Monitoring Cycle Design for Fast Link Failure Detection in All-Optical Networks

Bin Wu and Kwan L. Yeung
Dept. of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam, Hong Kong
E-mail: {binwu, kyeung}@eee.hku.hk

Abstract—**Fast link failure detection in all-optical networks (AONs) can be achieved using monitoring cycles (m-cycles). An m-cycle is a loop-back optical connection of supervisory wavelengths with a dedicated monitor. Compared to the channel-based or link-based monitoring schemes, m-cycle based schemes require much less number of monitors. In this paper, we propose an ILP (Integer Linear Program) formulation for m-cycle design to minimize the network cost. Our contributions are two-fold: 1) non-simple m-cycles are enabled; and 2) an efficient tradeoff is allowed between the monitor cost and the bandwidth cost. Numerical results show that our algorithm outperforms existing algorithms with a significant performance gain.**

Keywords-All-optical network (AON); ILP (Integer Linear Program); link failure detection; monitoring cycle (m-cycle).

I. Introduction

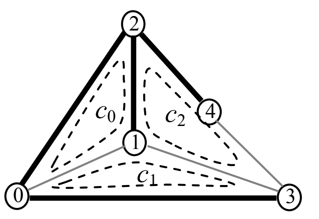
As the communication infrastructure evolves towards all-optical networks (AONs), optical network topology also evolves from ring to mesh. With WDM technology, a single fiber can carry hundreds of wavelengths at 10 Gb/s or higher data rate. To minimize data loss upon an accidental failure such as a fiber-cut, fast fault detection is a must. Because of the lack of electrical terminations, fault detection schemes in traditional optical networks cannot be transplanted to AONs.

Fault detection can be implemented at different protocol layers. In general, implementations using upper layer protocols [1] require a much longer detection time than the typical 50 ms requirement for optical recovery. To achieve fast fault detection, optical/physical layer schemes are preferred [2-3]. At the optical layer, a fault can be detected using a special optical device called *monitor* [4]. A channel-based monitoring scheme requires one monitor for each wavelength channel on each link, thereby requiring a very large number of monitors. A link-based monitoring scheme is more scalable, but still requires one monitor per link.

To further reduce the number of required monitors, monitoring-cycle (m-cycle) [2] is introduced. It is a loop-back optical connection with a dedicated monitor. It is implemented using a supervisory wavelength on each link it covers/passes through. Let the *length* of an m-cycle be the total number of links it covers (or the sum of distance-related costs of the links). If a set of m-cycles $\{c_0, c_1, \ldots, c_{M-1}\}$ covers every link in the network, then the set is called a *cycle cover*, and the *cover length* is the total length of all the M m-cycles. In this case, the total number of monitors required is M, and the total amount of bandwidth dedicated to the cycle cover is its cover length.

Assume there is at most a single link failure at a time. If a link fails, optical signals in the m-cycles covering this link will be disrupted, and the corresponding monitors will alarm. This generates an *alarm code* with the format of $[a_{M-1}, \ldots, a_1, a_0]$, where a_i=1 means that the monitor on m-cycle c_i alarms and a_i=0 otherwise. Fig. 1 shows a cycle cover consisting of three (dashed) m-cycles $\{c_0, c_1, c_2\}$. If link (0, 1) fails, the monitors on c_0 and c_1 will alarm to generate the alarm code [0, 1, 1]. Similarly, if link (0, 2) fails, the monitor on c_0 will alarm and the resulting alarm code is [0, 0, 1]. By decoding the alarm code, we can identify the location of the failure. Ideally, we should have a unique alarm code for every link in order to accurately localize each link failure. In reality, a cycle-based monitoring scheme *cannot* distinguish individual link failures on the same segment, where a *segment* is a path with at least two links and a degree of two at any intermediate node (such as 2−4−3 in Fig. 1a). This is because all the links on the same segment must be covered by the same set of m-cycles. If we need to distinguish the link failures on the same segment, we can add extra link-based monitors [2-3]. For example, in Fig. 1a we can add an extra link-based monitor to either (2, 4) or (3, 4) to distinguish the two link failures. This increases the total number of monitors from 3 to 4, but it is still less than 7, as required by a pure link-based scheme. In general, if two links/segments form a cut of a network, e.g. link (1, 6) and segment 2−3−4 in Fig. 2, then the corresponding link failures also cannot be unambiguously identified by a cycle-based monitoring scheme. For simplicity, we treat the links in such a cut as if they were on the same segment. In this paper, we only focus on cycle-based monitoring schemes.

To measure the accuracy of link failure localization, *localization degree* $D_L=\|\boldsymbol{E}\|/A$ is defined, where $\|\boldsymbol{E}\|$ is the total number of links in the network, and A is the size of the alarm code set. For cycle-based monitoring schemes, we have $A\leq\|\boldsymbol{E}\|$ and thus $D_L\geq 1$. This is because each link failure can trigger only one alarm code, but failures at different links (e.g. those on the same segment) may trigger the same alarm code. D_L=1 means that we can accurately localize every link failure in a



(a) m-cycles

Link	c_2	c_1	c_0	Decimal
(0,1)	0	1	1	3
(0,2)	0	0	1	1
(0,3)	0	1	0	2
(1,2)	1	0	1	5
(1,3)	1	1	0	6
(2,4)	1	0	0	4
(3,4)	1	0	0	4

(b) Alarm code table

Fig. 1. A simple example of m-cycles for link failure detection.

This work is supported by Hong Kong Research Grant Council Earmarked Grant HKU 7150/04E.

network. In this paper, we focus on constructing a set of m-cycles that not only yields the minimum localization degree, but also consumes the least amount of network resources. The amount of network resources consumed is defined as *network cost*. It consists of both monitor cost (measured by the total number of monitors required) and bandwidth cost (measured by the cover length).

Several algorithms have been proposed to construct a cycle cover with similar goals. HST [2] is based on a spanning tree. Fig. 1a shows the tree by the broad-brush links which are called *trunks*. Those links not in the tree are called *chords*. In HST, each m-cycle is generated from a chord where all other links on this m-cycle must be trunks. For example, the m-cycle generated from chord (1, 3) is 1−3−0−2−1. On the other hand, M^2-CYCLE [3] always prefers m-cycles with minimum cycle length. For example, in Fig. 1a, the minimum-length m-cycle covering link (0, 3) is 0−1−3−0 instead of 1−3−0−2−1. In fact, Fig. 1 shows an M^2-CYCLE solution. It is proved [3] that M^2-CYCLE always outperforms HST. For example, M^2-CYCLE requires a cover length of 10 in Fig. 1 but HST requires 11.

Existing algorithms [2-3] have several drawbacks. First, they only consider simple cycles. A simple cycle traverses a node at most once, whereas a non-simple cycle [5] can traverse a node multiple times. Fig. 3 shows a non-simple m-cycle. The dotted arrows indicate a possible connection of the supervisory wavelengths. The links on the cycle are defined as *on-cycle links*. If any on-cycle link fails, the associated monitor will alarm (same as using a simple m-cycle). Besides, existing algorithms [2-3] do not allow the tradeoff between the monitor cost and the bandwidth cost. In this paper, we propose an ILP formulation for m-cycle design to solve the above issues.

II. ILP FORMULATION

A. General Idea

To minimize the network cost for link failure detection, we need to consider the tradeoff between the monitor cost and the bandwidth cost. Without loss of generality, we define the cost function as a weighted sum of the two cost components, or

$$\begin{aligned} Cost &= \text{monitor cost} + \beta \times \text{bandwidth cost} \\ &= \text{number of monitors} + \beta \times \text{cover length} \qquad (1) \end{aligned}$$

where the value of β determines the relative importance between the two cost components. To formulate an ILP to minimize (1), we first need to define cycles using ILP. Following the approach in [5], we can weakly define a "cycle" by requiring each node in the network to have even number (including zero) of on-cycle links incident on it. However, such a weakly defined "cycle" may actually contain one or more disjoint cycles (i.e., cycles without any common node), where each cycle corresponds to an m-cycle. In what follows, we call the weakly defined "cycle" as a *cycle set* cs_j (where $j \in \{0, 1, \ldots, J-1\}$ is the cycle set index). Fig. 4 gives an example of cs_j which contains two disjoint m-cycles. The links covered by any m-cycle in cs_j are called on-cycle links of cs_j.

A limitation with the above cycle set definition is that we do not know the exact number of m-cycles/monitors in each cs_j. This makes it impossible to get an accurate monitor cost in (1). If we add extra constraints in ILP to ensure a unique m-cycle per cs_j, then an optimal ILP can be formulated and the total number of monitors can be obtained by just counting cs_j. But such an optimal ILP is generally too complex to be solved [5].

Instead of directly counting the total number of monitors required, we use the sum of all the decimal alarm codes as a heuristic measure for the monitor cost. A *decimal alarm code* is a decimal translation of the binary alarm code. An example is shown in Fig. 1b. Minimizing the sum of all the decimal alarm codes can suppress the required number of bits in the binary alarm codes. Since each bit corresponds to a cycle/cycle set, this in turn minimizes the total number of monitors and the monitor cost in (1). Another indirect effect is that, the total number of 1*s* in all binary alarm codes is minimized. This also minimizes the cover length. On the other hand, the bandwidth cost in (1) can be simply measured by the cover length.

The last issue is to formulate a unique alarm code for each link/segment to achieve the minimum localization degree. With decimal alarm codes, this is quite easy as it does not involve bit-wise comparison of binary alarm codes. Let J be the maximum number of cycle sets in an ILP solution. We can have at most J bits in each binary alarm code, which is chosen from a candidate set of size 2^J-1. To formulate the inequality among the chosen decimal alarm codes, our idea is to let each of them take one unique integer from $\{1, 2, \ldots, 2^J-1\}$ (unless the corresponding links are on the same segment).

B. Notations

J: The maximum number of cycle sets in the solution.

j: Cycle set index where $j \in \{0, 1, \ldots, J-1\}$.

e^j_{ab}: Binary variable. It equals to 1 if link (a, b) is covered by cs_j, and 0 otherwise.

z^j_a: General integer variable. It is the number of times that cs_j traverses node a.

α_{ab}: General integer variable. It is the decimal alarm code of link (a, b).

r^k_{ab}: Binary variable. It equals to 1 if α_{ab} takes the k-th integer from the candidate set $\{1, 2, \ldots, 2^J-1\}$, and 0 otherwise.

c_{ab}: The cost of adding a unit of supervisory wavelength to link (a, b). If hop-count is used as the cost metric, then c_{ab}=1; otherwise c_{ab} may include distance-related cost.

β : Predefined constant. Its value determines the relative importance between the two cost components.

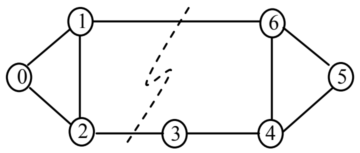


Fig. 2. A cut consisting of link (1, 6) and segment 2−3−4.

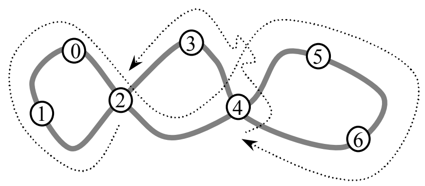


Fig. 3. Non-simple m-cycle.

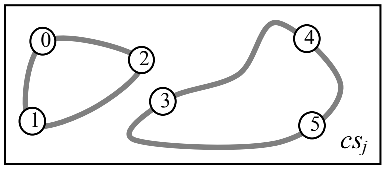


Fig. 4. Multiple disjoint m-cycles coexist in cs_j.

$\boldsymbol{S_{ab}}$: If link (a, b) is on a segment, then $\boldsymbol{S_{ab}}$ is the set of all the links on this segment. Otherwise $\boldsymbol{S_{ab}}$ is a null set ($\boldsymbol{\Phi}$).

$\boldsymbol{S}$: It is a set of links where each link is a delegate of a distinct segment (i.e., an arbitrary link on the segment).

$\boldsymbol{V}$: The set of all the nodes in the network.

$\boldsymbol{E}$: The set of all the links in the network.

C. *ILP Formulation*

Given a topology $\boldsymbol{G(V, E)}$ and the cost c_{ab} for each link (a, b)$\in\boldsymbol{E}$, we formulate an ILP below to generate a cycle cover with the minimum localization degree such that the network cost for link failure detection is minimized. We first assume no segment in the network. Then, every link must have a unique alarm code.

Objective:

$$\text{minimize}\left\{\sum_{(a,b)\in \boldsymbol{E}}\alpha_{ab}+\beta\sum_{j}\sum_{(a,b)\in \boldsymbol{E}}c_{ab}\times e_{ab}^{j}\right\}; \tag{2}$$

Subject to:

$$\sum_{(a,b)\in \boldsymbol{E}} e_{ab}^{j}=2z_{a}^{j}, \qquad \forall a\in \boldsymbol{V},\ \forall j; \tag{3}$$

$$\alpha_{ab}=\sum_{j}2^{j}\times e_{ab}^{j}, \qquad \forall (a,b)\in \boldsymbol{E}; \tag{4}$$

$$\alpha_{ab}=\sum_{k=1}^{2^{J}-1}k\times r_{ab}^{k}, \qquad \forall (a,b)\in \boldsymbol{E}; \tag{5}$$

$$\sum_{k=1}^{2^{J}-1}r_{ab}^{k}=1, \qquad \forall (a,b)\in \boldsymbol{E}; \tag{6}$$

$$\sum_{(a,b)\in \boldsymbol{E}}r_{ab}^{k}\le 1, \qquad \forall k\in\{1, 2, \ldots, 2^{J}-1\}. \tag{7}$$

The objective in (2) minimizes the sum of all the decimal alarm codes plus the cover length weighted by β. Cycle set is defined in (3). Binary alarm codes are converted into decimal codes by (4). A unique alarm code for each link failure is ensured by (5)~(7). Specifically, constraint (5) formulates α_{ab} into a combinatorial sum over the candidate alarm code set {1, 2, …, 2^J−1}. Constraint (6) ensures that each link (a, b) takes one (and only one) alarm code from this set. Constraint (7) says that each alarm code can be assigned to at most one link.

We then consider networks with segments. In this case, constraint (7) is replaced by (8)~(10) below.

$$\alpha_{ab}=\alpha_{cd}, \qquad \forall (a,b),\ (c,d)\in \boldsymbol{E}:\ \boldsymbol{S_{ab}}=\boldsymbol{S_{cd}}\neq\boldsymbol{\Phi}. \tag{8}$$

$$\sum_{(a,b)\in \boldsymbol{S}}r_{ab}^{k}\le 1, \qquad \forall k\in\{1, 2, \ldots, 2^{J}-1\}; \tag{9}$$

$$\sum_{(a,b)\in \boldsymbol{E}}r_{ab}^{k}\le 1+\sum_{(a,b)\in \boldsymbol{S}}\left(\left\|\boldsymbol{S_{ab}}\right\|-1\right)r_{ab}^{k}, \qquad \forall k\in\{1, 2, \ldots, 2^{J}-1\}; \tag{10}$$

Constraint (8) enforces the same alarm code for all the links on the same segment. Constraint (9) ensures one alarm code (k) for at most one segment. From (10), if a decimal alarm code k is assigned to link (a, b)$\in\boldsymbol{S}$, at most $\|\boldsymbol{S_{ab}}\|$ links in the network can have the same alarm code k; otherwise k can be assigned to at most one link that is not on any segment.

In fact, we can always use (8)~(10) to replace (7). If the network contains no segments, then $\boldsymbol{S}$ and $\boldsymbol{S_{ab}}$ are null sets ($\boldsymbol{\Phi}$), and (8)~(10) degenerate to (7).

D. *Discussion*

Let L_S be the total number of links on all the $\|\boldsymbol{S}\|$ segments. We can set the values of J and β using the guidelines below.

$$J=\left\lfloor \log_2\left(\|\boldsymbol{E}\|-L_S+\|\boldsymbol{S}\|\right)\right\rfloor+\delta_1 \tag{11}$$

$$\beta=2^{\left\lfloor \log_2\left(\|\boldsymbol{E}\|-L_S+\|\boldsymbol{S}\|\right)\right\rfloor+\delta_2} \tag{12}$$

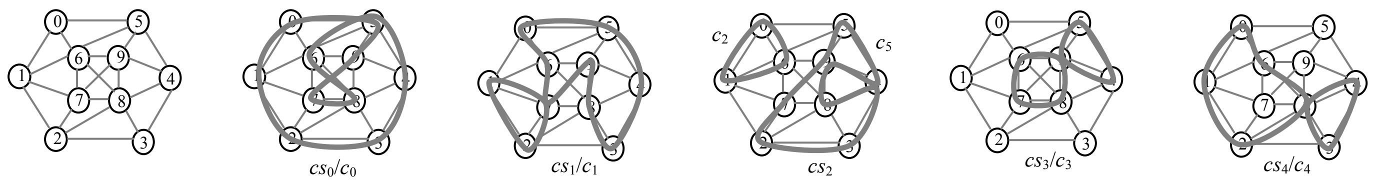
where δ_1 and δ_2 are small positive integers. Note that $\left\lfloor \log_2\left(\|\boldsymbol{E}\|-L_S+\|\boldsymbol{S}\|\right)\right\rfloor+1$ denotes the lower bound for the number of required cycle sets. The actual number of required cycle sets tends to be close to this lower bound. This is because adding an additional cycle set doubles the size of the candidate alarm code set, and results in much higher design flexibility. For example, if a network has $\|\boldsymbol{E}\|$=22 links and no segments, the lower bound is $\left\lfloor \log_2\left(\|\boldsymbol{E}\|-L_S+\|\boldsymbol{S}\|\right)\right\rfloor+1=\left\lfloor \log_2\|\boldsymbol{E}\|\right\rfloor+1=5$. Let δ_1 = 4, we have J=5+4=9. The size of the candidate alarm code set is $2^J-1=2^9-1=511$, which is much larger than $\|\boldsymbol{E}\|$=22. Therefore, the ILP has very high flexibility in choosing only 22 alarm codes from 511 candidates. Since the number of cycle sets is close to the lower bound, β in (12) can provide the necessary balance between the two cost components in (2).

In general, J should be set large enough whereas the ILP will return less number of cycle sets in the solution. But the number of variables and constraints in the ILP soars exponentially with J. From (11), the practical value of J is generally not too large. Therefore, the number of variables and constraints in our ILP can be limited at an acceptable level.

III. NUMERICAL RESULTS

We consider the same set of networks as studied in [2-3]. The ILP is implemented using ILOG CPLEX 10.0 on a standard Pentium IV 2.2 GHz computer, with the following environment parameter settings: 1→emphasis mip, 2→mip strategy probe, 3→mip strategy rins, 3→mip strategy heuristicfreq, 2→mip cuts all, 3→mip strategy dive, 3→preprocessing symmetry.

The SmallNet topology in Fig. 5a is first considered with J=8 and β=0. With β=0, we aim at minimizing the monitor cost only. The cycle sets obtained are shown in Fig. 5a, with the "alarm code table" in Fig. 5b. Note that each "alarm code table" in this section is based on cycle set cs_j (instead of m-cycles as in Fig. 1b). Though it may not have a proper physical meaning (due to multiple m-cycles in cs_j), it is the best way to record the original data directly returned by CPLEX. Besides, it can be easily translated to a true alarm code table. For example, Fig. 5a shows 5 cycle sets cs_0~cs_4 with 6 m-cycles c_0~c_5, where cs_2 consists of a simple m-cycle c_2 and a non-simple m-cycle c_5. Except cs_2, each cycle set cs_j ($0\le j\le 4$) matches an m-



(a) SmallNet topology (with 10 nodes and 22 links) and m-cycles/cycle sets.

Link	cs_7	cs_6	cs_5	cs_4	cs_3	cs_2	cs_1	cs_0	Decimal	Link	cs_7	cs_6	cs_5	cs_4	cs_3	cs_2	cs_1	cs_0	Decimal
(0,1)	0	0	0	1	0	1	0	1	21	(4,5)	0	0	0	0	1	1	1	1	15
(0,5)	0	0	0	0	0	0	1	1	3	(4,8)	0	0	0	1	0	1	0	0	20
(0,6)	0	0	0	1	0	1	1	0	22	(4,9)	0	0	0	0	1	1	0	0	12
(1,2)	0	0	0	1	0	0	1	1	19	(5,6)	0	0	0	0	0	0	0	1	1
(1,6)	0	0	0	0	0	1	0	0	4	(5,9)	0	0	0	0	1	1	0	1	13
(1,7)	0	0	0	0	0	0	1	0	2	(6,7)	0	0	0	0	1	0	1	0	10
(2,3)	0	0	0	0	0	1	0	1	5	(6,8)	0	0	0	1	0	0	0	1	17
(2,7)	0	0	0	0	0	1	1	0	6	(6,9)	0	0	0	0	1	0	0	0	8
(2,8)	0	0	0	1	0	0	0	0	16	(7,8)	0	0	0	0	1	0	0	1	9
(3,4)	0	0	0	1	0	1	1	1	23	(7,9)	0	0	0	0	0	1	1	1	7
(3,8)	0	0	0	1	0	0	1	0	18	(8,9)	0	0	0	0	1	1	1	0	14

(b) Alarm code table based on cs_j.

ILP in this paper:

J: 8
β: 0
Solution time: 581.36 sec
Gap to "optimality": 4.53%
Number of m-cycles: 6
Cover length: 49

HST [2]:

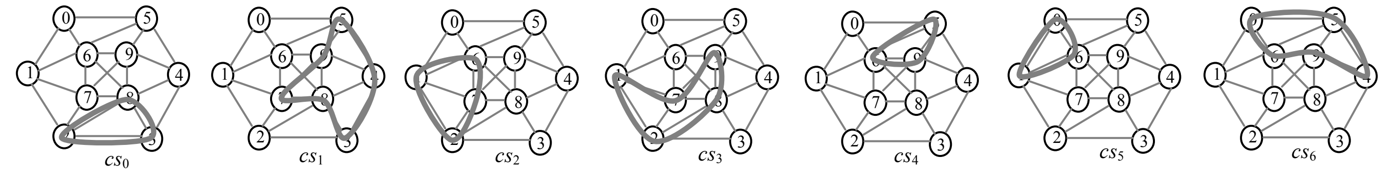
Number of m-cycles: 13
Cover length: 43

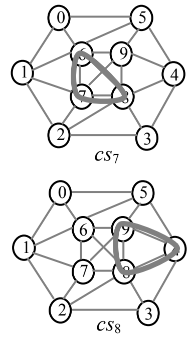
M^2-CYCLE [3]:

Number of m-cycles: 12
Cover length: 36

(c) Comparison with existing algorithms.

Fig. 5. m-cycle design for SmallNet with J=8 and β=0 (D_L=1).





ILP in this paper:

J: 9
β: 1024
Solution time: 19628.64 sec
Gap to "optimality": 1.38%
Number of m-cycles: 9
Cover length: 35

M^2-CYCLE [3]:

Number of m-cycles: 12
Cover length: 36

Link	cs_8	cs_7	cs_6	cs_5	cs_4	cs_3	cs_2	cs_1	cs_0	Decimal	Link	cs_8	cs_7	cs_6	cs_5	cs_4	cs_3	cs_2	cs_1	cs_0	Decimal
(0,1)	0	0	0	1	0	0	0	0	0	32	(4,5)	0	0	1	0	0	0	0	1	0	66
(0,5)	0	0	1	0	0	0	0	0	0	64	(4,8)	1	0	0	0	0	0	0	0	0	256
(0,6)	0	0	1	1	0	0	0	0	0	96	(4,9)	1	0	1	0	0	0	0	0	0	320
(1,2)	0	0	0	0	0	1	1	0	0	12	(5,6)	0	0	0	0	1	0	0	0	0	16
(1,6)	0	0	0	1	0	0	1	0	0	36	(5,9)	0	0	0	0	1	0	0	1	0	18
(1,7)	0	0	0	0	0	1	0	0	0	8	(6,7)	0	1	0	0	0	0	1	0	0	132
(2,3)	0	0	0	0	0	0	0	0	1	1	(6,8)	0	1	0	0	0	0	0	0	0	128
(2,7)	0	0	0	0	0	0	1	0	0	4	(6,9)	0	0	1	0	1	0	0	0	0	80
(2,8)	0	0	0	0	0	1	0	0	1	9	(7,8)	0	1	0	0	0	0	0	1	0	130
(3,4)	0	0	0	0	0	0	0	1	0	2	(7,9)	0	0	0	0	0	1	0	1	0	10
(3,8)	0	0	0	0	0	0	0	1	1	3	(8,9)	1	0	0	0	0	1	0	0	0	264

Fig. 6. m-cycle design for SmallNet with J=9 and β=1024 (D_L=1).

cycle c_j. For link (2, 3), its cs_j based "alarm code" [cs_7, cs_6, cs_5, cs_4, cs_3, cs_2, cs_1, cs_0]=[0, 0, 0, 0, 0, 1, 0, 1]=5 can be translated to a true alarm code [c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0]=[0, 0, 1, 0, 0, 0, 0, 1]=33. This is achieved by translating cs_2 in the former to c_5 in the latter, because (2, 3) is covered by c_5 in cs_2. On the other hand, since link (0, 1) is covered by c_2 in cs_2, its alarm code [0, 0, 0, 1, 0, 1, 0, 1]=21 is the same in both scenarios. Such a translation will change neither the m-cycles nor the localization degree. It only rearranges the values of the alarm codes.

Note that most m-cycles in Fig. 5a are non-simple m-cycles. Fig. 5c compares our ILP-based solution with that generated by HST [2] and M^2-CYCLE [3], all with D_L=1. It is proved in [3] that M^2-CYCLE always outperforms HST on both the monitor cost and the bandwidth cost. So we will only compare our ILP with M^2-CYCLE. Fig. 5c shows that our ILP saves 50% monitors but requires a larger cover length when β=0.

To provide the tradeoff between monitor and bandwidth costs, we vary β to 256, 512 and 1024 while keeping J=8. Our ILP returns solutions of 10, 8 and 9 m-cycles, all with the same cover length of 36 (Note that M^2-CYCLE requires 12 m-cycles and the same cover length of 36). Compared with the case of β=0 in Fig. 5, as expected that the bandwidth cost is reduced, and the monitor cost is increased.

Another observation is that, increasing β from 256 to 1024 does not decrease the cover length. This is because J=8 is not large enough. With J=9 and β=1024, a solution with 9 m-cycles and a cover length of 35 is obtained in about 5 hours, as shown in Fig. 6. Compared to M^2-CYCLE, this gives a 25% cut on monitor cost with one less supervisory wavelength-link. Fig. 6 also shows that large β favors more simple m-cycles in the solution, as more emphasis is put on reducing bandwidth cost.

We then consider networks with segments, namely, NSFNET, Bellcore and ARPA2 [2-3] in Fig. 7. Though our ILP can directly handle the original topologies in Fig. 7, its problem size can be reduced by simplifying the topologies first. This is achieved by treating each segment as a "link" with a

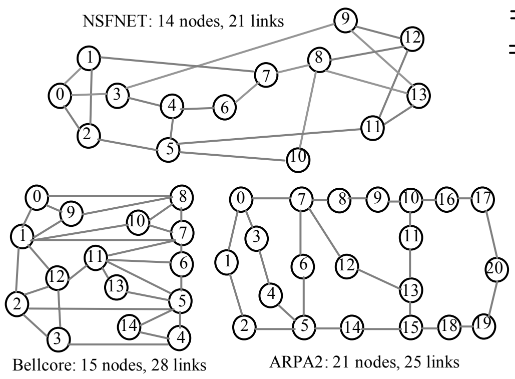

NSFNET: 14 nodes, 21 links
Bellcore: 15 nodes, 28 links
ARPA2: 21 nodes, 25 links

Fig. 7. Three typical topologies taken from [2-3].

Links	cs_5	cs_4	cs_3	cs_2	cs_1	cs_0	Decimal
(0,1), (1,2), (2,5)	0	1	0	0	0	0	16
(0,3), (3,4), (4,5)	0	1	1	0	0	0	24
(0,7)	0	0	1	0	0	0	8
(5,6), (6,7)	0	0	1	0	0	1	9
(5,14), (14,15)	0	0	0	0	0	1	1
(7,8), (8,9), (9,10)	0	0	0	1	0	0	4
(7,12), (12,13)	0	0	0	1	0	1	5
(10,11), (11,13)	0	0	0	1	1	0	6
(10,16), (16,17), (17,20), (19,20), (18,19), (15,18)	0	0	0	0	1	0	2
(13,15)	0	0	0	0	1	1	3

J: 6	No. of m-cycles: 5
β: 1024	Cover length: 35
Time: 170.38 sec	No. of m-cycles in M^2-CYCLE: 5
Gap: 0%	Cover length in M^2-CYCLE: 35

Fig. 10. ARPR2 (D_L=2.500).

Link	cs_7	cs_6	cs_5	cs_4	cs_3	cs_2	cs_1	cs_0	Decimal
(0,1)	0	0	1	0	0	1	0	0	36
(0,2)	0	0	1	0	0	0	0	0	32
(0,3)	0	0	0	0	0	1	0	0	4
(1,2)	0	0	1	0	0	0	0	1	33
(1,7)	0	0	0	0	0	1	0	1	5
(2,5)	0	0	0	0	0	0	0	1	1
(3,4)	0	0	0	0	1	1	0	0	12
(3,9)	0	0	0	0	1	0	0	0	8
(4,5)	0	0	0	1	1	0	0	0	24
(4,6)	0	0	0	1	0	1	0	0	20
(5,10)	0	0	0	1	0	0	0	0	16
(5,11)	0	0	0	0	1	0	0	1	9
(6,7)	0	0	0	1	0	1	0	0	20
(7,8)	0	0	0	1	0	0	0	1	17
(8,10)	0	0	0	1	0	0	0	0	16
(8,12)	0	0	0	0	0	0	1	0	2
(8,13)	0	0	0	0	0	0	1	1	3
(9,12)	0	1	0	0	1	0	1	0	74
(9,13)	0	1	0	0	0	0	1	0	66
(11,12)	0	1	0	0	1	0	0	0	72
(11,13)	0	1	0	0	0	0	0	1	65

J: 8
β: 1024
Time: 12905.52 sec
Gap: 11.08%
No. of m-cycles: 7
Cover length: 36
No. of m-cycles in M^2-CYCLE: 8
Cover length in M^2-CYCLE: 39

Fig. 8. NSFNET (D_L=1.105)

Link	cs_8	cs_7	cs_6	cs_5	cs_4	cs_3	cs_2	cs_1	cs_0	Decimal
(0,1)	0	0	0	1	1	0	0	0	0	48
(0,8)	0	0	0	1	0	1	0	0	0	40
(0,9)	0	0	0	0	1	1	0	0	0	24
(1,2)	0	1	0	0	0	0	0	0	0	128
(1,7)	1	0	0	1	0	0	0	0	0	288
(1,9)	1	0	0	0	1	0	0	0	0	272
(1,10)	0	0	0	0	0	0	1	0	0	4
(1,12)	0	1	0	0	0	0	1	0	0	132
(2,3)	0	0	0	0	0	1	0	1	0	10
(2,5)	0	0	0	0	0	1	0	1	1	11
(2,12)	0	1	0	0	0	0	0	0	1	129
(3,4)	0	0	0	0	0	0	0	1	1	3
(3,12)	0	0	0	0	0	1	0	0	1	9
(4,5)	0	0	0	0	0	0	0	1	0	2
(4,14)	0	0	0	0	0	0	0	0	1	1
(5,6)	0	0	1	0	0	1	0	0	0	72
(5,11)	0	0	1	0	1	0	0	0	0	80
(5,13)	0	0	0	0	1	0	0	0	0	16
(5,14)	0	0	0	0	0	0	0	0	1	1
(6,7)	0	0	1	0	0	0	0	0	0	64
(6,11)	0	0	0	0	0	1	0	0	0	8
(7,8)	0	0	0	1	0	0	0	0	0	32
(7,10)	1	0	0	0	0	0	1	0	0	260
(7,11)	0	0	1	0	0	0	1	0	0	68
(8,9)	1	0	0	0	0	1	0	0	0	264
(8,10)	1	0	0	0	0	0	0	0	0	256
(11,12)	0	0	0	0	0	1	1	0	0	12
(11,13)	0	0	0	0	1	0	0	0	0	16

J: 9	No. of m-cycles: 11
β: 1024	Cover length: 46
Time: 62271.03 sec	No. of m-cycles in M^2-CYCLE: 14
Gap: 4.69%	Cover length in M^2-CYCLE: 46

Fig. 9. BellCore (D_L=1.077).

cost equal to the sum of all link costs on it (e.g. 4−6−7 in NSFNET is treated as a "link" (4, 7) with c_{47}=2). Exceptions include 5−13−11, 4−14−5 in Bellcore (both segments are unchanged) and 0−1−2−5 in APAR2 (treated as a "link" (0, 2) with c_{02}=2 plus a link (2, 5) with c_{25}=1). This is to avoid multiple links between two neighboring nodes (e.g. if 4−14−5 in Bellcore is treated as a "link", two links will exist between nodes 4 and 5). For simplicity, we avoid this case although our ILP can be extended to handle it.

Based on the simplified topologies, the solutions obtained with β=1024 are shown in Figs. 8～10 and are compared to M^2-CYCLE solutions. For NSFNET, CPLEX runs for a very long time, so we terminate it after 3 hours with a gap to "optimality" of 11.08%. Note that in our ILP model this gap is different from the true gap-to-optimality in an optimal design model. From Fig. 8, we can see that our ILP generates a solution with one less m-cycles and a 7.69% cut on the cover length. For BellCore, our ILP achieves the same cover length as M^2-CYCLE, but saves 21.43% on monitor cost (see Fig. 9). For ARPR2, both our ILP and M^2-CYCLE generate the same solution, as shown in Fig. 10.

IV. Conclusion

Compared to channel-based and link-based monitoring schemes, a scheme based on monitoring cycles (m-cycles) can greatly reduce the required number of monitors for fast link failure detection in all-optical networks (AONs). Existing m-cycle design algorithms lack a tradeoff between the two cost components of the network cost (i.e., monitor cost and bandwidth cost). In this paper, we formulated an ILP to achieve the minimum localization degree and to minimize the network cost. The key idea is to minimize the sum of all the decimal alarm codes plus the cover length weighted by a factor, and to assign a unique alarm code to each link or segment.

Our contributions are two-fold: 1) we introduced non-simple m-cycles; 2) our ILP allows an efficient tradeoff between the two cost components. Compared to existing algorithms, our ILP can significantly reduce the network cost required in an m-cycle based monitoring scheme.

References

[1] M. Goyal, K. K. Ramakrishnan and W.-C. Feng, "Achieving faster failure detection in OSPF networks," *IEEE ICC '03*, vol. 1, pp. 296-300, May 2003.

[2] H. Zeng, C. Huang and A. Vukovic, "Spanning tree based monitoring-cycle construction for fault detection and localization in mesh AONs," *IEEE ICC '05*, vol. 3, pp. 1726-1730, May 2005.

[3] B. Wu and K. L. Yeung, "M^2-CYCLE: an optical layer algorithm for fast link failure detection in all-optical mesh networks," *IEEE GLOBECOM '06*, Dec. 2006.

[4] S. Stanic, S. Subramaniam, H. Choi, G. Sahin and H.-A. Choi, "On monitoring transparent optical networks," *Int'l Conf. on Parallel Processing Workshops*, pp. 217-223, Aug. 2002.

[5] B. Wu, K. L. Yeung and S. Z. Xu, "ILP formulation for *p*-cycle construction based on flow conservation," *IEEE GLOBECOM '07*, Dec. 2007.