Light-tree reconfiguration without flow interruption in sparse wavelength converter network

Amanvon Ferdinand Atta, Bernard Cousin, Joël Christian Adépo, Souleymane Oumtanaga

Light-tree Reconfiguration without Flow Interruption in a Sparse Wavelength Converter Network

Amanvon Ferdinand Atta[1*], Bernard Cousin[2†], Joël Christian Adépo[1,3†] and Souleymane Oumtanaga[1†]

[1*]Laboratoire de Recherche en Informatique et Télécommunication, Institut National Polytechnique Félix Houphouët-Boigny, Yamoussoukro, Côte d'Ivoire.
[2]Institut de Recherche en Informatique et Systèmes Aléatoires, Université de Rennes 1, Rennes, France.
[3]Unité de Recherche et d'Expertise Numérique, Université Virtuelle de Côte d'Ivoire, Abidjan, Côte d'Ivoire.

*Corresponding author(s). E-mail(s): amanvon.atta@inphb.ci;
Contributing authors: bernard.cousin@irisa.fr;
joel.adepo@uvci.edu.ci; Souleymane.oumtanaga@inphb.ci;
†These authors contributed equally to this work.

Abstract

Network reconfiguration is an important task in WDM optical networks, which enables the optimisation of network resources. With the growing demand for multicast applications (e.g., distance learning, IPTV), this study focuses on the reconfiguration of the routing of a multicast connection. In this study, the path of the multicast connection is represented by a light-tree. A light-tree reconfiguration consists of migrating an optical flow from a light-tree to a new one. However, it is very difficult to automate light-tree reconfiguration without flow interruption. Flow interruption is undesirable for network operators. Therefore, the problem studied here is to find a sequence of operations in order to migrate an optical flow from a light-tree to a new one without flow interruption in a sparse wavelength converter network. For solving the light-tree reconfiguration problem, we

propose a method based on a sub-tree approach. The comprehensive simulation results demonstrated the effectiveness of our method.

Keywords: Light-tree Reconfiguration, Flow Interruption, Wavelength Converter, Flow Migration, Optical Network

1 Introduction

A WDM network is made of optical nodes interconnected by WDM links. Wavelength Division Multiplexing (WDM) technology [Brackett, 1990] divides the bandwidth of an optical fiber into several wavelength channels [Pandya, 2020b], so that several users can transmit data at different wavelengths through the same fiber simultaneously. Hence, the WDM network is the dominant type of network to fulfil the bandwidth requirements of applications [Bajpai et al., 2020]. In addition, wavelength conversion is a technique that can reduce the probability of blocking in a network using WDM [Subramaniam et al., 1996]. This technique is made possible by the use of a node component called a wavelength converter. A wavelength converter is a node component that takes a signal at a given wavelength as an input and converts it into a signal using another wavelength [Iness and Mukherjee, 1999]. Due to the high cost of this node component [Petale and Jaisingh, 2018], in practice network operators equip a limited number of optical nodes with wavelength converters [Zakouni et al., 2019]. A WDM optical network in which only some optical nodes in the network are capable of performing wavelength conversion is called a sparse wavelength converter network [Pinto-Roa et al., 2015]. In this network, two types of connections are allowed: unicast (point-to-point) and multicast (point-to-multipoint) connections.

Multicast is an effective technique for simultaneously transmitting information from one source to multiple destinations. Most multicast applications such as weather forecasts, online video games, online banking, video conferencing and IPTV are increasingly popular with users [CISCO, 2020, Pradhan et al., 2017]. In order to support multicast in WDM optical networks, some requirements from both the data plane (hardware) and the control plane (software) must be satisfied. On the data plane, ideally, each optical node (also called a node) in the network would possess light splitters [Das et al., 2021]. A light splitter is a node component that can split an optical signal (also called an optical flow) into two or more outgoing links [Yang et al., 2016]. In this work, we assume that each network node is equipped with light splitters because a light splitter is relatively inexpensive [Lin et al., 2013]. On the control plane, a routing solution must be computed to transfer an optical flow from one source node to multiple destination nodes. A typical routing solution is a light-tree, which is a point-to-multipoint all-optical path established in the network [Zhou et al., 2018]. This type of all-optical path is created by allocating the same wavelength on every link of the tree.

Changes in traffic demand by users, network failures (at nodes or links) and the deployment of new network resources can force a network operator to re-optimize the allocation of network resources [Li and Wu, 2015, Wu, 2011]. To do this, the network operator may have to configure new paths with some optical resources in order to transfer an optical flow and release some optical resources used by the working paths: it is said that the network operator reconfigures the network.

Network reconfiguration is widely considered one of the most important tasks in all types of WDM networks [Wu, 2011]. Indeed, it allows the network operator to meet the ever-increasing needs of users while effectively managing its limited resources. A reconfiguration task can be performed either on unicast connections or on multicast connections or on both connections. The reconfiguration of unicast connections has been extensively studied in recent years according to the survey in [Li and Wu, 2015]. In addition, given the growing demand for multicast applications, we focus on multicast connection reconfiguration.

Let one multicast connection be established by a light-tree (called a working light-tree) on which an optical flow passes through a network. The problem of multicast connection reconfiguration studied here consists of migrating the optical flow (or flow) from the working light-tree to a new pre-calculated light-tree without flow interruption from a source node toward one or more destination nodes. Flow interruption is not recommended because the network operator must reimburse the customers concerned by flow interruption with a penalty [Valentini et al., 2019]. This reconfiguration problem becomes difficult to solve because some resources (i.e. wavelength channels) required by the new light-tree are already allocated to the working light-tree [Adépo et al., 2016] and some are not. This dependency between some resources of both light-trees requires going through one or more transient light-trees using spare wavelengths and some operations such as wavelength conversion to avoid flow interruption [Cousin et al., 2012]. Wavelength conversion cannot be performed on all nodes because in a sparse wavelength converter network, some nodes do not have wavelength converters. A spare wavelength is a wavelength not required by the set of all established routes (including the working light-tree) and the new light-tree [Khanam et al., 2019]. As the wavelength is a valuable resource, our method should use spare wavelengths parsimoniously.

To date, some methods of light-tree reconfiguration using a so-called branch approach [Cousin et al., 2012] have been proposed to migrate a flow from a working light-tree to a new light-tree without flow interruption and within a reasonable computation time while using spare wavelengths as few as possible. However, the branch approach is not adapted if only a small number of network nodes are capable of performing wavelength conversion. Therefore, we propose a method using a sub-tree approach. The key contributions of this paper are summarized as follows:

1. Identification of categories of sub-tree pairs from a pair of light-trees.

2. For each category of sub-tree pairs, an algorithm is proposed to find the sequence of operations required to reconfigure a sub-tree pair belonging to this category.
3. A Sub-tree by Sub-tree (i.e. Sub-tree approach) Reconfiguration Algorithm denoted by *SbSRA* to solve light-tree reconfiguration problem.

The remainder of this paper is organized as follows. Section 2, presents the operations that can be used to configure an optical node. Section 3 presents the formal specification of the problem. Section 4 discusses the related works. Section 5 explains the contributions of this study in detail. The evaluation of our work is presented in Section 6. Section 7 concludes our work.

2 Fundamental concepts

In this section, we first present the set of operations [Cousin et al., 2012] used in reconfiguration process of a light-tree pair, then some definitions that are useful for understanding the remainder of this paper.

Operations for node configuration:

The six operations that can be used for node configuration are the following :

1. The addition of wavelength switching on node x is denoted by $ADD(x_{w_i}^{p_i,\{p_{o1},...,p_{ok}\}})$, configures node x by adding it, the capability to switch an optical flow of wavelength w_i received by its input port p_i to multiple output ports (i.e. $\{p_{o1},...,p_{ok}\}$) without changing wavelength w_i . In other words, this operation notifies node x to switch a given optical flow from one input port to one or multiple output ports.
2. The deletion of wavelength switching on node x is denoted by $DEL(x_{w_i}^{p_i,\{p_{o1},...,p_{ok}\}})$, configures node x by deleting it, the capability to switch an optical flow of wavelength w_i received by its input port p_i to multiple output ports (i.e. $\{p_{o1},...,p_{ok}\}$) without changing wavelength w_i. In other words, this operation notifies node x to not longer switch a given optical flow from one input port to one or multiple output ports.
3. The wavelength conversion on node x is denoted by $CONV(x_{w_i,w_o}^{p_i,\{p_{o1},...,p_{ok}\}})$, configures node x to switch the optical flow of wavelength w_i received on its input port p_i to multiple output ports (i.e. $\{p_{o1},...,p_{ok}\}$) after converting wavelength w_i to another wavelength w_o.
4. The multi-changeover on node x is denoted by $MULT\_CHG(x_{w_i,w_o}^{p_i,p_{o1},\{p_{o2},...,p_{ok}\}})$. For an optical flow of wavelength w_i arriving at the input port p_i of node x and outgoing to its output port p_{o1}, this operation allows the changeover of the optical flow from p_{o1} to multiple output ports (i.e. $\{p_{o2},...,p_{ok}\}$) on the wavelength w_o. The wavelength w_i may be different from the wavelength w_o.
5. The convergence on node x is denoted by $CONVG(x_{w_i}^{\{p_{i1},p_{i2}\},\{p_{o1},...,p_{ok}\}})$. Convergence allows the wavelength flow w_i to converge to the same node x

by two of its input ports (p_{i1} and p_{i2}) without changing the output ports (i.e. $\{p_{o1}, ..., p_{ok}\}$).

6. The non-convergence on node x is denoted by $NCONVG(x_{w_i}^{\{p_{i1},p_{i2}\},\{p_{o1},...,p_{ok}\}})$. Non-convergence is the operation that cancels the effect of the convergence operation. In other words, it guarantees that the flow of wavelength w_i enters only through the input port p_{i2} (stopping the flow from entering by p_{i1}) without changing the output ports (i.e. $\{p_{o1}, ..., p_{ok}\}$).

It is important to note that the multi-changeover operation is the atomic combination of the deletion of wavelength switching and either the addition of wavelength switching or wavelength conversion. Also, a bad sequencing of these six operations can interrupt the flow.

Definition 1 ST is a sub-tree of the tree T if it connects a node of T to one or more other nodes of T. Branch B of tree T rooted at node r is the segment of T that connects r to a leaf node of T. Therefore, a branch is a particular type of sub-tree.

Definition 2 A changeover node is a node on which the multi-changeover operation can be applied.

Definition 3 A wavelength channel or channel denoted by $(x, po_x, w_i) - > (y, pi_y, w_i)$ is a fiber link that connects node x to node y and on which a wavelength w_i is busy on the output port po_x of node x and on the input port pi_y of y. A wavelength semi-channel or semi-channel $(x, po_x, -) \| - > (y, pi_y, w_i)$ is a fiber link that connects x to y on which a wavelength w_i is free on an output port po_x of x and busy on the input port pi_y of y.

Definition 4 The pre-establishment of a sub-tree consists of preparing it to transport a flow by configuring all the nodes of this sub-tree. In other words, just after having pre-established a sub-tree, no flow is yet passing through it. In fact, a pre-established sub-tree contains at least one semi-channel at its root.

3 Problem specification

3.1 System model and assumptions

A sparse wavelength converter network is modeled as an undirected graph $G(V, E)$. V is the set of network nodes and E is the set of network links. $V = V_{\bar{c}} \cup V_c$, where $V_{\bar{c}}$ and V_c represent respectively the set of nodes that do not have the wavelength conversion capability and the set of nodes that have the wavelength conversion capability. Let $C(r, D, w_c)$ be a multicast connection established in the network by using wavelength w_c. Node r is the source node of connection C and D is the set of destination nodes of this connection. Also, w_c is a wavelength that belongs to the set of wavelengths W usable in each link

of the network. The working light-tree used to transmit the optical flow from node r to the elements of D is modeled by a tree denoted by $T_0 = (V_0, E_0)$, rooted at r and spanning D. With $V_0 \subseteq V$, $E_0 \subseteq E$, $r \in V_0$ and $D \subseteq V_0 \setminus \{r\}$. The new pre-calculated light-tree (using the same wavelength as the new light-tree) to which the flow is to be migrated is also modeled by a tree denoted by $T_f = (V_f, E_f)$, rooted at r and spanning D. With $V_f \subseteq V$, $E_f \subseteq E$, $r \in V_f$ and $D \subseteq V_f \setminus \{r\}$. Our model assumes that each network node has the full capability of splitting the optical signal and the set of spare wavelengths W_s is not empty, with $W_s \subset W$.

3.2 Problem formulation

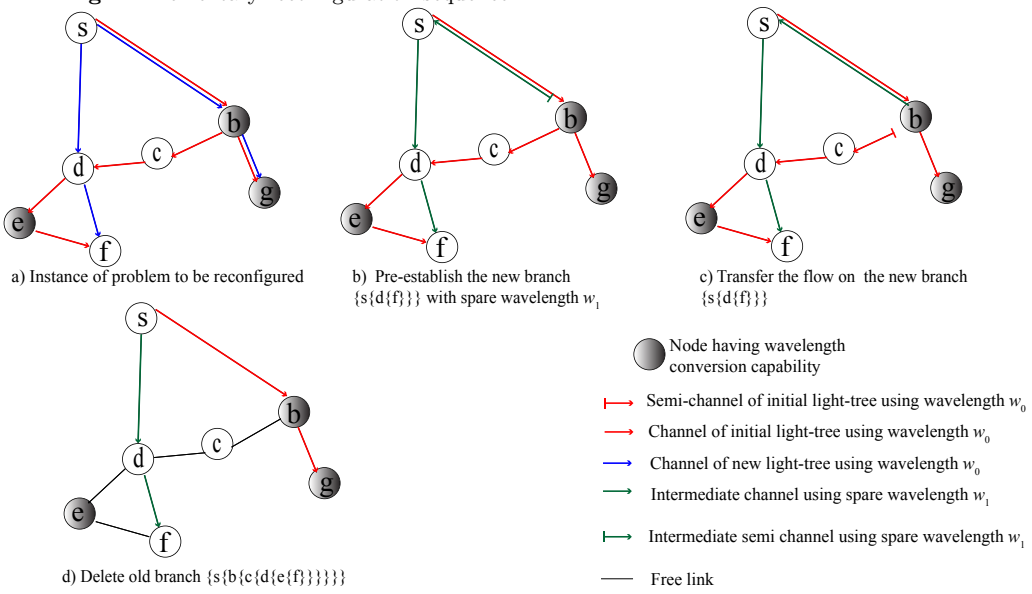
The problem of light-tree reconfiguration, which consists of migrating (an optical) flow from the working light-tree T_0 to the pre-calculated new light-tree T_f is equivalent to find a sequence of light-trees $< T_0, T_1, T_2, ..., T_{f-1}, T_f >$. Indeed, T_0 and T_f share some wavelength channels. In other words, to establish T_f, it is necessary to establish some wavelength channels already assigned to T_0. This situation of sharing wavelength channels can be compared with the coding problem when the two values of two variables have to be swapped. The solution to this problem involves the use of an transient variable (here an transient wavelength channel that can use a spare wavelength) to assign the value of a variable (here a wavelength channel of T_0) to another variable (here a wavelength channel of T_f). Note that each transient light-tree of the light-tree sequence to be found is composed by a set of transient wavelength channels. At step i of the reconfiguration process, The set of transient wavelength channels is obtained by applying configuration operations on at least one node of $V_{i-1} \cup V_f$, where V_{i-1} is the set of nodes of the light-tree T_{i-1}. T_{i-1} represents the current light-tree. Therefore, our problem is to find a sequence of configuration operations. Note that the current light-tree is the light-tree that is used to transmit an optical flow at a given step. Thus, in the first step, the current light-tree is the working light-tree. In the rest of this paper, a configuration operation is called an operation. In summary, the problem can be formulated as follows:

- **Given :** The working light-tree T_0 on which the flow passes, the new light-tree T_f on which the flow is to be transferred at the end of reconfiguration.
- **Our goal :** Find the sequence of operations $TREE\_OP\_SEQ =< O_1, O_2, ..., O_{k-1}, O_k >$ which if it is executed, it permits the migration of an optical flow from T_0 to T_f without flow interruption and within a reasonable computation time while using as few spare wavelengths as possible. With O_k, the *k-th* operation to be executed.

4 Related works

The problem of reconfiguring multicast routing in WDM networks has not yet been sufficiently addressed as mentioned in the survey [Wu, 2011]. In [Luekijna

Fig. 1 Elementary reconfiguration sequence



and Saivichit, 2007], the authors proposed some methods to find a new optimal light-tree to transfer an optical flow. One of the pioneering work was carried out in [Perényi et al., 2008]. The authors solved the problem of reconfiguring trees where the leaves of the tree change dynamically. However, all works aforementioned does not address how to switch the flow from one light-tree to another without flow interruption.

For reconfiguring unicast routing in connection-oriented networks without flow interruption, Make-Before-Break (MBB) [Coudert et al., 2009] is one of the most widely used policies. MBB consists of first pre-establishing the new path, then switching the flow to the new path before releasing the resources used by the old path [Jaumard, Duong, Armolavicius, Morris and Djukic, 2019]. In the context of multicast routing reconfiguration, an adaptation of MBB called *MBB_1* [Cousin et al., 2012] can be applied by considering each branch of the light-tree as a path. *MBB_1* reconfigures a light-tree pair (a working light-tree, a new light-tree) by applying MBB for each pair of branches (old branch belonging to the working light-tree, new branch belonging to the new light-tree). However, if the branches of a pair of branches share wavelength channels then *MBB_1* may induce an interruption of the flow. To remedy the flow interruption caused by *MBB_1*, some methods to solve the light-tree reconfiguration problem based on a branch approach [Cousin et al., 2012] have been proposed. This Branch approach uses an Elementary Reconfiguration Process (ERP) of each pair of branches (old branch, new branch). The old branch is a branch of the working light-tree and the new branch is a branch of the new light-tree. The elementary reconfiguration of a pair of branches consists of:

1. Choose a changeover node: the changeover node algorithm [Cousin et al., 2012] is proposed to select the changeover node which guarantees the continuity of the flow.
2. Use a spare wavelength w_1 to pre-establish the new branch from this changeover node.
3. At the changeover node, transfer the flow from the old branch to the new branch.
4. Delete the old branch (i.e. release the wavelength on the old branch).

Figure 1 describes the elementary reconfiguration process of a pair of branches on a reconfiguration problem instance. This instance (see Figure 1.a) concerns the multicast connection $(s, \{f, g\}, w_0)$. The working (or initial) light-tree T_0 is represented by the set of links in solid red and the final (or new) light-tree T_f by the set of links in solid blue. For this instance, only the pair of branches $(\{s\{b\{c\{d\{e\{f\}\}\}\}\}\}, \{s\{d\{f\}\}\})$ leading to node f needs to be reconfigured because the branch leading to node g is identical on both light-trees. For the only pair of branches to be reconfigured, the changeover node chosen by ERP is b. Figures 1.b to 1.d highlight the steps 2 to 4 of the elementary reconfiguration process. In the rest of this paper, to simplify the notation of each operation, and without loss of generality, the input port and output ports are respectively replaced by the name of the parent node and the names of the child nodes on a tree. So ERP returns an operation sequence using spare wavelength w_1 denoted by $ERP_{w_1}(\{s\{b\{c\{d\{e\{f\}\}\}\}\}\}, \{s\{d\{f\}\}\}) =< ADD(s^{b,\{d\}}_{w_1}), ADD(d^{s,\{f\}}_{w_1}), ADD(f^{d,\{-\}}_{w_1}), MULT\_CHG(b^{s,c,\{s\},}_{w_0,w_1}), DEL(s^{b,\{d\}}_{w_0}), DEL(c^{b,\{d\}}_{w_0}), DEL(d^{c,\{e\}}_{w_0}), DEL(e^{d,\{f\}}_{w_0}), DEL(f^{e,\{-\}}_{w_0}) >$.

Two methods based on the branch approach were proposed. These methods are *BpBAR_1* [Cousin et al., 2012] and *BpBAR_2* [Cousin et al., 2012]. These two methods carry out the reconfiguration process in two phases (i.e. a round trip). The elementary reconfiguration process is applied to each pair of branches (requiring reconfiguration) in the first phase using the spare wavelength w_1 and then again in the second phase using the wavelength w_0 requires by the new light-tree. The sequence of operations returned by the first phase of *BpBAR_1* concerning the pair of branches leading to node f (see Figure 1.a) is equal to $ERP_{w_1}(\{s\{b\{c\{d\{e\{f\}\}\}\}\}\}, \{s\{d\{f\}\}\})$. Second phase gives as output the sequence of operations $ERP_{w_0}(\{s\{b\{c\{d\{e\{f\}\}\}\}\}\}, \{s\{d\{f\}\}\}) = ¡ ADD(d^{s,\{f\}}_{w_0}), ADD(f^{d,\{-\}}_{w_0}), MULT\_CHG(s^{-,s,\{s\}}_{w_1,w_0}), DEL(s^{b,\{d\}}_{w_1}), DEL(d^{s,\{f\}}_{w_1}), DEL(f^{d,\{-\}}_{w_1})¿$. Thus, the sequence of operations requires to reconfigure this pair of branches is constituted by the sequence $ERP_{w_1}(\{s\{b\{c\{d\{e\{f\}\}\}\}\}\}, \{s\{d\{f\}\}\})$ follows by the sequence $ERP_{w_0}(\{s\{b\{c\{d\{e\{f\}\}\}\}\}\}, \{s\{d\{f\}\}\})$. The sequence of operations *TREE_OP_SEQ* returned by *BpBAR_1* is the sequence of operations required by the pair of branches leading to node f because only this pair of branches requires reconfiguration. The nodes s and f are respectively the root node and the leaf node of this pair of branches. Therefore, the input port (respectively the output ports) cannot specified concerning *ADD* operations and

MULT_CHG operations (respectively *DEL* operations and *ADD* operations) on the node s (respectively on the node f). This is expressed by a dash.

BpBAR_1 and *BpBAR_2* allow reconfiguration without flow interruption of the multicast connexion. However, *BpBAR_1* has a rather long reconfiguration time. To deal with this, *BpBAR_2* performs some operations in parallel. *BpBAR_1* and *BpBAR_2* allow light-tree reconfiguration without flow interruption but can lead a rather high cost of spare wavelengths used.

In order to reduce the cost of using spare wavelengths, the *TRwRC* [Cousin et al., 2014] method was proposed. However, it can generate a rather long reconfiguration time because it generates a series of light-tree sets to be reconfigured. In order to improve *TRwRC* by reducing its reconfiguration time, the *RCBRwPR* [Adépo et al., 2016] method was proposed. *RCBRwPR* proposed a sequence of operations to configure in parallel a set of branches and this operation sequence requires at least two different spare wavelengths at each step.

In summary, *MBB_1* is an MBB-based method that generates flow interruption because it not appropriate if a pair of branches share links. Branch-based methods require that any changeover node (when it is different from the root of the light-tree pair) selected at a reconfiguration step has wavelength conversion capability. This requirement is not easy to satisfy in the context of a sparse wavelength converter network. For instance, in Figure 1.b, if node b does not have the wavelength conversion capability then no other node can be chosen as a changeover node. As a result, pre-establishment is not possible without flow interruption. Therefore, using a branch-based method for the light-tree reconfiguration may generate flow interruption in a sparse wavelength converter network. To our knowledge, no method has been proposed today for solving the light-tree reconfiguration problem in a sparse wavelength converter network.

5 Our contribution

For solving the formulated problem (see section 3.2), previous branch-based methods were proposed to find a sequence of operation sequences of every pair of branches from the light-tree pair to be reconfigured. However, the sequence of operations returned by a branch-based method generates flow interruption if a changeover node does not have wavelength conversion capability. Therefore, we propose a method based on a sub-tree approach. This approach consists of selecting sequentially a sub-tree pair from a pair of light-trees (current light-tree, new light-tree) and finding an operation sequence for this sub-tree pair. In the sub-tree approach, the changeover node of a sub-tree pair is either the root of the light-tree pair or a non-leaf node x of this light-tree pair such that node x has the wavelength conversion capability or not. We remind you that the current light-tree is the light-tree that is used to transmit an optical flow at a given step. Note that, this light-tree is equal to the working light-tree specified in section 3.2. The solution of our method is the sequence of

the operation sequences required by each sub-tree pair. Because our method must use as few spare wavelengths as possible, the sub-tree pairs have been categorized. In the following subsections, we propose a categorization of sub-tree pairs and an algorithm to find an operation sequence for each category of sub-tree pairs. Then we propose an algorithm that takes as input a pair of light-trees and returns an operation sequence to reconfigure this pair.

5.1 Categorization of sub-tree pairs

At each step, a pair of sub-trees (ST_c, ST_f) must be selected and the appropriate operation sequence for (ST_c, ST_f) must be found. This sequence of operations should require as few spare wavelengths as possible. Note that execution of an operation sequence must make it possible to pre-establish the new sub-tree ST_f (i.e. a sub-tree of the new light-tree T_f), to transfer the flow from the current sub-tree ST_c (i.e. a sub-tree of the current light-tree T_c) to ST_f and to delete ST_c (i.e. to release the wavelength used by the wavelength channels of ST_c). A spare wavelength can be required for the pre-establishment of ST_f. To use spare wavelengths the fewest times, the operations in the sequence should use a spare wavelength only if it is required. Depending on whether a spare wavelength is required, there are three possible sub-tree pair categories:

1. The category of sub-tree pairs with identical links: A pair of sub-trees (ST_c, ST_f) belongs to this category if the links of the current sub-tree ST_c are the same as the links of the sub-tree ST_f. Such a sub-tree pair does not require to be reconfigured. Therefore, a spare wavelength is not required.
2. The category of sub-tree pairs with disjointed links: A pair of sub-trees (ST_c, ST_f) belongs to this category if ST_f does not share any links with ST_c. Therefore, the pre-establishment of ST_f does not require the use of a spare wavelength.
3. The category of sub-tree pairs with shared links: A pair of sub-trees (ST_c, ST_f) belongs to this category if ST_f shares links (but not all links) with ST_c. Therefore, the pre-establishment of ST_f requires the use of spare wavelengths.

In the following, for each of the two last categories of sub-tree pairs, we explain how to select[1] a sub-tree pair of this category and how to find the sequence of operations required for this selected pair of sub-trees.

5.1.1 Reconfiguration of a sub-tree pair with disjointed links

Let (ST_c, ST_f) be a selected sub-tree pair with disjointed links, where ST_c is a sub-tree of the current light-tree T_c and ST_f a sub-tree of the new light-tree T_f. This implies that (1) (ST_c, ST_f) is rooted at a divergent node n. In addition, in order to avoid flow interruptions, it is necessary that (2) $CG(T_c, T_f)_c = CG(T_c, T_f)_f = CG(ST_c, ST_f)$. Requirement (2) confers to

[1]Reader can see more details about the selection of sub-tree pairs here : https://hal.archives-ouvertes.fr/hal-02374828

(ST_c, ST_f) the property of non-interruption of the flow. $CG(T_c, T_f)_c$ denotes the set of convergent nodes of the light-tree pair (T_c, T_f) which belongs to ST_c, $CG(T_c, T_f)_f$ denotes the set of convergent nodes of the light-tree pair (T_c, T_f) that belongs to ST_f and $CG(ST_c, ST_f)$ denotes the set of convergent nodes of the light-tree pair (T_c, T_f) which belongs to both ST_c and ST_f. By definition, a divergent node belongs to (T_c, T_f) and has at least one child node on T_c that is different from its child nodes on T_f. By definition, a convergent node belongs to (T_c, T_f) and its parent node on T_c is different from its parent node on T_f. In the remainder of this paper, we refer to $Select\_SDL(T_c, T_f, n)$ as a function that implements the selection (refer to (1) and (2)) of the Sub-tree pair with Disjointed Links (SDL). For the light-tree pair (T_c, T_f) and the divergent node n as input, $Select\_SDL(T_c, T_f, n)$ returns (ST_c, ST_f) with disjointed links that does not interrupt flow.

Reconf_SDL (see Algorithm 1) finds an operation sequence to reconfigure the sub-tree pair with disjointed links (ST_c, ST_f). We introduce some use-

Algorithm 1 *Reconf_SDL*

Input: ST_c, ST_f // ST_c : The current sub-tree; ST_f : The new sub-tree
Output: SDL_OP_SEQ // the sequence of operations for (ST_c, ST_f) reconfiguration

1: $SDL\_OP\_SEQ = null$. $w_c = Wavelength(ST_c)$; // w_c : The current wavelength used by ST_c
2: **for** each $x \in IN(ST_f)$ **do**
3: $SDL\_OP\_SEQ \parallel= ADD(x_{w_c}^{Parent(x,ST_f),CHILD(x,ST_f)})$;
4: **end for**
5: **for** each $x \in CG(ST_c, ST_f)$ **do** // $CG(ST_c, ST_f)$: The set of convergent nodes of (ST_c, ST_f)
6: $SDL\_OP\_SEQ \parallel= CONVG(x_{w_c}^{Parent(x,ST_c),Parent(x,ST_f),CHILD(x,ST_c)})$;
7: **end for**
8: $x = \text{Root}(ST_c, ST_f)$; // x: The root node of (ST_c, ST_f) and also the changeover node
9: $SDL\_OP\_SEQ \mathrel{+}= MULT\_CHG(x_{w_c,w_c}^{\text{Parent}(x,ST_c),\text{CHILD}(x,ST_c),\text{CHILD}(x,ST_f)})$;
10: **for** each $x \in CG(ST_c, ST_f)$ **do**
11: $SDL\_OP\_SEQ \parallel= NCONVG(x_{w_c}^{Parent(x,ST_c),Parent(x,ST_f),CHILD(x,ST_c))}$;
12: **end for**
13: **for** each $x \in IN(ST_c)$ **do**
14: $SDL\_OP\_SEQ \parallel= DEL(x_{w_c}^{Parent(x,ST_c),CHILD(x,ST_c)})$;
15: **end for**
16: Return SDL_OP_SEQ ;

ful notations for a good understanding of the remainder of our work. Let

Wavelength(T) be the wavelength used by tree T, *Parent*(x, T) be the input port that connect node x to its parent node on tree T, *CHILD*(x, T) be the set of output ports that connects node x to its child nodes on tree T, *IN*(T) be the set of intermediate nodes of tree T and Root(T_0, T_1) be the common root node of tree T_0 and tree T_1. Note that node x is an intermediate node of tree T if x is not the root node of T and *CHILD*(x, T) is not empty. In short, to reconfigure the pair of sub-trees (ST_c, ST_f) with disjointed links, *Reconf_SDL* returns an operation sequence composed by : a set of *ADD* operations to be executed in parallel (1), a set of *CONVG* operations to be executed in parallel (2), a *MULT_CHG* operation (3), a set of *NCONVG* operations to be executed in parallel (4) and a set of *DEL* operations to be executed in parallel (5). It is important to note that a set of operations added to SDL_OP_SEQ via the $\|=$ operator (refer for instance to line 3) means that these operations must be executed in parallel. The execution of (1) and (2) should allow the pre-establishment of the new sub-tree ST_f. (3) allows the flow to be transferred from ST_c to ST_f. (4) and (5) delete the current sub-tree ST_c . The sequence of operations returned by Algorithm 1 does not require a spare wavelength because ST_c and ST_f do not share links: the sequence of operations requires the same wavelength (i.e. w_c). In addition, this operation sequence is composed by several operations to be executed in parallel. Therefore, this sequence is fast to execute.

5.1.2 Reconfiguration of a sub-tree pair with shared links

Let (ST_c, ST_f) be a sub-tree pair with shared links rooted at node n, where ST_c is a sub-tree of the current light-tree T_c and ST_f a sub-tree of the new light-tree T_f. This implies that (1) $\exists\ m \in CG(ST_c, ST_f)$. In other words, the set of convergent nodes of the sub-tree pair (ST_c, ST_f) is not empty. (2) $n =$ *First_ancestor*(m, T_c, T_f) such as $Dest(n, m, T_c) = Dest(n, m, T_f)$ and $n \in V_c$. *First_ancestor*(m, T_c, T_f) is the youngest ancestor of node m on T_c which is also an ancestor of node m on T_f. $Dest(n, m, T_c)$ (respectively $Dest(n, m, T_f)$) is the set of destination nodes such that each element of this set has at least one ancestor on the segment of T_c (respectively T_f) from n to m. We remind that V_c is the set of network nodes such that each element of this set has wavelength conversion capability. Requirement (2) confers to (ST_c, ST_f) the property of non-interruption of the flow. Note that if *First_ancestor* returns null then the root node n of sub-tree pair (ST_c, ST_f) is the root of the light-tree pair (T_c, T_f). We refer to *Select_SSL*(T_c, T_f, m) as a function that implements the selection (refer to (1) and (2)) of a Sub-tree pair with Shared Links (SSL). For the pair of light-trees (T_c, T_f) and the convergent node m as input, *Select_SSL*(T_c, T_f, m) returns (ST_c, ST_f) with shared links that does not cause flow interruptions. The Reconf_SSL function (See Algorithm 2) takes as input a pair of sub-trees (ST_c, ST_f) with shared links. Then it returns an operation sequence denoted by SSL_OP_SEQ that does not interrupt flow. ST_f shares links with ST_c, so its pre-establishment requires a spare wavelength denoted by w_s. Let AVAILABLE(T_0, T_1) be the set of available wavelengths that are common to

Algorithm 2 Reconf_SSL

Input: ST_c, ST_f

Output: SSL_OP_SEQ // Operation sequence for (ST_c, ST_f) reconfiguration

1: $SSL\_OP\_SEQ = null$. $w_c = Wavelength(ST_c)$; // w_c : The current wavelength used by ST_c
2: w_s = Random(AVAILABLE(ST_c, ST_f)) ; // w_s : A spare wavelength randomly selected
3: **for** each $x \in V'_f$ **do** // V'_f : The set of non-root nodes of ST_f
4: $SSL\_OP\_SEQ \parallel= ADD(x_{w_s}^{Parent(x,ST_f),CHILD(x,ST_f)})$;
5: **end for**
6: x = Root(ST_c, ST_f) ; // x: The root node of (ST_c, ST_f) and also the changeover node
7: $SSL\_OP\_SEQ \mathrel{+}= MULT\_CHG(x_{w_s,w_c}^{\text{Parent}(x,ST_c),\text{CHILD}(x,ST_c),\text{CHILD}(x,ST_f)})$;
8: **for** each $x \in V'_c$ **do** // V'_c : The set of non-root nodes of ST_c
9: $SSL\_OP\_SEQ \parallel= DEL(x_{w_c}^{Parent(x,ST_c),CHILD(x,ST_c)})$;
10: **end for**
11: Repeat line 3 to line 10 using now w_c as the spare wavelength and w_s as the current wavelength ;
12: Return SSL_OP_SEQ ;

T_0 and T_1. Note that w_s is randomly selected from AVAILABLE(ST_c, ST_f). If SSL_OP_SEQ obtained at line 10 is executed, then ST_f is configured using spare wavelength w_s. However, ST_f must use the same wavelength as ST_c (i.e. w_c) : SSL_OP_SEQ is not complete. Therefore, line 11 completes SSL_OP_SEQ by successively repeating instructions from line 3 to line 10, considering that the spare wavelength is w_c and w_s is the current wavelength.

5.2 Sub-tree based method for light-tree reconfiguration

5.2.1 Proposed algorithm

We now propose a Sub-tree by Sub-tree Reconfiguration Algorithm denoted by *SbSRA* (see Algorithm 3) to solve the formulated problem. Let us reminder that the objective here is to find the sequence of operations, which if executed, it allows the migration of an optical flow from the working light-tree T_0 to the new light-tree T_f without flow interruption and within a reasonable computation time while using as few spare wavelengths as possible. To do this, *SbSRA* sequentially selects a pair of sub-trees and finds an operation sequence appropriate for its reconfiguration. This ensures that at the end of Algorithm 3, *SbSRA* produces the sequence of the sequences of operations requires to reconfigure the pair of light-trees (T_0, T_f), which is given as input of this algorithm.

This sequence denoted by TREE_OP_SEQ must require as few spare wavelengths as possible. Therefore, Algorithm 3 starts by selecting the pairs of

Algorithm 3 Sub-tree by Sub-tree Reconfiguration Algorithm (*SbSRA*)

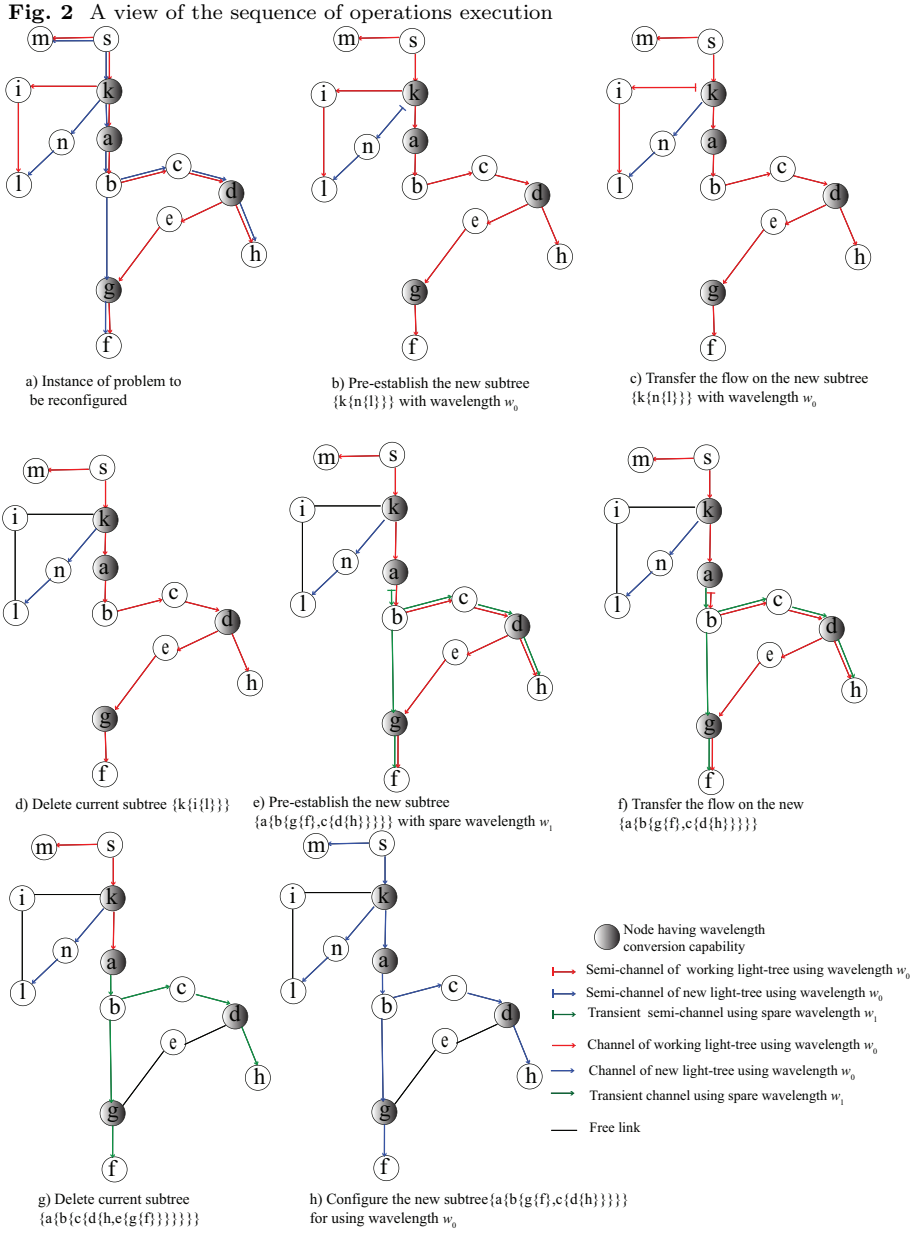
Input: T_0, T_f // T_0 : The working light-tree; T_f : The new light-tree
Output: TREE_OP_SEQ // The sequence of the sequences of operations
1: TREE_OP_SEQ = null. $T_c = T_0$; // T_c : The current light-tree
2: **while** $\overline{DG}(T_c, T_f)$!= null **do** // $\overline{DG}(T_c, T_f)$: The list of unvisited divergent nodes of the light-tree pair (T_c, T_f)
3: n = First element of $\overline{DG}(T_c, T_f)$;
4: (ST_c, ST_f) = Select_SDL(T_c, T_f, n) ; // (ST_c, ST_f) : A sub-tree pair with disjointed links
5: **if** (ST_c, ST_f) != (null,null) **then**
6: SDL_OP_SEQ = Reconf_SDL(ST_c, ST_f) ; // An operation sequence to reconfigure the sub-tree pair (ST_c, ST_f) with disjointed links
7: TREE_OP_SEQ += SDL_OP_SEQ ; // SDL_OP_SEQ is appended to TREE_OP_SEQ
8: $T_c = T_c - ST_c + ST_f$; // ST_f is added to T_c and ST_c is deleted from T_c
9: **end if**
10: **end while**
11: **while** $\overline{CG}(T_c, T_f)$!= null **do** // $\overline{CG}(T_c, T_f)$: The list of unvisited convergent nodes of the light-tree pair (T_c, T_f)
12: m = First element of $\overline{CG}(T_c, T_f)$;
13: (ST_c, ST_f) = Select_SSL(T_c, T_f, m) ; // (ST_c, ST_f) : A sub-tree pair with shared links
14: SSL_OP_SEQ = Reconf_SSL(ST_c, ST_f) ; // An operation sequence to reconfigure the sub-tree pair (ST_c, ST_f) with shared links
15: TREE_OP_SEQ += SSL_OP_SEQ ; // SSL_OP_SEQ is appended to TREE_OP_SEQ
16: $T_c = T_c - ST_c + ST_f$; // ST_f is added to T_c and ST_c is deleted from T_c
17: **end while**
18: Return *TREE_OP_SEQ* ;

sub-trees with disjointed links (which does not require spare wavelengths) and finding the appropriate operation sequence for each of them (refer from line 2 to line 10) while taking care to update TREE_OP_SEQ and the current light-tree T_c at each iteration. Note that, at each iteration, the first unvisited divergent node that is the first child of the root of the light-tree pair (T_c, T_f) is chosen as the root of the sub-tree pair with disjointed links to be selected. A divergent node is visited if it has been used for an iteration of the first loop. At the end of first loop (refer to line 10), if the light-tree pair (T_c, T_f) does not contain at least one unvisited convergent node then Algorithm 3 stops. Otherwise, the instructions contained in the second loop (refer from line 11 to line 17) are executed to select each sub-tree pair with shared links and to compute its sequence of operations to complete TREE_OP_SEQ. Note that if no operation in TREE_OP_SEQ concerns a convergent node of (T_c, T_f) then this convergent node is said unvisited.

5.2.2 Illustration of this algorithm

Let the problem instance shows in Figure 2.a. The set of links in solid red forms the working light-tree T_0 rooted at s. The set of links in solid blue forms the new light-tree T_f rooted at s. Note that initially, the current light-tree T_c is the working light-tree T_0. The first loop (refer from line 2 to line 10) is executed as follows. The list of unvisited divergent nodes of the light-tree pair (T_c, T_f) contains nodes k and d. Node k is the first divergent node which is a child node of the root node s. From divergent node k, Select_SDL(T_0, T_f, k) returns $(\{k\{i\{l\}\}\}, \{k\{n\{l\}\}\})$, which is a sub-tree pair with disjointed links, where $\{k\{i\{l\}\}\}$ is the current sub-tree ST_c and $\{k\{n\{l\}\}\}$ is the new sub-tree ST_f. Then, Reconf_SDL takes this pair of sub-trees as input and returns : OP_SEQ_1 = SDL_OP_SEQ $=<$ $\{ADD(n_{w_0}^{k,\{l\}})\}, \{CONVG(l_{w_0}^{\{i,n\},\{-\}})\}, MULT\_CHG(k_{w_0,w_0}^{s,i,\{n\}})$, $\{NCONVG(l_{w_0}^{\{i,n\},\{-\}})\}, \{DEL(i_{w_0}^{k,\{l\}})\} >$. Then, OP_SEQ_1 is appended to the sequence of the sequence of operations TREE_OP_SEQ. Figures 2.b to 2.d presents the execution of the sequence of operations OP_SEQ_1. The current light-tree is updated by replacing the current sub-tree with the new sub-tree in the current light-tree. The set of red and blue links forms (see Figure 2.d) the current light-tree T_c at the end of the first iteration of the first repeat loop. The new list of unvisited divergent nodes of the light-tree pair (T_c, T_f) contains only node $\{d\}$. Node g is the only convergent node descending from divergent node d on the current light-tree but does not descend from node d on the new light-tree. Therefore, From divergent node d, Select_SDL(T_c, T_f, d) does not return a sub-tree pair with disjointed links: it is the end of the first loop of Algorithm 3 because all divergent nodes have been visited. The convergent node g has not yet been visited. Therefore, the second loop (refer from line 11 to line 17) is executed as follows. Select_SSL(T_c, T_f, g) returns $(\{a\{b\{c\{d\{h, e\{g\{f\}\}\}\}\}\}\}, \{a\{b\{g\{f\}, c\{d\{h\}\}\}\}\})$ which is a sub-tree pair with shared links, where $\{a\{b\{c\{d\{h, e\{g\{f\}\}\}\}\}\}\}$ is the current sub-tree ST_c and $\{a\{b\{g\{f\}, c\{d\{h\}\}\}\}\}$ is the new sub-tree ST_f. Then, Reconf_SSL takes as input this pair of sub-trees and returns the sequence of operations using the spare wavelength w_1 : OP_SEQ_2 = SSL_OP_SEQ $=<$ $\{ADD(b_{w_1}^{a,\{g,c\}}), ADD(c_{w_1}^{b,\{d\}}), ADD(d_{w_1}^{c,\{h\}}), ADD(h_{w_1}^{d,\{-\}}), ADD(g_{w_1}^{b,\{f\}}),$ $ADD(f_{w_1}^{g,\{-\}})\}, MULT\_CHG(a_{w_0,w_1}^{k,a,\{a\}}), \{DEL(b_{w_0}^{a,\{l,c\}}), DEL(c_{w_0}^{b,\{d\}}),$ $DEL(d_{w_0}^{c,\{e,h\}}), DEL(h_{w_0}^{d,\{-\}}), DEL(e_{w_0}^{d,\{g\}}), DEL(g_{w_0}^{e,\{f\}}), DEL(f_{w_0}^{g,\{-\}})\},$ $\{ADD(b_{w_0}^{a,\{g,c\}}), ADD(c_{w_0}^{b,\{d\}}), ADD(d_{w_0}^{c,\{h\}}), ADD(h_{w_0}^{d,\{-\}}), ADD(g_{w_0}^{b,\{f\}}),$ $ADD(f_{w_0}^{g,\{-\}})\}, MULT\_CHG(a_{w_1,w_0}^{k,a,\{a\}}), \{DEL(b_{w_1}^{a,\{g,c\}}), DEL(c_{w_1}^{b,\{d\}}),$ $DEL(d_{w_1}^{c,\{h\}}), DEL(h_{w_1}^{d,\{-\}}), DEL(g_{w_1}^{b,\{f\}}), DEL(f_{w_1}^{g,\{-\}})\}$ $>$. Note that the operations placed between brackets in OP_SEQ_1 or OP_SEQ_2 must be executed in parallel. Then, TREE_OP_SEQ $=< OP\_SEQ_1, OP\_SEQ_2 >$. The current light-tree is updated (Figure 2.h). All convergent nodes have been visited : Algorithm 3 stops and returns the sequence of the sequences of operations TREE_OP_SEQ requires for all sub-tree pairs to be reconfigured. Figure 2.e to Figure 2.h give an execution view of OP_SEQ_2.

Fig. 2 A view of the sequence of operations execution



6 Performance study

6.1 Performance criteria

To determine the effectiveness of the proposed method, we compared our method (i.e., *SbSRA*) to *RCBRwPR* and *MBB_1*. Three criteria are used :

the duration of the reconfiguration process [Cousin et al., 2012], the cost of spare wavelengths [Cousin et al., 2012] and the average interruption rate. In the following, $Seq(T_0, T_f)$ denotes the sequence of operations obtained by any method.

The duration of a reconfiguration sequence denoted by $duration(Seq(T_0, T_f))$ is the total duration of operation constituting $Seq(T_0, T_f)$. It is expressed by Equation (1):

$$duration(Seq(T_0, T_f)) = \sum_{k=0}^{|Seq(T_0,T_f)|-1} duration(O_k) \tag{1}$$

$duration(O_k)$ is the duration of *k-th* operation contained in the sequence of operations $Seq(T_0, T_f)$. Without loss of generality, we assume that the duration of each O_k is unitary. Therefore :

$$duration(Seq(T_0, T_f)) = |Seq(T_0, T_f)| \tag{2}$$

In general, the execution of an operation contained in $Seq(T_0, T_f)$ allows either to create wavelength channels or to delete wavelength channels. These wavelength channels may use a spare wavelength. Let G_k be the graph whose set of links E_k represents the set of residual wavelength channels after the execution of the first *k-th* operations contained in $Seq(T_0, T_f)$. Therefore, the cost of spare wavelengths denoted by $add\_cost(Seq(T_0, T_f))$ is the total sum of spare wavelengths used by G_k graphs:

$$add\_cost(Seq(T_0, T_f)) = \sum_{k=0}^{|Seq(T_0,T_f)|-1} \sum_{j=0}^{|W_s|-1} \sum_{l=0}^{|E_k|} G_{k,l}^{w_s^j} \tag{3}$$

where W_s is the set of spare wavelengths (see section 3.1) that can be used in the reconfiguration process and

$$G_{k,l}^{w_s^j} = \begin{cases} 1 \text{ if the } l\text{-}th \text{ wavelength channel contained in } G_k \text{ used the spare wavelength } w_s^j \\ 0 \text{ otherwise.} \end{cases}$$

The interruption rate caused by execution of an operation O_k denoted by $interrupt\_rate(O_k)$ is the ratio of the number of destination nodes interrupted after the execution of O_k to the total number of destinations. $interrupt\_rate(O_k)$ is defined by Equation (4):

$$interrupt\_rate(O_k) = \frac{\sum_{d \in D} O_k(d)}{|D|} \tag{4}$$

where D is the set of destination nodes (see section 3.1) and

$$O_k(d) = \begin{cases} 1 \text{ if } O_k \text{ execution interrupts flow toward node } d \\ 0 \text{ otherwise.} \end{cases}$$

Therefore, the average interruption rate caused by the execution of operation sequence $seq(T_0, T_f)$ is defined by Equation (5) :

$$interrupt\_rate(seq(T_0, T_f)) = \frac{\sum_{k=0}^{|Seq(T_0,T_f)|-1} interrupt\_rate(O_k)}{|Seq(T_0, T_f)|} \quad (5)$$

6.2 Experimental setup

Our simulations are based on three real network topologies (see Table 1) that are widely used in the field of optical network research [Pavan et al., 2010]: NSFNET, which has a small size, GEANT, which has a medium size and CORONET [*The Internet Topology Zoo*, 2019], which has a large size.

Table 1 Topology size

Network	Number of nodes	Number of links
NSFNET	14	21
GEANT	40	75
CORONET	75	99

Each real topology is represented by a graph $G(V, E)$ implemented using Networkx [Hagberg et al., 2008], a python graph analysis package. V is the number of nodes and E is the number of links. Note that for each topology, the number of nodes that have the wavelength conversion capability $|V_c|$ is randomly selected (uniform law) belonging $[1; \frac{|V|}{2}[$. Moreover, without loss of generality, we assume that each link (fiber) supports at most $|W|$ wavelengths ($|W| = 16$).

For each topology, a Monte-Carlo simulation was run 5000 times to evaluate each performance criterion. One simulation consists of :

1. Randomly (uniform law) selects a node that is taken as the source node of a multicast connection.
2. Randomly (uniform law) selects a number of nodes that is used as the destination nodes of the multicast connection.
3. The working tree of the multicast connection is built by generating a tree of the shortest paths that spanning source and destination nodes using the Dijkstra algorithm [Dijkstra, 1959].
4. The new tree is built by generating a spanning tree of minimum weight using the Prim algorithm [Prim, 1957].

5. Both trees were assigned the same randomly selected wavelength (uniform law) in W.
6. This pair is used as an input by three methods : *MBB_1*, *RCBRwPR* and our method (i.e., *SbSRA*).

6.3 Analysis of Experimental Results

The results (Average (AVG), Standard Deviation (SD), Maximum (M) and Minimum (m)) of the performance criteria are listed in Table 2 to Table 4. The results of the performance evaluations noted in Table 2 show that all values concerned by our method (*SbSRA*) are equal to zero regardless of a given real topology (NSFNET, GEANT, CORONET). This confirms that *SbSRA* permits light-tree reconfiguration without flow interruption in sparse wavelength converter network. But, this Table and Figure 3 show that *MBB_1* and *RCBRwPR* interrupt the flow to a non-zero percentage of destinations: these methods caused flow interruption. In short, the performance study confirms that these last two methods are not usable for light-tree reconfiguration in sparse wavelength converter network.

Table 2 Average interruption rate (%) caused by the three methods

	NSFNET			GEANT			CORONET		
Methods	**AVG**	**SD**	**m/M**	**AVG**	**SD**	**m/M**	**AVG**	**SD**	**m/M**
SbSRA	0	0	0/0	0	0	0/0	0	0	0/0
RCBRwPR	19.5	22.33	0/66	8.87	9.17	0/77	13.37	13.22	0/77
MBB_1	25.72	16	2/83	20.69	13.75	0/96	44.38	21.95	2/96

Fig. 3 Comparison of the average interruption rate

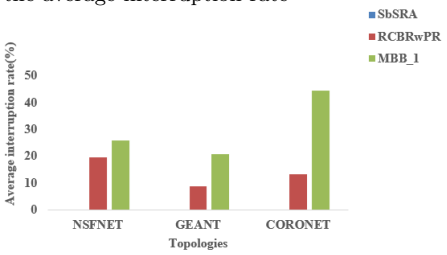


Table 3 shows that *RCBRwPR* provides the highest cost of spare wavelengths. This is due to the fact that this method requires to use at least two different spare wavelengths for the establishment of wavelength channels of the new light-tree while *SbSRA* requires to use a single spare wavelength and *MBB_1* does not require a spare wavelength. In addition, Figure 4 and Table 3 show that the cost of spare wavelengths increases with the size of the network

(so with the size of light-trees) for methods *RCBRwPR* and *SbSRA*. This finding is correct because, Equation (3) shows that the cost of spare wavelengths increases with the size of light-trees and the number of spare wavelengths used during the reconfiguration process.

Table 3 Spare wavelengths cost (WL) caused by the three methods

	NSFNET			GEANT			CORONET		
Methods	**AVG**	**SD**	**m/M**	**AVG**	**SD**	**m/M**	**AVG**	**SD**	**m/M**
SbSRA	6.09	4.35	0/20	22.79	14.98	0/77	42.03	19.15	0/93
RCBRwPR	24.43	10.62	6/48	67.8	55.43	4/264	160.73	92.55	8/434
MBB_1	0	0	0/0	0	0	0/0	0	0	0/0

Fig. 4 Comparison of the average of spare wavelengths cost

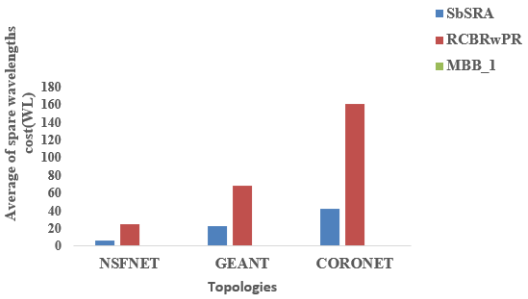
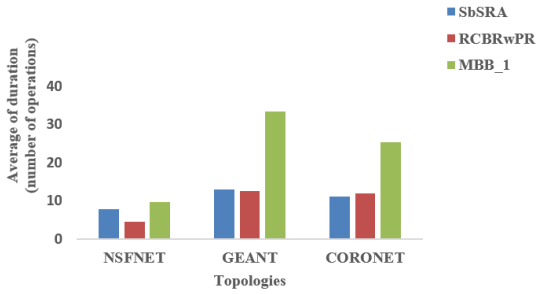


Table 4 and Figure 5 show that *MBB_1* has the highest duration. These results are due to the fact that at each reconfiguration step, *MBB_1*, *RCBRwPR* and *SbSRA* permits to reconfigure respectively the nodes of a single pair of branches, the nodes of a set of branch pairs and the nodes of a single sub-tree pair.

Table 4 Duration (number of operations) of the three methods

	NSFNET			GEANT			CORONET		
Methods	**AVG**	**SD**	**m/M**	**AVG**	**SD**	**m/M**	**AVG**	**SD**	**m/M**
SbSRA	7.97	2.81	5/12	13.11	5.65	5/27	11.17	5.51	5/28
RCBRwPR	4.62	2.14	2/9	12.56	8.88	2/39	11.92	5.83	2/23
MBB_1	9.78	3.54	6/21	33.46	15.35	6/63	25.35	12.58	6/60

Fig. 5 Comparison of the average of duration



7 Conclusion

In this work, we solve the problem, which consists of finding an operation sequence that allows an optical flow to be migrated from a working light-tree to a new light-tree without flow interruption and within a reasonable computation time while using the spare wavelengths as few as possible. Some methods based on the branch approach have been proposed previously. However, these methods are useful to solve the problem in the context where all nodes have the wavelength conversion capability. To our knowledge, no method had been proposed in the context where only some nodes have the wavelength conversion capability. In order to fill this lack of solution, we proposed the *SbSRA*, which is a method based on a sub-tree approach. In order to build the sequence of operations requires to solve the problem, *SbSRA* at first, adds in this sequence of operations, the sequence of operations requires by each sub-tree pair with disjointed links. Note that this kind of sub-tree pair does not require spare wavelengths. Next, if the sequence of operations is not complete then *SbSRA* adds in this sequence of operations, the sequence of operations requires by each sub-tree pair with shared links. Note that each sub-tree pair with shared (or disjointed) links selected has the property of non-interruption of the flow. Performance study has validated that our method is the only method that allows to reconfigure a pair of light-trees (working light-tree, new light-tree) without flow interruption in a sparse wavelength converter network. But on large networks, the duration of the reconfiguration and the cost of spare wavelengths may become not negligible. Therefore, in our future work, we will study the duration of the reconfiguration process and the cost of spare wavelengths could be further reduced.

References

Adépo, J. C., Aka, B. and Babri, M. [2016]. Tree Reconfiguration with Network Resources Constraint, *International Journal of Computer Science and Telecommunications* **7**(1): 1–4.

Bajpai, R., Iyer, S., Singh, S. P. and Sengar, S. [2020]. Energy-efficient and spectral-efficient mixed line rate optical WDM networks: a comparison, *International Journal of Communication Networks and Distributed Systems* **24**(4): 339–356.

Brackett, C. [1990]. Dense wavelength division multiplexing networks: principles and applications, *IEEE Journal on Selected Areas in Communications* **8**(6): 948–964.

CISCO [2020]. Cisco annual internet report (2018–2023) white paper, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internetreport/white-paper-c11-741490.html. Accessed: May 25, 2020. [Online].

Coudert, D., Huc, F., Mazauric, D., Nisse, N. and Sereni, J.-S. [2009]. Reconfiguration of the routing in wdm networks with two classes of services, *2009 International Conference on Optical Network Design and Modeling* pp. 1–6.

Cousin, B., Adépo, J. C., Babri, M. and Oumtanaga, S. [2014]. Tree Reconfiguration without Lightpath Interruption in Wavelength Division Multiplexing Optical Networks with Limited Resources, *International Journal of Computer Science Issues (IJCSI)* **11**(2): 7–17.

Cousin, B., Adépo, J. C., Oumtanaga, S. and Babri, M. [2012]. Tree reconfiguration without lightpath interruption in WDM optical networks, *International Journal of Internet Protocol Technology* **7**(2): 85–95.

Das, S., Rahbar, A., Wu, X. C., Wang, Z., Wang, W., Chen, A. and Ng, T. S. E. [2021]. Shufflecast: An optical, data-rate agnostic and low-power multicast architecture for next-generation compute clusters, *arXiv preprint arXiv:2104.09680* .

Dijkstra, E. W. [1959]. A note on two problems in connexion with graphs, *Numerische Mathematik* **1**(1): 269–271.

Hagberg, A., Swart, P. and S Chult, D. [2008]. Exploring network structure, dynamics, and function using networkx, *Proceedings of the 7th Python in Science Conference (SciPy2008),*, eds by Gäel Varoquaux Travis Vaught and J. Millman, Pasadena, CA USA, pp. 11–15.

Iness, J. and Mukherjee, B. [1999]. Sparse Wavelength Conversion in Wavelength-Routed WDM Optical Networks, *Photonic Network Communication* **1**(3): 183–205.

Jaumard, B., Duong, H. Q., Armolavicius, R., Morris, T. and Djukic, P. [2019]. Efficient Real-Time Scalable Make-Before-Break Network Re-Routing, *Journal of Optical Communications and Networking* **11**(3): 52–66.

Khanam, S., Dey, V., Chatterjee, M. and Bhattacharya, U. [2019]. An offline cost-efficient strategy for multicast traffic protection in WDM optical mesh networks to aid rapid recovery, *Optical Switching and Networking* **34**: 47–57.

Li, H. and Wu, J. [2015]. Survey of WDM network reconfiguration: topology migrations and their impact on service disruptions, *Telecommunication Systems* **60**(3): 349–366.

Lin, R., Zukerman, M., Shen, G. and Zhong, W.-D. [2013]. Design of Light-Tree Based Optical Inter-Datacenter Networks, *Journal of Optical Communications and Networking* **5**(12): 1443–1455.

Luekijna, K. and Saivichit, C. [2007]. Multicast traffic reconfiguration in wdm network for single node failure design, *The 9th International Conference on Advanced Communication Technology* **3**: 1833–1838.

Pandya, R. J. [2020b]. Survivable virtual topology search with impairment awareness and power economy in optical WDM networks, *International Journal of Communication Networks and Distributed Systems* **25**(1): 1–20.

Pavan, C., Morais, R. M., Ferreira da Rocha, J. R. and Pinto, A. N. [2010]. Generating Realistic Optical Transport Network Topologies, *Journal of Optical Communications and Networking* **2**(1): 80–90.

Perényi, M., Soproni, P., Cinkler, T. and Larrabeiti, D. [2008]. Regular reconfiguration of light-trees in multilayer optical networks, *2008 International Conference on Optical Network Design and Modeling* pp. 1–6.

Petale, S. and Jaisingh, T. [2018]. Optimal of wavelength converter deployment in wdm optical networks, *2018 3rd International Conference on Microwave and Photonics (ICMAP)*, pp. 1–2.

Pinto-Roa, D. P., Brizuela, C. A. and Barán, B. [2015]. Multi-objective routing and wavelength converter allocation under uncertain traffic, *Optical Switching and Networking* **16**: 1–20.

Pradhan, A. K., Keshri, S., Das, K. and De, T. [2017]. A heuristic approach based on dynamic multicast traffic grooming in WDM mesh networks, *Journal of Optics* **46**(1): 51–61.

Prim, R. C. [1957]. Shortest Connection Networks and some Generalizations, *Bell System Technical Journal* **36**(6): 1389–1401.

Subramaniam, S., Azizoglu, M. and Somani, A. [1996]. All-optical networks with sparse wavelength conversion, *IEEE/ACM Transactions on Networking* **4**(4): 544–557.

The Internet Topology Zoo, http://www.topology-zoo.org/dataset.html. Accessed: 2019-06-14.

Valentini, A., Vass, B., Oostenbrink, J., Csak, L., Kuipers, F., Pace, B., Hay, D. and Tapolcai, J. [2019]. Network resiliency against earthquakes, *2019 11th International Workshop on Resilient Networks Design and Modeling (RNDM)*, IEEE.

Wu, J. [2011]. A survey of WDM network reconfiguration: Strategies and triggering methods, *Computer Networks* **55**(11): 2622–2645.

Yang, W.-L., Yang, C.-T. and Huang, Y.-C. [2016]. Optimal and heuristic algorithms for all-optical group multicast in resource-constrained wdm networks, *International Journal of Communication Systems* **29**(15): 2292–2312.

Zakouni, A., Toumi, H., Saidi, A. and Mabrouk, A. [2019]. Wavelength assignment vs. wavelength converter placement in wavelength-routed optical WDM networks, *Procedia Computer Science* **160**: 766–771.

Zhou, F., Ju, M. and Ait-Ouahmed, A. [2018]. Joint optimization for multicast provisioning in mixed-line-rate optical networks with a column generation approach, *Journal of Lightwave Technology* **36**(3): 637–649.