Software Citations, Information Systems, and Beyond

Michael Kohlhase[1], Wolfram Sperber[2]

[1] Friedrich-Alexander Universität Erlangen-Nürnberg, Martenstr. 3, 91058 Erlangen, Germany
[2] FIZ Karlsruhe/zbMATH, Franklinstr. 11, 10587 Berlin, Germany

Abstract. Even though software plays an ever-increasing role in today's research and engineering processes, the scholarly publication process has not quite caught up with this. In particular, referencing and citing software remains problematic. Citations for publications are well-standardized but don't immediately apply to software as, for instance, *a*) software information is extremely heterogeneous, *b*) software code is not persistent, and *c*) the level of software information is often too coarse-granular.
Current initiatives try to solve *a*) by postulating "landing pages" for software that aggregate standardized meta-data and can be used as targets for citations and *b*) by version-specific sub-landing pages. However no information services that provide such landing pages currently exist, making these proposals ineffective in practice.
After an overview of the state-of-the-art, we propose to use swMATH's information system for mathematical software as a source of landing pages, show an approach for version-specific sub-pages, and discuss approaches to cope with problem *c*) (granularity).

1 Introduction

Today's scientific digital libraries contain publications and (increasingly) the accompanying research data to make results reproducible by the scientific community. As much of the data and the results are created using scientific software, that is also increasingly considered as "research data" worth conserving and referencing itself. Therefore, detailed and persistent information about software is needed in scholarly processes and the archival publication record. For the first, we need software information services, for the latter, we need good practices for software citations. Citations for publications are well-standardized but don't immediately apply to software as, e.g.,

a) *Software information is extremely heterogeneous*: it is distributed over web pages, manuals, scientific articles, generated system documentations, developer mailing lists, etc. This creates problems for software citations as it is unclear – i.e. there is no accepted best practice – which of the various information resources to cite as a substitute for the software itself.

b) *Software systems and code are not persistent*: the software itself and the information pertaining to it are usually released in discrete versions or are continuously updated in revision control systems. Some citation-relevant information is version-specific, other pertains to the software per se and is therefore static. Again, there are no established standards for identifying software versions or revisions and how to integrate this information into citations. Moreover, new versions often supersede the old ones, so accessing old versions or revisions can be difficult.

c) *The level of software information is often too coarse-granular*: we need to know which exact series of instructions or API functions were involved in producing a certain result to be reproducible.

These problems are generally recognized by the scholarly community, and there are a variety of initiatives that try to solve them. By and large, these initiatives have identified the problems and have proposed principles for dealing with them. For instance by stipulating "landing pages" for software that aggregate standardized meta-data and can be used as the targets for citations. While proposals abound of what (meta)-data such landing pages should contain, no public resources that provide such landing pages currently exist, making these proposals ineffective in practice.

In this paper we analyze the state of the art of and proposals for software citations and information systems (for mathematical software), derive requirements, and show how they can be met in existing systems, using our own swMATH system [swMath] which automatically aggregates information on mathematical software systems from the scientific literature and web information sources and the OpenDreamKit API theories [CICM1616] as examples.

In Section 2 the state of the art in software citations is previewed. In Section 3 we present the swMATH system and we show how it can be used to provide software object identifiers (SOI) – akin to document object identifiers (DOI) – and aggregated landing pages above; this addresses *a*). Section 3.2 presents an extension of swMATH to include software versions (micro-archives of version-specific software information) to address *b*). The pages in the micro-archives can be used to provide versioned SOIs and landing pages. Finally, in Section 4 we sketch how we can extend the ideas before to finer-grained citations of API functions to solve *c)*. Section 5 concludes the paper.

2 Approaches and Principles for Referencing and Citing Software

We will now review the current discussion of software information and citations. We will pay particular attention to math software, persistence/versions *b*), and granularity *c*) issues.

2.1 Mathematical Software Information on the Web

The landscape of software and software information in the Web is heterogeneous. Information on software exists in the following kinds of venues.

Websites The analogon of publications for software are their websites. For mathematical software we estimate that 2/3 of the software products run their own websites, see [Chr+17]. The websites contain the software code (if it is open), manuals and documentations, APIs, information of legal rights, programming languages used, and context information, e.g., programming languages, software dependencies, test data, etc. Software websites are usually updated for new versions, and thus contain a mixture of general information about the software product and the current version (code, manuals and documentations, APIs, etc.). Distinguisthing both categories is not always possible, but relevant for software citations. A further difficulty for processing information of the websites is that the websites are designed individually and have no common structure or meta-data. The visibility in the Web for small and non-prominent software packages is a problem: finding new or specialized software is difficult for potential users.

Repositories have been established as community-driven focal points for accessing and archiving of software products. Often they collect and contain software products in a certain language or environment. Repositories trace back to activities of communities and are more prominent than websites of a single software. Moreover, they define rules and (proprietary) standards for the software listed in the repositories. Often they provide also a version management, e.g., CRAN [CRAN] as well as links to the mathematical background.

Portals and Directories restrict themselves to general information about a software, sometimes they cover also information about the current version but manual updating of the versions is expensive. Software portals and directories support the users finding relevant software. Mathematical software portals are focused on the content and its mathematical background, not on technical details.

Further relevant resources Services like the OEIS [Inc] or cloud computing are based on mathematical software. Journals specialized to software play a pioneering role for evaluating software. Programming languages and environments, e.g., R for statistics [CRAN], define the base for software development in special mathematical domains, benchmarks are essential for the evaluation of performance, web archives are relevant for long-term storing.

2.2 The FORCE11 Software Citation Principles

The Software Citation Group of the FORCE11 Initiative [FORCE11] – a large community of scholars, librarians, archivists, publishers and research funders that aim to help facilitate the change toward improved knowledge creation and sharing – has analyzed the state of the art of software citations, has worked out the needs to software information resulting from different use cases, and has discussed the basics of software citations. It has formulated its findings as **Software Citation Principles** (SCPs) [SDK], six requirements for citing of software:

SCP1. **Importance of Software**: Software is a legitimate and citable product of research [. . .]

SCP2. **Credit and Attribution**: Software citations should facilitate giving scholarly credit and normative and legal attribution to all contributors to the software [. . .]

SCP3. **Unique Identification**: A software citation should include a method for identification that is machine actionable, globally unique, interoperable, [. . .]

SCP4. **Persistence**: Unique identifiers and meta-data describing the software and its disposition should persist [. . .]

SCP5. **Accessibility**: Software citations should facilitate access to the software itself and to its associated meta-data, documentation, data, and other materials necessary for both humans and machines [. . .]

SCP6. **Specificity**: Software citations should facilitate identification of, and access to, the specific version of software that was used. [. . .]

[SDK] goes on to point out that the development of a citation standard is not an isolated problem but essentially connected with maintaining of software and cannot solved without developing concepts for documentation and long-term archiving of software.

2.3 Some consequences: Persistent Identifiers and Landing Pages

The huge amount of distributed information on the Web requires methods for persistent identification. There are many concepts and schemes for persistent identifiers, e.g., PURLs [PURL], URNs [URN], DOIs [DOI], and (for persons) ORCIDs [ORCID]. Moreover, publishers, libraries, and also the reviewing databases zbMATH [ZBM], MathSciNet [Ame] have own unique identifiers for their data.

SCP6 (specificity) distinguishes between general information about software and another for versions. So, **SCP3** (unique identification), requires unique identifiers for both the general information on software which we group to software product and on versions. Of course, software products and versions are closely related and overlapping. The relationship between software products and versions should be part of the meta-information on software, especially if the general information and the information about the versions are distributed.

SCP5 refers to the persistence of software information. The SCPs do not require that the software code must be persistent. This is the role of **software landing pages** introduced in [SDK] – i.e. web pages that contain meta-information about software. The main idea is that meta-information about software should be persistently available and citable – similar to bibliographic records of publications. Meta-data of publications are more or less standardized. Meta-data of software are more complex, cover information about software code but also about implementation, content, technical and legal data, context, etc. Meta-data of software are also strongly influenced from different use cases, for details see below.

So in a nutshell landing pages provide persistent meta-information about (changing) software. But [SDK] does not define the content of the landing pages and the meta-data on them. To remedy this shortcoming and make the idea practical we propose to use the meta-data vocabulary of the CodeMeta project, see [CM], for software landing pages.

2.4 CodeMeta Meta-data for Landing Pages

Some aspects of meta-information of software have been discussed in the CodeMeta project. Basing on analysis of the common information of big software archives, a set of widely used meta-data fields for a machine-readable exchange format of software meta-data has been selected. Currently CodeMeta lists more than 40 meta-data fields encoded in JSON-LD describing relevant features of a software product and its versions and the developing parties (persons and institutions, called agents in CodeMeta).

We use the concepts from the SCPs to give an overview over the fields in Figure 1: The citation block on the left gives the meta-data that is specific to the citation, here only the reference date of the software. The second block concentrates on the software

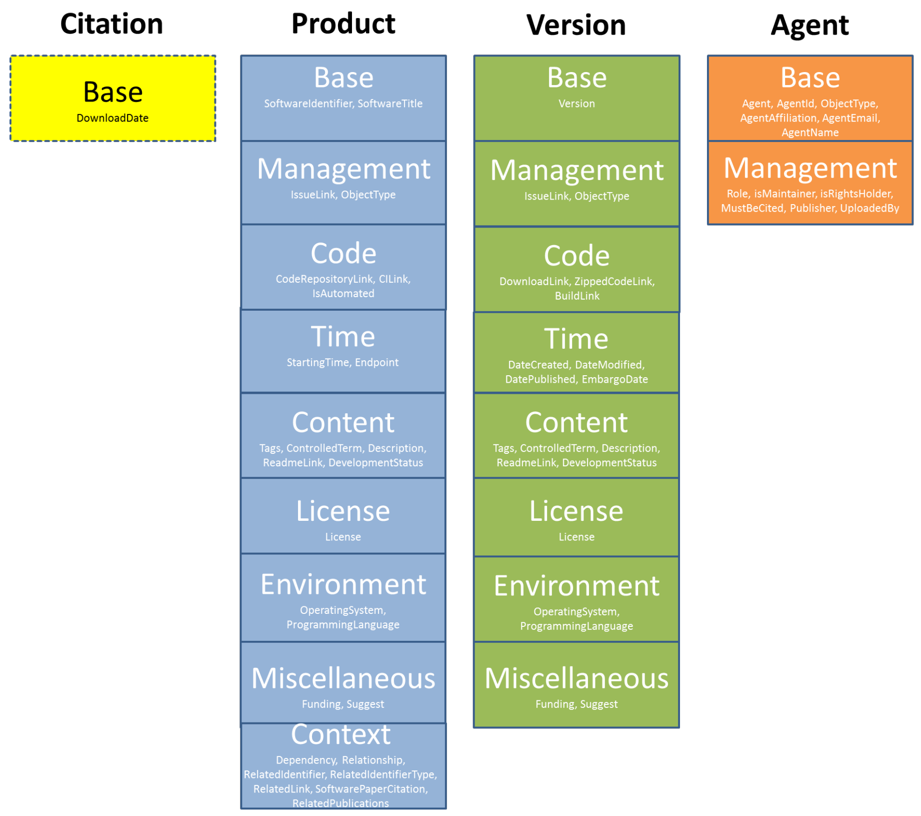

Citation
Base
DownloadDate
Product
Base
SoftwareIdentifier, SoftwareTitle
Management
IssueLink, ObjectType
Code
CodeRepositoryLink, CILink, IsAutomated
Time
StartingTime, Endpoint
Content
Tags, ControlledTerm, Description, ReadmeLink, DevelopmentStatus
License
License
Environment
OperatingSystem, ProgrammingLanguage
Miscellaneous
Funding, Suggest
Context
Dependency, Relationship, RelatedIdentifier, RelatedIdentifierType, RelatedLink, SoftwarePaperCitation, RelatedPublications
Version
Base
Version
Management
IssueLink, ObjectType
Code
DownloadLink, ZippedCodeLink, BuildLink
Time
DateCreated, DateModified, DatePublished, EmbargoDate
Content
Tags, ControlledTerm, Description, ReadmeLink, DevelopmentStatus
License
License
Environment
OperatingSystem, ProgrammingLanguage
Miscellaneous
Funding, Suggest
Agent
Base
Agent, AgentId, ObjectType, AgentAffiliation, AgentEmail, AgentName
Management
Role, isMaintainer, isRightsHolder, MustBeCited, Publisher, UploadedBy

Fig. 1. CodeMeta Meta-Data and Extensions

product itself. The version block specializes the meta-data that may change between versions (see **SCP6**), and finally the agent block is motivated by **SCP2**.

2.5 Software Citations

[SDK] also gives a recommendation for software citations in publications:

> *We recommend that all text citation styles support the following: a) a label indicating that this is software, e.g., [Software], potentially with more information such as [Software: Source Code], [Software: Executable], or [Software: Container], and b) support for version information, e.g., Version 1.8.7.*

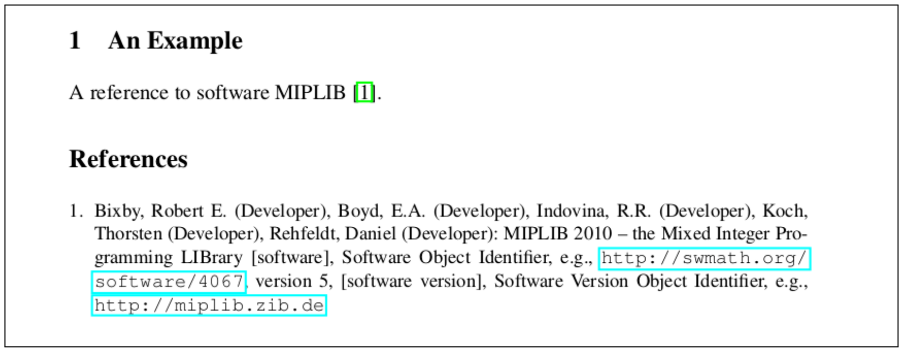
This proposal – while somewhat unspecific – significantly improves current practice of software citations in publications: The objects are uniquely defined as software and information about the version is given. We note that all information except the software type is already present in the CodeMeta meta-data set – which is the reason we propose it as the basis for software citations and landing pages.

The mathematical community typically uses TEX [CTAN], its bibliography add-on BibTEX, and increasingly BibLATEX [Leh10] for managing citations and creating references. BibLATEX provides citations formats for different object types, but unfortunately not for objects of type “software” or “software version” up to now.

To remedy this we have extended the BibLATEX version with the biber [Biber] backend by new entry types software and software version and corresponding meta-data schemes (by introducing field, especially for the typing, the role of agents or the dates for uploading and downloading. Figure 2 shows the source and result of a BibLATEX software citation. Note that we utilize the BibTEX/BibLATEX crossref directive to avoid duplicating information. According to **SCP2** we have special “author-type” keys for the various agent roles in the CodeMeta meta-data set.

```
@softwareversion{MIPLIB10:5,
  crossref = {MIPLIB10},
  version = {5},
  urldate = {2012−03−27}}
```

```
@software{MIPLIB10,
  developers = {Bixby, Robert E. and Boyd, E.A. and
                Koch, Thorsten and Rehfeld, Daniel},
  title = {MIPLIB 2010 − the Mixed Integer Programming LIBrary},
  swmath = {4067},
  url = {http://miplib.zib.de}}
```



1 An Example

A reference to software MIPLIB [1].

References

1. Bixby, Robert E. (Developer), Boyd, E.A. (Developer), Indovina, R.R. (Developer), Koch, Thorsten (Developer), Rehfeldt, Daniel (Developer): MIPLIB 2010 – the Mixed Integer Programming LIBrary [software], Software Object Identifier, e.g., http://swmath.org/software/4067, version 5, [software version], Software Version Object Identifier, e.g., http://miplib.zib.de

Fig. 2. An example for software citation

3 Persistent Identifiers, Landing Pages, and Versions in swMATH

The ideas formulated above can serve as a “requirement specification” for software citation metadata, and the bibTEX extension is an implementation of those. The main problem is that there are no public, comprehensive resources (software information systems) of landing pages that aggregate the kind of meta-information about software products and their versions either manually (software directories) or automatically (software information systems). For mathematical software, the situation is better: we have the swMATH [swMath] portal which supplies aggregated meta-data for more than 13,000 mathematical software systems. We will use swMATH in the following as an exemplar for a software information system.

3.1 The swMATH Information System for Mathematical Software

swMATH contains more than 17,000 records, each representing a software product with a unique identifier. swMATH is based on the more than 130,000 articles in the zbMATH publications database referring to mathematical software. The biggest challenge for a service like swMATH is to recognize these references. In many cases, only a name is mentioned, while a version or an explicit label as software is missing. swMATH tackles this with simple heuristics, by scanning titles, abstracts, as well as references of publications to detect typical terms – such as “solver”, “program”, or simply “software” – in combination with a name. After new candidates have been detected, they are checked manually to ensure high data quality. As part of this manual intervention step, additional meta-data, such as the URL of a software is added. Later on, websites are periodically checked and outdated URLs are removed or replaced. In case there is no permanent link that points to a website, the URLs of a corresponding repository record or a publication is used instead.

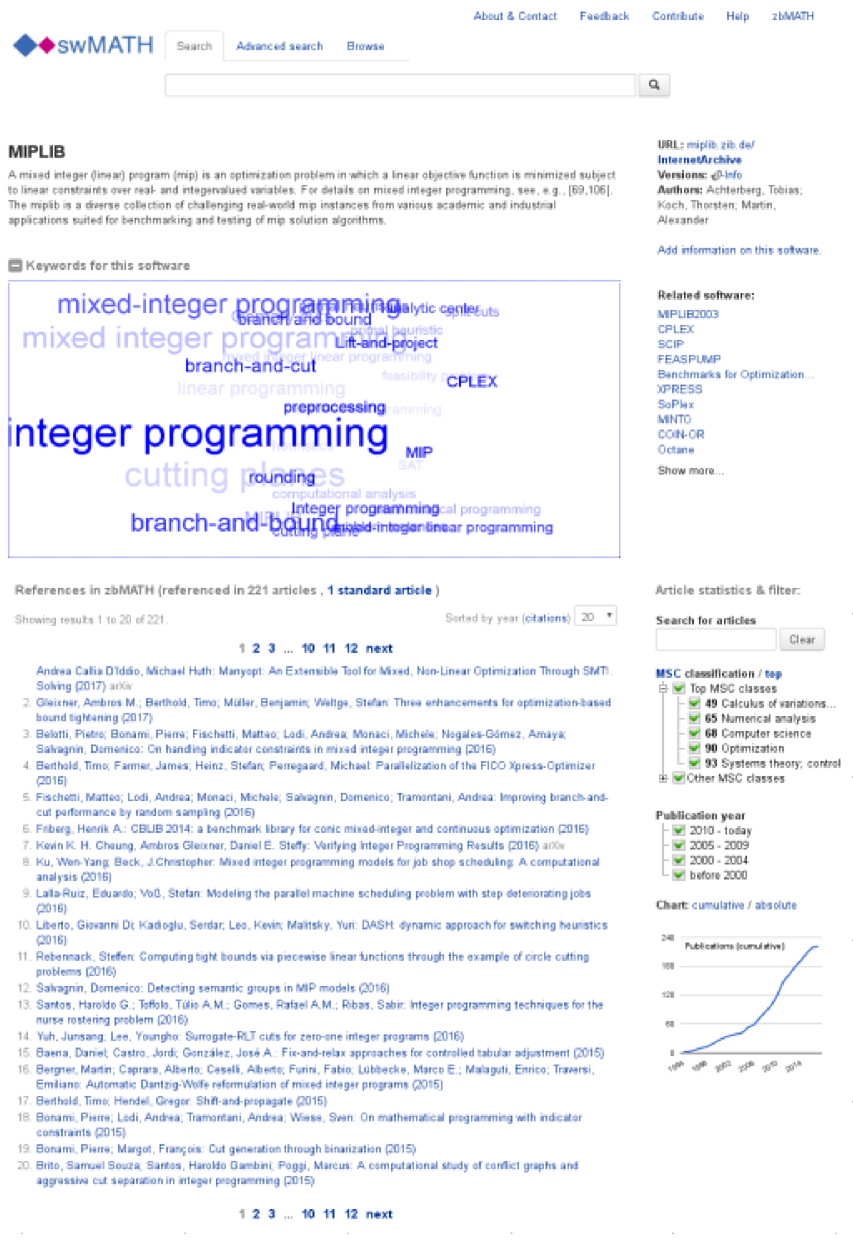
Another important feature for our analysis is the publication list for every software on swMATH. Each article in this list is annotated with its publication year. The publications can be sorted chronologically or by the number of citations an article has received. In swMATH, publications also serve as source for additional information, such as related software and the keyword cloud shown in every record (see Fig. 3).

Summarizing, the original approach of swMATH bases on an analysis of software citations in the mathematical literature. This indirect method provides statements especially about the mathematical background, the acceptance and applications of a software but not about versions, technical details and integrability of software. Therefore swMATH tries to extend the information about software by integrating and linking further resources.

3.2 The Wayback Machine and Micro-Archives for Software Versions

Web archives, e.g., the Internet Archive with the Wayback Machine [WB], periodically scan trusted websites. The Wayback Machine gives users the opportunity to nominate their websites for archiving and provides an overview page for each URL which documents the number of scans and the date of the scans. This is an interesting feature especially for software. It helps to avoid the so-called “reference rot”, links leading to nowhere or siginificantly modified resources. The problem of reference rot was addressed in the Hiberlink project [HL] which has created a macro-archive of scholarly publications – also from sources like the WayBack Machine – and has defined versioned links (hiberlinks), see [Kle+14].

The set of all scans provides a comprehensive – nearly complete – overview about the development of a software product. Crucially, it provides also information about different software versions.



MIPLIB

A mixed integer (linear) program (mip) is an optimization problem in which a linear objective function is minimized subject to linear constraints over real- and integervalued variables. For details on mixed integer programming, see, e.g., [69,106]. The miplib is a diverse collection of challenging real-world mip instances from various academic and industrial applications suited for benchmarking and testing of mip solution algorithms.

URL: miplib.zib.de/
InternetArchive
Versions: -Info
Authors: Achterberg, Tobias; Koch, Thorsten; Martin, Alexander

Add information on this software.

Keywords for this software

Related software:
MIPLIB2003
CPLEX
SCIP
FEASPUMP
Benchmarks for Optimization...
XPRESS
SoPlex
MINTO
COIN-OR
Octane
Show more...

References in zbMATH (referenced in 221 articles , 1 standard article)

Showing results 1 to 20 of 221.

Sorted by year (citations) 20

1 2 3 ... 10 11 12 next

1. Andrea Callia D'Iddio, Michael Huth: Manyopt: An Extensible Tool for Mixed, Non-Linear Optimization Through SMT Solving (2017) arXiv
2. Gleixner, Ambros M.; Berthold, Timo; Müller, Benjamin; Weltge, Stefan: Three enhancements for optimization-based bound tightening (2017)
3. Belotti, Pietro; Bonami, Pierre; Fischetti, Matteo; Lodi, Andrea; Monaci, Michele; Nogales-Gómez, Amaya; Salvagnin, Domenico: On handling indicator constraints in mixed integer programming (2016)
4. Berthold, Timo; Farmer, James; Heinz, Stefan; Perregaard, Michael: Parallelization of the FICO Xpress-Optimizer (2016)
5. Fischetti, Matteo; Lodi, Andrea; Monaci, Michele; Salvagnin, Domenico; Tramontani, Andrea: Improving branch-and-cut performance by random sampling (2016)
6. Friberg, Henrik A.: CBLIB 2014: a benchmark library for conic mixed-integer and continuous optimization (2016)
7. Kevin K. H. Cheung, Ambros Gleixner, Daniel E. Steffy: Verifying Integer Programming Results (2016) arXiv
8. Ku, Wen-Yang; Beck, J.Christopher: Mixed integer programming models for job shop scheduling: A computational analysis (2016)
9. Lalla-Ruiz, Eduardo; Voß, Stefan: Modeling the parallel machine scheduling problem with step deteriorating jobs (2016)
10. Liberto, Giovanni Di; Kadioglu, Serdar; Leo, Kevin; Malitsky, Yuri: DASH: dynamic approach for switching heuristics (2016)
11. Rebennack, Steffen: Computing tight bounds via piecewise linear functions through the example of circle cutting problems (2016)
12. Salvagnin, Domenico: Detecting semantic groups in MIP models (2016)
13. Santos, Haroldo G.; Toffolo, Túlio A.M.; Gomes, Rafael A.M.; Ribas, Sabir: Integer programming techniques for the nurse rostering problem (2016)
14. Yuh, Junsang; Lee, Youngho: Surrogate-RLT cuts for zero-one integer programs (2016)
15. Baena, Daniel; Castro, Jordi; González, José A.: Fix-and-relax approaches for controlled tabular adjustment (2015)
16. Bergner, Martin; Caprara, Alberto; Ceselli, Alberto; Furini, Fabio; Lübbecke, Marco E.; Malaguti, Enrico; Traversi, Emiliano: Automatic Dantzig-Wolfe reformulation of mixed integer programs (2015)
17. Berthold, Timo; Hendel, Gregor: Shift-and-propagate (2015)
18. Bonami, Pierre; Lodi, Andrea; Tramontani, Andrea; Wiese, Sven: On mathematical programming with indicator constraints (2015)
19. Bonami, Pierre; Margot, François: Cut generation through binarization (2015)
20. Brito, Samuel Souza; Santos, Haroldo Gambini; Poggi, Marcus: A computational study of conflict graphs and aggressive cut separation in integer programming (2015)

1 2 3 ... 10 11 12 next

Article statistics & filter:

Search for articles
Clear

MSC classification / top
- ✓ Top MSC classes
  - ✓ **49** Calculus of variations...
  - ✓ **65** Numerical analysis
  - ✓ **68** Computer science
  - ✓ **90** Optimization
  - ✓ **93** Systems theory; control
- ✓ Other MSC classes

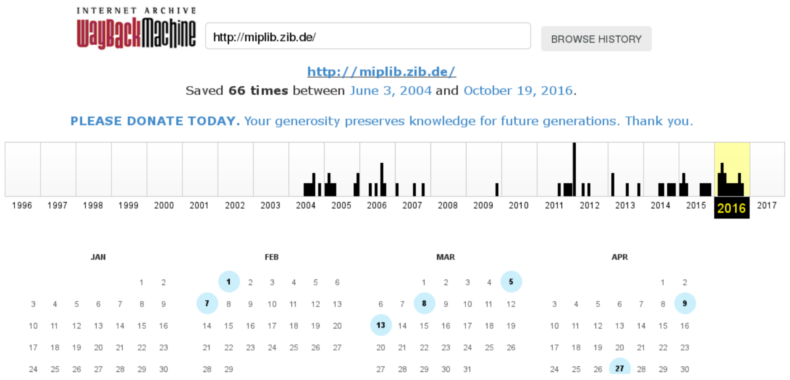
Publication year
- ✓ 2010 - today
- ✓ 2005 - 2009
- ✓ 2000 - 2004
- ✓ before 2000

Chart: cumulative / absolute

Fig. 3. The swMATH (landing) page for MIPLIB

This allows to create **micro-archives** for a mathematical software based on the timed scans of the websites by the Wayback Machine. This idea was developed into an swMATH extension by Helge Holzmann [HSR16]. He used the swMATH sftoware

URLs as seed list and analyzed and grouped the stored scans of the websites in the Internet Machine.


http://miplib.zib.de/

Saved **66 times** between June 3, 2004 and October 19, 2016.

PLEASE DONATE TODAY. Your generosity preserves knowledge for future generations. Thank you.

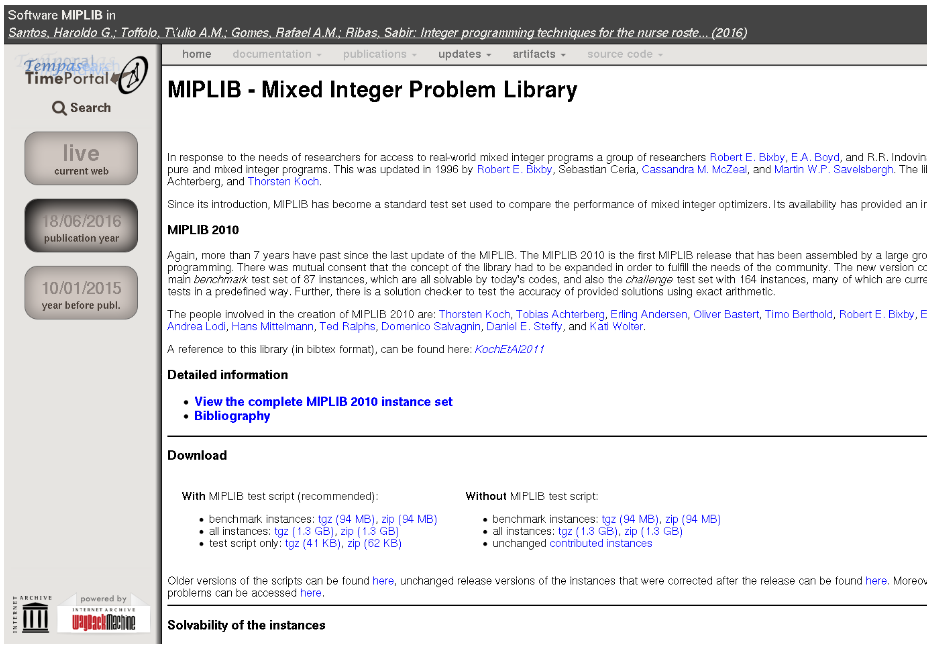
Fig. 4. The Wayback Machine page for the software MIPLIB

The information on each scan of a website representing the development stage of the software at a certain date – in other words: a certain version – can be uniformly classified, in particular with respect to

– Software name and version
– Documentation, manual
– Software code
– Developers
– Contact
– License information and legal rights
– Environment and programming languages, technical parameters
– Citation recommendation
– APIs
– Further data, e.g., publications or the development of a software, test data, benchmarks, etc.

This classification scheme constitutes an additional uniform structure for software metadata, which can be aggregated into a secondary link page for the software information and represents a **landing page for a particular version** of the software product. It contains links to meta-data descriptions for the objects – if existing – in the classes and/or also direct links to the objects. The micro-archives are defined as the link pages and the linked resources in the Wayback Machine.

The concept of micro-archives is flexible. Meta-data schemes can be involved but they are not a necessary requirement. In a first attempt, the documents on the websites are classified by their document formats. This simple method is only a first approach, we plan to use more enhanced tools, e.g., machine learning, for classification. In essence, the scans of a website together form a web archive for the software which can be analyzed to distinguish the general from version-specific information. With this we can create landing pages not only for the software product, but also for its versions.



Software **MIPLIB** in
Santos, Haroldo G.; Toffolo, T\'ulio A.M.; Gomes, Rafael A.M.; Ribas, Sabir: Integer programming techniques for the nurse roste... (2016)

MIPLIB - Mixed Integer Problem Library

In response to the needs of researchers for access to real-world mixed integer programs a group of researchers Robert E. Bixby, E.A. Boyd, and R.R. Indovin
pure and mixed integer programs. This was updated in 1996 by Robert E. Bixby, Sebastian Ceria, Cassandra M. McZeal, and Martin W.P. Savelsbergh. The li
Achterberg, and Thorsten Koch.

Since its introduction, MIPLIB has become a standard test set used to compare the performance of mixed integer optimizers. Its availability has provided an ir

MIPLIB 2010

Again, more than 7 years have past since the last update of the MIPLIB. The MIPLIB 2010 is the first MIPLIB release that has been assembled by a large gro
programming. There was mutual consent that the concept of the library had to be expanded in order to fulfill the needs of the community. The new version cc
main *benchmark* test set of 87 instances, which are all solvable by today's codes, and also the *challenge* test set with 164 instances, many of which are curre
tests in a predefined way. Further, there is a solution checker to test the accuracy of provided solutions using exact arithmetic.

The people involved in the creation of MIPLIB 2010 are: Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, E
Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter.

A reference to this library (in bibtex format), can be found here: *KochEtAl2011*

Detailed information

- **View the complete MIPLIB 2010 instance set**
- **Bibliography**

Download

With MIPLIB test script (recommended):

- benchmark instances: tgz (94 MB), zip (94 MB)
- all instances: tgz (1.3 GB), zip (1.3 GB)
- test script only: tgz (41 KB), zip (62 KB)

Without MIPLIB test script:

- benchmark instances: tgz (94 MB), zip (94 MB)
- all instances: tgz (1.3 GB), zip (1.3 GB)
- unchanged contributed instances

Older versions of the scripts can be found here, unchanged release versions of the instances that were corrected after the release can be found here. Moreov
problems can be accessed here.

Solvability of the instances

Fig. 5. Cutout of a micro-archive for the software MIPLIB

One current handicap for software is the missing standardization, completeness, specification, and semantification of information on the websites of software, e.g., missing information about APIs or the availability of source code. But this information is relevant for the applicability, integrability, and further development of software. The problem of enrichment of software information is addressed in Section 4.

3.3 Software Citations with swMATH Landing Pages

We observe that the swMATH pages and micro-archives are suitable candidates for landing pages in software citations: they are

1. *comprehensive*: all mathematical software mentioned in the literature have a swMATH page, even if they do not have a web page (recall, that is 1/3 of the systems). The swMATH pages combine the direct information about a software resulting from the micro-archives and the indirect information about a software coming from the literature.

2. *informative*: they aggregate information even if the software website does not have it (e.g. the publication lists, word clouds, and authors).
3. *persistent* and *specific*: they provide persistent identifiers and URLs (see also Section 3.4 below).
4. *low-maintenance*: analyzing publications and websites can be processed widely automated.

Therefore swMATH has added links to the overview page of a software in the Wayback Machine and from each publication to the corresponding version of thoftware. If the version is not explicitly referenced in a publication, the micro-archive is assigned to a paper by comparing the publication dates of the paper and the software versions.

Their main shortcoming is that they are optimized to be human-readable and do not currently offer access to the CodeMeta meta-data in a machine-oriented way. This can be easily remedied by a suitable web service that does as the swMATH database has (most of) the relevant information.

3.4 Persistent Identifiers for Software products and Software Versions in swMATH

In analogy to the DOIs we propose to use the swMATH identifiers (natural numbers: the mathematical software products are consecutively numbered as they become known to the system) as principal components for **Software Object Identifiers** (SOI) for a software product. Concretely, we propose to use identifiers of the form swMATH:4607 as SOIs; this allows to add different collections of software landing pages under other prefixes – e.g. for a software information system derived from a software directory; if there are overlaps, a conflict resolution mechanism could be established.

For the respective versions of the software product we can define **Software Version Object Identifiers** (SVOI) accordingly. As the general information about the software product and a special version are closely related, the SVOIs reference the corresponding SOI. Thus it seems to be natural to design the SOIVs as a two stage identifier which concatenates the SOI and the notation of the version. As separator can be used for example the '/' symbol. So for version 5 of the MIPLIB package from Figure 2, we would get the SVOI swMATH:4607/5; this would avoid separate identifiers for the software product and software version.

Note that the swMATH SOI for a mathematical software product only needs to be prefixed by `http://swmath.org/software/` to obtain the URL of the landing page. Thus swMATH directly acts as a catalogue service like `http://doi.org` that is often used to hyperlink DOI-based citations. With the concept of publicly accessible SOIs software citations could be simplified to

⟪Agent⟫ (⟪role⟫): ⟪Software Title⟫, ⟪Version⟫, [software], [⟪SVOI⟫], ⟪date of the version⟫

as all other information can be obtained from the software landing page (or the corresponding meta-data web service).

We are currently experimenting with web services on top swMATH, which allow to generate BibLaTeX entries of the form described above directly for any SOI/SVOI in the swMATH database.

Of course, other software information services could define also similar name conventions for SOIs and SVOIs which could be maintained by a trusted institution, e.g., by the DOI service of TIB.

4 Fine-Grained References for Software Libraries and Open-API-Systems

The discussion above concentrated on software products, i.e. software systems that can be described adequately as a monolithical entity without discernable sub-systems. This is inadequate for many mathematical software systems, including *a*) software libraries like the NumPy library [NPy] in Python which supplies a set of interface functions to be used for numerical computations in Python programs or *b*) computer algebra systems like GAP [GAP], Sage [Sage], or Mathematica. Systems like the latter provide a programming interface that gives access to a set of API functions. In the case of Sage, this is the Python language, for GAP and Mathematica, these are system-specific languages. Note that even proprietary libraries and systems make the specifications of the APIs public, therefore we collectively speak of **open-API** systems. This class also contains programs like Office suites and CAD programs.

For open-API systems, we often want to reference specific API functions rather than the whole systems. Reasons range from reproducibility of scholarly results – e.g., how exactly did we convert the input data in MatLab – to discussing unexpected behaviors of particular API functions of open-API systems.

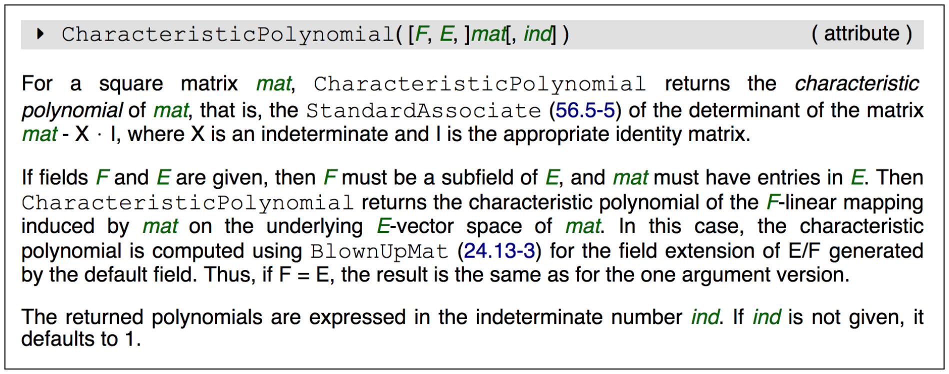
In analogy to the landing pages for software products we discussed above, we would need landing pages with meta-data for individual API functions. The most immediate way to do this is to reference a specific fragment of the manual, e.g., the documentation of `CharacteristicPolynomial` in GAP (see Figure 6). But this approach does not satisfy the SCP principles discussed above. The first problem is that for instance **SCP3** calls for unique identifiers. The most obvious choice for this would be the manual and the function name but in software with method overloading (for instance GAP) this is ambiguous. Moreover most manuals do not use the function name as document fragment identifiers. In the case of the GAP manual the nearest anchor is `https://www.gap-system.org/Manuals/doc/ref/chap24.html#X87FA0A727CDB060B`. One wonders whether this generated anchor is persistent over versions.

Manual pages usually give the **functional "meta-data" for API functions**:

i) the function name,
ii) the call pattern; i.e. arguments, possibly argument and return type, and
iii) a natural language description of the result to be expected and – if applicable – any side effects.

But there is no standardized and machine-actionable way of accessing these.

In the OpenDreamKit [ODK] project we have extended the idea of referencing API functions to a machine-understandable level. The goal of the OpenDreamKit is to develop a framework for virtual research environments for computational mathematics building on existing open-source systems like GAP, Sage, LMFDB, PARI-GP, and Jupyter. The integration of these systems requires a semantic description of their APIs, so that the respective object constructors and functions can be related to each


▸ CharacteristicPolynomial([*F*, *E*,]*mat*[, *ind*]) (attribute)

For a square matrix *mat*, CharacteristicPolynomial returns the *characteristic polynomial* of *mat*, that is, the StandardAssociate (56.5-5) of the determinant of the matrix *mat* - X · I, where X is an indeterminate and I is the appropriate identity matrix.

If fields *F* and *E* are given, then *F* must be a subfield of *E*, and *mat* must have entries in *E*. Then CharacteristicPolynomial returns the characteristic polynomial of the *F*-linear mapping induced by *mat* on the underlying *E*-vector space of *mat*. In this case, the characteristic polynomial is computed using BlownUpMat (24.13-3) for the field extension of E/F generated by the default field. Thus, if F = E, the result is the same as for the one argument version.

The returned polynomials are expressed in the indeterminate number *ind*. If *ind* is not given, it defaults to 1.

Fig. 6. The documentation of `CharacteristicFunction` in GAP

other. Therefore we generate system-API-specfic OMDoc/MMT Content Dictionaries that contain the functional API meta-data in form of *i*) symbol name, *ii*) type, and *iii*) CMP (commented mathematical property; a natural language description of the symbol meaning). We currently have ca. 350 generated CDs for the GAP system with an order of magnitude more classes and methods, and have recently added about the same amount of CDs for Sage. These CDs describe the system APIs in a uniform fashion and are keyed by function/constructor/class/- and category name. From this we can not only generate uniform human-oriented "landing pages" directly in the MMT system [Rab13], but we can also develop machine-oriented services that export linked open data about the system APIs, web services that generate BibTEX databases, and translation systems that make systems interoperable by aligning their APIs.

5 Conclusion & Future Work

We have discussed the problem of software citations in the special case of mathematical software. Based on the six "software citation principles" and the idea of software landing pages put forward by the FORCE11 group, we have proposed the CodeMeta meta-data vocabulary for landing pages and shown that the information in the swMATH software portal is sufficient to generate landing pages and persistent software identifiers for mathematical software. We have experimented with two extensions of this swMATH-based approach.

i) micro-archives as collections of version-specific landing pages to combat the specificity requirement and

ii) generated content dictionaries for referencing the API functions to alleviate the granularity restrictions of only dealing with whole software products.

Even though the swMATH portal limits itself to mathematical software, since it mainly relies on the zbMATH corpus for aggregating, the document-based aggregation method is not limited to mathematical software and could be extended given a suitably complete corpus (e.g., the Cornell ePrint arXiv, NumDam, PubMed or the back files of a large scientific publishing house). Given these considerations, we are confident that the ideas exposed here can be ported to major parts of the scholarly literature. As so often is the case (e.g., MathML was the first XML format standardized by the W3C) Mathematics (and Mathematical Knowledge Management) can play the role of a front-runner in scientific publication infrastructure.

Acknowledgements

We gratefully acknowledge fruitful discussions with Hagen Chrapary, Wolfgang Dalitz, Helge Holzmann, Heinz Kröger, Fabian Müller, Winfried Neun, and Olaf Teschke on software citations and the contributions of Paul-Olivier Dehaye, Alexander Konovalov, Marcus Pfeiffer, Nicolas Thierry, in the generation of OpenDreamKit content dictionaries. Finally, we acknowledge funding the of the research campus MODAL for the swMATH project and EU funding for the OpenDreamKit project in the Horizon 2020 framework under grant 676541.

References

[Biber] *biber - A BibTeX replacement for users of BibLATEX*. URL: https://www.ctan.org/pkg/biber (visited on 03/29/2017).

[Chr+17] Hagen Chrapary et al. "Design, Concepts, and the State of the Art of the swMATH Service". In: *Math.Comput.Sci* (2017). DOI: 10.1007/s11786-017-0305-5.

[CICM1616] Paul-Olivier Dehaye et al. "Interoperability in the OpenDreamKit Project: The Math-in-the-Middle Approach". In: *Intelligent Computer Mathematics 2016*. Ed. by Michael Kohlhase et al. LNCS 9791. Springer, 2016. URL: https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP6/CICM2016/published.pdf.

[CM] *CodeMeta CodeMeta focuses on metadata and discovery systems for software citation and attribution*. URL: https://github.com/codemeta/codemeta/blob/master/codemeta-concepts.md (visited on 03/29/2017).

[CRAN] *The Comprehensive R Archive Network*. URL: https://cran.r-project.org/ (visited on 03/29/2017).

[CTAN] *CTAN the Comprehensive TEX Archive Network*. URL: http://ctan.org (visited on 12/11/2012).

[DOI] *DOI*. URL: https://www.doi.org/ (visited on 03/29/2017).

[FORCE11] *FORCE11 — The future of research communications and e-scholarship*. URL: https://www.force11.org/ (visited on 03/29/2017).

[GAP] The GAP Group. *GAP – Groups, Algorithms, and Programming*. URL: http://www.gap-system.org (visited on 08/30/2016).

[HL] *hiberlink*. URL: http://hiberlink.org (visited on 03/12/2017).

[HSR16] Helge Holzmann, Wolfram Sperber, and Mila Runnwerth. "Archiving Software Surrogates on the Web for Future Reference". In: LNCS 9819 (2016), pp. 215–226.

[Inc] OEIS Foundation Inc., ed. *The On-Line Encyclopedia of Integer Sequences*. URL: `http://oeis.org` (visited on 05/28/2013).

[Kle+14] Martin Klein et al. "Scholarly Context Not Found: One in Five Articles Suffers from Reference Rot". In: *PLOS One* (2014). DOI: `10.1371/journal.pone.0115253`.

[Leh10] Philipp Lehmann. *The `biblatex` Package*. Tech. rep. CTAN: Comprehensive TEX Archive Network, 2010. URL: `http://ctan.org/pkg/biblatex`.

[NPy] *NumPy*. URL: `http://www.numpy.org/` (visited on 03/29/2017).

[ODK] *OpenDreamKit Open Digital Research Environment Toolkit for the Advancement of Mathematics*. URL: `http://opendreamkit.org` (visited on 05/21/2015).

[ORCID] *ORCID*. URL: `https://orcid.org/` (visited on 03/29/2017).

[PURL] *PURL Administration*. URL: `https://archive.org/services/purl/` (visited on 03/29/2017).

[Rab13] Florian Rabe. "The MMT API: A Generic MKM System". In: *Intelligent Computer Mathematics*. Ed. by Jacques Carette et al. Lecture Notes in Computer Science 7961. Springer, 2013, pp. 339–343. ISBN: 978-3-642-39319-8. DOI: `10.1007/978-3-642-39320-4`.

[Sage] The Sage Developers. *SageMath, the Sage Mathematics Software System*. URL: `ttp://www.sagemath.org` (visited on 09/30/2016).

[SDK] AM Smith, Katz DS, and Niemeyer KE. *FORCE11 Software Citation Working Group. (2016) Software Citation Principles*. URL: `https://peerj.com/preprints/2169/`.

[swMath] *swMath an information system for mathemtical Software*. URL: `http://www.swmath.org` (visited on 03/29/2017).

[URN] *Persistent Identifier*. URL: `http://www.persistent-identifier.de/` (visited on 03/29/2017).

[WB] *Internet Archive WayBack Machine*. URL: `https://archive.org/web/` (visited on 03/29/2017).

[ZBM] *Zentralblatt MATH*. URL: `http://www.zentralblatt-math.org/zbmath/` (visited on 06/12/2012).

[Ame] American Mathematical Society. *MathSciNet Mathematical Reviews on the Net*. URL: `http://www.ams.org/mathscinet/` (visited on 08/05/2010).