Les Cahiers du GERAD ISSN: 0711–2440

A Column Generation Approach for Design of Networks using Path-Protecting p-Cycles

B. Jaumard, C. Rocha,
D. Baloukov, W. Grover

G–2007–11

February 2007

A Column Generation Approach for Design of Networks using Path-Protecting p-Cycles

Brigitte Jaumard

GERAD and CIISE
Concordia University
Montréal (Québec) Canada, H3G 1M8
bjaumard@ciise.concordia.ca

Caroline Rocha

CRT and DIRO
Université de Montréal
Montréal (Québec) Canada, H3C 3J7
thennecy@crt.umontreal.ca

Dimitri Baloukov
Wayne Grover

TRLabs and Department of ECE
University of Alberta
Edmonton (Alberta), Canada, T6G 2V4
grover@trlabs.ca

February 2007

Les Cahiers du GERAD

G–2007–11

Abstract

We investigate a first column generation (CG) formulation for the design of failure independent path-protecting (FIPP) p-cycle survivable transport network. Previous work has proposed different formulations and heuristics for FIPP p-cycles which extend span-protecting p-cycles by adding the property of providing end-to-end failure independent path switching against either span or node failures. We develop a CG model that additionally allows the exploration of FIPP p-cycles without imposing mutual disjointness among working routes protected by the same cycle. The proposed CG model decomposes the FIPP p-cycle design problem into the master problem which takes care of the demand constraints, and the pricing problem which includes the constraints associated with the properties and the characteristics of a FIPP p-cycle. The key feature of a CG model lies in a generation of cycles motivated by the value of the reduced cost of the pricing problem, the key global indicator that is the driving element of the simplex algorithm. Results show a clear advantage of the CG model over the previous models, in particular in exploiting cycles that are not restricted to those satisfying a mutual disjointness condition on the working paths. Although it is not always possible to guarantee that the solutions obtained with the CG model are optimal, it can be shown that they are very close to the optimal ones.

Key Words: Path protecting p-cycles, column generation, path protection, preconnection, failure-independent protection.

Résumé

Nous étudions une première formulation de génération de colonnes, notée (CG), pour la conception de réseaux de transport fiables avec un schéma de protection de type FIPP, c'est-à-dire avec des p-cycles en guise de chemins de protection dans le cas de pannes indépendantes. Des travaux antérieurs ont proposé différentes formulations et heuristiques pour la définition de p-cycles de type FIPP, en étendant des modèles et algorithmes proposés pour la protection de liens à partir de p-cycles, en ajoutant la propriété de fournir une protection de bout en bout contre la panne d'un nœud ou d'un arc. Nous développons le modèle CG pour résoudre ce problème en ajoutant l'exploration de FIPP p-cycles qui n'imposent pas de contraintes d'exclusion mutuelle sur les routes d'opération protégées par un même p-cycle. Le modèle CG proposé décompose le problème de conception de p-cycles FIPP en un problème maître qui se préoccupe des contraintes de demande, et un problème auxiliaire qui inclut les contraintes associées avec les propriétés et les caractérisques d'un p-cycle FIPP. Le point central de ce modèle tient dans la génération de cycles motivée par le coût réduit du problème auxiliaire, l'indicateur global clé de l'algorithme du simplex. Les résultats montrent un avantage clair du modèle CG sur les modèles précédents, en particulier avec l'exploitation des p-cycles à ceux qui satisfont une condition d'exclusion mutuelle pour les chemins d'opération protégés par un cycle donné. Bien qu'il ne soit pas toujours possible de garantir que les solutions obtenues à l'aide du modèle CG soient optimales (temps de calcul excessif), nous montrons que les solutions obtenues sont très proches des solutions optimales.

Mots clés : p-cycles, chemin de protection, génération de colonnes, protection avec pannes indépendantes.

1 Introduction

Failure-independent path-protecting (FIPP) p-cycles is a recently proposed architecture for survivable networking that extends the concept of span-protecting p-cycles [1] to allow for end-to-end path protection. The most important property inherited by FIPP p-cycles is that of complete pre-connection of protection paths. This allows FIPP p-cycles to retain the 'ring like speed' of span protecting p-cycles while also gaining the property of failure independence of the protection reaction for each path. In other words, when a failure occurs on any span or node, the same end-node pre-planned protection switching response takes effect. A single pre-determined and fully pre-cross-connected protection path is enabled regardless of where the failure has occurred on the working path.

The fact that the protection path is completely known and completely pre-connected before failure is beneficial from the network operator point of view because it means the backup path can be tested and in a known working state before failure. This is advantageous compared to architectures such as Shared Backup Path Protection (SBPP) [2] where the protection route is known in advance, but no actual backup path is pre-connected and ready to use: it must be cross-connected on the fly immediately after failure. Pre-cross-connection means that adequate bit error rates (BER) can be assured when the protection path is substituted in real-time for the failed working signal. This property becomes even more important when considering transparent or translucent optical networks where nearly 20 [3] impairments need to be overcome before an 10Gb/s (or above) DWDM optical link can be established.

The FIPP p-cycle concept uses cyclical protection structures that can be shared by a set of working paths for protection as long as the working paths in this set are mutually disjoint or, if they are not, their protection paths are mutually disjoint. If these criteria are met, there will be no contention for spare capacity after a failure. Furthermore, the end-nodes of the working paths must also be crossed by the cycle assigned to protect them.

The original work on FIPP p-cycles, including a spare capacity placement (SCP) ILP model, are documented in [4]. Further work, as well as the disjoint route set (DRS) method, is covered in [5]. As mentioned in these works, and in interaction among the collaborating authors on this paper, a challenge for ongoing research on the FIPP p-cycle concept was the difficulty of obtaining strictly optimal solutions for FIPP p-cycle network designs.

Access to optimal solutions may not be essential in practice, but it is important to advance the basic networking science in this area. We need to know, for example, the theoretical limits of how efficient FIPP p-cycle network designs can be, to understand how this architecture really relates to SBPP, say, and to be able to approach the design of good heuristics by observing the actual properties of optimal solutions for the architecture. With this in mind, a CG-based approach appears to be a promising strategy to improve the solution quality of FIPP p-cycle network designs in a reasonable amount of run time. This paper accordingly proposes a CG-based method for FIPP p-cycle network design

and reports comparative performance results of the CG method and previous ILP-based solution models.

Note that the emphasis here is not on achieving a speed-up on the run times as might be important in a production-use model. Rather, the goal is to keep solution times about the same as were already being allowed or experienced with the initial approaches in [4] [5], but to achieve known-optimal terminations, or solutions with a much reduced gap against optimality.

2 Background and Literature

The logical operations of FIPP p-cycles are explained as follows by considering the different protection relationships of a given FIPP p-cycle to working routes.

Straddling routes. A route in a pure straddling relationship to its protecting FIPP p-cycle is such that it has no span in common with the p-cycle as the example shown in Figure 1(a) where the working routes and cycles are represented by dashed and solid connected lines, respectively. In this case, two distinct protection paths are available on the cycle, and thus up to two working paths on this route can be protected in case of a failure. This is a direct extension of the straddler concept from span protecting p-cycles which is one of the strongest traits contributing to their efficiency. In case of a failure, only the end-nodes of the route perform switching actions and any criteria can be adopted to assign working paths to unique protection paths. The pre-assigned direction is stored at the end-nodes where the switching action takes place as soon as a working path failure is detected. Note that the pre-defined switching action does not depend on failure type or location.

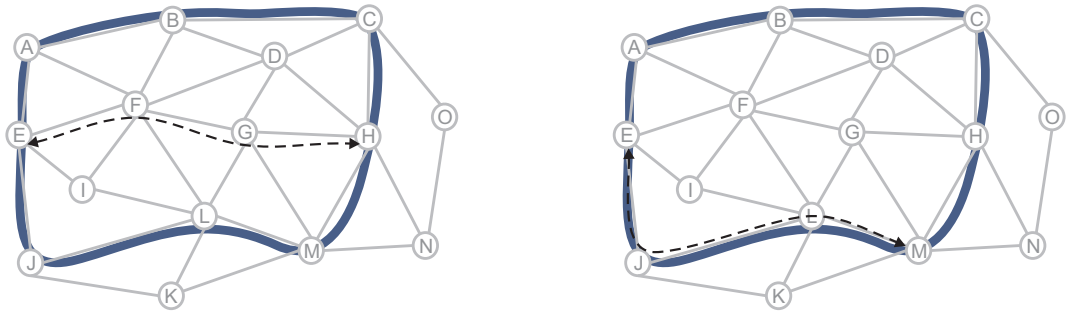


Figure 1: Different relationships between working routes and FIPP p-cycles: (a) fully straddling; (b) fully on-cycle.

On-cycle routes. The pure on-cycle relationship arises when all spans of the working route are crossed by the cycle protecting it as shown in Figure 1(b). This is the direct extension of the on-cycle concept from span protecting p-cycles. The protection path for such a relationship is unambiguously determined as the complementary part defined by the spans of the cycle that are not shared by the working route.

Partially on-cycle routes. The partially on-cycle relationship, not faced by basic p-cycles, arises from the extension to path protection. These occur when at least one but not all spans in the working route are shared by the cycle assigned to protect it. There are two operationally different types of this relationship:

Type 1. The type 1 partially on-cycle relationship occurs when the working path and the protection path provided by the cycle are disjoint. This is illustrated by Figure 2(a) and (b) where the dotted arrowed line represents the available protection path. Operationally, this is the same as the pure on-cycle relationship where the assigned protection path is enabled regardless of where the failure affects the working path.

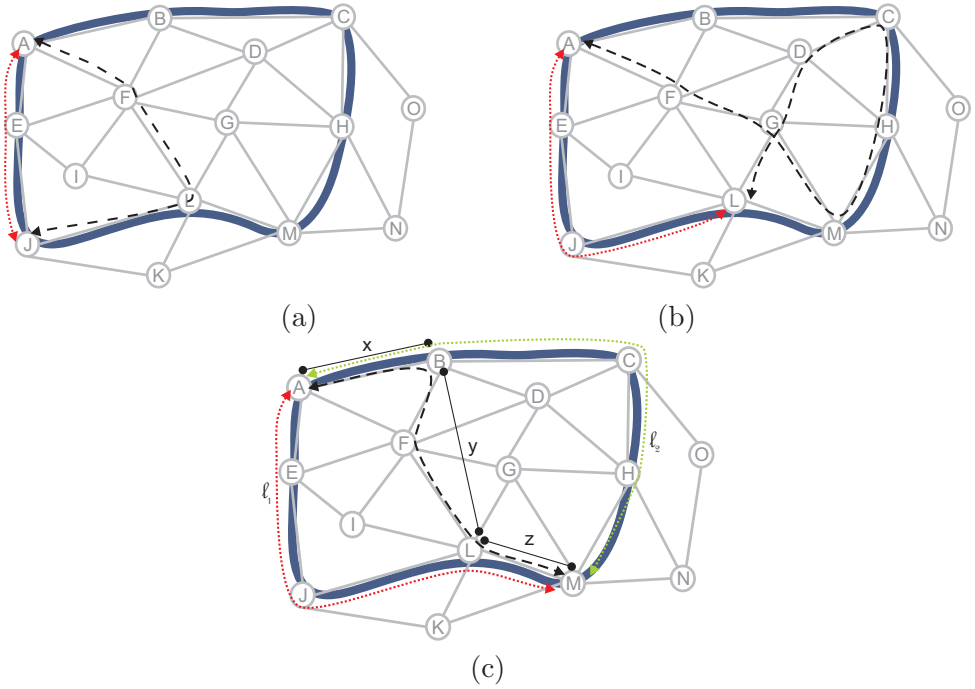


Figure 2: Examples of partially on-cycle routes.

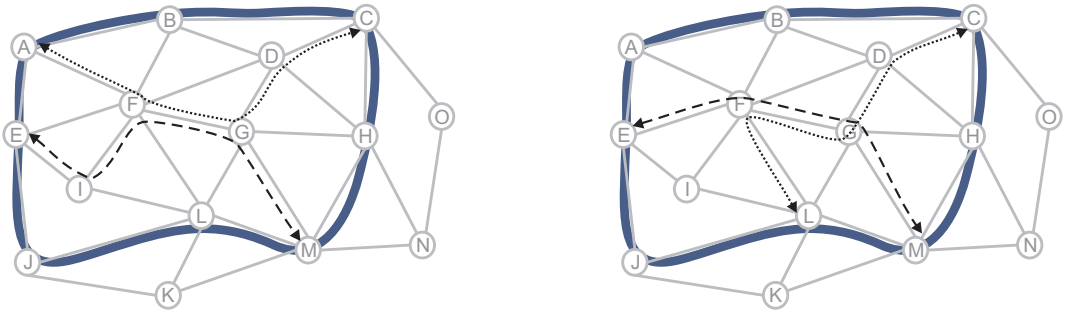
Type 2. The type 2 partially on-cycle relationship (also known as the z-case) occurs when, for a given working route, we cannot find a protection path on the p-cycle such that it does not share at least one span with its respective working route, as illustrated by the example

in Figure 2(c). In this case, two protection paths must be considered for protection and the switching logic is performed as follows. In the example, let us set the path ℓ_1 between nodes A and M as the default pre-assigned protection path.

If segments x (A-B) or y (B-F-L) are affected by a failure, the protection path will survive and the behavior is the same as before. However, a failure on segment z (L-M) implies that the protection path defined by the default direction also fails and the affected working path must now be protected by path ℓ_2. Fortunately, this can be realized locally at the end-nodes simply by determining which side of the cycle was affected by the failure along with the working path. In case of coincidence of failure states on both working path and its pre-assigned protecting path, the surviving protection path is selected for recovery.

Non-disjoint Working Route Sets

Prior work in [4] and [5] has focused on the idea of mutual disjointness between working routes protected by the same cycle. Only disjoint sets of working routes were allowed to share cycles for protection against failure. This was done to reduce the complexity of the problem and efficient results were obtained while under this assumption. However, as mentioned before, it is possible to protect non-disjoint routes using a single cycle as well, as long as the protection paths provided by the cycle are mutually disjoint from each other. This situation is illustrated in Figure 3(a) using the same network context as the previous examples. In the referred figure, there are two working routes, A-F-D-G-C and E-I-F-G-M, illustrated by a dotted and dashed line respectively. Although both routes go through span F-G, they can be protected by the same cycle because their protection paths (A-B-C and E-J-L-M) are mutually disjoint. Figure 3(b) shows an example in which two non-disjoint working routes cannot share a cycle because their protection paths cannot be disjoint. Note that the examples in Figure 3 only show an instance where non-disjoint straddling routes are protected by the same cycle. The idea of non-disjointness applies other route-cycle relationships as well, as long as the protection paths of the non-disjoint routes are disjoint from one another.



(a) (b)

Figure 3: Protection of non-disjoint working paths by FIPP p-cycles.

3 FIPP Mathematical Models

3.1 FIPP Models

Three different FIPP models are introduced in the following sections: FIPP-SCP-IH, FIPP-SCP-DRS [5] and a new one, called FIPP-CG that makes use of column generation. All correspond to a sequential optimization, i.e., working paths to be protected are defined in such a way that they are all routed over the lowest cost route between the end-nodes of each demand relation. While both FIPP-SCP models (Sections 4 and 5) assume that the set of working paths protected by a given FIPP p-cycle must be pairwise disjoint, this is no longer the case with the FIPP-CG model. The FIPP-SCP-DRS and FIPP-SCP-IH models allow for type 2 partially on-cycle relationships in the solutions while the FIPP-CG model does not, as it would entail too many variables to do so. Before going through the details of the various models, we first present a quick overview of a column generation modeling, as well as the common notations of the three models.

3.2 Motivation and Basic Theory for Column Generation

Column generation techniques offer solution methods for linear programs with a very large number of variables (e.g., exponential) where constraints can be expressed implicitly. They rely on a decomposition of the initial linear program into the *master problem* and the *pricing problem*. The master problem corresponds to a linear program associated with a restricted constraint matrix, with respect to the number of variables (or columns) of the initial constraint matrix. The pricing problem is defined by the optimization of the so-called reduced cost (refer to [6] if not familiar with linear programing) subject to the implicit constraints expressed by the coefficients of the constraint matrix of the master problem.

The column generation solution scheme is similar to that of the simplex algorithm: it is an iterative process where, at each step, we attempt to add one or more columns to the constraint matrix of the master problem in order to improve the value of its objective function. The search for such columns is made through the solution of the pricing problem. If its outcome corresponds to one or more columns with a negative reduced cost (assuming we deal with minimization), then it entails an improvement of the value of the master objective function; otherwise, if no solution of the pricing problem can be identified with a negative cost, we then conclude that the current solution is indeed optimal.

Column generation can be combined with branch-and-bound techniques for solving integer linear programs with a large number of variables, see [7] for a nice overview. Branching rules have to be devised properly in order to avoid generating a huge number of subproblems in the search tree associated with the branch-and-bound, either by branching on the variables of the master problem using cuts, or by branching on the variables of the pricing problem using classical branching schemes or cuts.

3.3 Notations

Consider a network represented by an undirected graph $G = (V, S)$ where V is the set of nodes and S is the set of spans, indexed by s. Let D be the set of demand relations, indexed by r, where, for each demand relation, d_r denotes the number of unit demand in the bundle and, e_r^1 and e_r^2 represent its two endpoints. Finally, let c_s be the cost of span s.

4 The FIPP-SCP Iterative Heuristic (IH) Method

The FIPP-SCP Iterative Heuristic (IH) is included for further comparison with the new column generation approach. This method has not yet been published and is currently undergoing further investigation at TRLabs. It works by solving a subproblem, called FIPP-SCP IH Sub-problem, for every cycle in the network in order to determine the most efficient working route set that can be protected by that cycle. Once this is done, the most efficient cycle (i.e., with the lowest spare capacity cost per protected working route) is recorded as being part of the solution and the working route set protected by this cycle is removed form the demand set. The FIPP-SCP IH subproblem is re-solved for every cycle in the network using the new demand set. This process continues until there are no demands left to protect. Figure 4 summarizes the algorithm.

```
procedure FIPP_SCP_IH;
1   while there are unprotected demands do
2       for every cycle do
3           Solve max credit sub-problem
                (FIPP-SCP-IH Sub-Problem) using
                the remaining unprotected demands;
4       end-for;
5       Choose the cycle and route set combination
            that resulted in the highest score;
6       Remove the demands protected by this cycle
            from the remaining unprotected demands;
7       Record the cycle in the solution;
8   end-while;
end FIPP_SCP_IH;
```

Figure 4: The FIPP-SCP IH algorithm.

The following notation is introduced in order to present the FIPP-SCP IH subproblem:

SETS

D^+ = set of all non-zero demand relations, indexed by r.

P = set of eligible cycles, indexed by p.

PARAMETERS

c^p = cost of cycle p.

$$x_r^p = \begin{cases} 2, & \text{if demand } r\text{'s end-nodes are on cycle } p \text{ and} \\ & \text{its working route straddles } p; \\ 1, & \text{if demand } r\text{'s end-nodes are on cycle } p \text{ and} \\ & \text{its working route uses at least one span of } p; \\ 0, & \text{otherwise.} \end{cases}$$

$$\delta_r^s = \begin{cases} 0, & \text{if demand } r \text{ crosses span } s; \\ 1, & \text{otherwise.} \end{cases}$$

VARIABLES

$$n_r^p = \begin{cases} 1, & \text{if demand } r \text{ is protected by cycle } p; \\ 0, & \text{otherwise.} \end{cases}$$

The FIPP-SCP IH sub-problem is defined as follows:
for a cycle $p \in P$,

$$\max \sum_{r \in D^+} x_r^p n_r^p / c^p$$

$$\text{subject to:} \quad \sum_{r \in D^+} \delta_r^s n_r^p \leq 1 \qquad s \in S. \tag{1}$$

$$n_r^p \in \{0,1\} \qquad r \in D^+, p \in P \tag{2}$$

The objective function maximizes the number of demand units that the cycle protects while taking the cost of the cycle into consideration. This value represents the efficiency of the cycle. The resultant objective value is the credit score used to compare the cycles in a single iteration. The cycle with the highest credit score is the cycle that gets added to the solution. Constraints (1) ensure that the working routes sharing the same cycle for protection are mutually disjoint.

5 The FIPP-SCP DRS Model

Briefly, the FIPP-SCP DRS method introduced in [5] works by first generating a large number of disjoint route sets (DRSs) of maximum specified size. A parameter controls the smallest number of DRSs that a demand is allowed to appear in. DRSs are generated by individually considering working routes and checking if they are disjoint with the working routes already in the DRS or not. If the route being considered is disjoint from the routes in the DRS, then the route is added to the DRS and is left out of the DRS otherwise. This process repeats until a given DRS size is reached. Once all the DRSs are generated, a given number of lowest cost eligible cycles are enumerated for every DRS. The eligible

route set as well as the eligible cycle set are both given as input to the solver, which uses the FIPP-SCP DRS model to determine the lowest cost combination of DRSs and cycles that satisfy the specified demand.

The FIPP-SCP DRS ILP model uses the following additional notation:

SETS

A = set of eligible DRSs, indexed by a.

P_a = set of eligible cycles for DRS a, indexed by p.

PARAMETERS

$$x_r^p = \begin{cases} 2, & \text{if demand } r\text{'s end-nodes are on cycle } p \text{ and} \\ & \text{its working route straddles } p; \\ 1, & \text{if demand } r\text{'s end-nodes are on cycle } p \text{ and} \\ & \text{its working route uses at least one span of } p; \\ 0, & \text{otherwise.} \end{cases}$$

$$\delta_s^p = \begin{cases} 1, & \text{if cycle } p \text{ crosses span s;} \\ 0, & \text{otherwise.} \end{cases}$$

VARIABLES

n_a^p = number of unit-capacity copies of cycle p used as a FIPP p-cycle to protect DRS a.

The ILP formulation of the DRS-based FIPP p-cycle network design model (FIPP-SCP DRS) is as follows ([5]):

$$\min \sum_{s \in S} \sum_{p \in P} \sum_{a \in A} c_s \delta_s^p n_a^p$$

$$\text{subject to:} \sum_{a \in A} \sum_{p \in P_a} x_r^p n_a^p \geq d_r \quad r \in D \tag{3}$$

$$n_a^p \in \mathbb{Z}^+ \qquad a \in A, p \in P \tag{4}$$

The objective function minimizes the total cost of placing spare capacity in the network. Constraints (3) ensure that, for each demand relation r, a sufficient number of FIPP p-cycles are assigned to protect all selected DRSs of which the working route of demand r is a member.

6 A Column Generation Model

We now present a column generation model, called CG model, where, although it is a formulation with an exponential number of variables, may lead to a more amenable ILP formulation for efficient solution. Each variable will correspond to a so-called cycle configuration.

A cycle configuration C is associated with a cycle p_C, and is represented by a vector $a^C = (a_r^C)_{r \in D}$ where

$a_r^C =$ number of protection units provided by cycle p_C in configuration C for demand relation $r \in D$, where $a_r^C \in \{0, 1, ..., d_r\}$.

The cost of a configuration is defined by $\text{COST}_C = \sum_{s \in p_C} c_s$.

In order to reduce the number of configurations, we introduce the concept of maximal configurations, i.e., a p-cycle configuration C is maximal if there does not exist another p-cycle configuration C' such that $a^{C'} \geq a^C$. Using maximal configurations, we may oversatisfy the demand only when doing so does not require any additional cost over the cost of satisfying the demand exactly.

A column generation always corresponds to a decomposition of the set of constraints between the master problem and the pricing problem. Here, the master problem will include the constraints that link the configurations, i.e., the demand constraints. The pricing problem will contain the constraints that are associated to a configuration, they will be detailed below in Paragraph 6.2.

6.1 Master Problem

The variables of the master problem, denoted by z_C, are decision variables on the configurations and are defined as follows. Let $\mathcal{C}$ be the set of all configurations. For $C \in \mathcal{C}$, $z_C \in \mathbb{Z}^+$ is the number of copies of configuration C used for protection.

The master problem is formulated as follows:

$$\min \sum_{C \in \mathcal{C}} \text{COST}_C z_C$$

$$\text{subject to:} \quad \sum_{C \in \mathcal{C}} a_r^C z_C \geq d_r \quad r \in D \tag{5}$$

$$z_C \in \mathbb{Z}^+ \qquad C \in \mathcal{C}. \tag{6}$$

Constraints (5) are the demand constraints where $a_r^C z_C$ gives the demand units of demand r protected by p_C .

6.2 Pricing Problem

By definition, the pricing problem corresponds to the optimization problem of minimizing the reduced cost (with respect to linear programming definition) subject to the constraints

that must be satisfied by a given configuration, which are: definition of a cycle, identification of the demand that can be protected by the cycle, prohibition for a span to be used as a working and a protection span at the same time for the same demand.

Additional notation for the mathematical formulation of the pricing problem is as follows:

SETS

$\omega(v) = \Big\{\{v,j\} : \{v,j\} \in S\Big\}$: set of all spans adjacent to node $v \in V$.

$S(V') = \Big\{\{i,j\} : i,j \in V'\Big\}$: set of all spans whose end-nodes belong to V', $V' \subseteq V$.

$\delta(V') = \Big\{\{i,j\} : i \in V', j \in V \backslash V'\Big\}$: set of all spans whose one end-node belongs to V' and the other does not, for $V' \subseteq V$.

CONSTANTS

u_r = dual prices

c_s = cost of span s

d_r = destination of demand r

$$\rho^{rr'} = \begin{cases} 1, & \text{if working paths of demands } r, r' \\ & \text{ are non-disjoint;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\delta_s^r = \begin{cases} 1, & \text{if working path of demand } r \text{ uses span } s; \\ 0, & \text{otherwise.} \end{cases}$$

VARIABLES

$$x_s = \begin{cases} 1, & \text{if the cycle uses span } s; \\ 0, & \text{otherwise.} \end{cases}$$

$$x_s^r = \begin{cases} 1, & \text{if any protection path for demand } r \\ & \text{ uses span } s; \\ 0, & \text{otherwise.} \end{cases}$$

The mathematical formulation of the pricing problem as follows. Let us first give the expression of its objective function corresponding to the reduced cost of a column of the master problem, i.e., the cost of the cycle less the prices of the protected demands.

$$\overline{\text{COST}}_C = \text{COST}_C - \sum_{r \in D} a_r^C u_r = \overbrace{\sum_{s \in S} x_s c_s}^{\text{COST}_C} - \sum_{r \in D} \overbrace{\sum_{s \in \omega(e_r^1)} x_s^r}^{a_r^C} u_r.$$

Let us now express the set of constraints of the pricing problem.

$$\min \overline{\text{COST}}_C$$

subject to:

$$\sum_{s\in\omega(v)} x_s \leq 2 \qquad v \in V \tag{7}$$

$$\sum_{s'\in\omega(v):s'\neq s} x_{s'} \geq x_s \qquad v \in V, s \in \omega(v) \tag{8}$$

$$\sum_{s\in\delta(V')} x_s \geq x_{s'} + x_{s''} - 1 \; V' \subseteq V,$$

$$s' \in S(V'), s'' \notin S(V') \tag{9}$$

$$x_s \geq x_s^r \qquad s \in S, r \in D \tag{10}$$

$$\sum_{s\in\omega(e_r^1)} x_s^r = \sum_{s\in\omega(e_r^2)} x_s^r \qquad r \in D \tag{11}$$

$$\sum_{s\in\omega(v)} x_s^r \leq 2 \qquad r \in D, v \in V \tag{12}$$

$$\sum_{s'\in\omega(v)|s'\neq s} x_{s'}^r \geq x_s^r \qquad r \in D, v \in V\backslash\{e_r^1, e_r^2\},$$

$$s \in \omega(v) \tag{13}$$

$$2 - x_s^r - x_s^{r'} \geq \rho^{rr'} \qquad s \in S, (r, r') \in D^2 : r \neq r' \tag{14}$$

$$x_s^r \leq 1 - \delta_s^r \qquad e \in S, r \in D \tag{15}$$

$$x_s^r \in \{0, 1\} \qquad s \in S, r \in D \tag{16}$$

$$x_s \in \{0, 1\} \qquad s \in S \tag{17}$$

The first three sets of constraints take care of the construction of the cycle. Constraints (7) and (8) ensure flow circulation on the cycle variables, i.e., they address the cycle construction by stating that, at all nodes, the number of incoming/outgoing flows must be either 0 (p-cycle does not go through node v) or 2 (p-cycle contains node v and two adjacent spans such that $x_s = 1, s \in \omega(v)$). Constraints (9) prevent subcycles, i.e., p-cycle made of more than one cycle. Those constraints are a variant of the classical subcycle elimination constraints of the Traveling Salesman Problem (TSP), with the difference that a p-cycle does not necessarily include all nodes while a TSP tour must include all nodes exactly once. A drawback of this formulation is that there is an exponential number of subcycle elimination constraints. Notice, however, that the pricing problem DOES NOT need to be solved exactly at each iteration, and if one is able to design an efficient heuristic to solve it, we may need to solve exactly the pricing problem only once to confirm that the heuristic has indeed found the optimal solution.

The following sets of constraints will define the protection path. Constraints (10) establish the relationship between the set of variables associated with the cycle, and those associated with the protection path. The next three sets of constraints correspond to the definition of a protection path. These constraints address the flow between the two end nodes of a demand relation and ensure that the amount of flow is equal to the protection amount (it can be only a fraction of the number of unit demand of the demand relation) provided by the cycle under construction. Constraints (11) state that the flow exchanged between the end-nodes must be equal, i.e., a protection path must end at both end-nodes of a demand. At the end-node, constraints (12) allow at most two protection paths for each demand relation. At intermediate nodes, constraints (12) and (13) together ensure flow conservation, i.e., the number of incoming/outgoing flows must be either 0 (protection path of demand relation r does not use node v) or 2 (protection path contains node v and two adjacent spans such that $x_s^r = 1, s \in \omega(v)$).

The last two sets of constraints take care of the exclusive use of a given span in either working or protection path. Constraints (14) state that non-disjointly routed demands cannot share a span for their protection. Constraints (15) prevents from using a given span in both working and protection paths of each demand relation.

6.3 Solving the CG Model

In order to solve the CG model, we propose to use a branch-and-bound method assuming the linear programing relaxation of the master problem is solved using column generation techniques. Let us first address the issue of solving the linear programming relaxation of the master problem, i.e., the problem described in 6.1 with constraints (6) relaxed to

$$z_C \in [0, 1] \qquad C \in \mathcal{C}.$$

The first difficulty lies in the solution of the pricing problem. As we mentioned before, that the pricing problem does not need to be solved exactly at each iteration of the column generation process, as long as we are able to find a configuration with a negative reduced cost in order to be able to iterate. The pricing problem can be described as multi-commodity flow problem with side constraints corresponding to the definition of a cycle (that is usually not a Hamiltonian cycle) on which the flows circulate. Constraints (9) that prevent from considering sub-cycles are very costly as there are an exponential number of them. In order to overcome this difficulty, we introduce them only as needed following the principle of the "lazy constraints" feature of CPLEX™: it means that the pricing problem is solved iteratively, starting with no (or a very small number of) sub-cycle constraints, and adding some sub-cycle constraints that are violated by the final solution if any. In order to save computing time, we also stop the solution of the pricing problem as soon as we obtain a solution (i.e., a configuration) with a negative reduced cost. Therefore, in the iterative process with the lazy constraint like feature, the final solution is the first found solution with a negative reduced cost. In practice, we need to introduce a very small number of sub-cycle constraints, and therefore the computing cost of iterative process which leads

to possibly several solutions of the pricing problem is counter balanced by the smaller sizes of the pricing problems that need to be solved. Indeed, in practice, we very rarely needed to solve the pricing problem more than twice. Note that we never eliminate any sub-cycle constraints that have been introduced. Consequently their number increases as we go further down in the branch-and-bound search tree. Although, we could eliminate some of them, it offers the advantages that in practice, most of the time, at each iteration, the solution of the first pricing problem is very often feasible, i.e., satisfies all sub-cycle constraints even if only a small number of them have been explicitly introduced in the pricing problem.

It was observed that the number of columns generated for exactly solving the linear programming relaxation of the master problem was rather limited (see the details in Section 7). Because of this we did not develop any branch-and-bound to get integer solutions for the master problem, but instead we used CPLEX™ to solve the ILP restricted master problem with the set of columns generated for the exact solution of its linear programming solution. Although we cannot claim that we have obtained the optimal solution of the CG model, the obtained solutions are already quite satisfactory in comparison with those obtained with previous models, as it is shown in Section 7.

7 Computational Results

The solutions for the FIPP CG, FIPP SCP IH and FIPP SCP DRS design models were obtained by implementing the prior model in C++ and the latter two models in AMPL 9.0. The FIPP SCP IH and FIPP SCP DRS models were solved using CPLEX 9.0 MIP solver and the results obtained are based on complete terminations with a MIPGAP of 0.01. The FIPP CG model was solved using CPLEX 10.0 MIP solver. The models were run using the COST239 European network [8] containing 11 nodes and 26 spans as well as a 15 node family of networks with spans varying from 16 to 30 [9]. The number of demand relations is 55 for the COST239 instance, and 105 for all instances of the 15 node family.

For the FIPP SCP DRS solutions provided in this paper, 30 DRSs were generated per demand in the network. This means that every demand appeared in at least 30 different DRSs. For every DRS, 3 lowest cost cycles eligible to protect that DRS were enumerated and added as input to the problem. The maximum number of working routes per DRS was set to 12. Additionally, to remain true to the method used in [5], a single route DRS was generated for every demand to counteract the effects of any strong forcer demands. In other words this was done to prevent any particularly high demand from forcing the solver to place additional large cycles where it could instead use a small dedicated cycle to protect that demand. The FIPP SCP IH solutions were obtained by running the FIPP SCP IH algorithm where all possible cycles in the network were considered as eligible candidates.

The FIPP-CG method uses the results obtained by FIPP-SCP IH heuristic as starting feasible solutions. The results for this method are summarized in Table 1. The table reports the final cost of spare capacity as well as the gap against optimality, i.e., the gap

Table 1: Results obtained by the FIPP-CG Algorithm

Problem Instances	Cost	Gap (%)	# Unit Cycles	# Distinct Cycles	Overall # Configurations $z_C > 0$	Overall # Configurations $z_C = 0$	# Non-disjoint Configurations	Overall # Cycles
COST239	68840	2.5	17	13	15	534	6	286 (20)
15n30s1-16s	387958	0.0	185	3	73	169	0	3 (3)
15n30s1-17s	286901	0.2	140	6	72	217	18	7 (6)
15n30s1-18s	266080	0.2	126	7	60	350	22	11 (10)
15n30s1-19s	229854	0.1	108	11	59	338	20	21 (16)
15n30s1-20s	217963	0.1	108	13	64	472	22	31 (23)
15n30s1-21s	192269	0.4	101	17	63	412	26	50 (27)
15n30s1-22s	182550	0.6	101	19	66	411	30	72 (34)
15n30s1-23s	177432	0.5	100	27	61	505	32	111 (45)
15n30s1-24s	175990	0.6^-	100	29	58	636	23	178 (45)
15n30s1-25s	157140	0.9^-	92	36	56	504	21	196 (43)
15n30s1-26s	106526	6.1^-	55	39	45	395	18	287 (39)
15n30s1-27s	117220	5.5^-	61	41	45	399	28	287 (51)
15n30s1-28s	114055	4.5^-	59	44	46	333	23	303 (44)
15n30s1-29s	106449	2.6^-	56	33	42	534	28	345 (39)
15n30s1-30s	110738	1.0^-	62	40	51	473	34	326 (38)

between the optimal solution of the linear relaxation of the master problem and the best known solution. In some cases, where the method reached a running time limit of 5 hours for solving the linear relaxation, this gap is only an under-estimation (values followed by $^-$). We also provide the total number of unit cycles used in the final solution, i.e., $\sum_{C \in \mathcal{C}} z_C$, along with the number of distinct cycles used (of which there could be more than one instance of). Average length of the cycles in the final solution varies between 9.5 and 11.6 spans for the 15 node family. In the following two columns, we indicate the overall number of generated configurations by distinguishing between the number of null and positive master variables (z_C). The second last column contains the number of configurations which take advantage of non-disjointness. We observe than about half of the configurations are associated with non pairwise disjoint working paths. Finally, the last column contains the overall number of generated cycles (the number in parenthesis corresponds to the distinct number of cycles in the initial solution). Note that the initial set of columns provided to the CG model (in parenthesis in the last column) is composed of the cycles and configurations deduced from the best solution of the FIPP-SCP IH model. Although using these solutions has no impact on the solution of the CG model (assuming that no time limit is set), it certainly speeds up the convergence, allowing the generation of more meaningful values for the dual variables, hence the generation of better configurations. In terms of the number of configurations in

the best solution (as indicated by the number of configurations such that $z_C > 0$), it is usually rather close to the number of cycles except for the first instances of the 15 node family as the number of eligible cycles is very limited for the first three instances. Although we can certainly look at an enhanced, more compact column generation model where each configuration would be associated to a cycle and vice-versa, it will not significantly reduce the number of generated columns.

The results for FIPP SCP DRS and FIPP SCP IH are documented in Tables 2 and 3, respectively. These tables contain the final design costs along with the number of unit cycles as well as the number of distinct cycles used in these solutions. Furthermore the relative cost difference to the CG solutions is also reported in the last column of these tables. Note that Table 2 only contains the results for the 15m30s network family up to the 23rd span instance. It was at this point that the solver did not return with a solution within the specified MIPGAP due to the large increase in the number of constraints due to the large number of parameters used.

One of the weaknesses of the DRS method is that it does not scale very well with the increase in network size. The more DRS/cycle combinations there are to consider, the harder it becomes to get the DRS method to solve the problem to completion. It is possible to reduce the number of parameters, but this severely degrades the quality of the solution that the solver arrives at. The DRS method is capable of reaching the optimal solution, but only if all possible combinations of cycles and DRSs are given to it as input. This cannot be done for anything larger than the simplest network because of the huge number of DRS/cycles possible even for a medium sized network. Therefore whenever parameters are introduced, the DRS method does not yield an optimal solution, in general, but only the optimal combination of the DRSs and cycles given to it as input.

Table 2: FIPP-SCP DRS Results

Problem Instances	Cost of spare	# Unit Cycles	# Distinct Cycles	Cost FIPP-CG (%)
COST239	93345	24	17	36 %
15n30s1-16s	393094	189	3	1 %
15n30s1-17s	305065	146	6	6 %
15n30s1-18s	272661	127	8	2 %
15n30s1-19s	246035	110	12	7 %
15n30s1-20s	239942	113	17	10 %
15n30s1-21s	208501	101	26	8 %
15n30s1-22s	202771	101	40	11 %
15n30s1- $\geq$23s	-	-	-	-

Table 3: FIPP-SCP IH Results

Problem Instances	Cost of spare	# Unit Cycles	# Distinct Cycles	Cost FIPP-CG (%)
COST239	94095	25	20	37 %
15n30s1-16s	396935	192	3	2 %
15n30s1-17s	310094	152	6	8 %
15n30s1-18s	291047	139	10	9 %
15n30s1-19s	259825	129	16	13 %
15n30s1-20s	246505	124	23	13 %
15n30s1-21s	215708	114	27	12 %
15n30s1-22s	198789	104	34	9 %
15n30s1-23s	199378	106	45	12 %
15n30s1-24s	201207	107	45	14 %
15n30s1-25s	179897	100	43	14 %
15n30s1-26s	115848	60	39	9 %
15n30s1-27s	137887	74	51	18 %
15n30s1-28s	128957	71	44	13 %
15n30s1-29s	126745	73	39	19 %
15n30s1-30s	132294	79	38	19 %

As an alternative to the FIPP SCP DRS method, the iterative heuristic method was introduced. It was possible to get solutions for the entire 15n30s network family using this method as documented in Table 3.

It can be observed that the spare capacity cost of the CG method is lower than both the DRS and the IH method solutions and that the DRS and the IH methods are relatively close to each other in cost. The cost improvement when comparing the DRS and IH results to the FIPP CG solutions is as much as 19% for the 15n30s and is 37% for the cost 239 network. Also it can be seen that the relative difference to the CG solutions increases as the number of spans is increased from 16 to 30 in the 15n30s family of networks. Taking into account that around half of the configurations in the final solution of the CG method take advantage of non disjoint working paths, it is unclear at this point whether the improved solutions are due to this effect or to the global search scheme entitled by the column generation method.

The benefit of the CG method is twofold: it is able to consider a more general view of the route sets that can be protected while also being able to arrive at nearly optimal solutions by being able to consider only the configurations that reduce the objective value of the master problem. The DRS method considers not only a smaller range of route combinations, but also only a small subset of all possible combinations and is thus not able

to achieve the efficiencies of the CG method. Likewise, the IH method also only considers route combinations that are disjoint from one another and while is able to optimally determine the best disjoint route combination for a given cycle that maximizes the credit score, it is unable to consider the problem as a whole and thus does not gain any benefit from the interdependencies of the cycles and their protected sets.

8 Conclusion

We have investigated a first column generation (CG) model for the efficient design of FIPP p-cycles, and compared it successfully against the FIPP-CSP DRS method as well as the FIPP-SCP IH heuristic. While previous FIPP-SCP methods work under the assumption that working paths are pairwise disjoint, this is not the case for the FIPP-CG model. This, in addition to the fact that the CG method is capable of generating near optimal solutions, can provide us with insight into what a really good FIPP solution actually looks like and and can thus help with the invention and improvement of heuristic methods. Despite the quality of the solutions presented, there is still room for improvement of the CG-model. In particular it could be sped up if a heuristic for the pricing problem, that is able to generate configurations with negative reduced cost, is found. Additionally, a more compact/exact CG model can still be designed.

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 *IEEE International Conference on Communications (ICC 1998)*, June 1998, pp. 537–543.

[2] S. Sengupta and R. Ramamurthy, "Capacity efficient distributed routing of mesh-restored lightpaths in optical networks," in *Proceedings of IEEE Global Telecommunications Conference (GlobeCom 2001)*, San Antonio, TX, November 2001, pp. 2129–2133.

[3] R. Freeman, *Fiber-Optic Systems for Telecommunications*. New York: Wiley, 2002, ch. 6 and 10.

[4] A. Kodian and W. Grover, "Failure-independent path-protecting p-cycles efficient and simple fully preconnected optical-path protection," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3241–3259, October 2005.

[5] A. Kodian, W. Grover, and J. Doucette, "A disjoint route sets approach to design of failure-independent path-protection p-cycle networks," in *DRCN Workshop*, October 2005, 8pp.

[6] V. Chvatal, *Linear Programming*. Freeman, 1983.

[7] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, no. 3, pp. 316–329, May-June 1998.

[8] P. Batchelor et al., "Ultra high-capacity optical transmission networks: Final report of action COST 239," Faculty of Electrical Engineering and Computing, University of Zagreb, Tech. Rep., 1999.

[9] J. Doucette, "Advances on design and analysis of mesh-restorable networks," Ph.D. dissertation, University of Alberta, Edmonton, AB, Canada, 2004.