Virtual Network Topology (VNT)

- Find a topology optimizing
  - Minimizing the link with heaviest load
  - Minimizing maximum delay between 2 hops
  - Number of hops

Outline

- Problem Statement
- System Model
- Results
- Conclusion

VNT Controller

- Protein-gene mechanism (Furusawa08)

\[
\frac{dx_i}{dt} = f \left( \sum_{j=1}^{m} w_{ij}x_j \right) y_i + (1 - y_i)x_i + \eta
\]

- If the system is in
  - good conditions: deterministic behavior
  - bad conditions: stochastic behavior
- Embed the topology information into a neural network

Previous Models

<table>
<thead>
<tr>
<th>Previous Work</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Builds a traffic matrix</td>
<td>Uses link load information</td>
</tr>
<tr>
<td>Not adaptive</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Mostly offline</td>
<td>Online</td>
</tr>
</tbody>
</table>

Hebbian Learning

- Simple and the most widely used

\[
\Delta w_{i,j} = \rho w_i x_j
\]

- Low computation times
Oja Learning

- Weight matrix is similar to Hebbian
- \[ \Delta w_i = \alpha(x_i y_i - y^2 w_i) \]
- Similar computation times as Hebbian

Oja vs Hebbian Learning

- Oja shows less variance in terms of variance

Adaptive Principal Component Extraction (APEX)

- Additional weight matrix: Lateral weight matrix
  \[ y = Wx + Py \]
- \[ \Delta p_{k,j} = \alpha(y_i y_j - p_{k,j} y_i^2) \]
- Converges quicker than Oja [Kung90]

Hebb vs APEX and Oja

APEX and Oja performs better than Hebbian

The Calculation Times

- Oja and Hebbian shows similar calculation times

The Number of Attractors

- As the number of attractors increases, the performance gain reduces
Orthogonal Projection

- $X^+$ Pseudo inverse
- $W_o = X^+X \cdot X^+$
- 100X slower than the Hebbian

Conclusion

- $X^+$, APEX and Orthogonal projection performs better than Hebbian
- Orthogonal projection is 100X slower
- Oja is best when the number of attractors are less than 20
- APEX is best for other cases

References