Data Linkage Graph: computation, querying and knowledge discovery of life science database networks

Matthias Lange, Axel Himmelbach, Patrick Schweizer, Uwe Scholz

Institute for Plant Genetics and Crop Plant Research (IPK) Gatersleben, Germany
{lange|himmelba|schweiz|scholz}@ipk-gatersleben.de

Summary

To support the interpretation of measured molecular facts, like gene expression experiments or EST sequencing, the functional or the system biological context has to be considered. Doing so, the relationship to existing biological knowledge has to be discovered. In general, biological knowledge is worldwide represented in a network of databases. In this paper we present a method for knowledge extraction in life science databases, which prevents the scientists from screen scraping and web clicking approaches.
We developed a method for extraction of knowledge networks from distributed, heterogeneous life science databases. To meet the requirement of the very large data volume, the method used is based on the concept of data linkage graphs (DLG). We present an efficient software which enables the joining of millions of data points over hundreds of databases. In order to motivate possible applications, we computed networks of protein knowledge, which interconnect metabolic, disease, enzyme and gene function data.
The computed networks enabled a holistic relationship among measured experimental facts and the combined biological knowledge. This was successfully applied for a high throughput functional classification of barley EST and gene expression experiments with the perspective of an automated pipeline for the provisioning of controlled annotation of plant gene arrays and chips.
Availability: The data linkage graphs (XML or TGF format), the schema integrated database schema (GML or GRAPH-ML) and the graph computation software may be downloaded from the following URL: `http://pgrc.ipk-gatersleben.de/dlg/`

1 Motivation

High throughput biotechnologies, like cDNA or oligo arrays, genome and EST sequencing projects towards proteomics techniques, produce data from measured molecular facts. Those facts could be gene expression, polypeptide concentration or EST sequences.

To support the interpretation of measured molecular facts in order to discover the functional role of the measured molecule, the relationship to existing biological knowledge presented in world wide distributed databases, has to be built. To make the data accessible to desktop computers in the labs, the World Wide Web (WWW) is very popular. Techniques like manual browsing towards web services are commonly used. Using these techniques, bioinformatician may facilitate database embedded tools for linking wet-lab data to database knowledge in an explorative way. Methods like sequence similarities, stochastic networks for pattern recognition or text similarities queries are the selection operations, which are used for this data domain [Ste02, SGBB01].

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. 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 heterogeneties 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 [BK03]. The navigation and benefits of exploiting the data networks was motivated in [LMP+04]. Here a system called BioNavigation supports the scientist in exploring the content within an integrated database system using the BioMediation-system. This system supports the graph based browsing and query building on top of integrated databases.

In this paper, we present a recently developed method 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 this 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 will be described.

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 developed.

2.1 Data schema graphs

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, Gene Ontology (GO) terms [ABB+00] or EC numbers. To keep the modeled context of the attributes, they are assigned to data entities. In the example above *GO term* and *EMBL sequence* are entities, *id* and *accession* are the key attributes. Following this, 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 [BK03]. 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 this 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. Thus, we write:

$$R \subseteq S \times S \tag{1}$$

Because, we are interested in real existing relationships among schema nodes, we restrict R by:

$$R_{\texttt{link}} = \{(s,f),(f,s) \mid s,f \in S \wedge f \text{ is in relation to } s\} \quad (2)$$

$$R_{\texttt{func}} = \{(s_1,s_2),(s_2,s_1) \mid s_1,s_2 \in S \wedge s_1.E = s_2.E\} \quad (3)$$

$$R = R_{\texttt{link}} \cup R_{\texttt{func}} \quad (4)$$

In other words, we consider two kinds of bidirectional relations among key attributes:

$R_{\texttt{link}}$**:** Inter-entity relationships, which are modeled as links in the particular databases. Those could be represented by: HTML hyper links, primary key/foreign key, object references etc.

$R_{\texttt{func}}$**:** 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, childs of the same XML parent item or modeling as attributes of one relational table.

With this rules we are able to construct a graph as a pair of entity and relationship sets: $\alpha = (S,R)$, that comprises a database network at schema level.

2.2 Data linkage graphs

The next step 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 of S, 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 a $=$ relation in all used data domains for our key attributes A, we can define relationships $\rightleftharpoons$ among two elements of a data node ($sv_1, sv_2 \in SV$), if one of the subsequent conditions is fulfilled. Furthermore, we assume the existence of two operations selection σ and projection π, which are borrowed from relational algebra. If we use SV as a relational table, the selection returns all those tuples (data nodes), in which a certain element is equal to a given one. To decide the equality, we use the $=$ operation. In our application, the projection reduce the data node tuples to one specified element.

$$\begin{aligned} &\{(sv_1.E, sv_1.A),(sv_2.E, sv_2.A)\} \subset R_{\texttt{link}} \\ &\pi_{sv_1.A}(\sigma_{sv_2.A=sv_2.V}(sv_2.E)) \\ &= \\ &\pi_{sv_2.A}(\sigma_{sv_1.A=sv_1.V}(sv_1.E)) \end{aligned} \quad (5)$$

or

$$\begin{aligned} &\{(sv_1.E, sv_1.A),(sv_2.E, sv_2.A)\} \subset R_{\texttt{func}} \\ &\sigma_{sv_1.A=sv_1.V}(sv_1.E) \\ &= \\ &\sigma_{sv_2.A=sv_2.V}(sv_2.E) \end{aligned} \quad (6)$$

Colloquially, two data nodes are in relation if,

1. the schema nodes are in relation and the data elements are the equal (equation 5), or
2. the data nodes are mappable to the same schema and the tuples come from the same instance, e.g. table row or Web page (equation 6).

To illustrate these types of relationships, the relationships between two data sources are visualized in figure 1: Assuming that there are the EMBL and the Gene Ontology (GO) database.

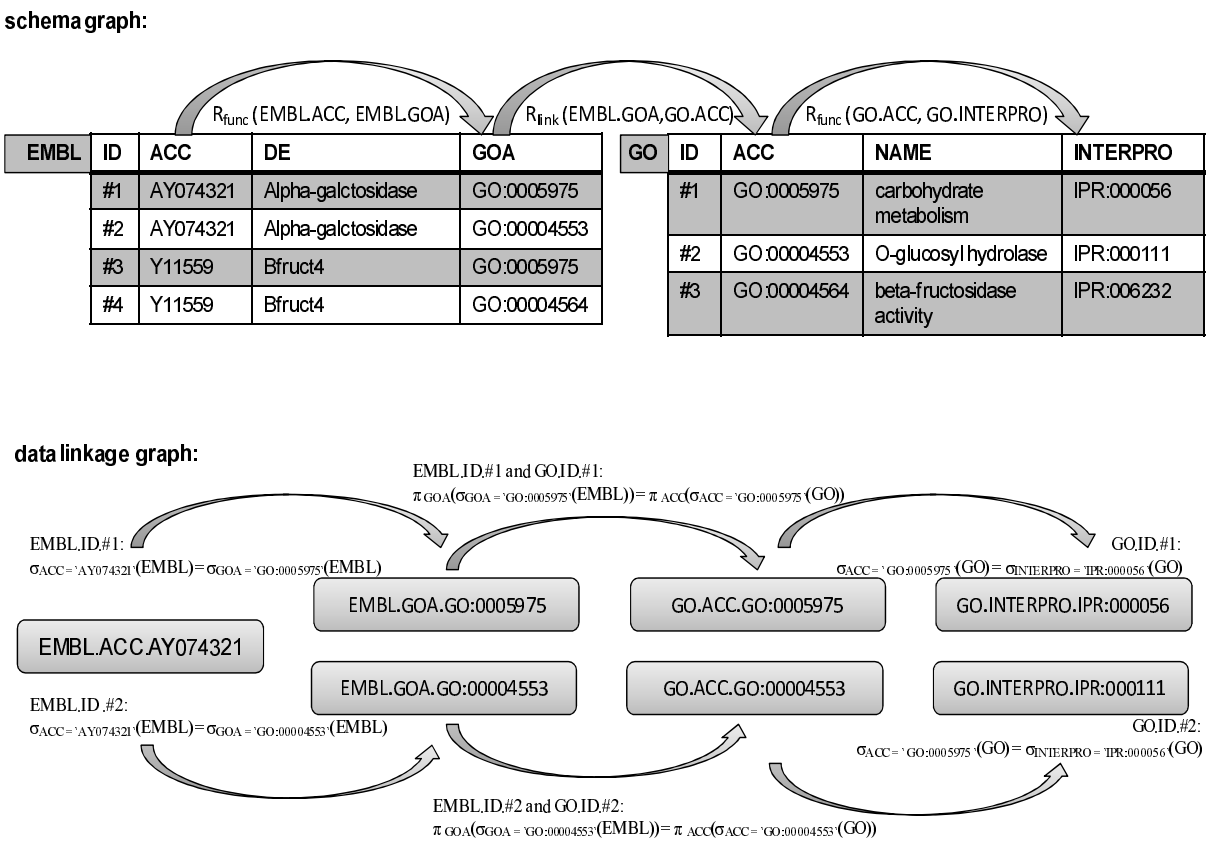

Figure 1: example for the relationships in schema and data linkage graphs

Each comprise three attributes in one entity. With known inter-attribute relationships (e. g. derived from hyperlinks among EMBL and GO), our example includes three relations:

$$\begin{aligned} R_{func_1} &= (EMBL.ACC, EMBL.GOA) \\ R_{func_2} &= (GO.ACC, GO.INTERPRO) \\ R_{link_1} &= (EMBL.GOA, GO.ACC) \end{aligned}$$

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

$$RV \subseteq SV \times SV \tag{7}$$

such that

$$(sv_1, sv_2) \in RV \wedge sv_1 \rightleftharpoons sv_2 \tag{8}$$

Finally, we are able to describe networks of data nodes as set of graphs with the structure $\beta = (SV, RV)$.

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 conflicts needs to be solved. Leigthweigt, because only the key attributes values have to be accessed physically, parsed and downloaded respectively.

Subsequently we describe the concrete construction of schema networks and the computation of data networks, in form of data linkage graphs (DLGs).

3 Implementation

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 prerequists are (1) a graph of schema nodes and (2) for all schema nodes, the attribute values.

3.1 Data Import

For the first task, 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 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 databases. In a collaboration with the University of Bielefeld, Germany, 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 in-house developed parser, the Sequence Retrieval System (SRS) [EUA96] and a database mediator called BioDataServer (BDS) [BLSS04]. In this way, a file of key values for each schema node was generated. The results are 318 files comprising around $5.6*10^7$ 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

This properties influenced the strategy for the DLG 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.

In other words, if all of the i schema nodes were joined ($\bowtie$) and a cascading merge join was used, we would have a polynomial complexity, because database management systems are optimized for a data storage using secondary memory, usually hard discs and the cascading sort-merge join is the commonly used join implementation in relational database management systems. The complexity of each merge join is $O(n)$. Consequently, the overall complexity is:

$$\begin{aligned} O_{join} &= O(n)_{table_1 \bowtie table_2} * \ldots * O(n)_{table_{i-1} \bowtie table_i} \\ &= O(n)^i \end{aligned} \tag{9}$$

The consequence is the limitation to use case specific data warehouses, which disables a comprehensive database integration. Thus, there is a complexity problem for comprehensive data queries over huge data warehouses using big systems like SRS or even ORACLE. They support efficient 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 investigated main memory based data structures and algorithms. We discussed previously that the nodes in our data networks show high redundancy and a compression rate of factor 10 can be achieved. Thus, we where able to represent all nodes by 5.6*10^7 64-bit pointer to 5.4*10^8 data nodes. This data structure took around 4GByte. Using this data structure, we pre-compute all possible joins in main memory.

A C++ program loads all data nodes into main memory and computes all possible graphs. For this, we used the recursive breadth-first search (BFS) algorithm in combination with index structures like hash maps, B-trees and bit vectors. The resulted graphs where stored in a flat relational structure: (`graphid, node_depth, parent_entity, parent_attribute, child_entity, child_attribute, parent_value, child_value`).

This graphs are available in csv-format at the URL given bellow. Because of the recently finished graph computation, no data format optimization was done. The representation in a hierarchical data structure, like XML, is ad-hoc realizable, but not yet used practically. The graph computation took 3 weeks on a 2GHz Opteron server, whereas 12 GByte of main memory and 2 CPUs were used.

3.3 Data network queries

We loaded the graphs into an ORACLE 10g database table. Some example queries were performed, like a query for all reachable data nodes starting from a given one. The application for such a query is e.g. the assignment of EMBL sequences to Gene Ontology controlled vocabulary, metabolic pathways in KEGG, diseases in OMIM etc. The former necessary join was replaced by a simple 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'
```

The complexity of such a query is derived from the complexity of the two selection operations for selecting the graphs containing the queried start and end node, and additionally their union. Using indexes for the selection, the complexity is $O(log\ n)$. Applying merge join on sorted graph identifiers, the complexity is $O(n)$. Therefore, the overall complexity of the above query is:

$$\begin{aligned} O_{graphquery} &= O(log\ n) + O(log\ n) + O(n) \\ &= O(n) \end{aligned} \tag{10}$$

If we don't take in account that we need to compute the DLG with the BFS complexity (because it will be necessary only once per database update), this will be a significant reduction of the polynomial join complexity.

4 Discussion

The analysis and interpretation of wet-lab data, performed at a typical scientist desk is an iterative process of tool application and database exploration. 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 of this paradigm is, for instance, (1) the mapping of translated nucleic sequences by direct sequence homology recognition using the BLAST tool [AGM+90] in an annotated, merged polypeptide sequence database like NRPEP (`ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nr.gz`). The hits which match known proteins link to the original databases and have to be (2) extracted eighter manually or using parsers. Finally, all linked databases have to be inspected for getting relevant information for the input sequences.

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 [MFG+02], the Gene Ontology Database [ABB+00], GeneQuiz [ABL+99], OMIM [HSA+02], KEGG [KGKN02] or MapMan [TBG+04]. All those databases where more or less manually constructed.

4.1 Functional mapping to controlled vocabulary

Here we present an application of the DLGs to map general functional sequence annotation to controlled vocabulary in a computational way.

The basic idea of the functional categorization is to (1) take BLAST [AGM+90] 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 finally (4) query the data link graphs to (5) find a path from the found database identifier to the interesting annotation database entity. In first use cases, this work flow was implemented to map Bests to their functional role in the Gene Ontology (GO) Database. The result is the assignment of ESTs to the three GO ontologies. The work flow has been implemented by an SQL-query, using the start-destination query pattern presented before. Figure 2 visualizes this process.

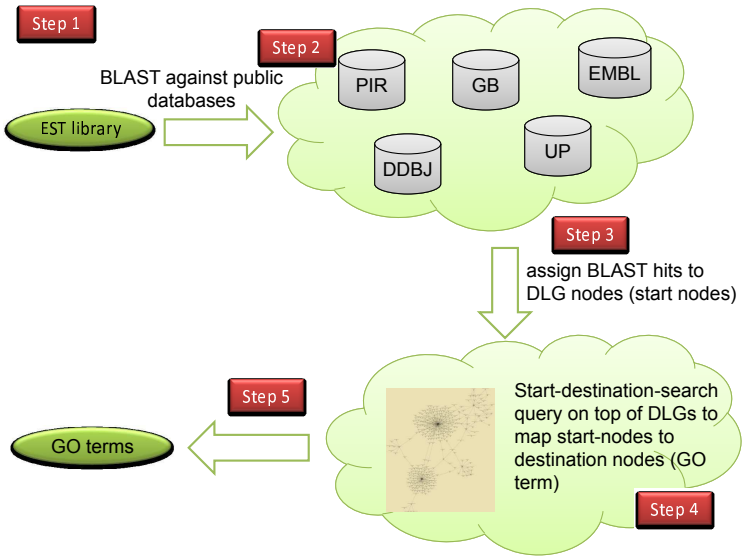

Figure 2: work flow to assign ESTs to GO terms

In order to evaluate the computational mapping of ESTs to GO terms, a set of 440 EST has been mapped computational to a GO term in around 2 minutes using an ORACLE 10g on a 2GHz Itanium server. Those ESTs represent genes which are used for an IPK internal cDNA macro array. Thus, the functional role of those gene has been studied in detailed from institute's scientists, but had never been before included in GO. For the automatic annotation using the described pipeline, the cut-off criteria for the used BLAST results was an e-value less then 1E-10. The results of the annotation have been manually checked. We discovered, that 6% had a false assignment to GO terms. 80% were assigned unquestionable correct. The rest of 14% were not clearly decidable but tended to be correct. Those first investigations have been recently done. Thus, the data is available on request from the authors.

Because of this promising results and to demonstrate the flexibility of the approach, another annotation mapping was computed to KEGG [KGKN02] metabolic pathways. This query took on the same server approximately 45 minutes. We were able to identify 18,607 out of 111,090 sequences stored in the IPK CR-EST database [KLF+05], which show enzymatic activity. By using the data linkage graphs and starting from either EMBL, Genbank, DDBJ, PIR or SWISSPROT/TREMBL in the BLAST results. 21% of these ESTs assigned to functional categories belong to carbohydrate metabolism, 19% to amino acid- and 17% energy metabolism. In contrast, using BLAST results only 1906 functional descriptions containing an EC numbers were found. Hence, compared to BLAST the sensitivity of functional annotation was increased 10-fold. Figure 3 shows a simple statistic over the non-redundant metabolic annotated ESTs.

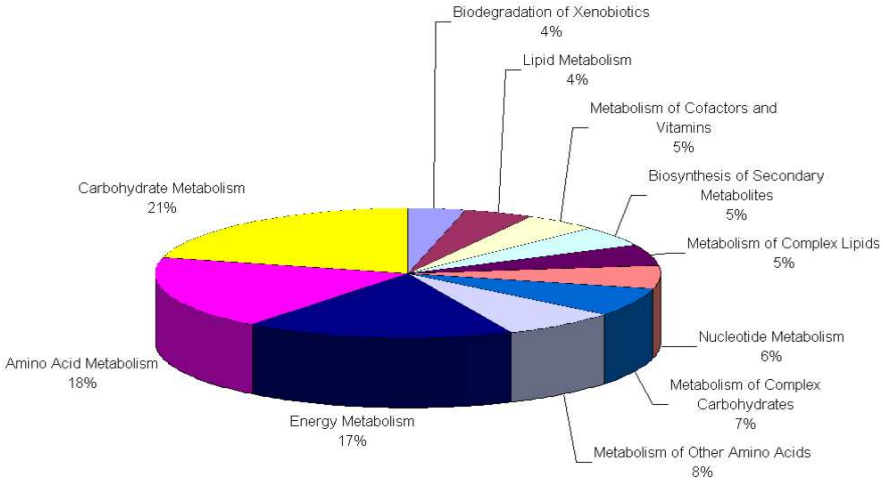

Figure 3: mapping of ESTs from the CR-EST database to GO terms

4.2 Database network discovery

Beside the above mentioned graph queries, further analysis will be done. Scenarios are *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 supported by a multidimensional, structural analysis of integrated databases, 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 e. g. instance extraction all networks that include contradicting data nodes for the same schema node.

The URL `http://pgrc.ipk-gatersleben.de/dlg/` can be used to download the data schema graph (GML or GraphML format), the data linkage graphs (CSV format) or examples of context specific extracted metabolic knowledge networks for plants (GML or GraphML format). Later are queried using SQL by joining the DLGs over the enzymes (EC numbers) which are present in glycolysis metabolic pathway and protein 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 is from this figure is available, using the graph datafiles in a graph viewer. The files and a link to a viewer are available using the URL given previously.

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 proof-of-principle. More use cases and further more detailed quality control of the computational sequence annotation mappings are on the road map. Further plans are underway to expand the data basis and implement a user friendly and flexible tool to perform queries on top of data linkage graphs to discover knowledge in networks of life science databases.

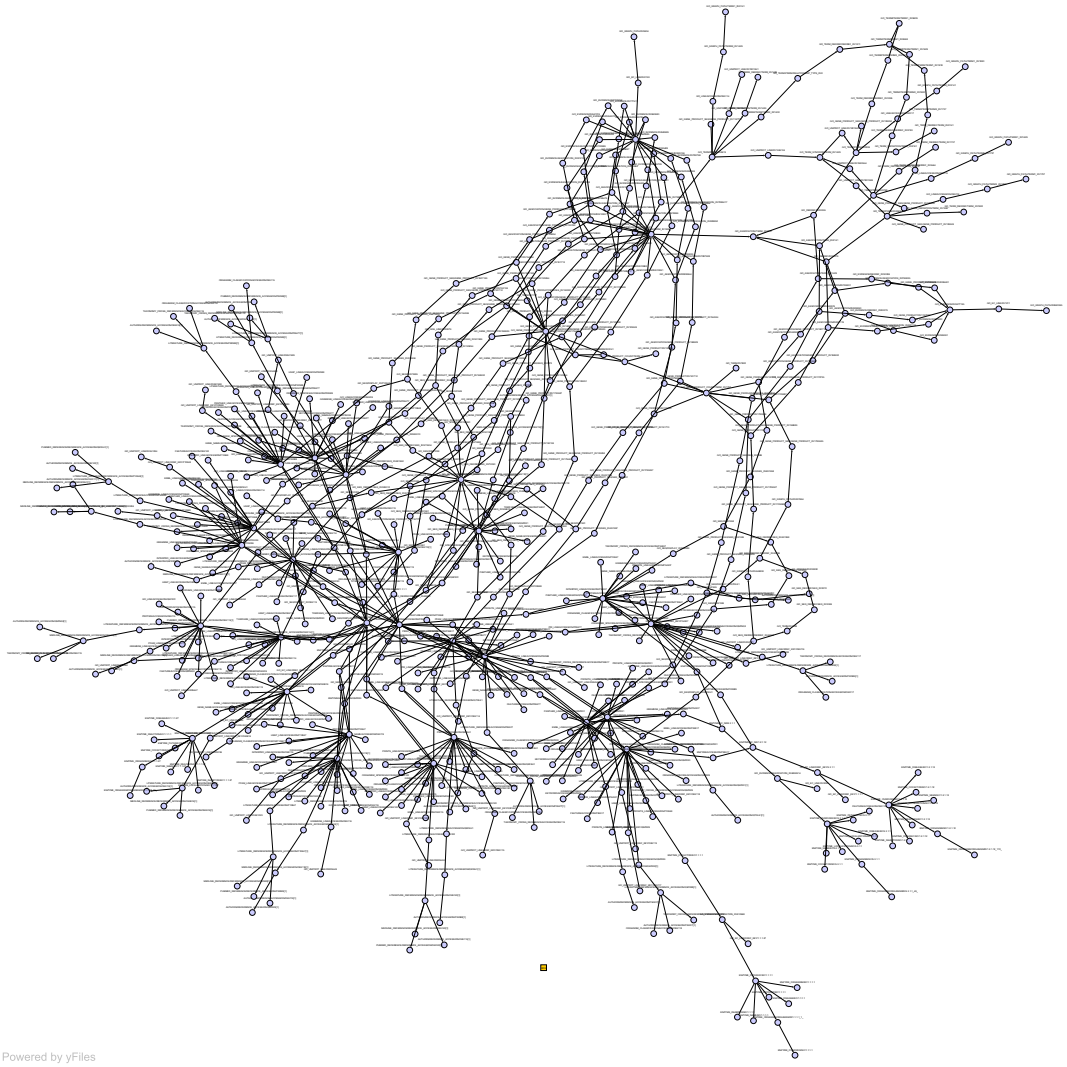

Figure 4: data network of enzymes in barley showing catalytic activity in glycolysis (high-resolution image: `http://pgrc.ipk-gatersleben.de/dlg/Hordeum_Glycolysis.pdf`)

References

[ABB+00] 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.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, Rubin G. M., and G. Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[ABL+99] 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*, 15(5):391–412, 1999.

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 3:403–410, 1990.

[BK03] F. Bry and P. Kröger. A Computational Biology Database Digest: Data, Data Analysis, and Data Management. *Distributed and Parallel Databases*, 13(1):7–42, 2003.

[BLSS04] S. Balko, M. Lange, R. Schnee, and U. Scholz. BioDataServer: An Applied Molecular Biological Data Integration Service. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Data Integration in the Life Sciences; First International Workshop, DILS 2004 Leipzig, Germany*, volume 2994 of *Lecture Notes in Bioinformatics*, pages 140–155. Berlin et al: Springer, 2004.

[EUA96] T. Etzold, A. Ulyanow, and P. Argos. SRS: Information Retrieval System for Molecular Biology Data Banks. *Methods in Enzymology*, 266:114–128, 1996.

[HSA+02] 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*, 30(1):52–55, 2002.

[KGKN02] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG database at GenomeNet. *Nucleic Acids Research*, 30(1):42–46, 2002. http://www.genome.ad.jp/kegg/.

[KLF+05] C. Künne, M. Lange, T. Funke, H. Miehe, I. Grosse, and U. Scholz. CR–EST: a resource for crop ESTs. *Nucleic Acids Research*, 33(suppl_1):D619–621, 2005.

[LMP+04] Z. Lacroix, T. Morris, K. Parekh, L. Raschid, and M.-E. Vidal. Exploiting multiple paths to express scientific queries. In *Proceedings of SSDBM 2004*, 2004.

[MFG+02] H. W. Mewes, D. Frishman, U. Güldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Münsterkötter, S. Rudd, and S. Weil. MIPS: a database for genomes and protein sequences. *Nucleic Acids Research*, 30(1):31–34, 2002.

[SGBB01] R. Stevens, C. Goble, P. Baker, and A. Brass. A classification of tasks in bioinformatics. *Bioinformatics*, 17(2):180–188, 2001.

[Ste02] L. Stein. Creating a bioinformatics nation. *Nature*, 417:119–120, 2002.

[TBG+04] O. Thimm, O. Blasing, Y. Gibon, A. Nagel, S. Me yer, P. Kruger, J. Selbig, L. A. Muller, 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. *Plant Journal*, 37(6):914–914, 2004.