

\textbf{p-Cycle Design in Survivable WDM Networks with Shared Risk Link Groups (SRLGs)}

Chang Liu and Lu Ruan  
Department of Computer Science  
Iowa State University, Ames, IA 50011  
email: \{liuc, ruan\}@cs.iastate.edu

\section*{Abstract}

\textit{p-Cycle} survivable network design under the single link failure assumption has been studied extensively. Shared risk link group (SRLG) is a concept that better reflects the nature of network failures. An SRLG is a set of links that may fail simultaneously because of the common risk they share. The capability of dealing with SRLG failures is essential to network survivability. In this paper, we extend the \textit{p-cycle} survivable network design from the single link failure model to the single SRLG failure model. An integer linear programming (ILP) formulation that minimizes spare capacity requirement is provided. To avoid enumerating all cycles of a network, we also provide a polynomial-time algorithm to generate a basic candidate \textit{p-cycle} set that guarantees 100\% restorability in case of any single SRLG failure given enough spare capacities. Moreover, we present the SRLG failure detection problem that prevents fast restoration upon an SRLG failure. To solve this problem, we introduce the concept of \textit{SRLG-independent restorability}, which enables the restoration of each link in a failed SRLG to start immediately without knowing which SRLG has failed. We present an approach to optimal \textit{p-cycle} design with SRLG-independent restorability and show that it is NP-hard to generate a candidate \textit{p-cycle} set such that each link can be SRLG-independently restored by at least one cycle in the set.

\textit{keywords:} Network survivability, \textit{p-cycle}, shared risk link group (SRLG).

\section{Introduction}

\textit{p-Cycle} is a promising approach for survivable WDM network design because of its ability to achieve ring-link recovery speed while maintaining the capacity efficiency of a mesh-restorable network [1]. A \textit{p-cycle} is a pre-configured cycle formed out of the spare capacity in the network, which occupies one unit of spare capacity on each on-cycle span. Like a self-healing ring, a \textit{p-cycle} provides one restoration path for every on-cycle span; unlike a self-healing ring, a \textit{p-cycle} also provides two restoration paths for every \textit{straddling span} – a span whose two end nodes are on the cycle but itself is not on the cycle. As shown in Fig. 1, \(a - b - c - d - a\) is a \textit{p-cycle}. For the on-cycle span \(a - b\), the \textit{p-cycle} provides one restoration path \(a - d - c - b\). For the \textit{straddling span} \(a - c\), the \textit{p-cycle} provides two restoration paths: \(a - b - c\) and \(a - d - c\). Thus, a \textit{p-cycle} can protect one unit of working capacity on every on-cycle span and protect two units of working capacity on every straddling span.

The problem of finding a set of \textit{p-cycles} to protect given working capacities in a network such that the total cost of spare capacity required to achieve 100\% restoration is minimized under the single failure assumption has received much study recently. An integer linear programming (ILP) formulation for this optimization problem was provided in [1]. In this approach, a set of candidate \textit{p-cycles} is precomputed and supplied to the ILP to find the optimal set of \textit{p-cycles} out of the candidate \textit{p-cycles}. The ILP will give the optimal solution when the candidate \textit{p-cycle} set includes all cycles in the network. However, since the number of cycles in a network grows exponentially with the network size, various methods have been proposed to reduce the size of the candidate \textit{p-cycle} set. One method is to limit the maximum length of the candidate \textit{p-cycles} [1][2]. Another method is to select a certain number of candidate cycles from all cycles according to the \textit{a priori efficiency} metric pre-computed for each cycle [3]. While both methods can reduce the number of candidate \textit{p-cycles}, they still require the enumeration of all cycles in the network. To address this problem, an algorithm called SLA was proposed in [4]. The idea is to generate one candidate \textit{p-cycle} for each span in the network so that the span is a straddling span of the \textit{p-cycle}. In [5], three algorithms (SP-Add, Expand, and Grow) were provided to find more efficient candidate \textit{p-cycles} based on those generated by SLA. In [6], a weighted DFS-based cycle search heuristic was proposed to find a candidate \textit{p-cycle} set of size \(O(m)\) (\(m\) is the number of links in the network) that helps to achieve solutions close to the optimal ones derived from all cycles. To circumvent the inherent hardness of ILP, a heuristic algorithm called CIDA was

Shared Risk Link Groups (SRLGs)
Fig. 1. A p-cycle example

proposed in [5]. In CIDA, a p-cycle is chosen from candidate p-cycles according to an efficiency score and placed in the network to reduce the unprotected working capacity iteratively until all working capacities are protected. Some recent works [7][8][9] explored p-cycle design for dual link failures.

In this paper, we consider single-SRLG failure model, which is more general than the widely studied single-link failure model. An SRLG (Shared Risk Link Group) is a set of links that share a common resource whose failure will cause the failure of all links in it [10]. For instance, multiple fiber links laid out in a common conduit in WDM networks can be viewed as an SRLG because the conduit cut will result in the failure of all fibers in it. In general, a network contains a set of SRLGs that can be pre-determined according to the resource sharing relationship. An example SRLG set for the network in Fig. 1 may be \{ab, \{ac, ad, ae\}, \{bc, ac\}, \{ad\}, \{ad, de\}\}. If the SRLG \{ac, ad, ae\} fails, the links ac, ad, and ae all fail. A well-studied path-based protection scheme for the single-SRLG failure model is SRLG-diverse routing, which finds a pair of SRLG-disjoint paths (primary path and backup path) between the source and destination nodes of a connection. SRLG-disjoint means that no single SRLG failure will break both the primary and the backup paths simultaneously. The NP-completeness proof of the SRLG-diverse routing problem as well as an ILP solution are given in [11][12]. Heuristic approaches to this problem are proposed in [13][14]. In this paper, we will discuss a new protection strategy for the single-SRLG failure model, that is, to apply p-cycle network design to networks with SRLGs.

The rest of the paper is organized as follows. In section II, we study the protection a p-cycle can offer upon an SRLG failure. In section III, we describe the p-cycle design problem and give an ILP formulation that solves it optimally. In section IV, we provide an algorithm for generating a subset of all cycles that can protect any single SRLG failure. In section V, we introduce the concept of SRLG-independent restorability to address the SRLG failure detection issue that affects fast restoration. In section VI, we present some simulation results on two networks with randomly generated SRLG sets. Finally, a conclusion is given in section VII.

II. p-CYCLE PROTECTION UPON AN SRLG FAILURE

In the single link failure model, the protection that can be provided by a p-cycle for a link depends on their relationship. Specifically, if the link is on-cycle, then the p-cycle can offer one restoration path in case the link fails; if the link straddles the p-cycle, then two restoration paths are provided by the p-cycle when the link is broken; otherwise the p-cycle cannot protect the link. Upon an SRLG failure, all links in the SRLG are gone. To restore such a failure, every failed link must be taken care of by some p-cycles. Meanwhile, since multiple links may fail in case of an SRLG failure, if two or more failed links happen to be on the same p-cycle, then the p-cycle is broken, which makes the situation more complicated than in the single link failure model. Fig. 2 illustrates the possible relationships between an SRLG failure and a p-cycle. Part (a) shows the case where the p-cycle remains a cycle after the SRLG failure. Two restoration paths can be provided for links s – t, u – v, and t – x respectively given enough copies of the p-cycle. However, no restoration path can be provided for link x – y since y is not on the p-cycle. In part (b), the p-cycle is broken into a path from s to t after the SRLG failure. One restoration
(a). An SRLG failure does not break a $p$-cycle.

(b). An SRLG failure breaks a $p$-cycle into a path.

(c). An SRLG failure breaks a $p$-cycle into segments.

Fig. 2. Relationship between an SRLG failure and a $p$-cycle (Solid lines are links in the failed SRLG. Dashed-line ellipses represent $p$-cycles.)

path can be provided for links $s - t$, $u - v$, and $t - x$ respectively given enough copies of the $p$-cycle. The $p$-cycle provides no restoration path for link $y - z$ because $y$ and $z$ are not on the $p$-cycle. In part (c), the $p$-cycle is broken into two segments $s - u$ and $v - t$. No restoration path is available for links $s - t$, $u - v$, and $x - z$ since their two end nodes are on different segments. However, the $p$-cycle can supply one restoration path for link $x - y$ since both $x$ and $y$ are on the segment $v - t$. To compute the protection a $p$-cycle can provide for a link upon an SRLG failure, we define a function called CYCLE\_LINK\_SRLG. CYCLE\_LINK\_SRLG takes a cycle $i$, a link $j$, and an SRLG $k$ as inputs, and returns the number of restoration paths that can be provided for link $j$ by cycle $i$ in case of the failure of SRLG $k$. The pseudocode of function CYCLE\_LINK\_SRLG is given below.

```c
int CYCLE\_LINK\_SRLG(cycle $i$, link $j$, SRLG $k$)
1. if $j \notin k$ then
2. return 0; // The link does not belong to the SRLG and therefore does not fail.
3. if link $j$’s end nodes are not both on cycle $i$ then
4. return 0; // The link cannot be protected by the cycle.
5. Remove links in SRLG $k$ from cycle $i$.
6. if cycle $i$ remains a cycle then
7. return 2;
8. else // Cycle $i$ is broken into one or more segments.
9. if link $j$’s end nodes are on the same segment then
10. return 1;
11. else
12. return 0;
```

III. AN ILP FOR OPTIMAL $p$-CYCLE DESIGN

A. Problem Description

We consider the following $p$-Cycle design problem: given a network represented by a graph $G = (V, L)$, a set of SRLGs in $G$, a set of distinct candidate $p$-cycles in $G$, and the working capacity on each link in $G$, compute a set of $p$-cycles that minimizes the total cost of spare capacity required to achieve 100% restoration in case of a single SRLG failure.

To guarantee the existence of a solution to this problem, the following conditions are assumed:

- The network is 2-edge-connected so that each link can be protected by at least one cycle.
- The failure of any single SRLG does not disconnect the network.
• In case of any single SRLG failure, for each link in this SRLG, at least one cycle exists in the candidate $p$-cycle set such that the cycle can provide at least one restoration path for it.
• There is enough capacity on each link in the network.

B. ILP Formulation

Sets: (input)
$L$: The set of all links.
$P$: The set of candidate $p$-cycles.
$R$: The set of SRLGs.

Parameters: (input or pre-computed)
$w_j$: The working capacity on link $j$.
$c_j$: The cost of one unit of spare capacity on link $j$.
$p_{ij}$: 1 if link $j$ is on cycle $i$, 0 otherwise.
$b_{jk}$: 1 if link $j$ is in SRLG $k$, 0 otherwise.
$x_{ijk}$: The number of restoration paths for link $j$ that can be provided by cycle $i$ in case SRLG $k$ fails. This value can take 0, 1, or 2, which is pre-computed by the function CYCLE_LINK_SRLG($i,j,k$).

Variables: (to be determined)
s$_j$: The spare capacity on link $j$.
n$_i$: The number of copies of cycle $i$ needed in the $p$-cycle design.
n$_{ik}$: The number of copies of cycle $i$ needed in case SRLG $k$ fails.
n$_{ijk}$: The number of copies of cycle $i$ needed for link $j$ in case SRLG $k$ fails.

Minimize $\sum_{j \in L} c_j \cdot s_j$

Subject to:

\[ s_j = \sum_{i \in P} p_{ij} \cdot n_i \quad \forall j \in L \]  
(1)

\[ b_{jk} \cdot w_j \leq \sum_{i \in P} x_{ijk} \cdot n_{ijk} \quad \forall j \in L, \forall k \in R \]  
(2)

\[ n_{ik} = \sum_{j \in L} b_{jk} \cdot n_{ijk} \quad \forall i \in P, \forall k \in R \]  
(3)

\[ n_i \geq n_{ik} \quad \forall i \in P, \forall k \in R \]  
(4)

Constraints in (1) reflect the relationship between the spare capacities and the result $p$-cycle design. Specifically, the spare capacity on link $j$ will be the total number of $p$-cycles that traverse it. Constraints in (2) guarantee that if link $j$ is in SRLG $k$ (i.e., $b_{jk} = 1$), its working capacity is protected in case SRLG $k$ fails. And if link $j$ is not affected by SRLG $k$’s failure (i.e., $b_{jk} = 0$), this constraint can be ignored since the left hand side is 0. This enforces that when an SRLG fails, only the links that belong to it need to have their working capacities restored by $p$-cycles. Constraints in (3) ensure that when SRLG $k$ fails, all links in $k$ should be restored. So the number of copies of cycle $i$ needed for SRLG $k$’s failure is the sum of the number of copies of cycle $i$ needed for all links in SRLG $k$. Constraint (4) is equivalent to $n_i = \max_{k \in R} n_{ik}$. This equation reflects the fact that under the single SRLG failure assumption, the number of copies of cycle $i$ needed is dominated by the maximum requirement over all single SRLG failures.

This ILP will be referred to as ILP1 in the rest of the paper.
IV. GENERATION OF CANDIDATE CYCLES

As in the single link failure model, the set of candidate \(p\)-cycles must contain all cycles in the network in order for the ILP to obtain the optimal \(p\)-cycle design. This requirement blows up the complexity of the \(p\)-cycle design since the number of cycles in a network grows exponentially with the network size. To overcome this difficulty, we give an algorithm for generating a small subset of all cycles as the candidate \(p\)-cycle set such that a \(p\)-cycle design can be found to fully survive any single SRLG failure in the network given enough spare capacities. The algorithm works as follows. For each SRLG, first remove all links in it from the network graph. Then for each pair of end nodes of a removed link, find its shortest path as well as two node-disjoint shortest paths (if exist) in the remaining graph. The shortest path is combined with the removed link to form a cycle that contains the removed link as on-cycle link, and the two node-disjoint shortest paths (if exist) are combined to form a cycle on which the removed link straddles. The distinct cycles generated are collected into the candidate \(p\)-cycle set. FIND\_BASIC\_CYCLES is a function implementing the above algorithm and the pseudocode of it is given below.

FIND\_BASIC\_CYCLES(network \(G\), SRLG set \(R\))
1. \(P = \emptyset\);
2. for each SRLG \(k \in R\) do
3. \(G' = (V, L - k)\);
4. for each link \(l = (u, v) \in k\) do
5. Compute the shortest path between \(u\) and \(v\) in \(G'\).
6. let \(c = \) the cycle formed by the shortest path and \(l\);
7. let \(P = P \cup \{c\}\);
8. Compute two node-disjoint shortest paths between \(u\) and \(v\) in \(G'\).
9. if such a pair of node-disjoint paths exist then
10. let \(c = \) the cycle formed by the two paths;
11. let \(P = P \cup \{c\}\);
12. return \(P\);

Given a 2-edge-connected network and an SRLG set such that any single SRLG failure does not disconnect the network, the algorithm can always find a candidate \(p\)-cycle set that can provide 100% restorability in case of any single SRLG failure given enough spare capacities. The reason is that in case of any SRLG failure, for each affected link, it is always possible to find a shortest path between the two end nodes of the link because the network is still connected. This guarantees that when an SRLG fails, each link in the SRLG has at least one cycle that can provide a restoration path for it.

For each link \(j \in L\), suppose \(t_j\) is the number of SRLGs that contains \(j\). Let \(t = \max_{j \in L} t_j\). Then the number of distinct cycles generated by FIND\_BASIC\_CYCLES is \(O(tm)\) where \(m\) is the number of links in the network.

V. SRLG-INDEPENDENT RESTORABILITY

A. Impact of SRLG Failure Detection Problem on Restoration Speed

An important feature of \(p\)-cycle survivable network design with the single link failure model is fast restoration. When a link fails, its end nodes can detect the failure immediately and start restoration right away using the pre-configured \(p\)-cycles. But the single SRLG failure model changes the situation. Upon an SRLG failure, for each failed link, although its end nodes can detect the link failure immediately as in the single link failure model, they may not be able to start restoration at that moment. The reason is that in order to figure out which pre-configured \(p\)-cycles should be used to restore the link, the end-nodes of the link need to know which SRLG has failed. Unless this can be inferred directly from the knowledge of the SRLG set, a signaling protocol is needed to enable all involved nodes to find out which SRLG has failed. For example, in Fig. 3, when SRLG \(g_1 = \{ab, ac\}\) fails, node \(c\) can detect the failure of link \(ac\) instantly; however, since \(ac\) also belongs to another SRLG \(g_2\), \(c\) cannot tell whether the failed SRLG is \(g_1\) or \(g_2\) until it gets more failure information from the network.
B. A Solution with SRLG-Independent Restorability

When an SRLG failure happens, for each affected link, we want its end nodes to start the restoration of the failed link before they find out which SRLG is down. To achieve this, we introduce the concept of *SRLG-independent restorability*. The idea can be illustrated by Fig. 4 in which we try to restore the traffic on link $a - c$ right after node $a$ and $c$ detect the failure of link $a - c$. Note that at this moment, node $c$ does not know whether the failed SRLG is $g_1$ or $g_2$. Consider two $p$-cycles, $c_1 = a - b - c - d - e - a$ and $c_2 = a - b - c - a$. It can be seen that $c_1$ can provide the failed link $a - c$ with either one restoration path ($a - e - d - c$) if $g_1$ fails or two restoration paths ($a - b - c$ and $a - e - d - c$) if $g_2$ fails. To accommodate the worst case scenario, we consider that $c_1$ can provide only one restoration path for link $a - c$ and can be used to restore traffic on link $a - c$ regardless whether $g_1$ or $g_2$ has failed. On the other hand, although $c_2$ can be used to restore link $a - c$ in case of $g_2$ failure, it cannot be used to restore link $a - c$ in case of $g_1$ failure. Therefore, we can’t use $c_2$ to restore link $a - c$ immediately after $a$ and $c$ detect the link failure. We say that link $a - c$ is *SRLG-independently restorable* by cycle $c_1$ but not by cycle $c_2$, i.e., $c_1$ can be used to restore link $a - c$ no matter which SRLG that contains $a - c$ has failed.

For convenience, the SRLG-independent restorability for a given a network $G = (V, L)$ with respect to an SRLG set $R$ is referred to as *$R$-independent restorability*. The formal definition of $R$-independent restorability is given as follows.

**Definition:** *$R$-independent restorability parameter* $x_{ij}^r$ is the number of restoration paths for link $j$ that can be provided by cycle $i$ in case of a failure of any SRLG $k \in R$ that contains $j$. And $x_{ij}^r = \min_{k \in R_j} x_{ijk}$ where
A. Settings

Two networks shown in Fig. 5 are used for simulations. Network1 is a 11-node 23-link network taken from the website of [15] and Network2 is a 15-node 28-link Metropolitan network.

For each network, two demand sets are used. One is the uniform demand set that contains one demand for each ununordered source-destination pair. The other is a set that contains 150 random demands for Network1 and 300

B. Hardness of Generating Candidate Cycles with SRLG-Independent Restorability

In Section IV, we give a polynomial time algorithm to generate a small candidate p-cycle set to guarantee 100% restorability so that the enumeration of all cycles is avoided. The algorithm guarantees that when an SRLG fails, each link in the SRLG has at least one candidate cycle that can provide a restoration path for it. A natural question to ask is whether a similar approach can be taken with regard to SRLG-independent restorability, that is, whether each link in the SRLG can be independently restorable by at least one candidate cycle. With a simple modification to the NP-hardness proof for the SRLG-diverse routing problem provided in [12], we can prove that it is NP-hard to generate a cycle that can SRLG-independently restore a link by reduction from the 3-SAT problem. For the complete proof, please refer to the Appendix of this paper. Because of this result, no effort should be spent on finding a polynomial time candidate cycle generation algorithm for a given network and its SRLG set with SRLG-independent restorability consideration.

VI. Numeric Results
random demands for Network2 respectively. In all demand sets, each demand requests for one unit of capacity. For each combination of a test network and a demand set, working capacities on the network links are obtained by routing each demand over the shortest path. The cost of each unit of spare capacity on a link is set to one, i.e., $c_j = 1$ for all $j \in L$.

In practice, SRLG sets are known \textit{a priori}. In our simulations, they are randomly generated. To facilitate this, we define the term “$r$-SRLG set” where $r$ is a positive integer. An $r$-SRLG set has the property that each SRLG in the set contains at most $r$ links. For each network, we randomly generate $r$-SRLG sets ($2 \leq r \leq 4$) conforming to the following rules for each SRLG set:

- Any single SRLG failure does not disconnect the network. This is necessary to guarantee a feasible $p$-cycle design.
- Each link in the network belongs to at least one SRLG. That is, there is no risk-free link in the network.
- Each SRLG is not a subset of another SRLG. Note that as long as an SRLG failure is restorable, a failure of any subset of it is also restorable without requiring more spare capacities.

Note that a network with single link failure has a 1-SRLG set.

All simulations are run on a Sun Ultra 10 workstation equipped with a single 440MHz CPU, 256MB RAM, and 4GB virtual memory. CPLEX8.1 is used as the solver for ILP formulations.

\textbf{B. $p$-Cycle Design without Considering SRLG-Independent Restorability}

To compute the optimal $p$-cycle design, we solve ILP1 for both test networks and all cycles are used as candidate $p$-cycles for ILP1. The number of candidate cycles, total working capacity on all links, and total spare capacity required under different simulation settings are shown in Table I. It can be seen that when a SRLG set has larger SRLGs (i.e., larger $r$ values), the total spare capacity needed becomes larger since more working capacity is affected by a single SRLG failure. On the other hand, this trend seems to be more dramatic for Network2 than for Network1. This can be explained by the fact that the average working capacity per link in Network2 is higher than that in Network1, so the amount of affected working capacity grows faster in Network2 than in Network1 as the size of SRLGs in the SRLG set becomes larger.

To evaluate the effect of using the basic candidate cycle set generated by the algorithm \texttt{FIND BASIC CYCLES} given in section IV as the candidate $p$-cycle set for ILP1, we compare the spare capacity requirement and running time of ILP1 with all cycles and with the basic candidate cycles in Table II and Table III. Table II shows the number of cycles generated for the basic candidate cycle set, the total spare capacity required for ILP1 with all cycles, and the total spare capacity required for ILP1 with the basic candidate cycles for various simulation settings. For ILP1 with basic candidate cycles, we also list the percentage of extra spare capacity required over the optimal solution in the parentheses. Table III gives the running time comparison between ILP1 with all cycles and ILP1 with basic
candidate cycles. In algorithm FIND_BASIC_CYCLES, we used a two-step algorithm to find two node-disjoint shortest paths for a pair of nodes. That is, we find a shortest path first, then remove all intermediate nodes along this path and try to find a shortest path in the remaining graph.

As shown in Table II, the spare capacity obtained by ILP1 with basic candidate cycles is always greater than the optimal value. This is expected because FIND_BASIC_CYCLES generates a small subset of all cycles. We notice that for both test networks, the spare capacity over usage with single link failure (1-SRLG set) is larger than with 2, 3, 4-SRLG sets. In other words, the negative effect of using a small basic candidate cycle set on an /D6-SRLG set (/D6/BQ/BQ/BQ) is less than on the single link failure model. This can be explained by the following two facts. Firstly, under the single link failure model, a link can always be protected by a cycle as long as its end nodes are on the cycle; while for an r-SRLG (r > 1) set, we need an extra requirement that the SRLG failure should not break the cycle. This makes the number of cycles that can potentially protect a link become smaller for an r-SRLG set (r > 1) compared to 1-SRLG set. Secondly, the simulation results show that FIND_BASIC_CYCLES generates more cycles for 2, 3, 4-SRLG sets than for 1-SRLG sets. Hence, the loss of candidate cycles in the basic candidate cycle sets is less significant for 2, 3, 4-SRLG sets than for 1-SRLG sets.

On the other hand, Table III shows that ILP1 with basic candidate cycle set runs significantly faster than ILP1 with all cycles because the basic candidate cycle set contains much fewer cycles and as a result the corresponding ILP formulation has much fewer variables and constraints. It can be seen from Table III that the speedup ranges from 13 to 132 for Network1, and from 60 to 1945 for Network2. Of course, the performance gain in running time is achieved at the cost of sacrificing spare capacity efficiency, as shown in Table II.

C. /D4-Cycle Design with SRLG-Independent Restorability

To evaluate the impact of SRLG-independent restorability on the optimal /D4-cycle design, we also run ILP2 with all cycles as the candidate cycles. The comparison between the total spare capacity required with and without

### Table I

<table>
<thead>
<tr>
<th></th>
<th>Network1</th>
<th></th>
<th>Network2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uniform</td>
<td>150 random</td>
<td>Uniform</td>
<td>300 random</td>
</tr>
<tr>
<td># candidate</td>
<td>307</td>
<td>307</td>
<td>976</td>
<td>976</td>
</tr>
<tr>
<td>cycles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working Capacity</td>
<td>96</td>
<td>244</td>
<td>228</td>
<td>679</td>
</tr>
<tr>
<td>Spare Capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1-SRLG set)</td>
<td>86</td>
<td>214</td>
<td>136</td>
<td>400</td>
</tr>
<tr>
<td>Spare Capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2-SRLG set)</td>
<td>122</td>
<td>302</td>
<td>260</td>
<td>763</td>
</tr>
<tr>
<td>Spare Capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3-SRLG set)</td>
<td>133</td>
<td>315</td>
<td>354</td>
<td>1035</td>
</tr>
<tr>
<td>Spare Capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4-SRLG set)</td>
<td>165</td>
<td>392</td>
<td>388</td>
<td>1174</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Spare Capacity</th>
<th></th>
<th>Network1</th>
<th></th>
<th>Network2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Uniform</td>
<td>150 random</td>
<td>Uniform</td>
<td>300 random</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALL</td>
<td>BASIC</td>
<td>ALL</td>
<td>BASIC</td>
</tr>
<tr>
<td>1-SRLG set (% over opt)</td>
<td>27</td>
<td>86</td>
<td>136 (58.1%)</td>
<td>30</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>2-SRLG set (% over opt)</td>
<td>39</td>
<td>122</td>
<td>175 (43.4%)</td>
<td>41</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>763</td>
</tr>
<tr>
<td>3-SRLG set (% over opt)</td>
<td>33</td>
<td>133</td>
<td>188 (41.4%)</td>
<td>42</td>
<td>354</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1035</td>
</tr>
<tr>
<td>4-SRLG set (% over opt)</td>
<td>38</td>
<td>165</td>
<td>251 (52.1%)</td>
<td>49</td>
<td>388</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1174</td>
</tr>
</tbody>
</table>
TABLE III
RUNNING TIME COMPARISON BETWEEN ALL CYCLES AND BASIC CYCLES

<table>
<thead>
<tr>
<th>Running time (second)</th>
<th># cycles generated in BASIC</th>
<th>Network1</th>
<th>150 random</th>
<th>Network2</th>
<th>300 random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Uniform</td>
<td>BASIC</td>
<td>Uniform</td>
<td>BASIC</td>
</tr>
<tr>
<td>1-SRLG set</td>
<td>27</td>
<td>3.213</td>
<td>0.224</td>
<td>3.470</td>
<td>0.266</td>
</tr>
<tr>
<td>2-SRLG set</td>
<td>39</td>
<td>15.661</td>
<td>0.822</td>
<td>109.94</td>
<td>0.831</td>
</tr>
<tr>
<td>3-SRLG set</td>
<td>33</td>
<td>6.721</td>
<td>0.337</td>
<td>7.786</td>
<td>0.371</td>
</tr>
<tr>
<td>4-SRLG set</td>
<td>38</td>
<td>49.998</td>
<td>0.519</td>
<td>39.734</td>
<td>0.511</td>
</tr>
</tbody>
</table>

TABLE IV
OPTIMAL p-CYCLE DESIGN WITH AND WITHOUT SRLG-INDEPENDENT RESTORABILITY

<table>
<thead>
<tr>
<th>Spare Capacity</th>
<th>Network1</th>
<th></th>
<th>Network2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ILP1</td>
<td>ILP2</td>
<td>ILP1</td>
<td>ILP2</td>
</tr>
<tr>
<td>2-SRLG set</td>
<td>122</td>
<td>135</td>
<td>302</td>
<td>331</td>
</tr>
<tr>
<td>3-SRLG set</td>
<td>133</td>
<td>142</td>
<td>315</td>
<td>333</td>
</tr>
<tr>
<td>4-SRLG set</td>
<td>165</td>
<td>N/A</td>
<td>392</td>
<td>N/A</td>
</tr>
</tbody>
</table>

SRLG-independent restorability (computed by ILP2 and ILP1 respectively) is shown in Table IV. Since for 1-SRLG set (single link failure model), the results are the same no matter SRLG-independent restorability is considered or not, we omit the results for 1-SRLG set in the table.

As expected, the introduction of SRLG-independent restorability results in more spare capacity requirement because $x_{ij}$ is a more restrictive restorability parameter than $x_{ijk}$ is. However, the extra spare capacity required by ILP2 over ILP1 is relatively small – between 5.7% and 10.7% for Network1, and between 12.3% and 16.4% for Network2.

Notice that for 4-SRLG set, ILP2 fails to find a solution for both networks, which means that there is at least one non-SRLG-independently restorable link in both networks. To find out how often such an undesirable situation happens and how many non-SRLG-restorable links exist in each case, we generate 100 r-SRLG (2 ≤ r ≤ 4) set for each network. Corresponding results are shown in Table V in which “bad” means non-SRLG-independently restorable.

As shown in Table V, as r increases, the probability that we run into a non-SRLG-independently restorable situation gets higher for both networks. Meanwhile, in those non-SRLG-independently restorable cases, the number of non-SRLG-independently restorable links is low. Actually, most non-SRLG-independently restorable cases are caused by only one or two non-SRLG-independently restorable links. This means that when an SRLG failure occurs, most links are SRLG-independently restorable and therefore can be restored immediately. Those a few non-SRLG-independently restorable links can be restored after their end nodes find out which SRLG has failed using a signaling protocol.

TABLE V
NON-SRLG-INDEPENDENTLY RESTORABLE CASES

<table>
<thead>
<tr>
<th></th>
<th>Network1</th>
<th></th>
<th>Network2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># bad cases (out of 100)</td>
<td>Avg. # bad links among bad cases</td>
<td># bad cases (out of 100)</td>
<td>Avg. # bad links among bad cases</td>
</tr>
<tr>
<td>r = 2</td>
<td>11</td>
<td>1.18</td>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>r = 3</td>
<td>31</td>
<td>1.19</td>
<td>29</td>
<td>1.14</td>
</tr>
<tr>
<td>r = 4</td>
<td>46</td>
<td>1.61</td>
<td>63</td>
<td>1.87</td>
</tr>
</tbody>
</table>
VII. CONCLUSION

In this paper, we extend the $p$-cycle survivable network design from the single link failure model to the single SRLG failure model. An ILP formulation is provided to compute a $p$-cycle design with minimum spare capacity requirement for an input network, its SRLG set, and its working capacities on the network links such that 100% restorability can be guaranteed in case of any single SRLG failure. To avoid the enumeration of all cycles in the input network, we propose a polynomial time algorithm called $\textsc{FindBasicCycles}$ to generate a basic candidate $p$-cycle set of size $O(tm)$ ($m$ is the number of links in the network and $t = \max_{j \in L} t_j$ where $t_j$ is the number of SRLGs to which link $j$ belongs). Given enough spare capacity, such a candidate $p$-cycle set can be used by the ILP to compute a $p$-cycle design that guarantees 100% restorability. Using the basic candidate $p$-cycle set can significantly reduce the time to compute an ILP solution while compromising the spare capacity optimality of the ILP solution due to the reduced number of candidate cycles. This trade-off is confirmed by our simulation results.

The SRLG failure detection issue undermines fast restoration, which is a key merit of $p$-cycle survivable network design with the single link failure model. We propose the concept of SRLG-independent restorability to solve this problem. The idea is to redefine the restorability parameter such that a broken link can be restored immediately by a $p$-cycle before its end nodes find out which SRLG has failed. We provide a revised ILP to compute an optimal $p$-cycle design with SRLG-independent restorability. Simulation results show that the additional spare capacity required by SRLG-independent restorability is reasonable. Moreover, we prove that it is NP-hard to compute a candidate $p$-cycle set to ensure 100% SRLG-independent restorability in case of any single SRLG failure.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Award Number ANI-0237592.

REFERENCES


APPENDIX

SRLG-independent restorability problem (SRLG-I-R) is defined as follows. For a network $G = (V, L)$, an SRLG set $R \subseteq \mathcal{P}(L)$, and a link $j \in L$, is there a cycle $i$ in $G$ such that $j$ can be SRLG-independently restored by $i$, i.e., both end nodes of $j$ are on cycle $i$ and $\forall k \in R_j, E(p_1) \cap k = \emptyset \lor E(p_2) \cap k = \emptyset$ where $R_j = \{g \in R : j \in g\}$ and
We will prove that 3-SAT \( \leq_p \) SRLG-I-R.

Given a conjunction normal form formula with \( n \) boolean variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses \( C_1, C_2, \ldots, C_m \), each of which is a disjunction of three literals where each literal takes the form of \( x_q \) or \( \overline{x_q} \) (\( 1 \leq q \leq n \)) and the three literals have different subscriptions, we construct an instance of the SRLG-I-R problem as follows.

1) Construct a network.
   - For each variable \( x_q \) (\( 1 \leq q \leq n \)), build a node \( q \). In addition, a node \( 0 \) is added. Add two parallel links labeled \( x_q \) and \( \overline{x_q} \) between nodes \( q-1 \) and \( q \) (\( 1 \leq q \leq n \)).
   - For each clause \( C_h \) (\( 1 \leq h \leq m \)), build a node \( h' \). In addition, a node \( 0' \) is added. For each clause \( C_h = l_1 \lor l_2 \lor l_3 \) (\( 1 \leq h \leq m \)), add three parallel links labeled \( C_h l_1 \), \( C_h l_2 \), and \( C_h l_3 \) between nodes \( (h-1)' \) and \( h' \).
   - Add two extra nodes \( s \) and \( t \). Add links \( s-t, s-0, s-0', n-t, \) and \( m'-t \).

2) Define an SRLG set.
   - For each literal \( l \), define an SRLG that contains \( l \) and \( C_h \overline{l} \) if \( l \) appears in clause \( C_h \) (\( 1 \leq h \leq m \)). In addition, each SRLG contains link \( s-t \).

3) Let \( j \) be \( s-t \), i.e., we need to determine whether link \( s-t \) is SRLG-independently restorable by a cycle in the network.

Fig. 6 shows an example that illustrates the construction. Note that parallel links in the network are treated as different links when they are used to form a cycle. It can be seen that there are three groups of cycles in the constructed network.

(a) Cycles with the form \( s-0-1-2-\cdots-n-t-m'-s \) (\( m'-1 \) – \( 0'-0' \)).
(b) “Top” cycles with the form \( s-0-1-2-\cdots-n-t-s \).
(c) “Bottom” cycles with the form \( s-t-m'-s \) (\( m'-1 \) – \( 0'-0' \)).

It’s easy to verify that no cycle in group (b) or group (c) can provide SRLG-independent restorability for link \( s-t \) because of the way we define the SRLG set. Therefore, only cycles in group (a) can possibly offer SRLG-independent restorability for link \( s-t \). And whether a cycle in group (a) can provide SRLG-independent restorability for link \( s-t \) depends on whether the cycle’s top and bottom paths contain any links that belong to the same SRLG.

We now show that link \( j = s-t \) is SRLG-independently restorable if and only if the given 3-CNF formula is satisfiable.

Suppose \( j \) is SRLG-independently restorable, then there exists a cycle \( i \) in group (a) that can always provide
either its top or bottom path between \( s \) and \( t \) as the restoration path for \( j \) no matter which SRLG fails. We obtain an assignment to the given 3-CNF formula based on cycle \( i \) as follows. If \( i \)’s top path traverses link \( x_q \), then \( x_q \) is assigned value 1. If \( i \)’s top path traverses \( \overline{x_q} \), then \( x_q \) is assigned value 0. Such an assignment must satisfy each clause \( C_h = l_1 \lor l_2 \lor \overline{l_3} \) (1 \( \leq h \leq m \)). The reason is the following. Without loss of generality, assume the bottom path of \( i \) traverses \( C_h l_1 \) between nodes \((h - 1)’ \) and \( h’ \), then \( i \)’s top path cannot traverse \( \overline{l_1} \) because if so, the SRLG-independent restorability would be violated since there is an SRLG that contains both \( l_1 \) and \( \overline{l_1} \) and the failure of this SRLG will break both the top and the bottom paths. Therefore, \( i \)’s top path must traverse \( l_1 \). According to the assignment rule, the literal \( l_1 \) is assigned value 1, which makes the clause \( C_h \) 1. Thus, the given 3-CNF formula is satisfiable.

Suppose the given 3-CNF formula is satisfiable, then it has a satisfying assignment. Based on the assignment, we construct a cycle using the following rules. If \( x_q \) is 1/0 in the assignment, then let the top path of the cycle traverse link \( x_q/\overline{x_q} \) and include all links with labels \( C_h x_q/C_h \overline{x_q} \) into the bottom path. Note that this may result in multiple links for a bottom path segment. If this happens, just arbitrarily pick one link out of them for the segment. Since the assignment satisfies the 3-CNF formula, a valid bottom path (i.e. each segment contains one link) can be built from \( s \) to \( t \). Furthermore, the bottom path together with the top path form a cycle that offers SRLG-independent restorability for link \( j \) because any single SRLG failure will not break both the top path and the bottom path. ■