OBTAINING LOWER BOUNDS FOR THE FIPP P-CYCLE PROBLEM

Caroline Rocha
Escola de Ciências e Tecnologia, UFRN
Campus Universitário Lagoa Nova, 59072-970, Natal-RN, Brazil
caroline.rocha@ect.ufrn.br

Brigitte Jaumard
CSE, Concordia University
1455 de Maisonneuve Blvd W., H3G 1M8, Montreal-QC, Canada
bjaumard@ciise.concordia.ca

Thomas Stidsen
IMM, Technical University of Denmark
Richard Petersens Plads, building 321, DK-2800 Lyngby, Denmark
tks@imm.dtu.dk

ABSTRACT

In this study, we investigate a hierarchical decomposition approach for the design of survivable networks based on failure-independent path-protecting (FIPP) *p*-cycles. FIPP *p*-cycles extend link-protection *p*-cycles by adding the property of providing end-to-end failure independent path switching against either link or node failures. Most existing work on FIPP *p*-cycles suffer from either a lack of scalability or a lack of information about the quality of their heuristic solutions. In order to overcome those drawbacks, we propose a hierarchical column generation formulation both with a more compact formulation and a decomposition of the pricing problem. It turns out to be a much more efficient formulation than the previously proposed column generation one. Computational results show that we are able to solve accurately large network and traffic instances.

KEYWORDS. Optimization, Network survivability. Column generation. Main Area: Mathematical Programming.

1. Introduction

With the growth of Internet services and the enormous bandwidth capability of optical networks brought with DWDM (Dense Wavelength Division Multiplexing) technology, the information society of today relies massively on communication networks demanding more and more high quality service provisioning. In these network architectures, services are constantly exposed to risks of breakdown, either due to human errors or to equipment malfunctions. Therefore, service survivability mechanisms play a crucial role in the deployment of optical networks. Although several types of failures can occur in a network, such as fiber cut, equipment defect or bad operation, the most predominant scenario is single link failures due to fiber cuts by entrepreneurs or by natural disasters [Ramaswami and Sivarajan (2002), Grover (2004), Stern *et al*.(2008)].

Several schemes have been proposed to achieve survivability by employing protection or restoration in the optical layer. A quite interesting and recent option among these schemes is *p*-cycle protection [Grover and Stamatelakis (1998)]. The main concept behind *p*-cycles is that they recover from on-cycle link failures exactly as BLSR rings, but they protect chord links, also called straddling links, as well. The practical importance of *p*-cycles is their ability to achieve a good trade-off between capacity efficiency and restoration time since they provide fully pre-cross-connected protection paths over their cyclic structure. This means that, upon a failure detection, no more action, besides switching the affected traffic at the end nodes, is needed for restoration. Since *p*-cycles were introduced in 1998, many studies have been carried out on the topic, including extensions of the *p*-cycle concept from link protection to node failure recovery [Stamatelakis and Grover (2000)], path-segment protection [Shen and Grover (2003)], and path protection [Kodian and Grover (2005)], among others. In this study, we are particularly interested in failure-independent path-protecting (FIPP) *p*-cycles proposed by Kodian and Grover (2005). FIPP *p*-cycles extend link-protection *p*-cycles to allow for end-to-end path protection. Likewise the original concept, the same end-node preplanned protection switching response takes effect.

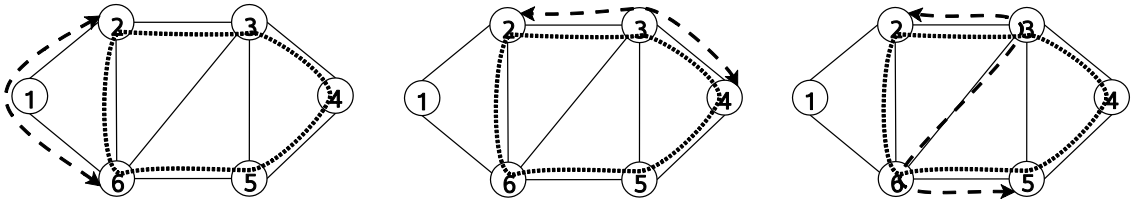
1.1. FIPP *p*-cycle concept

First of all, let us define the concept of working and protection paths. A working path is a path used to carry some traffic under normal operation conditions while a protection path is a backup path that is only used in case of failure.

The FIPP *p*-cycle concept is explained using the example illustrated in Figure 7.1. FIPP *p*-cycles and working paths are represented by dotted and dashed lines respectively. In Figure 1(a), path 2-1-6 is a straddling working path since it is link-disjoint from the cycle. Whether link 1-2 or 1-6 fails, protection paths 2-6 and 2-3-4-5-6 over the cycle can be used to restore the traffic on this path. In Figure 1(b), a failure on links 2-3 or 3-4 of on-cycle working path 2-3-4 can be recovered by using protection path 2-6-5-4. More complicated relationships between a working path and a FIPP *p*-cycle can appear as shown in Figure 1(c). In this case, called z-relationship, the whole cycle is needed for protecting working path 2-3-6-5 and the protection path used depends on which working link is affected. For example, protection path 2-6-5 can be used to recover from a failure on links 2-3 and 3-6, and protection path 2-3-4-5 protects against a failure on link 5-6.

The cyclical protection structure of a FIPP *p*-cycle can be shared by a set of working paths for protection as long as they are mutually disjoint or, if it is not the case, their corresponding protection paths are mutually disjoint. These criteria have to be met in order to avoid contention for protection resources after a failure. Using our example again, working paths 2-1-6 and 2-3-4 can share the same unit-capacity *p*-cycle, but working paths 2-3-4 and 2-3-6-5 cannot.

Figure 1: A FIPP *p*-cycle example. Working paths are represented by dashed lines and the *p*-cycle is represented by dotted lines.



(a) Straddling working path (b) Fully on-cycle working path (c) Partially on-cycle working path

1.2. Related work

The first work on FIPP *p*-cycles is documented by Kodian and Grover (2005) and includes an integer linear programming (ILP) model, entitled FIPP-SCP, to solve the non-joint design of FIPP *p*-cycle networks. In this problem, the working paths are routed prior to the placement of FIPP *p*-cycles. The solution approach consists in first identifying a set of candidate cycles and providing it for the model to determine the best assignment of working paths to cycles with respect to the protection cost. In their study, the authors assume that only mutually disjoint working paths can share the same *p*-cycle while one can impose different conditions.

An alternative approach, called FIPP-DRS, is proposed by Kodian *et al*. (2005). At first, sets of mutually disjoint routes (DRSs) are identified by a heuristic algorithm. Then, a number of candidate cycles is identified for each DRS and all preprocessed data are provided to an ILP model. Both previous methods have focused on the idea of mutual disjointness between working routes protected by the same cycle in order to reduce the complexity of the problem. Different assumptions are considered by Jaumard *et al*. (2007). Therein, the authors allow non-disjoint paths to be protected by the same cycle but not paths in a z-relationship. They proposed a column generation formulation and solution framework for the linear relaxation of the problem, and a heuristic approach to obtain integer solutions. Although this solution approach was shown to be more efficient in obtaining near optimal solutions than the existing methods, the complexity of the pricing problem compromises its scalability. An improved formulation of the pricing problem is then used by Rocha and Jaumard (2008) and Rocha *et al*. (2009b). Zhang and Zhong (2006) proposed a heuristic algorithm for the design of directed FIPP *p*-cycles. The key idea of their algorithm is to iteratively select the most efficient cycles from a list of enumerated cycles based on an efficiency metric, which resembles some of the previous approaches proposed for basic *p*-cycles.

The mentioned studies deal with the non-joint optimization FIPP *p*-cycle design problem, i.e., firstly, the design of the working paths, secondly, the design of the protection scheme. Two further studies, however, turn their attention to joint designs. The first one is due to Ge *et al*.(2007). Therein, a purely heuristic algorithm without any embedded ILP component is used to solve the problem. The second study was carried out by Baloukov *et al*. (2008). Their method works as an extension of the DRS method proposed by Kodian *et al*. (2005).

All previous work done on FIPP *p*-cycles suffer from either their lack of an assessment of the quality of their (heuristic) solutions or a lack of scalability. As shown by Jaumard *et al*. (2007), some heuristic solutions may be quite far from an optimal solution, sometimes up to 37%. The objective of this paper is to propose first an enhancement on the previous column generation formulation for the non-joint design of FIPP *p*-cycles. It consists of a much more compact formulation of the pricing problem. In addition, we present a new decomposition of the pricing problem which allows us to significantly speed up the run times.

The rest of the paper is organized as follows. In the next section, we formally define the FIPP *p*-cycle optimization problem, and then a mathematical formulation is presented. In Section 3, the pricing problem is formally defined. Also, we present a new enhanced formulation and a decomposition of the pricing problem. The column generation algorithm is described in detail in

Section 4. Finally, computational experiments are presented in Section 5, followed by the conclusions of the work.

2. Design of FIPP p-cycle networks

In the following, the problem of designing survivable networks using FIPP p-cycles is formally described. For this, some important assumptions and definitions are discussed and presented in Section 2.1, and then a mathematical formulation of the problem is provided in Section 2.2.

2.1. Assumptions and definitions

We consider an optical network represented by an undirected graph $G = (V, L)$ where V is the set of nodes and L is the set of links, which represent physical entities that collect all channels between neighbor nodes. For instance, a link can represent a set of cables co-routed in the same duct and each cable may contain multiple fibers. There is a linear cost c_ℓ for using one unit of spare capacity of each link $\ell \in L$. Link costs can represent information such as the cost of the interconnection equipment at endpoints, link length, etc. Furthermore, we are given a static set of connection requests, each of them associated with a distinct pair of nodes. For each connection request $k \in K$, we are given its required bandwidth b_k (number of optical channels) and its working route WP_k between its end nodes, o_k and d_k. Let P denote the set of all potential candidate cycles in the network. The cost of cycle $p \in P$ is given by $COST_P = \sum_{\ell \in P} c_\ell$.

The following assumptions are taken into consideration in this study:

(i) The network is composed of bidirectional (undirected) links, i.e., each topological link consists of a pair of fiber links, one in each direction. We also assume that the traffic is symmetric since the connection requests are undirected.
(ii) Although FIPP p-cycles can provide resilience against node failures, we only address single link failures, which are the predominant failure scenario in optical networks [Grover, 2004]. Thus, whenever we refer to route disjointness hereinafter, we mean link disjointness, except where otherwise stated.
(iii) The working traffic of each connection request is assumed to take a single route. However, we assume that the traffic can be split into integer parts (integer numbers of channels) during protection.
(iv) The working traffic is sent through the lowest cost routes, computed a priori, therefore our focus is on determining the cycles needed to protect those working paths.

The FIPP p-cycle design problem can now be defined as follows.

FIPP p-cycle design problem: Given the above definitions and assumptions, determine a set of FIPP p-cycles with associated protection capacity so as to fully protect all connection requests in K while minimizing the overall protection cost, given by $\sum_{p \in P} COST_p s_p$, where s_p is the number of unit copies of cycle p.

2.2. An ILP formulation

We now present an ILP formulation for the FIPP p-cycle design problem under the above assumptions. For each cycle $p \in P$, let us define set $\mathcal{C}_p$ as the set of all its cycle configurations. A cycle configuration corresponds to an association of a unit-capacity p-cycle and a subset of connection requests for which protection is provided. In particular, configuration $C \in \mathcal{C}_p$ is represented by vector α^C of length $|K|$, where coefficients $\alpha_k^C \in \{0,1,2\}, k \in K$, define

the level of protection (the number of protection paths) provided by cycle p for connection k. Let us recall that a connection can have two protection paths over a given cycle only if it fully straddles that cycle. In this study, we assume that a feasible cycle configuration meets the following requirements: i) It corresponds to a simple and unit-capacity cycle; ii) Only mutually disjoint connections are protected.

The formulation is based on the decision variables $n_C, C \in \mathcal{C}_p, p \in P$, representing the number of unit copies of p-cycle p reserved for configuration C. The FIPP p-cycle design problem can be formulated as follows:

$$\text{minimize} \sum_{p \in P} \left(COST_p \sum_{C \in \mathcal{C}_p} n_C \right) \tag{1}$$

subject to:

$$\sum_{p \in P} \sum_{C \in \mathcal{C}_p} \alpha_k^C n_C \geq b_k \qquad \forall k \in K \tag{2}$$

$$n_C \in \mathbb{Z}_+ \qquad \forall C \in \mathcal{C}_p, p \in P \tag{3}$$

The objective function minimizes the cost of the overall capacity used for protection. The demand constraints (2) ensure that enough capacity to protect each connection is allocated over all cycle configurations. Finally, we have constraints (3) defining the integer domain of the variables.

The main drawback of this model is that the number of possible cycles as well as cycle configurations grows exponentially with the network and traffic sizes. It clearly makes impracticable the optimization of this model through explicit enumeration of all cycle configurations. Fortunately, cycle configurations can be iteratively computed by means of a column generation algorithm so that, as a result, only a small fraction of cycles and configurations will be generated at the end of the process. In order to apply column generation to implicitly consider cycle configurations, the model above is relaxed by removing the integrality constraints, i.e., $n_C \in \mathbb{R}_+, \forall C \in \mathcal{C}_p, p \in P$. The resulting model is the so-called master problem. When only some columns are considered in the master problem, this is called the restricted master problem (RMP). Columns are further generated based on their reduced costs, which are derived from the dual variables of RMP. The problem of finding negative reduced cost columns to be added to RMP is discussed in Section 3, and the proposed column generation algorithm is described in detail in Section 4. For background information about linear programming techniques, we refer the reader to [Dantzig (1963), Chvatal (1983)].

3. Generating columns

The column generation algorithm iteratively requires the solution of the pricing problem. Assuming that the optimization problem under concern is a minimization one, by definition, the pricing problem corresponds to the problem of finding a column with minimum reduced cost, i.e., to the problem of generating an augmenting column that allows decreasing the current value of the objective function. Here, a cycle configuration with minimum reduced cost can be viewed as the one making the best compromise between protection cost and provision of protection for the connections.

Let $\lambda_k \geq 0, k \in K$ be the dual variables associated with constraints (2). The reduced cost of a cycle configuration $C \in \mathcal{C}_p$, given by expression (4), is composed of two terms: the cost of cycle p and the revenue obtained for protecting connections.

$$\overline{COST}_C = COST_p - \sum_{k \in K} \lambda_k \alpha_k^C \tag{4}$$

As mentioned, the pricing problem corresponds to the problem of finding a cycle configuration with a negative reduced cost, but it can be further decomposed thereby restraining the search to a set of mutually disjoint requests for a given cycle. At first, let us present a formal definition as well as a mathematical formulation for the original pricing problem. Then, we will explain how it can be decomposed and present a formulation for the resulting pricing problem. The pricing problem, hereinafter called the cycle configuration problem (CCP), can be defined as follows.

Cycle configuration problem: Let us consider an undirected graph $G = (V, L)$ with link costs $c: L \mapsto \mathbb{R}_+$. Also, let us denote by K the set of requests with associated revenues $\lambda: K \mapsto \mathbb{R}_+$and working routes WP_k between end nodes o_k and d_k for each request $k \in K$. The cycle configuration problem asks for a cycle configuration of minimum total cost, where the cost of configuration C associated with cycle p is defined by expression (4).

The following notation is introduced in order to present a formulation for CCP. Let us define binary variables x_ℓ, $\ell \in L$, and y_v, $v \in V$, as follows: $x_\ell = 1$ if and only if link ℓ belongs to the cycle; $y_v = 1$ if and only if node v is traversed by the cycle. In addition, let us define binary variables z_k and w_k, $k \in K$, as follows: $z_k = 1$ if and only if connection k is protected; $w_k = 1$ if and only if connection k is protected and straddles the cycle (k is a chord). Let us also define $\delta(S)$, $S \subset V$, as the cut induced by S, i.e, the set of links incident to a node in S and another node in $V \backslash S$. For a single node $v \in V$, we denote $\delta(S) = \delta(\{v\})$. Additionally, let $L(S)$, $S \subset V$, be the set of links induced by S, i.e., the set of links whose both adjacent nodes belong to S. The problem of generating cycle configurations can be formulated as follows:

$$\text{minimize} \sum_{\ell \in L} c_\ell x_\ell - \sum_{k \in K} \lambda_k (z_k + w_k) \tag{5}$$

subject to:

$$\sum_{\ell \in \delta(v)} x_\ell = 2y_v \qquad \forall v \in V \tag{6}$$

$$\sum_{\ell \in \delta(S)} x_\ell \geq 2(y_i + y_j - 1) \qquad \forall i \in S, j \in V \backslash S, S \subset V, 3 \leq |S| \leq |V| - 3 \tag{7}$$

$$z_k \leq y_v \qquad \forall k \in K, v \in \{o_k, d_k\} \tag{8}$$

$$2w_k + x_\ell - z_k \leq 1 \qquad \forall k \in K, \ell \in WP_k \tag{9}$$

$$\sum_{k \in K: WP_k \ni \ell} z_k \leq 1 \qquad \forall \ell \in L \tag{10}$$

$$z_k, w_k \in \{0,1\} \qquad \forall k \in K \tag{11}$$

$$x_\ell \in \{0,1\} \qquad \forall \ell \in L \tag{12}$$

$$y_v \in \{0,1\} \qquad \forall v \in V \tag{13}$$

The objective function (5) calculates the reduced cost. Degree constraints (6) require the degree of each node to be either 0 or 2. Inequalities (7) are connectivity constraints forcing each cut separating two visited nodes to be crossed at least twice. Constraints (8) ensure that each protected connection has both end nodes crossed by the cycle. Constraints (9) determine whether a protected connection is a chord of the cycle or not. Inequalities (10) are capacity constraints ensuring that concurrent connections are not simultaneously protected. Therefore, they prevent connections from using more than one unit of protection capacity. Finally, domain constraints (11), (12), and (13) ensure binary values for all variables.

This formulation can be considered as an improvement with respect to the one proposed by Rocha *et al*. (2009b) since it has a much smaller number of variables and constraints. In the previous formulation, there are variables representing the protection paths taken over the cycle by

the connections so as to avoid contention for capacity by non-disjointly routed requests. This is not the case here because we do not allow non-disjoint requests to share the same unit p-cycle, which is ensured by constraints (10). A shortcoming of this formulation, which is the major contributor to its high complexity, lies in the exponential number of constraints (7) used to avoid subtours (or, in other words, multiple cycles). Those subtour elimination constraints are needed because generating more than one cycle at a time would make the identification of the straddling links very difficult. However, these constraints can be added to the model only when needed using a branch-and-cut algorithm, as explained in Section 4.

The cycle configuration problem was proved NP-hard by Rocha *et al.* (2009a). Note, however, that the pricing problem does not need to be solved exactly at each iteration in order to obtain an improved solution, and hence heuristic generation of improving columns may be applied. Ultimately optimal solution of the pricing problem is still required to prove optimality of the column generation algorithm, nevertheless.

With this in mind, an interesting finding allows us to efficiently compute heuristic solutions for CCP. The key idea is to decompose the problem into two subproblems: find a cycle and then identify the most profitable configuration for that cycle. Indeed, if the cycle associated with the best configuration to be added to RMP is known, then one only needs to find the set of mutually disjoint requests that can be protected by that cycle and maximizes the revenues from the dual prices. The corresponding subproblem is called the cycle packing problem. Indeed, a set of candidate cycles is known at each iteration of the column generation algorithm: Those associated with the previously generated configurations. The main advantage of this decomposition is that the complexity of this subproblem is greatly reduced in comparison with the aggregate pricing problem. Thus, the resulting column generating algorithm turns out to be less time consuming and more scalable. Note that an approximate solution for the CCP is obtained as a result of this decomposition unless the optimal cycle for the given dual prices is known.

The cycle packing problem (CPP) can be formally defined as follows.

Cycle packing problem: For a given FIPP p-cycle p, let us define set $K_p = \{k \in K$ such that o_k and d_k are crossed by $p\}$. Each connection request $k \in K_p$ is associated with a nonnegative revenue q_k defined as follows: $q_k = \lambda_k$ if connection k is partially or fully on cycle p, otherwise $q_k = 2\lambda_k$. CPP asks for the subset of mutually disjoint requests in K_p, called a cycle packing, that maximizes the total revenue.

We strongly believe that the cycle packing problem is NP-hard, given its similarity to the maximum weight independent set problem (MWISP) [Bomze *et al*. (1999)]. In the following, we present a mathematical formulation for CPP. It also uses binary variables z_k, $k \in K_p$, such that $z_k = 1$ if and only if request k is protected by cycle p.

$$\text{maximixe} \sum_{k \in K_p} q_k z_k \tag{14}$$

subject to:

$$\sum_{k \in K_p : WP_k \ni \ell} z_k \leq 1 \qquad \forall\, \ell \in L \tag{15}$$

$$z_k \in \{0,1\} \qquad \forall k \in K \tag{16}$$

The objective function (14) calculates the total revenue and constraints (15) ensure that only mutually disjoint requests are protected. Remark that this formulation relates to the classical formulation for MWISP in a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ in which there are constraints $z_i + z_j \leq 1, \forall \{i,j\} \in \mathcal{E}$, stating that two adjacent nodes in the graph cannot be part of an independent set. However, CPP formulation is certainly tighter than that one since it gathers information from the working paths.

4. Column generation algorithm

The idea behind column generation technique is to only introduce variables when needed, i.e., when their reduced cost is negative. The method relies on a decomposition of the initial linear program into a master problem and a pricing problem. The master problem corresponds to a linear program subject to a first set of explicit constraints and a second set of implicit constraints expressed throughout properties of the coefficients of the constraint matrix. The pricing problem consists in the optimization of the reduced cost subject to the set of implicit constraints: It either identifies favorable columns to be added to the master problem or indicates that no such column exists.

The general framework of the proposed column generation algorithm is summarized as follows. Initially, the algorithm starts with a set of artificial (dummy) columns, one for each request. An artificial column corresponds to a cycle configuration providing protection for only one request, which is so costly that it will never be part of the optimal solution. Then, the restricted master problem containing the initial set of variables is solved and the resulting dual variables are used to guide the search for a cycle configuration with negative reduced cost. This process continues until no improving cycle configuration is found.

A typical iteration of our column generation works in the following fashion. Firstly, each known cycle is considered, from the last to the first generated one, and its corresponding cycle packing problem is solved. As soon as the solution of a given CPP produces an improving column, this is provided to the restricted master problem and a new iteration begins. Contrariwise, if no improving column can be found with the known cycles, we cannot yet claim optimality of the master problem and CCP ought to be solved. If the solution of CCP does not produce an improving column either, the optimal solution for the restricted master problem is also optimal for the original master problem and the algorithm terminates. Otherwise, the column with negative reduced cost is provided for the RMP and the algorithm iterates again.

Solving CPP is the bottleneck of the column generation algorithm. The solutions for CPP are obtained by solving the proposed formulation which unfortunately contains the costly connectivity constraints. In order to avoid this pitfall, the model is solved with a branch-and-cut solver using a cutting plane scheme so as the connectivity constraints are only introduced to the model when violated in the incumbent solution. The most violated constraints are identified by the exact separation algorithm proposed by Fischetti *et al*. (1997), which is based on the computation of maximum flows in the network. Let us recall that CPP does not need to be solved exactly as long as we are able to find a cycle configuration with negative reduced cost for improving the solution of RMP. Hence, the solution of CPP is stopped as soon as a column with negative reduced cost is obtained. This approach does not hamper the optimality of the master problem solution, instead, it often speeds up the solution process. All CPPs are solved using the proposed formulation within a branch-and-cut solver.

5. Computational results

In this section, we evaluate the efficiency of our solution approach using a set of 16 benchmark problem instances whose details are provided in Table 1. For each network, the number of nodes, the number of links, average node degree, and the number of connection requests are provided. When only asymmetric traffic was available for a given instance, we considered the maximum amount of traffic between each pair of nodes in order to obtain a

symmetric traffic matrix. The working route for each request was obtained by using Dijkstra algorithm to find the lowest cost route. All algorithms were implemented in C++ programming language using Concert Technology library and version 10.1 of CPLEX solver. The computational experiments were performed on a AMD 64-bit machine with 16GB of RAM.

Table 1: Characteristics of the problem instances

Instances	Nodes	Links	Avg. Node Degree	Requests
9N17S [Grover (2010)]	9	17	3.8	36
DFN-BWIN [Orlowski *et al.* (2007)]	10	45	9.0	45
COST239 [Batchelor *et al.* (1999)]	11	26	4.7	55
POLSKA [Orlowski *et al.* (2007)]	12	18	3.0	66
USA [Hülsermann *et al.* (2004)]	14	21	3.0	91
ATLANTA [Orlowski *et al.* (2007)]	15	22	2.9	210
GERMANY [Hülsermann *et al.* (2004)]	17	26	3.1	136
NEWYORK [Orlowski *et al.* (2007)]	16	49	6.1	240
EON-19 [Grover (2010)]	19	37	3.9	171
TA1 [Orlowski *et al.* (2007)]	24	55	4.6	163
NORWAY [Orlowski *et al.* (2007)]	27	51	3.8	351
BRAZIL [Noronha and Ribeiro (2006)]	27	70	5.2	351
EON-28 [Hülsermann *et al.* (2004)]	28	41	2.9	378
BT [Grover (2010)]	30	59	3.9	435
CSELT [Grover (2010)]	30	56	3.7	435
COST266 [Orlowski *et al.* (2007)]	37	57	3.1	666

The first performed experiments evaluate the proposed column generation algorithm with the new formulation for CCP but without the embedded pricing problem decomposition, i.e, the CPP component was not included in the algorithm. Table 2 provides information about the performance of the solution approach. In more details, we provide the number of columns (cycle configurations) and the number of distinct cycles generated during the column generation process. We also provide the total running time in seconds as well as the percentage of time required for the solution of the master and pricing problems. In the worst case, the column generation algorithm took more than one day to obtain the optimal solution for the linear relaxation of the FIPP *p*-cycle design problem. From the information provided in Table 2, we can also notice that the solution of CPP is very time consuming, responding for more than 99% of the total running time in most cases.

Table 2: Performance of the column generation algorithm without pricing decomposition

Instances	cost	RR	# columns	time(s)	master time	CCP time
9N17S	2,900.00	49.15%	78	4.68	0.43%	99.57%
DFN-BWIN	178,550.00	52.40%	19	0.58	0.01%	99.99%
COST239	60,263.91	43.93%	243	38.44	0.31%	99.69%
POLSKA	3,398,548.00	72.42%	165	12.86	0.62%	99.38%
USA	5,874,273.44	99.12%	253	49.32	0.43%	99.57%
ATLANTA	135,951.00	90.02%	232	33.46	0.36%	99.64%
GERMANY	446,372.50	109.64%	331	86.19	0.28%	99.72%
NEWYORK	485.93	33.82%	598	347.26	0.24%	99.76%
EON-19	89,489.39	98.05%	930	371.26	0.57%	99.43%
TA1	5,576,871.50	84.78%	868	402.52	0.54%	99.46%
NORWAY	5,504.75	51.01%	2,798	14,051.83	0.25%	99.75%
BRAZIL	1,905,574.40	70.84%	2,99	18,043.30	0.23%	99.77%
EON-28	1,908,707.50	106.24%	2,166	6,057.35	0.23%	99.77%
BT	2,752.69	42.72%	6,155	84,705.04	0.54%	99.46%
CSELT	1,710.23	41.12%	8,336	71,598.92	1.53%	98.47%
COST266	11,866,362.78	96.42%	6,708	109,859.13	0.35%	99.65%

In order to appraise the benefits of the pricing problem decomposition, the performance of the resulting column generation algorithm can be evaluated in Table 3. For each tested network, the table shows, besides the number of columns and cycles generated, the running time and the percentage of the time required for solving each component of the algorithm (master problem, CCP, and CPP). We can see now that the total running time is significantly reduced in comparison with the algorithm without pricing decomposition. Indeed, the pricing decomposition yields a reduction of up to 90% in the running time. The decomposition also produces a more even distribution of consumed time among the components. Because the solutions obtained for CCP tend to be suboptimal with respect to CPP, a larger number of columns needs to be generated in order to reach optimality.

Table 3: Performance of the column generation algorithm with pricing decomposition

Instances	# columns	#cycles	time(s)	master time	CPP time	CCP time
9N17S	206	26	2.56	2.34%	29.30%	68.36%
DFN-BWIN	19	19	0.70	2.86%	18.57%	78.57%
COST239	452	76	20.92	1.58%	23.18%	75.24%
POLSKA	321	24	4.21	3.09%	28.98%	67.93%
USA	561	45	14.17	2.82%	22.23%	74.95%
ATLANTA	470	20	6.00	8.67%	32.50%	58.83%
GERMANY	535	21	7.86	8.14%	25.06%	66.79%
NEWYORK	1,7	149	146.80	3.03%	17.72%	79.25%
EON-19	2,162	75	75.30	12.27%	32.58%	55.15%
TA1	1,668	74	68.12	6.93%	22.50%	70.57%
NORWAY	8,23	228	2,478.35	20.82%	12.96%	66.22%
BRAZIL	10,603	280	2,909.12	16.05%	8.58%	75.37%
EON-28	4,783	112	622.85	14.72%	16.08%	69.20%
BT	25,14	589	22,342.81	22.19%	9.88%	67.93%
CSELT	28,639	501	19,798.92	46.52%	10.07%	43.41%
COST266	31,019	441	21,056.27	29.82%	16.45%	53.73%

As mentioned when presenting the formulations, we made the assumption that only mutually disjoint connections can be protected by the same cycle. The column generation algorithm proposed by Rocha and Jaumard (2008) does not impose connection disjointness but it does not allow connections in a *z*-relationship with the cycle. In order to assess the impact of these assumptions on the obtained lower bounds, we tested both formulations on a 15-node family of related networks with number of links ranging from 16 to 30 [Doucette (2004)]. The time limit of 10 hours was used for both formulations. Figure 2(a) illustrates the lower bounds found with the previous and the new formulations for each network. From the results, there is no clear advantage of any formulation, except for the largest networks, for which slightly better bounds were found with the new formulation. However, regarding the running times, the column generation proposed in this paper is remarkably superior to the previous one. Note that, for a fair comparison, the column generation algorithm without pricing decomposition was used in these experiments. Figure 2(b) shows how much time is consumed for running the column generation algorithm using both formulations. Now, it is clear that the new formulation is much more effective.

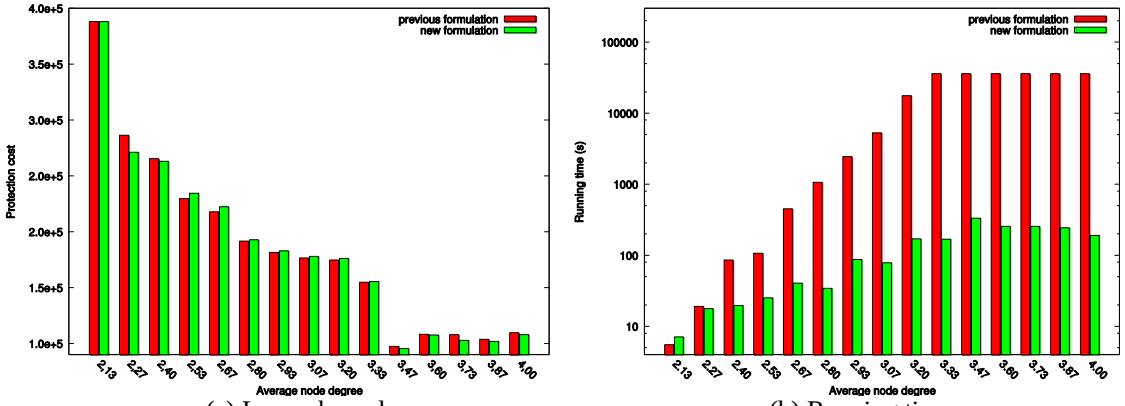
6. Conclusion

In this paper, we proposed a new column generation approach for the efficient computation of FIPP *p*-cycles in survivable networks. The method relies on a new decomposition strategy in which two pricing problems are used, one for generating new cycles and another one for improving the use of existing cycles. By imposing mutual disjointness among working paths protected by the same cycle, we came up with a much more compact formulation for the pricing problem. This, together with the embedded decomposition, greatly reduced the running times and

allowed us to approach problem instances of size never yet approached. Results have shown that this assumption did not affect the quality of the lower bounds obtained in comparison with an existing method, which in turn did not allow a cycle to protect requests in a *z*-relationship with the cycle.

As future research, it might be worthwhile to investigate valid inequalities to strengthen the formulation of the pricing problem thereby possibly accelerating the solution of the linear relaxation. Other research direction is the development of efficient heuristics in order to obtain good initial solutions for the column generation algorithm.

Figure 2: Results for a 15-node network family.



(a) Lower bounds (b) Running times

Acknowledgment

Support for this work from the Brazilian National Council for Scientific and Technologic Development (CNPq), under grant 200497/2005-7, is gratefully acknowledged.

References

Baloukov, D., Grover, W., and Kodian, A. (2008). Toward jointly optimized design of failure-independent path-protecting *p*-cycle networks. *Journal of Optical Networking*, 7(1):62–79.

Batchelor, P., van Caenegem, B., Daino, B., Heinzmann, P., Hjelme, D., Inkret, R., Jäger, H., Joindot, M., Kuchar, A., Coquil, E. L., Leuthold, P., de Marchis, G., Matera, F., Mikac, M., Nolting, H., Späath, J., Tillerot, F., Wauters, N., and Weinert, C. (1999). Ultra high-capacity optical transmission networks: Final report of action COST 239. *Technical report*, Faculty of Electrical Engineering and Computing, University of Zagreb.

Bomze, I., Budinich, M., Pardalos, P., and Pelillo, M. (1999). *The maximum clique problem. In Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic Publishers.

Chvatal, V. (1983). *Linear Programming*. Freeman.

Dantzig, G. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton.

Doucette, J. (2004). *Advances on Design and Analysis of Mesh-Restorable Networks*. PhD thesis, University of Alberta, Edmonton, AB, Canada.

Fischetti, M., Gonzalez, J. J. S., and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378-394.

Ge, C., Bai, N., Sun, X., and Zhang, M. (2007). Iterative joint design approach for failure-independent path-protecting *p*-cycle networks. *Journal of Optical Networking*, 6(12):1329–1339.

Grover, W. D. (2010). www.ece.ualberta.ca/~grover/book/.

Grover, W. D. (2004). *Mesh-Based Survivable Networks*. Prentice Hall.

Grover, W. D. and Stamatelakis, D. (1998). Cycle-oriented distributed pre-configuration: Ring-like speed with mesh-like capacity for self-planning network restoration. In *IEEE International Conference on Communications (ICC 1998)*, pages 537–543.

Hülsermann, R., Bodamer, S., Barry, M., Betker, A., Gauger, C., Jäger, M., Köhn, M., and Späth, J. (2004). A set of typical transport network scenarios for network modelling. *Technical report*, ITG-Fachtagung Photonische Netze, Leipzig.

Jaumard, B., Rocha, C., Baloukov, D., and Grover, W. D. (2007). A column generation approach for design of networks using path-protecting *p*-cycles. In *Proceedings of the Sixth International Workshop on the Design of Reliable Communication Networks (DRCN)*.

Kodian, A. and Grover, W. D. (2005). Failure-independent path-protecting *p*-cycles efficient and simple fully preconnected optical-path protection. *Journal of Lightwave Technology*, 23(10):3241–3259.

Kodian, A., Grover, W., and Doucette, J. (2005). A disjoint route sets approach to design of failure-independent path-protection *p*-cycle networks. In *Proceedings of the International Workshop on the Design of Reliable Communication Networks (DRCN)*.

Noronha, T. and Ribeiro, C. (2006). Routing and wavelength assignment by partition coloring. *European Journal of Operational Research*, 171(3):797–810.

Orlowski, S., Pióro, M., Tomaszewski, A., and Wessäly, R. (2007). SNDlib 1.0–Survivable Network Design Library. In Proceedings of the Third International Network Optimization Conference (INOC), Spa, Belgium. http://sndlib.zib.de.

Ramaswami, R. and Sivarajan, K. (2002). *Optical networks: A practical perspective*. Morgan Kaufmann, 2nd edition.

Rocha, C. and Jaumard, B. (2008). Revisiting *p*-cycles / FIPP *p*-cycles vs. shared link / path protection. In *17th International Conference on Computer Communications and Networks (ICCCN)*, St. Thomas, USA.

Rocha, C., Jaumard, B., and Bougué, P.-E. (2009a). Asymmetry issues in *p*-cycle and FIPP *p*-cycle protection schemes. *Submitted to publication*.

Rocha, C., Jaumard, B., and Bougué, P.-E. (2009b). Directed vs. undirected *p*-cycles and FIPP *p*-cycles. In *Proceedings of the International Network Optimization Conference (INOC)*, Pisa, Italy.

Shen, G. and Grover, W. (2003). Extending the *p*-cycle concept to path segment protection for span and node failure recovery. *IEEE Journal on Selected Areas in Communications*, 21(8):1306–1319.

Stamatelakis, D. and Grover, W. D. (2000). IP layer restoration and network planning based on virtual protection cycles. *IEEE Journal on Selected Areas in Communications*, 18(10):1938–1949.

Stern, T., Ellinas, G., and Bala, K. (2008). *Multiwavelength Optical Networks: Architectures, design, and Control*. Cambridge University Press, second edition.

Zhang, F. and Zhong, W.-D. (2006). A novel path-protecting *p*-cycle heuristic algorithm. In *Proceedings of the International Conference on Transparent Optical Networks (ICTON)*, volume 3, pages 203–206, Nottingham, UK.