Shared Risk Resource Groups and Colored Graph: Polynomial Cases and Transformation Issues

David Coudert, Stéphane Pérennes, Hervé Rivano, Marie-Emilie Voge

Shared Risk Resource Groups and Colored Graph : Polynomial Cases and Transformation Issues [*]

David Coudert, Stéphane Perennes, Hervé Rivano, and Marie-Emilie Voge
MASCOTTE Project, INRIA-I3S(CNRS/UNSA)
2004 Route Des Lucioles, B.P. 93 – 06902 Sophia-Antipolis Cedex – France
{*name*}@sophia.inria.fr

September 26, 2007

Abstract

In this paper, we characterize polynomial cases for several combinatorial optimization problems in the context of multilayer networks with shared risk resource groups.

Keywords: Reliability, Shared Risk Resource Group, colored graphs, complexity, approximability.

1 Introduction

Since the end of monopolistic national telecommunication operators, no company own a backbone network from the physical conduits buried underground to the Label Switched Path (LSP) topology actually carrying data trunks. Transporting a packet on the Internet thus involves many networking operators renting resources to each other. In this settings, breaking a conduct carrying terabits per second would lead to the loss of a significant amount of data and have a very strong impact on the revenue of many economic actors.

These complex interactions between independent actors yields a multilayered structure on the network: the optical fiber topology of a Wavelength Division Multiplexing (WDM) network is placed into rented conduits where disjoint fibers might share a conduit; at the application level, an overlay network, e.g. interconnecting a distributed web services platform, has virtually no guaranty that two disjoint virtual links are carried by disjoint paths.

In these settings, the design of survivable networks is more challenging than ever. Shared Risk Resource Group (SRG) has been introduced to capture the concept of a set of resources that would fail down simultaneously in case of a given network incident [1, 8, 9]: a conduit cut would bring several fibers down, each of them tearing off many LSPs, and so on, until the overlay network is broken into several connected components. In such an example, the set of LSPs failing down simultaneously would belong to the same SRG. The design of the overlay would have been non-survivable with, even though each layer might appear individually survivable.

[*]The research reported in this paper has been partially supported by ANR JC OSERA, European projects IST FET AEOLUS and COST 293 Graal, and has been done in the context of the CRC CORSO with France Telecom.

Survivable network design involves many NP-hard problems, even if considering mono-layer networks and has fostered deep graph theoretic and approximation algorithmic analysis.On the other hand, the theory of survivable network relies on basic combinatorial object that are quite easy to compute in the classic graph models for networks: shortest path, minimum cut, ...

Shared Risk Resource Groups twist the deal. Since links or nodes that may be arbitrary unrelated can share a common risk, the graph structure of the network is broken which makes it difficult to compute even basic objects such as path crossing a minimum number of SRGs, or the minimum number of SRGs that disconnect the network. A graph theoretic approach has been developed [3]identifying SRG with colors assigned to the edges or vertices of a graph. Approximability of the underlying combinatorial problems has then been investigated, with a proof of their deep complexity.

As a consequence of this complexity, many heuristic approaches have been proposed in the literature. (**state of the art heuristics**).

In this paper, we address this complexity issue with a complementary view point. Indeed, practical tests give the intuition that many situation are "not so hard" to solve, where problems appear to be computable in polynomial time. We believe that a deep and precise analysis of this polynomial cases may give enough insights and subroutines to solve more efficiently the hard cases.

1.1 Motivations of the paper

The aforesaid optimization problems related to multilayer network survivability are momentous in the future development of such networks and they have to be solved efficiently. Unfortunately, it was shown in [] that they are NP-hard and hard to approximate in the general case. However there are still some hopes since some special cases were shown to be polynomial. As we will show in section 2 there are other interesting polynomial cases with regard to real networks for the MC-Path and MC-*st*-Cut problems. In addition, modeling the multilayer networks with colored graphs as in [] necessitates a simple, but not without repercussions, transformation from the network to the model. We detail in section 3 what are exactly these repercussions and take advantage of them to obtain polynomial cases when it is possible. Furthermore, we touch on an other point which is worth noting about the colored graph model : if the optimization problems are NP-hard and hard to approximate for the colored graphs general class, there may still be some hopes for the specific class of colored graphs obtained from multilayer networks. Indeed we show that there are colored graphs that cannot be obtained from realistic multilayer networks. Consequently there may be other polynomial or at least approximable cases, yet unknown, for the problems of interest depending on specific properties of this subclass of colored graphs.

2 A new polynomial case for MC-Path, MC-*st*-Cut and MC-Cut

According to previous works, the case when the number of colors of span greater than 1 is bounded is of great interest : only a few colors have span > 1 in the graphs used to test the solving methods in the literature. See for example the inevitable networks COST239, NJLATA [2, 4] and NSFNET [7, 5, 13] or those from [4], [6] and [10].

Therefore in this section we investigate the consequences of this hypothesis on the complexity of some colored problems. We prove that when the number of colors of span greater than 1 is bounded the MC-Path and MC-*st*-Cut problems are polynomial and consequently MC-Cut too.

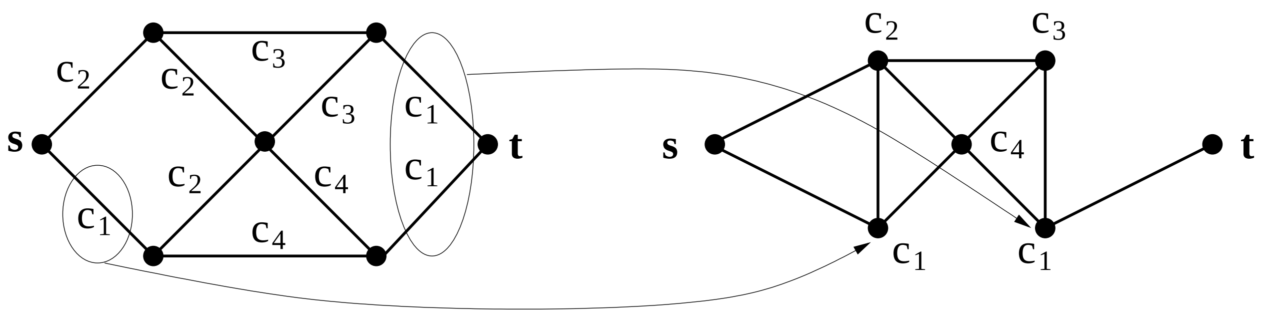


Figure 1: Transformation from G to H.

We also show that unfortunately 2-CDP is still NP-complete.

For both MC-Path and MC-*st*-Cut, the proof is based on a polynomial size MILP formulation of the problem. These formulations contains rather few binary variables and any solver can find an optimal solution in a reasonable time provided $|\mathcal{C}|$ remains *small*. However simple they look, they are not solvable in polynomial time since MC-Path and MC-*st*-Cut are NP-Hard. In case the number of colors of span greater than 1 is bounded, we show that only a bounded number of variables needs to be binary, and by the way, these problems are polynomial.

Theorem 1 *When the number of colors of span greater than 1 is bounded the MC-Path, MC-st-Cut and MC-Cut problems are polynomial.*

2.1 Minimum Color Path

For the MC-Path problem, the formulation is a flow like formulation based on a graph $H = (V_H, E_H)$ constructed from the colored graph $G = (V, E, \mathcal{C})$ in which we search for a minimum color *st*-path. Each vertex of H represents a connected component of some color of $\mathcal{C}$. Two vertices of V_H are joined by an edge of E_H if the two associated components have a common vertex in G. Two vertices s and t are added to V_H and an edge of E_H connects s (resp. t) to a vertex v of V_H if s (resp. t) belongs to the component represented by v as illustrated on Figure 1.

Trivially, any colored *st*-path of H is a colored *st*-path in G and vice versa. Although this transformation is not usefull to formulate the MC-Path problem as a MILP, it is necessary to relax some binary variables and prove the polynomiality of the problem.

Formulation 1 is based on a slightly modified classical flow formulation : there are no capacities on the edges and some binary variables are added. To find a path, a single unit of flow is to be routed in the graph H between s and t. The variable $w_e \geq 0$ represents the flow value on edge $e \in E_H$. The binary variable y_c is associated to color $c \in \mathcal{C}$, its value is 1 if the flow crosses at least one vertex corresponding to color c and 0 otherwise. In this formulation $\delta^-(u)$ is the set of edges supporting flow entering the vertex u and $\delta^+(u)$ the set of edges supporting flow leaving vertex u. The first constraint forces the existence of the flow between s and t and the second one insures that if a vertex of color c is crossed by the flow, then color c belongs to the colored path computed. The objective to be minimized is the number of colors crossed by the flow.

Formulation 1 **(MC-Path)**

$$
\begin{array}{lrcl}
 & \textit{Minimize} & \sum_{c\in\mathcal{C}} y_c & \\
\textit{s.t.} & \displaystyle\sum_{e\in\delta^+(u)} w_e - \sum_{e\in\delta^-(u)} w_e & = & \begin{cases} 0 & \textit{if} \quad u \neq s, u \neq t \\ 1 & \textit{if} \quad u = s \\ -1 & \textit{if} \quad u = t \end{cases} \qquad \forall u \in V_H \\
 & y_c & \geq & w_e \quad \forall c \in \mathcal{C},\ \forall e = \{u,v\}\ \textit{s.t.}\ u\ \textit{or}\ v \in c \\
 & w_e & \geq & 0 \quad \forall e \in E_H \\
 & y_c & \in & \{0,1\} \quad \forall c \in \mathcal{C}
\end{array}
$$

In the Formulation 1, the variables w_e need not be integer since even if the flow is divided onto several paths, they all necessarily use the same optimal set of colors, otherwise we could remove some colors from the solution and still get a path thanks to the absence of edge capacities.

In addition, we can relax the binary constraint on the variables y_c corresponding to colors of span 1. Assume we obtain an optimal solution in which at least one y_c variable is not binary, this means that the flow is divided onto several paths. These paths are necessarily of same cost otherwise contradicting the optimality of the solution. We can then arbitrarily chose any one of these paths to obtain an optimal colored path.

As a consequence, when the number of colors of span greater than 1 is bounded, there remain only a bounded number of binary variables in our formulation, that can then be solved in polynomial time. Hence, in this special case, the MC-Path problem is polynomial.

2.2 Minimum Color *st*-Cut problem

To complete the proof of Theorem 1, we present in this section a MILP formulation and a simple algorithm producing in polynomial time a minimum colored *st*-cut.

The following MILP formulation expresses MC-*st*-Cut as an assignment of *potentials* to the vertices of the graph through the non negative fractional variables $x_u,\ u \in V$.

The graph is partitioned into disjoint subsets according to vertex potentials : two vertices u, v belong to the same subset if and only if they have equal potentials $x_u = x_v$.

If two adjacent vertices u and v have distinct potentials, $x_u \neq x_v$, then the color c of the edge connecting them belongs to the colored cut, that is the binary variable y_c associated to c is forced to have value one (line 2).

Line 4 ensures that vertices s and t are disconnected by the colored cut since they are assigned distinct potentials.

$$
\begin{array}{rll}
\text{Minimize} & \sum_{c\in\mathcal{C}} y_c & \\
\text{s.t.} & y_c \geq |x_u - x_v| & \forall c \in \mathcal{C},\ \forall u, v \in V \text{ adjacent to } c \\
 & x_u \geq 0 & \forall u \in V \\
 & x_s = 0 \text{ and } x_t = 1 & \\
 & y_c \in \{0,1\} & \forall c \in \mathcal{C}
\end{array}
$$

Note that the objective function can be replaced with $\sum_{c\in\mathcal{C}} w_c y_c$ to handle the weighted case, when $w_c \geq 0$ is a weight associated to color c.

As in the colored path case, if the number of colors of span greater than 1 is bounded, we relax the binary constraint on the y_c variables associated to colors of span 1. This relaxed MILP can be solved in polynomial time, and from its solution it is easy to derive an optimal colored st-cut as follows.

Once the MILP is solved, each vertex of the graph is assigned a potential $x_u \in [0,1]$. Let $p_0, p_1, \ldots, p_k$ be the different potential values existing in the graph such that $0 = p_0 < p_i < p_{i+1} < p_k = 1$. Then, let i be *any* integer in $\{0, \ldots, k-1\}$ and $V_0^i = \{u | x_u \leq p_i\}$ be the set containing vertices whose potential is not greater than p_i. As Lemma 1 states, the st-cut $(V_0^i, V - V_0^i)$ is an optimal colored st-cut.

Lemma 1 *Let $p_0, p_1, \ldots, p_k$ be the different potential values existing in the graph such that $0 = p_0 < p_i < p_k = 1$ and let $V_0^i = \{u | x_u \leq p_i\}$ for $i \in \{0, \ldots, k-1\}$. The colored st-cut given by $(V_0^i, V - V_0^i)$ $\forall i \in \{0, \ldots, k-1\}$ is an optimal colored st-cut.*

Proof : According to their potential values, the vertices of V can be portioned into several subsets : let $V_i \subset V$ be the set of vertices whose potential equals p_i. Note that $s \in V_0$ and $t \in V_k$.

The idea of the proof is to iteratively merge the aforesaid subsets V_i by modifying their vertex potentials until there remain only binary potentials and only two subsets, $V_0 \ni s$ and $V_k \ni t$, in the graph.

More precisely, let i be any integer in $\{1, \ldots, k-1\}$, we show that the objective value computed by the MILP is not modified if we either increase the potential p_i of V_i up to p_{i+1} to merge V_i and V_{i+1}, or decrease it down to p_{i-1} to merge V_i and V_{i-1}.

To prove this we define p^c_{max} and p^c_{min} to be respectively the maximum and minimum potential values among vertices that are adjacent to color c.

First, for a color c such that $p^c_{max} = p^c_{min}$, the relaxed variable y_c representing the belonging to the cut of color c equals $p^c_{max} - p^c_{min} = 0$: it has already a binary value. Furthermore, all vertices adjacent to color c have an equal potential value $p_j = p^c_{max} = p^c_{min}$ and belong to the same subset V_j of the partition. Even if their common potential value is modified (case $j = i$), the vertices still belong to the same subset after the modification of p_i, which has to be in this case either V_{i-1} or V_{i+1}. We can deduce from this, that when the value of y_c reach 0, it can not be modified by any subset merging.

In addition, note that the only vertices concerned by the modification of p_i are indeed the vertices of potential value p_i. Thus, in case both p^c_{max} and p^c_{min} are different from p_i, they are not modified, which implies that y_c is not modified either for color c.

Then, the most important part of the proof concerns the colors for which either $p^c_{max} = p_i$ or $p^c_{min} = p_i$. Indeed, the potential of vertices of such colors are necessarily modified at iteration i, which consequently increases some y_c values and decreases some others. Table 1 details these modifications on the y_c for both type of colors and for both modifications of the value p_i. For example, let c be a color such that $p^c_{min} = p_i$. If p_i is decreased down to p_{i-1}, y_c is increased by $p_i - p_{i-1}$, while if p_i is increased up to p_{i+1}, then y_c is decreased by $p_{i+1} - p_i$. It is worth noting that in this very later case, after the modification of p_i, $p^c_{min} = p_{i+1}$ since the potential values $p_0 < p_i < p_{i+1} \leq p_k$ are ordered and there is no potential value between p_i and p_{i+1}. The same reasoning applies for colors such that $p_i = p^c_{max}$ in case p_i is decreased down to p_{i-1}.

What appears in Table 1 is that if there are more colors such that $p^c_{min} = p_i$ than such that $p^c_{max} = p_i$ then p_i can be increased up to p_{i+1} without increasing the objective value which can even be decreased, while in the opposite case, p_i can be decreased down to p_{i-1} with the same effect.

	$p_i \leftarrow p_{i-1}$	$p_i \leftarrow p_{i+1}$
$p^c_{min} = p_i$	$+(p_i - p_{i-1})$	$-(p_{i+1} - p_i)$
$p^c_{max} = p_i$	$-(p_i - p_{i-1})$	$+(p_{i+1} - p_i)$

Table 1: Effects of the possible operations on the y_c value for relevant colors when p_i is modified.

Note however that there must be an equal number of colors on each side since otherwise the objective value can be decreased by a modification of p_i. As a consequence, the binary solution obtained is strictly better than the mixed one given by the MILP computation whose optimality is then contradicted. This remark easily extends to the weighted case.

To conclude, for any integer $i \in \{1, \dots, k-1\}$ we can increase or decrease the value p_i to merge V_i with either V_{i+1} or V_{i-1} without modifying the objective value obtained by the MILP. Then, by iteratively merging the subset V_j we obtain only binary potential values, binary y_c values and two subsets $V_0 \ni s$ and $V_k \ni t$ which define a cut and by the way a colored *st*-cut. Since the cost of this cut is the same as the cost of the mixed solution given by the MILP it is necessarily an optimal colored *st*-cut. In addition the merging rule for V_i, that is either with V_{i+1} or V_{i-1}, implies that the final subsets V_0 and V_k can be directly obtained by the choice of any integer $i \in \{1, \dots k-1\}$ such that $V_0 = \{u | x_u \leq p_i\}$ and $V_k = \{u | x_u > p_i\}$. Finally there are at most $|V|$ subsets at the beginning since there are $|V|$ vertices in the graph, which means that the merging process can be done in polynomial time.

□

In a nut shell, solving the relaxed MILP is polynomial since it contains only a bounded number of binary variables. Obtaining binary values for the relaxed variables from this solution can also be done in polynomial time. Consequently the MC-*st*-Cut problem is solvable in polynomial time when the number of color of span greater than 1 is bounded.

2.3 2-Color Disjoint Paths

The 2-Color Disjoint Paths problem being a very important issue in multilayer network survivability, it would be very convenient if it was polynomial too when the number of color of span > 1 is bounded. Unfortunately as the following theorem states, it is not the case.

Theorem 2 *There is a polynomial time reduction from 3SAT to the 2-Color Disjoint Paths problem. Consequently, finding two color disjoint paths in a colored graph is NP-hard.*

Proof : Consider an instance of 3SAT with n variables x_i, $1 \leq i \leq n$, and m clauses C_j, $1 \leq j \leq m$, each being a disjunction of at most three literals. The 3SAT problem consists of finding a truth assignment that satisfies all the m clauses.

From this instance we construct an instance of the 2-Color Disjoint Paths problem as follows. First let's define the color set $\mathcal{C}$. It contains a color c_1 which one of the two paths will necessarily use and two colors, p^i_{jk} and p^i_{kj}, for each pair of clauses (C_j, C_k) such that x_i belongs to C_j and $\overline{x_i}$ belongs to C_k. We will add some colors of less interest later.

The colored graph $G = (V, E, \mathcal{C})$ can be divided in two parts, each one connecting the vertices $s \in V$ and $t \in V$. The first part represents the clauses and is thus divided into m sections delimited by $m+1$ vertices $u_1 = s, u_2, \dots, u_{m+1} = t$. For each $j \in \{1, \dots, m\}$ and for each literal x_i (resp.

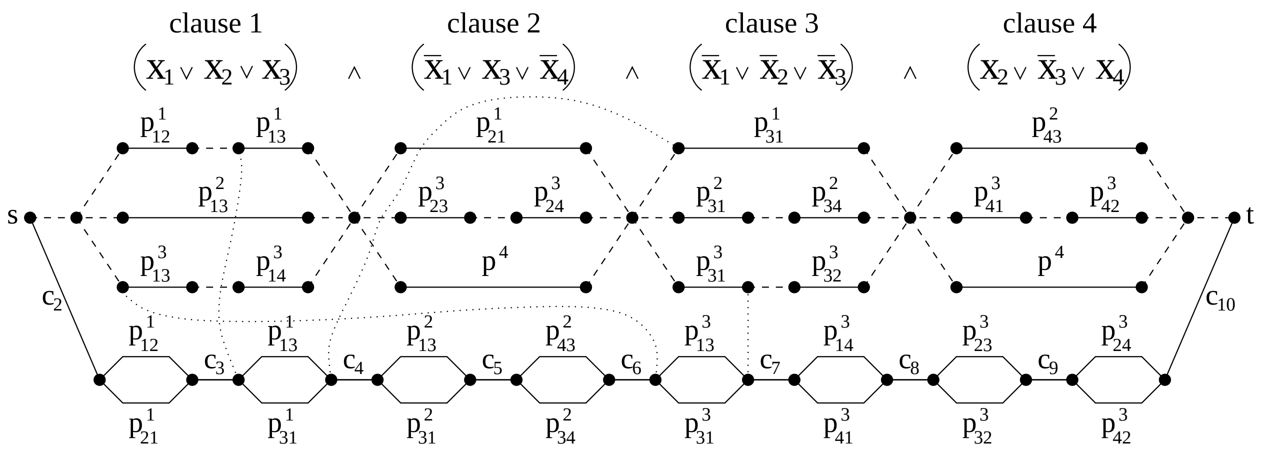


Figure 2: Reduction from 3SAT to 2-Color Disjoint Paths. Only four crossing edges are represented (color p^1_{13}, p^1_{31}, p^3_{13} and p^3_{31}).

$\overline{x_i}$) belonging to C_j there is a path connecting u_j to u_{j+1} containing one edge of each color p^i_{jk} (resp. p^i_{kj}) such that C_k contains $\overline{x_i}$ (resp. x_i). Between each of these edges we insert an edge of color c_1. In addition, this $u_j u_{j+1}$-path has to start and end with color c_1 as shown on Figure 2.

The second part of the graph is a kind of path composed of an alternating succession of simple and multiple edges. This path is long enough to contain a multiple edge for each variable x_i and for each pair of clauses (C_j, C_k) such that x_i belongs to C_j and $\overline{x_i}$ belongs to C_k. Such a multiple edge consists of two parallel edges of color respectively p^i_{jk} and p^i_{kj}. The simple edges all have a distinct and new color reduced to one edge and the path begins and ends with such simple edges.

The two parts are connected in s and t but also through each color of the p^i kind. Indeed, note that each of these colors has span 2 with exactly one edge in the first part of the graph and one edge in the second. Since we want these color to be of span 1, we add an edge of color p^i_{xy} between one extremity of the first occurrence of the color p^i_{xy} and one extremity of its second occurrence. In the following we refer to such edges as *crossing edges*.

Now that the colored graph is complete, we can try to find two color disjoint st-paths. The first remark is that since s and t both have degree two with one edge of color c_1 in the first part and an other edge in the second part of the graph. Thus if there are two color disjoint st-paths, there is necessarily one (*the first path*) that begins and ends with color c_1 in the first part of the graph, and an other one (*the second path*) which begins and ends in the second part.

Then, since the first path has to use color c_1 and since each edge of the p^i color kind is inserted between two edges of color c_1, the second path can not use any crossing edge in order to go on in the first part : its progress is immediately blocked by color c_1. Consequently, according to the topology of the second part, the second path has to use all the simple edges of the second part of the graph and exactly one color from each pair (p^i_{xy}, p^i_{yx}). This also implies that the first path cannot use any crossing edge either because then it has to use either a simple edge of the second part or an other crossing edge of color p^i_{xy} associated to the color p^i_{yx} of the crossing edge it has used to come in the second part.

Therefore, we have established that if two color disjoint st-paths exist in the graph we have constructed, one of them is contained in the first part of the graph, and the other in the second part. We now need to prove that if there are two color disjoint st-paths in the graph, there is a truth assignment satisfying all clauses of the instance of 3SAT.

Assume that the first path uses the subpath corresponding to literal x_i (resp. $\overline{x_i}$) between u_j ans u_{j+1}. Then it uses all the color p^i_{jk} (resp p^i_{kj}) such that x_i belongs to C_j and $\overline{x_i}$ belongs to C_k. This implies that the second path has to uses all the color p^i_{kj} (resp. p^i_{jk}) to reach t and be color disjoint from the first path. Consequently, the first path cannot use these last colors and thus cannot use the subpath corresponding to $\overline{x_i}$ (resp. x_i) between u_k and u_{k+1} for all the k concerned.

Therefore, when there are two color disjoint st-paths in the graph we can deduce a truth assignment by setting to true any literal corresponding to a subpath used by the first path between s and t. From the preceding paragraph we are sure that no variable can be assigned to true (literal x_i true) and false (literal $\overline{x_i}$ true) at the same time. In addition, since the first path has to go through every section representing a clause, at least one literal per clause is set to true, which means that all the clauses are satisfied. If some variables are not assigned any value, then they are not necessary to satisfy any clause, and we can assign them any value without changing the answer to the 3SAT instance. Consequently, if there are two color disjoint st-paths connecting s and t, there is a truth assignment satisfying all the clauses of the 3SAT instance.

We now only need to prove that when there is a truth assignment satisfying all the clauses of the 3SAT instance, there are also two color disjoint st-paths connecting s and t. So we consider a truth assignment satisfying the instance of 3SAT. We define the first path to be composed of all the subpaths associated to literal set to true by this assignment. Since the assignment satisfies all clauses, the path connects s and t. In addition, by construction of the graph, the first path can use only one color from each pair (p^i_{jk}, p^i_{kj}), the other color of each pair is then available for the second path which thus exists and concludes the proof. □

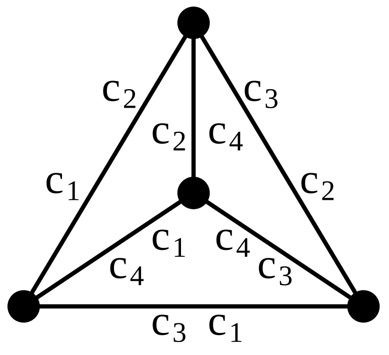
3 Transformation

3.1 What the problem really is

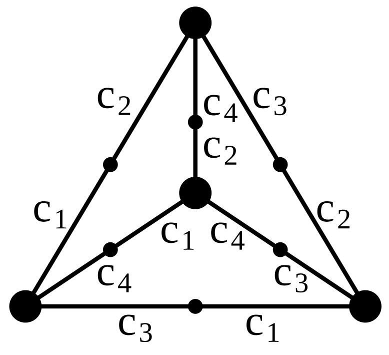
We have learnt from [3] and the previous sections that the span is a determining parameter in the complexity of MC-Path and MC-st-Cut. When considering the complexity of colored problems we work with colored graphs as a general class of elements completely disconnected from its origin, the multilayer networks. In this context the span is known for each graph and cannot be changed by interverting edges on paths. However, when we are given a multilayer network and a problem to solve, we first apply a AND transformation and then work on a colored graph.

This transformation has already highlighted the role of the span in the complexity and approximability properties of the colored problems. It is also momentous in the solving process since it can have serious repercussions on the span of colors of the colored graph obtained as shows Figure 3. Indeed, both Figures 3(b) and 3(c) represents a colored graph obtained with a AND transformation of the multicolored network of Figure 3(a), but the span of all colors is 3 in the graph of Figure 3(b) while it is only 1 in the graph of Figure 3(c). Consequently, if the graph of Figure 3(b) is used, the colored problem considered is hard to solve, while it is polynomial if the graph of Figure 3(c) is chosen.

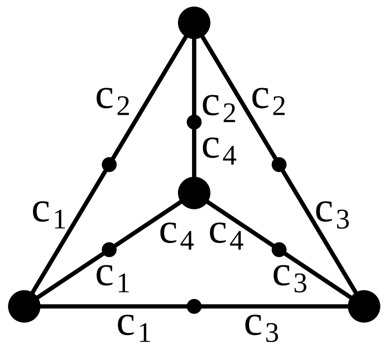
Therefore, the next step in the study of this AND transformation is to find out if it can be done in such a way to obtain only *interesting* colored graphs with regard to solving possibilities. In



(a) Example of multicolored network



(b) Possible transformation of 3(a), span 3



(c) Possible transformation of 3(a), span 1

Figure 3: The AND transformation of a multicolored network can result in several distinct colored graphs.

the light of the known polynomial cases, we have studied two questions detailed in the following sections.

3.2 Maximization problem

The problem of maximizing the number of colors of span 1 (Transformation) takes its interest from the results of section 2 : it comes down to minimizing the number of colors of span > 1. It may then be possible to use efficiently the solving method based on the MILP formulation for MC-Path and MC-st-Cut proposed in section 2.

Problem **1** (Transformation)

Input : *A multicolored network $\mathcal{R} = (V_\mathcal{R}, E_\mathcal{R}, \mathcal{C})$.*

Output : *A colored graph $G = (V_G, E_G, \mathcal{C})$ obtained from $\mathcal{R}$ through the* AND *transformation, that is by replacing each multicolored edge of $E_\mathcal{R}$ by a path of monochromatic edges in G.*

Measure : *Maximize the number of colors of span 1 in G.*

However, the result we detail in this section leaves no hope of solving the problem Transformation. This complexity and inapproximability result was just touched on in [11].

Theorem 3 *There is an approximability preserving reduction from Maximum Set Packing to* Transformation.

Proof : Let $\mathcal{R}$ be a network constructed from an instance of Maximum Set Packing. This multicolored network contains a vertex u_i for each element $u_i \in U$. Each of these vertices is connected to a same additional vertex v of $\mathcal{R}$ by a multicolored edge. $\mathcal{R}$ is thus a star (Figure 4).

Each subset $c_j \in \mathcal{C}$ of the Maximum Set Packing instance is identified to a color of the network $\mathcal{R}$. The set of colors of a multicolored edge vu_i represents the subcollection of subsets $c_j \in \mathcal{C}$ containing u_i in the Maximum Set Packing instance.

If a subset c_k is reduced to a single element u_k, a vertex $u_{k\text{bis}}$ is added to the network $\mathcal{R}$. This vertex is connected to v by an edge whose color set is reduced to c_k.

A subset $c_j \in \mathcal{C}$ is thus represented by its color on the multicolored edges connecting v to each of the vertices corresponding to one of its elements. And each subset $c_j \in \mathcal{C}$ is represented on at least two edges.

As a consequence, a color can be of span 1 after transformation only if it is adjacent to v on every path obtained from a vu_i edge on which this color is represented. Furthermore, two colors represented on a same edge cannot be of span 1 simultaneously because only one of them can be adjacent to v. Hence by construction of $\mathcal{R}$ two colors of span 1 after transformation correspond to two disjoint sets of the Maximum Set Packing instance since they cannot share any edge in $\mathcal{R}$.

Therefore, a set of colors of span 1 in the colored graph obtained after the transformation of network $\mathcal{R}$ corresponds to a subcollection of disjoint sets of $\mathcal{C}$ in the instance of Maximum Set Packing of same size.

In addition, a subcollection of disjoint sets in the instance of Maximum Set Packing induces a set of colors of same size that can be of span 1 simultaneously in the colored graph obtained from $\mathcal{R}$. Since the sets are disjoint, the colors that represent them in the network do not share any edge and can then be all adjacent to v at the same time. □

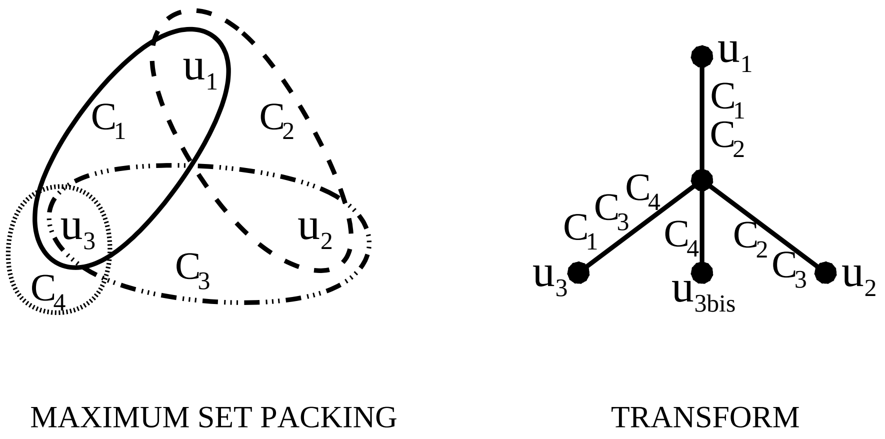


Figure 4: An instance of the Maximum Set Packing problem and a network constructed from it.

Corollary 1 *The* TRANSFORMATION *problem is NP-Hard and is not approximable within a factor* $|\mathcal{C}|^{\frac{1}{2}-\varepsilon}$ $\forall\varepsilon > 0$ *unless P = NP. Furthermore, this problem is not approximable within a factor* $|\mathcal{C}|^{1-\varepsilon}$ $\forall\varepsilon > 0$ *unless NP = ZPP.*

3.3 Decision problem

In this section we show that deciding whether a multicolored network can be transformed into a colored graph of span 1 is polynomial. In other words, we are sure to detect in polynomial time every networks on which almost all colored problems are polynomial.

Theorem 4 *There is a polynomial time algorithm that decide whether a multicolored network* $\mathcal{R} = (V_{\mathcal{R}}, E_{\mathcal{R}}, \mathcal{C})$ *can give a colored graph of span 1 after a* AND *transformation.*

The following subsections outline the polynomial time algorithm and prove Theorem 4. The principle of this algorithm is first to determine for each color *independently* whether it can be of span 1. During this first step, the final position of colors on some edges is determined : these colors can not be of span 1 otherwise. The second step consists in verifying that these necessary, or forced, positions are not in contradiction between two colors. Each step produces either a proof that the transformation can not result in a colored graph of span 1 or the forced positions of colors on edges, that is a colored graph of span 1.

3.3.1 Preliminary remark

If more than two colors sharing an edge are also present on at least one other edge, then they can not be of span 1 simultaneously since they all need to be put at an extremity of the path replacing the edge after transformation, but there are only two extremities. It is the case for the edge $\{u, v\}$ of the network of Figure 5. The three colors c_1, c_2 and c_4 need to be adjacent to either u or v to be of span 1, but only two of them can get these places.

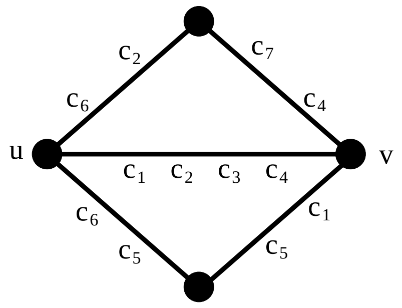


Figure 5: More than two colors but only two extremities for edge $\{u, v\}$.

In addition, a color such as c_3 in Figure 5 which is present on a single edge will be of span 1 after transformation whatever happens. Thus we need not consider these colors.

Consequently in the following we consider only networks in which each edge contains at most two colors.

3.3.2 Definitions and properties on edges

The edges of a multicolored network can be divided into two classes, the *fixed* edges and the *positionable* ones.

The fixed edges are edges containing a single color such as the edges $\{x, y\}$ and $\{z, t\}$ of the network 6 which contain only the color c_1. Consequently, these edges are not affected by the transformation. Their most important property is that whatever the transformation of the network is, the two extremities of a fixed edge for a color c will belong to the same connected component with regard to the color c.

The positionable edges are the remaining edges of the network containing two colors each : the position of the two colors with regard to the two extremities of a positionable edge has to be determined. Unlike the fixed edges case, if two vertices of the network are connected only by positionable edges for a given color c, these vertices can not belong to the same connected

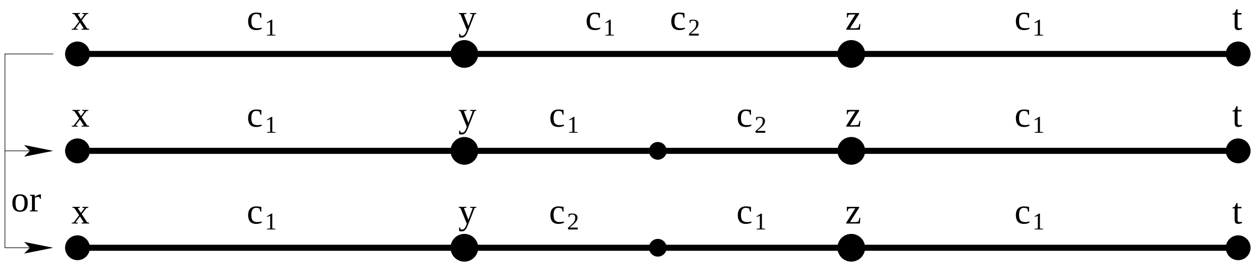


Figure 6: Fixed and positionable edges.

component for color c after the AND transformation. Indeed, after the transformation, the color c is adjacent either to one of these vertices or to the other, but not to both. On Figure 6 the path obtained from the edge $\{y, z\}$ illustrate this property of the positionable edges, the color c_1 can be adjacent to y or z but not to both simultaneously.

We can deduce from this property of the positionable edges that if a color contains several fixed edges, they have to form a connected component since they can not be connected by positionable edges.

3.3.3 Definitions and properties on colors

A color c is *free* on an edge $e = \{u, v\}$ if the position of color c (adjacency to u or v) on edge e has no incidence on its span. A color can be free on an edge in two cases : when it is present only on this edge like color c_3 of Figure 5 or color c_7 on the multiple edge $\{u, v\}$ of Figure 7, or when the two extremities of the edge already belong to the same connected component with regard to that color, like color c_2 of Figure 7.

A color is *half-free* on a multiple edge of the network when it is present only on the edges of a multiple edge. The multiple edge $\{u, v\}$ of figure 7 contains three *half-free* colors : c_3, c_5 and c_6. Each of these colors can be of span 1 if on all the edges it is adjacent to a same vertex, either u or v in this network. The multiple edge $\{u, v\}$ also contains color c_4, however it is not half-free since there is an other edge of color c_4 adjacent to u but whose other extremity is not v. Indeed, to be of span 1, c_4 has to be adjacent to u because of this other edge. Note then, that a color cannot be half-free if it is not present only on edges belonging to a single multiple edge. Consequently, to check that a color is half-free it is enough to check that the set of vertices to which it can be adjacent is of size two.

3.3.4 Existence of an algorithm

The existence of a polynomial time algorithm to decide whether a network can be transformed into a colored graph of span 1 is based on the following points :

Step 1 : deciding independently for each color

First, a color can not be of span 1 if it does not contain a *central vertex set*, that is either a single connected set of *fixed* edges to which each *positionable* edge is adjacent or, when there is no *fixed* edge, a single vertex to which each positionable edge containing the color is adjacent. The necessity of such kind of central vertex sets is based on the impossibility for a color on a positionable edge to

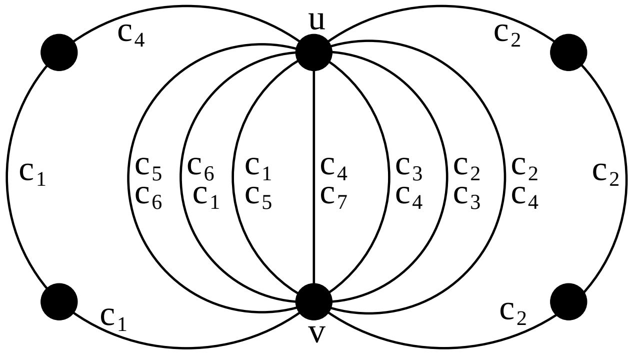


Figure 7: Network with a multiple edge and half-free colors.

be adjacent to both extremities of the edge after the AND transformation and thus the impossibility to make a connection between two distinct components. Determining the central vertex set or that there is no such set can be done by a simple and polynomial graph searching on the fixed edge set of the color. In case there is no central vertex set, the graph searching produces at least two distinct components both containing fixed edges for the color, which proves that this color can by no means be of span 1.

When a color c has a central vertex set, the position of color c on the positionable edges on which c is not free is forced. Indeed there is at most one extremity of such edges that can be adjacent to the central vertex set otherwise the color c would be free on it. Therefore, c can be of span 1 only if its position after the transformation is adjacent to that vertex set : it is forced. Once the central vertex set is known, an other graph searching on the positionable edges of the color either forces all the positions, or find an edge which is not adjacent to the central vertex set.

Consequently it is polynomial to determine whether a color can be of span 1 independently of the other colors, in addition the process either gives all forced positions or produces a certificate of the impossibility for the color to be of span 1.

Step 2 : confronting positions of all colors

Trivially, two colors sharing a positionable edge can both be of span 1 only if their respective forced positions are distinct. If they are not, one of the color can not be of span 1.

An other trivial remark concerns *half-free* colors on multiple edges. They can be of span 1 only if the positions left by the (non-free) colors with which they share edges are all adjacent to a same vertex. Since *free* colors can be positioned arbitrarily, the decisions about half-free colors need to take into account only the forced positions. The forced positions of other colors also produce a certificate of the impossibility for the half-free colors to be of span 1 if it is indeed the case.

Finally, the position of *free* colors can be assigned.

Conclusion

All these points permit to detect the possible structural impossibilities to be of span 1 of each color, but also conflicts between two colors by the position assignment they induce. When there are no conflicts, the knowledge of a central vertex set for a color gives its position on every edge where it appears. Then the position of all the *half-free* colors can also be determined and finally the position of *free* colors.

Finally, a detailed algorithm was proposed in [12] that runs in $\mathcal{O}(|\mathcal{C}||E_{\mathcal{R}}| + |\mathcal{C}|^2|E_{\mathcal{R}}| + |E_{\mathcal{R}}|)$.

3.3.5 Algorithm

In this section we detail the different steps of the decision algorithm and give upper bounds on their time complexity.

The first processing of the network consists simply for each color to check whether it is present on a single or several edges. Then for each edge, enough colors which are present only on that edge are removed in order to keep only two colors, except for fixed edges. This processing can be done in $\mathcal{O}(|\mathcal{C}||\mathcal{E}_{\mathcal{R}}|)$ and also permits to detect fixed edges.

A second processing in time $\mathcal{O}(|\mathcal{C}||\mathcal{E}_{\mathcal{R}}|)$ permits to detect half-free colors : the idea is simply to compute for each color the union of the extremities of all edges containing this color. If this set is reduced to two vertices, then the color is half-free.

The next processing produces the central vertex set for all colors in time $\mathcal{O}(|\mathcal{C}||\mathcal{E}_{\mathcal{R}}|)$. For a color without fixed edges it consists only in computing the intersection of the extremities of all edges containing this color. Indeed, the central vertex set in this case is a single vertex to which every positionable edge containing the color is adjacent. For a color with at least one fixed edge, a graph searching on the fixed edges of the color permits to check the connexity of the central vertex set and to know which vertices it contains.

Once the central vertex sets are known, another $\mathcal{O}(|\mathcal{C}||\mathcal{E}_{\mathcal{R}}|)$ searching forces the position of all positionable edges.

Finding the positions of half-free colors is a bit more complicated than the previous processings since it is first necessary to find a color whose position is forced on an edge it shares with an half-free color ($\mathcal{O}(|\mathcal{E}_{\mathcal{R}}|)$). The position of the half-free color is then forced too and the process can be repeated for every half-free colors (at most $|\mathcal{C}|$ times).

However, there might be some half-free colors sharing edges only with colors whose position is not forced yet, for example, other half-free colors or free ones. Therefore, the position of such half-free colors is not forced by other colors : the algorithm has to force arbitrarily the position of a randomly chosen color ($\mathcal{O}(|\mathcal{E}_{\mathcal{R}}|)$). The normal process for half-free colors need then to be repeated until there remain no half-free colors not forced. and then go on with the normal process which can thus be repeated at most $|\mathcal{C}|$ times since there are at most $|\mathcal{C}|$ half-free colors to position randomly. The processing for the half-free colors can thus be the more time consuming one with at most $\mathcal{O}(|\mathcal{C}^2||\mathcal{E}_{\mathcal{R}}|)$ steps.

When the position of positionable and half-free colors is known, forcing the position of free colors can be done in time $\mathcal{O}(|\mathcal{E}_{\mathcal{R}}|)$.

Finally the algorithm needs only to check that there are no conflict between color positions on each edges ($\mathcal{O}(|\mathcal{E}_{\mathcal{R}}|)$).

As a conclusion, identifying positionable and fixed edges, half-free and free colors as well as central vertex sets can be done in polynomial time by simple graph searching. It is then easy and still polynomial to force the positions of colors and then check the compatibility of the assigned positions. The decision problem is therefore polynomial.

3.4 Note on the design of heuristic algorithms for the maximisation problem

As was proved in Section 3.2 it is not possible to guaranty anything on the quality of a transformation in terms of number of colors of span 1. However in this section we identify some interesting points from the above algorithm description that should be considered in the design of heuristic algorithm for the Transformation problem.

First, an heuristic algorithm that maximize the number of colors of span 1 need not consider colors which do not contain a central vertex set since these colors can not be of span 1. Actually, all the choices that such an algorithm need to make concerns sets of colors that can be of span 1 independently but not simultaneously. It is the case when three colors sharing an edge are also present on other edges, or when two colors need to be put at the same extremity of an edge they share, in order to be of span 1.

The starting point for an heuristic algorithm could be to discard all free colors on the involved edges and all colors that can not be of span 1 independently. Then the main part of the algorithm is to make choices between colors that can not be of span 1 simultaneously. Since several strategies are possible to make a decision, several algorithms can be designed from this general frame.

Finally, note that the number of colors that can independently be of span 1 is an upper bound on the maximum number that can be of span 1 simultaneously.

3.5 Some hope dwells in a subclass of colored graphs

The consequence of Theorem 1 about the Transformation problem is that from a given network we have no guaranty to find a good colored graph on which the solving method based on the MILP formulation of Section 2.1 for MC-Path is efficient, even when such a colored graph exists.

The practical resolution of the MC-Path problem is thus compromised since in the general case MC-Path has been proved hard to approximate within a large factor [3].

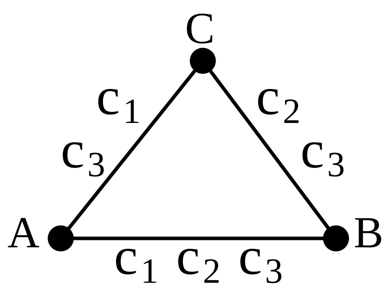
However, the study of the transformation from the network to the colored graph is not over yet and may still bring usefull results from practical and theoretical points of view. Indeed, the NP-hardness results and inapproximability factors known so far are proved for the general class of colored graphs, but actually only a subclass of colored graphs can represent multilayer networks. Figure 8 gives an example of multicolored network which do not represent *any* multilayer network.

To prove that, let's try to find an underlying network on which the multicolored network of Figure 8(a) could be routed. In this network there must be a path of length three joining vertices A and B since there are three colors on the edge $\{A, B\}$ of network 8(a).

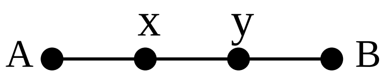
Therefore the network we search must contain the graph of Figure 8(b). In this graph, all the vertices have to be distinct, otherwise the routing of the multicolored edge $\{A, B\}$ of network 8(a) would not be valid due to the loop it would contain.

There are now six possible color assignments on the three edges as depicted on Figure 8(c). However, among these assignments, only two are valid : color c_3 has to be adjacent to both c_1 and c_2, otherwise either $\{A, C\}$ or $\{B, C\}$ of the multicolored network have no possible route in the graph we are constructing.

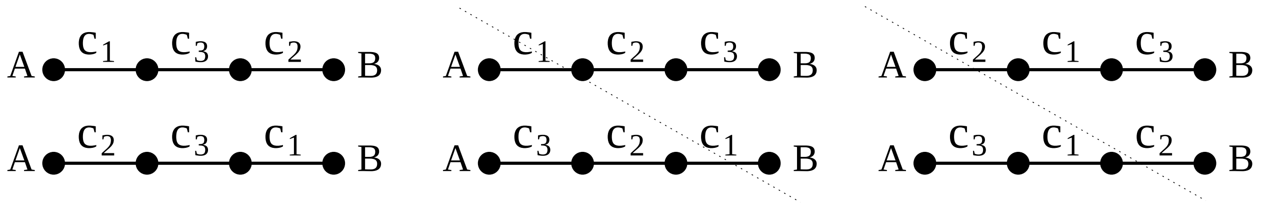
Finally, we have to place vertex C in at least one of the two remaining graphs. To avoid loops C has to be distinct from A and B. In the (c_1, c_3, c_2) graph, if $C = X$ then the multicolored edge $\{A, C\}$ can not be routed without loops, and if $C = Y$ then it is the edge $\{B, C\}$ which has no route. Consequently we have to discard the (c_1, c_3, c_2) graph, and for similar reasons the last graph too.



(a) A multicolored network



(b) Topology of an eventual underlying graph for network 8(a)



(c) Assignments of colors on graph 8(b)

Figure 8:

To conclude the example, there is no underlying network on which the multicolored network of Figure 8(a) can be routed : the layout of the colors does not permit it.

Several questions arise from this observation. The most important one concerns the specific properties of colored graphs representing a network. Knowing these properties, we could then try to establish the complexity of the colored problems on real, or at least feasible, multilayer networks, and perhaps find new polynomial cases. From a theoretical point of view the problem of deciding whether a colored graph represents a feasible network would then be interesting to investigate.

References

[1] A. L. Chiu, J. Strand, and R. Tkach. Issues for routing in the optical layer. *IEEE Communications Magazine*, 39(2):81–87, 2001.

[2] H. Choi, S. Subramaniam, and H. Choi. On double-link failure recovery in WDM optical networks. In *IEEE Infocom*, pages 808–816, New-York, USA, June 2002.

[3] D. Coudert, P. Datta, S. Perennes, H. Rivano, and M-E. Voge. Shared risk resource group: Complexity and approximability issues. *Parallel Processing Letters*, 17(2):169–184, June 2007.

[4] P. Datta and A.K. Somani. Diverse routing for shared risk resource groups (SRRG) failures in WDM optical networks. In *Proceedings of IEEE BroadNets*, pages 120– 129, October 2004.

[5] B. Jaumard. Exemples de réseaux.

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

[7] M. O'Mahony, D. Simeonidu, A. Yu, and J. Zhou. The design of the european optical network. *IEEE/OSA Journal of Lightwave Technology*, 13(5):817–828, 1995.

[8] D. Papadimitriou, F. Poppe, J. Jones, S. Venkatachalam, S. Dharanikota, R. Jain, R. Hartani, and D. Griffith. Inference of shared risk link groups. IETF Draft, OIF Contribution, OIF 2001-066.

[9] P. Sebos, J. Yates, G. Hjlmtsson, and A. Greenberg. Auto-discovery of shared risk link groups. In *Proceedings of IEEE/OSA OFC*, volume 3, pages WDD3–1 – WDD3–3, 2001.

[10] A. Todimala and B. Ramamurthy. Survivable virtual topology routing under shared risk link groups in WDM networks. In *First Annual International Conference on Broadband Networking (BroadNets '04)*, pages 130–139, San Jose, CA, Oct. 2004.

[11] M.-E. Voge. How to transform a multilayer network into a colored graph. In *IEEE-LEOS ICTON / COST 293 annual conference on GRAphs and ALgorithms in communication networks*, Nottingham, June 2006.

[12] M.-E. Voge. *Optimisation des réseaux de télécommunication : Réseaux multiniveaux, Tolérance aux pannes et Surveillance du trafic*. PhD thesis, Ecole doctorale STIC, Université de Nice-Sophia Antipolis, novembre 2006.

[13] Y. Ye, S. Dixit, and M. Ali. On joint protection/restoration in IP-centric DWDM-based optical transport networks. *IEEE Communications Magazine*, pages 174–183, June 2000.