A New Tool for the Alignment of Metabolic Pathways

Ron Y. Pinter
Oleg Rokhlenko
Esti Yeger-Lotem
Michal Ziv-Ukelson

Dept. of Computer Science
Technion – Israel Institute of Technology

November 11, 2015
Metabolic Pathways

- Metabolic pathways – detailed chemical reactions and catalyzing enzymes
  - a product of one reaction is a substrate of another
- Metabolic network of an organism – a set of metabolic pathways
- Insight into
  - cellular responses to environmental changes
  - evolutionary processes
Metabolic Networks
Data and Analysis

• Current efforts to reconstruct genome-scale metabolic networks
  – *E. coli*, *S. cerevisiae*, human
  – *in vivo* and *in silico* [Palsson, UCSD]

• Metabolic pathways databases
  – EcoCyc, KEGG, SGD
  – tools for pathway visualization, queries on pathway components

• Dearth of tools for comparison between pathways
Goal

• Develop a pathway alignment tool for the detection of
  – a novel pathway against a database of pathways
  – a pathway fragment in a network
  – pathway conservation between species
  – pathway evolution within a species

• Pathway similarity (not necessarily identity!) should rely on
  – pathway structure
  – enzyme similarity

• A good abstraction is needed…
Representation

Metabolic pathway => labeled (line) graph:

- nodes = enzymes
- node labels = Enzyme Commission (EC) classification numbers (functionality)
- edges = between two enzymes if one’s product is the other’s substrate

- Uridine phosphorylase: EC: 2.4.2.3
- Cytidine deaminase: EC: 3.5.4.5
Computational Problem

• Search for highly similar substructures in larger ones
• Gives rise to numerous pattern matching problems on labelled graphs
  – Recall that subgraph isomorphism is NP-hard [Garey and Johnson, 1979]
• Our approach: approximate labeled subtree homeomorphism (ALSH)
Subtree Homeomorphism

Pattern

Text

2 deletions

1 deletion
Best Homeomorphism

Pattern

Text

2 deletions

1 deletion
Best Homeomorphism

Pattern

Text

2 deletions

1 deletion
Best Homeomorphism

Pattern

Text

2 deletions

1 deletion
Approximate Labelled Subtree Homeomorphism (ALSH)

\[ LSH(M) = D \times (|T_2| - |T_1|) + \sum_{i,j} \Delta[i, j] \]
Subtree Homeomorphism Complexity

Rooted trees: \( m^{1.5}n/\log n \) [Shamir and Tsur, 1999]
Unrooted trees: \( m^{1.5}n/\log n \) [Shamir and Tsur, 1999]
## Our Results [CPM’04]

<table>
<thead>
<tr>
<th>$\textbf{ALSH}$</th>
<th>Rooted</th>
<th>Unrooted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unordered Trees</td>
<td>$O(nm^2 + nm \log n)$</td>
<td>$O(nm^2 + nm \log n)$</td>
</tr>
<tr>
<td>Ordered Trees</td>
<td>$O(nm)$</td>
<td>$O(nm)$</td>
</tr>
</tbody>
</table>

$m$ - # of vertices in $P$

$n$ - # of vertices in $T$
Related Work (Algorithmic)

Related Work (Biological)

ALSH on Rooted Unordered Trees

\[ P \]
\[ T \]

\[ x_1 \quad x_2 \quad \ldots \quad u \quad \ldots \]
\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
\[ \ldots \]
\[ v \]

\[ w_{11} \quad w_{12} \quad w_{1u} \]
\[ w_{21} \quad w_{22} \quad w_{2u} \]
\[ w_{31} \quad w_{32} \quad w_{3u} \]
ALSH on Rooted Unordered Trees

\[
P \quad \begin{array}{cccccc}
 & u & & v & & \\
 x_1 & x_2 & y_1 & y_2 & y_3 & \\
 y_1 & w_{11} & w_{12} & w_{1u} & & \\
 y_2 & w_{21} & w_{22} & w_{2u} & & \\
 y_3 & w_{31} & w_{32} & w_{3u} & & \\
 & \cdots & & & & \\
 v & & & & & \\
 & \cdots & & & & \\
\end{array}
\]
ALSH on Rooted Unordered Trees

Weighted Assignment on \( G \)

\[
RScore(v, u) = \max \left\{ \Delta[v, u] + AssignmentScore(G), \text{BestChild}(v, u) + D \right\}
\]
The algorithm computes an assignment for each pair $(u_i \in P, v_j \in T)$, where $i = 1 \ldots m$ and $j = 1 \ldots n$.

The assignment for $u_i$ and $v_j$ is computed using the bipartite graph $G(V = X \cup Y, E)$, where $|X| = k_i \leq |Y| = l_j$.

[Fredman & Tarjan 87] show how to compute $\text{AssignmentScore}(G)$ in $O(X(E + V \log V)) = O(k_i^2 l_j + k_i l_j \log l_j)$. 
Time Complexity of ALSH Rooted Unordered Trees

- Observation 1: \[ \sum_{i=1}^{m} k_i = m - 1 \] and \[ \sum_{j=1}^{n} l_j = n - 1 \]

Summing up all \((u_i, v_j)\) node pairs:

\[
O\left( \sum_{i=1}^{m} \sum_{j=1}^{n} k_i^2 l_j + k_i l_j \log l_j \right) = O\left( \sum_{i=1}^{m} k_i^2 n + k_i n \log n \right) = O(m^2 n + mn \log n)
\]

- Under the similarity assumption weighted assignment can be solved in \(O(V^{0.5}E \log(VC))\) [Gabow and Tarjan, 89] and ALSH can be solved in \(O(m^{1.5}n \log(nC))\).
A Word on Unrooted Trees

• Less important for this application
• Can still be done with same complexity
  – Requires decremental analysis of weighted matching
• Different technique in the ordered case
  – Involves cyclic string comparison
More on Complexity

• For unordered trees (rooted and unrooted): can be reduced to

\[ O\left(\frac{nm^2}{\log m} + nm \log n\right) \]

using \textit{compressed graphs}

• Rooted, ordered trees
  – No cross-edges in the bipartite graph
  – Gives rise to dynamic programming like algorithm

• Algorithms and results: \textit{Pinter et al., Combinatorial Pattern Matching (CPM), 2004}
Why is ALSH Suitable?

- The replacement can take place if
  - the replacing enzyme is multifunctional
    - *can thus catalyze several consecutive reactions*, or
  - the enzyme uses an alternative catalysis
    - *leads directly from the initial substrate to the final product*

- Translating this into graph terms degree-2 nodes may be deleted from the graph: perfectly captured by subtree homeomorphism
Beyond Trees (1)

Everything extends to multi-source trees:
Beyond Trees (2)

• Breaking cycles
  – generate alternatives
  – surprisingly rare

• Handling DAGs
  – duplicate and split merge nodes
  – fits well with biology: *alternative pathways*
Similarity Measures

- Sequence similarity (Alignment)
- Reaction similarity (EC hierarchy)
- Semantic similarity (Gene Ontology)
- Cluster based similarity (ProtoNet)
Similarity Measures

- Sequence similarity (Alignment)
- Reaction similarity (EC hierarchy)
- Semantic similarity (Gene Ontology)
- Cluster based similarity (ProtoNet)
Enzymes Similarity Scoring

- Functional similarity based on EC classification
  - Hierarchical scheme developed by the IUBMB
  - Consists of 4 levels + a root level
  - Each enzyme has an EC number, e.g. [2.7.3.112]

- Similarity score of $e_1, e_2$ ~ the size of the subtree under the lowest common ancestor of $e_1, e_2$
**Similarity Measures**

\[ I(h) = -\ln E(h) \]

where \( E(h) \) is the number of enzymes under the class \( h \).

\[ \text{sim}(e_1, e_2) = I(lca(e_1, e_2)) \]
The *MetaPathwayHunter* Tool
[Bioinformatics 2005]

- Searches a pattern $P$ against a database of text trees ($T$s)
- Each alignment is given a score $s$ and statistical significance $p$
- Statistical significance computation:
  - Executes $P$ against a database of 1000 randomized $T$s
  - $p(P)$: the fraction of randomized $T$s whose alignment with $P$ scored $\geq s$
- Enables visualization of the alignment, which is color-coded by similarity
Experimental Evaluation: Data

- **Biocyc**
  - *E. coli*: 113 pathways
- **SGD**
  - Yeast: 151 pathways
- Pathways based on genome annotations + manual curation
- <10 cycles per organism
Executions

• Aligned all possible pairs of analogous metabolic pathways in *E. coli* and *S. cerevisiae* (budding yeast)

• All-against-all intra-species alignment

• Labels: EC numbers; *matches* are color coded
Inter-Species Alignments (1)

- Metabolic pathways are conserved: the alignments between 62 out of 80 pairs of analogous pathways were significant.
- Conservation is not limited to small pathways, e.g.

Phenylalanine, tyrosine, and tryptophan biosynthesis, *E. coli* vs. yeast: $s = -4.28; p < 0.01$

- TyrA (*E. coli*) and Tyr1 (yeast) –
  - differ in the acceptor (NAD+ vs. NADP+)
  - functional homologs - no sequence similarity
Inter-Species Alignments (2)

Alignments with gaps:
- Met17 in yeast does the work of MetB and MetC in *E. coli*
- All Met* genes bear sequence homology

Potential gene fusion (*E. coli* to yeast)

Homoserine & methionine biosynthesis, *E. coli* vs. yeast: 
$s = -13.15$; $p < 0.01$
Intra-Species Alignments

Hints to evolution of paralogous pathways

Amino acid biosynthesis pathways are conserved

- Valine, Leucine, Isoleucine (hydrophobic aa’s) have similar biosynthesis pathways
  - Valine and isoleucine employ the same pathway but on different substrates
    \[ s = 0, \ p < 0.01 \]
  - Leucine employs a homologous pathway
    \[ s = -19.75, \ p < 0.03 \]
MetaPathway Queries

Flexible searching – possible queries:

• Pathway endpoints (source and product)
• Common pathway core

Metabolic pathway tinkering

\[ E. \text{ coli alantoine degradation: } s=0; \ p < 0.01 \]

\[ \text{Yeast ureide degradation: } s=0; \ p<0.01 \]
Status and Availability

- Engine in C++, GUI in Java
- Available upon request (download)
  - whole or client only
- Requires preprocessing (to trees)
Future Work

• Modeling
  – affine deletion scores
  – hypergraphs

• Algorithms
  – extensions to e.g. bounded tree-width graphs
  – multiple alignment

• Applications
  – compare different similarity metrics
  – apply to other data types (e.g. phylogenies)
  – other domains (semantic queries, NLP, etc.)
Thanks

- Elad Ben-Yosef
- Moshe Itzhaki
- Mattan Winaver
- Seffi Naor

- Dekel Tsur
- Hanah Margalit
- Roded Sharan
- Michael Waterman
EC hierarchy

1st level:
- six broad classes of enzymatic activity:
  1. Oxidoreductase
  2. Transferase
  3. Hydrolase
  4. Lyase
  5. Isomerase
  6. Ligase
Similarity Measures

EC hierarchy

2nd level:

- Identifies the subclass.
- For oxidoreductases the second level specifies the kind of donor which is oxidized.
- For example, sub-class [1.1] means that the enzyme acts on the CH-OH group of donors.
Similarity Measures

EC hierarchy

3rd level:
- Specifies the kind of acceptor.
- For example, the sub-sub-class [1.1.1] means that NAD+ or NADP+ are the acceptors.
Similarity Measures

EC hierarchy

4th level (leaves):

- Identifies the particular reaction.
- For example, the donor of [1.1.1.1] and [1.1.1.2] is CH-OH and their accepters are NAD+ and NADP+, respectively.
Unrooted Unordered Trees (naïve approach)

Pattern (P)  Text (T)

Weighted Assignment 2x4 …
Unrooted Unordered Trees (naïve approach)

Pattern (P)  Text (T)

Weighted Assignment 2x4 ...
Unrooted Unordered Trees

Pattern (P)

Text (T)

Weighted Assignment $2 \times 4$ ...
Unrooted Unordered Algorithm

Pattern (P)

Text (T)

Weighted Assignment 3x4 ...
Unrooted Unordered Algorithm
Unrooted Unordered Algorithm

Pattern (P)

Text (T)

One augmentation!
Unrooted Unordered Algorithm

Pattern (P)

Text (T)

$u \quad x \quad q$

$y_1 \quad y_2 \quad y_3 \quad y_4$

X

Y
Unrooted Unordered Algorithm

Pattern (P)

Text (T)

One augmentation!
Unrooted Unordered Algorithm

Pattern (P)

Text (T)

X

Y
Unrooted Unordered Algorithm

Pattern (P)

Text (T)

\[ UScore(v, u, x_q) = \max \begin{cases} 
\Delta[v, u] + AssignmentScore(X \setminus \{x_q\}, Y) \\
BestChild(v, u) + D 
\end{cases} \]
Decremental Property of Weighted Assignment

Lemma:

Let

- \( G(X \cup Y, E) \) be a bipartite graph
- \( k_i = |X| \leq l_j = |Y| \)
- \( G_q = (X_q \cup Y, E) \) for \( X_q = X \setminus \{x_q\} \)

Computing the weighted assignments for the series of bipartite graphs \( G_q \), for \( q = 1 \ldots k_i \) can be done in time \( O(k_i^2 l_j + k_i l_j \log l_j) \)
Time Complexity of ALSH
Unrooted Unordered Trees

Correction path
Dijkstra’s algorithm
\[ O(E + V \log V) \]

For all neighbors of \( u \):
\[ O(k_i^2 l_j + k_i l_j \log l_j) \]

→ ALSH on Unrooted Unordered trees can be computed in
\[ O(m^2 n + mn \log n) \]
Decremental Property of Weighted Assignment (proof)
Decremental Property of Weighted Assignment (proof)
Decremental Property of Weighted Assignment (proof)
Decremental Property of Weighted Assignment (proof)
Decremental Property of Weighted Assignment (proof)
Decremental Property of Weighted Assignment (proof)

Theorem [Ford & Fulkerson 62, Edmonds & Karp 72, Tarjan 83]:
A flow $f$ is minimum cost iff its residual graph $R$ has no negative cycles.
Theorem [Ford & Fulkerson 62, Edmonds & Karp 72, Tarjan 83]:

A flow $f$ is minimum cost iff its residual graph $R$ has no negative cycles.
Theorem [Ford & Fulkerson 62, Edmonds & Karp 72, Tarjan 83]:
A flow $f$ is minimum cost iff its residual graph $R$ has no negative cycles.
Decremental Property of Weighted Assignment (proof)

Theorem [Ford and Fulkerson 62, Edmonds and Karp 72, Tarjan 83]:

A flow $f$ is minimum cost iff its residual graph $R$ has no negative cycles.

$$\text{Cost}(p \oplus c) = \text{Cost}(p) + \text{Cost}(c) < \text{Cost}(p)$$
Decremental Property of Weighted Assignment (proof)

Possible combination of $p$ and $c$:
Compressed Graph

• Motivation: $O(nm^2 + nm \log n)$

  $O\left(\frac{nm^2}{\log m} + nm \log n\right)$

• Assuming a constant-sized label alphabet
• Using the notion of clique partition of a bipartite graph [Feder and Motwani 1991, Shamir and Tsur 1999]
Compressed Graph

- Each node in bipartite graph represents the whole subtree.

- Bounded alphabet ➔ the number of distinct trees is bounded.

- Lemma (a la [Shamir and Tsur, 1999]):
  - The number of distinct labeled rooted trees in a forest of $n$ vertices is: $O\left(\frac{n}{\log n}\right)$
Compressed Graph

Graph G

-7
-4
-2
-7
-4
-2
-4
-5
-3
0
0
0
0
0
0
0
Compressed Graph

Graph $G$

Graph $G^*$
Compressed Graph

• Lemma:
  – The assignment between \( u \in P \) and \( v \in T \) can be computed in time:
    \[
    O(d(u)(D(u)c(v) + c(v)\log c(v)))
    \]
    where \( E + V \log V \)
  – \( d(u) \) is the number of neighbors of node \( u \)
  – \( D(u) \) is the number of distinct trees in the forest of trees rooted at neighbors of \( u \)
  – \( c(v) \) is the number of children of node \( v \)
Compressed Graph

• Observation 2:

The sum of vertex degrees in an unrooted tree $P$ is:

$$\sum_{u=1}^{m} d(u) = 2m - 2$$

• Summing up the work over all $(v, u)$ pairs we get:

$$O(\sum_{u=1}^{m} \sum_{v=1}^{n} (d(u)(\text{clusters}_u c(v) + c(v) \log c(v)))) =$$

$$O(\sum_{u=1}^{m} (d(u) \times \text{clusters}_u \times n + d(u)n \log n) =$$

$$O(n \sum_{u=1}^{m} d(u)D(u) + mn \log n)$$
Time Complexity

• Lemma:

\[
\sum_{u=1}^{m} d(u)D(u) = O\left(\frac{m^2}{\log m}\right)
\]

• Thus the algorithm computes the optimal \textit{ALSH} solution for two rooted unordered trees in

\[
O\left(\frac{m^2 n}{\log m} + mn \log n\right)
\]
ALSH on Rooted Ordered Trees
ALSH on Rooted Ordered Trees
The main property: 
NO CROSS-EDGES 
in the bipartite graph

ALSH on Rooted Ordered Trees
The main property: **NO CROSS-EDGES in the bipartite graph** !!!
ALSH on Rooted Ordered Trees

\[ y_1 \quad y_2 \quad y_3 \]

\[ x_1 \quad x_2 \]

\[ k_{i+1} \]

\[ l_{j+1} \]
ALSH on Rooted Ordered Trees

\[ \sum_{i=1}^{m} \sum_{j=1}^{n} O(k_i \times l_j) = \sum_{i=1}^{m} O(n \times k_i) = O(nm) \]

Observation 1
ALSH on Unrooted Ordered Trees
(Cyclic order)
Cyclic String Comparison

T: BCBADEDCD

P: ABCDE

Destination: Column n's Maxima
Cyclic String Comparison
Cyclic String Comparison
Cyclic String Comparison
Cyclic String Comparison
Cyclic String Comparison

Diagram showing cyclic string comparison with arrows representing matches and mismatches.
Time Complexity

Real numbers score metric: $O(kl \log k)$
[Maez 1990]
Rational numbers score metric: $O(kl)$
[Schmidt 1998]
Time Complexity

Real numbers score metric: [Maez 1990]
Rational numbers score metric: [Schmidt 1998]

\[ O(kl \log k) \]

Observation 1

\[ \sum_{i=1}^{m} \sum_{j=1}^{n} O(k_i l_j \log k_i) = \sum_{i=1}^{m} O(n \times k_i \log k_i) = O(nm \log m) \]

\[ \sum_{i=1}^{m} \sum_{j=1}^{n} O(k_i \times l_j) = \sum_{i=1}^{m} O(n \times k_i) = O(nm) \]
ALSH on Unrooted Ordered Trees
(Linear order)
ALSH on Unrooted Ordered Trees (Linear order)

\[ y_1 \quad y_2 \quad \ldots \quad \ldots \quad \ldots \quad y_n \]

A
B
C
D
E

BACKWARD
ALSH on Unrooted Ordered Trees (Linear order)

Forward

Backward

searching the best $[i-1, j] + [i+1, j+1]$
ALSH on Unrooted Ordered Trees
(Linear order)

\[ y_1 \quad y_2 \quad \ldots \quad y_j \quad y_{j+1} \quad \ldots \quad y_n \]

**Forward**

**Backward**

searching the best \([i-1,j] + [i+1,j+1]\)
Time Complexity

• Weighted assignment for each pair \((u,v)\) can be computed in:

\[ O(kl) + O(kl) + k \times O(l) = O(kl) \]

• And summing up the work for all pairs \((u,v)\):

\[
\sum_{i=1}^{m} \sum_{j=1}^{n} O(k_i \times l_j) = \sum_{i=1}^{m} O(n \times k_i) = O(nm)
\]
Why is ALSH Suitable?

• A single enzyme in one pathway may replace a few consecutively acting enzymes in another pathway.
• The replacement can take place if
  – the replacing enzyme is multifunctional
    ▪ *can thus catalyze several consecutive reactions*, or
  – the enzyme uses an alternative catalysis
    ▪ *leads directly from the initial substrate to the final product*
• Enzymes that catalyze just a single reaction are more likely replaced, for both biochemical and parsimony reasons.
• Translating this into graph terms, degree-2 nodes may be deleted from the graph: perfectly captured by subtree homeomorphism.
Compressed Graph

<table>
<thead>
<tr>
<th></th>
<th>Graph G*</th>
<th>Graph G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Graph G*" /></td>
<td><img src="image2" alt="Graph G" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image3" alt="Graph G*" /></td>
<td><img src="image4" alt="Graph G" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image5" alt="Graph G*" /></td>
<td><img src="image6" alt="Graph G" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image7" alt="Graph G*" /></td>
<td><img src="image8" alt="Graph G" /></td>
</tr>
<tr>
<td>5</td>
<td><img src="image9" alt="Graph G*" /></td>
<td><img src="image10" alt="Graph G" /></td>
</tr>
<tr>
<td>6</td>
<td><img src="image11" alt="Graph G*" /></td>
<td><img src="image12" alt="Graph G" /></td>
</tr>
</tbody>
</table>

**Lemma:**

A min-cost max flow in $f^*$ corresponds directly to a min-cost max flow in $f$. 
The Sequence Alignment Problem

A = c t a c g a g a c

B = a a c g a c g a t

The Scoring Matrix $\delta$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>a</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>c</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>g</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Measure the similarity of A and B by finding an alignment of the strings which optimizes the similarity score.
The Sequence Alignment Problem

Optimal Global Alignment.

A = ctacga gac
B = aacgacgat

Similarity Score: 2

Optimal Local Alignment.

A = ctacga gac
B = aacgacgat

Similarity Score: 5