Approximate Graph Patterns for Biological Network

1) Conserved patterns of protein interaction in multiple species
   - Proceedings of the National Academy of Science (PNAS) - 2003

2) Conserved pathways within bacteria and yeast as revealed by global protein network alignment
   - Proceedings of the National Academy of Science (PNAS) - 2005

3) Automatic Parameter Learning for Multiple Network Alignment
   - Proceedings of the Computational Molecular Biology (RECOMB) - 2008

- Presented by
  Arijit Khan
  Computer Science
  University of California, Santa Barbara
Presentation Outline

1) Problem Formulation
   - Motivation
   - Multiple Network Alignment
   - Conserved Pathways
   - Scoring Function
   - Automatic Parameter Learning

2) Græmlin 2.0
   - Automatic Parameter Learning Protocol
   - Multiple Network Alignment Protocol

3) Comparison of Græmlin 2.0 with Existing Protocols
Problem Formulation

• MOTIVATION:

-- Understand the complex networks of interacting genes, proteins, and small molecules that give rise to biological form and function.

-- Understand Evolution and Mutation, which lead to change in protein structure.

-- to realize the protein – protein interaction among different species.

-- one way is to assign functional roles to interactions, thereby separating true protein-protein interactions from false positives.
Problem Formulation

• Multiple Network Alignment:

  INPUT - n networks, \( G_i = (V_i, E_i); 1 \leq i \leq n. \)

  Example: Protein Interaction Network, each \( G_i \) represents a different species, nodes represent proteins and edges represent interactions between proteins.

  OUTPUT - an equivalence relation \( \alpha \) over the nodes \( V = V_1 \cup V_2 \cup ... \cup V_n; \) that partitions \( V \) into a set of disjoint equivalence classes and has the maximum score determined by a scoring function.
Problem Formulation

- **Biological Interpretation:**
  - Nodes in the same equivalence class are functionally orthologous.
  - The subset of nodes in a local alignment represents a conserved module or pathway.
Problem Formulation

• **Scoring Function** $s$: 

  mapping $s: A \rightarrow \mathbb{R}$, where $A$ is the set of potential network alignments of $G_1, \ldots, G_n$.

  objective is to capture the “features” of a network alignment.
Problem Formulation

- **Feature Function** $f$: vector-valued function $f : \mathcal{A} \rightarrow \mathbb{R}^n$, which maps a global alignment to a numerical feature vector.

\[
f(a) = \begin{bmatrix}
\sum_{[x] \in a} f^N([x]) \\
\sum_{[x],[y] \in a, [x] \neq [y]} f^E([x],[y])
\end{bmatrix}
\]

- **Node feature function** $f^N$ maps equivalence classes to a feature vector. (e.g. Protein present, Protein count, Protein deletion, Protein duplication)

- **Edge feature function** $f^E$ maps pairs of equivalence classes to a feature vector. (e.g. edge deletion, paralog edge deletion)
Problem Formulation

• Parameter Vector $w$:
  Given a numerical parameter vector $w$, the score of an alignment $a$ is
  \[ s(a) = w \cdot f(a) \]

• Automatic Parameter Learning Problem

  - INPUT: training set of known alignments. The training set is a collection of $d$ training samples; each training sample specifies a set of networks $G(i) = G(i)_1, \ldots, G(i)_n$ and their correct alignment $a(i)$.

  - OUTPUT: numerical parameter vector $w$
Græmlin 2.0

- **Automatic Parameter Learning Protocol:**

**Loss Function:** \( \mathcal{L}: \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}^+ \)

Let \([x]_{a(i)}\) denote the equivalence class \( x \in V^{(i)} = \bigcup_j V^{(i)}_{j(i)} \) in \( a^{(i)} \) and \([x]_a\) denote the equivalence class of \( x \) in \( a \).

We define \( \mathcal{L}(a^{(i)}, a) = \sum_{x \in V^{(i)}} |[x]_a \setminus [x]_{a^{(i)}}| \)

So, loss function is the number of nodes aligned in \( a \) that are not aligned in the correct alignment \( a^{(i)} \).

\( \mathcal{L}(a^{(i)}, a) \) is \( 0 \) when \( a = a^{(i)} \) and positive when \( a \neq a^{(i)} \).

Intuitively, learned parameter vector, \( w \) should assign higher scores, \( s(a) = w \cdot f(a) \) to alignments \( a \) with smaller loss function values \( \mathcal{L}(a^{(i)}, a) \).
Græmlin 2.0

• **Automatic Parameter Learning Protocol:**

Formally, given a training set and loss function, the learned $w$ should score each training alignment $a^{(i)}$ higher than all other alignments $a$ by at least $\mathcal{L}(a^{(i)}, a)$.  

$$\forall i, a \in \mathcal{A}^{(i)}, w \cdot f(a^{(i)}) \geq w \cdot f(a) + \mathcal{L}(a^{(i)}, a)$$  

... [1]

$\mathcal{A}^{(i)}$ is the set of possible alignments of $\mathcal{G}^{(i)}$.

• Using Convex Programming, optimal $w$ minimizes

$$c(w) = \frac{1}{d} \sum_{i=1}^{d} r^{(i)}(w) + \frac{\lambda}{2} ||w||^2$$  

... [2]

Where

$$r^{(i)}(w) = \max_{a \in \mathcal{A}^{(i)}} (w \cdot f(a) + \mathcal{L}(a^{(i)}, a)) - w \cdot f(a^{(i)})$$

$d = \text{number of training samples}$

$\lambda = \text{regularization term used in Convex Programming} = 0$. 

Græmlin 2.0

• **Automatic Parameter Learning Protocol:**

Sub gradient Descent Technique to determine $w$.

$$g = \lambda w + \frac{1}{d} \sum_{i=1}^{d} (f(a^{(i)}_*) - f(a^{(i)}));$$

where $a^{(i)}_* = \arg \max_{a \in A^{(i)}} \left( w \cdot f(a) + \mathcal{L}(a^{(i)}, a) \right)$

$$w = (w - \alpha g) \text{ iteratively}; \; \alpha \text{ is learning rate} = 0.05$$

stop when it performs 100 iterations that do not reduce the objective function.

• At each iteration it uses the loss function and the current $w$ to compute the optimal alignment.
• Then decreases the score of features with higher values in the optimal alignment than in the training sample.
• increases the score of features with lower values in the optimal alignment than in the training sample.
Græmlin 2.0

Automatic Parameter Learning Protocol:

\[
\text{LEARN}(\{G_1^{(i)}, \ldots, G_n^{(i)}, a(i)\}_{i=1}^d : \text{training set}, \alpha : \text{learning rate}, \lambda : \text{regularization})
\]

1. \text{var} w \leftarrow 0 // \text{the current parameter vector}
2. \text{var} m_* \leftarrow \infty // \text{a measure of progress}
3. \text{var} w_* \leftarrow w // \text{the best parameter vector so far}
4. \text{while} m_* \text{ updated in last 100 iterations}
5. \text{do}
6. \quad \text{var} g \leftarrow 0 // \text{the current subgradient}
7. \quad \text{var} m = 0 // \text{the current margin}
8. \quad \text{for} i = 1 : d
9. \quad \text{do} // \text{sum over all training samples}
10. \quad \quad \text{var} a_*^{(i)} = \text{ALIGN}(G_1^{(i)}, \ldots, G_n^{(i)}, w, \mathcal{L})
11. \quad \quad g \leftarrow g + f(a_*^{(i)}) - f(a^{(i)}) // \text{update the subgradient}
12. \quad m \leftarrow m + w \cdot f(a_*^{(i)}) + \mathcal{L}(a^{(i)}, a_*^{(i)}) - w \cdot f(a^{(i)}) // \text{update the margin}
13. \quad g \leftarrow \frac{1}{d} g - \lambda w; m \leftarrow \frac{1}{d} m + \frac{\lambda}{2} \|w\|^2 // \text{add in regularization}
14. \quad \text{if} m < m_*
15. \quad \quad \text{then}
16. \quad \quad \quad m_* \leftarrow m; w_* = w // \text{update the best parameter vector}
17. \quad \quad \text{w} \leftarrow \text{w} - \alpha g // \text{update parameter vector}
18. \quad \text{return} w_*$
Græmlin 2.0

- **Automatic Parameter Learning Protocol:**

  \[
  a_{*}^{(i)} = \arg \max_{a \in A^{(i)}} \left( w \cdot f(a) + L(a^{(i)}, a) \right)
  \]

  - Multiple Alignment Problem augmented by a loss function.

  - At each iteration of Automatic Parameter Learning protocol, Multiple Alignment Algorithm is applied.

  - Learning algorithm converges at a linear rate to a small region surrounding the optimal \( w \).
Græmlin 2.0

- Multiple Alignment Problem:

  - local Hill-Climbing algorithm (iterative).

  - each iteration, it processes each node and evaluates a series of moves for each node:

  1) Leave the node alone.
  2) Create a new equivalence class with only the node.
  3) Move the node to another equivalence class.
  4) Merge the entire equivalence class of the node with another equivalence class.

  - For each move, algorithm computes the score before and after the move and performs the move that increases the score the most.

  - stops when an iteration does not increase the alignment score.
Græmlin 2.0

Multiple Alignment Problem

\( \text{ALIGN}(G_1, \ldots, G_n : \text{set of networks}, \ w : \text{parameter vector}, \ \mathcal{L} : \text{optional loss function}) \)

1. \textbf{var} \( a \leftarrow \text{an alignment with one equivalence class per node} \)
2. \textbf{while} \ true
3. \textbf{do}
4. \quad \textbf{var} \( \Delta_t = 0 \) \ // \text{the total change in score of this iteration}
5. \quad \textbf{for each node} \( n \in \bigcup_i G_i \)
6. \quad \textbf{do}
7. \quad \quad \textbf{var} \( \Delta^* \leftarrow 0 \) \ // \text{best score}
8. \quad \quad \textbf{var} \( m^* \leftarrow \text{undef} \) \ // \text{best move}
9. \quad \quad \textbf{for each move} \( m \)
10. \quad \quad \textbf{do}
11. \quad \quad \quad \textbf{var} \( a_t \leftarrow m(a) \) \ // \text{alignment after move} \( m \)
12. \quad \quad \quad \Delta \leftarrow w \cdot f(a_t) + \mathcal{L}(a_t) - (w \cdot f(a) + \mathcal{L}(a)) \ // \text{change in score after move} \( m \)
13. \quad \quad \quad \textbf{if} \( \Delta > \Delta^* \)
14. \quad \quad \quad \textbf{then}
15. \quad \quad \quad \quad \Delta^* = \Delta; \ m^* = m \ // \text{new best move}
16. \quad \quad \quad \quad a \leftarrow m^*(a) \ // \text{do best move on alignment}
17. \quad \quad \quad \Delta_t \leftarrow \Delta_t + \Delta^* \ // \text{update total change in score of this iteration}
18. \quad \quad \quad \textbf{if} \( \Delta_t = 0 \)
19. \quad \quad \quad \textbf{then} \ \text{break}
20. \quad \quad \textbf{return} \ w
Græmlin 2.0

- **Multiple Alignment Problem**

- Algorithm is approximate but efficient.
- running time = \(O(b \cdot c \cdot (n + m))\)
  
  \(b = \) number of iterations
  
  \(c = \) average number of candidate classes in each iteration
  
  \(n = \) number of nodes
  
  \(m = \) number of edges

- \(b < 10\) (empirically)
  
  \(c = \) can be huge; but can be small if we neglect classes with BLAST e-value

  \(\ll 10^{-5}\)

- linear in \((n + m)\)
COMPARISON ANALYSIS

- Tested on 3 different Network Datasets:
  a) Human and Mouse IntAct Networks
  b) Yeast and Fly DIP Networks
  c) Stanford Network Database (SNDB)

- **Specificity Measurement** Metrics:
  1. the fraction of equivalence classes that were correct ($C_{eq}$)
  2. the fraction of nodes that were in correct equivalence classes ($C_{node}$)

- **Sensitivity Measurement** Metrics:
  1. the total number of nodes in correct equivalence classes ($C_{or}$)
  2. the total number of equivalence classes with k species, for $k = 2, \ldots, n$

- Compared with NetworkBLAST, MaWISh, Graemlin 1.0, lSO RANK, and Graemlin-global alignment protocols.
### COMPARISON ANALYSIS

#### Average consistency equivalence class consistency

<table>
<thead>
<tr>
<th></th>
<th>SNDB</th>
<th>IntAct</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>eco/stm</td>
<td>eco/cce</td>
<td>6-way</td>
</tr>
<tr>
<td>Local aligner comparisons</td>
<td>$C_{eq}$</td>
<td>$C_{node}$</td>
<td>$C_{eq}$</td>
</tr>
<tr>
<td>NB</td>
<td>0.77</td>
<td>0.45</td>
<td>0.78</td>
</tr>
<tr>
<td>Gr2.0</td>
<td>0.95</td>
<td>0.94</td>
<td>0.79</td>
</tr>
<tr>
<td>MW</td>
<td>0.84</td>
<td>0.64</td>
<td>0.77</td>
</tr>
<tr>
<td>Gr2.0</td>
<td>0.97</td>
<td>0.96</td>
<td>0.77</td>
</tr>
<tr>
<td>Gr</td>
<td>0.80</td>
<td>0.77</td>
<td>0.69</td>
</tr>
<tr>
<td>Gr2.0</td>
<td>0.96</td>
<td>0.95</td>
<td>0.82</td>
</tr>
<tr>
<td>Global aligner comparisons</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GrG</td>
<td>0.86</td>
<td>0.86</td>
<td>0.72</td>
</tr>
<tr>
<td>Iso</td>
<td>0.91</td>
<td>0.91</td>
<td>0.65</td>
</tr>
<tr>
<td>Gr2.0</td>
<td>0.96</td>
<td>0.96</td>
<td>0.78</td>
</tr>
</tbody>
</table>

- Eco = E. coli, Stm = S. typhimurium, cce = C. crescentus, hsa = human, mmu = mouse, sce = yeast, dme = fly
## COMPARISON ANALYSIS

### Number of nodes in correct equivalence classes

<table>
<thead>
<tr>
<th></th>
<th>SNDB</th>
<th></th>
<th></th>
<th>IntAct</th>
<th></th>
<th></th>
<th>DIP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>eco/stm</td>
<td>eco/cce</td>
<td>6-way</td>
<td>hsa/mmu</td>
<td>sce/dme</td>
<td>3-way</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cor</td>
<td>Tot</td>
<td>Cor</td>
<td>Tot</td>
<td>Cor</td>
<td>Tot</td>
<td>Cor</td>
<td>Tot</td>
<td>Cor</td>
</tr>
<tr>
<td>NB Gr2.0</td>
<td>457</td>
<td>1016</td>
<td>346</td>
<td>697</td>
<td>–</td>
<td>–</td>
<td>65</td>
<td>1010</td>
<td>43</td>
</tr>
<tr>
<td>MW Gr2.0</td>
<td>1309</td>
<td>2050</td>
<td>458</td>
<td>841</td>
<td>–</td>
<td>–</td>
<td>87</td>
<td>241</td>
<td>10</td>
</tr>
<tr>
<td>Gr Gr2.0</td>
<td>985</td>
<td>1286</td>
<td>546</td>
<td>847</td>
<td>1524</td>
<td>2287</td>
<td>108</td>
<td>203</td>
<td>35</td>
</tr>
<tr>
<td>Gr2.0</td>
<td>2026</td>
<td>–</td>
<td>1014</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>306</td>
<td>–</td>
<td>534</td>
</tr>
</tbody>
</table>

**Local aligner comparisons**

**Global aligner comparisons**
COMPARISON ANALYSIS

Number of species per equivalence class

- Gr
- GrG
- Gr2.0
CONCLUSION

• Græmlin 2.0 is a multiple global network aligner protocol.

• Automatically learn the scoring function’s parameters.

• The feature function isolates the biological meaning of network alignment.

• Align multiple networks approximately in linear time.

• Learning Algorithm also converges linearly.

• Higher specificity and higher sensitivity.
QUESTIONS / COMMENTS / DOUBTS ???

Thank You !!!