ILP Formulations for p-Cycle Design Without Candidate Cycle Enumeration

Bin Wu, *Member, IEEE*, Kwan L. Yeung, *Senior Member, IEEE*, and Pin-Han Ho

***Abstract*—The concept of p-cycle (preconfigured protection cycle) allows fast and efficient span protection in wavelength division multiplexing (WDM) mesh networks. To design p-cycles for a given network, conventional algorithms need to enumerate cycles in the network to form a candidate set, and then use an integer linear program (ILP) to find a set of p-cycles from the candidate set. Because the size of the candidate set increases exponentially with the network size, candidate cycle enumeration introduces a huge number of ILP variables and slows down the optimization process. In this paper, we focus on p-cycle design without candidate cycle enumeration. Three ILPs for solving the problem of spare capacity placement (SCP) are first formulated. They are based on recursion, flow conservation, and cycle exclusion, respectively. We show that the number of ILP variables/constraints in our cycle exclusion approach only increases linearly with the network size. Then, based on cycle exclusion, we formulate an ILP for solving the joint capacity placement (JCP) problem. Numerical results show that our ILPs are very efficient in generating p-cycle solutions.**

***Index Terms*—Integer linear program (ILP), p-cycle (preconfigured protection cycle), protection, wavelength division multiplexing (WDM) mesh networks.**

I. Introduction

OPTICAL networks based on wavelength division multiplexing (WDM) provide the backbone infrastructure for high-speed data communications. In WDM networks, hundreds of wavelengths can be multiplexed onto a single fiber for parallel data transmission. This not only fully utilizes the fiber bandwidth, but also efficiently reduces the network cost. Because of the high-speed nature of WDM networks, network survivability is of paramount importance. Upon an accidental failure such as a fiber-cut, it is imperative that the network can achieve fast optical recovery in order to minimize data loss.

As optical network topology evolves from ring to mesh, the concept of p-cycle (preconfigured protection cycle) [1] enables fast span/link protection with high capacity efficiency. The idea is to organize the spare capacity in the network into a set of preconfigured cycles to protect all the traffic on each span (i.e., 100% span protection). Unless otherwise specified, a p-cycle in this paper always refers to a unity-p-cycle, which is implemented by using one unit of spare capacity (or one wavelength) on each span it traverses. A span traversed by a p-cycle is called an *on-cycle span* of this p-cycle. If a span is not traversed by a p-cycle but its two end nodes are, then it is a *straddling span* of this p-cycle. In an undirected network with at most a single span failure at a time, a p-cycle can protect one unit of traffic on each on-cycle span and two units on each straddling span. Fig. 1 gives an example. Spans traversed by the dashed p-cycle are on-cycle spans. Other spans, i.e., (0, 1) and (3, 4), are straddling spans. If on-cycle span (2, 3) fails, one unit of traffic on (2, 3) can be rerouted to the other side of the p-cycle, or path 2-0-4-1-3. If straddling span (0, 1) fails, two backup paths 0-4-1 and 0-2-3-1 are available and thus two units of traffic on span (0, 1) can be protected. Although a p-cycle does not have any spare capacity reserved on the straddling spans, the spare capacity on the p-cycle can be shared to protect both the on-cycle and straddling spans. As a result, this link-based p-cycle protection scheme yields very high capacity efficiency comparable to shared backup path protection (SBPP) [2]. On the other hand, since the spare capacity on the p-cycle is preconfigured, only the two end nodes of the failed span need to fulfill real-time switching upon failure. This leads to a bidirectional line switched ring (BLSR) [3] ring-like fast recovery speed.

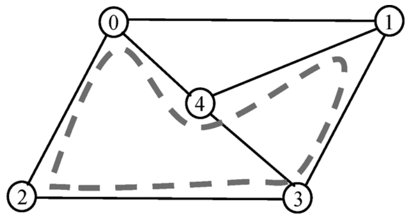


Fig. 1. An example of p-cycle protection.

Because of its outstanding performance on both recovery speed and capacity efficiency, p-cycle has attracted extensive research interests [4]–[23] since it was first introduced in 1998 [1]. Recently, it was also extended to path/segment protection [24], [25] at the cost of a slower recovery speed.

For a given network, p-cycle design [26] can be formulated as either the problem of spare capacity placement (SCP), or joint capacity placement (JCP). In SCP [1], [9], traffic load on each span is given. That means traffic demands have been properly routed according to some routing algorithm (such as shortest path routing). The objective of SCP is to minimize the spare capacity required for 100% span protection. In contrast, JCP design [16], [17] jointly optimizes both the routing scheme and

Manuscript received January 05, 2008; revised November 15, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Subramaniam. First published September 29, 2009; current version published February 18, 2010.

B. Wu and P.-H. Ho are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada N2L 3G1 (e-mail: b7wu@uwaterloo.ca; pinhan@bbcr.uwaterloo.ca).

K. L. Yeung is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam, Hong Kong (e-mail: kyeung@eee.hku.hk).

Digital Object Identifier 10.1109/TNET.2009.2025769

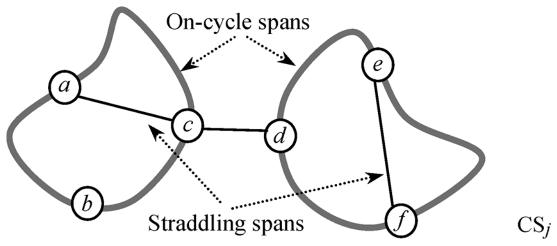


Fig. 2. A cycle set CS_j with two disjoint cycles. Broad-brush lines/curves denote on-cycle spans, and regular lines denote off-cycle spans.

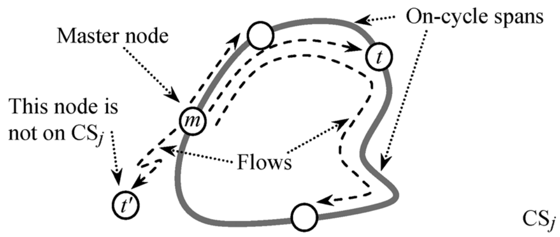


Fig. 3. Master node in Schupke's ILP [22].

the spare capacity placement to minimize the total capacity required.

Conventional algorithms follow a two-step approach to design p-cycles in a given network. The first step enumerates all or a part of all the distinct cycles [27] in the network to form a candidate set. Based on some cycle preselection heuristics [18]–[21], the candidate set may only include a given number of cycles with high merit. The second step finds a set of p-cycles from the candidate set based on an integer linear program (ILP). Generally, the total number of all the cycles in a network soars exponentially with the network size. For p-cycle design in a large-size network, even if only a part of all the cycles are selected as candidate cycles, cycle enumeration may still lead to a large candidate set which slows down the ILP optimization process. While not as theoretically complex as some survivable network design problems, p-cycle design with guaranteed solution quality using cycle enumeration may be too complex to be solved in practice. To this end, some works focus on pure heuristics [19], or modified/relaxed ILP models with only a small set of necessary cycles enumerated [17], [28], [29]. Among them, the column generation approach [28], [29] is the most promising one by employing advanced ILP optimization techniques. It separates the ILP problem into a master problem and a pricing problem, and then adopts an iterative process to find the solution.

In this paper, we focus on formulating efficient optimal ILP models without candidate cycle enumeration. Three ILPs are first formulated for solving the SCP p-cycle design problem. They follow three different approaches, recursion, flow conservation and cycle exclusion. In particular, the number of ILP variables and constraints in the cycle exclusion-based ILP only increases linearly with the network size. We further extend the cycle exclusion-based ILP to solve the problem of JCP p-cycle design.

The rest of the paper is organized as follows. Section II introduces the cycle set definition, which is an important concept in our ILP formulations. Some related work on ILP formulation without candidate cycle enumeration is also reviewed. In Section III, four ILPs are formulated, three for solving the SCP problem and one for the JCP problem. Section IV gives numerical results and Section V concludes the paper.

II. Cycle Set and Related Work

In an ILP formulation, cycles can be defined by requiring each node in the network to have either 2 or 0 on-cycle spans incident on it [22]. With this definition, multiple disjoint cycles may be generated at a time, where two disjoint cycles do not have any common on-cycle span or node. Fig. 2 shows an example of two disjoint cycles. Without loss of generality, we call the set of cycles generated from the above definition as a *cycle set* CS_j with index $j \in \{1, 2, \ldots, J\}$, where J is the maximum number of cycle sets in a solution. We further define that if a span is an on-cycle or straddling span of any cycle in CS_j, it is an on-cycle or straddling span of CS_j. If a cycle set CS_j is chosen to provide p-cycles, it can protect all its on-cycle and straddling spans. In Fig. 2, span (c, d) is *not* a straddling span of CS_j because it straddles *two* disjoint cycles. Therefore, (c, d) is not protected by CS_j. For simplicity, we say that a node or span is on CS_j if it is traversed by any cycle in CS_j.

To the best of our knowledge, Schupke's ILP [22] is the only work for p-cycle design without cycle enumeration. It adopts a flow-based analysis to ensure a single cycle in each CS_j. As a result, a span can be protected by the only cycle in CS_j if its two end nodes are on CS_j. In Schupke's ILP, a unique *master node* is defined on each cycle set CS_j and it serves as the source of all flows, as shown in Fig. 3. Other nodes in the network are *target* nodes. Each target node t receives a flow generated by the master node. If the flow can move along the on-cycle spans of CS_j to reach t, then t is on CS_j. Otherwise, the flow must traverse some spans not on CS_j and, thus, the target node is not on CS_j (e.g., node t' in Fig. 3).

Despite of its theoretical significance of removing candidate cycle enumeration, Schupke's ILP needs a very long running time. This is mainly due to four reasons: 1) the ILP must check all the master-target node pairs in the network to ensure a single cycle in CS_j; 2) the ILP must determine a specific master node on CS_j in order to prove the optimality of the solution, though it does not matter which master node is chosen (as long as it is on CS_j); 3) since each p-cycle is not treated as a unity-p-cycle, the ILP must also determine the number of required copies of each cycle; 4) the number of variables and constraints in the ILP is a quadratic function of the number of nodes in the network. To reduce the running time of this ILP, a four-step heuristic is proposed in [22] for finding suboptimal solutions.

III. ILP Formulations

We formulate ILPs for solving both SCP and JCP problems. For easy reference, we summarize the notations in our ILPs in Fig. 4. Undirected networks are considered with at most a single span failure at a time. We also assume there are enough wavelength channels and wavelength converters to support all necessary optical connections.

A. Recursion-Based ILP for SCP

We observe that it is not necessary to ensure a single cycle in each cycle set CS_j (as in Schupke's ILP [22]). With multiple disjoint cycles in a CS_j, a p-cycle solution can still be con-

Notations

Common notations in all ILP formulations:

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

j: Cycle set index where $j \in \{1, 2, \ldots, J\}$.

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

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

l_{uv}: The number of traffic load units on span (u, v) after routing. In SCP scenario, it is a given parameter. In JCP scenario, it is a general integer variable.

c_{uv}: The cost of adding one unit of spare capacity (i.e., one wavelength) to span (u, v). If hop-count is used as the cost metric, then $c_{uv}=1$ for each span $(u, v) \in \boldsymbol{E}$. Otherwise c_{uv} may include distance-related cost for using amplifiers, plus the cost of any O/E and E/O interfaces, and the amortized cost of OXCs or wavelength converter pools.

L: Predefined length limit of each p-cycle. L can be based on either hop-count or distance-related circumference.

Variables in the recursion-based ILP:

e_{uv}^{j}: Binary variable. It takes 1 if span (u, v) is an on-cycle span of CS$_j$, and 0 otherwise.

z_{u}^{j}: Binary variable. It takes 1 if node u is on CS$_j$, and 0 otherwise.

χ_{uv}^{j}: Binary variable. It takes 1 if span (u, v) can be protected by CS$_j$, and 0 otherwise.

$x_{av}\big|_{(a,b)}^{j}$: Binary variable. Assume we are checking whether span (a, b) can be protected by CS$_j$. It takes 1 if there is a route on CS$_j$ to connect nodes a and v, and 0 otherwise.

$y_{av}^{u}\big|_{(a,b)}^{j}$: Binary variable. Assume we are checking whether span (a, b) can be protected by CS$_j$. For $u \neq b$, it takes 1 if there is a route on CS$_j$ to connect nodes a and u, and at the same time (u, v) is an on-cycle span of CS$_j$ (defined as "a connects to v via u" for simplicity). Otherwise, it takes 0. For $u=b$, $y_{av}^{b}\big|_{(a,b)}^{j}=0$ is always enforced by the ILP.

Variables in the flow conservation-based ILP:

e_{uv}^{j}: Binary variable. It takes 1 if span (u, v) is an on-cycle span of CS$_j$, and 0 otherwise.

z_{u}^{j}: Binary variable. It takes 1 if node u is on CS$_j$, and 0 otherwise.

χ_{uv}^{j}: Binary variable. It takes 1 if span (u, v) can be protected by CS$_j$, and 0 otherwise.

$f_{uv}\big|_{ab}^{j}$: Binary variable. Assume we are checking whether span (a, b) can be protected by CS$_j$. It takes 1 if there is a flow on span (u, v), and 0 otherwise.

$k_{u}\big|_{ab}^{j}$: Binary variable. Assume we are checking whether span (a, b) can be protected by CS$_j$ and $u \neq a$, $u \neq b$. It takes 1 if there is an outbound or inbound flow incident on node u, and 0 otherwise.

Variables and constants in the cycle exclusion-based ILP:

θ_{uv}^{j}: Binary variable. It takes 1 if there is an on-cycle vector $u \rightarrow v$ on span (u, v) when constructing CS$_j$, and 0 otherwise.

z_{u}^{j}: Binary variable. It takes 1 if node u is on CS$_j$, and 0 otherwise.

χ_{uv}^{j}: Binary variable. It takes 1 if span (u, v) can be protected by CS$_j$, and 0 otherwise.

r_{u}^{j}: Binary variable. It takes 1 if node u is a root node on CS$_j$, and 0 otherwise.

p_{u}^{j}: Fractional variable. It is the voltage value of node u when constructing CS$_j$.

α: A predefined fractional constant where $\frac{1}{|\boldsymbol{V}|} \geq \alpha > 0$.

Additional notations in the JCP scenario:

$\boldsymbol{D}$: A given traffic matrix. We assume symmetric demands between any node pair, and only consider the upper triangle of the traffic matrix for routing.

d_{st}: For simplicity, we use $d_{st} \in \boldsymbol{D}$ to denote not only the demand between a source s and a destination t, but also the total number of traffic units of this demand.

d_{st}^{uv}: General integer variable. Since a demand d_{st} may have multiple working paths in the JCP scenario, we use d_{st}^{uv} to denote the number of traffic units of demand d_{st} that are routed through span (u, v) from u to v.

Fig. 4. Notations in the ILP formulations.

structed if we can properly identify individual spans that can be protected by each CS$_j$.

To check whether a span (a, b) can be protected by a cycle set CS$_j$, we examine if there is a route on CS$_j$ connecting nodes a and b, where the route can only consist of on-cycle spans of CS$_j$. We refer this process as checking the *connectivity* between a and b. If such a route exists, then nodes a and b are on the same cycle, or "a connects to b" for short. Accordingly, span (a, b) is either an on-cycle or a straddling span of this cycle, and it can be protected by CS$_j$. Otherwise span (a, b) cannot be protected by CS$_j$.

Motivated by some classic routing algorithms such as Dijkstra's and Floyd-Warshall algorithms [30], we design a recursive process for connectivity checking. In Fig. 5, checking the connectivity between nodes a and b is equivalent to checking the connectivity between nodes a and v, because (v, b) is an on-cycle span. This further depends on the connectivity between nodes a and u if (u, v) is also an on-cycle span, and so on. Define node b as the *starting node* of the recursion. The nodes involved in the recursion are sequentially referred to in a unidirectional manner along the cycle, as indicated by the dashed arrows in Fig. 5. If a and b are on the same cycle, the recursion will be stopped when node a is referred to (defined as *the first stop condition*). Then, span (a, b) can be protected by CS$_j$. On

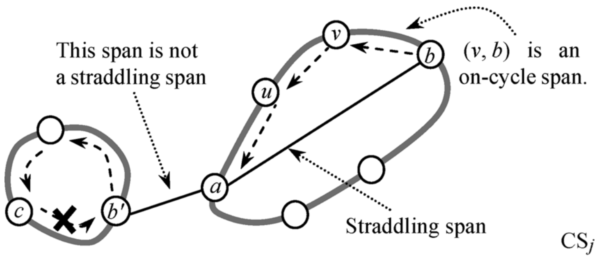

This span is not a straddling span
(v, b) is an on-cycle span.
v
b
u
a
b'
c
Straddling span
CS_j

Fig. 5. Recursive process.

the other hand, if span (a, b') in Fig. 5 is checked and node b' is the starting node, the recursion will be stopped when node b' is revisited (defined as *the second stop condition*). In this case, nodes a and b' are not connected and thus span (a, b') cannot be protected by CS_j.

Based on the above mechanism, an ILP for solving the SCP problem is formulated below, with given traffic load l_{uv} and cost c_{uv} for each span $(u, v) \in \boldsymbol{E}$.

$$\text{minimize} \left\{ \sum_j \sum_{(u,v)\in \boldsymbol{E}} c_{uv} e^j_{uv} \right\} \tag{1}$$

$s.t.$

$$\sum_{(u,v)\in \boldsymbol{E}} e^j_{uv} = 2z^j_u, \quad \forall u \in \boldsymbol{V}, \quad \forall j \tag{2}$$

$$\sum_j \left(2\chi^j_{uv} - e^j_{uv}\right) \geq l_{uv}, \quad \forall (u,v) \in \boldsymbol{E} \tag{3}$$

$$x_{aa}|^j_{(a,b)} = 1, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \tag{4}$$

$$y^u_{av}|^j_{(a,b)} \leq \frac{1}{2}\left(x_{au}|^j_{(a,b)} + e^j_{uv}\right), \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \quad \forall v \in \boldsymbol{V} : v \neq a, \quad \forall u \in \boldsymbol{V} : (u,v) \in \boldsymbol{E} \tag{5}$$

$$x_{av}|^j_{(a,b)} \leq \sum_{(u,v)\in \boldsymbol{E}} y^u_{av}|^j_{(a,b)}, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \quad \forall v \in \boldsymbol{V} : v \neq a \tag{6}$$

$$\sum_{(v,b)\in \boldsymbol{E}, \quad v\neq a} y^b_{av}|^j_{(a,b)} \leq 0, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \tag{7}$$

$$\chi^j_{ab} = x_{ab}|^j_{(a,b)}, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \tag{8}$$

$$x_{av}|^j_{(a,b)} \leq \sum_{(a,u)\in \boldsymbol{E}} e^j_{au}, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \quad \forall v \in \boldsymbol{V} : v \neq a \tag{9}$$

$$y^u_{av}|^j_{(a,b)} + y^v_{au}|^j_{(a,b)} \leq 1, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \quad \forall (u,v) \in \boldsymbol{E} : u \neq a, v \neq a. \tag{10}$$

Objective in (1) aims at minimizing the total cost of all cycle sets/p-cycles. Constraint (2) defines a cycle set CS_j by requiring each node in the network to have either 2 or 0 on-cycle spans incident on it. Constraint (3) ensures 100% span protection, where $2\chi^j_{uv} - e^j_{uv}$ gives the number of traffic units on span (u, v) that can be protected by CS_j. $2\chi^j_{uv} - e^j_{uv}$ takes 1 if (u, v) is an on-cycle span, 2 if (u, v) is a straddling span, and 0 otherwise. Constraints (4)–(10) formulate the recursive process to check whether span (a, b) can be protected by CS_j. Constraint (4) specifies that node a must always "connect" to itself. This provides the first stop condition for the recursion when node a is referred to. Constraint (5) defines "a connects to v via u", where $y^u_{av}|^j_{(a,b)} = 1$ if a connects to u (i.e., $x_{au}|^j_{(a,b)} = 1$) and (u, v) is an on-cycle span (i.e., $e^j_{uv} = 1$). Constraint (6) stipulates that, a connects to v on CS_j only if there exists a neighbor u of v such that "a connects to v via u." Constraint (7) prohibits any possible loopback to the starting node b by preventing a to connect with any node via b. This provides the second stop condition for the recursion. Constraint (8) determines whether span (a, b) can be protected by CS_j. Constraint (9) means that, if there is no on-cycle span incident on node a, then we cannot find a route on CS_j between a and any other node in the network. Finally, (10) ensures that the recursion is carried out in a unidirectional manner. Assume (u, v) is an on-cycle span, and we have referred to node u for checking the connectivity between nodes a and v. With constraint (10), we cannot refer back to node v when we further check the connectivity between nodes a and u.

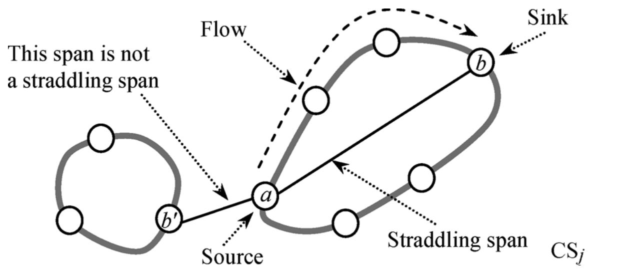


Fig. 6. Flow conservation approach.

Compared with Schupke's ILP [22], the above recursion-based ILP runs much faster. Instead of checking all the node pairs to ensure a single cycle in each CS_j, it only checks whether each individual span can be protected by a CS_j. Since each p-cycle is defined as a unity-p-cycle, this ILP does not calculate the required number of copies of each p-cycle. Approximately, the ILP formulated in (1)–(10) has $J(2|\boldsymbol{E}| + |\boldsymbol{V}|)(|\boldsymbol{E}| + 1)$ variables and $3J|\boldsymbol{E}|^2 + J(|\boldsymbol{E}| + |\boldsymbol{V}| + 2|\boldsymbol{E}| \times |\boldsymbol{V}|) + |\boldsymbol{E}|$ constraints (the exact number of variables and constraints depends on whether node a or b serves as the starting node). Note that J is the maximum number of cycle sets allowed in the solution. Its value can be set according to (37).

B. Flow Conservation-Based ILP for SCP

Unlike the recursion-based ILP above, we now adopt a flow conservation-based analysis to check whether span (a, b) can be protected by CS_j. Without loss of generality, we assume that node a is the source and node b is the sink. We check if a flow between a and b can be carried only by the on-cycle spans of CS_j. (Note that Schupke's ILP [22] allows a flow to move along any span in the network.) Source a can generate at most one flow but it does not receive any flow. Similarly, sink b can receive at most one flow but it does not generate or relay any flow. Except source a and sink b, all other nodes in the network must obey flow conservation [30]. If both a and b are on the same cycle as in Fig. 6, then a flow between a and b exists, and span (a, b) can be protected by CS_j. If span (a, b') in Fig. 6 is checked instead, no flow exists between a and b', and span (a, b') cannot be protected by CS_j.

For an arbitrary node in the network, if an on-cycle span incident on it carries a flow, we say that there is a unit-flow incident on this node. To reduce the number of variables and constraints in ILP, we can use an undirected flow to replace the directed one in Fig. 6. Then, flow conservation can be formulated by requiring each node (except source a and sink b) to have either 2 or 0 unit-flows incident on it, whereas a or b can have at most 1 unit-flow incident on it. With the notations defined in Fig. 4,

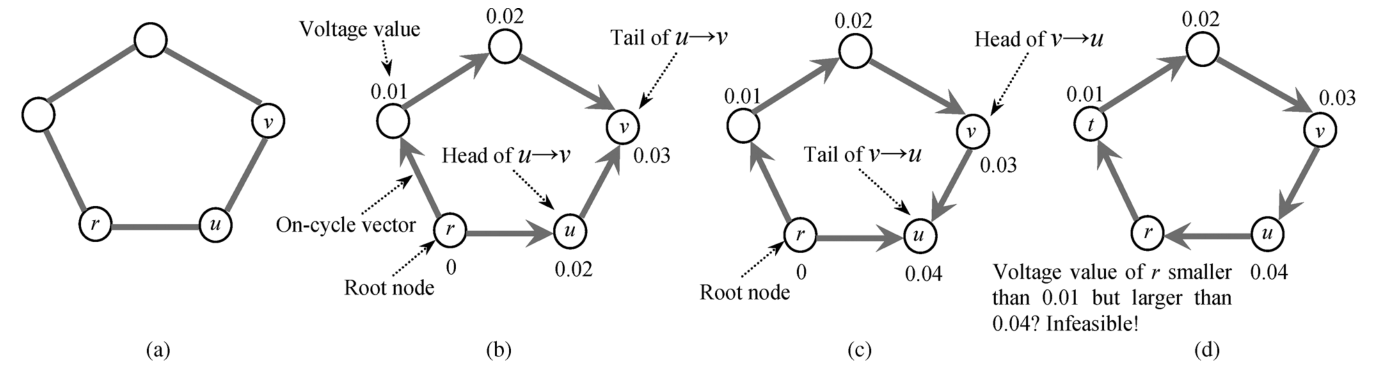


Fig. 7. On-cycle vector, voltage, reversal and root node. (a) A p-cycle in CS_j. (b) A reversal at node v. (c) A reversal at node u. (d) Voltage value conflict at node r.

the flow conservation-based ILP is formulated here for solving the SCP problem.

$$\text{minimize} \quad \left\{ \sum_j \sum_{(u,v)\in \boldsymbol{E}} c_{uv} e_{uv}^j \right\} \tag{11}$$

$s.t.$

$$\sum_{(u,v)\in \boldsymbol{E}} e_{uv}^j = 2z_u^j, \quad \forall u \in \boldsymbol{V}, \quad \forall j \tag{12}$$

$$\sum_j \left(2\chi_{uv}^j - e_{uv}^j\right) \geq l_{uv}, \quad \forall (u,v) \in \boldsymbol{E} \tag{13}$$

$$\chi_{ab}^j \leq \frac{1}{2}\left(z_a^j + z_b^j\right), \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \tag{14}$$

$$f_{uv}|_{ab}^j \leq e_{uv}^j, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall (u,v) \in \boldsymbol{E}, \quad \forall j \tag{15}$$

$$\sum_{(a,v)\in \boldsymbol{E}} f_{av}|_{ab}^j \leq 1, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \tag{16}$$

$$\sum_{(u,b)\in \boldsymbol{E}} f_{ub}|_{ab}^j \leq 1, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j \tag{17}$$

$$\sum_{(u,v)\in \boldsymbol{E}} f_{uv}|_{ab}^j = 2k_u|_{ab}^j, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j$$
$$\forall u \in \boldsymbol{V} : u \neq a, u \neq b \tag{18}$$

$$\chi_{ab}^j \leq \sum_{(a,v)\in \boldsymbol{E}} f_{av}|_{ab}^j, \quad \forall (a,b) \in \boldsymbol{E}, \quad \forall j. \tag{19}$$

Objective in (11) aims at minimizing the total cost of all p-cycles (or cycle sets). Constraint (12) defines cycle sets and (13) ensures 100% span protection. Constraint (14) gives a necessary (but not sufficient) condition to identify those spans protected by CS_j, i.e., a span can be protected by CS_j only if its two end nodes are on CS_j. This constraint confines the solution space and thus speeds up the optimization process. Constraints (15)–(19) check whether a span (a,b) can be protected by CS_j based on flow conservation. By (15), flows can only move along the on-cycle spans of CS_j. Constraints (16) and (17) require source a and sink b to have at most 1 unit-flow incident on each. Constraint (18) formulates the flow conservation property for other nodes in the network. Finally, (19) indicates that, (a,b) can be protected by CS_j if there is a unit-flow incident on source a. In fact, (a,b) can be protected by CS_j if any flow exists on CS_j.

Compared with the recursion-based ILP, the above flow conservation-based ILP is simpler. To check whether a span can be protected by a CS_j, the former must retrieve nodal connectivity in a recursive manner, but the latter only examines the existence of a unit-flow on CS_j. In other words, the former requires "node level" details but the latter only requires "path level" details. The flow conservation-based ILP in (11)–(19) involves $J|\boldsymbol{E}|^2 + J|\boldsymbol{E}| \times |\boldsymbol{V}| + J|\boldsymbol{V}|$ variables and $J|\boldsymbol{E}| \times (|\boldsymbol{E}| + |\boldsymbol{V}| + 2) + J|\boldsymbol{V}| + |\boldsymbol{E}|$ constraints. Both numbers are smaller than that in the recursion-based ILP.

C. *Cycle Exclusion-Based ILP for SCP*

Schupke's ILP [22] pays great effort in ensuring a single cycle in each cycle set CS_j. In return, it is very simple in checking whether a span can be protected by a CS_j. In Schupke's ILP, span (a,b) can be protected by CS_j if and only if both a and b are on the only cycle in CS_j. In contrast, our recursion and flow conservation based ILPs greatly simplify cycle formulation by allowing multiple disjoint cycles in CS_j, but the downside is that we need a more complex process to check whether each span (a,b) can be protected by CS_j. In the following, we formulate a cycle exclusion-based ILP to take the advantages from both sides, i.e. simple cycle formulation and easy protection checking.

We use a directed *on-cycle vector* (*vector* for short) to denote each on-cycle span. An on-cycle span (u,v) of CS_j is denoted by either $u \to v$ or $v \to u$, but not both. The direction of the vector is not important, as either vector sufficiently denotes the on-cycle span (u,v). If a span in the network is not associated with a vector, then it is not an on-cycle span of CS_j. For the p-cycle in Fig. 7(a), Fig. 7(b) and (c) give two possible representations using different sets of vectors. For each vector $u \to v$, we define node u as the *head* and node v as the *tail*. A value called *voltage* is assigned to each node in the network, and its absolute value is not important. For each vector $u \to v$, we only require that its tail v has a larger voltage value than its head u. If two vectors have a common tail on CS_j, e.g. node v in Fig. 7(b) and node u in Fig. 7(c), we say that there is a *reversal* at this node, and the common tail is called a *reversal node*. If two vectors have a common head on CS_j, e.g., node r in Fig. 7(b) and (c), we call this common head a *root node*.

In Fig. 7(b)–(d), a number (fraction) next to each node denotes a possible voltage value for that node. If there is a single root-reversal node pair on the cycle [as in Fig. 7(b) and (c)], then

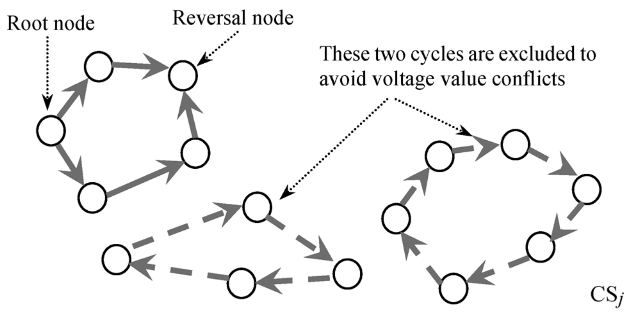

Root node
Reversal node
These two cycles are excluded to avoid voltage value conflicts
CS_j

Fig. 8. Only the cycle with a root-reversal node pair can exist in CS_j.

a feasible set of voltage values exists. If a cycle does not have any root-reversal node pair [as in Fig. 7(d)], there must exist a *voltage value conflict* at some node on the cycle. An example is shown in Fig. 7(d). If we start from node r and move along the direction of the vectors, the nodal voltage value increases monotonically until we loop back to the starting point r. Node r should have a voltage value larger than 0.04 of node u, but smaller than 0.01 of node t. This causes a conflict.

Based on the above observation, we can see that ensuring a single cycle in each CS_j is equivalent to ensuring a unique root-reversal node pair in each CS_j. When there are multiple cycles in CS_j (as in Fig. 8), only the one with the root-reversal node pair remains, and all other cycles will be excluded to avoid voltage value conflicts. We call this process *cycle exclusion*. After cycle exclusion, a span can be protected by the only cycle in CS_j if and only if its two end nodes are on CS_j. With the notations defined in Fig. 4, the cycle exclusion-based ILP is formulated below for solving the SCP p-cycle design problem. Note that we still call it an ILP for simplicity, although a fractional relaxation on voltage values is used to speed up the optimization.

$$\text{minimize}\left\{\sum_j \sum_{(u,v)\in\boldsymbol{E}} c_{uv}\left(\theta^j_{uv}+\theta^j_{vu}\right)\right\} \tag{20}$$

$s.t.$

$$\theta^j_{uv}+\theta^j_{vu}\le 1, \quad \forall(u,v)\in\boldsymbol{E}, \quad \forall j \tag{21}$$

$$\sum_{(u,v)\in\boldsymbol{E}}\left(\theta^j_{uv}+\theta^j_{vu}\right)=2z^j_u, \quad \forall u\in\boldsymbol{V}, \quad \forall j \tag{22}$$

$$\sum_j\left(2\chi^j_{uv}-\theta^j_{uv}-\theta^j_{vu}\right)\ge l_{uv}, \quad \forall(u,v)\in\boldsymbol{E} \tag{23}$$

$$\sum_{u\in\boldsymbol{V}} r^j_u\le 1, \quad \forall j \tag{24}$$

$$\sum_{(u,v)\in\boldsymbol{E}}\theta^j_{uv}\le 1+r^j_u, \quad \forall u\in\boldsymbol{V}, \quad \forall j \tag{25}$$

$$p^j_v-p^j_u\ge\alpha\times\theta^j_{uv}-\left(1-\theta^j_{uv}\right) \quad \forall(u,v),(v,u)\in\boldsymbol{E}, \quad \forall j \tag{26}$$

$$\chi^j_{uv}\le\frac{1}{2}\left(z^j_u+z^j_v\right), \quad \forall(u,v)\in\boldsymbol{E}, \quad \forall j. \tag{27}$$

The total cost of all p-cycles is minimized by (20). Constraint (21) associates each span $(u,v)\in\boldsymbol{E}$ to at most one vector in each CS_j, either $u\to v$, $v\to u$, or none. Constraint (22) defines cycle sets CS_j using vectors. Each node in the network has either 2 or 0 on-cycle vectors incident on it (regardless of the direction of the vectors). Constraint (23) ensures 100% span protection. Constraint (24) allows at most one root node in each CS_j. Constraint (25) says that, only the root node can serve as a common head for two vectors, and any other node in the network can be a head for at most one vector. This ensures a single root-reversal node pair in CS_j. Constraint (26) requires the tail of each vector to have a larger voltage value than the head. Constraint (27) indicates that, a span can be protected by the only cycle in CS_j if its two end nodes are on CS_j.

From (26), we have $p^j_v-p^j_u\ge -1$ and $p^j_u-p^j_v\ge -1$ if (u,v) is not an on-cycle span of CS_j, or $|p^j_v-p^j_u|\le 1$. Note that the network contains $|\boldsymbol{V}|$ nodes and their voltage values may be arranged in a monotonically increasing order with a step of (at least) α as formulated in (26). To ensure that each node has a proper voltage value, we set

$$\frac{1}{|\boldsymbol{V}|}\ge\alpha>0. \tag{28}$$

Note that α is a predefined constant. Its specific value is also not important as long as (28) is satisfied. On the other hand, voltage values can also be defined as integers if constraint (26) is replaced by (29) below, where β is an arbitrary constant not smaller than $|\boldsymbol{V}|$.

$$p^j_v-p^j_u\ge\theta^j_{uv}-\beta\left(1-\theta^j_{uv}\right), \quad \forall(u,v),(v,u)\in\boldsymbol{E}, \quad \forall j. \tag{29}$$

This turns the formulation into a true ILP.

Compared with the recursion and the flow conservation based ILPs, which require either "node level" or "path level" details, the cycle exclusion-based ILP only involves "cycle level" details. Since this ILP ensures a single cycle per CS_j, it is very efficient in checking whether a span can be protected by a CS_j. In contrast, both the recursion and the flow conservation based ILPs need a dedicated set of variables and constraints for each span. The cycle exclusion-based ILP in (20)–(27) involves $3J(|\boldsymbol{E}|+|\boldsymbol{V}|)$ variables and $4J|\boldsymbol{E}|+2J|\boldsymbol{V}|+|\boldsymbol{E}|+J$ constraints, which only increase linearly with the network size if J is given.

D. Cycle Exclusion-Based ILP for JCP

In the JCP problem, traffic demands are expressed as the bandwidth requirements for all source-destination node pairs (i.e., traffic matrix $\boldsymbol{D}$). Since working paths are jointly optimized with spare capacity placement, working capacity l_{uv} on each span (u,v) becomes a design parameter. For simplicity, we focus on an undirected network, and only consider the upper triangle of a symmetric traffic matrix $\boldsymbol{D}$. To facilitate our formulation on the routing process, we first treat each demand d_{st} as a directed demand from source s to destination t. Then, the required working capacity l_{uv} on each span (u,v) is obtained by adding up the working capacity on (u,v) in both directions.

Our ILP for solving the JCP problem is based on the cycle exclusion approach formulated in (20)–(27), due to its simplicity as compared with both the recursion and the flow conservation

TABLE I
COMPARISON OF THE NUMBER OF VARIABLES AND CONSTRAINTS IN DIFFERENT ILPS

ILPs	Number of variables	Number of constraints
Conventional ILP with cycle enumeration [1]	$\|\boldsymbol{C}\|$ (the total number of cycles in the network)	$\|\boldsymbol{E}\|$
Schupke's ILP [22]	$J(\|\boldsymbol{V}\|^2+\|\boldsymbol{V}\|\times\|\boldsymbol{E}\|+4\|\boldsymbol{E}\|+\|\boldsymbol{V}\|+1)$	$J(\|\boldsymbol{V}\|^2+\|\boldsymbol{V}\|\times\|\boldsymbol{E}\|+9\|\boldsymbol{E}\|+\|\boldsymbol{V}\|+1)+\|\boldsymbol{E}\|$
Recursion-based ILP for SCP (1)-(10)	$\approx J(2\|\boldsymbol{E}\|+\|\boldsymbol{V}\|)(\|\boldsymbol{E}\|+1)$	$\approx 3J\|\boldsymbol{E}\|^2+J(\|\boldsymbol{E}\|+\|\boldsymbol{V}\|+2\|\boldsymbol{E}\|\times\|\boldsymbol{V}\|)+\|\boldsymbol{E}\|$
Flow conservation-based ILP for SCP (11)-(19)	$J\|\boldsymbol{E}\|^2+J\|\boldsymbol{E}\|\times\|\boldsymbol{V}\|+J\|\boldsymbol{V}\|$	$J\|\boldsymbol{E}\|\times(\|\boldsymbol{E}\|+\|\boldsymbol{V}\|+2)+J\|\boldsymbol{V}\|+\|\boldsymbol{E}\|$
Cycle exclusion-based ILP for SCP (20)-(27)	$3J(\|\boldsymbol{E}\|+\|\boldsymbol{V}\|)$	$4J\|\boldsymbol{E}\|+2J\|\boldsymbol{V}\|+\|\boldsymbol{E}\|+J$
Cycle exclusion-based ILP for JCP (30), (21)-(27) and (31)-(34)	$3J(\|\boldsymbol{E}\|+\|\boldsymbol{V}\|)+2\|\boldsymbol{D}\|\times\|\boldsymbol{E}\|$	$4J\|\boldsymbol{E}\|+2J\|\boldsymbol{V}\|+\|\boldsymbol{E}\|+J+\|\boldsymbol{E}\|+\|\boldsymbol{D}\|\times\|\boldsymbol{V}\|$

based approaches. In particular, objective in (20) is replaced by (30).

$$\text{minimize}\left\{\sum_j \sum_{(u,v)\in\boldsymbol{E}} c_{uv}\left(\theta_{uv}^j+\theta_{vu}^j\right) + \sum_{d_{st}\in\boldsymbol{D}} \sum_{(u,v)\in\boldsymbol{E}} c_{uv}\left(d_{st}^{uv}+d_{st}^{vu}\right)\right\} \quad (30)$$

In addition to (21)–(27), constraints (31)–(34) below are required for routing optimization.

$$l_{uv}=\sum_{d_{st}\in\boldsymbol{D}}\left(d_{st}^{uv}+d_{st}^{vu}\right),\quad \forall(u,v)\in\boldsymbol{E} \quad (31)$$

$$\sum_{(s,v)\in\boldsymbol{E}} d_{st}^{sv}=d_{st},\quad \forall d_{st}\in\boldsymbol{D} \quad (32)$$

$$\sum_{(u,t)\in\boldsymbol{E}} d_{st}^{ut}=d_{st},\quad \forall d_{st}\in\boldsymbol{D} \quad (33)$$

$$\sum_{(u,k)\in\boldsymbol{E}} d_{st}^{uk}=\sum_{(k,v)\in\boldsymbol{E}} d_{st}^{kv},\quad \forall d_{st}\in\boldsymbol{D} \quad \forall k\in\boldsymbol{V}: k\neq s,\quad k\neq t. \quad (34)$$

Constraint (31) calculates l_{uv} by adding up the working capacity on span (u,v) for all demands d_{st}. For any demand d_{st}, (32)–(33) specify that the number of traffic units generated at source s and arrived at destination t must equal to d_{st}. Constraint (34) ensures flow conservation for each demand d_{st}. Except source s and destination t, all other nodes in the network must have equal number of inbound and outbound flow/traffic units for each demand d_{st}.

In addition to the $3J(|\boldsymbol{E}|+|\boldsymbol{V}|)$ variables and $4J|\boldsymbol{E}|+2J|\boldsymbol{V}|+|\boldsymbol{E}|+J$ constraints in (21)–(27), (30)–(34) introduce additional $2|\boldsymbol{D}|\times|\boldsymbol{E}|$ variables and $|\boldsymbol{E}|+|\boldsymbol{D}|\times|\boldsymbol{V}|$ constraints, where $|\boldsymbol{D}|$ is the number of nonzero demands in the traffic matrix $\boldsymbol{D}$.

E. Discussion

Our ILPs can be easily extended to p-cycle design with a hop-count or circumference limit L. In either recursion or flow conservation based ILP, this can be achieved by adding (35).

$$\sum_{(u,v)\in\boldsymbol{E}} c_{uv}e_{uv}^j\leq L,\quad \forall j. \quad (35)$$

For the two cycle exclusion based ILPs (for SCP and JCP, respectively), (36) is needed instead.

$$\sum_{(u,v)\in\boldsymbol{E}} c_{uv}\left(\theta_{uv}^j+\theta_{vu}^j\right)\leq L,\quad \forall j. \quad (36)$$

Note that J (the maximum number of cycle sets allowed in the solution) is a predetermined parameter in all our ILPs. It should be set sufficiently large whereas the final solution may contain less cycle sets. But, a large J may slow down the optimization process, because for a given network the number of variables and constraints increases linearly with J. Define a *segment* as a path consisting of at least two spans and with a degree of 2 at any intermediate node. Let $\boldsymbol{S}$ be the set of spans on all segments and δ be a small positive integer. For solving the SCP p-cycle design problem, we can set J according to (37).

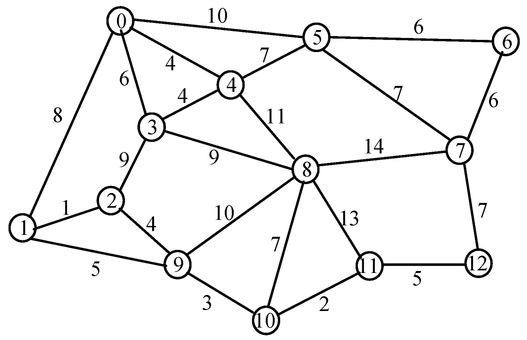
$$J=\max\left\{\left\lceil\frac{l_{uv}}{2}\right\rceil:(u,v)\in\boldsymbol{E}-\boldsymbol{S},\quad l_{uv}:(u,v)\in\boldsymbol{S}\right\}+\delta. \quad (37)$$

This is because p-cycles tend to straddle the most heavily loaded span so as to protect two units of traffic on it. On the other hand, p-cycles cannot straddle a span on a segment, and thus only one unit of traffic on such a span can be protected by a p-cycle traversing it. For JCP p-cycle design, (37) cannot be applied because l_{uv} on each span (u,v) is yet to be determined. As a substitute, we can first try shortest path routing based on the given traffic matrix to calculate an estimated l_{uv} for each span (u,v), and then use (37) to calculate the required value of J.

Table I compares the number of variables and constraints involved in different ILPs. In the conventional ILP with candidate cycle enumeration [1], the number of variables equals to the total number of cycles in the network, which increases exponentially with the network size. In Schupke's ILP [22] and our recursion and flow conservation based ILPs, both the number of variables and constraints are quadratic functions of the number of nodes $|\boldsymbol{V}|$ or spans $|\boldsymbol{E}|$. Notably, those numbers only increase linearly with $|\boldsymbol{V}|$ and $|\boldsymbol{E}|$ in our cycle exclusion-based ILP for SCP (assume J is given).

IV. NUMERICAL RESULTS

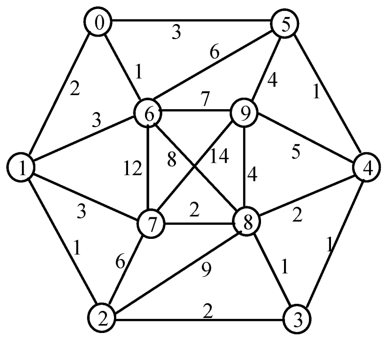
We use CPLEX 10.0 [31] to solve the ILPs on a standard Pentium IV 2.2 GHz computer, with CPLEX parameters below



J=7	RC	FC	CE
Time in seconds	844.63	4293.03	334.59
Total cost	85	85	85
Gap to optimality	4.31%	4.31%	4.90%

SCP solution generated by CE	
4–5–6–7–12–11–10–9–8–4	0–1–2–9–10–11–12–7–6–5–4–8–3–0
0–1–9–2–3–8–10–11–12–7–6–5–4–0	0–1–2–9–8–11–12–7–6–5–4–3–0
0–1–2–9–10–11–12–7–6–5–4–8–3–0	0–3–2–1–9–8–10–11–12–7–5–4–0
0–3–2–1–9–8–10–11–12–7–6–5–4–0	

(a)



J=7	RC	FC	CE
Time in seconds	341.21	139.86	4.29
Total cost	48	48	48
Gap to optimality	4.17%	4.17%	4.17%

SCP solution generated by CE	
0–1–2–7–8–6–9–4–5–0	1–2–3–4–5–9–6–8–7–1
1–2–7–8–9–6–1	1–2–7–8–9–6–1
6–7–8–9–6	6–7–8–9–6
0–1–7–8–2–3–4–5–9–6–0	

(b)

Fig. 9. SCP solutions for two networks with $c_{uv} = 1$ at each span $(u, v) \in \boldsymbol{E}$. (a) A network taken from [23] ($|\boldsymbol{V}| = 13, |\boldsymbol{E}| = 23$). (b) SmallNet with $|\boldsymbol{V}| = 10$ and $|\boldsymbol{E}| = 22$.

to speed up the optimization.

$$\begin{cases} 1 \rightarrow \text{emphasis mip} \\ 2 \rightarrow \text{mip strategy probe} \\ 3 \rightarrow \text{mip strategy rins} \\ 3 \rightarrow \text{mip strategy heuristicfreq} \\ 2 \rightarrow \text{mip cuts all} \\ 3 \rightarrow \text{mip strategy dive} \\ 3 \rightarrow \text{preprocessing symmetry.} \end{cases} \tag{38}$$

Unless otherwise specified, solutions obtained are SCP solutions without cycle length limit. In all examples, we set $\alpha = 0.01$ for the cycle exclusion-based ILPs. For brevity, in the figures we denote the four ILPs formulated in this paper by RC (recursion-based ILP for SCP), FC (flow conservation-based ILP for SCP), CE (cycle exclusion-based ILP for SCP), and JCP (cycle exclusion-based ILP for JCP), respectively.

The two networks in Fig. 9 are first considered, with span cost $c_{uv} = 1$ at each span $(u, v) \in \boldsymbol{E}$. A number next to each span in the topologies gives the number of traffic units l_{uv} on that span (same for other examples). Since the ILPs need relatively long time to prove the optimality of the solution, we only take solutions within a 5% gap to optimality (by setting $0.05 \rightarrow$ mip tolerance mipgap in CPLEX). Note that the total cost 85 in Fig. 9(a) is actually the optimal result according to [23]. Though all of our three ILPs for SCP generate a solution in a reasonable amount of time, we can see that the time required by the cycle exclusion-based ILP is the shortest.

A case study based on the pan-European COST 239 network (with 11 nodes and 26 spans) is shown in Fig. 10, with distance-related span costs defined in Fig. 10(a). The traffic matrix in Fig. 10(b) is obtained by dividing the traffic matrix in [32] by 10 Gbps. The demands are then routed according to shortest path routing, and Fig. 10(c) shows the traffic load on each span after routing. Based on Fig. 10(c), the SCP solutions obtained from our recursion, flow conservation and cycle exclusion based ILPs with $J = 7$ are summarized in Fig. 10(d). We can see that the cycle exclusion-based ILP dramatically cuts down the running time when compared with the other two. In particular, Fig. 10(e) lists the set of p-cycles generated by the cycle exclusion-based ILP, which is obtained in only 34.68 s. Among the 7 p-cycles in Fig. 10(e), the one marked with an asterisk has the largest circumference of 5940 km. Fig. 10(f) gives another SCP solution with a p-cycle circumference limit of 5800 km. It is obtained by including constraint (36) in the cycle exclusion-based ILP.

JCP is also considered in Fig. 10 with the same traffic matrix in Fig. 10(b). The solution obtained (without p-cycle length limit) contains not only a set of p-cycles as listed in Fig. 10(g), but also a routing table as shown in Fig. 10(h). In Fig. 10(g), we can see that there are only 4 p-cycles in the solution. This explains why the JCP solution is obtained faster (only in 18.26 s) than its SCP counterpart (in 34.68 s with 7 p-cycles). In Fig. 10(h), we can observe the differences between JCP routing and shortest path routing adopted in the SCP scenario. Specifically, the demands in the dashed rectangles are assigned with two working paths, whereas in shortest path routing we have required each demand to follow a single working path. Besides, in Fig. 10(h) many demands do not follow the shortest path. For example, the working path for demand d_{06} is $0 \rightarrow 6$ with a length of 740 km, but the shortest path between nodes 0 and 6 is 0-3-6 with a total length of 730 km. We also calculate the traffic load on each span based on the routing table in Fig. 10(h), and the result is shown in Fig. 10(i). Compared with

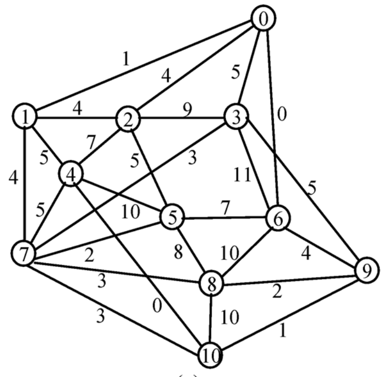
Span	Cost	Span	Cost	Span	Cost
(0, 1)	1310	(2, 5)	390	(5, 8)	350
(0, 2)	760	(3, 6)	340	(6, 8)	565
(0, 3)	390	(3, 7)	1090	(6, 9)	320
(0, 6)	740	(3, 9)	660	(7, 8)	600
(1, 2)	550	(4, 5)	220	(7, 10)	820
(1, 4)	390	(4, 7)	300	(8, 9)	730
(1, 7)	450	(4, 10)	930	(8, 10)	320
(2, 3)	660	(5, 6)	730	(9, 10)	820
(2, 4)	210	(5, 7)	400		

0:	Copenhagen
1:	London
2:	Amsterdam
3:	Berlin
4:	Brussels
5:	Luxembourg
6:	Prague
7:	Paris
8:	Zürich
9:	Vienna
10:	Milan

(a)

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ & 0 & 2 & 2 & 1 & 1 & 1 & 3 & 1 & 1 & 1 \\ & & 0 & 2 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \\ & & & 0 & 2 & 1 & 1 & 3 & 3 & 3 & 3 \\ & & & & 0 & 1 & 1 & 2 & 2 & 1 & 1 \\ & & & & & 0 & 1 & 1 & 1 & 1 & 1 \\ & & & & & & 0 & 1 & 1 & 1 & 1 \\ & & & & & & & 0 & 2 & 1 & 2 \\ & & & & & & & & 0 & 1 & 2 \\ & & & & & & & & & 0 & 1 \\ & & & & & & & & & & 0 \end{bmatrix}$$

(b)



(c)

J=7	RC	FC	CE (α=0.01)
Running time in seconds	7225.45	958.54	34.68
Total cost	32450	32570	32760
Gap to optimality	4.93%	3.84%	4.81%

(d)

SCP solution based on CE with 7 p-cycles	
0–3–2–4–10–8–5–6–0	0–6–9–10–8–5–4–7–1–2–3–0
0–3–9–10–4–2–5–8–6–0	4–5–8–10–4
0–3–9–10–4–2–5–8–6–0	∗ 0–1–4–2–5–8–7–10–9–6–3–0
1–2–3–9–6–5–4–10–8–7–1	

(e)

SCP solution based on CE with a p-cycle length limit of 5800 km	
0–3–6–8–5–2–4–10–7–1–0	4–5–8–6–9–10–4
2–3–9–6–5–8–10–4–2	0–2–4–10–8–5–6–9–3–0
0–3–2–4–5–8–10–9–6–0	0–3–2–1–4–5–7–10–8–6–0
0–3–9–10–7–1–2–4–5–8–6–0	
J=7 α=0.01	Total cost: 32670
Running time: 71.75 seconds	Gap to optimality: 4.84%

(f)

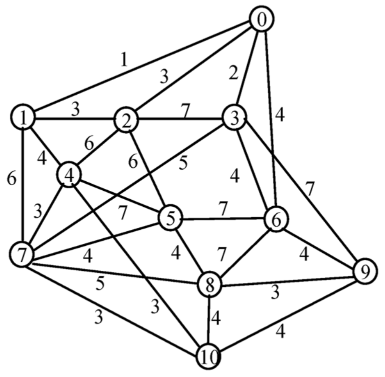
p-cycles in the JCP solution
0–1–2–4–7–5–8–10–9–6–3–0
0–2–1–4–7–5–8–10–9–6–3–0
1–2–3–6–9–10–8–5–7–4–1
1–2–5–8–10–9–6–3–7–4–1

J=7 α=0.01
Total cost: 83245
Running time: 18.26 seconds
Gap to optimality: 3.05%

(g)

Demand	Routes	Load	Demand	Routes	Load	Demand	Routes	Load
d_{01}=1	0→1	1	$d_{1\,10}$=1	1→4→10	1	d_{47}=2	4→7	2
d_{02}=1	0→2	1	d_{23}=2	2→3	2	d_{48}=2	4→5→8	2
d_{03}=1	0→3	1	d_{24}=1	2→4	1	d_{49}=1	4→5→6→9	1
d_{04}=1	0→2→4	1	d_{25}=1	2→5	1	$d_{4\,10}$=1	4→10	1
d_{05}=1	0→2→5	1	d_{26}=1	2→5→6	1	d_{56}=1	5→6	1
d_{06}=1	0→6	1	d_{27}=2	2→4→7	1	d_{57}=1	5→7	1
d_{07}=1	0→3→7	1		2→5→7	1	d_{58}=1	5→8	1
d_{08}=1	0→6→8	1	d_{28}=1	2→5→8	1	d_{59}=1	5→6→9	1
d_{09}=1	0→6→9	1	d_{29}=1	2→3→9	1	$d_{5\,10}$=1	5→7→10	1
$d_{0\,10}$=1	0→6→8→10	1	$d_{2\,10}$=1	2→4→10	1	d_{67}=1	6→5→7	1
d_{12}=2	1→2	2	d_{34}=2	3→2→4	2	d_{68}=1	6→8	1
d_{13}=2	1→2→3	1	d_{35}=1	3→2→5	1	d_{69}=1	6→9	1
	1→7→3	1	d_{36}=1	3→6	1	$d_{6\,10}$=1	6→8→10	1
d_{14}=1	1→4	1	d_{37}=3	3→7	3	d_{78}=2	7→8	2
d_{15}=1	1→4→5	1	d_{38}=3	3→6→8	3	d_{79}=1	7→8→9	1
d_{16}=1	1→4→5→6	1	d_{39}=3	3→9	3	$d_{7\,10}$=2	7→10	2
d_{17}=3	1→7	3	$d_{3\,10}$=3	3→9→10	3	d_{89}=1	8→9	1
d_{18}=1	1→7→8	1	d_{45}=1	4→5	1	$d_{8\,10}$=2	8→10	2
d_{19}=1	1→7→8→9	1	d_{46}=1	4→5→6	1	$d_{9\,10}$=1	9→10	1

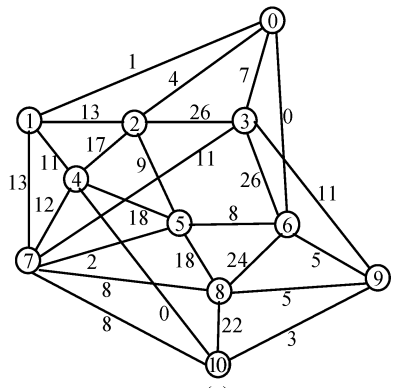
(h)



(i)

Fig. 10. A case study based on the pan European COST 239 network with 11 nodes and 26 spans (10 Gbps per wavelength). (a) Span costs in kilometers. (b) Traffic matrix. (c) Span load under shortest path routing. (d) Comparison among RC, FC , and CE based on (c). (e) SCP solution based on CE for (c) ($J = 7, \alpha = 0.01$). (f) SCP solution based on CE for (c) with a p-cycle length limit of 5800 km. (g) p-cycles in the JCP solution. (h) Routing table in the JCP solution. (i) Span load in the JCP scenario.

Fig. 10(c), the traffic loads on the spans are better balanced in Fig. 10(i). This explains why the JCP solution only requires 4 p-cycles [Fig. 10(g)], in contrast with 7 p-cycles in the SCP solution [Fig. 10(e)]. Generally, JCP routing can achieve a better load balancing than shortest path routing. As a result, less number of p-cycles is required in the JCP solution. Recall that in JCP, we have proposed to try shortest path routing to estimate l_{uv} before calculating the required value of J using (37). Our above analysis validates this approach because the required value of J in JCP is generally not larger than that in SCP. Based on Fig. 10(a), (c), and (e), the total capacity required in the SCP solution (without p-cycle length limit) is 93 760 (spare capacity 32 760 plus working capacity 61 000). We can see that the JCP solution with a total capacity of 83 245 in Fig. 10(g) achieves a capacity saving of 11.21% over the SCP solution.



(a)

J=15	RC	FC	CE (α=0.01)
Time in seconds	18001.93	8948.91	879.85
Total cost	80370	77160	76650
Gap to optimality	39.38%	4.85%	4.96%

(b)

SCP solution based on CE		
0–2–5–4–1–7–10–8–6–9–3–0	1–2–4–5–6–3–9–10–8–7–1	0–2–1–4–5–8–7–10–9–6–3–0
0–2–5–8–10–9–6–3–0	0–3–9–10–4–2–5–8–6–0	0–2–4–1–7–10–8–5–6–9–3–0
1–2–5–8–6–3–9–10–7–4–1	0–3–2–1–4–5–8–7–10–9–6–0	0–3–9–10–4–2–5–8–6–0
2–3–9–6–8–5–2	0–3–9–10–4–2–5–8–6–0	0–2–4–1–7–10–8–5–6–9–3–0
0–3–2–5–8–7–10–9–6–0	0–3–2–5–4–1–7–8–10–9–6–0	0–1–2–4–10–7–8–5–6–9–3–0

(c)

Fig. 11. SCP solutions for the pan-European COST 239 network with span costs in Fig. 10(a) (2.5 Gbps per wavelength).

J=50
Time in seconds: 2679.35
Total cost: 532
Gap to optimality: 4.82%

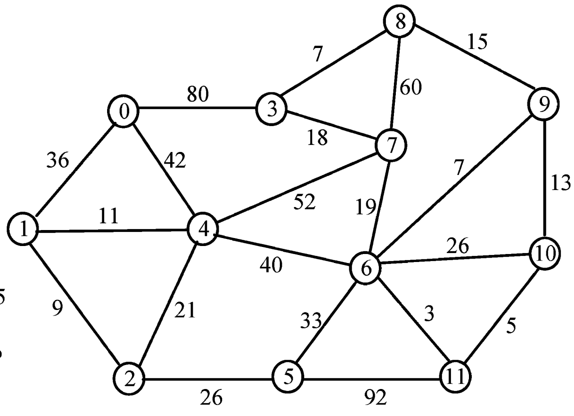
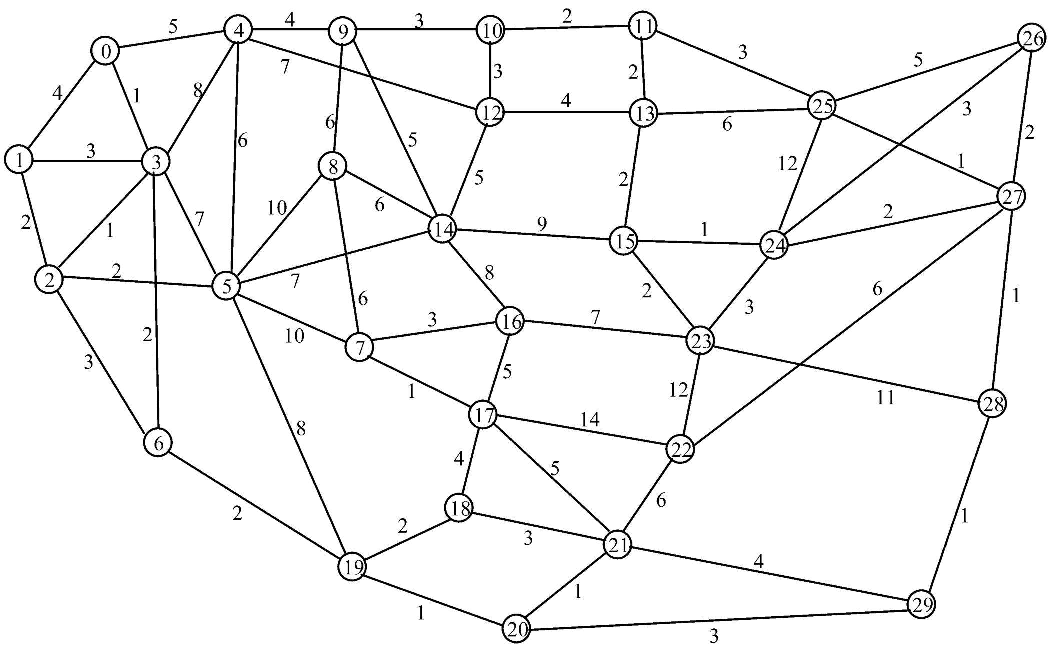


Fig. 12. SCP solution based on CE with $J = 50$ and span cost $c_{uv} = 1$ for $(u, v) \in \boldsymbol{E}$.

In the pan-European COST 239 network, if we use 2.5 Gbps instead of 10 Gbps to divide the traffic matrix in [32] and then route the demands according to shortest path routing, we can get the traffic load on each span as shown in Fig. 11(a). Compared with Fig. 10(c), the number of traffic units on each span is larger due to the finer wavelength granularity. So, we set J to a larger value of 15 according to (37). Fig. 11(b) compares the running time of the recursion, flow conservation and cycle exclusion based ILPs, and Fig. 11(c) shows the p-cycles generated by the cycle exclusion-based ILP. Again, the cycle exclusion-based ILP runs much faster than the other two. Note that the recursion-based ILP is terminated after 5 h (by setting 18 000 $\rightarrow$ timelimit in CPLEX). Comparing Fig. 11(b) with Fig. 10(d), we can see that the running time increases with J for each ILP. To check the efficiency of our cycle exclusion-based ILP for large J values, we consider the network in Fig. 12 with $J = 50$. We can see that a good SCP solution can still be returned in a reasonable running time.

We also compare the cycle exclusion-based ILP with the conventional ILP [1] with candidate cycle enumeration. Cycles are enumerated using the algorithm in [27]. For small size networks, cycles can be enumerated in a short time, and the size of the candidate set is not too large. For example, the pan-European COST 239 network in Fig. 10(c) contains 3531 distinct cycles, which can be found in less than 2 s. As network size increases, cycle enumeration needs a long time and the size of the candidate set soars exponentially. For the randomly generated network in Fig. 13 (with 30 nodes and 62 spans), we find 13 343 782 cycles



	SCP solution based on CE
	0–1–2–5–4–9–10–11–25–26–24–23–15–13–12–14–8–7–16–17–21–22–27–28–29–20–19–6–3–0
J=7, α=0.01	0–1–2–5–4–9–10–11–25–27–28–29–21–22–23–24–15–13–12–14–8–7–17–18–19–6–3–0
Time in seconds: 31520.35	3–5–14–15–23–24–26–25–13–12–4–9–8–7–16–17–18–21–22–27–28–29–20–19–6–3
Total cost: 145	0–3–1–2–5–19–20–29–28–27–22–21–17–7–16–23–15–24–26–25–13–12–14–8–9–4–0
Gap to optimality: 4.83%	7–8–9–10–11–25–26–24–23–22–27–28–29–21–17–7
	5–14–8–7–17–16–23–15–24–26–25–27–22–21–18–19–5
	16–17–21–22–27–28–23–16

Fig. 13. SCP solution for a randomly generated large-sized network using CE.

in 16 420 s (4.56 h). Though cycle enumeration is still manageable, it is infeasible for the conventional ILP [1] to handle so many cycles. In contrast, our cycle exclusion-based ILP (with $J = 7$ and $\alpha = 0.01$) can return an SCP solution with a gap to optimality of 4.83% in 31 520.35 s (8.76 h).

It should be noted that we do not compare our ILPs to the conventional ILP [1] with candidate cycle preselection [18]–[21], or other heuristic algorithms (e.g., CIDA [19]), due to multiple reasons as follows. First, we focus on an optimal ILP model, whereas those approaches are heuristic-based. As a result, the solution quality of those approaches is not guaranteed, whereas our ILPs always ensure an explicit gap to the true optimality in the solution; Second, such a comparison is meaningful in evaluating the performance of those heuristics, which is out of the scope of this paper. In fact, such heuristic performance evaluation has been investigated in [19] and [21] based on some small-size networks. It is observed that the solution quality obtained from those heuristics is generally 10%–20% worse than the optimal solution, or a close-to-optimal solution can be achieved if 20%–40% of all cycles are preselected (in a large-size network as the one in Fig. 13, this still introduces a huge number of ILP variables); Third, the performance evaluation in [19], [21] is obtained based on some small-size networks. As network size increases, the solution quality of the heuristic-based approaches intends to be even worse. For example, Grow algorithm [19] preselects $O(|\boldsymbol{E}|^2 \times |\boldsymbol{V}|)$ cycles, but the total number of cycles in the network increases exponentially with network size (i.e., grows faster than $O(|\boldsymbol{E}|^2 \times |\boldsymbol{V}|)$). As network size increases, the ratio of the number of preselected cycles to the total number of all cycles decreases rapidly, leading to a poorer solution quality (than 10%–20% gap-to-optimality); Finally, even if we use Grow algorithm [19] to preselect candidate cycles, for the network in Fig. 13, the number of candidate cycles still reaches a magnitude of $|\boldsymbol{E}|^2 \times |\boldsymbol{V}| = 62^2 \times 30 = 115320$. This is only 0.86% of all 13 343 782 cycles, but still introduces 115 320 variables which could be hard to be handled by the ILP. In contrast, our cycle exclusion-based ILP for SCP (with $J = 7$) only involves $3J(|\boldsymbol{E}| + |\boldsymbol{V}|) = 1932$ variables and $4J|\boldsymbol{E}| + 2J|\boldsymbol{V}| + |\boldsymbol{E}| + J = 2225$ constraints. Though the number of constraints in our ILP is larger than $|\boldsymbol{E}| = 62$ in the conventional approach, the overall problem size of our ILP is still much smaller.

V. Conclusion

We focused on optimal p-cycle design without candidate cycle enumeration and preselection. Three ILPs were first formulated for solving the SCP problem, based on recursion, flow conservation and cycle exclusion, respectively. We showed that the cycle exclusion-based ILP is the most efficient one. The number of ILP variables and constraints involved in the cycle exclusion-based ILP only increases linearly with network size.

This is a great advantage over the conventional ILP with candidate cycle enumeration, where the number of ILP variables increases exponentially with network size. We also formulated another ILP for solving the JCP problem by extending our cycle exclusion approach. Numerical results showed that our ILPs are very efficient for p-cycle design in various WDM networks.

REFERENCES

[1] W. D. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration," in *Proc. IEEE ICC* , Jun. 1998, vol. 1, pp. 537–543.

[2] D. A. A. Mello, D. A. Schupke, and H. Waldman, "A matrix-based analytical approach to connection unavailability estimation in shared backup path protection," *IEEE Commun. Lett.*, vol. 9, no. 9, pp. 844–846, Sep. 2005.

[3] H. Liu and F. A. Tobagi, "Traffic grooming in WDM/SONET BLSR rings with multiple line speeds," in *Proc. IEEE GLOBECOM*, Dec. 2005, vol. 4, pp. 2096–2101.

[4] D. Stamatelakis and W. D. Grover, "Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles ("p-cycles")," *IEEE Trans. Commun.*, vol. 48, no. 8, pp. 1262–1265, Aug. 2000.

[5] D. Stamatelakis and W. D. Grover, "IP layer restoration and network planning based on virtual protection cycles," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1938–1949, Oct. 2000.

[6] A. Kodian, W. D. Grover, J. Slevinsky, and D. Moore, "Ring-mining to p-cycles as a target architecture: Riding demand growth into network efficiency," in *Proc. 19th Annu. NFOEC*, Sep. 2003, pp. 1543–1552.

[7] G. X. Shen and W. D. Grover, "Design of protected working capacity envelopes based on p-cycles: An alternative framework for survivable automated lightpath provisioning," in *Performance Evaluation and Planning Methods for the Next Generation Internet*, A. Girard, B. Sansò, and F. Vazquez-Abad, Eds. Norwell, MA: Kluwer, 2005.

[8] G. X. Shen and W. D. Grover, "Performance of protected working capacity envelopes based on p-cycles: Fast, simple, and scalable dynamic service provisioning of survivable services," in *Proc. APOC*, Nov. 2004, vol. 5626, pp. 519–533.

[9] D. A. Schupke, C. G. Gruber, and A. Autenrieth, "Optimal configuration of p-cycles in WDM network," in *Proc. IEEE ICC* , May 2002, vol. 5, pp. 2761–2765.

[10] W. S. He, J. Fang, and A. K. Somani, "A p-cycle based survivable design for dynamic traffic in WDM networks," in *Proc. IEEE GLOBECOM*, Dec. 2005, vol. 4, pp. 1869–1873.

[11] H. Huang and J. A. Copeland, "A series of Hamiltonian cycle-based solutions to provide simple and scalable mesh optical network resilience," *IEEE Commun. Mag.*, vol. 40, no. 11, pp. 46–51, Nov. 2002.

[12] D. A. Schupke, "Multiple failure survivability in WDM networks with p-cycles," in *Proc. Circuits Syst., Int. Symp. ISCAS '03*, May 2003, vol. 3, pp. 866–869.

[13] D. A. Schupke, "The tradeoff between the number of deployed p-cycles and the survivability to dual fiber duct failures," in *Proc. IEEE ICC*, May 2003, vol. 2, pp. 1428–1432.

[14] A. Kodian, A. Sack, and W. D. Grover, " p-cycle network design with hop limits and circumference limits," in *Proc. 1st Int. Conf. Broadband Netw. BroadNets*, 2004, pp. 244–253.

[15] W. Li, J. Doucette, and M. Zuo, "p-cycle network design for specified minimum dual-failure restorability," in *Proc. IEEE ICC*, Jun. 2007, pp. 2204–2210.

[16] D. A. Schupke, "Analysis of p-cycle capacity in WDM networks," *Photon. Netw. Commun.*, vol. 12, no. 1, pp. 41–51, Jul. 2006.

[17] H. N. Nguyen, D. Habibi, V. Q. Phung, S. Lachowicz, K. Lo, and B. Kang, "Joint optimization in capacity design of networks with p-cycle using the fundamental cycle set," in *Proc. IEEE GLOBECOM*, Nov. 2006, pp. 1–5, QRP02-5.

[18] W. D. Grover and J. Doucette, "Advances in optical network design with p-cycles: Joint optimization and pre-selection of candidate p-cycles," in *Proc. IEEE LEOS Summer Topic.*, Jul. 2002, pp. 49–50.

[19] J. Doucette, D. He, W. D. Grover, and O. Yang, "Algorithmic approaches for efficient enumeration of candidate p-cycles and capacitated p-cycle network design," in *Proc. 4th Int. Workshop DRCN*, Oct. 2003, pp. 212–220.

[20] H. X. Zhang and O. Yang, "Finding protection cycles in DWDM networks," in *Proc. IEEE ICC*, May 2002, vol. 5, pp. 2756–2760.

[21] C. Liu and L. Ruan, "Finding good candidate cycles for efficient p-cycle network design," in *Proc. IEEE ICCCN*, 2004, pp. 321–326.

[22] D. A. Schupke, "An ILP for optimal p-cycle selection without cycle enumeration," in *Proc. 8th Conf. ONDM*, 2004.

[23] A. Sack and W. D. Grover, "Hamiltonian p-cycles for fiber-level protection in homogeneous and semi-homogeneous optical networks," *IEEE Netw.*, vol. 18, no. 2, pp. 49–56, Apr. 2004.

[24] A. Kodian and W. D. Grover, "Failure-independent path-protecting p-cycles: Efficient and simple fully preconnected optical-path protection," *J. Lightw. Technol.*, vol. 23, no. 10, pp. 3241–3259, Oct. 2005.

[25] G. X. Shen and W. D. Grover, "Extending the p-cycle concept to path segment protection for span and node failure recovery," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 8, pp. 1306–1319, Oct. 2003.

[26] W. D. Grover, *Mesh-Based Survivable Networks: Options and Strategies for Pptical, MPLS, SONET and ATM Networking*. Englewood Cliffs, NJ: Prentice-Hall PTR, 2003.

[27] B. J. Donald, "Finding all the elementary circuits of a directed graph," *SIAM J. Comput.*, vol. 4, no. 1, pp. 77–84, Mar. 1975.

[28] B. Jaumard, C. Rocha, D. Baloukov, and W. Grover, "A column generation approach for design of networks using path-protecting p-cycles," in *Proc. 6th Int. Workshop DRCN*, Oct. 2007.

[29] C. Rocha and B. Jaumard, "Revisiting p-cycles/FIPP p-cycles vs. shared link/path protection," in *Proc. 17th ICCCN*, Aug. 2008, pp. 1–6.

[30] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1993.

[31] CPLEX Solver [Online]. Available: www.ilog.com

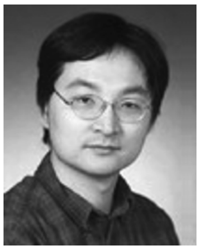
[32] P. Batchelor *et al.*, "Ultra high capacity optical transmission networks," Final rep. action COST 239, 1999.

Bin Wu (S'04–M'07) received the B.Eng. degree in electrical engineering from Zhe Jiang University, Hangzhou, China, in 1993, the M.Eng. degree in communication and information engineering from University of Electronic Science and Technology of China, Chengdu, China, in 1996, and the Ph.D. degree in electrical and electronic engineering from The University of Hong Kong, Pokfulam, Hong Kong, in 2007.

From 1996 to 2001, he served as the Department Manager of TI-Huawei DSP co-lab at Huawei Tech. Co. Ltd., Shenzhen, China. He is currently a Post-Doctoral Fellow at University of Waterloo, Waterloo, ON, Canada, where he is involved in optical and wireless networking research.

Kwan L. Yeung (S'93–M'95–SM'99) received the B.Eng. and Ph.D. degrees in information engineering from The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, in 1992 and 1995, respectively.

He joined the Department of Electrical and Electronic Engineering, The University of Hong Kong, in July 2000, where he is currently an Associate Professor. His research interests include next-generation Internet, packet switch/router design, all-optical networks, and wireless data networks.



Pin-Han Ho received the B.Sc. and M.Sc. degrees from the Electrical and Computer Engineering Department, National Taiwan University, Taipei, Taiwan, in 1993 and 1995, respectively, and the Ph.D. degree from Queen's University, Kingston, ON, Canada, in 2002, focusing on optical communications systems, survivable networking, and QoS routing problems.

He joined the Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON, Canada, as an Assistant Professor in 2002. He is the author/co-author of more than 100 refereed technical papers and book chapters and the co-author of a book on optical networking and survivability.

Dr. Ho is the recipient of the Distinguished Research Excellence Award in the ECE Department, University of Waterloo, the Early Researcher Award in 2005, the Best Paper Award at SPECTS '02 and the ICC '05 Optical Networking Symposium, and the Outstanding Paper Award in HPSR '02.