



**HAL**  
open science

## **Tree Reconfiguration without Lightpath Interruption in Wavelength Division Multiplexing Optical Networks with Limited Resources**

Bernard Cousin, Joel Christian Adépo, Michel Babri, Souleymane Oumtanaga

► **To cite this version:**

Bernard Cousin, Joel Christian Adépo, Michel Babri, Souleymane Oumtanaga. Tree Reconfiguration without Lightpath Interruption in Wavelength Division Multiplexing Optical Networks with Limited Resources. International Journal of Computer Science Issues, 2014, 11 (2), pp.7 - 17. hal-01144766

**HAL Id: hal-01144766**

**<https://hal.science/hal-01144766v1>**

Submitted on 12 Aug 2015

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Tree Reconfiguration without Lightpath Interruption in WDM Optical Networks with Limited Resources

Bernard Cousin, Joel Adépo, Michel Babri, Souleymane Oumtanaga

## Abstract

**Today, operators use reconfiguration to improve the performance of connection oriented networks. In our previous work, we studied tree reconfiguration without data flow interruption in WDM optical networks. In our previous approach, the available resources of the network (wavelengths) are not considered as limited. In this work, we study tree reconfiguration in a network which has a limited number of wavelengths per link. This paper proposes an algorithm called *TRwRC* (tree reconfiguration with resources constraint). *TRwRC* reconfigures a tree without data flow interruption in a network with a limited number of wavelengths.**

*Keywords:* Path reconfiguration, optical WDM network, multicast tree, reconfiguration sequence, lightpath interruption.

## I. Introduction

Path reconfiguration is used by network operators to improve network performance. It consists of moving from an initial path to a new one. Since networks are disturbed by failures, overloads, and deployments of new network resources or maintenance operations, network reconfiguration becomes very important because it enables the fulfillment of QoS requirements with the available network resources. In our previous work (Cousin et al.), we presented an efficient algorithm which achieves tree reconfiguration in WDM optical networks in which a multicast connection is established. It was the first work on that problem in the literature. The objectives of this work are: first, tree reconfiguration without data flow interruption; second, reduction of the latency between the triggering event and the effective reconfiguration of the nodes on the paths by the tree reconfiguration process; and third, reduction in the resources used

during the reconfiguration by the tree reconfiguration process. In our previous work, we proposed a branch-by-branch process called *BpBAR\_2*. In that approach, the number of available wavelengths per link was not considered as a constraint. At each step of the reconfiguration process, the proposed algorithm computed a light-forest to span all the destinations. Since the network resources are supposed to be unlimited, the number of trees per light-forest is also unlimited. In this current work, the number of available wavelengths per link is a constraint. So, the number of trees per light-forest produced at each step should be limited. This constraint first allows the tree with the available network resources to be reconfigured, and second, contributes to optimizing the network resources used. The problem addressed in this paper is tree reconfiguration without data flow interruption in a network with a limited number of available wavelengths per network link. The proposed algorithm must maintain the continuity of the data flow toward all the destination nodes and reduce the cost of resources used and the duration of the reconfiguration process.

In Section II, we describe the tree reconfiguration in WDM optical networks. Section III presents the related works. In Section IV, we prove the continuity of data flows during the reconfiguration with the *BpBAR\_2* process. Section V presents the context of tree reconfiguration with limited resources and the proposed algorithms called *TRwRC*. Section VI presents the evaluation criteria and the results of the evaluation. This algorithm allows tree reconfiguration in a network which has limited resources and it reduces the resource cost during the reconfiguration. Finally, we conclude this article in the final section.

## II. Multicast Connection in WDM Optical Networks

The path of a connection in WDM optical networks (Mukherjee, 2000) consists of optical channels. An optical channel in a WDM optical network is an optical signal transmitted over one wavelength in one fiber. The ends of an optical channel consist of an output port of a switch called the source port and an input port of another switch called the sink port. If the source port belongs to a node, then this node is the source node of the optical channel. If the sink port belongs to a node, then this node is the sink node of the optical channel.

A lightpath is an all-optical path on the same wavelength between two nodes: a lightpath is made of a set of successive optical channels. The set of paths used by a multicast connection (Ding and Poo, 2003; Malli, Zhang and Qiao, 1998) in a WDM optical network forms a tree called a light-tree. A light-tree is an all-optical structure on the same wavelength between a source node and a set of destinations nodes. The light-tree is obtained with the use of particular nodes which are capable of duplicating an incoming light signal from one node input port into one or several node output ports. In this work, we assume that all network nodes have this light splitting capability. In Zhou (2009), the reader may find some interesting considerations when this assumption is not fulfilled, and may find some efficient computation heuristics or protocols in Zhou (2011) and Jawhar (2011). A structure called a light-forest is used when a single light-tree is not sufficient to cover all destinations. A light-forest is a set of light-trees originated at the same source node but established with different wavelengths. For example,  $T_0^{l_0}$  and  $T_0^{l_1}$  are the two light-trees of the light-forest in Fig. 1 between the source node  $s$  and the destinations  $\{d_1, d_2, d_3\}$ .  $T_0^{l_0}$  is established with a wavelength  $l_0$  between the source node  $s$  and the destination  $\{d_1, d_2, d_3\}$ .  $T_0^{l_1}$  is established with a wavelength  $l_1$  between the source node  $s$  and the destination  $\{d_1, d_2\}$ . A branch  $Br(T, s, d)$  of the light-forest is a lightpath between the source node  $s$  and the destination  $d$  on the tree  $T$ .

In a WDM optical network, before transmitting data from a source  $s$  to a destination  $d$ , a connection has to be set up first along the path from  $s$  to  $d$ . This connection is established by allocating a wavelength on each link along the connection's path and each node is instructed to switch the wavelength transparently through its switch fabric. We called this path with the allocated wavelength a pre-established path. A pre-established path between node  $y$  and the destination  $d$  is denoted  $p(y, d)$ .  $p(s, d_3)$  in yellow in Fig. 1 is a pre-established path between  $s$  and  $d_3$ .

If there is no confusion, a tree  $T_i^{l_0}$  will be noted  $T_i$  and a pre-established  $p(y, d)$  will be noted  $p$ .

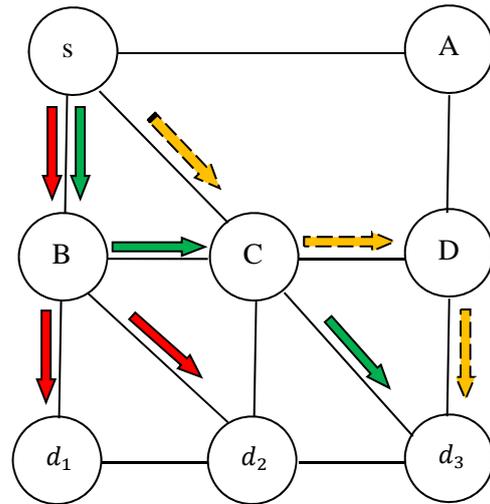


Fig. 1 Light-forest between the source node  $s$  and the destinations  $\{d_1, d_2, d_3\}$  and the pre-established path (in yellow) between  $s$  and  $d_3$ .

The key characteristic of the multicast in optical networks is that an optical channel can be shared to feed multiple output ports and reach many destinations. In Fig. 1, the optical channel between  $s$  and  $B$  is shared to reach the destinations  $d_1$  and  $d_2$ .  $B$  becomes a branching node. An optical channel which has  $B$  as a sink node is an upstream optical channel of  $B$ . An optical channel which has  $B$  as a source node is a downstream optical channel of  $B$ .

The number of upstream optical channels of a node  $x$  on a tree  $T_i$  is noted  $|deg_T^+(x)|$ . The number of downstream optical channels of  $x$  is noted  $|deg_T^-(x)|$ . A branching node  $x$  of a light-tree  $T$  is such that its number of downstream optical channels is superior or equal to 2 ( $|deg_T^-(x)| \geq 2$ ). Any node connected to a multicast tree has a single upstream optical channel (except the source, which has none) and any node may have multiple downstream optical channels (except the destinations, which have none).

Two optical channels of a light-forest are adjacent if one optical channel is an upstream channel of a node on a tree  $T$  and the other is a downstream channel of the same node on the same tree.

### III. Tree Reconfiguration Operations

Four reconfiguration operations are applicable on an optical switch. We denote  $C_j$  as the configuration (set of lightpaths) obtained at the  $j^{th}$  step of the tree reconfiguration process.

- 1) Addition of a light switching: *Add*

$Add(C_j) \Rightarrow \exists x \in C_j / (+ x_i^l, x, l, x_j^o)$ .

If  $x_i^l$  is the sink port of an optical channel  $u$ ,  $Add(C_j)$  adds one optical channel  $v$  with wavelength  $l$  in the configuration  $C_i$  such that:

- $x_j^o$  is the source port of  $v$
- $v$  is adjacent to  $u$
- The signal coming from  $x_i^l$  feeds  $v$

2) Deletion of a light switching:  $Del$

$Del(C_j) \Rightarrow \exists x \in C_j / (- x_i^l, x, l, x_j^o)$ .

If  $x_j^o$  is the source port of an optical channel  $u$ ,  $Del(C_j)$  deletes the link  $u$  from the configuration  $C_j$ .

3) Combined operation:  $Comb$

$Comb(C_j) \Rightarrow \exists x \in C_j / (x_i^l, x, l, x_j^o \Rightarrow x_k^o)$ .

If  $x_i^l$  is the sink port of an optical channel  $u$ , and  $x_j^o$  the source port of an optical channel  $v_1$ ,  $Comb(C_j)$  simultaneously deletes the link  $v_1$  from the configuration  $C_i$  and adds an optical channel  $v_2$  with wavelength  $l$  such that  $x_k^o$  is the source port of  $v_2$ , and is adjacent to  $u$ .

4) Wavelength conversion:  $Conv$

$O_4(C_j) \Rightarrow \exists x \in C_j / (x_i^l, x, l \Rightarrow m, x_j^o)$ .

If  $x_j^o$  is the source port of an optical channel  $u$ ,  $Conv(C_j)$  converts the wavelength  $l$  of  $u$  to wavelength  $m$ .

#### IV. Related Works

In the literature, reconfiguration policy and path computation have been studied by Baldine and Rouskas (1999), Kárász and Pándi (2005) and Kárász (2006). They do not consider the configuration process in itself. The well-known reconfiguration processes *MBB* (make-before-break) and *BBM* (break-before-make) are proposed for path reconfiguration. *MBB* achieves path reconfiguration without path interruption but not regardless of the paths and the cost of many reconfiguration steps. Cousin and Molnar (2006) studied fast path reconfiguration without data flow interruption as they are interested in unicast connections.

The connections considered in all these previous works are unicast connections. In this paper, since the problem considered is tree reconfiguration without

data flow interruption, we first describe an *MBB* adaptation to the tree reconfiguration problem. This *MBB-based* algorithm is called *MBB\_2*. Then we present the *BpBAR\_2* process proposed in our previous work. It was the first time that a tree reconfiguration algorithm which achieves uninterrupted flow was proposed in the literature. We compare our new algorithm to these two algorithms.

##### 1. *MBB\_2*

*MBB* is a well-known path reconfiguration process. The *MBB* reconfiguration process causes an interruption when the initial and final paths make a cyclic dependency (Cousin and Miklos, 2007). In the adaptation of *MBB* to tree reconfiguration which uses no additional resource, the no interruption requirement is not fulfilled. In tree reconfiguration without additional resources, since the same link may be shared by several branches of the tree, the probability of having cyclic dependency between the branches of the trees increases. Thus, *MBB\_2* is an *MBB-based* algorithm which uses one additional wavelength during the reconfiguration process. *MBB\_2* consists of two phases. In each phase, it uses the function *TRwMBB2* described in Algorithm 1. In the first phase of *MBB\_2*, *TRwMBB2* configures each sub-tree of the new tree with the additional wavelength. In the second phase, it configures each sub-tree of the new tree with the final wavelength. The sub-trees of the initial tree (respectively the final tree) considered in this algorithm are rooted at the son nodes of the tree source  $s$  in the initial tree (respectively in the final tree). The set of destinations of each sub-tree is denoted *Dest*.

---

##### Algorithm 1: Function used by *MBB\_2*

---

*TRwMBB2* ( $T_0, T_z$ )

1. {
  2. Configure in one step all the nodes on the new tree  $T_z$  to build all the pre-established branches with the additional wavelength
  3. For each sub-tree of the new tree  $T_z$
  4. {
  5. Select a sub-tree of the old tree not configured
  6. Interrupt in one step the data flow on branches (on the initial tree  $T_0$ ) toward all the destinations  $d_j$  not configured and such that  $d_j \notin Dest$
  7. Switch wavelength from the sub-tree of the old tree to the sub-tree of the new tree at the tree source  $s$  to feed the new tree
  8. Configure in parallel all the ports of the nodes on the old sub-tree to remove its branches
  9. }
  10. }
-

**Algorithm 2: MBB<sub>2</sub>** $MBB2(T_0, T_z)$ 

1. {
2. TRwMBB2( $T_0, T$ )
3. TRwMBB2( $T, T_z$ )
4. }

2. *BpBAR<sub>2</sub>*

*BpBAR<sub>2</sub>* is a *BpBAR*-based algorithm which consists of two phases. Each phase consists of several stages. Each stage is a series of reconfiguration steps to reconfigure one branch. Figure 3 represents a diagram of operations scheduling in a stage of the *BpBAR* process to ensure the continuity of the lightpath during the reconfiguration. This diagram can be subdivided into five parts according to the successive operations executed. *BpBAR<sub>2</sub>*, which is a *BpBAR*-based process, uses parallel execution of operations to reduce the reconfiguration delay. *BpBAR<sub>2</sub>* executes, in parallel, all the operations belonging to the same part of a *BpBAR* stage as shown Table II. Figure 4 represents the diagram of a reconfiguration stage with *BpBAR<sub>2</sub>*. In the first phase of a reconfiguration with *BpBAR<sub>2</sub>*, each branch is established per stage; a new tree, a copy of the final tree which uses wavelengths different from the initial, is obtained. In the second phase, each branch is established per stage; a tree, a copy of the final tree which uses the initial wavelength, is obtained. We describe below a reconfiguration stage with the *BpBAR* process.

Table I describes the different parts of a reconfiguration stage with the *BpBAR* process. The operations are executed sequentially in this diagram. The length of each part (number of steps) is equal to the number of ports to be configured in this part.

After the setup of the pre-established path in part 1 of a stage, the combined operation is executed in part 2 to feed the pre-established path. *Comb* is executed at the end node of the pre-established path. Since it feeds the pre-established path and interrupts data flow on the other path, the appropriate switching node is the one which maintains the data flow continuity toward all the destinations after its configuration. This appropriate switching node depends on the initial tree  $T$  and the destination of the pre-established path  $d$ . It is denoted  $Sw\_node(T, d)$ . For example, in Fig. 2, C is the node  $Sw\_node(T_0, d_1)$ . Then, in part 3, the wavelength switching is deleted on all nodes of the exclusive part of the old branch on which flow is interrupted. Parts 4 and 5 occur when the switching node is noted as the source node  $s$ .

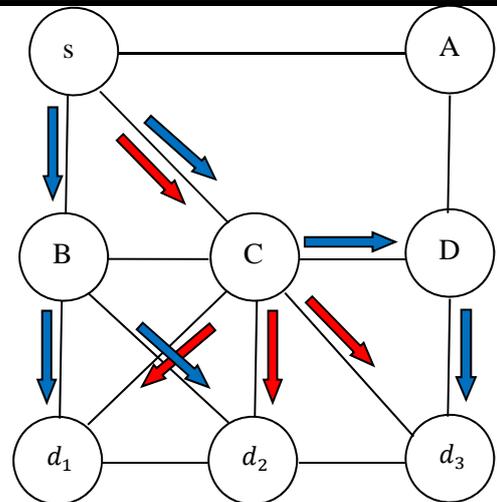
**Definition 1:** The end part of path  $p$  beginning at path node  $x$  is denoted  $term(p, x)$ .

**Definition 2:**  $Sw\_node(T, d)$  is the root node of the sub-tree of the tree  $T$  which will be used during the branch configuration between the source node and the destination  $d$ . A node  $x$  is the node  $Sw\_node(T, d)$  of the branch  $Br(T, s, d)$  if:

- (1)  $x$  is a branching node of tree  $T$  which uses the initial wavelength
- (2) For each node  $y \in term(Br(T, s, d), x)$  and  $y \notin \{x, d\}$ ,  $|deg_T^-(x)| = 1$ .

Table I: Distinct parts of a stage in *BpBAR* process

Parts	Description
1	From $C_i$ to $C_{i+k_1}$ : This part consists of setting up the pre-established path between the destination $d$ and the node $Sw\_node(T_j, d)$ . The operation used is <i>Add</i> : $C_{j+1} = Add(C_j)$ . The length is equal to the number of ports to be configured.
2	From $C_{i+k_1}$ to $C_{i+k_1+1}$ : This part feeds the pre-established path by executing the combined operation at $Sw\_node(T_j, d)$ : $C_{j+1} = Comb(C_j)$ . It consists of one step.
3	From $C_{i+k_1+1}$ to $C_{i+k_2}$ : This part deletes the pre-established path between the destination $d$ and the node $Sw\_node(T_j, d)$ . The operation used is <i>Del</i> : $C_{j+1} = Del(C_j)$ . The length is equal to the number of ports to be configured.
4	From $C_{i+k_2}$ to $C_{i+k_2+1}$ : This part consists of executing the combined operation <i>Comb</i> at $Sw\_node(T_j, d)$ to interrupt the flow between $Sw\_node(T_j, d)$ and the source node. It consists of one step.
5	From $C_{i+k_2+1}$ to $C_{i+k_3}$ : $C_{j+1} = Del(C_j)$ . This part deletes the pre-established path between the node $Sw\_node(T_j, d)$ and the source node $s$ . The length is equal to the number of ports to be configured.

Fig. 2 Example of tree reconfiguration problem. The old tree  $T_0$  is in red and the new tree  $T_z$  is in blue.



## VI. Proof of Data Flow Continuity Toward All Destinations During Tree Reconfiguration with *BpBAR\_2*

*BpBAR\_2* is a *BpBAR*-based algorithm which uses parallel execution of operations. First, we prove that a reconfiguration stage with *BpBAR* maintains lightpath continuity. Second, we prove that the parallel execution of all the operations belonging to the same part of a stage with *BpBAR* maintaining continuity of the lightpath.

Let us assume that a stage begins with a configuration  $C_i$  and is noted  $\langle O(C_i), O(C_{i+1}), \dots, O(C_{i+k_3}) \rangle$  where  $C_i$  is the  $i^{\text{th}}$  configuration which contains a light-forest spanning all the destinations of a multicast connection  $\langle s, \{d\} \rangle$  and  $O$  is a tree reconfiguration operation (described in Section III).  $C_i$  is represented by  $C_i = LF_i$  where  $LF_i = \sum_k T_k$  is a light-forest spanning all the destinations. We will prove that each part of a reconfiguration stage with a *BpBAR* process maintains the continuity of the lightpath.

### (1) Part 1

Let us consider a configuration  $C_j$  of part 1 which contains a light-forest spanning all the destinations of a multicast connection,  $C_j = LF_j + p_j$  where  $p_j$  is a pre-established path. Let  $V_j$  denote the set of optical channels of  $C_j$  such that  $LF_j$  spans all the destinations of the multicast group.  $Add(C_j)$  implies add new optical channel  $u$  in  $C_j$  such that  $u$  is adjacent to the end link of  $p_j$ . Thus, the new set of optical channels after  $Add(C_j)$  is  $V_{j+1} = V_j \cup \{u\}$ . The optical channels of  $LF_j$  are not modified so  $LF_j$  spans all the destinations. Thus,  $C_{j+1} = Add(C_j)$  contains a light-forest spanning all the destinations.

In *BpBAR*, the *Add* operation  $((+ x_i^l, x, l, x_j^o))$ , is carried out such that the ports  $x_i^l$  and  $x_j^o$  of the switch  $x$  do not belong to the set of ports of  $LF_j$ . Thus, the new pre-established path  $p_{j+1}$  is not fed and there is no loop.

### (2) Part 2

*Comb* is executed on the switching node  $Sw\_node(T_j, d)$ . The conditions to execute *Comb* in a configuration  $C_j$  with *BpBAR*, in part 2 are:

- $C_j = LF_j + p_j$  where the end nodes of  $p_j$  are  $Sw\_node(T_j, d)$  and  $d$ .
- $Comb(C_j) \Rightarrow (Sw_i^l, Sw, l, Sw_j^o \Rightarrow Sw_k^o)$ , where  $Sw$  is the node  $Sw\_node(T_j, d)$ .  $Sw_i^l$

and  $Sw_j^o$  belong to  $LF_j$ ,  $Sw_k^o$  does not belong to  $LF_j$  but belongs to  $p_j$ .

$Comb(C_j)$  interrupts the wavelength coming from  $Sw_i^l$  to  $Sw_j^o$  and switches the wavelength coming from  $Sw_i^l$  to  $Sw_k^o$ . Since  $Sw_k^o$  belongs to  $p_j$ , the pre-established path is now fed and ensures the continuity of the lightpath toward the destination  $d$ . Since  $Sw_j^o$  belongs to  $term(Br(T_j, s, d), Sw)$ ,  $term(Br(T_j, s, d), Sw)$  is interrupted. We know by definition that each node of  $term(Br(T_j, s, d), Sw)$  has only one downstream optical channel on  $T_j$  and feeds only destination  $d_j$  before its interruption. Since the pre-established path ensures the continuity of the lightpath toward  $d$  after  $Comb(C_j)$ , a light-forest spanning all the destinations of a multicast connection exists. If we assume that  $term(Br(T_j, s, d), Sw)$  is the new pre-established path,  $C_{j+1} = Comb(C_j) = LF_{j+1} + p_{i+1}$ .

### (3) Part 3

The condition to execute *Del* in a configuration  $C_j$  with *BpBAR*, in part 3 is:

- $C_j = LF_j + p_j$
- $Del(C_j) \Rightarrow (- x_i^l, x, l, x_j^o)$ , where  $x$  is the end node of  $p_j$ ,  $x_j^o$  does not belong to  $LF_j$  but belongs to  $p_j$

We know by definition that the pre-established path  $p_j$  is not used to transmit data ( $x_j^o$  does not belong to  $LF_j$  but belongs to  $p_j$ ).  $Del(C_j)$  configures the port  $x_j^o$  on  $p_j$  to delete the wavelength switching. After  $Del(C_j)$ , the ports of  $LF_j$  are not modifying and a light-forest spanning all the destinations of the multicast connection exists. If  $p_{j+1}$  is the new pre-established path after the *Del* operation,  $C_{j+1} = Del(C_j) = LF_{j+1} + p_{i+1}$ .

### (4) Part 4

*Comb* is executed on the source node of the multicast connection  $s$  in this part. The conditions to execute *Comb* in a configuration  $C_j$  with *BpBAR*, in this part are:

- $C_j = LF_j$
- The lightpath between the source node  $s$  and a destination  $d$  contains  $s$  twice: for example  $(s, x, \dots, Sw, \dots, s, y, \dots, d)$ .

- $Comb(C_j) \Rightarrow (-, s, l, s_k^O \Rightarrow s_k^O)$  where  $s_k^O$  belongs to  $LF_j$ .

This operation is equivalent to  $(-, -, s, l, s_k^O)$  followed by  $(+, -, s, l, s_k^O)$ . The signal coming from any input port of  $s$  is interrupted. The lightpath between  $s$  and the destination  $d$  becomes  $(s, y, \dots, d)$ . The part of path  $(Sw, \dots, s)$  which is not used to transmit data after the operation becomes the pre-established path  $p_{i+1}$ . Then  $C_{j+1} = Comb(C_j) = LF_{j+1} + p_{i+1}$  contains a light-forest  $LF_{j+1}$  spanning all the destinations of the multicast connection.

### (5) Part 5

The condition to execute  $Del$  in a configuration  $C_j$  with  $BpBAR$  in this part is:

- $C_j = LF_j + p_j$
- $Del(C_j) \Rightarrow (- x_j^I, x, l, x_j^O)$ , where  $x$  is the end node of  $p_j$ ,  $x_j^O$  does not belong to  $LF_j$  but belongs to  $p_j$ .

$p_j$  is not used to transmit data ( $x_j^O$  does not belong to  $LF_j$  but belongs to  $p_j$ ).  $Del(C_j)$  configures the port  $x_j^O$  on  $p_j$  to delete the wavelength switching. After  $Del(C_j)$ , the ports of  $LF_j$  are not modifying and a light-forest spanning all the destinations of the multicast connection exists. If  $p_{j+1}$  is the new pre-established path after the  $Del$  operation,  $C_{j+1} = Del(C_j) = LF_{j+1} + p_{i+1}$ .

We have proved that all the parts of a reconfiguration stage maintain the lightpath continuity. We conclude that a reconfiguration stage with  $BpBAR$  maintains data flow continuity during a branch reconfiguration.

## VII. Reconfiguration with Wavelengths Limit Constraint

### 1. Context and Problem Formulation

We consider a WDM optical network which has a limited number of available wavelengths per link. A multicast connection established in this network on the tree  $T_0$  uses wavelength  $l_0$ . We want to move from the tree  $T_0$  to a new tree  $T_z$ . The initial tree  $T_0$  and the final tree  $T_z$  must use the same wavelength  $l_0$ . We assume that the new tree is known. The reconfiguration must be carried out without data flow interruption. The tree configuration process proposed in our previous work (Cousin et al., 2012) produces a series of light-forests since it must maintain lightpath continuity. The number of trees per light-forest is not

increased with the proposed algorithm. This number is proportional to the number of branches to be established. In a network with a multicast connection established, several branches can share the same link which has very limited available resources. In this context,  $BpBAR_2$  may be inapplicable. Some branches which share the same link cannot be established. The problem in this work is to reduce the number of trees per light-forest at each step of the reconfiguration, regardless of the initial and final trees, and the network topology.

### 2. Our Proposition: TRwRC

The proposed tree reconfiguration is the TRwRC (tree reconfiguration with resources constraint) algorithm. It has two phases. TRwRC produces a series of light-forests during the tree configuration process. Let us assume that the destinations of the multicast connection is subdivided into two sets during the reconfiguration process: a set of destinations whose lightpaths are reconfigured (uses paths on the new tree) denoted  $Dest\_Config$  and a set of destinations whose lightpaths are not reconfigured (uses paths on the initial tree) denoted  $Dest\_NConfig$ . The maximal number of trees per light-forest in a configuration produced with  $TRwRC$  is two. Each tree, at each step, spans the destinations of one set described above.

$TRwRC$  uses the function  $ReconfBr$  described in Algorithm 3 to reconfigure each branch of the initial tree  $T_0$ . The reconfiguration of each branch with  $ReconfBr$  consists of two stages. In the first stage, the branch is established with an additional wavelength  $l_2$ . In the second stage, the branch is established with the additional wavelength  $l_1$ . The tree obtained after the reconfiguration of all the branches of the initial tree  $T_0$  is a copy of the new tree  $T_z$  using the additional wavelength  $l_1$  denoted  $T'$ . In the second phase,  $TRwRC$  configures simultaneously the ports of the nodes of the tree  $T'$  obtained after the first phase to convert the additional wavelength  $l_1$  into the initial wavelength. Algorithm 4 represents  $TRwRC$ . We described below the function  $ReconfBr$  used to reconfigure a branch.

Let  $C_0$  be the initial configuration and  $Br(T_z, s, d)$  a branch we want to establish.  $C_0$  contains a light-tree spanning all the destinations. We note  $C_0 = T_0$ .  $TRwRC$  uses the  $Add$  operation to setup in one step a pre-established path between  $Sw\_node(T_0, d)$  and  $d$  with an additional wavelength (for example  $l_2$ ). After this operation,  $C_1 = Add(C_0) = T_1 + p_1$  where  $T_1 = T_0$  and  $p_1$  is the pre-established path. The combined operation is executed on  $Sw\_node(T_0, d)$  to simultaneously feed the pre-established path and

interrupt the flow on the old path. The pre-established path now ensures the continuity of the lightpath toward the destination  $d$ . Before the execute *Comb* operation,  $Term(Br(T_0, s, d), Sw\_node(T_0, d))$  is shared only by the lightpath toward  $d$ . After the switching,  $C_2 = Comb(C_1) = T_2 + T'_2 + p_2$  where  $T_2$  spans the destinations of *Dest\_NConfig*,  $T'_2$  spans the destinations of *Dest\_Config* ( $\{d\}$ ) and,  $p_2$  is  $Term(Br(T_0, s, d), Sw\_node(T_0, d))$ .  $p_2$  is deleted in one step and  $C_3 = Del(C_2) = T_3 + T'_3$  where  $T_3 = T_2$  and  $T'_3 = T'_2$ . If the node  $Sw\_node(T_0, d)$  is not the source node  $s$ ,  $s$  is reconfigured to interrupt the data flow between  $Sw\_node(T_0, d)$  and  $s$ , then this part of the branch is deleted. After this stage, the new branch is established with the additional wavelength  $l_2$ . The same process is repeated in a new stage to establish the branch with the additional wavelength  $l_1$ . The two stages for each branch are repeated. In the first phase, we obtain a tree  $T'$ , a copy of the new tree  $T_z$  using the additional wavelength  $l_1$ .

In the second phase, the ports of the nodes of the tree  $T'$ , obtained after the first phase, configure simultaneously to convert the additional wavelength  $l_1$  into the initial wavelength.

---

**Algorithm 3: Function used to established a new branch**

---

*ReconfBr* ( $T_0, T_z, d, T'$ )

1. {
  2. If ( $Br(T_z, s, d)$  is different from  $Br(T_0, s, d)$ )
  3. {
  4. Determine node  $Sw\_node(T_0, d)$  and configure in parallel the ports of the nodes between each destination node  $d$  and the node  $Sw\_node(T_0, d)$
  5. Configure the node  $Sw\_node(T_0, d)$  to feed the new branches and interrupt the flow of the old branches
  6. Remove, in parallel, the part of the old branches in Which flow is interrupted
  7. If  $T_0$  and  $T_z$  have different topologies
  8. {
  9. Configure  $s$  to interrupt the part between  $s$  and  $Sw\_node(T_0, d)$  which is different to
  10.  $s$  Configure, simultaneously, the ports of the nodes between  $s$  and  $Sw\_node(T_0, d)$  to remove this part
  11. }
  12. }
  13. }
- 

---

**Algorithm 4: TRwRC algorithm**

---

*TRwRC* ( $T_0, T_z$ )

1. {
  2. For each destination  $d$
  3. {
  4. *ReconfBr* ( $T_0, T_z, d, T'$ )
  5. *ReconfBr* ( $T', T_z, d, T$ )
  6. }
  7. Configure simultaneously the ports of the nodes of the tree  $T$  to convert the wavelength used into the initial wavelength.
  8. }
- 

**VIII. Performance Evaluation**

1. Criteria and Method

We use COST 239 (see Fig. 4) and NSF networks (see Fig. 5) to evaluate our algorithm. Well-known topologies are used in many similar papers as performance evaluation topologies.

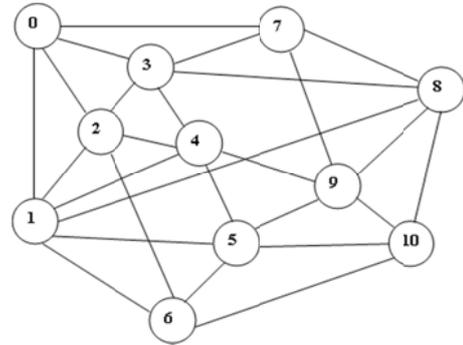


Fig. 4: Topology of COST 239 network

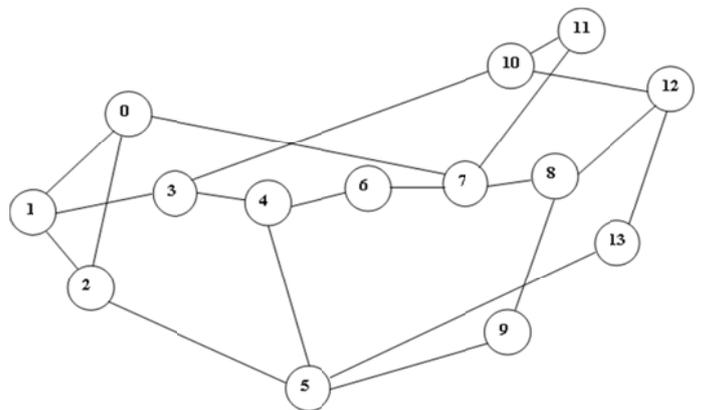


Fig.5: Topology of NSF Network

The initial and final trees are selected arbitrarily (they are called pairs of trees, above). One is the minimum spanning tree (MST) computed with Prim's heuristic (Prim, 1957). This algorithm calculates a good approximation of the Steiner tree for a multicast group. The other is the shortest paths tree (SPT) which uses Dijkstra's algorithm (Dijkstra, 1959) for the multicast group. We compare our algorithm *TRwRC* to *BpBAR\_2* and *MBB\_2*. The criteria are: the duration of the process, the additional resources used and the

number of interruptions during the tree configuration process. In addition to the above criteria, also used in our previous work, we define a new criterion which is the number of unsolved tree configuration problems. An unsolved problem is a pair of trees which a certain algorithm cannot reconfigure without data flow interruption. The number of unsolved problems depends on the number of available wavelengths per link. If the number of pairs of trees generated is  $nb\_tirage$ :

$$nb\_unsol\_prob(w) = \sum_{k=1}^{nb\_tirage} \alpha_k^w(T_0, T_z) \quad (1)$$

where  $\alpha_k^w(T_0, T_z) = 1$  if for the  $k^{th}$  pair of trees,  $w$  available wavelengths per link is not enough to reconfigure all the tree branches without flow interruption and  $\alpha_k^w(T_0, T_z) = 0$  otherwise. In the evaluation,  $nb\_tirage$  is chosen experimentally and is equal to 5000.

## 2. Results and Analysis

The results of the different evaluations are noted in Tables III to VII. Tables III to V show the performance results when the initial tree is SPT and the final tree MST. Tables VI and VII show the results obtained when we swap the algorithms used to compute the trees. The initial tree is now MST and the final tree SPT.

Table III: Average and standard deviation of the number of steps for each reconfiguration

CONF.	COST 239			NSF NETWORK		
	AVG	SD	MIN/MAX	AVG	SD	MIN/MAX
PROCESS	(STEP)	(STEP)	(STEP)	(STEP)	(STEP)	(STEP)
<i>BpBAR_2</i>	18.07	12.09	6/51	22.62	17.35	6/74
<i>TRwRC</i>	18.31	11.61	7/54	22.68	16.15	7/79
<i>MBB2</i>	8.32	2.42	6/17	8.58	2.04	6/17

Table IV: Average and standard deviation of additional cost for each reconfiguration process

CONF.	COST 239			NSF NETWORK		
	AVG	SD	MIN/MAX	AVG	SD	MIN/MAX
PROCESS	(WL)	(WL)	(WL)	(WL)	(WL)	(WL)
<i>BpBAR_2</i>	166.84	195.57	10/861	370.14	448.16	9/2273
<i>TRwRC</i>	107.52	96.45	13/430	227.85	220.44	13/1104
<i>MBB2</i>	85.95	30.13	18/177	122.72	40.34	18/229

Table V: Average and standard deviation of interruption duration for each reconfiguration

CONF.	COST 239			NSF NETWORK		
	AVG	SD	MIN/MAX	AVG	SD	MIN/MAX
PROCESS	(STEP)	(STEP)	(STEP)	(STEP)	(STEP)	(STEP)
<i>BpBAR_2</i>	0	0	0/0	0	0	0/0
<i>TRwRC</i>	0	0	0/0	0	0	0/0
<i>MBB2</i>	2.85	1.47	0/6	3.42	1.86	0/6

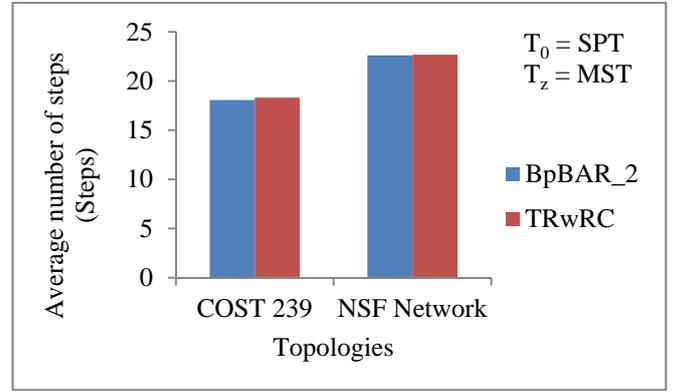


Fig.6: Setup duration of each algorithm in COST 239 and NSF Network

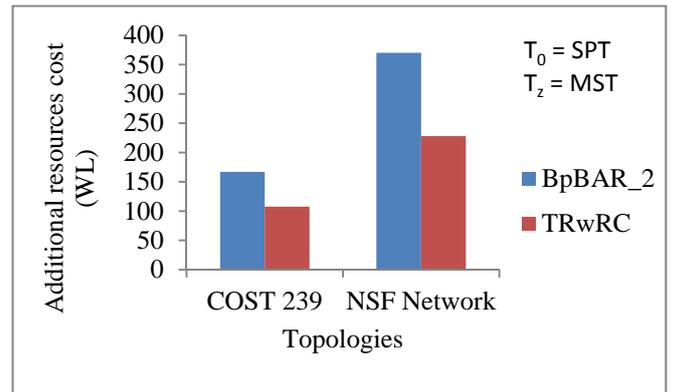


Fig.7: Additional resource cost of each algorithm in COST 239 and NSF Network

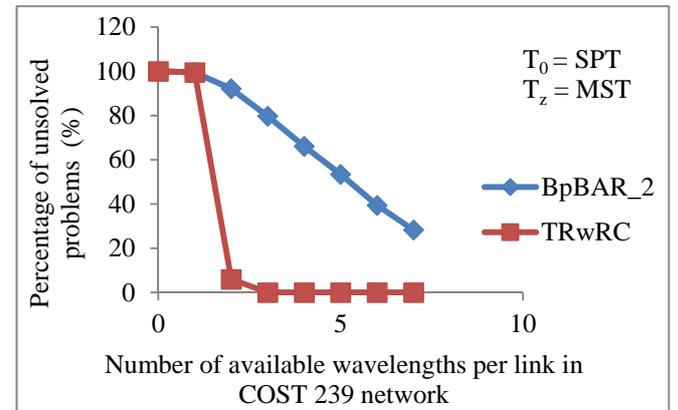


Fig.8: Percentage of unsolved problems in COST 239

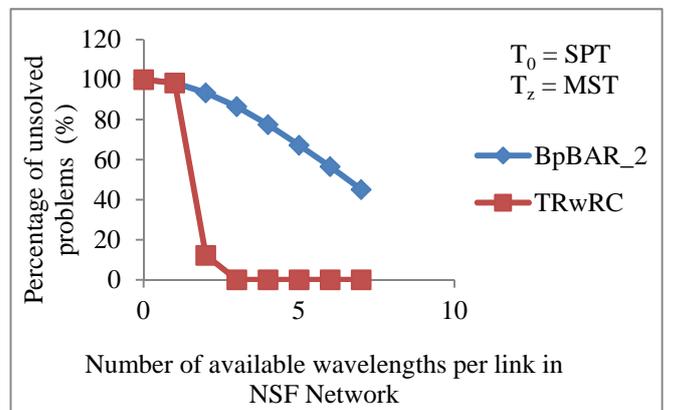


Fig.9: Percentage of unsolved problems in NSF Network

The results of the performance evaluations noted in Table III show that the number of steps produced by *TRwRC* is approximately equal to the number of steps produced by *BpBAR\_2* (Fig. 6), but is greater than those produced by *MBB\_2*. This is due to the fact that *TRwRC* is a back and forth tree reconfiguration process. Table IV shows that tree reconfiguration with *TRwRC* uses more resources than reconfiguration with the *MBB\_2* process. This is due to the fact that *MBB\_2* uses the same initial wavelength during all the process. So, it requires no additional wavelength while *TRwRC* uses additional resources to establish new branches. Table V shows that *MBB\_2* causes data flow interruption while *TRwRC* maintains continuity of data flow. The continuity of data flow is due to the properties of the *BpBAR* process used by the *TRwRC* algorithm. A good tree reconfiguration algorithm is one which does not interrupt the flow toward any destination during the reconfiguration. Thus, the *TRwRC* algorithm is much better than the *MBB\_2* algorithm.

*BpBAR\_2* and *TRwRC* maintain the continuity of data flow and produce approximately the same number of steps (see Fig. 6). The important improvement is shown in Figs. 7, 8, and 9. Figs. 8 and 9 show the evolution of the percentage of unsolved problems with the *BpBAR\_2* *TRwRC* algorithms as a function of the number of available wavelengths per link. They show that when the number of available wavelengths per link is under or equal to two, generally the *BpBAR*-based algorithms produce tree reconfiguration with data flow interruption. But the performance produced by *TRwRC* is better than the performance produced by *BpBAR\_2*. These figures show also that when the number of available wavelengths per link is equal to three, *TRwRC* allows reconfiguration of any tree without data flow interruption regardless of the topology, while *BpBAR\_2* requires more available resources. Moreover, Fig. 7 shows that the cost of additional resources produced by *TRwRC* is less than the cost of additional resources produced by *BpBAR\_2*. So *TRwRC* does not use more additional resources than *BpBAR\_2* during its process. This is due in the fact that each configuration produced by *TRwRC* contains at most two light-trees and one pre-established path, while with *BpBAR\_2* the number of trees increases with the number of branches.

When we swap the trees used to compute the tree configuration, Figs. 10 to 12 show that the same performance gains are kept: *TRwRC* reconfigures any pair of trees when the number of available wavelengths per link is equal to or higher than three

and does not use more additional resources compared to *BpBAR\_2*.

Table VI: Average and standard deviation of reconfiguration duration for each reconfiguration process

CONF.	COST 239			NSF NETWORK		
	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)
<i>BpBAR_2</i>	19.02	12.81	6/61	22.59	16.43	6/80
<i>BpBAR_4</i>	20.33	12.80	7/62	22.21	12.92	7/62
<i>MBB2</i>	10.90	3.20	6/17	8.77	2.4	6/17

Table VII: Average and standard deviation of additional cost for each reconfiguration process

CONF.	COST 239			NSF NETWORK		
	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)	AVG (STEP)	SD (STEP)	MIN/MAX (STEP)
<i>BpBAR_2</i>	92.75	100.65	5/542	183.88	219.63	5/1088
<i>BpBAR_4</i>	93.15	89.67	8/427	116.42	127.78	9/701
<i>MBB2</i>	62.98	32.24	9/149	10.51	40.01	18/229

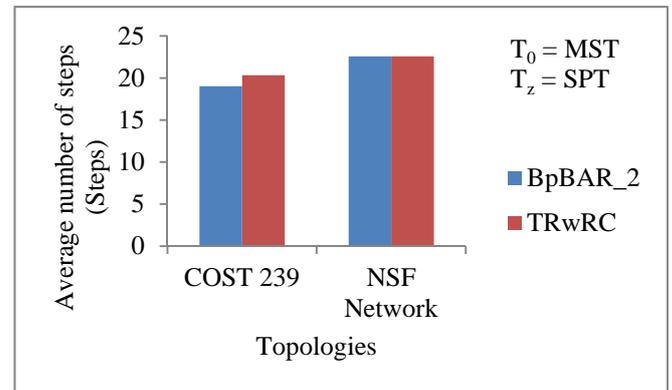


Fig.10: Setup duration of each algorithm in COST 239 and NSF Network

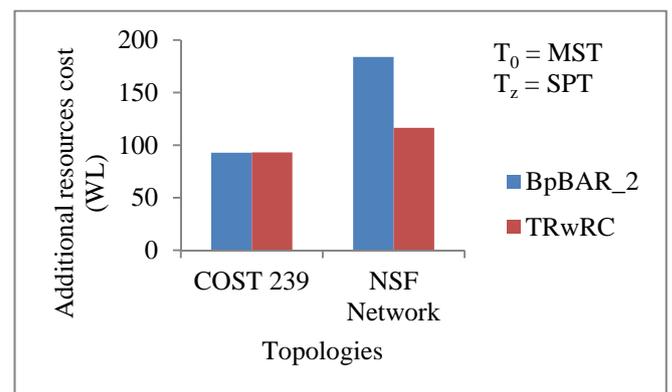
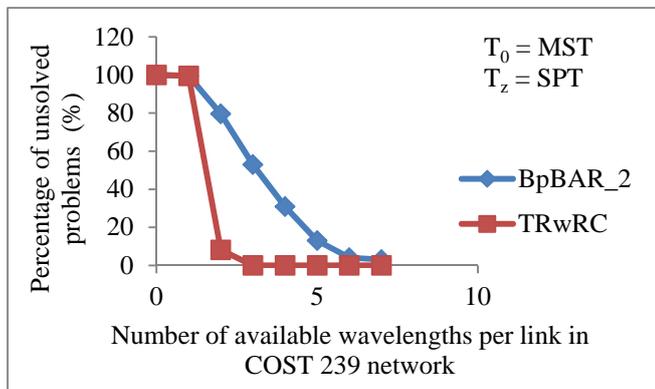


Fig.11: Additional resource cost of each algorithm in COST 239 and NSF Network



**Fig.12:** Percentage of unsolved problems in COST 239

## IX. Conclusion

Tree reconfiguration in WDM optical networks is studied in this paper. In the literature, the well-known path reconfiguration process *MBB* is proved to induce flow interruption. Similarly, adaptation of *MBB* to tree reconfiguration with no additional wavelengths may induce interruption. Even more, the tree branches may increase the probability of cycle outbreak and thus interruption. Adaptation of *MBB* to tree reconfiguration with an additional wavelength is also inefficient because it produces flow interruption. In our previous work, we proposed a branch-by-branch algorithm (*BpBAR\_2*) which aims to reconfigure trees without data flow interruption, and reduces the reconfiguration setup duration and the network resources used during the reconfiguration process. The availability of network resources was not considered in this previous work. In this paper, we consider tree reconfiguration problems in a network wherein the number of available wavelengths is limited. In this context, the algorithm *BpBAR\_2* becomes inefficient. In this paper, we propose *TRwRC* which allows reconfiguration of trees when each optical link has a limited number of available wavelengths. *TRwRC* maintains the continuity of the flow since it uses the appropriate switching node  $Sw\_node(T_0, d)$  and the combined switching operation to feed the new branches. The reconfiguration setup duration produced is also good compared to the previous algorithm. *TRwRC* uses, at most, two trees per light-forest to span all the destinations during the different configurations produced. Thus, it allows tree reconfiguration in all-optical networks with limited resources. It also reduces the cost of used resources during the reconfiguration process. When the number of available wavelengths per link is three (at least), any tree reconfiguration without data flow interruption is possible with *TRwRC*.

## REFERENCES

- I. Baldine and G. N. Rouskas, "Dynamic Reconfiguration Policies for WDM Networks", 18th Annual Joint Conference of the IEEE Computer and Communications Societies 1999 (INFOCOM), Vol. 1, pp. 313–320.
- I. Baldine and G. N. Rouskas, "Traffic Adaptive WDM Networks: A Study of Reconfiguration Issues", *IEEE/OSA Journal of Light Wave Technology*, Vol. 19 No. 4, 2001, pp. 433–455.
- N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse, "Tradeoffs in Routing Reconfiguration Problems", 12<sup>th</sup> International Symposium on Algorithmic Aspects of Telecommunications (AlgoTel), 2010.
- D. Coudert, F. Hue, and J-S. Sereni, "Reconfiguration of the Routing in WDM Networks with Two Classes of Services", 13<sup>th</sup> International Conference on Optical Network Design and Modeling (ONDM), 2009, pp. 1–6.
- B. Cousin, J. C. Adépo, S. Oumtanaga, and M. Babri, "Tree Reconfiguration Without Lightpath Interruption in WDM Optical Networks", *International Journal of Internet Protocol Technology*, Vol. 7 No. 2, 2012, pp. 85–95.
- B. Cousin and M. Molnar, "Fast Reconfiguration of Dynamic Networks", International Workshop on Dynamic Networks, 2007.
- E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *NumerischeMathematik*, Vol. 1 No. 1, 1959, pp. 269–271.
- A. Ding and G-S. Poo, "A survey of optical multicast over WDM networks", *Computer Communications*, Vol. 26 No. 2, 2003, pp. 193–200.
- S. Jawhar, B. Cousin, and S. Lahoud "Effect of Splitting Factor on the Generated Multicast Trees in Optical Networks", International Conference on Network Communication and Computers (ICNCC 2011).
- T. Kárász, "Consolidation Strategies of Provisioning Oriented Optical Networks", *Journal of Optical Networking*, Vol. 5 No. 6, 2006, pp. 445–462.
- T. Kárász and Z. Pándi, "Optimal Reconfiguration of Provisioning Oriented Optical Networks", 3<sup>rd</sup> International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks (HET-NETs), 2005.

- R. Malli, X. Zhang, and C. Qiao, "Benefit of Multicasting in All-Optical Networks", Photonics East (ISAM, VVDC, IEMB). International Society for Optics and Photonics, 1998.
- B. Mukherjee, "WDM Optical Communication Network: Progress and Challenges", IEEE Journal on Selected Areas in Communication, Vol. 18 No. 10, 2000, pp. 1810–1824.
- M. Perényi, P. Soproni, T. Cinkler, and D. Larrabeiti, "Regular Reconfiguration of Light-Trees in Multilayer Optical Networks", 12<sup>th</sup> International Conference on Optical Networking Design and Modeling (ONDM), 2008, pp. 1–6.
- R. C. Prim, "Shortest Connection Networks and Some Generalizations", Bell System Technical Journal, Vol. 36, 1957, pp.1389–1401.
- F. Solano, "Analyzing Two Different Objectives of the WDM Network Reconfiguration Problem", in: IEEE Global Communications Conference (Globecom), 28<sup>th</sup> IEEE Conference on Global Telecommunications, 2009.
- F. Zhou, M. Molnar, and B. Cousin, "Is Light-Tree Structure Optimal for Multicast Routing in Sparse Splitting WDM Networks?", 18th IEEE International Conference on Computer Communications and Networks (ICCCN 2009).
- F. Zhou, M. Molnar, B. Cousin, and C. Qia, "Cost Bounds and Approximation Ratios of Multicast Light-trees in WDM Networks", Journal of Optical Communications and Networking, Vol. 3 No. 4, 2011, pp. 323–334.