# Virtual Network Topology Control with Oja and APEX Learning

Y. Sinan Hanay, Yuki Koizumi, Shin'ichi Arakawa and Masayuki Murata
Graduate School of Information Sciences and Technology
Osaka University
Suita, Osaka 565-0871, Japan
Email: {s-hanay, ykoizumi, arakawa, murata}@ist.osaka-u.ac.jp

***Abstract*—Virtual Network Topology (VNT) is any possible topology established between a subset of nodes in optical wavelength-division multiplexing (WDM) networks. This virtual topology can be designed according to the traffic load of the network. VNT control searches for a suitable virtual topology satisfying the specified requirements. Previously, VNT control based on attractor selection using Hebbian learning has been studied. This work extends the previous work with the inclusion of two new learning algorithms (i.e. Oja and APEX learning) and orthogonal projection with the aim of increasing the stability of attractors. Our results show that, both methods achieves better performance than Hebbian learning.**

## I. Introduction

Optical communication networks provide higher bandwidth than the electronic networks. WDM provides carrying of multiple channels through a single fiber by using different wavelengths in a similar fashion to frequency division multiplexing (FDM). In each optical node, there are a limited number of receivers and transmitters. The nodes are connected to each other with optical crossconnects (OXCs). Lightpaths are formed between nodes by wavelength allocation. Since there are limited numbers of receivers and transmitters on a node, it is not possible to form light paths between every node pairs. Thus, some nodes are connected to each other, and the underlying topology in which lightpaths continuity is preserved, is called virtual topology. It is an intriguing problem to determine the most suitable network topology under various network traffic loads.

Previous research on VNT control focused on two different approaches: online and offline. In online approaches, the network conditions are monitored continuously and the network topology is changed by changing the lightpaths as needed [1], [2], [3], [4]. In offline approaches, the network topology is constructed only once [5], [6]. These methods either assume traffic characteristics do not change significantly or can be predicted. The drawback of the offline approaches is that the VNT may not be as efficient if the network condition change cannot be predicted precisely.

## II. Related Work

VNT control with mixed integer linear programming (MILP) was studied in [1]. The proposed method update network topology based on load imbalances inside the network. Similarly, another online approach uses load balance information and use heuristics for topology control was proposed in [4].

This work extends the attractor selection based VNT control, which was studied in [7]. In the previous work, Hebbian learning algorithm was used to embed the attractors. In this work, instead of Hebbian learning, we considered some other learning algorithms such as Oja and APEX. Also, we analyzed the orthogonal projection method to increase the stability of attractors.

## III. Problem Setting

In VNT control, various performance metrics can be used: such as average number of hops, minimizing maximum delay between two hops and

minimizing maximum utilization on a link. In this work, we consider minimizing the maximum link utilization.

In the simulations, the network consist of 100 nodes, with each node having 16 transmitters and 16 receivers. In 100 network nodes, there are 9900 possible node pairs. Thus the attractor vector is a binary row vector of length 9900. Since the system has more than one attractor, the attractor matrix consists of attractor vectors.

Figure 1 demonstrates the virtual network topology configuration on a physical network. Based on the traffic demand one of the two VNTs can be used to utilize the network.

## IV. Model

Activity is the condition of the network, and when the network is in favourable conditions, the deterministic behaviour dominates. On the other hand, when the network conditions are poor; the stochastic behaviour dominates.

The VNT control model based on attractor selection is same as the one proposed in [2].

### A. Attractor Selection

In this work, we extend VNT control based on attractor selection that is based on a biological gene regulatory network which was proposed in [8]. VNT topology control by attractor selection is first proposed in [7], and its adaptability was studied in [2]. Similar to [8], we use a system variable called expression level, denoted as $x_i$ for the link i, to determine if a link should be established or teared down. If $x_i$ is greater than 0.5 the link is established; otherwise it is teared down.

VNT control mechanism selects the appropriate virtual topology based on network conditions. We embed some possible VNTs to the system by crafting them as attractors. Through the weight matrix, attractors can be memorized based on the neural learning algorithm.

$$\frac{dx_i}{dt} = f\left(\sum_{j=1}^{n} w_{ij}x_j\right)V_g + (1 - V_g)x_i + \eta \quad (1)$$

Here, $V_g$ is the activity, $\eta$ is White Gaussian Noise and $f(.)$ is the sigmoid function, which is

$$f(z) = \frac{1}{1 + e^{-\mu z}} \quad (2)$$

, and

$$V_g = \frac{\gamma}{1 + e^{\sigma(\mu_{max} - \zeta)}} \quad (3)$$

.

In the above equation, $\zeta$ represents the threshold for activity; $\gamma$ is a system parameter that scales $V_g$ and $\mu_{max}$ is the maximum utilization. So higher $V_g$ means that the system is in a more preferable condition than a system with a lower $V_g$.

## V. Background

In this section, we present the necessary background information before proceeding to our method. In associative networks, it is better using bipolar (-1,1) coding rather than the binary coding (0,1) [9]. Thus, we use the simple conversion of $w_p = (2 \times w_n - 1)$. For clarity of the discussion, we will not embed that modification in the following sections.

### A. Hebbian Learning

In the general case, the weight matrix W is constructed for a pattern vector X, as $W_c = X^T X$.

Hebbian learning

$$\Delta w_{i,j} = \rho x_i x_j \quad (4)$$

W becomes the correlation matrix,

$$x^p \cdot W = x^p(x^p \cdot x^p) + \sum_{l=p}^{m} \neq x^l(x^l \cdot x^p). \quad (5)$$

Here aim is to develop a new learning technique to minimize the cross talk term $\sum_{l=p}^{m} \neq x^l(x^l \cdot x^p)$.

### B. Orthogonal Projection

Any input vector is projected linearly by weight matrix $W_c$, but this projection is not necessarily orthogonal. Non-orthogonal projection may lead to suboptimal selection by falsifying the information [9]. This situation can be avoided by orthogonal projection. For orthogonal projection, the weight

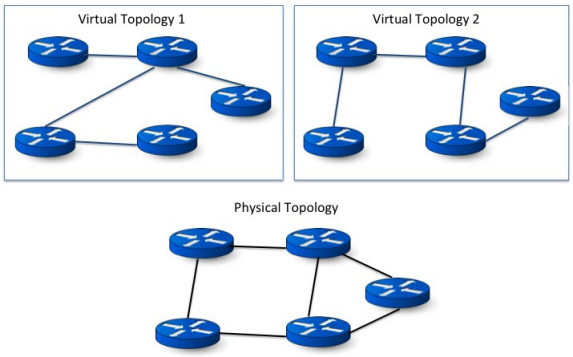


Fig. 1. Virtual Network Topology configuration

matrix $W_o = X^+X$. $X^+$is Moore-Penrose pseudoinverse of matrix X. If X is not singular then, $X^+ = X^{-1}$, otherwise

$X^+$ is a unique matrix satisfying the following :

1) $X^+XX^+ = X^+$
2) $XX^+X = X$
3) $XX^+$ is hermitian.

Thus, $W_o$ projects any input vector x to linear subspace spanned by stored input vectors $X$ orthogonally. Thus new learning algorithm needs to construct $W = W_o$.

### C. Oja Learning

Oja learning is a modified version of the Hebb learning, introduced by Oja [10]. For Oja learning, weight are ajdusted according to

$$\Delta w_i = \alpha(x_i y_i - y^2 w_i) \tag{6}$$

### D. APEX Learning

APEX learning algorithm is proposed in [11]. In APEX learning, the output y is calculated as

$$y = Wx + Py \tag{7}$$

Here weight matrix W is same as in Oja learning algorithm, and the weights $p_{i,j}$ for lateral weight matrix P is calculated as

$$\Delta p_{i,j} = \alpha(y_i y_j - p_{i,j} y_i^2) \tag{8}$$

It is shown that APEX converges quicker than the Oja learning [11]. For that reason, we included APEX in our work.

## VI. Results

For Hebbian learning, weight matrix $W = X^T \times X$, to increase stability of attractors orthogonal projection can be used, $W = X^+ \times X$. For 100 nodes, there are 9900 links, and weight matrix is 9900 by 9900. For Oja learning, weight are ajdusted according to $\Delta w_i = \alpha(x_i y_i - y^2 w_i)$

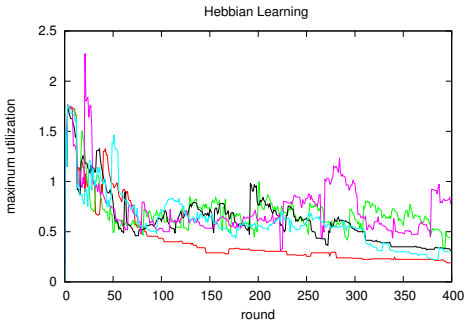


Fig. 2. Hebbian Learning with 5 different trials. Only one trial (i.e. the red one) converges to an attractor that achieves a maximum utilization less than the pre-specified threshold.

Figure 4-7 shows the activity for the three learning algorithms for varying number of attractors. The

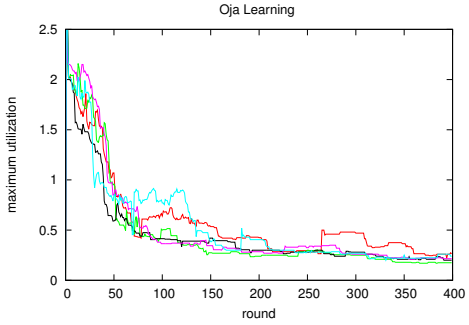


Fig. 3. Oja Learning with 5 different trials. In all the trials, the network converges to an attractor that achieves a maximum utilization under the pre-specified threshold.

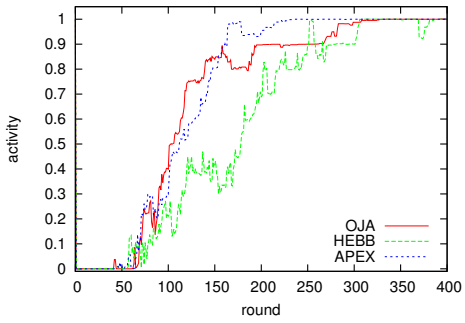


Fig. 5. Activity for 10 attractors. The figure shows a similar behaviour to the case of 5 attractors, while in this case Hebbian learning succeeds in reaching the optimum conditions.

simulations were run for 20 trials, and the average of 20 runs were taken. Figure 4 and 6 show a similar behaviour in which APEX achieves the highest activity in earlier rounds, and then Oja learning and lastly Hebbian learning. On the other hand Figure 6 and 7 shows that for 20 and 30 attractors, Oja learning achieves the best performance, and then APEX and lastly Hebbian learning algorithm. From these figures we can conclude that APEX algorithm is most suitable when the number of attractors is less than 10 and Oja learning is best when there are more than 20 attractors.

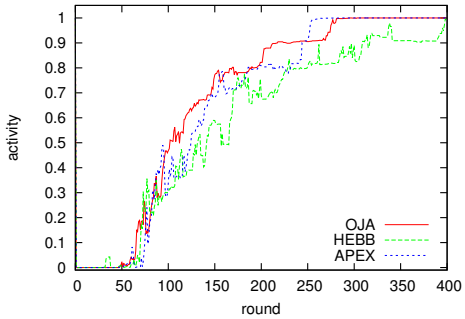


Fig. 6. Activity for 20 attractors. All three algorithms shows similar behaviour while Oja and APEX is slightly better.

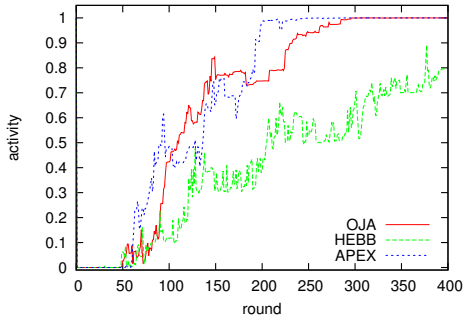


Fig. 4. Activity for 5 attractors. The plot shows that APEX converges to optimum conditions quickest, while Hebbian could not reach the optimum conditions.

Figure 2 and 3 shows the maximum utilization for the two learning methods with 5 matched trials. The Hebbian learning shows more variance in performance, and not all the trials converge to an attractor in 400 rounds. However, the system converged an attractor in every case with Oja learning. Also, Oja learning showed a smaller variance in performance.

Figure 8 shows the cumulative calculation times for 3 different learning methods. The learning algorithms take similar times in total. Clearly, APEX has the heaviest computation complexity, while Hebbian is the lightest. The figure reveals that situation at the beginning where initial weight matrix calculation occurs. The weight matrix calculation is fastest with Hebbian learning due to the relative simplicity. However, Hebbian performs poorly than the other methods in converging an attractor. As a consequence, the system calculates weight matrix more often in the Hebbian learning approach. Thus, overall Oja is the fastest, while APEX is the slowest.

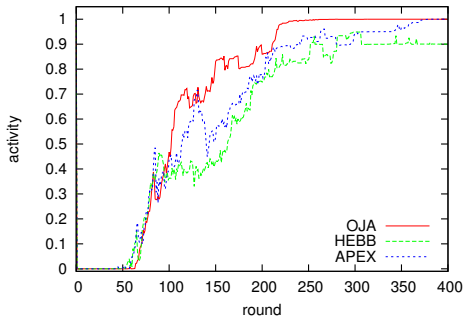


Fig. 7. Activity for 30 attractors. Oja and APEX learning achieves the optimum conditions under 400 rounds, while Hebbian could not reach it.

The figure shows the calculation times for 10 attractors; however we saw similar behaviour different numbers of attractors, as well.

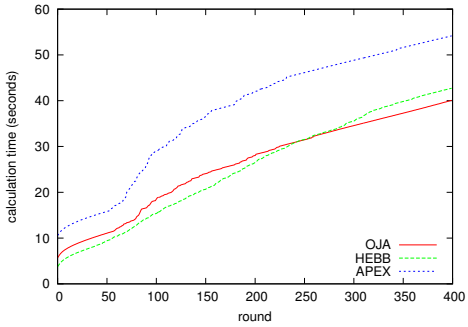


Fig. 8. Calculation times for 10 attractors (Cumulative).

Figure 9 and 10 shows the maximum utilization for 5 and 30 attractors. It reveals that, the difference in the maximum utilization among the three learning algorithms reduces.

In the next set of simulations, we compared orthogonal projection against Hebbian learning. Figure 11 and 12 reveals that the orthogonal projection is better in terms of faster convergence and performance. However, as 13 shows, orthogonal projection is slower than the Hebbian learning about two orders of magnitude.

This is due to heavy matrix calculations involved (i.e. "pseudoinverse") with the orthogonal projection. A fast pseudoinverse method is presented in

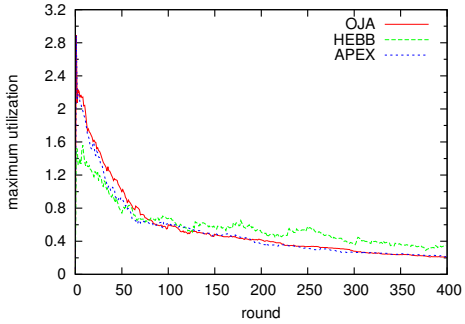


Fig. 9. Maximum utilization, 5 attractors. The figure shows, for 5 attractors, Oja and APEX succeeds in finding a virtual topology of which most heavily loaded link stays under the threshold, while Hebbian could not succeed.

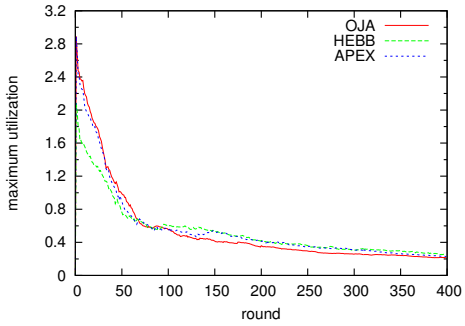


Fig. 10. Maximum utilization, 30 attractors.

[12] for rank deficient matrices. We implemented that method, however, since the matrix (attractor matrix in that case) is not rank deficient; that method did not provide any improvement on run time. In the figure, the calculation time for the orthogonal projection takes about 90 seconds, and it happens two times; whereas for Hebbian case it takes around a few seconds.

## VII. Conclusion

In this paper, we extended the previous work on VNT control based on attractor selection by adding two new learning methods. We used Oja, APEX learning algorithms and orthogonal projection to increase the stability of attractors which is lower with the Hebbian learning.

Simulation results showed that Oja, APEX and

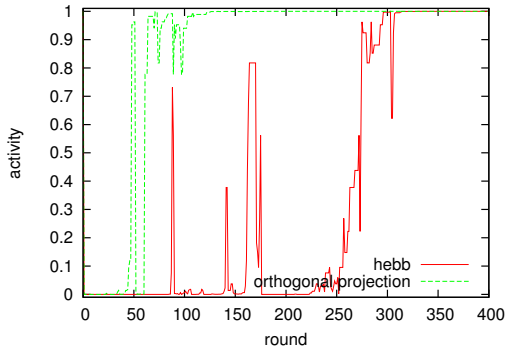


Fig. 11. Activity

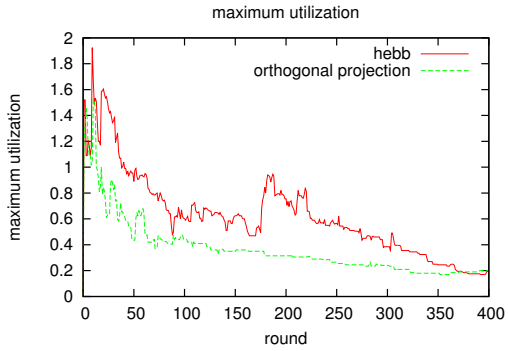


Fig. 12. Maximum utilization

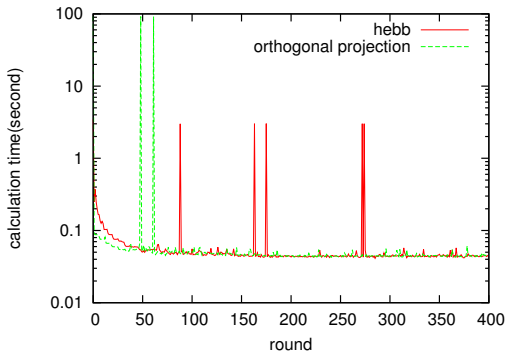


Fig. 13. Calculation time

orthogonal projection increases stability of attractors and thus provide better performance by reducing the maximum link utilization on a given link. However, long calculation times make orthogonal projection a less viable solution. APEX performs best when the number of attractors are 10 or less, while Oja performs best for 20 or more attractors.

## REFERENCES


[1] A. Gençata and B. Mukherjee, "Virtual-topology adaptation for WDM mesh networks under dynamic traffic." *IEEE/ACM Trans. Netw.*, pp. 236–247, 2003.

[2] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Adaptive virtual network topology control based on attractor selection," *J. Lightwave Technol.*, vol. 28, no. 11, pp. 1720–1731, Jun 2010. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-28-11-1720

[3] B. Ramamurthy and A. Ramakrishnan, "Virtual topology reconfiguration of wavelength-routed optical WDM networks," in *Proceedings of the GLOBECOM '00. IEEE Global Telecommunications Conference*, vol. 2, 2000, pp. 1269–1275. [Online]. Available: http://dx.doi.org/10.1109/GLOCOM.2000.891340

[4] S. F. Gieselman, N. K. Singhal, and B. Mukherjee, "Minimum-cost topology adaptation for an IPS's mesh network," in *Proceedings of the Optical Fiber Communication Conference and Exposition and The National Fiber Optic Engineers Conference*. Optical Society of America, 2005, p. OTuP3. [Online]. Available: http://www.opticsinfobase.org/abstract.cfm?URI=OFC-2005-OTuP3

[5] B. Chen, G. N. Rouskas, and R. Dutta, "Clustering methods for hierarchical traffic grooming in large-scale mesh WDM networks," *J. Opt. Commun. Netw.*, vol. 2, no. 8, pp. 502–514, Aug 2010. [Online]. Available: http://jocn.osa.org/abstract.cfm?URI=jocn-2-8-502

[6] F. Ricciato, S. Salsano, A. Belmonte, and M. Listanti, "Off-line configuration of a MPLS over WDM network under time-varying offered traffic," in *Proceedings of the IEEE INFOCOM02*. Press, 2002, pp. 57–65.

[7] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Stability of virtual network topology control for overlay routing services," *J. Opt. Netw.*, vol. 7, no. 7, pp. 704–719, Jul 2008. [Online]. Available: http://jon.osa.org/abstract.cfm?URI=jon-7-7-704

[8] C. Furusawa and K. Kaneko, "A generic mechanism for adaptive growth rate regulation," *PLoS Comput Biol*, vol. 4, p. e3, 01 2008.

[9] R. Rojas, *Neural Networks - A Systematic Introduction*. Berlin: Springer-Verlag, 1996.

[10] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, November 1982. [Online]. Available: http://dx.doi.org/10.1007/BF00275687

[11] S. Kung and K. Diamantaras, "A neural network learning algorithm for adaptive principal component extraction (apex)," in *Proceedings of the Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, apr 1990, pp. 861 –864 vol.2.

[12] P. Courrieu, "Fast computation of moore-penrose inverse matrices," *Neural Information Processing - Letters and Reviews*, 2005.