SIMPLE AND FAST ALIGNMENT OF METABOLIC PATHWAYS BY EXPLOITING LOCAL DIVERSITY

SEBASTIAN WERNICKE AND FLORIAN RASCHE

Institut für Informatik, Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
E-mail: {wernicke,m3rafl}@minet.uni-jena.de

An important tool for analyzing metabolic pathways is being able to do homology searches, that is, for a given pattern network one would like to find occurrences of similar (sub)networks within a set of host networks. In the context of metabolic pathways, Pinter et al. [Bioinformatics, 2005] recently proposed to solve this computationally hard problem by restricting it to the case where both the pattern and host network are trees. This restriction, however, severely limits the applicability of their algorithm. Here, we propose a novel algorithm that does not restrict the topology of the host or pattern network in any way; instead, we exploit a natural property of metabolic networks that we call "local diversity property," which allows us to obtain a very fast and simple algorithm for the alignment of metabolic pathways. Experiments on a testbed of metabolic pathways extracted from the BIOCYC database indicate that our algorithm is much faster than the restricted algorithm of Pinter et al. and yet has a wider range of applicability and yields new biological insights.

1. Introduction

Motivation. Shifting attention from linear data to more complex functions and interactions, recent years have seen a surge in the availability of biological *network* data.[a] An important tool for analyzing these data is being able to search for homologous (sub)networks to a given pattern network: This promises to be useful, for example, for interaction predictions, functional annotation, data integration, knowledge transfers, and for developing a better understanding of biological network organization.[5,10] A recent survey by Sharan and Ideker[10] even advances the opinion that "network comparison techniques promise to take a leading role in bioinformatics [...]."

Unfortunately, the task of performing a network homology search turns out to be quite hard as it can be traced back to the NP-complete SUBGRAPH ISOMORPHISM problem.

SUBGRAPH ISOMORPHISM
Input: Two graphs G_P (the *pattern*) and G_H (the *host*).
Task: Find whether G_H contains a subgraph that is isomorphic to G_P.

This problem is even NP-complete when restricted to graph classes that usually render NP-complete problems tractable—for instance, if the pattern is a forest and the host is a tree.[2] Polynomial-time algorithms are only known when the host graph has bounded treewidth ω

[a] We use the term "network" discussing biological aspects and the term "graph" for discussing algorithmic aspects.

and the pattern graph has either a high connectivity or bounded degree—in these cases, SUBGRAPH ISOMORPHISM can be solved in $O(n_P^{\omega+1} \cdot n_H)$ time for an n_P-vertex pattern and n_H-vertex host.[1,3,6]

To overcome the hardness of SUBGRAPH ISOMORPHISM when performing a network homology search on biological networks, various algorithms have been proposed:

- For protein interaction networks, Kelley et al.[5] presented an algorithm called PATHBLAST that, given a linear pathway as a query, randomly decomposes the host graph into linear pathways to find homologous pathways among these.
- For linear patterns, Shlomi et al.[11] proposed an algorithm that is based on random graph colorings.
- Pinter et al.[7] presented a homology search algorithm for metabolic pathways that is based on restricting the host and pattern graph to be trees, in which case polynomial-time algorithms are possible[8].

Notably, all of these algorithms basically take the same approach: They make network homology searches algorithmically feasible by restricting the topology of the network to be cycle-free. This, in turn, causes all of these algorithms to suffer from basically the same problems, namely at least one of the following three:

(1) *Limited Applicability.* Most biological networks of interest contain cycles and the proposed algorithms therefore cannot be directly applied to them.
(2) *Long Running Time.* To apply the existing algorithms to a network that contains cycles, the network must be decomposed in some way. Irrespective of whether these decompositions are randomized or deterministic, a great number of them is necessary in order to ensure that all good matches for the pattern are found. This usually leads to exponential running times that are only practical for very small pattern sizes (for example, PATHBLAST requires $O(\ell!)$ runs for a pattern of length ℓ).
(3) *Requirement of Manual Labor and Expert Knowledge.* One might choose to use expert knowledge and manually decompose networks into cycle-free subgraphs. Such an approach was chosen, for example, by Pinter et al.[7] to obtain some of the dataset for their pathway alignment tool (they also excluded some of the networks that have cycles).[9] It is clear that such a process is not always applicable (for example, when little information is known about a network beforehand or we have no idea what the result should roughly look like), tedious, and error prone.

Intriguingly, there is another thing that the existing network homology search algorithms have in common besides basically taking the same approach: They do not algorithmically expose the fact that vertices in biological networks are labeled; rather, the vertex labels are used only for scoring the similarity of the pattern graph to a given subgraph in the host. In the context of metabolic pathway alignment, this work proposes an approach that *does* expose vertex labels in order to obtain a network alignment algorithm that is simple, fast, and imposes no restrictions concerning the topology of the input networks.

Organization of this Work. After introducing some notation, Section 2 presents our new alignment algorithm for metabolic pathways in three steps: First, Subsection 2.1 formalizes our network alignment problem and presents a simple—yet impractical—algorithm for it called MATCH. Second, Subsection 2.2 introduces the local diversity property of metabolic networks which, third, is exploited in Subsection 2.3 by slightly modifying the MATCH algorithm so as to obtain our new metabolic pathway alignment algorithm FIT-MATCH. The FIT-MATCH algorithm has been implemented in C++; the source code is freely available at `http://theinf1.informatik.uni-jena.de/graphalignments/`. Section 3 reports experiments with our implementation on a testbed of metabolic pathways from the BIOCYC database[4]. These indicate that our algorithm is much faster than the algorithm of Pinter et al.[7] and yet—because the topology of the input networks is not restricted—is simpler to use, yields new insights, and has a wider range of applicability.

2. A New Fast and Simple Pathway Alignment Algorithm

Before formalizing the problem of metabolic pathway alignment and discussing our new algorithm FIT-MATCH to solve this problem, it is useful to establish some notation:

Notation. We model metabolic pathways as connected directed graphs. Each vertex represents an enzyme and is labeled with the *Enzyme Commission number (EC number)* of that enzyme.[b] Two vertices u and v are connected by a directed edge (u, v) if a product of the pathway reaction catalyzed by u is a substrate of the reaction catalyzed by v. The *pattern* graph for which we seek a homolog is always denoted $G_P = (V_P, E_P)$, the *host* graph in which we seek an occurrence of the pattern graph is denoted $G_H = (V_H, E_H)$. A vertex with exactly one outgoing and one incoming edge (not counting self loops) is called a *path vertex*; all other vertices are called *branch vertices*.[c] A path in a graph that consists just of path vertices and where every vertex occurs at most once is called *simple*.

An *isomorphism* between two graphs $G = (V, E)$ and $G' = (V', E')$ is a one-to-one mapping $\Phi : V \to V'$ such that $(u, v) \in E \Leftrightarrow (\Phi(u), \Phi(v)) \in E'$ (note that this definition ignores the labels of the vertices). If there exists an isomorphism between two graphs, we call them *isomorphic*. Two graphs are called *homeomorphic* if we can subdivide their edges (that is, edges can be replaced by simple paths of arbitrary length in the same direction) in such a way that the resulting graphs are isomorphic. The corresponding *homeomorphism* is a function φ that bijectively maps the branch vertices of the two graphs onto each other.

2.1. *Formalization and a Simple Backtracking Algorithm*

In order to formalize the problem of metabolic pathway alignment, we must define two things, namely what we mean by "alignment" and what we view as a "high-scoring" align-

[b] The EC number is a four-level hierarchical scheme that classifies enzymes on a functional basis. Thus, each enzyme is classified by four numbers (as in "3.4.23.48"), the first number representing the top level classification and the three following numbers the subsequent refinements thereof.

[c] Although somewhat counterintuitive, to simplify the overall presentation we chose to use the term "branch vertex" also for vertices with degree one.

ment. Concerning a formalization of alignments, we follow Pinter et al.[7] and rely on a notion that is based on subgraph homeomorphism:

Definition 2.1. An *embedding* of a pattern graph G_P into a host graph G_H is a tuple (G'_H, φ) where G'_H is a subgraph of G_H that is homeomorphic to G_P and φ is a homeomorphism between G'_H and G_P.

We can use the notion of an embedding to phrase metabolic pathway alignment as a combinatorial problem called MAXIMUM-SCORE EMBEDDING.

MAXIMUM-SCORE EMBEDDING
Input: Two directed labeled graphs $G_P = (V_P, E_P)$ and $G_H = (V_H, E_H)$.
Task: Find the maximum-score embedding of G_P into G_H.

It remains to define the scoring scheme that we plug into this problem definition. Again, we follow Pinter et al.[7] and make use of a scoring scheme due to Tohsato et al.[12] that is based on mutual vertex–vertex similarities (observe that topological similarity is already ensured by relying on homeomorphisms). The similarity of two enzymes is calculated from their functional EC numbers—the more of this number two enzymes have in common, the more similar they are considered to be.[d] The scoring scheme also incorporates an information-theoretic consideration, namely that the similarity of two enzymes is more significant the less their common EC number prefix occurs among all enzymes.

Definition 2.2. Let the vertices u and v represent two enzymes e_1 and e_2, respectively. If the lowest common enzyme class of e_1 and e_2 as determined by their EC numbers contains h enzymes, then the *similarity* of u and v is defined as $\mathit{sim}(u, v) := -\log_2 h$.

Using the scoring for pairwise similarity, we can define a similarity score for two simple paths that is based on the notion of a sequence alignment.

Definition 2.3. Given two simple paths $p_1 = u_1 \dots u_x$, $p_2 = v_1 \dots v_y$ and a negative *gap penalty* g, their similarity $\mathit{sim}(p_1, p_2, g)$ is defined as the maximum possible score of a sequence alignment between p_1 and p_2 using g as the gap penalty and the scoring scheme of Definition 2.2 to evaluate pairwise similarities.

For the sake of simplicity in our presentation, let us assume from now on that there is at most one simple path between any two branch vertices and that neither the pattern nor the host is a simple cycle. Our implementation in Section 3 does not make any of these restrictions, but handling them explicitly in the remainder of this section obfuscates the main ideas. To render the scoring of an embedding precise, we use the following definition:

Definition 2.4. Given an embedding (G'_H, φ) of a pattern graph G_P in a host graph G_H, let $B(G_P)$ denote the branch vertices of G_P. For two branch vertices u and v let $p(u, v)$

[d] For some applications, a purely functional classification might be suspect and one might want to additionally include genetic similarity information for the enzymes; we do not consider this here, however.

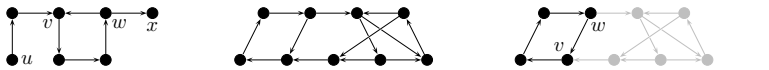
be the simple path between u to v; if no such path exists, then $p(u, v)$ is the empty graph. Given a gap penalty $g < 0$, the *score* of (G'_H, φ) is defined as

$$\mathit{score}(G'_H, \varphi) := \sum_{v \in B(G_P)} \mathit{sim}(v, \varphi(v)) + \sum_{u,v \in B(G_P)} \mathit{sim}\big(\, p(u,v),\, p(\varphi(u),\, \varphi(v)\,)\,,\, g\,\big)\,.$$

Naïvely, MAXIMUM-SCORE EMBEDDING can be solved by a simple backtracking algorithm that exhaustively explores all possible embeddings of a given pattern graph G_P into a host graph G_H. Formally, this algorithm is best described by using the notions of a *partial embedding* and *extensions* thereof.

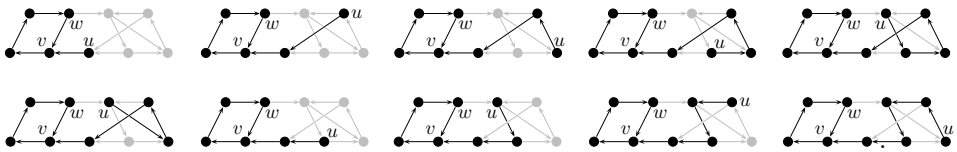
Definition 2.5. A *partial embedding* of a pattern graph G_P into a host graph G_H is an embedding of a connected subgraph G'_P of G_P into G_H. It is denoted by (G'_P, G'_H, φ) (where φ is the homeomorphism between G'_P and G'_H). Let p be a simple path in G_P that connects two branch vertices u and v such that at least one of these branch vertices is in G'_P but no path vertex of p. An *extension* of a partial embedding (G'_P, G'_H, φ) by p is a partial embedding of the subgraph induced in G_P by G'_P, u, v, and p that is identical to (G'_P, G'_H, φ) when restricted to the vertices of G'_P.

To illustrate the concept of a partial embedding and its extensions, consider the following example graphs G_P and G_H and a partial embedding of G_P into G_H:



G_P G_H Partial embedding of G_P into G_H

The shown partial embedding has ten possible extensions by the path from u to v:



We can now describe our naïve backtracking algorithm for solving MAXIMUM-SCORE EMBEDDING. This algorithm, which we call MATCH, starts out by aligning a branch vertex of the pattern to a branch vertex in the host graph and then uses a recursive subprocedure EXTEND that takes as input a partial embedding and tries all possible extensions for it, thus enumerating all embeddings of the pattern graph into the host graph.

Algorithm: MATCH(G_P,G_H,g)
Input: Two labeled graphs $G_P = (V_G, E_G), G_H = (V_H, E_H)$ and a gap penalty g.
Output: A maximum-score embedding of G_P into G_H, if one exists.
Global variables: Graphs G_P and G_H, score *maxscore*, and embedding *best*.

```
01  best ← null ; maxscore ← −∞
02  u ← arbitrary vertex from V_G
03  for each v ∈ V_H do
04      (G'_P, G'_H, φ) ← partial embedding by mapping u to v
05      call EXTEND(G'_P, G'_H, φ)
06  return best
```

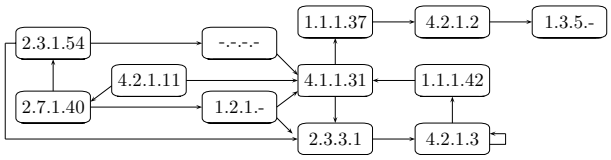


Figure 1. Anaerobic respiration pathway of *Escherichia coli* that illustrates the local diversity property. The label "-.-.-.-" denotes an unclassified enzyme.

```
EXTEND(G'_P, G'_H, φ)
E1  if G'_P ≠ G_P then
E2      p ← simple path in G_P not contained in G'_P such that at
              least one of the connected branch vertices is in G'
E3      for each extension (G''_P, G''_H, φ') of (G'_P, G'_H, φ) by p do
E4          call EXTEND(G''_P, G''_H, φ')
E5  else if score(G'_P, G'_H, φ) > maxscore then
E6      best ← (G'_P, G'_H, φ); maxscore ← score(G'_P, G'_H, φ)
E7  return
```

Analysis of MATCH. The running time of MATCH is primarily determined by the number of recursive calls that are made in lines *05* and *E4* of the algorithm. While this number is upper-bounded by a constant—both the maximum path length and the maximum degree of a metabolic pathway are naturally bounded by some constant for biological reasons—it turns out to be rather large.[e] In our experiments, we have found that if the pattern graph consists of k simple paths, then the size of the search tree that is explored by MATCH is, on average, around 6^k. Considering that our dataset from the BIOCYC database contained a considerable amount of pathways with more than ten paths, this leads to a very long running time for MATCH.

2.2. *The Concept of Local Diversity*

As a typical example for a metabolic pathway, consider the anaerobic respiration pathway of *Escherichia coli* that is shown in Figure 1. The following observation can be made here which seems to hold for most metabolic pathways and is hence crucial to our approach:

Observation 2.1. Two paths that have the same starting vertex often carry out very different biological functions.

This observation describes what we refer to as the *local diversity property* of metabolic networks. There are plausible reasons why a metabolic network is expected to generally have this property: First, most metabolic products offer only very few possibilities where a certain reaction can chemically take place. Second, identical reactions for a certain substrate within a pathway are usually carried out by only one enzyme for reasons of efficiency.

[e]Note that *all* paths and not only simple paths in the host graph must be considered for an extension because a branch vertex in the host may become a path vertex in its subgraph.

Local diversity is an important property for the algorithmic alignment of metabolic pathways: Intuitively, SUBGRAPH ISOMORPHISM is hard because even very different graphs might appear similar based on local information. The local diversity property, however, means that metabolic pathways usually provide very rich and *diverse* local information that can be exploited to overcome this phenomenon.

2.3. *Exploiting Local Diversity*

When we compute all extensions of a partial embedding by a path p, some of these might not make sense from a biological perspective because the biological function of the pattern path p does not *fit* the biological function of the host path that it is aligned to. The key to making MATCH more efficient is to observe that the local diversity property implies that usually *a lot of* extensions of a partial embedding do not make sense from a biological perspective. Thus, to exploit local diversity and make MATCH more efficient, we need to devise a formal definition of "fitting biological function" for two given paths and then modify MATCH such that it only explores *fitting* embeddings.

Definition 2.6. Given a real number $0 \leq f \leq 1$, a gap score g, a simple x-vertex path p_1 and a simple y-vertex path p_2, we say that p_1 and p_2 *fit* if a maximum-score alignment between them aligns at most $\min\{\lceil(1-f)\cdot x\rceil, \lceil(1-f)\cdot y\rceil\}$ vertices to a gap. An extension of a partial embedding (G'_P, G'_H, φ) *fits* if every simple path between two branch vertices $u, v \in V'_G$ fits the corresponding simple path between $\varphi(u), \varphi(v) \in V'_H$.

As an illustration, if we have a fitting parameter of $f = 0.50$, then a four-vertex path fits no path that consists of seven or more vertices; a higher fitting parameter of $f = 0.75$ would cause it to fit no path that consists of six or more vertices.[f] To exploit local diversity, we now modify MATCH so that it only explores fitting embeddings. For this purpose, lines *05* and *E4* need to be modified so that the EXTEND-subprocedure is only called for fitting extensions. We name the resulting algorithm of this modification FIT-MATCH.

Analysis of FIT-MATCH Experiments show that, indeed, exploring only fitting extensions is a very effective pruning strategy due to the local diversity property of metabolic networks. More precisely, they show that whereas MATCH explored a search tree of size around 6^k to align a k-path pattern, even a conservative fitting parameter of $x = 0.5$ reduces this to around 2.5^k, "conservative" meaning that we found no meaningful alignment in our experiments that is missed by this setting.

3. Experiments on Metabolic Networks

We implemented FIT-MATCH in C++ to test its practical performance; the source is available at `http://theinf1.informatik.uni-jena.de/graphalignments/`.

[f] For some applications, Definition 2.6 might be considered too strict in its handling of very short paths: In particular, a one-vertex path never fits a length-3 path, regardless of the fitting parameter. While we have not found this property to be an issue in practice, one can easily circumvent it by introducing a minimum number of gaps that is always allowed regardless of the path lengths or fitting parameter.

Table 1. Runtimes of our FIT-MATCH implementation for all-against all alignments between the five datasets described in the text. For each combination of host and pattern, we show the total runtime including I/O overhead and excluding I/O overhead. All values are given in seconds.

Pattern	Run time of FIT-MATCH in seconds (including / excluding I/O overhead)				
	B. subtilis	*E. coli*	*H. sapiens*	*S. cerevisiae*	*T. thermophilus*
B. subtilis	82 / 0.41	120 / 2.25	102 / 2.25	95 / 0.29	147 / 2.28
E. coli	120 / 0.02	121 / 0.22	112 / 0.19	151 / 0.02	227 / 0.20
H. sapiens	107 / 0.02	120 / 0.19	89 / 0.20	130 / 0.02	190 / 0.29
S. cerevisiae	93 / 0.06	141 / 0.09	121 / 0.09	114 / 0.08	172 / 0.10
T. thermophilus	140 / 0.02	135 / 0.22	107 / 0.23	167 / 0.03	264 / 0.24

Method and Results. Our testing machine is an AMD Athlon 64 3400+ with 2.4 GHz, 512 KB cache, and 1 GB main memory running under Debian GNU/Linux 3.1. Sources were compiled with the GNU g++ 4.2 compiler using the option "-O3".

To evaluate the performance of FIT-MATCH, metabolic pathways were extracted from the BioCyc database[4] for five different organisms, yielding 145 pathways of *B. subtilis*, 220 pathways of *E. coli*, 190 pathways of *H. sapiens*, 176 pathways of *S. cerevisiae*, and 267 pathways of *T. thermophilus*. If the full EC number of an enzyme was not specified, the unknown part of the code was treated as "don't care", meaning that the enzyme is scored as if it were identical to every enzyme for which the known part of the codes match. All 25 possible all-against-all inter- and intra-species alignments between the five datasets where performed, resulting in a total of 996 004 homology searches.

Following the suggestion of Pinter et al.[7] to set the gap score to about one third of the worst vertex–vertex similarity score, we set $g = -4.5$. The fitting parameter x was set to $.50$ as a conservative choice, meaning that we never encountered an interesting alignment that is only found with a lower fitting parameter in some preliminary experiments. The obtained runtimes are shown in Table 1; some sample alignments are shown in Figure 2.

Discussion. The experiments show that our FIT-MATCH implementation is capable of quickly aligning metabolic pathways; the complete dataset can be aligned in under an hour on our testing machine (including the I/O overhead, which turned out to consume far more time than the algorithm itself). This is much faster compared to the pathway alignment tool of Pinter et al.[7]: Their implementation (called MetaPathwayHunter) requires some hours alone to align the simplified trees of the *E. coli* and *S. cerevisiae* pathways whereas FIT-MATCH can align the corresponding unsimplified data in roughly seven minutes.

The alignments shown in Figure 2 exemplify some interesting application scenarios where FIT-MATCH can efficiently be used:

- *Pathway Comparison.* Figure 2a shows the highlighting of alternative metabolic pathways by comparing the classical TCA cycle with a more complex variant (note how the complex variant uses more pathways and the succinate dehydrogenases 1.3.99.1 instead of 1.3.5.1).
- *Enzyme Classification.* In Figure 2b, our results align all unclassified enzymes

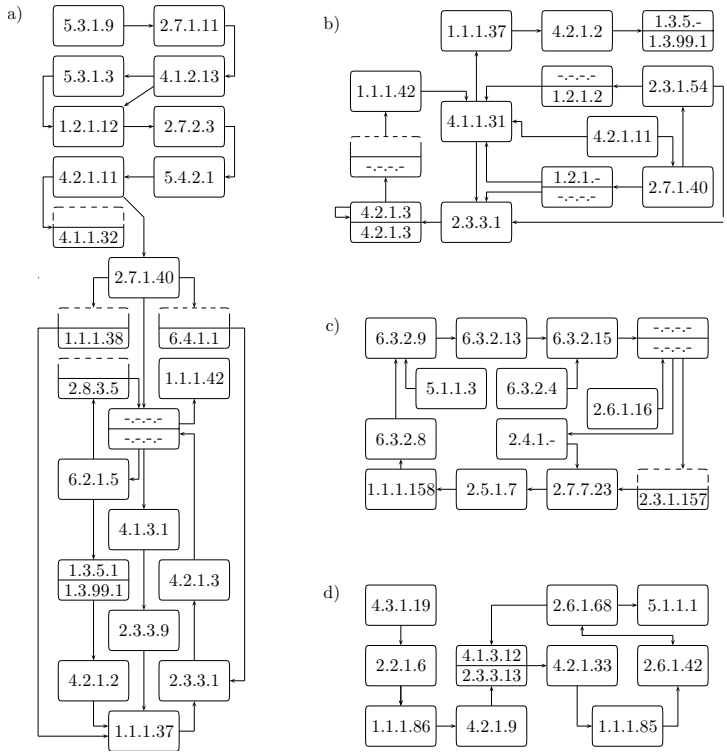


Figure 2. Four examples for the alignments that were found by the FIT-MATCH algorithm. In all graphs, the vertices are not split if they have the same label in the host and the pattern, otherwise, the pattern enzyme is shown at the top and the host enzyme at the bottom. A dashed top half indicates that a vertex is only present in the host graph. The four alignments (pattern/host) that are shown are **a)** superpathway of of glycolysis, pyruvate dehydrogenase, TCA, and glyoxylate bypass versus Embden-Meyerhof pathway in *B. subtilis* **b)** anaerobic respiration pathway of *E. coli* versus the same pathway in *B. subtilis* **c)** peptidoglycan and lipid A precursor biosynthesis in *B. subtilis* versus the same pathway in *T. thermophilus* **d)** superpathway of leucine, valine, and isoleucine biosynthesis in *E. coli* versus the same pathway in *T. thermophilus*.

(denoted "-.-.-.-") with already known enzymes, possibly hinting at their function.

- *Identifying Enzyme Complexes* The pathways shown in Figure 2c are almost identical, except that *B. subtilis* does not possess the enzyme 2.3.1.157 (an acyltransferase) but is rather aligned to a gap. The preceding enzyme is unclassified in both organisms. We can derive from the alignment that the unclassified enzyme in *B. subtilis* fulfills a task that requires two enzymes in T. thermophilus.
- *Data Integration.* Figure 2d shows an example where we can use FIT-MATCH to detect the consistency of a database: The two enzyme classification numbers that are seemingly totally different are the result of a change in nomenclature.

The results we found moreover demonstrate that the topological restrictions imposed by the algorithm of Pinter et al.[7] cause relevant alignments to be missed in several cases. For example, if the methylglyoxal pathway and the chorismate superpathway of *E. Coli*

are aligned, MetaPathwayHunter does not produce any results whereas FIT-MATCH finds an alignment. Or, as a second example, MetaPathwayHunter misses the possible alignment between the cobalamin biosynthesis and the KDO_2 lipid biosynthesis superpathway of *E. coli* (which FIT-MATCH found).

4. Conclusion

We have presented the concept of local diversity for metabolic networks and shown how this property can be exploited to obtain a simple alignment algorithm FIT-MATCH for metabolic pathways that is both faster and more generally applicable than previous approaches. We are currently turning the FIT-MATCH implementation into a graphical tool for the discovery and analysis of metabolic pathway alignments.

All biological networks carry labels at their vertices. We think that the concept of local diversity is likely to occur in other types of biological networks than metabolic networks and could thus be exposed for alignment algorithms there, too. Given the nice properties of FIT-MATCH, this certainly seems worthwhile to investigate.

Acknowledgments. This work was supported by the Deutsche Telekom Stiftung (Sebastian Wernicke) and the Deutsche Forschungsgemeinschaft (DFG) project PEAL (parameterized complexity and exact algorithms, NI 369/1) (Florian Rasche). The authors are grateful to Rolf Niedermeier (Jena) for discussions and comments.

References

1. A. Dessmark, A. Lingas, and A. Proskurowski. Faster algorithms for subgraph isomorphism of k-connected partial k-trees. *Algorithmica*, 27(3):337–347, 2000.
2. D. S. Garey, M. R.and Johnson. *Computers and Intractability*. Freeman, 1979.
3. M. Hajiaghayi and N. Nishimura. Subgraph isomorphism, $\log$-bounded fragmentation and graphs of (locally) bounded treewidth. In *Proceedings of 27th MFCS*, volume 2420 of *LNCS*, pages 305–318. Springer, 2002.
4. P. D. Karp, C. A. Ouzounis, C. Moore-Kochlacs, et al. Expansion of the BioCyc collection of pathway/genome databases to 160 genomes. *Nucleic Acids Research*, 19:6083–6089, 2005.
5. B. P. Kelley, B. Yuan, F. Lewitter, et al. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, 32(web server issue):83–88, 2004.
6. J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.
7. R. Y. Pinter, O. Rokhlenko, E. Y. Lotem, and M. Ziv-Ukelson. Alignment of metabolic pathways. *Bioinformatics*, 21(16):3401–3408, 2005.
8. R. Y. Pinter, O. Rokhlenko, D. Tsur, and M. Ziv-Ukelson. Approximate labelled subtree homeomorphism. In *Proceedings of 15th CPM*, volume 3109 of *LNCS*, pages 59–73. Springer, 2004.
9. O. Rokhlenko. Personal communication, 2006.
10. R. Sharan and T. Ideker. Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 24(4):427–433, 2006.
11. T. Shlomi, D. Segal, E. Ruppin, and R. Sharan. QPath: a method for querying pathways in a protein–protein interaction network. *BMC Bioinformatics*, 7:199, 2006.
12. Y. Tohsato, H. Matsuda, and A. Hashimoto. A multiple alignment algorithm for metabolic pathway analysis using enzyme hierarchy. In *Proceedings of 8th ISMB*, pages 376–383, 2000.