

GraphArchive

An Online Graph Data Store

Philip Effinger¹, Michael Kaufmann¹,
Sascha Meinert², Matthias Stegmaier¹

WSI-2011-03

ISSN 0946-3852

¹ Arbeitsbereich Algorithmik
Wilhelm-Schickard-Institut für Informatik
Mathematisch-Naturwissenschaftliche Fakultät
Eberhard Karls Universität
Sand 14, 72076 Tübingen, Germany
Email:
{effinger,mk,stegmaie}@informatik.uni-tuebingen.de

² Lehrstuhl Algorithmik I
Institut für Theoretische Informatik
Karlsruher Institut für Technologie
Am Fasanengarten 5
76131 Karlsruhe, Germany

meinert@kit.edu

1 Introduction

One of the most powerful tools to model a problem is a *graph*. It allows for an abstract representation of objects that are related to each other. This representation can then be used to solve the problem at hand and makes graphs so powerful that researchers all over the world work on problems that are modeled using graphs. Often, their task is to improve the quality of the solution, to solve new problems or at least solve known problems faster than it was possible before. However, if approaches should be compared experimentally they have to be tested on the same data set.

Sharing graphs among researchers might be an obvious solution to distribute data sets and indeed, various approaches exist that tackle the problem of distributing data sets, e.g., Stanford GraphBase [4], Matrix Market [5] DIMACS [2] or UF SMC [1].

However, our approach is neither dedicated to a specific area of application nor to provide single benchmark sets. We aim at establishing a central repository that contains valuable data for every interested researcher. Therefore, we present *GraphArchive* in this report, which is our approach of a central graph repository.

GraphArchive is a *web-based platform* for sharing graphs. The back-end of this system consists of a database that stores the graphs. GraphArchive automatically computes images of graph layouts and analyzes basic graph properties. As already mentioned GraphArchive is no special purpose graph repository but allows for arbitrary domains of interest. Hence, the system also provides a search mechanism, which allows for a free search on multiple attributes. GraphArchive follows the basic principles of *GraphDB*¹, which is also a platform for sharing graphs in a central place, but its development has been discontinued.

GraphArchive ports the approach of GraphDB to the web including the benefits from online platforms, e.g., user-interactivity, email notifications or automatic graph analysis and layout. Our major goal is to provide an easy-to-use and yet minimalist platform to maximize the usability, which is a main criterion for user acceptance. To this end, we work with and support users that often have to deal with graphs, e.g., researchers from the GraphDrawing (GD) community.

In the following we present details on our approach GraphArchive, which allows for storing graphs in a central place. We start by briefly describing the idea of a graph archive and the history of GraphDB in Section 2. Then, we will present the underlying architecture, supported features and principles that build the foundation of our new approach GraphArchive in Section 3. In Section 4, we show the usability of our system by the demonstration of a user's typical work flow. We conclude this report with future topics in Section 5.

2 History of the GraphDB

In this section we give an overview of the history of the *GraphDB*. First, we start with the motivation and the origin of the idea to build a graph archive. Then, the features of the system as well as their intended interplay are presented. Finally, we analyze the acceptance-rate of GraphDB, which led to the conclusion that an update of GraphDB would not be sufficient and that a new graph archive system had to be developed.

¹Information on GraphDB can be found at <http://www.graph-archive.org>, last-accessed 2011-07-30

2.1 Motivation and Origin

Several working groups participated in the Priority Programme no. 1126 “Algorithmics of Large and COmplex Networks” (ALCON) of the Deutsche Forschungsgemeinschaft (DFG, Germany’s largest research funding organisation). Within this Priority Programme many participants started collaborations with each other, if not already present. Working groups that were interested in experimental algorithmics needed to exchange their data sets and results. First, this was done via e-mail or, due to the size of the data sets, via servers on each working group’s sites. Doing so does not preserve the *reliability* and the *repeatability* of experiments, which renders both exchange methods questionable. For example, if data is exchanged that way, no possibility exists to version the graphs, or to prevent neither data corruption nor data losses. In addition to the problems that may occur when handling the data, the acquisition of data sets itself can become a problem, too. Sites that provide data sets are scattered and acquiring data can often be a tedious and long lasting task, which distracts researchers from their scientific work.

All this led to the idea to develop and maintain a central repository, where people can exchange and archive their data and results at the same time. Several participants of the Priority Programme ALCON worked together to specify the demands on such a graph archive. The early approach of a system should:

- allow to *exchange* and *archive* graphs in *heterogeneous* systems
- work via the *Internet* despite the presence of *firewalls*
- be *persistently* stored in a *central* place
- allow to *add*, *maintain*, *query* and *download* graphs

2.2 GraphDB

The first system that realized the above demands on a graph archive was *GraphDB*, which was designed by Sascha Meinert within the scope of his master thesis. The three-tier architecture consists of a downloadable *client*, a *server* and a *database*. In the context of his master thesis, a prototype system was developed to validate whether the system was capable to fulfill the given requirements.

In 2004, when the development of GraphDB started, technologies that allowed for running an application within a browser, were not available. In particular, the *AJAX* technology was presented in 2005 [3]. Hence, we decided to develop a stand-alone client. To fulfill the requirement to work in heterogeneous systems the client and the server were written in *Java*. The communication between client and server is based on *web-services* to allow for secure message transportation in the internet despite the presence of firewalls. Additionally, this XML-based form of communication allows third-party applications to directly access the graph archive, e.g., automated test tools. The central server uses as back-end a database to persistently store business objects and binary data.

GraphDB elements As already mentioned, the GraphDB system should allow participants to exchange and archive large graphs and results. To realize this demand, the prototype system allowed the user to interact with the following elements:

graph: an atomic element that represents binary data

graph groups: a container element that groups graphs
meta-data: a (key, type, value) triple, e.g., (*directed*, *boolean*, *true*)
result: a (key, type, value) triple, e.g., (*max degree*, *integer*, *500*)

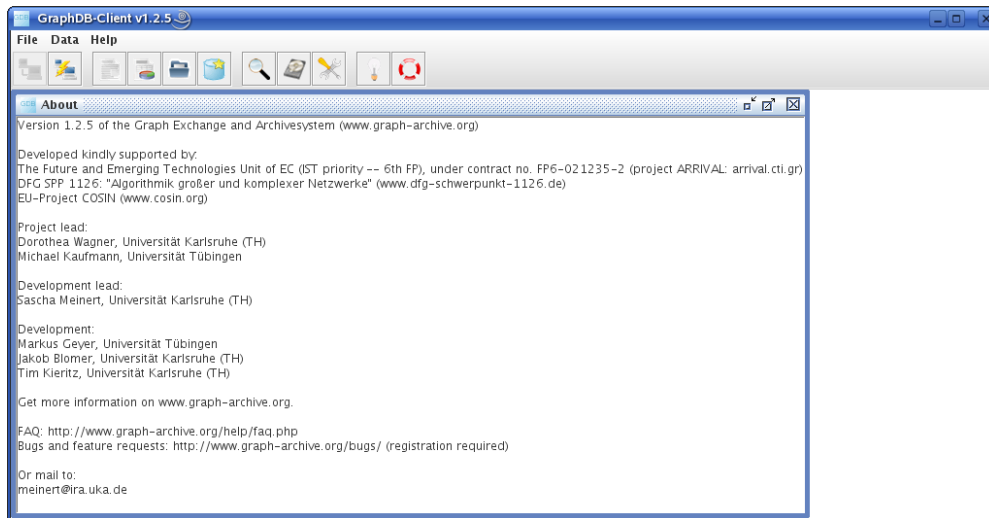
Both meta data as well as results can be created or reviewed in the corresponding administration panel; see Figure 1b. During the *creation* of such an element the *key* and the *type* is specified. Note that meta data and results look identical but are separated to clarify their semantic usage. Graphs and graph groups can be added to the system, each getting a permanent *unique ID* (UID). Both graph groups and graphs can be further specified by attaching meta data to them. While *attaching* a meta-data element, its *value* has to be specified. If a graph is specified to be *part of* a graph group, the graph *inherits* the graph group's meta data. To allow for sharing findings on a specific graph, results can be attached to a graph similar to meta-data elements. A query mechanism allows to search the system for graphs or graph groups by UID, meta data or results. Figure 2a shows the query panel after a query has been performed. The detailed information of a selected graph group can be seen as well as the graphs that are members of the graph group. The detailed information of a graph can be seen in Figure 2b. Note that in this example all meta data is inherited from the parent graph group. The graphs or graph groups found by such a query can be downloaded afterwards. This prototype implementation was able to fulfill the requirements and it was presented to the participants of ALCON.

This resulted in the decision to further develop the prototype to release the first stable version with financial support from the Priority Programme. In the following time several bugs were removed that occurred during heavy load and multiple user tests, e.g., race conditions. Additionally, the overall performance as well as the usability of the application was greatly improved. The resulting final version 1.0 of GraphDB was advertised within ALCON and access was given to all of its participants, which led to a rather hesitant usage.

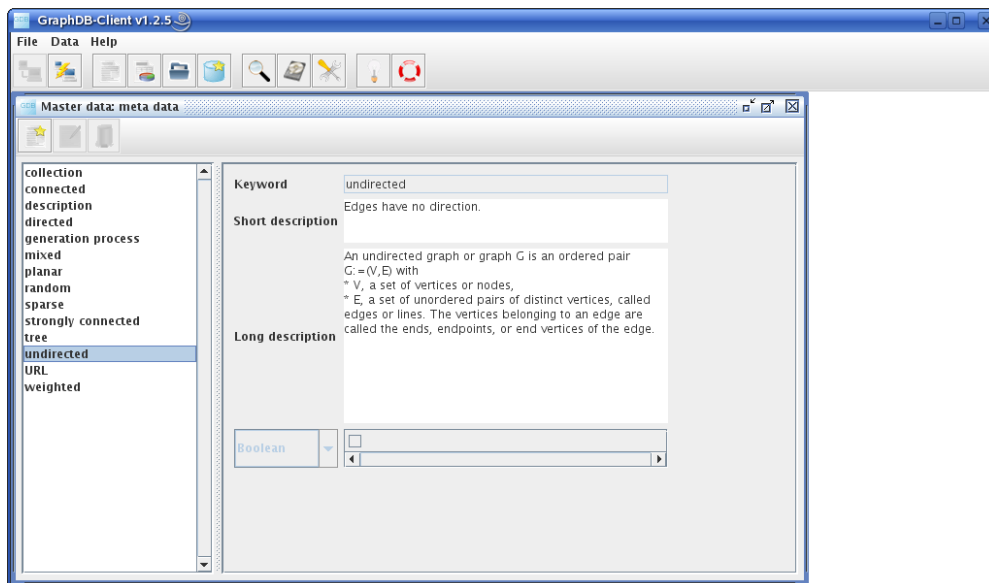
GraphDB licensing extension Some members of ALCON were also participants of the project “Algorithms for Robust and online Railway optimization: Improving the Validity and reliability of Large scale systems” (ARRIVAL), which is supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2. One aim of the project was to bring results from academic research to the industry. Hence, many participants of ARRIVAL collaborated with national railway companies, which provided confidential data. So ALCON members came up with the idea to extend the demands on a graph archive by security and licensing elements that would allow to use a graph archive within ARRIVAL. Hence, with the financial support of ARRIVAL GraphDB was further developed. The system already contained the concept of users but no direct interaction was possible. The features of this extension are:

user: used for authentication and identification of server interactions
user group: a container element that groups users
visibility: type of the visibility of the data, one of:

- *public:* data can be seen by everyone
- *semi-private:* data can be seen by certain groups
- *private:* data can only be seen by its creator



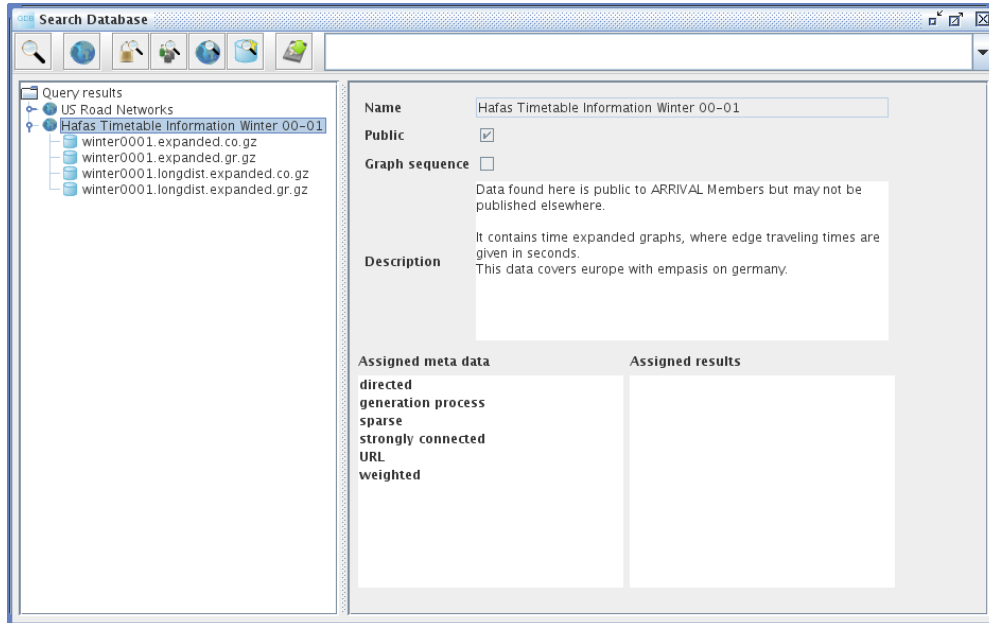
(a) GraphDB client standard view after login



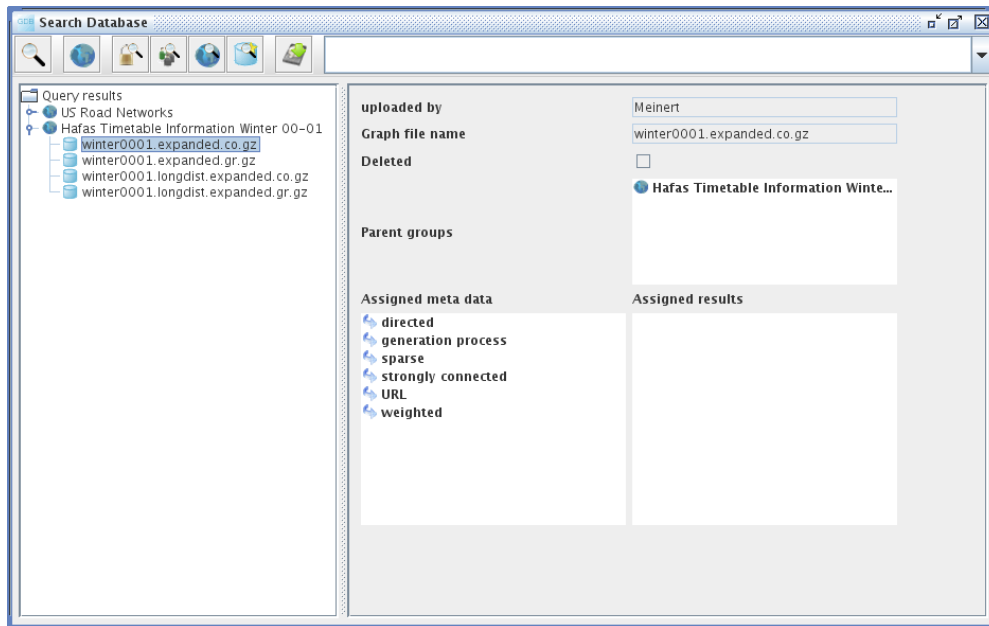
(b) GraphDB overview and administration screen of meta data

Figure 1: The standard view of the GraphDB client after the successful login (a). The features of the client can be quickly accessed via icons.

The bottom figure (b) shows the overview and administration screen of the meta data. This screen allows for either reviewing already existing meta-data elements or creating new elements. On the left-hand side the existing meta-data elements are listed. On the right-hand side the detailed information of the selected meta-data element is shown.



(a) GraphDB query screen with a selected graph group



(b) GraphDB query screen with a selected graph

Figure 2: The top figure (a) shows a query result on the left-hand side of the panel, where one graph group has been selected. The detailed information of this selected graph group is shown on the right-hand side of the panel. The bottom figure (b) shows the detailed information of the selected graph, which is a member of the previously selected graph group.

Now, users can be assigned to be members of user groups. This has to be done by the administrating authority. The implementation reduces the visibility of data to its users. Thus, when data sets are created, the creator has to assign the visibility level. Depending on that choice he also has to assign who may view his data. The server-side query mechanism additionally checks the visibility of the data and delivers only the subset the requesting user may access; see Figure 2a. After the new security requirements had been added, access to GraphDB was given to the members of ARRIVAL, who could now use the system to exchange their data in a safe way.

2.3 Lessons learned

One of the basic reasons to build a graph archive is the long lasting and tedious task of gathering data — especially when dealing with real-world data sets. Often, companies fear that competitors might get access to this data and might thus gain a competitive advantage. Other reasons that might prohibit data sharing are licensing policies or data privacy policies.

In order to become a primary resource, the graph archive needs two things. The first is a valuable and large collection of data sets. The second, which is an immediate consequence of the former, is a strong community that participates by not only acquiring but also providing data. A large pool of data sets will attract people and if they find valuable data, they are willing to give something back.

For several reasons GraphDB was not accepted by the community in the way we hoped for. From our experience with GraphDB, we conclude the following recommendations for a follow-up system:

- **access:** access to the data should be as easy as possible. Users prefer websites over downloadable clients. Additionally, the method to get access to the system itself should be easy too.
- **rights management:** The latest security requirements for GraphDB did not allow for an automated registration system (only guests were allowed, which had very limited download rights). Users do not want to care about licensing policies, they prefer open-source data. This allows for an easy trade of data to give and data to get.
- **data formats and conversion:** Users do not want to care about file formats. If there is no common base file format they will in most cases not spend time on a conversion tool. Hence, a graph archive should support multiple file formats and, possibly, their conversion.
- **graph analysis:** Users do not want to spend time on computing or maintaining basic properties of data sets. Thus, an automated property test, which is run for uploaded data sets can do the job.

All of these aspects influenced the requirements and design decisions done with our new system, which is presented in the next section.

3 Features of the new GraphArchive

In the following, we will provide a list of the main features of our new approach and present our system architecture. Then, we will present selected key features in more detail.

3.1 Main features of GraphArchive

All features are chosen supporting the guideline that our major goal is to provide an open and easily accessible system. In the following, we present the main features of the new system:

- **web-based user interface:** All user interaction is done online via a browser. A web portal offers all functionality that is needed to handle a graph from uploading data, inspection of existing graphs and search for others and, finally, downloading a found graph.
- **automated registration (email opt-in):** Registration is performed online using a registration form, which is handled automatically. The system sends immediately a registration link via email after submitting the form.
- **limited rights management:** There are no groups of users that define rights for small circles of users. Licenses for graphs limiting usage are not encouraged in our open approach, thus, if necessary, a license can be attached to single selected graphs.
- **open access to all graphs after registration:** After confirming registration by fulfilling the email opt-in process, a user has access to all graphs and can initiate queries without restrictions.
- **categorization of graphs (e.g., fields of application):** For search, graphs can be assigned to the field(s) of application that they derive from. This enables researchers from different fields to use GraphArchive as a common platform.
- **automatic graph analysis after upload (for graphs with < 100.000 nodes):** After upload, graphs are analyzed in order to provide consistent data. The consistency is very important for search queries on graph properties. Also, automatic analysis might reveal more properties than manual assignment.
- **search for graphs using multiple criteria:** Search queries can be executed on multiple parameters, among them are graph properties, categories, author, name and upload date. Also, parameters can be combined to further narrow down the result set.
- **support of user-defined tags attachable to graphs:** Users can define individual tags to identify special attributes of graph(s). All user-defined tags are made fully searchable.
- **support of grouping of graphs:** Graphs can be grouped to mark their relation, e.g., graphs that stem from a specific test data set. Graphs that are uploaded as a single zip file are also grouped using one distinct tag, e.g. the zip file's name.
- **support of graph layouts to create visualizations (images) of graphs:** An image of a graph is valuable if a user quickly wants to inspect visually a graph's properties. Layouts are computed automatically in the background and also can be changed after upload.
- **support for creating comments and references:** Commenting on graphs might initiate discussions on certain graphs. Also, descriptions can be stored as comments. References can be assigned to a graph in order to highlight publications and/or websites that made use of any kind of this graph.
- **unique links to a graph (URI) for referencing in publications:** A URI allows for a permanent reference in publications. Stating the URI in a publication

enables the reader to quickly find the used graph data set.

- **'multiview' for comparing multiple graphs on a single page:** For quickly comparing multiple graphs at a time, we support the presentation of various graphs at a time. Properties are displayed for all graphs. Boolean properties, e.g., directed/undirected, are presented visually on a scale (property can be fulfilled by (a) no graph, (b) a subset of the displayed graphs or (c) all graphs).
- **support of various graph file formats:** Since it is impossible to decide on a specific file format when supporting many fields of applications, we aim at providing support for as many formats as possible. Our system allows to add further formats in the future.
- **support of graph file format conversion for downloads:** For downloading graphs, a user can choose the format that fits best to his/her work environment. We provide cross conversion (the users can select any supported format and the system starts the conversion automatically).
- **support of zipped files for import/export of multiple graphs:** When handling a test data set of graphs, we allow to upload/download several graphs at a time using zip compression. In an upload process, each file in the compressed file can optionally be processed individually (for properties analysis and layout computation). When downloading several files, the system automatically creates a compressed file containing all selected graphs.
- **graph authorship management featuring *my graphs* for graph authors:** An author of graphs can easily manage his/her graphs using the view 'my graphs' where inspections and actions, e.g., deletions of multiple graphs, are quickly accessible because the author rights in this view are limited to the current user.
- **guest access for non-registered users:** If a user wants to check a specific graph, he/she can access a detailed view on the graph using the URI. All properties and attributes of the graph are made visible entering via the guest account. However, actions, e.g., commenting, changing properties, download, are disabled in this view.

3.2 Architecture

Our system architecture is built similar to a common Web-browser application including a couple of necessary extensions for handling of graphs. The application is written in PHP5 ² using Apache2 ³ for online presentation. For graph analysis and layout computation, we make use of the java graph library yFiles ⁴, which is handled in the background via PHP/JAVA Bridge ⁵. Data storage is provided by a PostgreSQL database ⁶. A schema of the system architecture is depicted in Figure 3.

3.3 Presentation of selected key features:

Rights management In the former approach, many graphs were not public by default. Thus, rights handling was a major issue. A hierarchy of rights was integrated,

²see project homepage: <http://www.php.net>, last accessed 2011-07-12

³see project homepage: <http://www.apache.org>, last accessed 2011-07-12

⁴developed and maintained by yWorks GmbH: <http://www.yworks.com>, last accessed 2011-07-29

⁵Online source to the SourceForge project available at:

<http://php-java-bridge.sourceforge.net/pjb/index.php>, last accessed 2011-07-12

⁶see project homepage: <http://www.postgresql.org/>, last accessed 2011-07-12

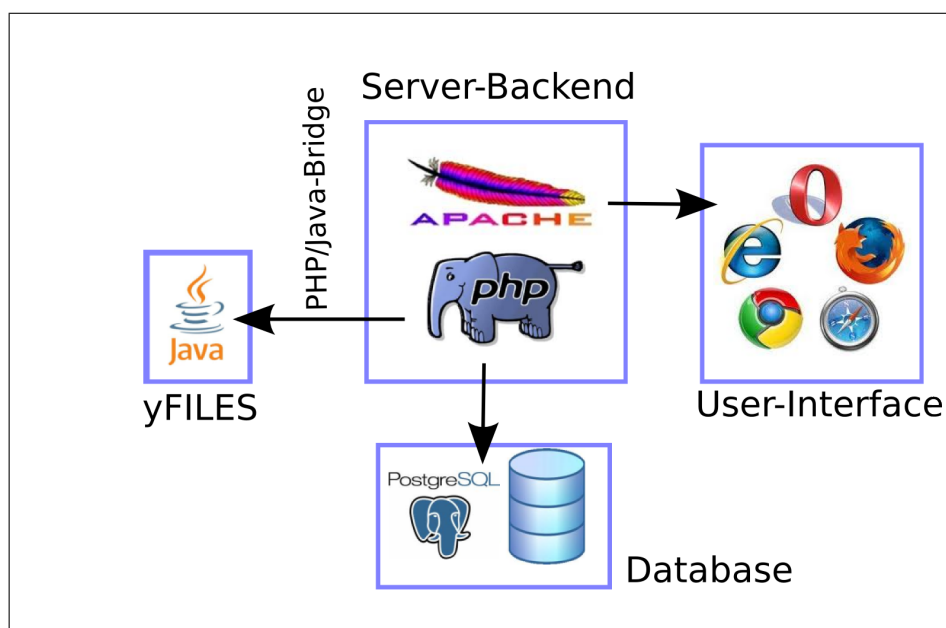


Figure 3: Architecture of GraphArchive.

involving group rights and user rights for a graph. Download of a graph was allowed only if a user was granted the appropriate rights. If a user was not specifically granted the right for a graph, and the user was also not assigned to a group that had access to the graph, access and download of the graph was denied.

Our system pushes rights for access and downloads towards an open-access approach. After registration, all graphs are accessible and may be inspected, e.g., to analyze graph properties. When uploading a graph to GraphArchive, the author of a graph needs to confirm that he/she holds/obtained the rights to publish the graph. Also, the author agrees that the graph is shared in GraphArchive.

Since some graphs come with usage limitations and/or demands, it is possible to integrate a license to a graph when uploading as graph author. In this case, a later graph download demands confirmation of the license before file transfer is started. Due to the fact that GraphArchive is intended to be an open-access platform we restrict licenses to be assignable on a 'per-graph' base only.

Tagging For assigning properties to graphs, we use the principle of tags. Tags consist of a (key, value) pair, whose value is of type boolean, integer or double. Since the principle of tags is general, we can use it for several purposes:

Tags allow ...

- ... graph categorization, for instance assignment to field(s) of application (e.g., metabolic networks, electrical circuit, class diagrams and many others).
- ... assigning of graph properties, e.g., acyclic, directed or degree.
- ... graph grouping, e.g., graphs of one group have the same tag set.
- ... user-defined properties, e.g., user can create new tags and assign them to graphs.

Graph analysis and visualization Graph analysis can be a tedious task when done manually, it even may prevent users to upload graphs. However, graph properties are of essential importance when it comes to search for specific properties. To provide a valuable query mechanism, an archive depends on sufficiently set properties. To free the user from this task, we perform an automatic graph analysis on the graphs after upload. Graphs are analyzed for a pre-defined default set of properties. The set comprises the following:

node count	edge count	biconnected	bipartite
connected	cyclic	forest	multiple edge free
planar	rooted tree	self loop free	simple
strongly connected	tree	component count	minimum degree
maximum degree	average degree	median degree	

In parallel to the graph analysis, layouts of the graph are created and stored as images for later presentation on the graph's detail page. The images are created using a standard layout algorithm provided in yFiles, a Java library to work with graphs. The default layout is computed by a spring layout algorithm [6]. The library is integrated into the system with the help of the PHP/Java Bridge, which allows to connect JAVA classes to PHP scripts. The layout algorithm can be changed later on the graph detail page where new layouts can be created (e.g. orthogonal/hierarchical/spring/circular layout).

Since the computation for some properties and layouts is very time consuming, we perform a complete analysis only for graphs with < 100.000 nodes. The analysis is done in the background to not disturb the user while browsing in GraphArchive; this also holds for the computation of layouts. For the analysis, we use data structures and algorithms provided by the yFiles library.

Referencing graphs Often, researchers use sets of graphs to perform experiments. In order to render such experiments repeatable for other researchers it is preferable that these data sets are referenced in the corresponding publication. To allow this, we introduced the possibility to add references to a graph in our system. A reference consists of a description and an optional link to the relating publication. For each graph, multiple references are possible. The references are also searchable to be found easily via the main page.

Additionally, we create a unique description for each graph (URI). Given the URI of a graph, it can be reached online by adding the URL of our system, e.g.,

```
http://algo.inf.uni-tuebingen.de/forschung/graphdb/graphs/showgraph.php?
graph=bdc3639a
```

where *bdc3639a* represents a graph's URI. URIs are considered static such that they are not supposed to undergo changes even in case that the underlying system is modified heavily. Also, given the URI, one can view the corresponding graph as a non-registered user. Thus, readers of a publication given a URI of our GraphArchive can have a look at the graph. This is provided by our guest access. The guest access is entered by browsing to a URL as described above. Major differences between a guest and registered users are: guests can only view a single graph, they have no access to the main page; guests are not allowed to perform actions, e.g., search, upload or download.

Search for graphs The query mechanism of our system allows to search for graphs by selecting and specifying query parameters. The parameters can be combined. A picture of the search form is given in Figure 4. Main search criteria are:

- **graph properties:** when searching for a graph property, e.g., number of nodes, a distinct value is supported as well as a given range or upper/lower bound, e.g., graphs with more that 10 nodes but less than 100 nodes.
- **graph categories:** the categories are stored using tags. Thus, graphs with a specific field of application carry the name of their field as an attributed keyword.
- **author/graph name:** search for graphs uploaded by a special user or named by a specific name, e.g., *Metro map*.
- **upload date:** search for graphs according to an upload date, we provide search for specific dates but also for periods of dates, if the exact date is unknown.
- **additional keywords (tags):** user-defined keywords are treated as tags and are searchable by selecting the appropriate keyword in the search form.
- **references:** graphs can also be found by a lookup according to the references that are connected to them or their specific URI.

Figure 4: Query form of the free search: retrieving graphs by giving a range of upload dates is also among the possible search queries.

File formats In the field of Graph Drawing, there are numerous tools with very different file formats. The reasons for the usage of a distinct file format can be multifaceted, e.g., text graph format (tgf) can be favoured for its simplicity.

The reasons why a file format is preferred over others depends on the field of application. One aim of GraphArchive is to become a central graph repository for all domains of interest. Therefore, we do not favor one of the file formats but try to achieve support of as many file formats as possible. We are convinced that limitation to a few file formats might prevent people to use the GraphArchive. We also support conversion between our supported formats when the user wants to download a graph. As we continuously improve our approach, we are open for source code contributions to enlarge our set of supported file formats. Currently, the following formats are supported:

Description	Abbreviation
Text graph file format	.tgf
GraphML	.graphml
Compressed GraphML	.graphmlz
Graph Markup Language	.gml
Graph Markup Language (XML)	.xgml
Y Graph Format	.ygf

If a graph is uploaded in an unknown format, it is left unprocessed and stored as a binary file. However, graph analysis and layout computation as well as conversion for export is not possible in this case.

For ease of import/export and the handling of graph libraries with numerous graphs, we also support zip files. When uploading a zip file, the compressed file is optionally extracted and each contained file is processed individually as a graph file. Downloading several graphs (without format conversion) is facilitated by compressing them using zip compression before download.

4 Presentation of the new system

In this section, we want to give an impression of the design and online appearance of GraphArchive by taking a virtual walk through a typical use case. The reader is encouraged to make a tour on his own by browsing to the current GraphArchive via our institute entry page:

<http://algo.inf.uni-tuebingen.de/?site=forschung/graphdb/grapharchive>

In Figure 5 (attached at the end of this document), the main page is depicted, graphs are displayed in a table. The table is sortable ascending/descending in any of the columns. For a quick overview on the main page, the user may show detailed information of the displayed graphs, where key facts of the graphs are given, see Figure 6. The details single graph page is shown in Figure 7 including an image visualization and the attributes of the graph. On the details page, comments or references can be updated and users may add additional tags. The default set of tags that is analyzed automatically and the insertion for user-defined keywords is presented in Figure 8.

The upload form provides multiple features for specifying information on the graph to be uploaded, see Figure 9. Additional to references, comments, field(s) of application, the author has to confirm his/her right to upload and share the graph. Here, licenses may be added to the graph. In case of multiple graph upload, keywords, comments and references can be set for all uploaded graphs at a time.

The multiview feature is presented in Figure 10. Details pages of multiple graphs are presented in a combined fashion on a single page. Properties are either marked grey or bold black depending on whether the property is matched by all graphs or only a subset. Comments and references are handled similarly; changes performed on a multiview page are passed to all displayed graphs. A download that is initialized on a multiview page starts the creation of a zip file that contains all displayed graphs.

The limited view of a guest access on a graph using the graph's unique URI is depicted in Figure 11. Note that actions such as download or layout form are disabled for guests.

5 Summary

In this report, we presented our system GraphArchive. It enables the community to exchange graphs in a central place. Also, it provides a data store for archiving graphs, e.g., graphs that are used in test suites.

We discussed the reasons and benefits of a central graph repository. This need was already identified earlier and led to the now discontinued system GraphDB. Further, we analyzed the reasons that might have led to the rather hesitant usage of the GraphDB. With our new approach GraphArchive we tackle the identified weak spots of GraphDB. The new application is developed as an online tool supporting and exploiting modern web technologies. The portal is fully accessible via a common browser. The goal was to provide an easy-to-use and powerful yet simple graph data platform.

GraphArchive enables interested researchers to find, share and store graphs of various fields of applications, e.g., social networks, road networks, class diagrams or metabolic networks. Additionally, it provides a persistency mechanism, which allows for storing data sets and permanently referencing them by a URI. This allows to reference data sets in future publications, which makes experiments more transparent, repeatable and thus, more reliable.

Also, the automated analysis of graphs increases usability and data consistency. Integrated layout computations provide visualizations for quickly grasping mental maps of graphs.


As a matter of course, development of the tool is not completed. In the future, we will keep improving the running system and adding new features to it, e.g., allowing an image gallery for a graph to integrate different visualizations uploaded by users. We will use <http://www.graph-archive.org> as a platform to post news and development progress of our system. We hope that our system succeeds in providing a helpful service and is being promoted and supported by the community to establish a central place to go for sharing graphs. The rise and fall of the system depends on user acceptance and its regular usage.

References

- [1] T. Davis and Y. Hu. The University of Florida sparse matrix collection. <http://www.cise.ufl.edu/research/sparse/matrices/>, May 2011.
- [2] 10th DIMACS implementation challenge - graph partitioning and graph clustering. <http://www.cc.gatech.edu/dimacs10/downloads.shtml>, May 2011.
- [3] J. J. Garrett. Ajax: A new approach to web applications. online: <http://adaptivepath.com/ideas/ajax-new-approach-web-applications>, February 2005. last accessed (2011-07-27).
- [4] D. Knuth. *The Stanford GraphBase*. ACM Press, 1994.
- [5] Matrix market. <http://math.nist.gov/MatrixMarket/>, National Institute of Standards and Technology, May 2011.
- [6] R. Tamassia, G. DiBattista, P. Eades, and I. Tollis. *Graph Drawing*. Prentice Hall, 1999.

GraphArchive

Main Menu



Logout

Account administration

My graphs

Search for graphs: +

Administration

Upload graphs

All graphs currently in the database:

[show detailed infos](#)

Graphs per page: [5](#) [10](#) [20](#) [50](#)

Page 1 of 3 | 2 | 3 | Next > | Last >>

<input type="checkbox"/>	ID	Name	Author	Upload date	URI
<input type="checkbox"/>	384	Krugsche Testgraphen	Robert Krug	01.02.2011	cd4ae111
<input type="checkbox"/>	385	kotter-graphs	Stephan Kottler	01.02.2011	7ac72019
<input type="checkbox"/>	386	Krugsche Testgraphen	Robert Krug	01.02.2011	a1e16118
<input type="checkbox"/>	387	kotter-graphs	Stephan Kottler	01.02.2011	14fca21e
<input type="checkbox"/>	388	graph3	Stephan Kottler	01.02.2011	a941ed24
<input type="checkbox"/>	389	graph4-kottlers	Stephan Kottler	01.02.2011	1e5c2c20
<input type="checkbox"/>	390	graph3	Robert Krug	01.02.2011	cd6dfcd0
<input type="checkbox"/>	391	graph4-krug	Robert Krug	01.02.2011	7a703dd4
<input type="checkbox"/>	405	3D-Nodes	Philip Efinger	08.02.2011	bdc3639a
<input type="checkbox"/>	455	Edge Bundling David Auber	Philip Efinger	11.05.2011	c8235e0d

[download selected](#) | [view selected](#)
[download all](#) (Warning: This may take a while!)

Page 1 of 3 | 2 | 3 | Next > | Last >>






Figure 5: Screenshot of the GraphArchive main page.

All graphs currently in the database:

[hide detailed infos](#)

Graphs per page: [5](#) [10](#) [20](#) [50](#)

<< First | < Prev | 1 | Page 2 of 5 | 3 | 4 | 5 | Next > | Last >>

<input type="checkbox"/>	ID	Name	Author	Upload date	URI
<input type="checkbox"/>	389	graph4-kottlers	Stephan Kottler	01.02.2011	1e5c2c20
			<ul style="list-style-type: none"> node count: 300 edge count: 500 not connected 	<ul style="list-style-type: none"> cyclic minimum degree: 0 maximum degree: 8 	<ul style="list-style-type: none"> component count: 13 multiple edge free not planar
<input type="checkbox"/>	390	graph3	Robert Krug	01.02.2011	cd6dfcd0
			<ul style="list-style-type: none"> node count: 110 edge count: 150 not connected 	<ul style="list-style-type: none"> cyclic minimum degree: 0 maximum degree: 7 	<ul style="list-style-type: none"> component count: 5 multiple edge free not planar
<input type="checkbox"/>	391	graph4-krug	Robert Krug	01.02.2011	7a703dd4
			<ul style="list-style-type: none"> node count: 300 edge count: 500 not connected 	<ul style="list-style-type: none"> cyclic minimum degree: 0 maximum degree: 8 	<ul style="list-style-type: none"> component count: 13 multiple edge free not planar
<input type="checkbox"/>	405	3D-Nodes	Philip Efinger	08.02.2011	bdc3639a
			<ul style="list-style-type: none"> node count: 6 edge count: 6 connected 	<ul style="list-style-type: none"> cyclic minimum degree: 1 maximum degree: 3 	<ul style="list-style-type: none"> component count: 1 multiple edge free planar
<input type="checkbox"/>	455	Edge Bundling David Auber	Philip Efinger	11.05.2011	c8235e0d
			<ul style="list-style-type: none"> node count: 1715 edge count: 6529 not connected 	<ul style="list-style-type: none"> cyclic minimum degree: 1 maximum degree: 231 	<ul style="list-style-type: none"> component count: 28 multiple edge free not planar

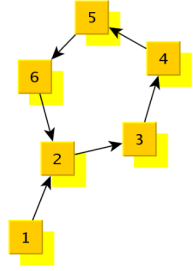
[download selected](#) | [view selected](#)
[download all](#) (Warning: This may take a while!)

<< First | < Prev | 1 | Page 2 of 5 | 3 | 4 | 5 | Next > | Last >>

Figure 6: Expanded view of the main page with quick facts of graphs.

Details of graph with ID 405

[new layout](#)



[Download graph](#)
Choose format:

Name
3D-Nodes

Author
Philip Effinger

Upload date
08. February 2011

Appears in

- Journal for Graphs
- (add new reference...)

URI
<https://kandinsky.informatik.uni-tuebingen.de/forschung/graphdb/graphs/showgraph.php?graph=bd:3639a>

Comments

- kann man die Schatten der Knoten sehen? posted on 08. February 2011 by you
- ja, super! posted on 08. February 2011 by you
- das möchte ich lesen, nochmal! posted on 08. February 2011 by you
- (add new comment...)

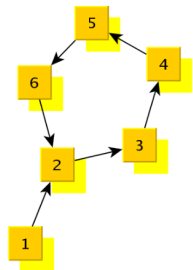
Keywords

- node count:
- edge count:
- biconnected:
- bipartite:
- connected:
- cyclic:
- forest:

Figure 7: **Graph detailed page**; downloads can be initialized here and layout calculation can be selected. On the right, attributes are listed; at the bottom, user-defined tags can be added, as shown in Figure 8.

Details of graph with ID 405

[new layout](#)



[Download graph](#)

Keywords

- node count:
- edge count:
- biconnected:
- bipartite:
- connected:
- cyclic:
- forest:
- multiple edge free:
- planar:
- rooted tree:
- self loop free:
- simple:
- strongly connected:
- tree:
- component count:
- minimum degree:
- maximum degree:
- average degree:
- median degree:

Add keyword manually:

Name of keyword:

Type:

Value:

[add keyword](#)

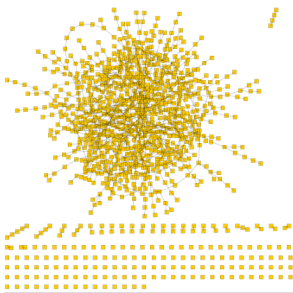
Figure 8: Complete display of **default graph tags**, the user-defined tags can be set at the bottom and will be added immediately to the tags.

Figure 9: **Upload page:** (multiple) files can be selected and various attributes can be assigned to the uploading graph: name, comment, publication or website references, up to 3 fields of applications are available, individual properties can be determined and licenses might be attached. Before upload, a user has to ensure that he agrees to share the graph and has the right to do so.

Keywords	Count
node count:	9 110 300 6 10
edge count:	9 150 500 6 11
biconnected:	<input type="checkbox"/>
bipartite:	<input type="checkbox"/>
connected:	<input checked="" type="checkbox"/>
cyclic:	<input checked="" type="checkbox"/>

Figure 10: **MultiView:** view of multiple graphs at a time. Common properties are displayed in bold black font colors; properties that are matched by a subset are marked in grey. On mouse-over, the IDs are shown of the graph(s) that match the selected properties.

Details of graph with ID 384



FERDINAND KARLS
 UNIVERSITÄT
 TUBINGEN

Name	Author
Krugsche Testgraphen	Robert Krug
Upload date	Appears in
01. February 2011	• Google
URI	
http://kandinsky.informatk.uni-tuebingen.de/forschung/graphdb/graphs/showgraph.php?graph=cddae111	
Comments	
<ul style="list-style-type: none"> • Testcase fuer die GraphDB posted on 01. February 2011 by Robert Krug • Das sind aber dicke Kanten ;) posted on 01. February 2011 by Stephan Kottler • zwei tolle Graphen posted on 01. February 2011 by Robert Krug • wirklich toller Graph posted on 01. February 2011 by Robert Krug 	
Keywords	
• node count:	1000
• edge count:	1000
• biconnected:	<input type="checkbox"/>
• bipartite:	<input type="checkbox"/>
• connected:	<input type="checkbox"/>
• cyclic:	<input checked="" type="checkbox"/>
• forest:	<input type="checkbox"/>

Figure 11: This display is shown when a graph is accessed using the **guest view** via a link containing the graph's URI. Information on the graph is displayed in full; performing changes or actions, e.g. downloads, is deactivated.