BIOINFORMATICS

Vol. 18 Suppl. 2 2002
Pages S219–S230

BioMiner—modeling, analyzing, and visualizing biochemical pathways and networks

M. Šīrava[1], T. Schäfer[1], M. Eiglsperger[2], M. Kaufmann[2], O. Kohlbacher[1], E. Bornberg-Bauer[3] and H. P. Lenhof[1]

[1]Center for Bioinformatics, Saarland University, P.O. Box 151150, Saarbrücken, 66041, Germany, [2]Wilhelm-Schickard Institut für Informatik, University of Tübingen, Sand 13, Tübingen, 72076, Germany and [3]Umber, School of Biological Science, University of Manchester, Manchester, M13 9PT, UK

Received on April 8, 2002; accepted on June 15, 2002

ABSTRACT

Motivation: Understanding the biochemistry of a newly sequenced organism is an essential task for post-genomic analysis. Since, however, genome and array data grow much faster than biochemical information, it is necessary to infer reactions by comparative analysis. No integrated and easy to use software tool for this purpose exists as yet.

Results: We present a new software system—BioMiner—for analyzing and visualizing biochemical pathways and networks. BioMiner is based on a new comprehensive, extensible and reusable data model—BioCore—which can be used to model biochemical pathways and networks. As a first application we present PathFinder, a new tool predicting biochemical pathways by comparing groups of related organisms based on sequence similarity. We successfully tested PathFinder with a number of experiments, e.g. the well studied glycolysis in bacteria. Additionally, an application called PathViewer for the visualization of metabolic networks is presented. PathViewer is the first application we are aware of which supports the graphical comparison of metabolic networks of different organisms.

Availability: http://www.zbi.uni-saarland.de/chair/projects/BioMiner

Contact: schaefer@bioinf.uni-sb.de

Supplementary Information: Additional information on experimental results can be found on our web site.

Keywords: Biochemical data model, metabolic and regulatory pathways, visualization, Java, XML.

INTRODUCTION

In the age of genomics and proteomics, large scale data are produced by numerous scientific groups all over the world. Even now, after the race for the human genome has come to an end by the publication of the first drafts of the human genome (Venter *et al.*, 2001; International Human Genome Sequencing Consortium, 2001), the total amount of available biochemical data, e.g. proteomics data, protein-protein interaction data, or information on metabolic and regulatory networks, is growing at an exponential rate.

However, because of its direct implications for pathogenicity, agricultural productivity and drug-interaction, it is the biochemistry of an organism which is of actual interest. Only few model organisms are biochemically well characterized. Inferring biochemcial information through comparative analysis of large scale array data is therefore both important and challenging. The main hurdles for an efficient exploitation are, besides the sheer amount of data, the plethora of different—and usually incompatible—data formats of the databases[†]. This lack of integration and interoperability gave rise to numerous bioinformatics tools (Goesmann *et al.*, 2002; Küffner *et al.*, 2000; Arita *et al.*, 2000) and a few integrative systems (Kanehisa and Goto, 2000; Karp *et al.*, 2000; Overbeek *et al.*, 2000; van Helden *et al.*, 2000, 2001b).

Nevertheless, there is a strong need for user-friendly systems allowing the user to handle and combine large and diverse data sets from different sources and to gather the required information and generate new insights from it.

A convenient user interface for the analysis of the data should support visualization since biochemical knowledge is difficult to conceptualise. For metabolic networks, most graphical interfaces (Kanehisa and Goto, 2000; van Helden *et al.*, 2000; Appel *et al.*, 1994; Overbeek *et al.*, 2000) are designed to display only expert-curated static images of pathways and networks, i.e. the textbook view of things. These static representations are often inconsistent with the (more up-to-date) data found in the database itself. Current graph layout algorithms as described in Di Battista *et al.* (1999) are not sufficient to fulfill the special requirements and graphical conventions

[†] See for example Kanehisa and Goto (2000); Karp *et al.* (2000); Overbeek *et al.* (2000). An overview of metabolic databases can be found in Wittig and De Beuckelaer (2001)

for metabolic networks. A few systems provide dynamically generated views of the underlying data and fulfill the special requirements and graphical conventions for metabolic pathways (Goesmann *et al.*, 2002; Mendes *et al.*, 2000; Karp *et al.*, 2000; Ellis *et al.*, 2001; Brandenburg *et al.*, 2001), but they are not capable of handling data from more than one organism and multiple pathways.

We present BioMiner, a new software framework for rapid prototyping of integrative analysis tools for complex biochemical data. BioMiner was designed to simplify the implementation of tools for answering a broad range of biochemical questions. It is based on a comprehensive data model, named BioCore, representing complex and diverse biochemical data in a unified fashion. This model is currently able to handle metabolic pathways, transcription data, protein-protein interaction data, and some of the more important regulatory processes (e.g. signaling pathways). The implementation uses LEDA (Mehlhorn and Naher, 1999), a software library providing many standard graph algorithms. This makes BioMiner extremely helpful for rapid prototyping of biochemical software tools for the comparative analysis of complex biochemical networks.

In the following sections we give an overview of BioMiner. We then discuss the architecture and describe the biochemical data model in general. PathFinder and PathViewer are introduced before we describe the experimental results obtained. We finally discuss these results and point out future directions for the development of BioMiner.

SYSTEMS AND METHODS

BioMiner consists of two components, BioCore and yWays (see Figure 1). BioCore is a comprehensive biochemical data model consisting of a number of classes as well as class methods implementing core graph functions, i.e. a set of functions to be applied to these classes. yWays is the visualization component of BioMiner including a special data model and functions to layout metabolic pathways and networks.

In the following sections we use a number of technical terms like *pathway*, *graph* etc. For definitions of these terms see appendix.

Design of BioMiner

The system is designed as a three-tier application. The first tier provides interfaces to existing databases[†]. The second tier is responsible for the analysis of biochemical data. The result of the analysis is visualized in the third tier. For data exchange between analysis and visualization applications,

[†] At the time of writing, a KEGG (Kanehisa and Goto, 2000) interface exists, interfaces to other databases like BIND, DIP, TRANSFAC and TRANSPATH are in preparation.

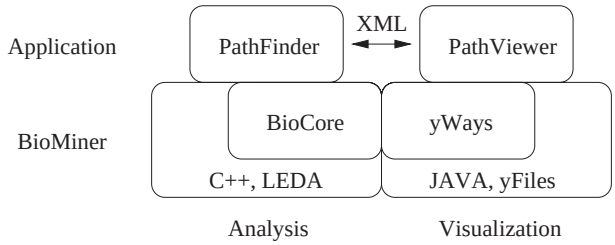

Fig. 1. BioMiner architecture.

BioMiner offers an XML export. For PathFinder and PathViewer, e.g. which have been implemented on top of BioMiner, an XML exchange format is provided.

The analysis and visualization components are separated because of different requirements: While performance is most important for the data analysis, for the visualization it is more important to provide a rich set of user interaction possibilities and to be platform independent. The analysis part is implemented in C++ using LEDA (Mehlhorn and Naher, 1999), a library which supplies many of the relevant data structures and algorithms. The visualization part is implemented in JAVA. This ensures availability for a wide variety of platforms. We used the JAVA-based library yFiles (Wiese *et al.*, 2001) which provides a powerful framework for visualization applications.

BioCore

Based on a thorough analysis of the biochemical terminology and existing models (van Helden *et al.*, 2001b, 2000; Paton *et al.*, 2000; Rzhetsky *et al.*, 2000) as well as on discussions with biochemists, we designed a comprehensive framework of classes to model biochemical entities. BioCore contains a large number of classes, which are sufficient to model the majority of biochemical processes of interest. We will only briefly discuss some of the central classes and refer to the documentation of BioCore (see our web page) for details.

Modeling biochemical data. We consider all occurring processes to be some kind of *event*. Let us consider one of the most fundamental processes, a biochemical *reaction*. Obviously, all entities taking part in that event are compounds of one kind or other. The biochemical community uses a vast set of sometimes fuzzy, often overlapping terms to describe and to classify those compounds, e.g. simple ions, small molecules, amino acids, peptides etc. From that terminology, we derived a small set of rather well-defined classes describing (chemical or rather abstract) entities participating in the events: `Participant` serves as a base class for several derived classes like `Protein` or `NucleicAcid` on the one hand and more abstract objects like `Pathway` or `Gene` on the other hand. Substance classes which are not modeled

explicitly (e.g. lipids, nucleotides) are comprised in the class `Compound`, which provides member flags for the further characterization of its properties, i.e. a technique to use a generic object which can be further specified by setting special 'switches' depending on the kind of object to be stored. This prevents the explosion of our class hierarchy due to the large number of existing compound classes.

Each of these compounds can take part in various events, the most important one being a `Reaction`. In that reaction, as in all other events, each participating compound plays a certain *role*. Many of these roles are implicitly given by conventions for writing biochemical reactions: Reactants (or educts) are on the left-hand side of the equation, products on the right-hand side, and so on. In our model, we express this relationship through *roles*. Each `Participant` plays a certain `Role` in a given `Event` or `Reaction`, e.g. we can denote that substrates (educts) and products, although both being of type `Participant`, have different functions in an event. A further distinction between `Main Educt/Product` and `Side Educt/Product` is, e.g. important for the PathFinder algorithm.

`Participant`, `Role`, and `Event` thus form the keystones of our object model. The model itself was developed using the Unified Modeling Language (UML, Booch *et al.* (1999)). We assume that the reader is somewhat familiar with the notation of UML class diagrams for the remainder of this section[†].

Figure 2 shows the relationships of the central classes. Most classes are self-explanatory through their names. `Role` connects compounds and events, which allows us to model the different contextual functions a compound may have (e.g. a single compound can be an educt in one reaction, but a product of another). Modeling the role as an object also offers a convenient way for classifying and extending the set of roles a compound can play in different events. Figure 2 only shows metabolic classes; for a complete overview we refer to our web site.

BioCore and other biochemical models. Comparing existing data models described in van Helden *et al.* (2001b); Paton *et al.* (2000); Rzhetsky *et al.* (2000); Karp *et al.* (2000); Bader *et al.* (2001), the aMAZE data model (van Helden *et al.*, 2001b) seems to be the most flexible in our sense because it does not focus on one kind of biochemical interactions (like protein-protein-interactions in BIND (Bader *et al.*, 2001) or metabolic pathways in Eco/MetaCyc (Karp *et al.*, 2000)) but integrates a large variety of biochemical processes. Compared to other models, the BioCore `Role` object allows to model biochemical processes in a rather natural way describing the *function* of a participant in detail. Additionally, the number of classes and therewith the flexibility of BioCore is very high.

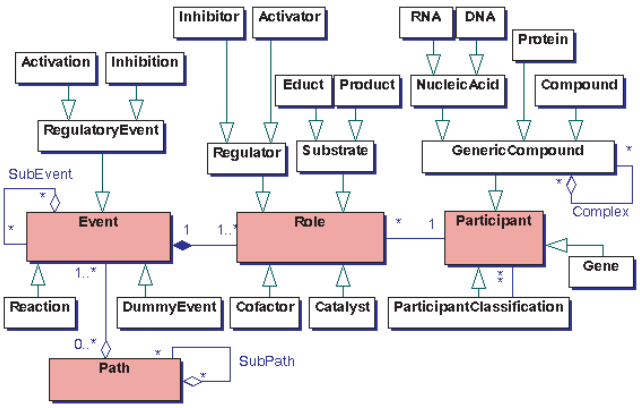

Fig. 2. Central classes of BioMiner (simplified; core classes in red). Only metabolic classes are shown here.

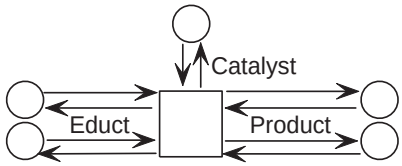

Fig. 3. Node types used in BioMiner. Boxes represent event nodes, circles stand for participant nodes. Edges are labeled with the respective roles.

Graph core functions. Events in biochemistry are generally represented as networks (Michal, 1993). One can either use very general representations (e.g. directed graphs (Arita, 2000; van Helden *et al.*, 2001a)) or more specialized ones, like petri nets (Hofestädt and Thelen, 1998; Küffner *et al.*, 2000).

Our graph representation is based on LEDA graphs. In contrast to typical representations in biochemical textbooks, our graph consists of only one type of directed edges but two types of nodes. Figure 3 shows the representation of a metabolic reaction in BioMiner.

The usual representations of (metabolic) reactions in biochemical textbooks use hyperedges as given in Figure 4. Representing a graph in such a manner, the uniqueness of direction and sequence of such an event is lost. In order to explicitly represent the direction and the sequence of the involved participants, we decided to use two types of nodes, one for events and one for participants. Edges in a BioMiner graph are directed. If

[†] Just to recall the notation: Arrows with hollow triangles represent generalization, e.g. a `Protein` *is a* `GenericCompound`. General associations between classes are shown as lines with cardinalities. Relationships with an open diamond stand for part/whole relationships. Closed diamonds indicate that the part cannot exist without the whole. Self associations as the complex association in Figure 2 allow objects to include subobjects of same type.

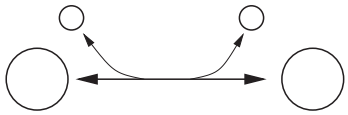

Fig. 4. Metabolic reaction as given in many biochemical textbooks: This kind of edges are called hyperedges.

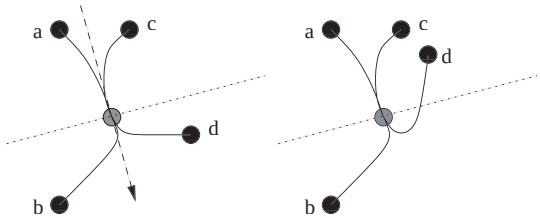

Fig. 5. Example for partition aesthetic criterion for a reaction with educts (a,c) and products (b,d).

(as in most cases) an event is bidirectional, two edges are inserted, one for each direction. To comprise an event node, it is connected to all its participants via several role edges. Thus, a participant node can take part in more than one event, playing different roles.

BioMiner is based on LEDA, a library offering a set of standard graph algorithms which had to be adapted to fit our graph model with two different node types. Examples are the query for a path between two arbitrary chemical compounds *s* and *t* or the enumeration of *all* biochemically valid paths and connections between *s* and *t*. Biochemically valid means, e.g. only sequences like substrate—enzyme—substrate are allowed.

yWays

In this section we discuss the visualization of metabolic pathways as diagrams. Although there is no standard notation for metabolic pathways, the diagrams in most publications are quite similar. We use a notation of Michal (1993). This notation has the following properties:

- A compound[†] is represented by a shape (rectangle, circle, etc.) usually containing the name of the compound.
- A reaction is represented by a set of curves, connecting the compounds participating in it. All curves connecting the educts join at a single point and all curves connecting the products join at a single point. These points either coincide or are connected by a curve.
- The names of enzymes are placed near the lines representing the catalyzed reaction.

In the diagram we distinguish between main and side compounds. Whereas main compounds may be connected to multiple reactions, side compounds are always connected to exactly one reaction. Typically, the side compounds are placed near this reaction and have a smaller font than main compounds. In a diagram there may be many occurrences of the same compound, every time being side compound in a different reaction. The reason for using the side compound notation is that some compounds are ubiquitous in some network, e.g. ATP and ADP in the glycolysis. Representing these compounds by multiple occurrences makes the diagram much more readable.

The data model for the visualization can be kept very simple, since it needs to model only the visual appearance of the network and not the biochemistry behind it. We represent a metabolic network (or more general a biochemical network) by a directed hypergraph. Each occurrence of a compound in the diagram of the metabolic network corresponds to a node in the hypergraph, and each reaction to a hyperarc.

Besides the model, the layout algorithm is important to get a pleasing visualization result. The task of a *layout algorithm* is to calculate the geometric properties of the hypergraph such that the resulting diagram is as readable as possible. These geometric properties are the coordinates of the shapes representing nodes and the description of the curves representing hyperedges. Because it is hard to mathematically define a concept such as 'readability', a layout algorithm tries to optimize some aesthetic criteria which can be formulated mathematically. We can divide these criteria into application independent criteria, which apply to any layout algorithm regardless of its application domain, and domain specific aesthetic criteria. Application independent criteria are, e.g. the number of line crossings, the number of bends, the occupied area and the length of the edges, see, e.g. Di Battista *et al.* (1999). We will concentrate now on the domain specific aesthetic criteria for diagrams of metabolic networks. The first aesthetic criterion, called *partition criterion*, is that the partition of a reaction into educts and products should be clear from the visualization. The second aesthetic criterion, called *direction criterion*, is that, if parts of the metabolic network are directed, i.e. the metabolic network contains a metabolic pathway, this should be shown on the diagram.

Layout algorithm. In this section we discuss the automatic layout algorithm for metabolic networks. The automatic layout algorithm can be roughly divided into the following phases: In the first phase we construct a graph from the metabolic network, the *induced graph G*. The induced graph contains a vertex for each compound and a vertex for each reaction. A compound vertex and a reaction vertex are connected by an edge if the compound is

[†] In the following we describe the visualization of metabolic pathways. Thus, the participants of events are mainly compounds.

an educt or product of the reaction. In the second phase we calculate an orientation for G such that the resulting directed graph D is acyclic. In the third phase we calculate the layout of D with the hierarchic graph layout algorithm of yFiles. This graph layout algorithm is an implementation of the layered graph drawing approach for directed acyclic graphs, see, e.g. Bastert and Matuszewski (2001) for a comprehensive overview of this approach. Using a sophisticated implementation of a classical graph drawing algorithm as back end has the advantage that the application independent aesthetic criteria are optimized. It remains to handle the domain specific criteria.

The result of the graph drawing algorithm is always downward directed, i.e. all edges in the drawing are monotonically decreasing in y-direction according to the orientation of D. Therefore, the relative ordering of the compounds according to the y-coordinate depends only on the orientation which is defined in phase two. We can now define the domain specific aesthetic criteria in terms of the orientation.

- To ensure the partition aesthetic criterion, edges connecting the educts of an reaction must be oriented opposite to the edges connecting the products.
- To ensure the direction aesthetic criterion for irreversible reactions, the edges connecting educts are directed towards and the edges connecting products are directed away from the reaction vertex.

Unfortunately, the problem of minimizing the number of edges, violating one of the above criteria, is NP-hard. This follows directly from the fact that already the problem for minimizing the number of edges for the second criterion is NP-hard, which is a well known result in graph drawing see, e.g. Bastert and Matuszewski (2001). We use a heuristics for solving this problem. It is based on depth-first search (DFS) and has linear running time. Details can e.g. be found in Bastert and Matuszewski (2001).

APPLICATIONS

PathFinder

As a first application of BioMiner, we developed and implemented PathFinder, a new tool for finding metabolic pathways in newly sequenced organisms. The construction of metabolic network is similar to the method described in Küffner *et al.* (2000) while PathFinder uses a different scoring function and another graph type.

The algorithm integrates known metabolic information for a group of organisms collected from diverse data sources into a metabolic graph. This graph is built using the biochemical data model of BioMiner. On the basis of sequence similarity and graph algorithms, pathways can be predicted for newly sequenced organisms. The list of found putative pathways is then ranked by assigning weights to each generated pathway.

The input consists of the whole genome sequence of a query organism q and the metabolic information (networks) for a set of reference organisms $O_{ref} = \{o_1, o_2, \ldots\}$. All organisms in O_{ref} should be closely related to q. The aim is to predict whether a known pathway between two compounds s and t in one of the reference organisms exists in the query organism as well or if possible alternative pathways can be inferred.

A pathway $P = (r_1, r_2, \ldots, r_k)$ is defined as a sequence of reactions[†] r_i, where the length k of the pathway is defined as the number of reactions involved. The number of possible pathways between two given compounds s and t is often very large due to the high degree of substrate connectivity observed in metabolic graphs (Jeong *et al.*, 2000). We therefore limit our search to pathways of a reasonable maximum length k. A pathway may be linear or cyclic.

The PathFinder algorithm. The metabolic data available for O_{ref} constitutes our initial metabolic graph $MG(C, R, E)$, which contains vertices for all compounds C (substrates) and all reactions R (corresponding to enzymes), and the corresponding edges E.

To speed up the search or to exclude specific pathways, it can be useful to restrict the set of compounds included in MG to a subset of all compounds in C. Let C_{side} the set of all side substrates (ubiquitous compounds as defined in KEGG) and C_{ex} a set of user-defined excluded compounds. We then define three restricted compound sets

1. $C_{res1} = C \backslash C_{side}$,
2. $C_{res2} = C \backslash C_{ex}$,
3. $C_{res3} = C \backslash (C_{side} \cup C_{ex})$.

If one of these these restricted sets is used, the *search graph* MG_s reduces from the full metabolic graph MG to the corresponding restricted graphs defined by C_{res} and the induced reactions and edges.

The algorithm consists of four steps:

1. **Construction of a *compressed* metabolic graph MG_c:** To reduce the complexity of the search graph, we collapse all primitive reactions r_i with the same set of ingoing/outgoing compounds in MG_s into a *compressed reaction* $rc = \{r_1, r_2, \ldots, r_n\}$. We call the resulting graph $MG_c(C, \{rc_i\}, E')$ the *compressed metabolic graph*. Each compressed reaction can contain identical reactions present in different organisms of O_{ref} and reactions with the same educts and products but catalyzed by a different enzyme in the same organism.

[†] For ease of reading, in the context of metabolic pathways we will use the term *reaction* instead of the generic term *event*.

2. **Refine search space in MG_c:** To limit our search to paths of at most k reactions, we compute the set of nodes that are reachable in $k/2$ reactions from s and t. All nodes outside this set cannot be part of pathways of length k or less and are deleted from MG_c along with the corresponding edges.
3. **Assigning weights to reactions:** For each compressed reaction rc we consider its primitive reactions r_i and align the enzyme sequence catalyzing r_i with the whole genome of q using BLAST (Altschul *et al.*, 1997). We then compute the weight $w(r_i)$ of the primitive reaction based on the p-value $p_{max}(r_i)$ of the best alignment computed:

$$w(r_i) = \ln(1 - p_{max}(r_i)) \quad (1)$$

   We define the weight $w(rc)$ of the compressed node as the maximum of the weights of its primitive reactions

$$w(rc) = \max_{r_i \in rc} w(r_i) \quad (2)$$

4. **Pathway search:** We explore all putative pathways $P = (rc_1, rc_2, \ldots, rc_N)$ using depth-first search and assign a weight

$$w(P) = \exp(\frac{1}{N} \sum_{i=1}^{N} w(rc_i)) \quad (3)$$

   All putative pathways are ranked with respect to their weight and reported in that order.

Implementation status. The PathFinder algorithms are implemented and several experiments have been run (see experimental section). At the time of writing, the necessary data import filters and methods to configure the system via configuration files exist. PathFinder exports its results into ASCII text files as well as into XML data files containing the pathways identified. A graphical user interface will be implemented in the near future.

Comparison with other systems. Several systems exist that include a search for pathways between two compounds s and t.

KEGG contains an algorithm to generate possible pathways of length k between two given compounds (Goto *et al.*, 1997). However, the resulting pathways are not organism specific and no other information than the EC number is used for computation. So the result can only give a rough overview of possible paths between s and t without biological meaning.

In Küffner *et al.* (2000), a very similar method for a pathway search is implemented. Additionally to the KEGG data, other data from BRENDA (Schomburg *et al.*, 2002) and ENZYME (Bairoch, 1999) are used. The actual algorithm is different from ours: First, a petri net containing all biochemical data is constructed. Then, all possible pathways of length k between s and t are computed, resulting in a huge number of putative pathways which then is reduced by several additional user defined and biologically motivated restrictions. In the last step, the remaining pathways are weighted by using expression data. Our algorithm starts by redefining the search space (see step 2 of algorithm description). The following edge weighting is based on sequence similarity and reduces the search space again. The resulting metabolic graph is quite small compared to the original search graph. The actual pathway search can then be performed efficiently.

In Goesmann *et al.* (2002), annotation data are parsed. Based on KEGG reference pathways, enzymes that were found in the annotation data are marked in the underlying graph as well as enzymes that were not mentioned in the annotation although given in the KEGG reference pathway. Possible subways are calculated and presented to the user. Subways are defined as acyclic subpaths between external nodes with external nodes being metabolites playing distinct roles in a metabolic pathway. It is the task of the user to find appropriate DNA or protein sequences in the organism of interest to support the pathway hypothesis of the system.

In aMAZE (as described in van Helden *et al.* (2001a)), the metabolic network containing data from sequence databases is also represented as a graph consisting of two types of nodes. In contrast to BioMiner, no roles describing reaction nodes exist. Edges only describe substrate-reaction and reaction-product relationships. With help of expression data, those reaction nodes that are supported by fitting enzymes are marked. The algorithm then tries to link all marked reaction nodes together with the respective substrate and product nodes to form meaningful pathways. These pathways are then compared to known pathways contained in KEGG, Eco/MetaCyc and aMAZE databases.

PathViewer

PathFinder results can be exported in XML[†] and loaded into PathViewer, the second application we built using BioMiner. To visually compare metabolic networks and verify found pathways, PathViewer can display arbitrary parts of biochemical networks for different organisms graphically at the same time for comparison purposes. PathViewer is based on yWays and has many features, among them automatic layout of diagrams, displaying of side compounds and enzymes can be switched on/off, custamizable set of visualized organisms, coloring of reactions according to organism, and highlighting of found pathways.

These features allow the user to interactively explore the data. To get an overview over the data the user can

[†] The XML format description (DTD) is available from our web site.

choose not to show any details, e.g. side compounds and enzymes can be hidden. This leads to a very compact representation of the metabolic network. By zooming, the user can focus on a special part of the pathway and, if deemed useful, details can be switched on. Furthermore, an additional window can be opened which shows details of the currently selected compound or reaction. The predicted paths can be highlighted in the network in various ways.

PathViewer and other systems. Compared to other systems, PathViewer combines several advantages and adds further functionality. A number of systems for visualizing metabolic pathways are based on existing libraries. Often, these libraries were not originally designed to visualize biochemical processes. E.g., the visualization component in Goesmann *et al.* (2002) is based on a tool implemented to draw compiler graphs and thus results in rather unusual images that biologists are not familiar with. This visualization is restricted to reactions with a single substrate and product, and it does not support cosubstrates or -products in the drawings.

Other approaches as given in Brandenburg *et al.* (2001) or Karp *et al.* (2000) come closer to biologists demands concerning the visualization of metabolic processes. But still, as far as we know, no system exists that would allow a comfortable analysis of metabolic data integrating the visualization of calculated results with the help of a dynamic, interactive viewer. Furthermore, most systems are restricted in the input data by allowing only one organism and/or one pathway to be chosen as source of input data. Finally, often only a small part of the metabolic network can be visualized in a certain level of detail.

See appendix for figures showing screenshots of PathViewer, displaying PathFinder results described in the next section.

EXPERIMENTAL RESULTS

Currently our main data source is KEGG[†]. KEGG includes all necessary genome and enzyme sequences. For convenience, in the following we give the respective RefSeq numbers for NCBI when introducing organisms.

We defined different test data sets on which we ran the PathFinder algorithm. In the following, we give a short overview of the results. Details can be found on our web site.

Dandekar *et al.* (1999) and Bork *et al.* (1998) discuss the plasticity of the glycolysis in various archaea and bacteria. By a comparative genome analysis and computation of elementary modes a number of alternative routes from glucose to pyruvate where identified by these groups. We used similar organism data as input. The maximum length of a pathway was set to ten reactions, leading from glucose (s) to pyruvate (t). To reduce the search space, the substrate set was chosen to be C_{res3}, thus excluding ubiquitous substrates and typical side metabolites. PathFinder results included the alternative pathways and detours given in Dandekar *et al.* (1999) and Bork *et al.* (1998). Additionally, several other variants were computed by PathFinder. Here we give a short description of two experiments we performed:

Experiment 1

In the first experiment, *H. pylori*[‡] served as query organism q. O_{ref} included four strains of *E. coli*[§]. PathFinder correctly stated that there is no phosphofructokinase (EC 2.7.1.11) present in *H. pylori*. This coincides with Dandekar *et al.* (1999) and Bork *et al.* (1998); the best sequence similarity found was less than 5 %. Thus, the standard glycolysis could not be found by PathFinder. Instead, a detour via the pentose phosphate pathway (PPP) was detected, similar to the detour described in Dandekar *et al.* (1999) for *M. hominis*[¶], starting at β-D-fructose 6-phosphate, leading via transketolase (EC 2.2.1.1) to D-Xylulose 5-phosphate and again via transketolase back to glycolysis, namely glyceraldehyde 3-phosphate. From thereon, Pathfinder states the conversion of glyceraldehyde 3-phosphate by standard pathway via glyceraldehyde 3-phosphate dehydrogenase (EC 1.2.1.12) to 3-phospho-D-glyceroyl phosphate, followed by phosphoglycerate kinase (EC 2.7.2.3) to 3-phospho-D-glycerate and phosphoglycerate mutase (EC 5.4.2.1) to 2-phospho-D-glycerate, and finally via phosphopyruvate hydratase (EC 4.2.1.11) to phosphoenolpyruvate.

As described in the above mentioned papers, *H. pylori* has no pyruvate kinase (EC 2.7.1.40) which would be the next step in standard glycolysis. Instead, PathFinder proposed an alternative reaction catalysed by phosphoenolpyruvate synthase (EC 2.7.9.2), converting phosphoenylpyruvate to pyruvate, which is actually a detour via the pyruvate metabolism. Figure 6 shows the result of experiment 1 graphically.

This pathway is given as the first and top ranked in the result list and thus the most likely pathway with a pathway weight $w(P) = 1$. It is therewith supporting the detours described in Dandekar *et al.* (1999). Detailed information can be found on our web site.

Experiment 2

In the second experiment we chose *H. influenza*[‖] to be q. O_{ref} included the same strains of *E. coli* as given in

[†] Version dated from 03/11/2002

[‡] Helicobacter pylori 26695, NCBI RefSeq NC_000915

[§] Escherichia coli K-12 MG1655, Escherichia coli K-12 W3110, Escherichia coli O157 EDL933, Escherichia coli O157 Sakai

[¶] Mycoplasma hominis

[‖] Haemophilus influenzae Rd, NCBI RefSeq NC_000907

experiment one, additionally *B. subtilis*, *T. pallidum*, *Synechocystis* and two strains of *H. pylori*[**]. The top ranked pathway with a pathway weight $w(P) = 1$ reflects the standard glycolysis (see Figure 7). Furthermore, the results contained variants of the PPP as described in Dandekar *et al.* (1999) and Bork *et al.* (1998), e.g. leading from glucose via glucokinase (EC 2.7.1.2) resp. hexokinase (EC 2.7.1.1) to α-D-glucose 6-phosphate, then entering the oxidative part of the PPP with glucose-6-phosphate 1-dehydrogenase (EC 1.1.1.49) leading via D-glucono-1,5-lactone 6-phosphate to 6-phosphogluconolactonase (EC 3.1.1.31) resulting in 6-phospho-D-gluconate, followed by phosphogluconate dehydrogenase (EC 1.1.1.44) to D-ribulose 5-phosphate. This is the starting point for the non-oxidative PPP: Ribose 5-phosphate epimerase (EC 5.3.1.6) leads to D-ribose 5-phosphate, followed by transketolase (EC 2.2.1.1). Figure 8 shows the generated pathway.

Another highly ranked pathway corresponds to the Entner-Doudoroff pathway (EDP), shown in Figure 9: Starting at α-D-glucose via glucokinase resp. hexokinase and via α-D-glucose 6-phosphate it leads with glucose-6-phosphate 1-dehydrogenase (EC 1.1.1.49) to D-glucono-1,5-lactone 6-phosphate. 6-phosphogluconolactonase (EC 3.1.1.31) outputs 6-phospho-D-gluconate. Then the two key enzymes for the EDP, phosphogluconate dehydratase (EC 4.2.1.12) and 2-dehydro-3-deoxyphosphogluconate aldolase (EC 4.1.2.14), follow. The pathway gets back to glyceraldehyde 3-phosphate, and the standard pathway is realized again.

Several variants of the described and other pathways from glucose to pyruvate were identified, too. We refer to our web site.

Similarly as stated in Dandekar *et al.* (1999), those results are theoretical and based only on sequence alignment. They have to be verified by experiments. Nevertheless, they can be used to direct the search for pathways in the right direction, whenever a genome of a newly sequenced organism becomes available. The integration of, e.g. expression data which will be implemented in the near future will lead to a further improvement of pathway prediction.

Complexity

Tables 1 and 2 give an overview of the time and space complexity based on the above described experiments[†].

The time consumption depends on several factors. Table 1 shows that the number of nodes and edges (and thus the search space) reduce noticable in step 1 by a factor of nearly five (nodes) resp. fourteen (edges). This is done by compressing *MG* and taking out side substrates and manually excluded compounds. E.g. in experiment 1 the number of nodes reduces from 13 970 to 2802, the number of edges goes down from 58 119 to 4209.

Table 1. Space and run time of the experiments performed, concerning step 1 (construction of compressed graph MG_c) in seconds (s) resp. number of nodes/edges

Step	**Experiment 1**	**Experiment 2**
Build *MG* (s)	0.4	0.68
# nodes in *MG*	13,970	19,617
# edges in *MG*	58,119	84,995
Build MG_c (s)	30.27	47.50
# nodes in MG_c	7,729	10,1929
# edges in MG_c	21,955	29,137
Reduce C in MG_c to C_{res1} (s)	0.09	0.1
# remaining nodes	2,822	2,990
# remaining edges	4,509	4,838
Reduce C in MG_c to C_{res3} (s)	0.009	0.01
# remaining nodes	2,802	2,969
# remaining edges	4,209	4,513

Table 2 points out how the time complexity for the pathway search benefits from step 2 of the algorithm. Reducing the number of nodes and edges leads to a significant performance improvement. E.g. in the case of $k = 10$ the number of remaining nodes in MG_c can be reduced from 2802 to 1097 in only 0.05 seconds. The weighting step reduces from 54.6 minutes to 36 minutes. The pathway search needs 6.76 seconds instead of 8.06 seconds. The main reason for the time saving is the number of alignments to be performed: When MG_c is not reduced by step 2, the number of nodes and edges is much larger than in the case that step 2 is applied. Furthermore, the reaction nodes MG_c are *compressed*, i.e. each reaction node can comprise more than one biochemical reaction resulting in a number of alignments necessary per node. In the case of $k = 10$, the number of necessary alignments without step 2 is 2672 (see table 2), when step 2 was applied this number reduced to 1766. Table 2 shows that the time for the pathway search and the number of generated pathways grow exponentially in k.

Limitations

PathFinder still has some limitations which we describe in the following paragraphs.

Correctness of data. Because KEGG is the only database used by PathFinder at the time of writing, the algorithm heavily depends on the correctness of the data in this database. Importing incorrect data leads to generating biochemically incorrect pathways. This cannot be pre-

[**]Helicobacter pylori 26695, Helicobacter pylori J99

[†] System: UltraSPARC III CPU, 750 Mhz. 8 Gbyte RAM. gcc compiler, v2.95.3 on SOLARIS 8, optimizing flag O2.

Table 2. Space and run time of experiment 1, refining search space (step 2), weighting and pathway generation (step 3)

	step 2 applied		without step 2	
	$k = 10$	$k = 11$	$k = 10$	$k = 11$
run time step 2 (s)	0.06	0.07		
# nodes in MG_C	1,097	1,186	2,802	2,802
and edges after step 2	2,139	2,291	4,209	4,209
weighting (s)	2158	2400	3274	3274
# alignments	1766	1957	2672	2672
Pathway search (s)	6.76	24.61	8.06	26.08
# pathways	36,526	117,392	36,526	117,392

vented automatically, and it is the users task to evaluate the PathFinder results to exclude mistaken pathways. By integrating other data sources, the algorithm will become more robust to this kind of 'noise' in the input data.

Pathway length. The complexity of the PathFinder algorithm grows exponentially in k, the maximal length of generated pathways. Run time depends furthermore on how 'dense' the metabolic network is, i.e. how may reactions exist connecting the substrate nodes. If there are a lot of variants to convert one substrate into another, the number of putative pathways grows very fast by increasing k. Generally a number of round about 10 reactions should be sufficient to get meaningful biochemical pathways. This is a number which still results in resonable runtime and number of generated pathways.

CONCLUSION

In this paper we present BioMiner, a software system for representing, analyzing and visualizing diverse biochemical data. Its underlying comprehensive data model BioCore is capable of integrating complex data and contains powerful methods to gain new biological information. PathViewer as application of the visualization component yWays allows the comparison of metabolic information of different organisms. We presented first applications of BioMiner and its data model and showed the usability of the contained graph structures to represent and analyze biochemical networks. PathFinder and PathViewer demonstrate that powerful applications can be rapidly implemented with the help of BioMiner. BioMiner will be extended by integrating methods for target identification and comparison of cells in normal and disease (e.g. cancer) states. The weighting function will be extended by several other parameters, e.g. expression data and stoichiometry. Furthermore, including calculation of elementary modes can help to improve the pathway prediction of PathFinder.

Although the data model is already capable of representing the most important regulatory processes, new classes for more complex regulatory components are under construction. Import functions for expression data from DNA micro arrays and proteomics data will be implemented soon. At the time of writing, interfaces to BIND (Bader *et al.*, 2001), DIP (Xenarios *et al.*, 2002), TRANSFAC (Wingender *et al.*, 2001) and TRANSPATH (Schacherer *et al.*, 2001) are in development.

ACKNOWLEDGEMENTS

The authors would like to thank Kurt Mehlhorn for fruitful discussions. We are further grateful to Hans Werner Adolph, Vicki Kelly and Michaela Falb for support in biochemical questions. This work has been supported by the Max-Planck-Institut für Informatik and the Klaus-Tschira-Stiftung.

REFERENCES

Altschul,T.L. *et al.* (1997) Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Appel,R. *et al.* (1994) A new generation of information retrieval tools for biologists: the example of the expasy www server. *Trends Biochem. Sci.*, **19**, 258–260.

Arita,M. (2000) Graph modeling of metabolism. *J. JSAI*, **15**, 703–710.

Arita,M. *et al.* (2000) Reconstructing metabolic pathways with new enzyme classification. In *Proceedings of the German Conference on Bioinformatics (GCB 2000)*. pp. 99–106.

Bader,G. *et al.* (200)) BIND—the biomolecular interaction network database. *Nucleic Acids Res.*, **29**, 242–245.

Bairoch,A. (1999) The ENZYME data bank in 1999. *Nucleic Acids Res.*, **27**, 310–311.

Bastert,O. and Matuszewski,C. (2001) Layered drawings of digraphs. *Drawing Graphs: Methods and Models*. LNCS Tutorial, Springer, pp. 104–139.

Booch,G. *et al.* (eds) (1999) *The Unified Modelling Language User Guide*. Addison Wesley.

Bork,P. *et al.* (1998) Predicting function: from genes to genomes and back. *J. Mol. Biol.*, **283**, 707–725.

Brandenburg,F.J. *et al.* (2001) BioPath: visualization of biochemical pathways. In *Proceedings of the German Conference on Bioinformatics (GCB 2001)*. pp. 11–15.

Dandekar,T. *et al.* (1999) Pathway alignment: application to the comparative analysis of glycolytic enzymes. *Biochem. J.*, **343**, 115–124.

Di Battista,G. *et al.* (1999) *Graph drawing: algorithms for the visualization of graphs*. Prentice-Hall, New Jersey.

Ellis,L. *et al.* (2001) The University of Minnesota biocatalysis/biodegradation database: emphasizing enzymes. *Nucleic Acids Res.*, **29**, 340–343.

Gallo,G. *et al.* (1993) Directed hypergraphs and applications. *Discrete Appl. Math.*, **42**, 177–201.
Goesmann,A. *et al.* (2002) Pathfinder: reconstruction and dynamic visualization of metabolic pathways. *Bioinformatics*, **18**, 124–129.
Goto,S. *et al.* (1997) Organizing and computing metabolic pathway data in terms of binary relations. In *Pacific Symposium on Biocomputation (PSB) 1997*. pp. 175–186.
Hofestädt,R. and Thelen,S. (1998) Quantitative modeling of biochemical networks. In *Silico Biol.*, **1**, 39–53.
International Human Genome Sequencing Consortium (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.
Jeong,H. *et al.* (2000) The large-scale organization of metabolic networks. *Nature*, **407**, 651–654.
Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, **28**, 27–30.
Karp,P. *et al.* (2000) The EcoCyc and MetaCyc databases. *Nucleic Acids Res.*, **28**, 56–59.
Küffner,R. *et al.* (2000) Pathway analysis in metabolic databases via differential metabolic display (DMD). *Bioinformatics*, **16**, 825–836.
Mehlhorn,K. and Naher,S. (1999) *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge.
Mendes,P. *et al.* (2000) PathDB: a second generation metabolic database. In *BTK 2000 (Minibook)*. pp. 207–212.
Michal,G. (1993) *Biochemical Pathways Poster*. Boehringer Mannheim GmbH.
Overbeek,R. *et al.* (2000) WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic Acids Res.*, **28**, 123–125.
Paton,N.W. *et al.* (2000) Conceptual modelling of genomic information. *Bioinformatics*, **16**, 548–557.
Rzhetsky,A. *et al.* (2000) A knowledge model for analysis and simulation of regulatory networks. *Bioinformatics*, **16**, 1120–1128.
Schacherer,F. *et al.* (2001) The TRANSPATH signal transduction database: a knowledge base on signal transduction networks. *Bioinformatics*, **17**, 1053–1057.
Schomburg,I. *et al.* (2002) Brenda, enzyme data and metabolic information. *Nucleic Acids Res.*, **30**, 47–49.
van Helden,J. *et al.* (2000) Representing and analyzing molecular and cellular function using the computer. *Biol. Chem.*, **381**, 921–35.
van Helden,J. *et al.* (2001a) Application of regulatory sequence analysis and metabolic network analysis to the interpretation of gene expression data. In *JOBIM 2000*, LNCS, Springer, pp. 147–163.
van Helden,J. *et al.* (2001b) From molecular activities and processes to biological function. *Brief Bioinform.*, **2**, 81–93.
Venter,J. *et al.* (2001) The sequence of the human genome. *Science*, **291**, 1304–1351.
Wiese,R. *et al.* (2001) yFiles: visualization and automatic layout of graphs. In *Proceedings of the 9th International Symposium on Graph Drawing*, LNCS, Springer.
Wingender,E. *et al.* (2001) The transfac system on gene expression regulation. *Nucleic Acids Res.*, **29**, 281–283.
Wittig,U. and De Beuckelaer,A. (2001) Analysis and comparison of metablic pathwaydatabases. *Briefings in Bioinformatics*, **2**, 126–142.
Xenarios,I. *et al.* (2002) DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.*, **30**, 303–305.

APPENDIX

DEFINITIONS

In the following we give definitions that are important in the context of biochemical pathway modeling.

The most important term in this article may be *pathway*. Several definitions exist, and it seems to depend heavily on the viewpoint how this term is defined. From our point of view, two definitions have to be distinguished:

DEFINITION 1. A *biochemical pathway* is a sequence of events or reactions that lead from one compound to another. This comprises *metabolic* and *regulatory* pathways. The length k of a biochemical pathway is the number of its events or reactions.

The analysis part in our algorithm generates biochemical pathways in the sense of definition 1. Often, the term pathway is used in a more general meaning which is reflected by the next definition:

DEFINITION 2. A *textbook pathway* is a general, *named* pathway as given in a tyipcal biochemical textbook. It describes the general set of reactions that have to be present in an organism to realize this pathway. A textbook pathway is generally organism independent and could be seen as a kind of *reference pathway*.

Examples for textbook pathways are glycolysis or TCA cycle.

KEGG (Kanehisa and Goto, 2000) pathways are pathways in sense of definition 2: KEGG contains a set of predefined pathway maps (GIF images and corresponding database files), each describing the set of enzymes necessary for the pathway to be realized. To map a KEGG reference pathway onto an organism, the user can choose an organism from a list, thereby instantiating the pathway.

Further important terms in the context of biochemical pathways are *graph* and *network* which are often used as synonyms. Mathematically, one would state:

DEFINITION 3. A *network* (*directed network*) is a weighted graph (digraph).

We use the term network to describe the whole metabolic and regulatory network present in an organism. With the term graph we describe the technical representation of this network.

Mathematicians also use the term *path*:

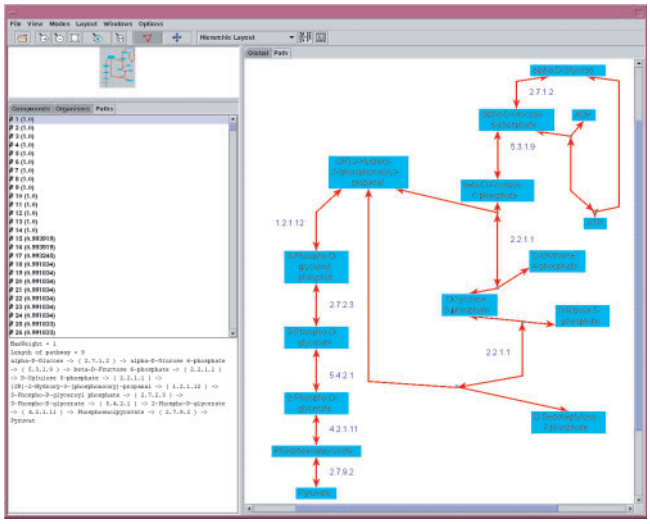

Fig. 6. PathViewer visualizing the results of PathFinder on the glycolysis (experiment 1): PathViewer offers a number of options, among them *reaction clustering* and *show EC numbers/co-compounds on/off*. In default modus, reactions are clustered according to their EC number, i.e. all edges with the same EC number are collapsed into a single edge. In the above case, the reactions are clustered and displayed in red. EC numbers are shown next to the respective reaction edge. Co-compounds are hidden. If clustering is disabled, each organism is coded with a certain color used for each reaction that belongs to this organism. The user can select which organisms are displayed. Furthermore, different *views* of PathFinder results are possible, e.g. a *global* and a *path* view. In the global view, the whole metabolic network (*MG*) is displayed. When the user chooses a pathway from the list of found pathways, all according reactions are highlighted in the graph. The user can browse through the list to see the differences between the found pathways. In the path view, only those reactions and substrates are shown that actually belong to the chosen pathway. Further subwindows show, amongst other things, the pathway as a list in the form substrate - EC number - substrate, the list of organisms, the list of all compounds with s and t being marked, an overview window, a reaction window where a selected reaction is shown in detail, and a compound window with detailed informations for a chosen compound. The user can select from different layout algorithms and a number of different parameters. The graph can interactively be rearranged by clicking an object in the graph and dragging it to a new position. Several tools for zooming, printing, image export and graph browsing are available.

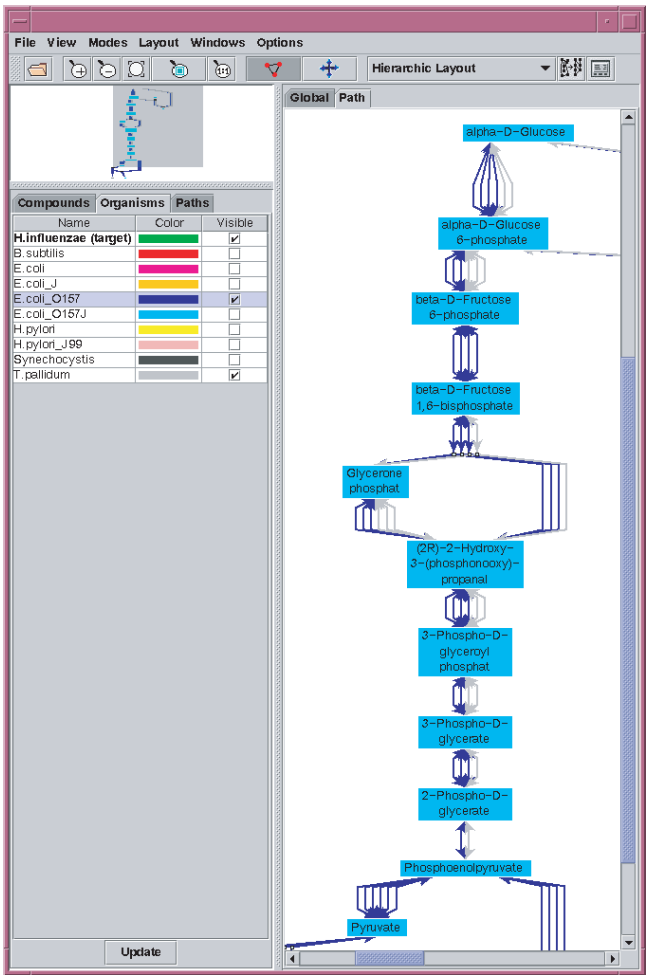

Fig. 7. Top ranked pathway for experiment 2: Here, only reactions from *E. coli O157* and *T. pallidum* are displayed. The pathway represents the standard glycolysis converting α-D-glucose to pyruvate.

DEFINITION 4. A *path* in a graph from vertex s to vertex t is a sequence of edges (s, v_i), $(v_i, v_{i+1}),\ldots(v_{k-2}, v_{k-1})$, (v_{k-1}, t) such that s is the start node and t is the target node. The above path has length k.

We do not distinguish between paths and pathways; following definition 1 a pathway in our meaning is a sequence of k events or reactions, leading from s to t. Although the algorithm includes weighting mechanisms and a scoring funtion to score the generated pathways, it is the task of the user to evaluate if the putative pathways are also *meaningful* and perhaps even textbook pathways.

We can assign weights not only to single edges but also to pathways and graphs:

DEFINITION 5. The weight of a graph (pathway) is the sum of the edge weights of the graph (pathway).

A further distinction is necessary between the internal graph on which the analysis algorithms work, and the graph used for visualization. Both are distinct objects. The internal graph is a *multigraph*, while the vizualisation graph is a *hypergraph*:

DEFINITION 6. A *(directed) hyperedge* is an (di-

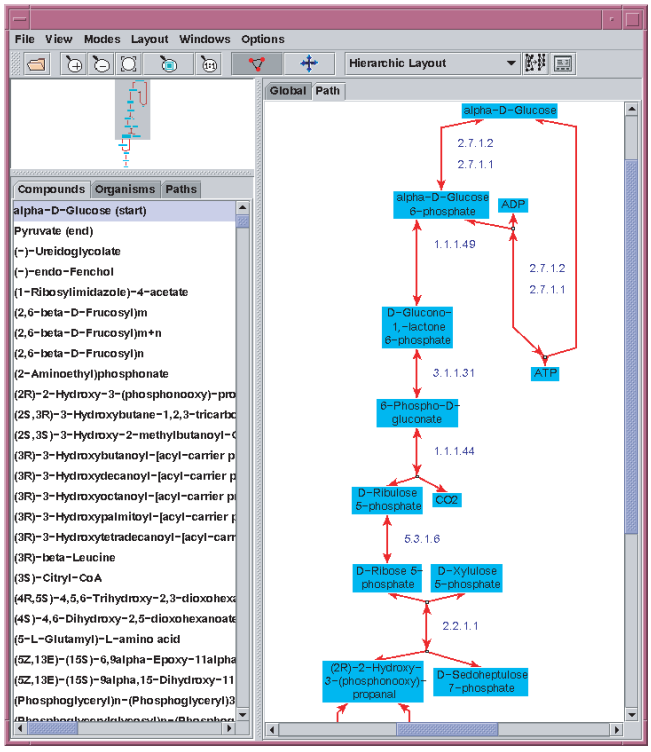

Fig. 8. Pathway result of experiment 2: This shows the oxidative and the non-oxidative part of the pentose phosphate path as described in the text. Reaction clustering was activated, enzyme numbers are shown.

rected) edge which connects two or more vertices. A *(directed) hypergraph* is a graph consisting of (directed) hyperedges.[†]

DEFINITION 7. A *directed multigraph G* is a graph whose edges are unordered pairs of vertices, and the same pair of vertices can be connected by multiple edges.

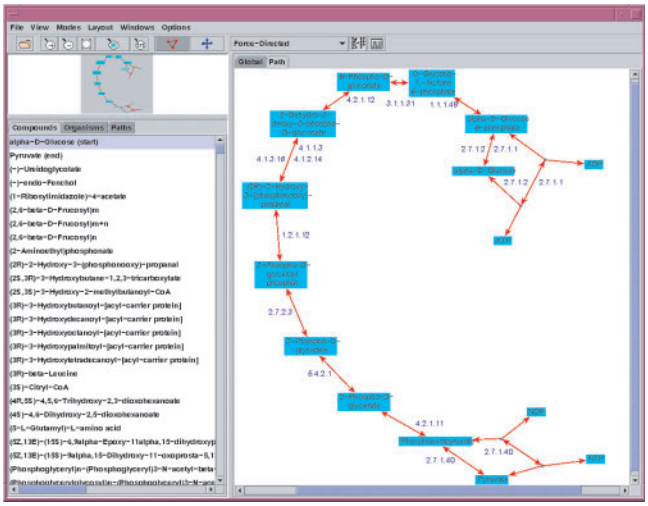

Fig. 9. Pathway result of experiment 2: Shown is the Entner-Doudoroff-Pathway as described in the text. In the above picture, instead of *hierarchic layout* mode, *force-directed* layout was chosen.

Using directed multigraphs to represent the biochemical network in the computer enables us to model, e.g. that a compound can take part in several reactions with different roles.

Our algorithms can cope with graphs that contain *cycles*:

DEFINITION 8. A *cycle* is a path(way) starting and ending on the same vertex (compound).

The algorithm takes care of only looking for *simple cycles*:

DEFINITION 9. A *simple cycle* is a cycle in which all vertices are visited exactly once except the starting vertex which is visited only twice.

This is, e.g. the case for metabolic pathways like TCA cycle.

[†] For more details of directed hypergraphs see, e.g. Gallo *et al.* (1993).