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.

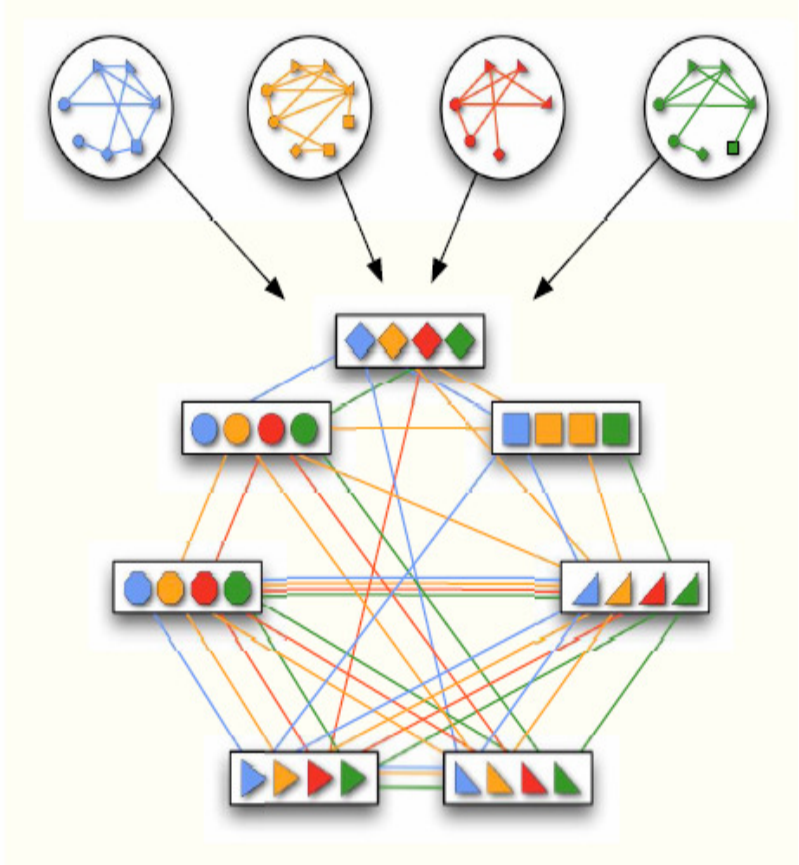
OUTPUT - an equivalence relation *a* over the nodes $V = V_1 \cup V_2 \cup \ldots \cup V_n$; that partitions V into a set of disjoint equivalence classes and has the maximum score determined by a scoring function.

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 $s : \mathcal{A} \rightarrow \mathbb{R}$, 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 $\mathbf{f} : \mathcal{A} \to \mathbb{R}^n$, which maps a global alignment to a numerical feature vector.

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

-node feature function $\mathbf{f}^N$ maps equivalence classes to a feature vector.

(e.g. Protein present, Protein count, Protein deletion, Protein duplication)

-edge feature function $\mathbf{f}^E$ 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 , . . .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 $x \in V^{(i)} = \bigcup_j V_j^{(i)}$ 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 $\mathcal{L}(a^{(i)}, a)$.

$$\forall i, a \in \mathcal{A}^{(i)}, \mathbf{w} \cdot \mathbf{f}(a^{(i)}) \geq \mathbf{w} \cdot \mathbf{f}(a) + \mathcal{L}(a^{(i)}, a) \quad \text{... [1]}$$

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

- Using Convex Programming, optimal *w* minimizes

$$c(\mathbf{w}) = \frac{1}{d} \sum_{i=1}^{d} r^{(i)}(\mathbf{w}) + \frac{\lambda}{2} ||\mathbf{w}||^2 \quad \text{... [2]}$$

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

d = number of training samples

λ = regularization term used in Convex Programming = *0*.

Græmlin 2.0

- Automatic Parameter Learning Protocol:

  Sub gradient Descent Technique to determine *w.*

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

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

  $w = (w - \alpha\, g)$ iteratively; α 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:

LEARN($\{G_1^{(i)}, \ldots, G_n^{(i)}, a^{(i)}\}_{i=1}^{d}$: training set , α : learning rate , λ : regularization)

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

Græmlin 2.0

- Automatic Parameter Learning Protocol:

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

- 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

```
ALIGN(G_1, ..., G_n : set of networks , w : parameter vector , L : optional loss function )
 1  var a ← an alignment with one equivalence class per node
 2  while true
 3  do
 4      var Δ_t = 0 //the total change in score of this iteration
 5      for each node n ∈ ⋃_i G_i
 6      do
 7          var Δ* ← 0 //best score
 8          var m* ← undef //best move
 9          for each move m
10          do
11              var a_t ← m(a) //alignment after move m
12              Δ ← w · f(a_t) + L(a_t) − (w · f(a) + L(a)) //change in score after move m
13              if Δ > Δ*
14                 then
15                      Δ* = Δ; m* = m //new best move
16          a ← m*(a) //do best move on alignment
17          Δ_t ← Δ_t + Δ* //update total change in score of this iteration
18      if Δ_t = 0
19         then break
20  return w
```

Græmlin 2.0

- Multiple Alignment Problem

- Algorithm is approximate but efficient.
- running time = O(b . c . (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

  << 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, . . . , n

- Compared with NetworkBLAST, MaWISh, Graemlin 1.0, IsoRank, and Graemlin-global alignment protocols.

COMPARISON ANALYSIS

Average consistency equivalence class consistency

	SNDB						IntAct		DIP			
	eco/stm		eco/cce		6-way		hsa/mmu		sce/dme		3-way	
	C_{eq}	C_{node}	C_{eq}	C_{node}	C_{eq}	C_{node}	C_{eq}	C_{node}	C_{eq}	C_{node}	C_{eq}	C_{node}
Local aligner comparisons												
NB	0.77	0.45	0.78	0.50	–	–	0.33	0.06	0.39	0.14	–	–
Gr2.0	**0.95**	**0.94**	**0.79**	**0.78**	–	–	**0.83**	**0.81**	**0.58**	**0.58**	–	–
MW	0.84	0.64	**0.77**	0.54	–	–	0.59	0.36	0.45	0.37	–	–
Gr2.0	**0.97**	**0.96**	**0.77**	**0.76**	–	–	**0.88**	**0.86**	**0.90**	**0.91**	–	–
Gr	0.80	0.77	0.69	0.64	0.76	0.67	0.59	0.53	0.33	0.29	0.23	0.15
Gr2.0	**0.96**	**0.95**	**0.82**	**0.81**	**0.86**	**0.85**	**0.86**	**0.84**	**0.61**	**0.61**	**0.57**	**0.57**
Global aligner comparisons												
GrG	0.86	0.86	0.72	0.72	0.80	0.81	0.64	0.64	0.68	0.68	0.71	0.71
Iso	0.91	0.91	0.65	0.65	–	–	0.62	0.62	0.63	0.63	–	–
Gr2.0	**0.96**	**0.96**	**0.78**	**0.78**	**0.87**	**0.87**	**0.81**	**0.80**	**0.73**	**0.73**	**0.76**	**0.76**

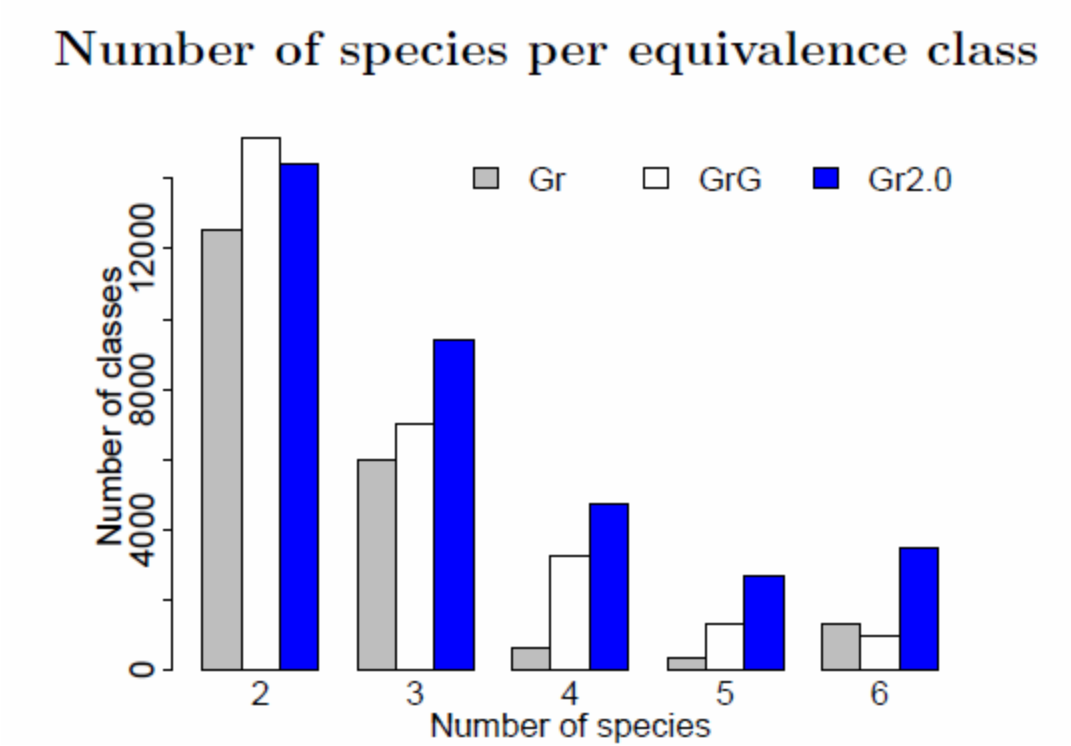
- Eco = E. coli, Stm = S. typhimurium, cce = C. crescentus, hsa = human, mmu = mouse, sce = yeast, dme = fly

COMPARISON ANALYSIS

Number of nodes in correct equivalence classes

	SNDB						IntAct		DIP			
	eco/stm		eco/cce		6-way		hsa/mmu		sce/dme		3-way	
	Cor	Tot	Cor	Tot	Cor	Tot	Cor	Tot	Cor	Tot	Cor	Tot
Local aligner comparisons												
NB	457	1016	346	697	–	–	65	1010	43	306	–	–
Gr2.0	**627**		**447**		–		**258**		**155**		–	
MW	1309	2050	458	841	–	–	87	241	10	27	–	–
Gr2.0	**1611**		**553**		–		**181**		**20**		–	
Gr	985	1286	546	847	1524	2287	108	203	35	122	27	180
Gr2.0	**1157**		**608**		**2216**		**151**		**75**		**86**	
Global aligner comparisons												
GrG	1496	–	720	–	2388	–	268	–	384	–	564	–
Iso	**2026**		**1014**		–		306		534		–	
Gr2.0	2024		1012		**3578**		**350**		**637**		**827**	

COMPARISON ANALYSIS



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 !!!