# Designing VNT Candidates Robust Against Network Failures

*Onur Alparslan, Shin'ichi Arakawa, Masayuki Murata*
*Graduate School of Information Science and Technology*
*Osaka University*
*1-5 Yamadaoka, Suita, Osaka 565-0871, Japan*
*{a-onur,arakawa,murata}@ist.osaka-u.ac.jp*

## Abstract

Future-generation networks are expected to be more robust to network failures. However, as each physical link carries multiple lightpaths when a VNT (Virtual Network Topology) is applied on an optical network, even the failure of a single link may tear down many links in the VNT, which can slow down the network or make it unusable. In this paper, we propose an algorithm called MFLDA (Minimum Flow Logical topology Design Algorithm) for designing VNT candidates that can accommodate a wide range of traffic patterns. Moreover, we show that the variant called MFLDA-FO (MFLDA with Failure Optimization) can design VNT candidates that have lower probability of congestion right after the failure of multiple nodes compared to HLDA, which is one of the best performing VNT design algorithms. Furthermore, we show that when these VNT candidates are used as attractors in an attractor selection algorithm, which was modeled on biological systems and proposed as a robust and self-adaptive control for future-generation networks, the average time to recover from difficult failure scenarios is less than the attractors designed by HLDA. Unlike HLDA, our VNT design algorithms and the attractor selection algorithm does not require the traffic matrix and the topology information after failure.

**Keywords:** Virtual network topology design; Attractor selection; Network failure

## 1 Introduction

The global Internet traffic is growing at a tremendous rate. Currently, the only scalable way to carry such a large amount of traffic is using optical networks. The wavelength division multiplexing (WDM) allows accommodating high amount of IP traffic on fast optical networks and can span longer distances than electrical cabling, so it is a promising solution to handle the fast-growing Internet traffic demanding more and more capacity. However, due to the difficulties of high granularity switching at ultra-high speed of optical networks, processing and switching at optical nodes have important limitations. An optical fiber may carry hundreds of wavelengths, but it is difficult to terminate each wavelength and moreover process and switch each packet carried on each wavelength at each node. However, new services and applications such as Internet of Things (IoT) and Video-on-demand (VOD) are emerging on the Internet, which cause fast changes in the traffic matrices and patterns. While optical wavelengths can carry tremendous amount of data, the connectivity limitations of optical nodes and wavelengths make it difficult to adapt the network to the fast-changing traffic.

As one of the aims of the future-generation networks is adaptability and robustness to changing and fluctuating traffic, there are many works in the literature to solve this problem. A common solution is constructing a Virtual Network Topology (VNT) in which only the physical nodes that are the transmitter and receiver edges of a lightpath are shown as connected by a link. Modifying the VNT by changing the placement of lightpaths between nodes allows adapting the network for changing traffic conditions and new application layer services. A poorly configured VNT may cause congestion on some links even when there is enough capacity.

The works in the literature for designing VNTs may be classified into two groups as online and off-line approaches [1], [2] in general. The off-line approaches create VNTs suitable for a set of possible traffic demand matrices. However, Internet traffic is difficult to predict as new applications and services, which can dramatically change the traffic, appear in time [3]. Moreover, it is difficult to predict the traffic changes due to node/link failures, cyber-attacks etc. The online approaches sample the traffic demand periodically and design a new VNT for the current environment [4]. However, they require up-to-date traffic demand matrix information, which can be challenging to retrieve. Even if it is possible to retrieve traffic matrix information, it usually takes time to correctly estimate it, which may take too long to solve a congestion in a short time. For example, [4] assumed that traffic demand is changing gradually with a period of more than several hours, so it may not be good for solving congestion due to traffic spikes. Moreover, some of them cannot handle traffic changes due to node or link failures and some of them need to know detailed information like exact place of failures in the topology in order to work.

The future-generation networks are expected to be robust to network failures and disasters. Likewise, living organisms are well-known to adapt to the changes in the environment such as disasters. Thanks to the powerful adaptation mechanisms stored in genes of living organisms, life continues on earth in spite of big and unexpected changes in the environment. Therefore, several methods based on modeling biological systems have been developed. It is shown that an attractor selection mechanism is adopted by biological systems to adapt to the environment and recover in order to increase the probability of survival. Therefore, an attractor selection control mechanism based on modeling the biological

systems was proposed for future-generation optical networks in order to recover from topology failures and find a VNT that minimizes the maximum lightpath load in the network [5]. Unlike most on-line methods in the literature, attractor selection does not require a priori knowledge like the place of failed nodes/links or detailed information about the current environment like the up-to-date traffic matrix information. Attractor selection requires only the maximum lightpath utilization level, which can be retrieved quickly by Simple Network Management Protocol (SNMP) [6]. Using the maximum load level in the network as a simple feedback, the attractor selection algorithm also recovers the network from high congestion after multiple node/link failures. Many papers on VNT failures in the literature concentrate only on preventing the disconnection of the remaining nodes in the VNT after a failure. Moreover, most off-line analytical approaches in the literature propose protection against failure of only one or two random nodes/links at a time or a regional failure. On the other hand, attractor selection can solve complex problems with randomly distributed multiple node/link failures, which may occur due to a large scale distributed denial of service (DDoS) cyber-attack.

In attractor selection, the system tries to find an equilibrium point by evolving around the attractors where the conditions are known or expected to be preferable. The attractor selection algorithm uses a list of VNTs as attractors. Ref. [7] showed that even when random VNTs are used attractors, attractor selection has better performance than I-MLTDA algorithm [8] in general. However, when the attractor VNTs were not suitable for the network, it took a long time to find a solution in some cases. The first work on designing VNT candidates as attractors was in [9], which showed that its attractors further decrease the convergence time compared to attractors selected in a random manner. While the algorithm is good for designing VNTs with low utilization for a wide range of traffic matrices, it does not take the physical topology into account. Therefore, lightpaths may end up passing through many hops in the physical topology. Long route length increases the probability that a lightpath passes through a failed node or link and tears down in case of a network failure. Moreover, the algorithm does not provide backup paths against big changes in the utilization distribution in the network due to failures.

In this paper, which is an extended version of [10], we propose an algorithm called MFLDA (Minimum Flow Logical topology Design Algorithm) for designing VNT candidates that can accommodate a wider range of traffic patterns compared to a random VNT. Moreover, we present an extended version called MFLDA-FO (MFLDA with Failure Optimization), which is more robust against traffic changes after failure of multiple nodes. We show that right after a network failure a VNT designed by MFLDA-FO has a lower congestion probability compared to a VNT designed HLDA (Heuristic Logical topology Design Algorithm) [2], which is one of the best performing VNT design algorithms in the literature. Furthermore, we show that when the VNT candidates designed by MFLDA-FO are used as attractors in an attractor selection algorithm, the average time to recover from difficult failure scenarios is less than using the attractors designed by HLDA. Unlike HLDA, our VNT design algorithms and the attractor selection algorithm does not require the traffic matrix and the topology information after the failure. In [10], the simulations were carried out on Waxman [11] topology only. In this paper, we present new simulation results using Erdős–Rényi (ER) [12] topology to show the performance of the proposed algorithm. Moreover, we describe the proposed algorithm in more detail with more discussions.

The paper is organized as follows. In Section 2, we present the algorithm for designing VNT candidates. In Section 3, we present the architecture of attractor selection. Section 4 shows the simulation results and discusses the performance of the architecture. Section 5 concludes the paper.

# 2 Problem Formulation

An optical network architecture consists of optical routers connected with fiber links. The nodes are connected with dedicated virtual circuits called lightpaths. Optical routers can switch the lightpaths by optical cross-connects (OXC) and receive/transmit lightpaths through receivers and transmitters. As the number of receivers and transmitters in a node is limited, it is not possible establish lightpaths on each wavelength between adjacent nodes. If a lightpath is not terminated at a node, the lightpath can only pass-through it towards the output fiber selected by the OXC. These pass-through nodes can be neglected in the routing table of the IP network. Therefore, a virtual network topology can be drawn by considering the lightpaths as direct links between their receiver and transmitter nodes. Ref. [9] showed that it is possible to design VNT candidates that give low maximum link utilization for a wide range of traffic matrices by optimizing the placement of lightpaths between nodes heuristically. However, the VNTs designed by [9] were not robust against network failures as it did not take the physical topology into account. In this paper, our aim is to design VNTs robust against both wide range of traffic matrices and network failures, so it is more challenging.

Let's denote the traffic from a source to destination node as a flow. The flows are carried over the lightpaths established on the physical topology. The probability of a congestion on a lightpath increases with the increasing number of flows passing through. In order to minimizes the number of flows on the lightpaths, we propose an algorithm called MFLDA (Minimum Flow Logical topology Design Algorithm) for designing VNT candidates that can accommodate a wide range of traffic patterns. Moreover, we extend it to MFLDA-FO (MFLDA with Failure Optimization) variant, which minimizes the number of flows on the lightpaths after node failures.

```
1:  function INITIALIZE
2:    n ← number of nodes
3:    stop_all ← 0
4:    for k ← 0 to n do
5:      tra[k] ← number of transmitters on k
6:      rec[k] ← number of receivers on k
7:      token_tra[k] ← number of transmitters on k
8:      token_rec[k] ← number of receivers on k
9:    end for
10:   for each fiber from src to dst on physical topology
do
11:     if tra[src] > 0 AND rec[dst] > 0 then
12:       establish a lightpath from src to dst
13:       tra[src]--
14:       rec[dst]--
15:       token_tra[src]--
16:       token_rec[dst]--
17:     end if
18:   end for
19:   token ← MIN(MAX(token_tra), MAX(token_rec))
20:   for k ← 0 to n do
21:     token_tra[k] ← token_tra[k] - token + 1
22:     token_rec[k] ← token_rec[k] - token + 1
23:   end for
24:   return n, tra, rec, token_tra, token_rec, stop_all
and initial VNT
25: end function
```

Fig. 1. The initialization of parameters and setting the initial VNT

The pseudocode code of the main algorithm is shown in Fig. 3. First, the parameters are initialized and the initial VNT is established as shown in Fig. 1. In Fig. 1, *n* denotes the number of nodes in the network. The number of transmitters and receivers available on each node are stored in *tra* and *rec* arrays. When choosing the node pairs for new lightpaths, we give priority to the nodes that have highest number of available transmitters/receivers, so we apply a token based priority scheme. The transmitter and receiver tokens on each node are stored in *token_tra* and *token_rec* arrays and first initialized to the number of transmitters and receivers available. In the FOR loop on line 10, initially the VNT is set to the physical topology by establishing lightpaths on the fibers between adjacent nodes. The reason is that as the number of physical hops that a lightpath traverses increases, the probability of being hit by a failure increases, so priority is given to establish single hop lightpaths. In case the nodal degree is higher than the number of transmitters/receivers in a node, priority is given to the links that make the topology connected. This initial VNT serves as a substrate for adding new lightpaths. If network failures are not considered, it is also possible to use a simple random ring topology passing through all nodes as an initial VNT. The important point is that the initial VNT should be fully connected, so all nodes are reachable. On line 19 and in the next FOR loop, the number of tokens are normalized. As an example, assume that there are four nodes in a network and after initial VNT is created on line 10, they have 3, 4, 1 and 4 transmitters and receivers left, respectively. After normalization, their token count becomes 0, 1, -3 and 1. Only the nodes with tokens more than zero are allowed to establish a lightpath, so the second and the fourth nodes are candidates.

```
1: function COLLECTDATA
2:   for f ← ID of nodes that may fail and finally n do
3:     if f < n then
4:       simulate failure of node f
5:     end if
6:     routing[f] ← new routing table
7:     for each node pair src and dst do
8:       hop[f][src][dst] ← number of hops from src to dst
9:     end for
10:    for each lightpath from src to dst do
11:      pass[f][src][dst] ← number of s-d pairs on
lightpath from src to dst
12:    end for
13:    for each node pair src and dst without a direct
lightpath do
14:      decrease[f][src][dst] ← number of node pairs
whose hop count will decrease if a lightpath is
established from src to dst
15:    end for
16:    if f < n then
17:      node f recovers
18:    end if
19:  end for
20:  return routing, hop, pass, decrease
21: end function
```

Fig. 2. Collecting data on the current logical topology

In order to select and add new lightpaths, the algorithm starts the main loop on line 2 in Fig. 3. Each time a new lightpath is established, the algorithm returns to here. In order to select the s-d pairs for establishing lightpaths, the algorithm collects data on the current logical topology as shown in Fig. 2. In order to analyze the effect of node failures, the algorithm simulates single node failures in the FOR loop on line 2 in Fig. 2 and stores the data about the lightpath stats after the failure. The *f* variable is set to ID of the nodes that may fail. In the last loop, the stats are calculated for the VNT without any failure by setting *f* to *n*, which prevents node failures. As MFLDA creates a VNT without considering any failures, *f* is set to only *n* in MFLDA. The algorithm is called MFLDA-FO, when *f* includes the set of nodes that may fail. This allows the created VNT to be more robust against the failures at these nodes. If *f* loops over IDs of all nodes, the algorithm creates a VNT that is robust against all possible node failures.

As a first stat, the routing algorithm is run to determine the paths on line 6 in Fig. 2. When there are multiple possible paths, the selection varies with the implementation of the algorithm, so the exact behavior of the routing algorithm must be known. Using the routing information, the hop count distribution among s-d (source-destination) pairs is calculated on line 7. Then the number of total number of s-d pairs on each lightpath is

```
1: INITIALIZE()
2: repeat
3:    COLLECTDATA()
4:    stop_pass ← 0
5:    sort pass[f][src][dst] in descending order with
corresponding (f, src, dst)
6:    repeat
7:       repeat
8:          get next (f, src, dst) in the sorted pass list
9:          set the routing table to routing[f]
10:          for each node pair src2 and dst2 whose route
includes the lightpath from src to dst do
11:             impact[src2][dst2] ← decrease[f][src2][dst2]
* (hop[f][src2][dst2] - 1)
12:          end for
13:          sort impact[src2][dst2] in descending order with
corresponding (src2, dst2)
14:          repeat
15:             get next (src2, dst2) in the sorted impact list
16:             if token_tra[src2] > 0 AND token_rec[dst2] >
0 AND tra[src2] > 0 AND rec[dst2] > 0 then
17:                establish a lightpath from src2 to dst2
18:                token_tra[src2]--
19:                token_rec[dst2]--
20:                tra[src2]--
21:                rec[dst2]--
22:                stop_pass ← 2
23:             end if
24:          until stop_pass is 2 OR impact list is empty
25:          if stop_pass is 0 AND pass list is empty then
26:             if MIN(token_tra) ≤ 0 OR MIN(token_rec) ≤
0 then
27:                token_tra++
28:                token_rec++
29:                stop_pass ← 1
30:             else
31:                stop_all ← 1
32:             end if
33:          end if
34:       until stop_pass > 0
35:       re-initialize pass to last sorted list
36:    until stop_all is 1 OR stop_pass is 2
37: until stop_all is 1
```

Fig. 3. The main algorithm

calculated on line 10 and stored in the array *pass*. On line 13, we find the number of node pairs whose hop count will decrease if a lightpath is established between a s-d node pair and store it as a metric for this s-d pair in an array denoted by *decrease*. While it is easy to estimate these stats in shortest path routing, it may be difficult with some routing algorithms. Their estimation in different routing algorithms is left as a future work.

After collecting the stats, the main algorithm starts selecting the s-d pairs for establishing lightpaths in Fig. 3. Among the possible candidates, the algorithm gives priority to the ones that will decrease the number of s-d pairs on the lightpaths that are carrying the highest number of s-d pairs. Therefore, the values in the *pass* array is sorted in descending order on line 5. By the loop on line 7, the algorithm loops over *pass* list with corresponding *(f,src,dst)* in order to decrease the number of s-d pairs on the lightpath from *src* to *dst* with the failure scenario *f*, until a new lightpath is established or *pass* is empty. In order to decrease the s-d pair count on the selected lightpath, the algorithm tries to establish a direct lightpath between one of the s-d pairs on this lightpath. Therefore, the algorithm creates a list of s-d pairs passing through this lightpath and stores them in the array *impact* with an impact factor metric, which is the amount of decrease in total hop count between all node pairs after a lightpath is established between this s-d pair. After sorting the impact factor list in descending order, the algorithm tries to establish a lightpath among s-d pairs in the list in a loop starting on line 14. Before establishing the lightpath, the algorithm checks whether the lightpath satisfies the conditions like the node pair has enough number of tokens, transmitter and receivers. It may also need to satisfy other conditions like the maximum number of wavelengths on the fibers depending on the architecture.

While not mandatory, checking whether the congestion probability decreases after establishing the lightpath can further decrease the congestion probability. Even though each new lightpath decreases the average hop count in the network, in some cases the new lightpath may end up carrying many s-d pairs and become a bottleneck further increasing the congestion probability. If the distribution of the traffic matrix is known, the non-congestion probability for a given of s-d pairs on a lightpath can be estimated by a simulation. As a comparison metric, the overall non-congestion probability of a VNT can be roughly approximated by multiplying the non-congestion probability of all lightpaths and the lightpath is established only if it decreases. While the metric usually has a high deviation from the real non-congestion probability of the VNT due to the high correlation among adjacent lightpaths, our simulations revealed that it is useful for comparison and provides some small improvement in the congestion probability of the designed VNT.

If a new lightpath is established, the algorithm stops trying establishing new lightpaths and goes back to the beginning of the main loop on the line 2. When *pass* is empty and no new lightpaths are established, the algorithm checks the tokens of all nodes. If there is a node with token less than one, the algorithm increases the tokens of all nodes by one so that more nodes can establish a lightpath and re-runs the loop after re-initializing the *pass* array to the last sorted list. Otherwise, the algorithm stops and outputs the designed VNT. As many *(f,src,dst)* give the same value in *pass* and *impact* arrays, it is possible create different VNTs by random shuffling the order of *(f,src,dst)* sets before sorting the *pass* and *impact* arrays.

Due to the token based architecture, it is difficult to state an order of complexity to the algorithm. As a reference, it takes around 30 minutes to design a VNT on a Waxman topology with 100 nodes and 400 optical fibers with 16 transmitters and receivers per node, using a not-so-optimized single-threaded C++ simulator on a single core of Intel 3960x CPU. Speed improvement of

several orders of magnitude seems possible using an optimized and multi-threaded program. For example, *CollectData* function of MFLDA-FO in Fig. 2 can be run *O(n)* faster than a single-threaded program by processing each iteration of the FOR loop on line 2 using a different CPU core by parallel programming.

# 3 Attractor Selection Control

In biological systems, the interaction between metabolic reaction network and gene regulatory network controls the growth of cells. The gene regulatory network produces the proteins necessary for the cell growth. Each gene has an expression level, which shows the level of protein production. In the metabolic reaction network, proteins use the nutrition in the environment and produce the substances necessary for the growth. The concentration of substances necessary for growth is used as a feedback by the gene regulatory network about the current environment. High level of concentration means that the conditions are preferable, which implies an attractor state, so the deterministic behavior drives the biological system to continue using the current state. On the other hand, if the concentration of substances necessary for growth is low, the growth level decreases, which causes the system to be driven by stochastic behavior. In order to find a new attractor state for high growth, the system starts to change the expression levels randomly. The system returns back to deterministic behavior only after a new set of expression levels that is giving high growth rate is found. In other words, the system is driven by two behaviors, i.e., deterministic and stochastic. When the system conditions are preferable, the deterministic behavior drives the system to converge to an attractor. If the conditions are not preferable, the stochastic behavior dominates the system. The system searches for a new attractor by randomly changing the state by adding noise. When the system finds a new operating point with preferable conditions, deterministic behavior takes over the control again. The system is controlled by the current state as a feedback.

### 3.1 Analytical Model in Biological Systems

The expression level of *n* genes, which shows the level of protein production, is represented as $x = (x_1, x_2, \ldots, x_n)$. The protein production level is calculated by

$$\frac{dx}{dt} = \alpha \cdot g(x) + \eta. \qquad (1)$$

In the equation, $\alpha$ is the growth rate showing the concentration of required substances. *g(x)* gives the deterministic behavior. The $\eta$ is the Gaussian noise term showing the strength of stochastic behavior. If the concentration of required substances is low, $\alpha$ decreases, so $\eta$ dominates the equation, which causes the gene regulatory network to do a random walk by fluctuations. Ref. [13] gives a detailed description of metabolic reaction network.

### 3.2 VNT Control

As the traffic conditions and the physical topology may change in time, a VNT reconfiguration control is necessary to adapt to the changes. A single VNT may not be able to give low utilization after each change in traffic pattern or each change in topology after a failure. In a biological system, the gene regulatory network adapts to the changing environment by taking the growth rate as a feedback as a result of the metabolic reaction network. In our work, we tried to adapt to the changing IP network conditions by reconfiguring the VNT, so we interpret the VNT as the gene regulatory network and the IP network as the metabolic reaction network. The growth rate in biological system shows the how much preferable the conditions are. Our aim is to minimize the congestion on the highest utilized lightpath in order to decrease the packet drop rates and buffering delays. As high packet drop rates and buffering delays mean bad conditions, we chose the maximum lightpath utilization as a growth rate metric. When node/link failures occur, the capacity of available lightpaths may no longer enough to carry the traffic and cause high congestion in the IP network. The VNT control method, takes it as a feedback and reconfigures the VNT until the maximum utilization in the IP network decreases below a threshold value. The outline of the algorithm is as follows:

1. Measure and receive the maximum lightpath utilization by SNMP
2. Convert the lightpath utilization information to the growth rate feedback. Using the growth rate feedback, calculate the new expression level of genes.
3. Establish or tear lightpaths in order to reconfigure VNT according to the new expression level vector.
4. Use the newly established VNT until next iteration.

### 3.3 Analytical Model of VNT Control

Each lightpath is controlled by a gene, so *{i}*-th lightpath has an expression level of $x_i$, which is calculated by

$$\frac{dx_i}{dt} = \alpha \cdot \left( f\left( \sum_j W_{ij} \cdot x_j - \theta \right) - x_i \right) + \eta. \qquad (2)$$

The α is the growth rate, which is calculated according the maximum utilization level in the IP network as a feedback. The η is the Gaussian noise term showing the strength of stochastic behavior. If the maximum utilization level is high, $\alpha$ decreases, so $\eta$ dominates the equation, which causes the VNT to change randomly to find a new attractor. The rate of change in the expression level by the deterministic behavior is given by the sigmoidal regulation function

$$f(z) = \tanh(\mu z), \qquad (3)$$

where $\mu$ is the gain parameter. *W* is the regulatory matrix, which shapes the system to convergence to an attractor state when the growth rate is high. $\theta$ is the threshold value for expression level.

We use the SNMP to retrieve the maximum utilization level $u_{max}$ in the IP network to calculate $\alpha$. It may be possible to use other metrics, but utilization is commonly used in many papers as a metric in the literature [2], [4]. $u_{max}$ is converted to $\alpha$ by

$$\alpha = \frac{1}{1 + exp(\delta \cdot (u_{max} - \zeta))}, \quad (4)$$

where $\delta$ is the gradient and the $\zeta$ is the threshold utilization parameter. When $u_{max}$ equals to the threshold, $\alpha$ is 0.5. When the $u_{max}$ surpasses $\zeta$, the value of $\alpha$ converges to zero, which means low growth rate. In this case, the noise term $\eta$ dominates the control and the VNT changes randomly, until the VNT control finds a VNT with low $u_{max}$. After each iteration, the $x_i$ values are sorted from highest to lowest and the corresponding lightpaths are established in this order.

The regulatory matrix $W$ is a Hopfield neural network containing a set of possible attractors [14], [15], [16]. We use it as an associative memory to store the attractors by using orthogonal projection. Let's assume that we have $m$ attractors, where attractor $k$ has the VNT expression vector $x(k) = (x_1^{(k)}, \ldots, x_i^{(k)})$. The VNT lightpaths are coded according bipolar coding by setting $x_i$ to 1 if the lightpath is established and -1 otherwise. Bipolar coding is used because bipolar vectors have a greater probability of being orthogonal than binary vectors [16]. Let $X$ be a matrix whose rows are the attractors. First we calculate the pseudo-inverse matrix $X^+$. Then the regulatory matrix is calculated by simply

$$W = X^+ X. \quad (5)$$

In order to store the attractors, it is also possible to use Hebbian learning, which has lower computation complexity than orthogonal projection. However, we used orthogonal projection because it has a higher memory capacity than Hebbian learning [17], [18].

The convergence time to a solution depends on the performance of the attractors used. In the previous works on attractor selection, random VNTs were used as attractors. In this paper, we propose using the VNT candidates designed by MFLDA and MFLDA-FO as initial attractors in the regulatory matrix.

# 4 Performance Evaluation

We evaluated the performance of the proposed algorithm against network failures by a simulation study. In [10], only the simulation results on Waxman model [11] topology were shown. In this paper we also show the simulation results on Erdős–Rényi (ER) model [12] topology. Two physical topologies with Waxman and ER models were designed by BRITE tool [19] with default parameters. Both Waxman and ER topologies had 100 nodes and 400 optical fibers, one optical fiber for each direction. The number of transmitters and receivers in a node was limited to 16. As the transmitter/receiver count was the main limit, there were enough number of wavelengths on a fiber to carry the lightpaths. The amount of traffic per s-d node pair had a LogNormal(-0.5,1) distribution. In order to show the effect of traffic intensity, the traffic matrix was multiplied by $k$. A VNT was marked as congested if the utilization of one of its lightpaths was more than 50%. The attractor selection algorithm parameters were $\mu = 10$, $\delta = 50$, and $\zeta = 0.5$. The variance $n$ of the noise $\eta$ was 0.15. Shortest path routing is applied on both physical and logical topologies.

We compared four different VNT candidate sets. The VNT candidates denoted by MFLDA were designed by the proposed algorithm without optimization for node failure, by setting $f$ to only $n$ on line 2 of the pseudocode, which prevents an optimization for failures. The VNT candidates denoted by MFLDA-FO were designed by the proposed algorithm with optimization for possible node failures by looping $f$ from 0 to $n$. For comparison, the VNT candidates denoted by RLDA (Random Logical Topology Design Algorithm) and HLDA (Heuristic Logical topology Design Algorithm) [2] were also simulated. In RLDA, the VNTs were created by establishing lightpaths among randomly chosen node pairs. In HLDA, the VNTs were created by establishing lightpaths among the s-d pairs with the highest traffic according to the traffic matrix. In order to maximize the performance of HLDA in failure scenarios, HLDA is applied to the topology after failure with the knowledge of the failed nodes. Therefore, the VNTs created by HLDA were specially designed for the topology after failure. On the other hand, we simulate MFLDA and MFLDA-FO under harsher conditions without providing the traffic matrix information and the place of failed nodes. As the exact place of failed nodes are not known to the network, when the transmitters/receivers of failed lightpaths become idle, these idle transmitters/receivers were used for establishing new lightpaths among randomly chosen node pairs.

Each VNT candidate set was simulated with 500.000 traffic matrices and failure patterns to estimate their congestion probability. A set of 10 VNTs were designed by MFLDA-FO and MFLDA for both ER and Waxman topologies. In this paper, the random number generator seed, which is used for generating VNTs, was different from [10], so a different set of VNTs was used in all Waxman topology simulations than in [10]. In each simulation, one of the 10 designed VNT candidates was randomly selected and simulated. In order to increase the randomness, a different VNT was used by RLDA in each simulation. As HLDA optimizes the VNT for a given traffic matrix and failed node set, again a different VNT was designed and used by HLDA in each simulation. When a node fails, the lightpaths passing through its fibers fail at the same time. Instead of rerouting the failed lightpaths, the traffic on the failed lightpaths is rerouted to other available lightpaths like in [7].

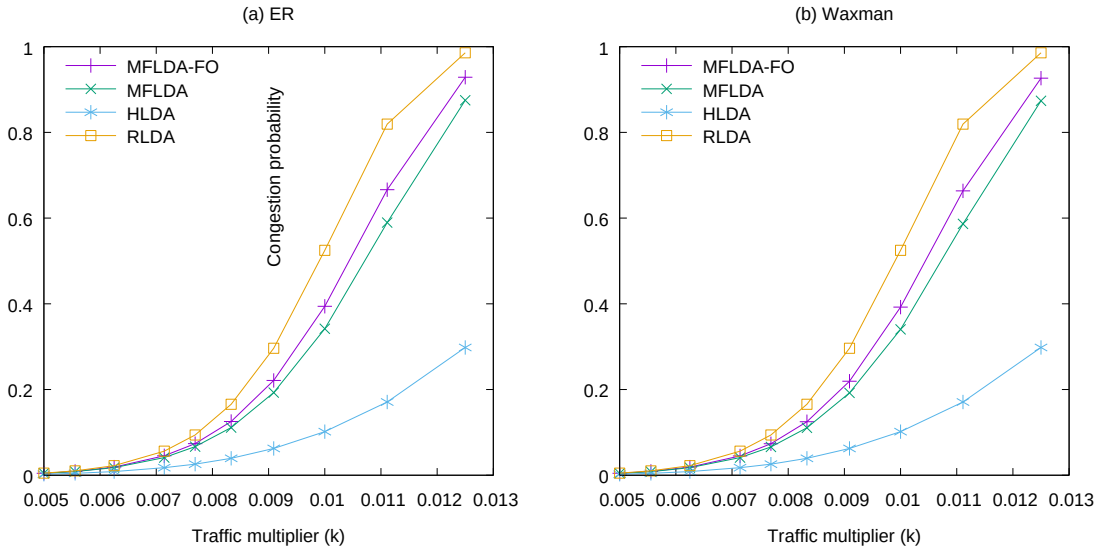


Fig. 4. The congestion probability when there is no failure

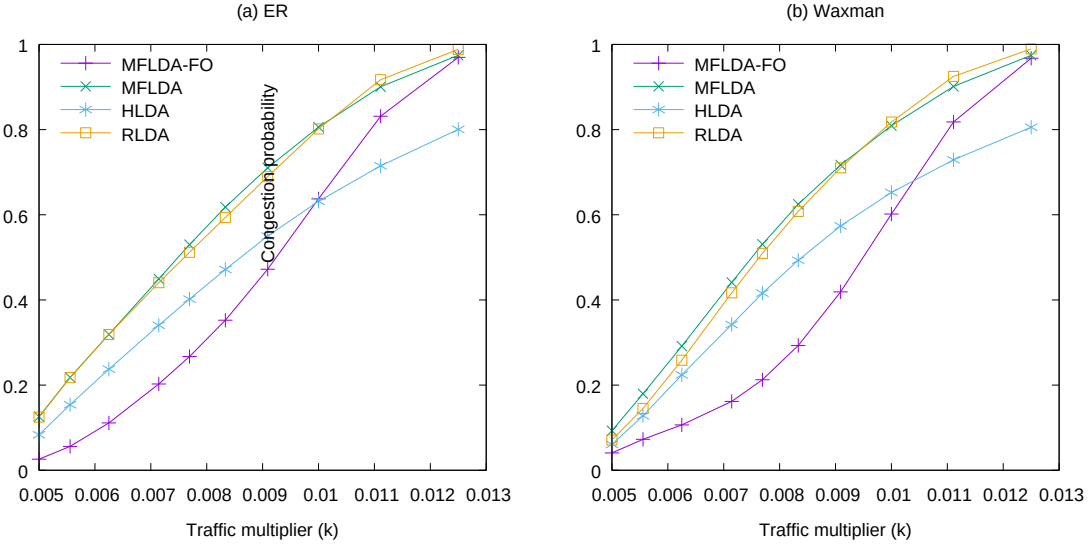


Fig. 5. The congestion probability after 5 nodes fail

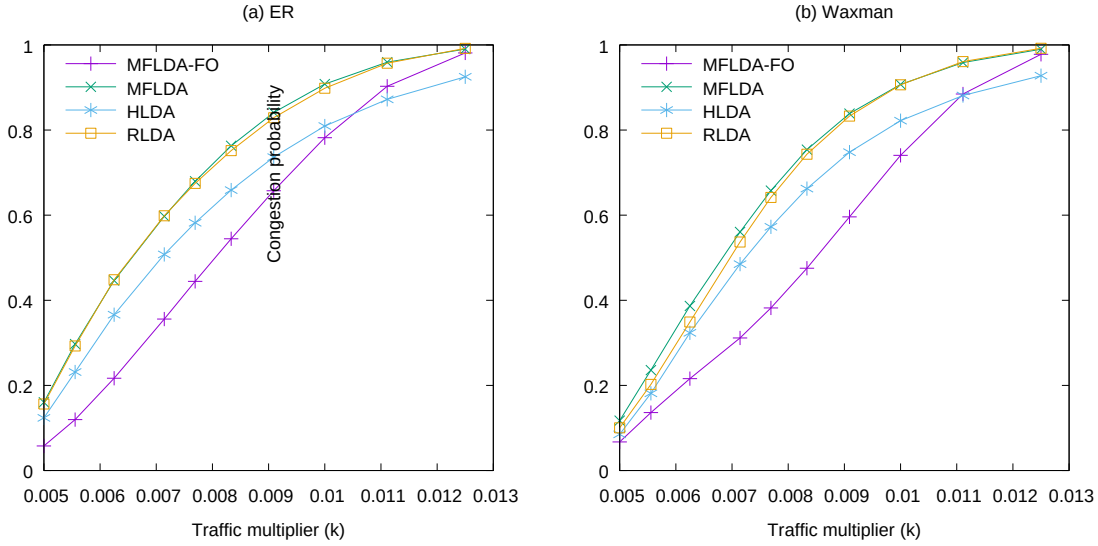


Fig. 6. The congestion probability after 10 nodes fail

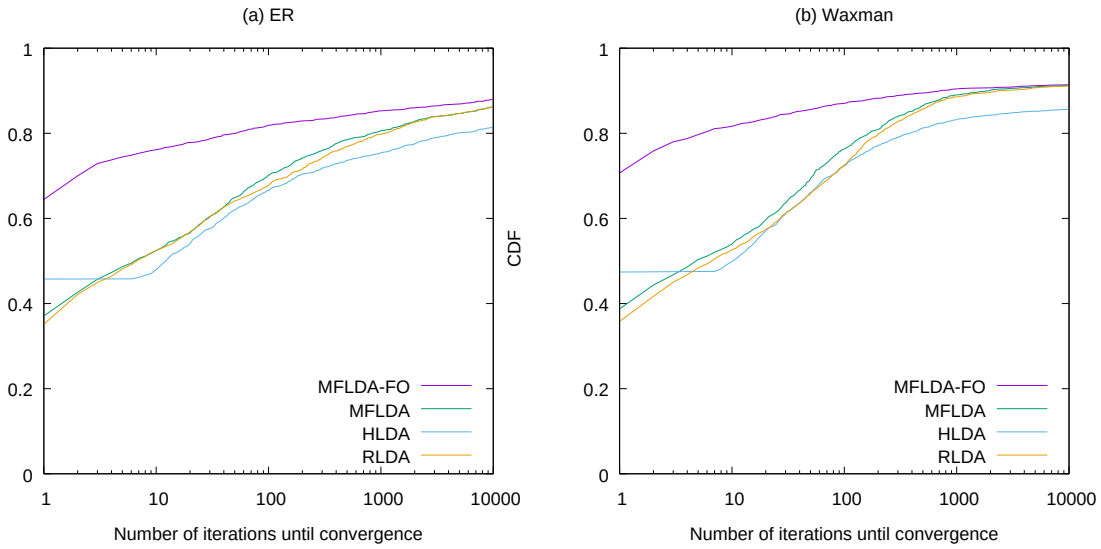


Fig. 7. The CDF of convergence time to a solution after 5 nodes fail

Fig. 4 shows the congestion probability for ER topology in Fig. 4(a) and Waxman topology in Fig. 4(b) when there was no node failure. The x-axis is the traffic multiplier and the y-axis is the congestion probability. As HLDA and RLDA are independent of the physical topology, they gave the same result in both graphs. While MFLDA-FO and MFLDA design VNTs specific to the physical topology, they gave very close results in ER and Waxman topologies in Fig. 4.

Compared to other algorithms, HLDA gave the lowest probability of congestion. As HLDA has the traffic matrix information and it designs a specific VNT for each traffic matrix, this is an expected result. However, our aim is to design VNT candidates without traffic matrix information. In Fig. 4, MFLDA gave lower congestion probability than MFLDA-FO. The reason is that the failure optimization causes the VNT to include backup paths against possible failure scenarios. These lightpaths may not be so useful when there is no failure, so the congestion probability of MFLDA-FO was a bit higher than MFLDA. RLDA gave the highest congestion probability.

Fig. 5 and 6 show that when 5 and 10 nodes failed, MFLDA-FO gave lower congestion probability than the other algorithms unless the traffic was too high in both ER and Waxman topologies. When 5 nodes failed and the traffic intensity $k$ was 0.005 in ER topology, the congestion probability of MFLDA-FO was more than three times lower than HLDA. When 5 nodes failed and $k$ was 0.0077 in the Waxman topology, the blocking probability of MFLDA-FO was half of HLDA. While HLDA had both the traffic matrix and failed node list information, it could not create direct lightpaths among some of the s-d pairs with high traffic, whose lightpath route pass through failed nodes. As HLDA does not provide any optimization for these s-d pairs, they may end up using multiple lightpaths and concentrate on some lightpaths and cause congestion. On the other hand, MFLDA-FO optimizes the VNT by taking failures into account to prevent hot-spots, so it gave lower congestion probability unless the traffic was too high. As many lightpaths become unavailable and the characteristics of the topology greatly changes after multiple node failures, the optimizations by MFLDA no longer work, so the VNTs designed MFLDA gave similar congestion probability to RLDA.

While MFLDA-FO had lower probability of congestion right after failure, not all possible failure scenarios could be solved by VNT optimization only. In such cases, attractor selection mechanism allows solving complex failure scenarios after some iterations. However, the convergence time to a solution depends on the performance of the attractors used. Fig. 7 shows the cumulative distribution function (CDF) of the number of iterations by attractor selection algorithm until it finds a VNT that has maximum lightpath utilization less than 50% for ER and Waxman topologies. The VNT candidates designed by MFLDA-FO, MFLDA, HLDA and RLDA were set as the attractors of the attractor selection algorithm and the initial VNT. Each attractor set was simulated with 2000 different failure patterns and traffic matrices. The traffic intensity $k$ was set to 0.0083. Five randomly chosen nodes failed before the first iteration. The x-axis is the number of iterations until convergence. As seen in the figure, the attractors designed by our MFLDA-FO gave much faster convergence than both RLDA and HLDA algorithms, even though HLDA had both the traffic matrix and failed node list information that were not available to MFLDA-FO. Around 10% of the simulations could not converge as there was no solution or the solution domain was too small.

# 5 Conclusion

In this paper, we proposed an algorithm called MFLDA for designing VNTs that can accommodate a wider range of traffic patterns without using traffic matrix

information in order to increase the adaptability and robustness of optical networks to the changing and fluctuating traffic due to emerging applications and services in future-generation networks. We also presented an extended version called MFLDA-FO to design VNTs robust against congestion due the traffic changes after network failures. The simulation results showed that the VNT candidates designed by MFLDA-FO can accommodate a wider range of traffic both before and right after a failure of multiple nodes. As not all possible failure scenarios could be solved by applying a single optimized VNT, an attractor selection control mechanism proposed for future-generation optical networks was applied. The simulations showed that the converge time is faster when VNTs designed by MFLDA-FO are used in the attractor selection algorithm, compared to the attractors designed by HLDA algorithm. Unlike HLDA, our VNT design algorithms and the attractor selection algorithms do not require the traffic matrix information and the failed node list.

While it is easy to estimate the routing stats used in our algorithm when shortest path routing is applied, it may be difficult with some routing algorithms. As a future work, we will investigate the possible implementation issues with other routing algorithms and evaluate their performance.

## Acknowledgment

This research was supported in part by Grant-in-Aid for Scientific Research (A) 15H01682 of the Japan Society for the Promotion of Science (JSPS) in Japan.

## References

[1] B. Mukherjee, D. Banerjee, S. Ramamurthy, and B. Mukherjee, Some Principles for Designing a Wide-area WDM Optical Network, *IEEE/ACM Transactions on Networking*, Vol. 4, No. 5, pp. 684–696, October, 1996.

[2] R. Ramaswami and K. Sivarajan, Design of Logical Topologies for Wavelength-routed Optical Networks, *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 5, pp. 840–851, June, 1996.

[3] Y. Liu, H. Zhang, W. Gong, and D. Towsley, On The Interaction Between Overlay Routing and Underlay Routing, *IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, Florida, 2005, pp. 2543–2553.

[4] A. Gencata and B. Mukherjee, Virtual-topology Adaptation for WDM Mesh Networks Under Dynamic Traffic, *IEEE/ACM Transactions on Networking*, Vol. 11, No. 2, pp. 236–247, April, 2003.

[5] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, Adaptive Virtual Network Topology Control Based on Attractor Selection, *Journal of Lightwave Technology*, Vol. 28, No. 11, pp. 1720–1731, June, 2010.

[6] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, and D. Sadok, A Survey on Internet Traffic Identification, *IEEE Communications Surveys Tutorials*, Vol. 11, No. 3, pp. 37–52, July, 2009.

[7] Y. Koizumi, S. Arakawa, S. Kamamura, D. Shimazaki, T. Miyamura, A. Hiramatsu, and M. Murata, Adaptability of Virtual Network Topology Control Based on Attractor Selection Against Multiple Node Failures, *18th OptoElectronics and Communications Conference / Photonics in Switching*, Kyoto, Japan, 2013.

[8] D. Banerjee and B. Mukherjee, Wavelength-routed Optical Networks: Linear Formulation, Resource Budgeting Tradeoffs, and a Reconfiguration Study, *IEEE/ACM Transactions on Networking*, Vol. 8, pp. 598–607, 1997.

[9] T. Ohba, S. Arakawa, Y. Koizumi, and M. Murata, Scalable Design Method of Attractors in Noise-induced Virtual Network Topology Control, *Journal of Optical Communications and Networking*, Vol. 7, No. 9, pp. 851–863, September, 2015.

[10] O. Alparslan, S. Arakawa, and M. Murata, Designing VNT Candidates Robust Against Congestion due to Node Failures, *IEEE High Performance Switching and Routing*, Tokyo, Japan, 2016.

[11] B. M. Waxman, Routing of Multipoint Connections, *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, pp. 1617–1622, December, 1988.

[12] P. Erdos and A. Renyi, On The Evolution of Random Graphs, *Publications of The Mathematical Institute of The Hungarian Academy of Sciences*, pp. 17–61, 1960.

[13] A. Kashiwagi, I. Urabe, K. Kaneko, and T. Yomo, Adaptive Response of a Gene Network to Environmental Changes by Fitness-induced Attractor Selection, *PLoS ONE*, Vol. 1, No. 1, p. e49, December, 2006.

[14] J. J. Hopfield, Neurons with Graded Response Have Collective Computational Properties Like Those of Two-state Neurons, *National Academy of Sciences*, Vol. 81, No. 10, pp. 3088–3092, 1984.

[15] Y. Baram, *Orthogonal Patterns in Binary Neural Networks*, Technical Memorandum 100060, NASA, 1988.

[16] R. Rojas, *Neural Networks: A Systematic Introduction*, New York, Springer-Verlag, 1996.

[17] J. J. Hopfield, Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *National Academy of Sciences of the United States of America*, Vol. 79, No. 8, pp. 2554– 2558, April, 1982.

[18] Y. S. Hanay, Y. Koizumi, S. Arakawa, and M. Murata, Virtual Network Topology Control with Oja and Apex Learning, *24th International Teletraffic Congress*, Krakow, Poland ,2012, p. 47.

[19] A. Medina, A. Lakhina, I. Matta, and J. Byers, Brite: An Approach to Universal Topology Generation, *Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Washington, DC, USA, 2001, pp. 346–353.