DALIGRES – a Graph Query Tool to Unravel the Life Science Database Maze

Matthias Lange, Matthias Klapperstuck, Karl Spies, Uwe Scholz
Leibniz Institute of Plant Genetics and Crop Plant Research, Gatersleben, Germany

Abstract: *The knowledge found in the Web, in particular in life-science databases, are a major resource in molecular biology dry lab research. To support the interpretation of measured facts from the wet lab, like transcriptional activity, DNA sequences or protein masses, the functional context has to be determined. This is done using dry lab research, which is dependent on existing biological knowledge. In general, biological knowledge is worldwide represented in a network of databases.*

In this paper we present methods and tools for knowledge extraction from life science databases, which prevents the scientists from expensive web navigation and allows a large scale automated database exploration. Using computed data linkage graphs (DLG), we present a query language to extract interlinked biological knowledge.

Based on the software tool DALIGRES which enables the joining of millions of data points over hundreds of databases, we computed data networks of protein knowledge, which interconnect metabolic, disease, enzyme and gene function data. Availability: The data linkage graphs (XML or TGF format), the integrated database schema (GML or GRAPH-ML) and the DALIGRES tool can be downloaded from the following URL: http://pgrc.ipk-gatersleben.de/dlg/

Keywords: *bioinformatics, database networks, database integration, graph queries*

1. MOTIVATION

High throughput biotechnologies, like cDNA or oligo arrays, sequencing projects towards proteomics and metabolomics techniques, produce a massive amount of data from measured molecular facts. Those facts could be gene expression, polypeptide concentration or EST sequences. In order to support the data interpretation and to discover the functional role of the measured molecule, the relationship to existing biological knowledge presented in worldwide distributed databases, has to be explored. To present the life science data to desktop computers, the World Wide Web (WWW) is the most popular medium. Studies show, that Web based, in silico, research is a major task in life science [1], [2]. In [3] a study was presented, whereupon 37% of all scientists use over 80% of their time for Web investigations. In detail, scientists facilitate database embedded tools for linking wet-lab data to database knowledge in an explorative way. Methods like sequence similarities e. g. BLAST [4] or text similarities queries like Entrez text queries [5] or GoPubMed [6] are the database search operations, which are commonly used in life science dry labs. In order to extract the Web content, the dry lab scientists use the following workflow:

1) search the databases using appropriate tools
2) hyper link tracking, e.g. use the database cross references

If the scientist has programming skills, this work flows can be scripted using plain popular programming languages like PERL or modeled using work flow systems like TAVERNA [7].

One limiting drawback in this scenario is the data and tool distribution. Nowadays, we know about hundreds of databases, with different but also overlapping content [8]. Efforts for integrating them were made in several projects using several methods. Because the majority is focused on particular analytic scenarios and applications, none of them provided a holistic, uninterpreted view to the world wide interlinked knowledge. Popular techniques, like data warehouses, multi database query languages or database mediators solved the problem of database heterogeneities and provided powerful query interfaces and helpful tools.

But the original character of life science databases as a highly dynamic and flexible network of knowledge is not supported. Because the nature of the knowledge stored in the WWW is to be a network, we need to provide an efficient method for the extraction of data and in future knowledge networks [9]. The navigation and benefits of exploiting the data networks was motivated in [10], [11]. The two systems BioNavigation and BioGuide support the scientist in exploring the content of integrated life science databases. The core idea is a graph based browsing and query building on top of integrated databases.

In this paper, we present a recently developed method and software for the efficient extraction and querying of data networks from distributed, heterogeneous life science databases. Surprisingly comprehensive knowledge networks were computed and are available for public access. First applications of these networks were successfully used for a high throughput functional classification of EST towards an automated pipeline for the provisioning of controlled annotation of plant gene expression arrays. This and further applications are presented.

2. METHOD

To meet the requirement of the given flexibility of knowledge networks and to enable an efficient but also lightweight processing of very large volume of integrated data, the concept of Data Linkage Graphs (DLG) and an algorithm to efficient compute them has been presented in [12]. Subsequently, a methodical overview is presented.

The concept is the representation of integrated databases as a network of key attributes and its values. Key attributes are those one which define unique identifier like EMBL sequence accessions [13], Gene Ontology (GO) terms [14] or EC numbers in [15]. Life science data in the Web is structured. The modeled data structure itself comprises important meta data like attributes and attribute groups, called entities. For example, the EMBL Sequence database present the Web data structured into the entities:

- General Information
- Description
- References
- Features
- Sequence

Those entities comprise attributes, like the EMBL-entity Description:

- Description
- Keywords
- Organism
- Organism Classification

All entities and its attributes form one database entry. In order to consistent access data, database entries should be uniquely identified. In EMBL, the attribute accession is used. We call such attributes key attributes. This metadata is the used for the Data Linkage Graphs. We define schema nodes S as a set of pairs S = (E, A), whereas E is the entity and A is the key attribute. In order to construct networks, edges have to be defined. In the matter of fact, life science database are strongly interconnected. Studies assume a connection rate of at least 80% in life science databases [9]. This is manifested by a number of hyperlinks in the Web presentation of the databases, references in XML documents or in primary key / foreign key pairs in relational data structures. We use these relationships and extract out of them the edges R among the schema nodes S. Formally, R is a subset of all possible combinations of two elements from S:

$$R \subseteq S \times S$$

In order to determine R, two kinds of bidirectional relations among key attributes are valid [12]:

Rlink : Inter-entity relationships, which are modeled as references between data entries. Those could be represented by: HTML hyper links, primary key/foreign key, object references etc.

Rfunc: Inner-entity relationships, which are modeled as functional dependency among attributes in one entity. Those could be represented by: co-occurrence in one HTML page, children of the same XML parent item or modeling as attributes of one relational table.

With this rules Data Schema Graphs (DSG) may be constructed, with: DSG =(S, R). Consequently, the DSG comprises a database network at schema level. The next idea is the extension of the schema graph toward the instantiated data networks. Thus, we have to consider the attribute values, the stored data in the databases. In the context of the above example, we have a Gene Ontology term id GO:0005975, which identifies the term carbohydrate metabolism or the EMBL accession BAY074321, which identifies a sequence, coding for alpha-galactosidase. If we include this attribute values in our model and construct data networks, we will end up with graphs of interacting database entries. We extend the pair S to a triple, with one additional element V. Thereby, V is an element of the data domain dom(S) of an attribute, e.g. all valid EMBL accessions or Gene Ontology term identifier. We define data nodes SV as a set of a triple:

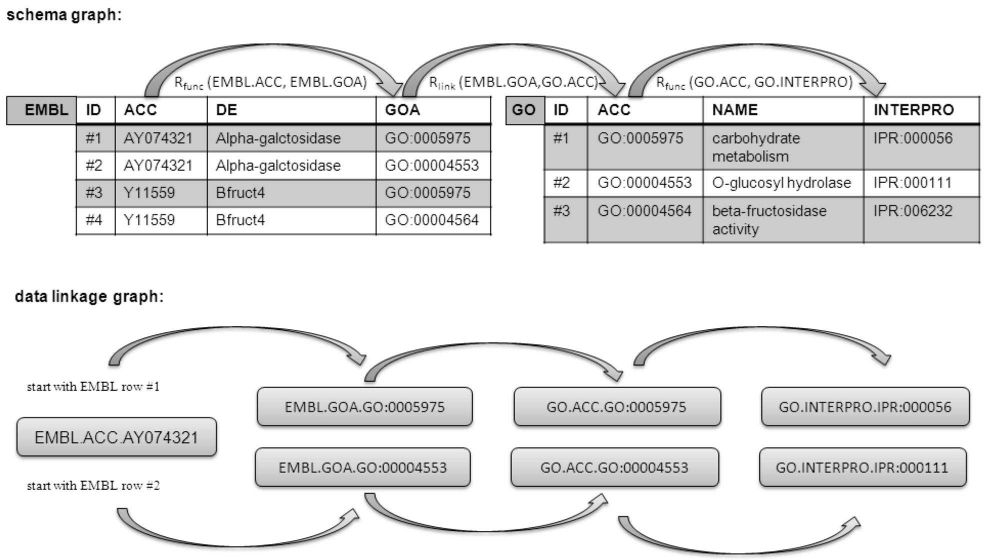
$$SV = (E, A, V) \mid V \in dom(S)$$

If we imply the existence of operation to check the identity of key attribute values in A, we can define a relationship ⇋ among two data nodes (sv_1, sv_2 ∈ SV), if one of the subsequent conditions is fulfilled:

1) the schema nodes are in a valid relation (∈ Rlink v Rfunc) and the attribute values are identical, or

2) the database table or Web page.

To illustrate these types of relationships, the relationships between two data sources are visualized in figure 1.



EMBL	ID	ACC	DE	GOA
	#1	AY074321	Alpha-galctosidase	GO:0005975
	#2	AY074321	Alpha-galctosidase	GO:00004553
	#3	Y11559	Bfruct4	GO:0005975
	#4	Y11559	Bfruct4	GO:00004564

GO	ID	ACC	NAME	INTERPRO
	#1	GO:0005975	carbohydrate metabolism	IPR:000056
	#2	GO:00004553	O-glucosyl hydrolase	IPR:000111
	#3	GO:00004564	beta-fructosidase activity	IPR:006232

Fig. 1 example for the relationships in schema and data linkage graphs

In this example, we simplified the EMBL and the Gene Ontology (GO) database to two tables comprising four and three rows. Each comprises three attributes in one entity. With known inter-attribute relationships (e. g. derived from hyperlinks among EMBL and GO), our example includes three data schema relations:

$$R_{func_1}=(EMBL.ACC,EMBL.GOA)$$
$$R_{func_2}=(GO.ACC,GO.INTERPRO)$$
$$R_{link_1} = (EMBL.GOA,GO.ACC)$$

On the basis of the above defined relation $\leftrightharpoons$ we can compute a set of data node pairs:

$$RV \in SV \times SV$$

such that

$$(sv_1, sv_2) \in RV \wedge sv_1 \leftrightharpoons sv_2$$

Finally, we are able to describe networks of data nodes as set of graphs with the structure DLG =(SV,RV). In figure 1. this concepts are visualized.

Using DLGs in combination with an efficient graph query engine, a general concept for a novel flexible, lightweight database integration becomes possible. Flexible means no schema or data integration, with all the problems needs to be solved. Lightweight, because only the key attributes values have to be extracted physically, parsed and downloaded respectively. In the next section we describe the concrete construction of schema networks and the computation of data networks, in form of data linkage graphs (DLGs).

3. COMPUTATION OF DATA NETWORKS

In order to use the described concept for joining millions of data points in over hundreds of databases, we designed and implemented an efficient algorithm for the computation of DLGs. The prerequisites are (1) a graph of schema nodes and (2) for all schema nodes, the key attribute values.

3.1. Data import

To meet the first requirement, we designed a protein knowledge network, which interconnects metabolic, disease, enzyme and gene functional data. In this context, we merged the data schema of the public databases UNIPROT (UP), Gene Ontology, OMIM, BRENDA and KEGG and derived a graph of schema nodes. This schema graph, available at http://pgrc.ipk-gatersleben.de/dlg, comprises 318 nodes and 234 edges, which cover the majority of the currently available protein related metadata. In an ongoing project, methods are developed to construct schema graphs automatically. This will be realized based on public available life science database repositories, like the NAR Database Categories List, pattern recognition in life science database WWW-presentation and the tracking of their hyperlinks.

The retrieval of the key attribute values was realized by a combination of an inhouse developed parser, the Sequence Retrieval System (SRS) [16] and a database mediator called BioDataServer (BDS) [17]. In this way, a file of key values for each schema node was generated. The results are 318 files comprising around 5.6*107 key values. Hereby, we observed a 10-fold redundancy. This potential for high compression rate motivates to the computation of a hash table, mapping each key value to an integer. If we assume a 64-bit coded integer, a 10-character identifier from the Gene Ontology can be stored in 8 bytes. This saves storage capacity in a way, so that is it possible now to represent the data linkage graphs in modern computer main memory.

3.2. Data network computation

The standard way for the computation of transitive entity relations are index supported joins. Such queries are commonly used to find data relationships in integrated databases. Examples of such queries are those, mapping sequence data to the predicted functional role of the coded gene. The common established strategy is to import relevant data into a data warehouse and execute queries. Because of the complexity of such queries, the feasible number of tables and tuples in such joins is limited.

The consequence is the limitation to use case specific data warehouses, which disables comprehensive database integration as proposed in [18]. Thus, there is a complexity problem for comprehensive analytical data queries in huge data warehouses. Even state of the art database management systems show pleasable response time for joins only for a limited number of data, tables or databases. More comprehensive queries would result in days of waiting or even system failures. To avoid those problems and to improve the implementation, we used main memory based data structures and algorithms. The nodes in our data networks show high redundancy and a compression rate of factor 10 can be achieved. Thus, we were able to represent all nodes by $5.6*10^7$ 64-bit pointers to $5.4*10^8$ data nodes. This data structure took around 4 GByte for UNIPROT (UP), Gene Ontology, OMIM, BRENDA and KEGG. Using this data structure, we pre-compute all possible joins, the Data Linkage Graphs, in main memory. This is implemented in a C++ program, which loads all data nodes into main memory and computes all possible DLGs. We implemented the recursive breadth-first search (BFS) algorithm in combination with index structures like hash maps, B-trees and bit vectors using the C++ Standard Template Library (STL). The graph computation took 3 weeks on a 2GHz Opteron server, whereas 12 GByte of main memory and 2 CPUs were used.

The resulted DLGs where stored in a flat relational structure: (graphid, node depth, parent entity, parent attribute, child entity, child attribute, parent value, child value). The graphs are available in csv-format at the given URL. This DLGs may be easily imported into a relational database system. Furthermore, the representation in a hierarchical data structure, like XML, is ad-hoc realizable, but not yet used practically.

4. KNOWLEDGE LINKING

To illustrate the possibilities of knowledge linking, we present a typical uses case from the dry lab. A frequent task in biology is the functional gene annotation, which is often accomplished by finding similarities or patterns in known, experimentally annotated sequence data. One common application is, the mapping of coding sequences to an annotated, non-redundant polypeptide sequence database like NRPEP (ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nr.gz). The mapping is done by a sequence similarity search using BLAST. The BLAST hits provide database accession numbers, that link to the original databases and have to be extracted eighter manually or using parsers. Finally, all linked databases have to be inspected for getting relevant information for the input sequences.

In addition of this use case, we have to consider the following aspect. The datasets on functional aspects of sequences and proteins present in life science databases are mainly derived from literature or provided from the database annotation team in textual form in natural language. In contrast, the use of a controlled vocabulary (CV) is a necessary concept for making annotations computational comparable. Using CVs, automized phenotypic classification for a huge number of wet-lab data is possible without human interaction or interpretation of textual description of a protein function. There are several systems, implementing such kind of systematic knowledge representation: the MIPS Functional Catalog [19], the Gene Ontology Database [4], GeneQuiz [20], OMIM [21], KEGG [15] or MapMan [22]. All those databases where more or less manually constructed.

To meet the requirement of an automate execution of the above work flow, we present an application of the DLGs to map general functional sequence annotation to controlled vocabulary in a computational way. As visualized in figure 2, the basic idea of the functional categorization is

1) take BLAST hits for a number of EST, e. g. an EST library,
2) extract the database identifiers from the hit descriptions
3) find them in the data linkage graphs and
4) query the data link graphs to
5) find a path from the found database identifier to the interesting annotation database entity.

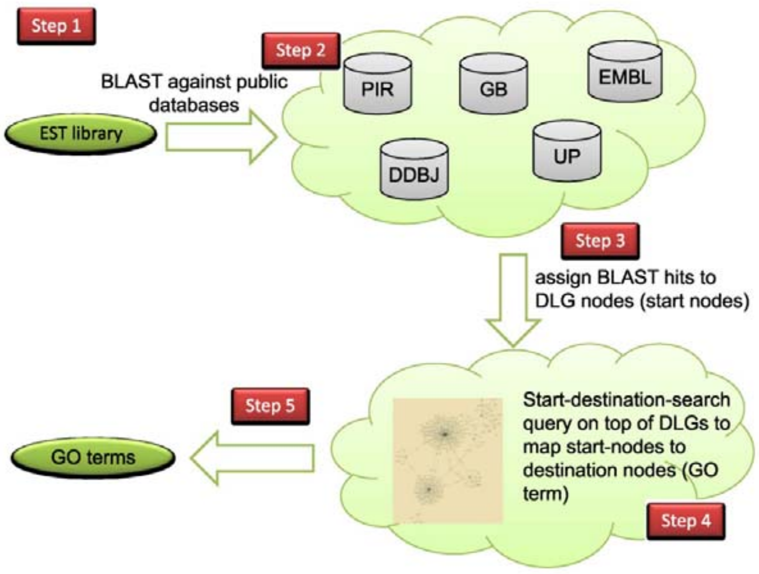

Step 1
Step 2
BLAST against public databases
EST library
PIR
GB
EMBL
DDBJ
UP
Step 3
assign BLAST hits to DLG nodes (start nodes)
Step 5
GO terms
Start-destination-search query on top of DLGs to map start-nodes to destination nodes (GO term)
Step 4

Fig. 2 workflow to assign ESTs to GO terms

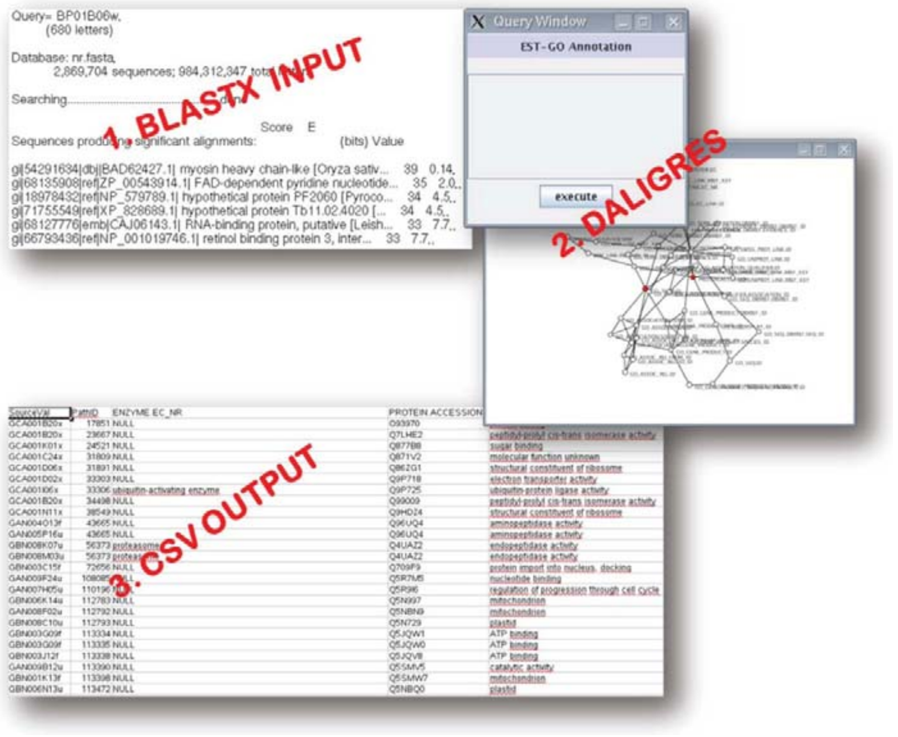
In an example scenario, visualized in 2, the work flow implement a mapping of ESTs to their Gene Ontology annotation. In order to implement the workflow, we loaded the graphs into an ORACLE 10g database table. Thereby, the former necessary join for the analysis of possible interconnections between life science data entries was replaced by an efficient start-destination-search in the graphs. This search retrieves all graphs that connect a given start-node to an end-node. In SQL this may expressed as a self join:

select g2.data from graphs g1, graphs g2 where g1.graphid = g2.graphid and g1.parent entity = 'EMBL' and g1.parent attribute = 'ACCESSION' and g2.child entity = 'GO TERM' and g2.child attribute = 'ACCESSION'

It is obvious, that the complexity of such a graph query is less than multiple joins in a data warehouse. Using indexes for the selection a and merge join, the complexity class of the above query is O(n). The prof is given in [12].

4.1. The BLAST mapping tool DALIGRES

In order to provide a tool support for the above discussed work flow, we implemented the tool DALIGRES. The objective is to map the hit descriptions from a BLAST output file to comparable functional terms, like KEGG EC numbers or GO terms. In the present implementation we provide Gene Ontology terms. For further application, more annotation source featuring controlled vocabulary can be targeted. The screen shot in figure 3 gives a snapshot of a mapping of 37004 BLASTX alignments, computed for 6342 EST subjects.



```
Query= BP01B06w.
        (680 letters)

Database: nr.fasta,
           2,869,704 sequences; 984,312,347 tot[illegible]

Searching...............................d[illegible]

                                                        Score    E
Sequences producing significant alignments:            (bits) Value

gi|54291634|dbj|BAD62427.1| myosin heavy chain-like [Oryza sativ...   39   0.14,
gi|68135908|ref|ZP_00543914.1| FAD-dependent pyridine nucleotide...   35   2.0,.
gi|18978432|ref|NP_579789.1| hypothetical protein PF2060 [Pyroco...   34   4.5,.
gi|71755549|ref|XP_828689.1| hypothetical protein Tb11.02.4020 [...   34   4.5,.
gi|68127776|emb|CAJ06143.1| RNA-binding protein, putative [Leish...   33   7.7,.
gi|66793436|ref|NP_001019746.1| retinol binding protein 3, inter...   33   7.7,.
```

SourceVal	PathID	ENZYME.EC_NR	PROTEIN.ACCESSION	
GCA001B20x	17851	NULL	O93970	[illegible]
GCA001B20x	23667	NULL	Q7LHE2	peptidyl-prolyl cis-trans isomerase activity
GCA001K01x	24521	NULL	Q877B8	sugar binding
GCA001C24x	31809	NULL	Q871V2	molecular function unknown
GCA001D06x	31891	NULL	Q86ZG1	structural constituent of ribosome
GCA001D02x	33303	NULL	Q9P718	electron transporter activity
GCA001I06x	33306	ubiquitin-activating enzyme	Q9P725	ubiquitin-protein ligase activity
GCA001B20x	34498	NULL	Q99009	peptidyl-prolyl cis-trans isomerase activity
GCA001N11x	38549	NULL	Q9HD24	structural constituent of ribosome
GAN004O13f	43665	NULL	Q96UQ4	aminopeptidase activity
GAN005P16u	43665	NULL	Q96UQ4	aminopeptidase activity
GBN008K07u	56373	proteasome	Q4UAZ2	endopeptidase activity
GBN008M03u	56373	proteas[illegible]	Q4UAZ2	endopeptidase activity
GBN003C15f	72656	NULL	Q709F9	protein import into nucleus, docking
GAN009F24u	108085	[illegible]	Q5R7M5	nucleotide binding
GAN007H05u	110196	[illegible]	Q5R9I6	regulation of progression through cell cycle
GBN006K14u	112783	NULL	Q5N997	mitochondrion
GAN008F02u	112792	NULL	Q5NBN9	mitochondrion
GBN008C10u	112793	NULL	Q5N729	plastid
GBN003G09f	113334	NULL	Q5JQW1	ATP binding
GBN003G09f	113335	NULL	Q5JQW0	ATP binding
GBN003J12f	113338	NULL	Q5JQV8	ATP binding
GAN009B12u	113390	NULL	Q5SMV5	catalytic activity
GBN001K13f	113398	NULL	Q5SMW7	mitochondrion
GBN006N13u	113472	NULL	Q5NBQ0	plastid

Fig. 3 work flow to assign ESTs to GO terms

This was executed in about 1 minute on a standard desktop PC. The software is based on a hierarchical query language. This language have been designed and implemented with the goal to extract mapping from existing DLGs. The idea of the language is to query DLG networks which fulfill certain criteria:

1) include all DLG-paths p start **from** data schema node α and reach **to** schema node ω

2) p includes a edge, which cross **over** defined schema nodes n_1 ,...n_x | $n_x \in$ p

3) checks if certain values **bind** on an intermediate schema node n_x construct existing data nodes in p

The main language elements are constructs to express three kind of node criteria: **from**, **to** and **over**. The query language is given informally by the following notation example:

```
Select
PROTEIN.ACCESSION,GO_TERM.ID,ENZYME.EC_NR
with
PROTEIN.ACCESSION[<BLAST hit database IDs>]
from
PROTEIN.ACCESSION
to
GO_TERM.ID
<!-- optional -->
over
ENZYME.EC_NR>EC{[<identifiers to be crossed>]}
```

The tool workflow for BLAST to GO mapping is implemented, based on the workflow given before in figure 2 as follows:

1) parse the BLAST result file to extract the database identifier (consider quality parameter like e-value, score, database)

2) map BLAST subjects (sequence IDs) to the mapped and filtered Gene Ontology terms

a) query all possible paths, which include the extracted database identifier and Gene Ontology database entries

b) optional filter the paths by evaluate if certain nodes are included in the path

3) write a CSV output file

This workflow has to be parameterized with the input and output files, and the DLG-query. All this parameter has to be specified in a configuration file. In the following example parameterization, the BLAST output file small IPK Hv_ESTs_vs_NCBI_NRPEP.blastx is specified as input and the file outputtest.csv as CSV-output. All those parameter specify the configuration of this specific ”BLAST” to ”GO” use case. The complete configuration of this example is given below:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<config>
<title name="EST-GO Annotation"/>
<graphsourcepath path="config/schema_graph.graphml"/>
<connections>
<link name="Oracle" refname="$ora">
<class name="de.klapperstueck.dbinterfaceOracle.DBRealOracleType" />
<param name="filename" value="config/dbconnect.ini" />
</link>
</connections>
<input name="DLGDB" dblink="$ora"/>
<output name="test">
<class name="de.klapperstueck.writer.MappingCSVWriter" />
<param name="filename" value="outputtest.csv" />
<param name="mapping:ENZYME.EC_NR" value="ENZYME_NAME.EC>NAME" />
<param name="mapping:GO_TERM.ID" value="GO_TERM.ID>NAME" />
<param name="inputlink" value="$ora" />
</output>
<query> select PROTEIN.ACCESSION,GO_TERM.ID,ENZYME.EC_NR with
PROTEIN.ACCESSION[$reader2] from PROTEIN.ACCESSION to GO_TERM.ID </query>
<readers>
<reader name="BlastXParser" refname="$reader2">
<class name="de.klapperstueck.reader.ReaderBLASTFile" />
<param name="filename" value="small_IPK_Hv_ESTs_vs_NCBI_NRPEP.blastx" />
<param name="mapping:emb" value="EMBL_LINK.SECONDARY_ID>ACCESSION" />
<param name="mapping:dbj" value="EMBL_LINK.SECONDARY_ID>ACCESSION" />
<param name="inputlink" value="$ora" />
<param name="usedatabase" value="sp,emb,dbj" />
<param name="maxexpect" value="1e-10" />
<param name="minscore" value="20" />
<param name="maxscore" value="510" />
</reader>
</readers>
</config>
```

4.2. Database network discovery

Beside the above mentioned use case, further analysis scenarios are interesting. This include knowledge discovery, by context specific joining of graphs, like building context networks of data on specific diseases, regulatory elements, protein data and gene variants. This can be featured by a multidimensional, structural graph analysis of the data networks, like centrality analysis for the identification of key elements of data networks. Furthermore, quality control of public databases is an important challenge in modern bioinformatics. Using DLGs, we can for instance extract all networks that include functional contradicting database entries connected to the same gene sequence.

For first draft studies some specific metabolic knowledge networks for plants where extracted (available in GML or GraphML format under the URL http://pgrc.ipk-gatersleben.de/dlg/). Here we merged DLG over shared enzymes (EC numbers) which are present in glycolysis metabolic pathway and shared proteins which are reported in Brassicae, Triticeae and Solanacea. Thus we extracted networks of knowledge to enzymes in barley which show catalytic activity in glycolysis. Figure 4 gives an impression of such a data network. A high resolution and comfortable version from this figure is available, using the provided graph data files in a graph viewer. The files and a link to a viewer are available using the URL previously given.

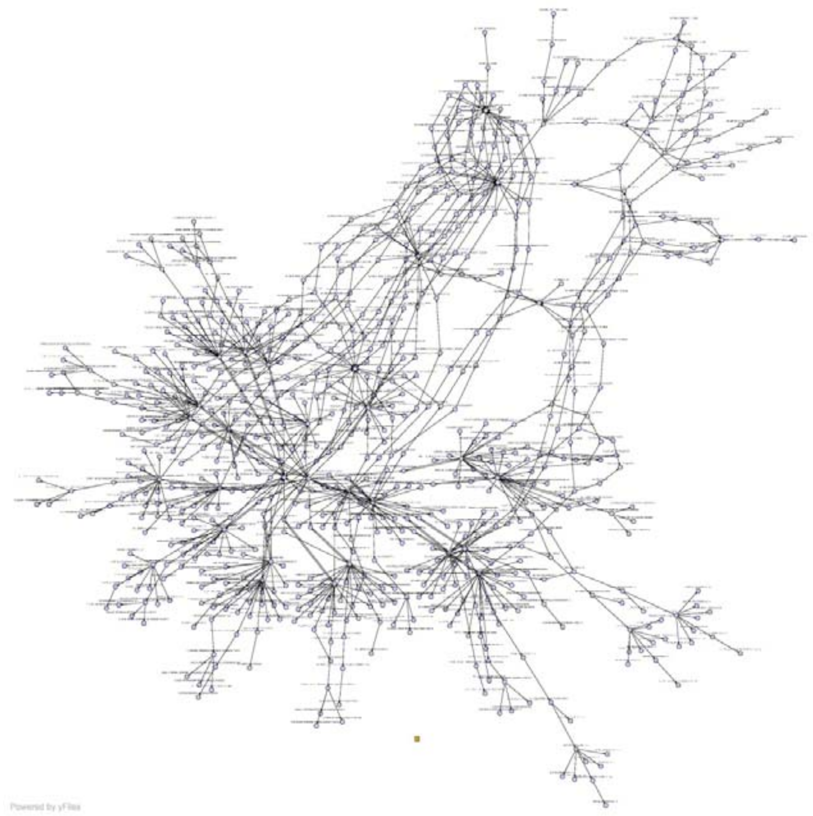


Fig. 4 data network of enzymes in barley showing catalytic activity in glycolysis (high-resolution image: http://pgrc.ipk-gatersleben.de/dlg/Hordeum_Glycolysis.pdf)

5. OUTLOOK

Beside the presented application of the data linkage graphs, it is possible to perform more comprehensive applications of the presented concepts. The description was used to give a proofof-principle. More use cases, extensions to the DALIGRES tool and further more detailed quality control of the computational sequence annotation mappings are on the road map. Further plans are underway to extract the data schema graph from the life science Web databases.

6. REFERENCES

[1] L. Stein, "Creating a bioinformatics nation", Nature, vol. 417, no. 6885, pp. 119–120, 2002.

[2] R. Stevens, C. Goble, P. Baker, and A. Brass, "A classification of tasks in bioinformatics", Bioinformatics, vol. 17, no. 2, pp. 180–188, 2001.

[3] A. Divoli, M. Hearst, and W. M. A., "Evidence for Showing Gene/Protein Name Suggestions in Bioscience Literature Search Interfaces", in Pacific Symposium on Biocomputing, vol. 13, 2008, pp. 568–579.

[4] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", Nucleic Acids Research, vol. 25, no. 17, pp. 3389–3402, 1997.

[5] G. Schuler, J. Epstein, H. Ohkawa, and J. Kans, "Entrez: Molecular Biology Database and Retrieval System", Methods in Enzymology, vol. 266, pp. 141–161, 1996.

[6] A. Doms and M. Schroeder, "GoPubMed: exploring PubMed with the Gene Ontology", Nucleic Acids Research, vol. 33, no. suppl 2, pp. W783–786, 2005.

[7] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li, "Taverna: a tool for the composition and enactment of bioinformatics workflows", Bioinformatics, vol. 20, no. 17, pp. 3045–3054, 2004.

[8] M. Y. Galperin, "The Molecular Biology Database Collection: 2008 update", Nucleic Acids Research, vol. 36, no. suppl 1, pp. D2–4, 2008.

[9] F. Bry and P. Kröger, " A Computational Biology Database Digest: Data, Data Analysis, and Data Management", Distributed and Parallel Databases, vol. 13, no. 1, pp. 7–42, 2003.

[10] Z. Lacroix, T. Morris, K. Parekh, L. Raschid, and M.-E. Vidal, "Exploiting multiple paths to express scientific queries", in SSDBM'04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM'04). Washington, DC, USA: IEEE Computer Society, 2004, p. 357.

[11] S. Cohen-Boulakia, O. Biton, S. Davidson, and C. Froidevaux, "BioGuideSRS: querying multiple sources with a user-centric perspective", Bioinformatics, vol. 23, no. 10, pp. 1301–1303, 2007.

[12] M. Lange, A. Himmelbach, P. Schweizer, and U. Scholz, "Data Linkage Graph: computation, querying and knowledge discovery of life science database networks", Journal of Integrative Bioinformatics, vol. 4, no. 3, p. e68, 2007.

[13] C. Kanz, P. Aldebert, N. Althorpe, W. Baker, A. Baldwin, K. Bates, P. Browne, A. van den Broek, M. Castro, G. Cochrane, K. Duggan, R. Eberhardt, N. Faruque, J. Gamble, F. G. Diez, N. Harte, T. Kulikova, Q. Lin, V. Lombard, R. Lopez, R. Mancuso, M. McHale, F. Nardone, V. Silventoinen, S. Sobhany, P. Stoehr, M. A. Tuli, K. Tzouvara, R. Vaughan, D.Wu,W. Zhu, and R. Apweiler, " The EMBL Nucleotide Sequence Database", Nucleic Acids Research, vol. 33, no. suppl 1, pp. D29–33, 2005.

[14] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, R. G. M., and G. Sherlock, " Gene Ontology: tool for the unification of biology", Nature Genetics, vol. 25, pp. 25–29, 2000.

[15] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya, "The KEGG database at GenomeNet", Nucleic Acids Research, vol. 30, no. 1, pp. 42–46, 2002.
[16] T. Etzold, H. Harris, and S. Beaulah, SRS: An Integration Platform for Databanks and Analysis Tools in Bioinformatics, ser. Bioinformatics: Managing Scientific Data. Morgan Kaufmann Publishers, 2003, pp. 109–145.
[17] S. Balko, M. Lange, R. Schnee, and U. Scholz, "BioDataServer: An Applied Molecular Biological Data Integration Service", in Data Integration in the Life Sciences; First International Workshop, DILS 2004 Leipzig, Germany, ser. Lecture Notes in Bioinformatics, E. Rahm, Ed., vol. 2994. Berlin et al: Springer, 2004, pp. 140–155.
[18] A. Kasprzyk, D. Keefe, D. Smedley, D. London, W. Spooner, C. Melsopp, M. Hammond, P. Rocca-Serra, T. Cox, and E. Birney, "EnsMart: A Generic System for Fast and Flexible Access to Biological Data", Genome Research, vol. 14, no. 1, pp. 160–169, 2004.
[19] H. W. Mewes, D. Frishman, U. Güldner, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Münsterkötter, S. Rudd, and B. Weil, "MIPS: a database for genomes and protein sequences", Nucleic Acids Research, vol. 30, no. 1, pp. 31–34, 2002.
[20] M. A. Andrade, N. P. Brown, C. Leroy, S. Hoersch, A. de Daruvar, C. Reich, A. Franchini, J. Tamames, A. Valencia, C. Ouzounis, and C. Sander, "Automated genome sequence analysis and annotation", Bioinformatics, vol. 15, no. 5, pp. 391–412, 1999.
[21] A. Hamosh, A. F. Scott, J. Amberger, C. Bocchini, D. Valle, and V. A. McKusick, "Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders", Nucleic Acids Research, vol. 30, no. 1, pp. 52–55, 2002.
[22] O. Thimm, O. Bläsing, Y. Gibon, A. Nagel, S. Meyer, P. Krüger, J. Selbig, L. A. Müller, S. Y. Rhee, and M. Stitt, " MAPMAN: a user driven tool to display genomics data sets onto diagrams of metabolic pathways and other biological processes", The Plant Journal, vol. 37, no. 6, pp. 914–939, 2004.