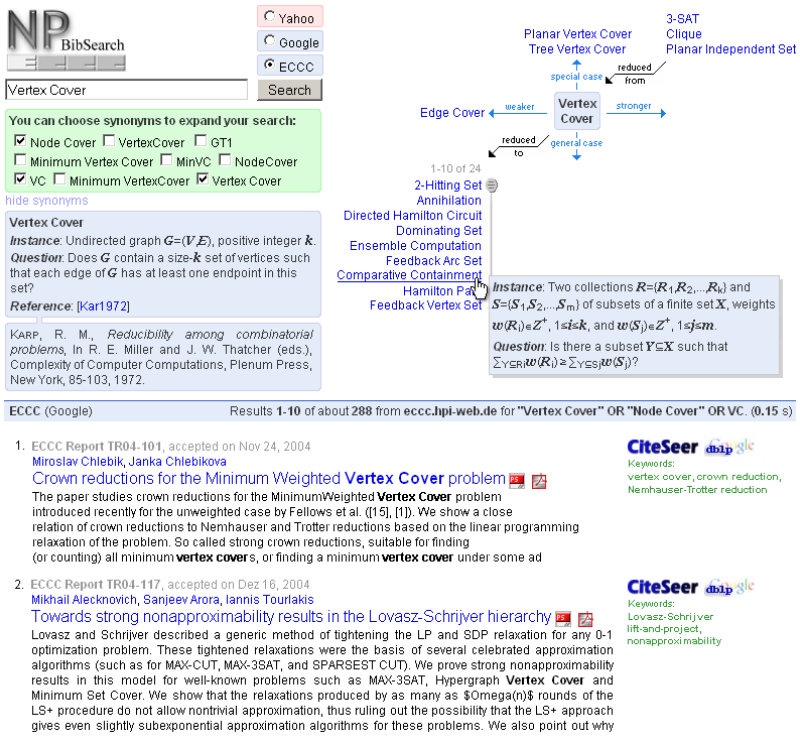

Figure 5: Results Provided by NPBibSearch for the VERTEX COVER Decision Problem.

5 Outlook and Conclusion

NP-complete decision problems are one of the most important subjects in computer science. We have developed a knowledge base on NP-complete decision problems that addresses more than 350 decision problems. It reflects relationships between particular decision problems and provides additional information resources such as bibliographic references or author information. *NPBibSearch* applies this knowledge base to augment web based bibliographic search. By providing a suitable user interface for navigation within the field of NP-complete problems and by integrating full text search capabilities, *NPBibSearch* enables the user to perform a web based search within the ECCC bibliographic database.

Currently, the search capabilities of the *NPBibSearch* prototype are extended in two ways: Besides restricting the search to a specific bibliographic database, other web based bibliographic databases are to be integrated. As long as they provide a suitable interface for search requests and deliver the search results in a predefined and structured way, as e.g. based on web services, a seamless integration will be possible.

For all other search request on web based bibliographic databases a simple HTTP request can be performed, which includes the search parameters within its body or encoded inside the requested URL. In this case, the decoding of the delivered search results becomes difficult, because most times the database response is delivered as an unstructured HTML document. The actual search results have to be retrieved from

this HTML document based on its layout with the help of screen scraping methods. But, because the layout of an HTML document is subject to frequent changes, the configuration of the screen scraping has to be adapted every time.

Another way to extend the search is to include different ontologies representing knowledge about other subjects of theoretical computer science, computer science in general, or even about any other subject. By integrating those ontologies within a suitable hierarchy, the graphical navigation tool of *NPBibSearch* can be upgraded to represent a topic map like user interface that enables navigation within the represented knowledge domains. Thus, also a general web search engine can be augmented by the integration of suitable domain ontologies for navigating the represented search domain.

References

[BMR+96] F. Buschmann, R. Meunier, H. Rohnert, P.Sommerlad, and M. Stal. *A system of patterns - pattern oriented software architecture*. Wiley, 1996.

[BMS98] J. Bern, Ch. Meinel, and H. Sack. Electronic Colloquia: Idea and Practice. In *Proceedings of the ACM SIGDOC 1998 Conference, Quebec City, Canada*, 1998.

[Cla] Clay Mathematics Institute. The Millennium Problems, `http://www.claymath.org/millennium/`.

[Dub] DublinCore Initiative. Dublin Core Metadata Element Set, Version 1.1: Reference Description `http://dublincore.org/documents/dces/`.

[ECC] The Electronic Colloquium on Computational Complexity (ECCC), `http://eccc.hpi-web.de/eccc/`.

[GBL98] C. Lee Giles, K. D. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. *3rd ACM Conf. on Digital Libraries*, pages 89–98, 1998.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA, 1979.

[Goo] Google Web APIs Reference, `http://www.google.com/apis/reference.html`.

[HM01] Gerd Hoff and Martin Mundhenk. Finding scientific papers with homepagesearch and MOPS. In *SIGDOC*, pages 201–207, 2001.

[Joh05] David S. Johnson. The NP-completeness column. *ACM Transactions on Algorithms*, 1(1):160–176, 2005.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Ley95] Michael Ley. DB&LP: A WWW Bibliography on Databases and Logic Programming. *Compulog Newsletter*, 1995.

[McB02] B. McBride. Jena: A Semantic Web Toolkit. In *IEEE Internet Computing*, volume 6, pages 55–59, 2002.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language: Overview, W3C Recommendation, 10 February 2004.

[NSD+01] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating Semantic Web contents with Protege-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[Pap94] Christos M. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pap97] Christos H. Papadimitriou. NP-Completeness: A Retrospective. In *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming (ICALP 1997)*, volume 1256 of *Lecture Notes in Computer Science*, pages 2–6. Springer, 1997.

[Ree79] Trygve M. H. Reenskaug. Models - Views - Controllers. Technical note, Xerox PARC, `http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf`, 1979.

[SK] H. Sack and U. Krüger. NPBibSearch - an Ontology Augmented Bibliographic Search Engine `http://ipc755.inf-nf.uni-jena.de:8084/NPBibSearch/`.

[SVNK] H. Sack, J. Vogel, R. Niedermeier, and U. Krüger. Ontology for NP-complete Problems, `http://www.informatik.uni-jena.de/~sack/Material/NPtheory.owl`.

[Yah] Yahoo Web Search APIs Reference, `http://developer.yahoo.com/search/web/`.