Shortest Paths with Pairwise-Distinct Edge Labels:
Finding Biochemical Pathways in Metabolic Networks

Sándor Fekete* Tom Kamphans* Michael Stelzer†

Abstract

A problem studied in Systems Biology is how to find shortest paths in metabolic networks. Unfortunately, simple (i.e., graph theoretic) shortest paths do not properly reflect biochemical facts. An approach to overcome this issue is to use edge labels and search for paths with distinct labels.

In this paper, we show that such biologically feasible shortest paths are hard to compute. Moreover, we present solutions to find such paths in networks in reasonable time.

Key words: Shortest path, NP-completeness, edge label, bioreaction database, metabolic network, pathway.

1 Introduction

In the past decade, a lot of work has been done in the field of bioreaction databases, which are a powerful tool for studying biochemical processes (see, for example, Francke et al. [12]). Examples for such databases are the *Roche wall chart of Biochemical Pathways* [23], the *BioCyc* databases (e.g. [4]), the *Kyoto Encyclopedia of Genes and Genomes* databases (*KEGG*) [19] and the *BRaunschweig ENzyme DAtabase BRENDA* [31].

*Braunschweig University of Technology, Department of Computer Science, Algorithms Group, 38106 Braunschweig, Germany. s.fekete@tu-bs.de, tom@kamphans.de, michaelstelzer@freenet.de

†Helmholtz Centre for Infection Research (HZI), Systems Biology, 38124 Braunschweig, Germany

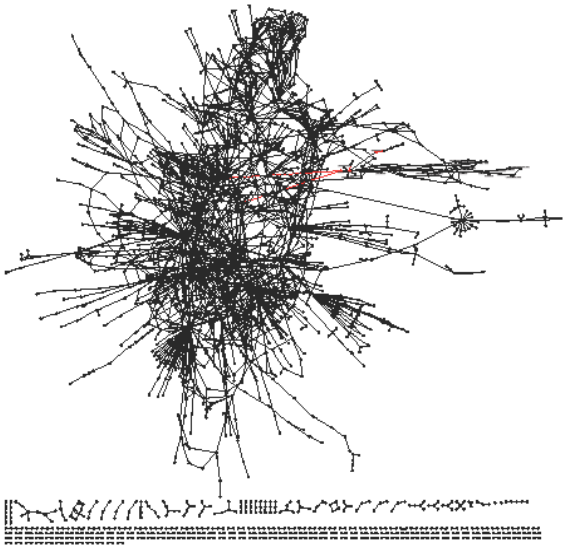

Figure 1: Example of an organism-specific metabolic network (*H. sapiens*), reconstructed from a bioreaction database [32].

Biochemical reaction databases allow the reliable reconstruction of metabolic networks, see Figure 1, which—in turn—help to improve the annotation of genome sequences to interpret high-throughput omics data, to understand biological processes of pathogenesis or industrial production at a system level, and to rationally control or design biological systems (e.g., [9, 16, 28, 34]).

Such databases are subject to perpetual research and update: New insights lead to extensions and corrections, resulting in more accurate databases and more detailed views on biological aspects. Under certain circum-

stances, database updates can be evaluated by considering shortest paths in metabolic networks that have been reconstructed from the database [21, 29, 30]. In the case of a metabolic network, a shortest path describes the number of necessary reactions to convert one metabolite into another—provided that the network does not contain so-called currency metabolites, which would introduce biologically infeasible shortcuts. Ma and Zeng [21] developed such a database, and demonstrated its usefulness by reconstruction of high-resolution metabolic networks. In this database, Ma and Zeng defined the reversibility of the reactions according to literature data and biochemistry knowledge and introduced the concept of reactant pairs by considering currency metabolites. (The background for our work is a recent upgrade of this database [32].)

Furthermore, shortest paths are of particular interest in organism-specific metabolic networks, because they can help to find alternative pathways (e.g., [27]). The tool *gapFiller* uses shortest paths to detect gaps that occur in biochemical pathways (i.e., biomass that is produced by an organism but not documented in the network due to a lack of complete annotation) [20]. Additionally, shortest paths can be used to compute centrality indices [3].

From a computer science point of view, shortest paths in a graph are easy to compute using *Breadth-First Search* or *Dijkstra's algorithm*; see, for example, Cormen et al. [7].

However, simple (i.e., graph-theoretic) shortest paths do not properly reflect biochemical facts: Biochemical pathways in metabolic networks lead from one node (representing a metabolite) to another one. The edges passed on this pathway represent biochemical reactions that convert a substrate to a product. Given a reaction that leads from a substrate to two different products and back–such as from *2-dehydro-3-deoxy-D-galactonate 6-phosphate* (KEGG compound ID C01286) to *glyceraldehyde 3-P* (C00118) and *pyruvate* (C00022)—it is possible to find a shortest path that leads from one product via the substrate to the other one. One way to avoid this effect is to store the reaction types as labels on the edges of the network, and ensure that a shortest path passes no label twice [29, 30].

Thus, this work considers shortest paths with pairwise-distinct edge labels for computing *biologically feasible* shortest paths (in metabolic networks that have been reconstructed based on a database without currency metabolites and with reversibility information). In particular, we show in Section 3.1 that this kind of shortest paths is hard to compute.

Related Work

Shortest paths have been considered in many settings and applications such as robotics and traffic optimization. For an overview see the survey articles by Mitchell [24, 25]. If the graph is unknown, common solutions are the A^*- or D^*-algorithms [14, 33]. Fleischer et al. [11] give a general framework for computing short paths for settings where the environment as well as the location of the target is unknown.

For finding biologically feasible shortest paths, the approach by Faust et al. [10] is to take the chemical structure of reactants into account to differentiate between side and main compounds of a reaction. Croes et al. [8] use shortest paths in networks where a compound is assigned a weight equal to its number of incident edges. Finding pathways based on atomic transfers was presented by Boyer and Viari [2] and Heath et al. [15]. McShan et al. [22] use heuristic search methods for finding metabolic pathways. Kaleta et al. [17] compute elementary flux patterns to detect pathways.

Independently from our work, Chakraborty et al. [5] show that it is NP-hard to find the minimum number of labels such that any two

vertices are connected by a *rainbow path* (i.e., a path with pairwise distinct edge labels).

There are many tools available for computing shortest paths and analyzing biological networks in various contexts. The tool *Pajek* [1] has its roots in Social Sciences. The origin of *Cytoscape* (e.g., [6]) is Systems Biology. The main focus of both tools is on network visualization but both allow also some analysis of networks. Another program is the *Pathway Hunter Tool* [26]. All these tools allow for the computation of shortest paths but none is—so far—able to compute shortest paths with unique labels. Software for computing such paths is available from the authors [18].

2 Background from Computer Science

In this section, we briefly review some basics in computer science for readers with more background in biology than in computer science.

2.1 Modeling Metabolic Networks

Mathematically, a metabolic network is a graph, so let us briefly review some terms from graph theory. A graph consists of a set of *vertices* or *nodes*. Relations between two nodes are modeled by edges between nodes. If relations are unidirectional (i.e., the edges are arrows), the graph is called *directed* graph.

In our case, the nodes represent the metabolites, the edges model reactions converting one metabolite into another. Usually, reactions are reversible; that is, the corresponding edges are undirected. However, there are a few cases where the reverse direction is not feasible from thermodynamic aspects. As directed graphs are easier to handle than mixed graphs, we use only directed edges, and model reversible reactions by two oppositely directed edges.

A *path* in a graph is a sequence of edges, where—except for the first edge—every edge starts in that node where the preceding edge ends. The *length* of a path is defined as the number of edges on the path (if the edges have no weights). Consequently, the shortest path between two given nodes in a graph is the path having the smallest number of edges.

2.2 Shortest Paths in Unlabeled Networks

Given a graph, $G = (V, E)$, it is quite simple to compute the shortest path between two vertices of the graph or even from a given vertex to every other one. *Breadth-First Search* (BFS), see Algorithm 1, is known to every first-grade student of computer sciences. Even if the edges have different length, the problem is still easy to compute using *Dijkstra's algorithm* (e.g., Cormen et al. [7]).

Note that BFS creates a tree of all shortest paths by storing for every vertex, v, its predecessor, *v.father*, on the shortest path to the start vertex. Thus, a shortest path from a given vertex to the start vertex can be found simply by following the *father* pointers.

```
Let Q be a queue of vertices
insert start vertex into Q
while Q is not empty do
    v := first vertex from Q
    remove first vertex from Q
    for all vertices v' adjacent to v do
        if v' was not visited before then
            v'.father := v
            mark v' as visited
            append v' to Q
            report shortest path to v'
        end if
    end for
end while
```

Algorithm 1: Breadth-First Search (BFS)

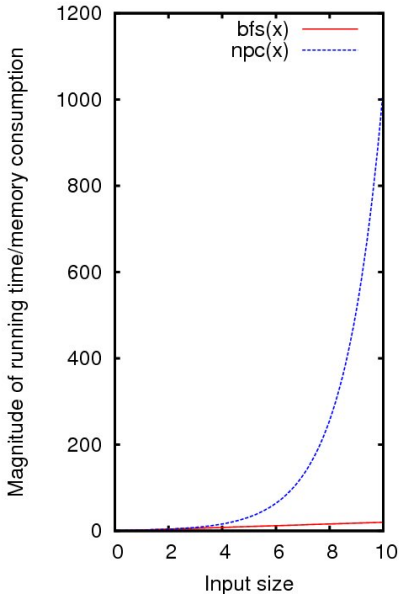

Figure 2: Magnitude of resource request for an NP-complete problem compared to BFS.

2.3 Computational Complexity

As we will see in the following section, things get much harder in our case. To express the hardness of a problem, it is very common in computer science to estimate the order of growth in the resources (i.e., running time and memory requirements) needed by a program as a function in the input size. In our case, the input size is the number of vertices and edges in the graph. While the running time of BFS is linear in the input size (that is, for example, if we double the input size, the running time doubles also), we can show that the running time is most likely to be exponential for shortest paths with unique labels; that is, if we have a graph with v vertices and e edges, the running time is in the order of 2^{v+e}. More precisely, we can show that our problem belongs to the class of *NP-complete* problems [13]. It is a widely held belief that there is no sub-exponential solution for NP-complete problems. The impact of this running time is shown in Figure 2.

NP-completeness is usually shown using a common technique in computer science known as proof by *reduction*. That is, we take a well-known hard problem—in our case 3-SAT—and show that this problem would be easy to solve if our problem *shortest path with unique labels problem* (SPUL), as defined in the following section, would be easy to solve. This is done by describing how to translate an input to 3-SAT to SPUL such that a solution to SPUL yields a solution to 3-SAT.

Given a set of *binary variables*, $x_1, \ldots, x_n$, and a set of *clauses*, $C_1, \ldots, C_m$, consisting of three literals (i.e., $C_i = L_{i1} \vee L_{i2} \vee L_{i3}$, where L_{ik} denotes a negated ($\overline{x_k}$) or unnegated variable (x_k)), the problem *3-SAT* asks if there is a assignment of $x_1, \ldots, x_n$ to 0 or 1 such that all clauses are fulfilled [13].

3 Shortest Paths with Unique Labels

Let G be a directed graph (in our case a metabolic network) that consists of a set of vertices (metabolites), V, and a set of edges (reactions), E. Even if G does not contain currency metabolites, which would introduce biologically infeasible shortcuts, a shortest path in G may still be biologically infeasible, because it may use the same reaction twice, as explained in Section 1.

Thus, we are interested only in shortest *feasible* paths; that is, shortest paths from s to t with distinct reaction types. Such a path may be longer than the overall shortest path, see Figure 4. To store the reaction types in the graph, we use *labels* for the edges. Altogether, a mathematical model for our problem is:

Problem *Shortest Path with Unique Labels* (SPUL): Given a graph $G = (V, E)$, a mapping $\ell : E \longrightarrow \mathbb{N}$ that assigns a label to every edge, and two vertices, $s \in V$ and $t \in V$, find a shortest path $P = (e_1, e_2, ..., e_k)$ starting in

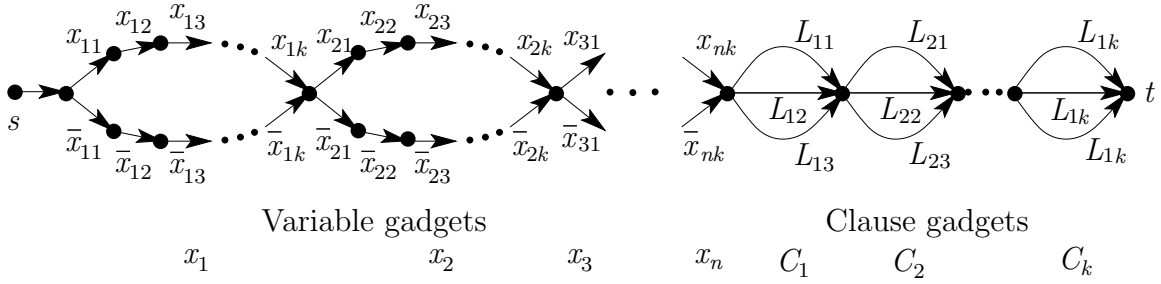

Figure 3: Transforming a 3-SAT instance to an SPUL instance.

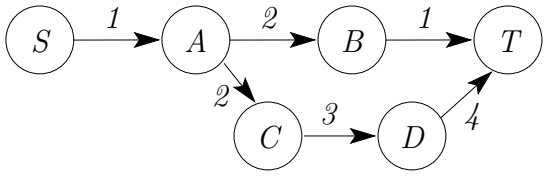

Figure 4: The shortest path from S to T is $S \xrightarrow{1} A \xrightarrow{2} B \xrightarrow{1} T$, but this path is infeasible, because it passes the label "1" twice. The shortest feasible path is $S \xrightarrow{1} A \xrightarrow{2} C \xrightarrow{3} D \xrightarrow{4} T$.

s and ending in t with pairwise distinct edge labels; that is, for $1 \leq i < j \leq k : \ell(e_i) \neq \ell(e_j)$.

3.1 Computational Complexity

In an unlabeled network, we can simply apply BFS. Things get considerably harder, if we add labels to the edges and require that no label is passed twice on a path between two vertices. Note that neither BFS nor Dijkstra's algorithm are able to find the shortest feasible path shown in Figure 4.

Theorem 1 *Given a graph $G = (V, E)$ with a mapping $\ell : E \longrightarrow \mathbb{N}$ that assigns a label to every edge, it is NP-complete to determine if there is a path $P = (e_1, e_2, \ldots, e_k)$ that uses every label at most once; that is, for $1 \leq i < j \leq k : \ell(e_i) \neq \ell(e_j)$.*

Proof. We show our theorem by reduction from 3-SAT. A 3-SAT instance can be transformed to a SPUL instance as follows: For every clause C_i we use a clause gadget that consists of three parallel edges labelled with L_{ik} (i.e., with x_{ik} or $\overline{x_k}$ for an unnegated or negated variable, respectively). The variable gadget for variable x_j consists of two parallel paths, one with all negated labels, one with all unnegated labels. For the whole input, we start with a vertex, s, and add all variable gadgets followed by all clause gadgets. The last vertex is labeled t; see Figure 3. To find a path from s to t, we have to pass either the negated or the unnegated branch for every variable. Thus, after passing the variable gadgets we have either all negated or all unnegated variables left to pass the clause gadgets without using a label twice. This is possible if and only if the given formula is satisfiable. □

3.2 Computing Shortest Path with Unique Labels

3.2.1 A Memory-Consuming Solution

We use a modified version of BFS to solve our problem, see Algorithm 2. Similar to BFS, we store all paths found so far. But instead of storing a shortest-path tree for the vertices as in BFS, we construct a shortest-feasible-path tree on the edges of the graph, see Figure 5. That is, we store every possible feasible path leading to a vertex in the tree during the search. When the search reaches a vertex, v, via an

Let Q be a queue of edges
insert "dummy edge" to start vertex into Q
while Q is not empty **do**
  e := first edge from Q
  remove first edge from Q
  for all edges e' adjacent to $e.target$ **do**
    if $e'.label$ was not used on the shortest path from s to e **then**
      $e'.father := e$
      append e' to Q
      if $e'.target$ was not visited before **then**
        report shortest path to $e'.target$
      end if
    end if
  end for
end while

Algorithm 2: Shortest Path with Unique Labels

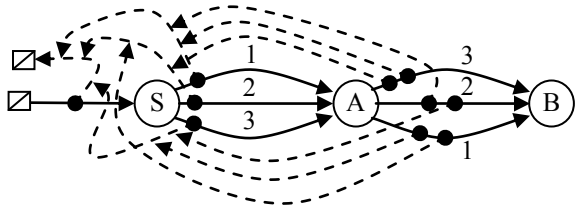

Figure 5: An example for a shortest feasible path tree constructed by Algorithm 2. The network consists of three nodes (S, A, and B), three edges from S to A, and three edges from A to B. The dashed lines show the shortest feasible path tree.

edge, e', we can determine if the path to v via e' is feasible (i.e., no label occurs twice). By the BFS-manner of this algorithm, the first feasible path found to a vertex is also the shortest feasible path. The drawback is that the algorithm is quite memory consuming, because it stores all feasible paths to all vertices.

3.2.2 Balancing Time and Memory Requirements

To save memory, we tried a different solution by exploiting the fact that a metabolic network has many parallel edges. Thus, our search does not have to explore all edges incident to a vertex, but only those edges that lead to different vertices. Instead of storing one label per edge on a shortest path, we store a set of labels. This significantly decreases the number of shortest paths that we have to store. The drawback is that we have to find a feasible combination of labels when the search progresses (i.e., when we want to add a new edge to a path). This can be solved by a simple backtracking; that is, we successively test combinations of labels until we either find a feasible combination or no more combination is possible. The idea is that in most cases this backtracking may not require much time, because a feasible combination is found quickly. Only if there is no feasible combination, we have to test all of them. Clearly, this heavily depends on the structure of the input network.

3.2.3 Preprocessing

Before we start our algorithm, we perform a simple BFS to determine, which vertices can be reached at all (i.e., there is a feasible or infeasible path). We store these vertices to be able to abort the search as soon as feasible paths to all of them have been found. In a second stage of the preprocessing, we perform a simple BFS again, this time checking if the found path is feasible and reporting feasible paths.

3.2.4 Comparison

Table 1 shows results for large databases. There was not sufficient memory to compute all paths. Thus, we compare the number of paths found until the program was stopped because there was no memory left. It turned out

Table 1: Examples of running time and found paths starting in vertex 1 (XEON CPU 3.0 GHz, 16 GB memory).

Network	Vertices in G	Edges in G	Vertices reachable from start	Paths found			Running time in min		
				Alg. A	Alg. B	B with preproc.	Alg. A	Alg. B	B with preproc.
A. niger	2547	7818	1488	1283	1369	1382	1.2	23.4	23.6
E. coli	1895	5525	1111	911	1012	1021	1.3	35.6	35.9
H. sapiens	2474	7873	1614	1347	1507	1526	1.2	20.2	20.5

Table 2: Comparison on the number of paths found in several organism-specific metabolic networks: Total number of shortest paths (SP), number of correct/infeasible shortest paths found by BFS, number of shortest paths with unique labels (SPUL).

	$\text{uur}^{-\text{cm}}$	$\text{uur}^{+\text{cm}}$	$\text{mpn}^{-\text{cm}}$	$\text{mpn}^{+\text{cm}}$	$\text{bbu}^{-\text{cm}}$	$\text{bbu}^{+\text{cm}}$	$\text{mge}^{-\text{cm}}$	$\text{mge}^{+\text{cm}}$
SP	2372	17038	2191	6652	6357	39552	6793	36246
Correct SP	1259	9302	1288	3929	2470	16864	2868	18460
Infeasible SP	1113	7736	903	2723	3887	22688	3925	17786
SPUL	1308	13306	1601	4577	2513	22809	3061	22281

uur = *U. urealyticum*, mpn = *M. pneumoniae*, bbu = *B. burgdorferi*, mge = *M. genitalium*,
+cm/-cm: with/without currency metabolites

that the memory consuming solution (Alg. A) is much faster than our second approach, but finds less paths. The preprocessing with BFS further improves the number of found paths. Smaller organisms are compared in Table 2: We compared the number of (graph theoretically) shortest paths (SP) to the number of shortest paths with unique labels (SPUL). Furthermore, we listed the number of correct shortest paths and the number of biologically infeasible shortest paths (i.e., paths such as S$\longrightarrow$ A$\longrightarrow$ B $\longrightarrow$ T in Fig. 4) that were found using BFS.

4 Conclusion

We showed that the problem of finding paths pairwise-distinct edge labels in (directed) graphs is NP-hard. A straightforward modification of BFS to this problem results in a memory-consuming solution that allows for computing only small instances. Balancing time- and memory requirements, we are able to deal with larger instances.

Acknowledgements

This work was financially supported by the *German Research Foundation* (*DFG*), project *SFB 578* and by *Federal Ministry of Education and Research* (*BMBF*), project *InterGenomics*. Tom Kamphans was supported by 7th Framework Programme contract 215270 (FRONTS).

References

[1] V. Batagelj and A. Mrvar. Pajek – program for large network analysis. *Connections*, 21:47–57, 1998.

[2] F. Boyer and A. Viari. Ab initio reconstruction of metabolic pathways. *Bioinformatics*, 19:ii26–ii34, 2003. DOI: 10.1093/bioinformatics/btg1055.

[3] U. Brandes. A faster algorithm for betweenness centrality. *J. Math. Sociology*, 25:163–177, 2001.

[4] R. Caspi, T. Altman, J. M. Dale, K. Dreher, C. A. Fulcher, F. Gilham, P. Kaipa, A. S. Karthikeyan, A. Kothari, M. Krummenacker, M. Latendresse, L. A. Mueller, S. Paley, L. Popescu, A. Pujar, A. G. Shearer, P. Zhang, and P. D. Karp. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Research*, 38:D473–D479, 2010. DOI:10.1093/nar/gkp875.

[5] S. Chakraborty, E. Fischer, A. Matsliah, and R. Yuster. Hardness and algorithms for rainbow connectivity. In *Proc. 26th Internat. Sympos. Theor. Aspects Comput. Sci.*, pages 243–254, 2009. arxiv:0902.1255v2.

[6] R. Christmas, I. Avila-Campillo, H. Bolouri, B. Schwikowski, M. Anderson, R. Kelley, N. Landys, C. Workman, T. Ideker, E. Cerami, R. Sheridan, G. Bader, and C. Sander. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Am Assoc Cancer Res Educ Book*, pages 12–16, 2005. http://www.cytoscape.org.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[8] D. Croes, F. Couche, S. J. Wodak, and J. van Helden. Metabolic pathfinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Research*, 33:W326–W330, 2005. DOI:10.1093/nar/gki437.

[9] N. Duarte, M. Herrgard, and B. Palsson. Reconstruction and validation of *Saccharomyces cerevisiae* ind750, a fully compartmentalized genome-scale metabolic model. *Genome Research*, 14:1298–1309, 2004. DOI: 10.1101/gr.2250904.

[10] K. Faust, D. Croes, and J. van Helden. Metabolic pathfinding using RPAIR annotation. *J. Mol. Biol.*, 388:390–414, 2009. DOI:10.1016/j.jmb.2009.03.006.

[11] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. *Siam J. Comput.*, pages 881–898, 2008.

[12] C. Francke, R. J. Siezen, and B. Teusink. Reconstructing the metabolic network of a bacterium from its genome. *Trends in Microbiology*, 13:550–558, 2005. DOI:10.1016/j.tim.2005.09.001.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, SCC-4(2):100–107, 1968.

[15] A. P. Heath, G. N. Bennett, and L. E. Kavraki. Finding metabolic pathways using atom tracking. *Bioinformatics*, 26:1548–1555, 2010. DOI:10.1093/bioinformatics/btq223.

[16] B. Junker, C. Klukas, and F. Schreiber. *VANTED*: a system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics*, 7, 2006. DOI:10.1186/1471-2105/7/109.

[17] C. Kaleta, L. F. de Figueiredo, and S. Schuster. Can the whole be less than the sum of its parts? Pathway analysis in genome-scale metabolic networks using elementary flux patterns. *Genome Research*, 19:1872–1883, 2009.

[18] T. Kamphans and M. Stelzer. SPUL: Shortest path with unique labels. C++ program, 2008. http://www.kamphans.de/spul.html.

[19] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38:D355–D360, 2010. DOI:10.1093/nar/gkp896.

[20] M. Lang. personal communication, 2010.

[21] H. Ma and A. P. Zeng. Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms. *Bioinformatics*, 19:270–277, 2003.

[22] D. McShan, S. Rao, and I. Shah. PathMiner: predicting metabolic pathways by heuristic search. *Bioinformatics*, 19:1692–1698, 2003. DOI: 10.1093/bioinformatics/btg217.

[23] G. Michal and D. Schomburg. *Biochemical Pathways: An Atlas of Biochemistry and Molecular Biology*. Wiley-Interscience, 2010.

[24] J. S. B. Mitchell. Shortest paths and networks. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 445–466. CRC Press LLC, Boca Raton, FL, 1997.

[25] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[26] S. A. Rahman, P. Advani, R. Schunk, R. Schrader, and D. Schomburg. Metabolic pathway analysis web service (Pathway Hunter tool at CUBIC). *Bioinformatics*, 21:1189–1193, 2005.

[27] S. A. Rahman, P. S. Jonnalagadda, J. Padiadpu, K. Hartmann, R. Schrader, and D. Schomburg. Metabolic network analysis: implication and application. *BMC Bioinformatics*, 6 (Suppl 3):12, 2005.

[28] S. A. Rahman and D. Schomburg. Observing local and global properties of metabolic pathways: *load points* and *choke points* in the metabolic networks. *Bioinformatics*, 22:1767–1774, 2006. DOI:10.1093/bioinformatics/btl181.

[29] M. Rosa da Silva. *Bioinformatics tools for the visualization and structural analysis of metabolic networks*. Phd thesis, Braunschweig University, 2006.

[30] M. Rosa da Silva, J. Sun, H. Ma, F. He, and A. P. Zeng. Metabolic networks. In B. H. Junker and F. Schreiber, editors, *Analysis of Biological Networks*, Wiley Series in Bioinformatics, pages 233–253. Wiley & Sons, 2008.

[31] M. Scheer, A. Grote, A. Chang, I. Schomburg, C. Munaretto, M. Rother, C. Söhngen, M. Stelzer, J. Thiele, and D. Schomburg. BRENDA, the enzyme information system in 2011. *Nucleic Acids Research*, to appear, 2011. DOI:10.1093/nar/gkq1089.

[32] M. Stelzer, J. Sun, A.-P. Zeng, S. P. Fekete, and T. Kamphans. An extended bioreaction database that significantly improves reconstruction and analysis of genome-scale metabolic networks. In preparation.

[33] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proc. IEEE Internat. Conf. Robot. Automat.*, pages 3310–3317, 1994.

[34] J. Sun, X. Lu, U. Rinas, and A. P. Zeng. Metabolic peculiarities of *Aspergillus niger* disclosed by comparative metabolic genomics. *Genome Biology*, 8, 2007. R182.