Titre: Title:	Heuristiques taboues pour le routage et l'affectation de longueurs d'onde dans les réseaux optiques
Auteur: Author:	Christiane Félicité Dzongang
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Dzongang, C. F. (2004). Heuristiques taboues pour le routage et l'affectation de longueurs d'onde dans les réseaux optiques [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7246/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7246/
Directeurs de recherche: Advisors:	Philippe Galinier, & Samuel Pierre
Programme: Program:	Génie informatique

UNIVERSITÉ DE MONTRÉAL

HEURISTIQUES TABOUES POUR LE ROUTAGE ET L'AFFECTATION DE LONGUEURS D'ONDE DANS LES RÉSEAUX OPTIQUES

CHRISTIANE FÉLICITÉ DZONGANG
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)

JANVIER 2004

Your file Votre référence
ISBN: 0-612-89198-4
Our file Notre référence
ISBN: 0-612-89198-4

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

HEURISTIQUES TABOUES POUR LE ROUTAGE ET L'AFFECTATION DE LONGUEURS D'ONDE DANS LES RÉSEAUX OPTIQUES

présenté par : DZONGANG Christiane Félicité

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen composé de :

Mme. NICOLESCU Gabriela, Doctorat, présidente

M. GALINIER Philippe, Doct., directeur

M. PIERRE Samuel, Ph.D., codirecteur

M. CHAMBERLAND Steven, Ph.D., membre

REMERCIEMENTS

Je voudrais tout d'abord remercier mon directeur de recherche, M. Philippe Galinier, ainsi que mon codirecteur, M. Samuel Pierre, pour avoir cru que je pouvais mener à bien ce travail et pour leur patience dans les moments difficiles.

Je remercie également mes collègues du Laboratoire de Recherche en Réseautique et Informatique Mobile (LARIM), pour leurs critiques et leurs conseils. Merci tout spécialement à Désiré.

Ma reconnaissance va également à mes parents, pour tous les sacrifices qu'ils ont faits pour moi. Merci à mes frères et sœurs ainsi qu'à mes amis pour leur soutien permanent et leurs encouragements.

RÉSUMÉ

Au cours des dernières années, le monde des télécommunications a connu des changements qui ont créé un immense besoin en bande passante. Les réseaux optiques sont alors devenus le support privilégié des communications. Avec l'utilisation de la fibre optique, la bande passante disponible dans de tels réseaux est énorme. Pour des applications telles que Internet, les réseaux optiques assurent des transmissions à des débits très élevés et à un coût raisonnable. Ces capacités de transmission et de bande passante ont créé un intérêt grandissant pour le développement des réseaux optiques. En plus de la conception topologique et du dimensionnement des liens, les planificateurs de réseaux optiques doivent gérer le problème de routage. En effet, avec l'utilisation du multiplexage en longueur d'onde, dans les réseaux optiques, contrairement aux réseaux jusque là existant, en plus de trouver un chemin pour un trafic, il faut lui affecter une longueur d'onde. Ceci constitue le problème de routage optique qui fait l'objet de ce mémoire.

L'objectif de ce mémoire est de proposer une méthode de résolution pour le problème de routage optique. Nous avons appliqué la contrainte de continuité de longueurs d'onde pour les trafics et nous avons considéré le cas où l'ensemble des connexions à établir est connu dès le départ. Pour chaque panne simple de lien du réseau, notre but est de maximiser le nombre de demandes de connexions satisfaites, étant donné un nombre de longueurs d'onde disponibles sur chaque lien de fibre optique.

Le problème de routage et d'affectation de longueurs d'onde ayant été répertorié NP-complet, nous avons choisi de développer des heuristiques pour le résoudre. La solution de routage est basée sur la recherche des plus courts chemins et l'affectation des longueurs d'onde sur une méthode de coloriage de graphes utilisant la recherche taboue.

La méthode que nous avons développée est une méthode générique. Nous avons implémenté deux versions de cette méthode et nous avons fait des séries d'expériences pour analyser la performance, en mettant l'accent sur le nombre de trafics bloqués et les temps d'exécution. Nous avons étudié surtout l'effet du nombre de longueurs d'onde.

Nous avons remarqué que le fait d'augmenter le nombre de longueurs d'onde diminuait le nombre de trafics bloqués. Nous avons comparé les résultats obtenus pour trois réseaux, un réseau aléatoire, ARPANET et NSFNET, à ceux de l'algorithme H+. Dans l'ensemble, les gains par rapport à cet algorithme sont très bons. Nous avons remarqué que même pour nos deux algorithmes, les gains étaient parfois très élevés en ce qui concerne le nombre de trafics acheminés. Pour le réseau aléatoire, nous avons comparé nos résultats à la borne inférieure calculée avec le logiciel CPLEX, pour le cas où il n'y aurait pas de panne dans le réseau. Dans la majorité des cas, le résultat que nous avons obtenu pour un nombre de longueurs d'onde donné, était le même que celui fourni par CPLEX. Toutefois, les temps d'exécution de nos méthodes sont supérieurs à ceux de H+, mais assez raisonnables. Ces résultats étaient prévisibles dans la mesure où H+ utilise une heuristique de coloriage séquentielle moins performante qu'une heuristique basée sur la recherche locale comme les heuristiques taboues.

ABSTRACT

In the several past years, due to the tremendous progress made in the communications area, the demand of a greater bandwidth has grown. With the use of the optical fiber, the bandwidth available in optical networks is enormous. For applications such as Internet, and videoconference, optical networks are able to transmit information at a very high rate. These transmission capacities have created a growing interest for the development of optical networks. The design of an optical network is a long process witch mainly includes the topology design, the link dimensioning and the routing problem. With the introduction of wavelength multiplexing in optical networks, the routing problem now consists of finding a path to route a traffic and to assign a wavelength to that path. This is called the Routing and Wavelength Assignment problem (RWA) and will be study in this paper.

Our objective is to propose a solution to the optical routing problem, considering the wavelength continuity constraint and a given set of connections to satisfy. For each single link failure in the network and for a number of available wavelengths on each link, our goal is to maximize the number of routed connections. The routing and wavelength assignment has been proved to be NP-complete. That is why the solution we proposed is based on the use of heuristics. Finding the shortest paths for each traffic solves the routing part, the selection of possible routes. For the wavelength assignment, we proposed a graph coloring method using tabu search.

We developed a generic solving method. We implemented two versions of that method. For the performance analysis, we made several tests, and we compare our results with those of algorithm H+. We focused on the blocking rate and the execution time. Generally, our results were better than H+ except for the execution time witch are a little bit higher. These results were predictable, since H+ is based on a sequential graph coloring heuristic less powerful than a heuristic based on tabu search.

TABLE DES MATIÈRES

REMERCIEMENTS iv
RÉSUMÉ v
ABSTRACT vii
TABLE DES MATIÈRES viii
LISTE DES FIGURES xi
LISTE DES TABLEAUX xiii
LISTE DES SIGLES ET ABRÉVIATIONS xiv
CHAPITRE I INTRODUCTION 1
1.1 Définitions et concepts de base 1
1.2 Éléments de problématique 5
1.3 Objectifs de recherche 7
1.4 Esquisse méthodologique 7
1.5 Plan du mémoire 8
CHAPITRE II ROUTAGE ET AFFECTATION DE LONGUEURS D'ONDE DANS LES RÉSEAUX OPTIQUES 9
2.1 Caractéristiques de base des réseaux optiques 9
2.1.1 La technologie WDM 9
2.1.2 Le routage optique 11
2.1.3 La gestion des pannes 11
2.2 Le coloriage de graphes 14
2.2.1 Définitions 14
2.2.2 Les heuristiques gloutonnes 15
2.2.3 Les méthodes de recherche locale 15
2.2.4 Le coloriage de graphes et l'affectation de longueurs d'ondes 18
2.3 L'algorithme H+ 18
2.4 Le problème max-RWA 22
2.5 L'algorithme WCA 24

2.6 L'Approche quasi-statique 27
2.7 Autres méthodes de résolution 28
2.8 Algorithmes d'affectation de longueurs d'onde 29
CHAPITRE III MÉTHODES DE ROUTAGE PROPOSÉES 31
3.1 Définition du problème 31
3.1.1 Exemple illustratif 33
3.2 Heuristiques proposées 35
3.2.1 Schéma général 35
3.2.2 Routage et construction du graphe auxiliaire dans l'algorithme *TabouP* 40
3.2.3 Heuristique pour l'affectation des longueurs d'onde 43
3.2.4 Illustration de l'heuristique d'affectation de longueurs d'onde 47
3.3 Analyse de complexité des heuristiques proposées 51
3.3.1 Le routage et la construction du graphe auxiliaire 52
3.3.2 L'affectation des longueurs d'onde 53
CHAPITRE IV IMPLÉMENTATION ET RÉSULTATS 54
4.1 Structures de données utilisées 54
4.1.1 La classe CGraphe 54
4.1.2 La classe CReseau 55
4.1.3 La classe CGrapheAux 57
4.1.4 Les classes de coloriage 57
4.2 Données utilisées pour les tests 59
4.2.1 Les fichiers de topologie 60
4.2.2 Les fichiers de trafic 62
4.3 Plan d'expérience 62
4.3.1 Les facteurs de simulations 62
4.3.2 Choix des niveaux des facteurs 63
4.3.3 Types d'expérience choisies 64
4.3.4 Le comportement aléatoire de l'algorithme 65
4.4 Expérimentation et tests 66

4.4.1 Expériences « un facteur à la fois » 66
4.4.2 Le temps d'exécution des itérations pour le coloriage 70
4.4.3 Comparaison entre TabouN, Tabou1 et H+ 72
4.4.4 Temps d'exécution des algorithmes 75
4.4.5 Effet de la variation de la graine aléatoire 77
4.4.6 Comparaison avec CPLEX 80
CHAPITRE V CONLUSION 81
5.1 Synthèse des travaux 81
5.2 Limitations des travaux 82
5.3 Travaux futurs 83
BIBLIOGRAPHIE 84

LISTE DES FIGURES

Figure 1.1 Exemple de réseau optique 2
Figure 1.2 Exemple de topologie logique 4
Figure 1.3 Adjacence des chemins 5
Figure 2.1 Principe d'une liaison WDM 10
Figure 2.2 Protection 1+1 12
Figure 2.3 Protection 1 :*N* 13
Figure 2.4 Illustration de l'approche de réparation : initialisation aléatoire 17
Figure 2.5 Illustration de l'approche de réparation : état après un mouvement 17
Figure 2.6 Algorithme H 20
Figure 2.7 Algorithme H+ 21
Figure 2.6 Illustration de la transformation de graphe pour l'algorithme WCA 26
Figure 3.1 Illustration du problème de RWA 34
Figure 3.2 Schéma général de résolution 35
Figure 3.3 Illustration de l'approche de résolution pour le scénario sans panne t_0 40
Figure 3.4 Détail des étapes de routage et de construction du graphe auxiliaire pour *TabouP* 42
Figure 3.5 Heuristique d'affectation de longueurs d'onde 45
Figure 3.6 Sélection et exécution d'un mouvement 46
Figure 3.7 Configurations de départ pour l'affectation de longueurs d'onde : scénario sans panne 48
Figure 3.8 Illustration des itérations pour l'affectation de longueurs d'onde : scénario sans panne 50
Figure 3.9 Configuration de départ pour G_t : Panne du lien d-c 51
Figure 4.1 Exemple de fichier de trafic 56
Figure 4.2 Topologie du réseau de 20 nœuds et 25 liens 60
Figure 4.3 Topologie du réseau ARPANET 61
Figure 4.4 Topologie du réseau NSFNET 61

Figure 4.5 Trafics bloqués en fonction du nombre d'itérations pour le scénario de base λ=2, LT = 10 (réseau pseudo-aléatoire) 66
Figure 4.6 Trafics bloqués en fonction du nombre d'itérations pour tous les scénarios λ=2, LT = 10 (réseau pseudo-aléatoire) 67
Figure 4.7 Trafics bloqués en fonction de λ pour le scénario sans panne LT=10, M=1000 (réseau pseudo-aléatoire) 67
Figure 4.8 Trafics bloqués en fonction de λ pour tous les scénarios LT=10, M=1000 (réseau pseudo-aléatoire) 68
Figure 4.9 Trafics bloqués en fonction de LT pour tous les scénarios λ=2, M=1000 (réseau pseudo-aléatoire) 68
Figure 4.10 Trafics bloqués en fonction de LT pour tous les scénarios : λ=2, M=1000 (réseau pseudo-aléatoire) 69
Figure 4.11 Gains pour le réseau pseudo-aléatoire 73
Figure 4.12 Gains pour ARPANET 74
Figure 4.12 Gains pour NSFNET 75
Figure 4.13 Temps de calcul en fonction des itérations pour H+, Tabou1 et TabouN : λ=4, LT=15 (réseau pseudo-aléatoire) 77
Figure 4.14 Trafics bloqués pour le scénario de base en fonction de la graine aléatoire : λ=8, LT=15, M=10000 (réseau pseudo-aléatoire) 78
Figure 4.15 Nombre total de trafics bloqués en fonction de la graine aléatoire : λ=8, LT=15, M=10000 (réseau pseudo-aléatoire) 79

LISTE DES TABLEAUX

Tableau 4.1 Taille des réseaux utilisés 60
Tableau 4.2 Répartition du trafic dans les fichiers utilisés 62
Tableau 4.3 Niveaux des facteurs 64
Tableau 4.4 Nombre total de trafics bloqués pour LT et M variables : réseau pseudo-aléatoire 70
Tableau 4.5 Temps des itérations pour TabouN 71
Tableau 4.6 Temps des itérations pour Tabou1 72
Tableau 4.7 Nombre total de trafics bloqués pour H+, Tabou1, TabouN (réseau 20 nœuds, 25 liens) 73
Tableau 4.8 Nombre total de trafics bloqués pour H+, Tabou1, TabouN (ARPANET) .. 74
Tableau 4.8 Nombre total de trafics bloqués pour H+, Tabou1, TabouN (NSFNET)..... 75
Tableau 4.9 Temps d'exécution de TabouN 76
Tableau 4.10 Temps d'exécution de Tabou1 76
Tableau 4.11 Temps d'exécution de H+ 77
Tableau 4.12 Moyenne et écart type (scénario de base) 78
Tableau 4.13 Moyenne et écart type (tous les scénarios) 79

LISTE DES SIGLES ET ABRÉVIATIONS

WDM	Wavelength Division Multiplexing (Multiplexage en longueurs d'onde)
RWA	Routing and Wavelength Assignment (Routage et affectation de longueurs d'onde)
RWAP	Routing and Wavelength Assignment Problem (Problème de Routage et d'affectation de longueurs d'onde)
ILP	Integer Linear Program (Programme entier linéaire)

CHAPITRE I
INTRODUCTION

Depuis le début des années 1990, l'explosion des technologies de l'information et l'engouement des usagers de par le monde ont créé un immense besoin en bande passante. C'est dans ce contexte que les communications optiques basées sur l'utilisation de la fibre optique sont rapidement devenues le support privilégié des réseaux actuels. Toutefois, la technologie optique ne pouvant être déployée sur les équipements électroniques jusque là en vogue, il a fallu revoir la planification des réseaux ainsi que d'autres aspects tels que la maintenance et le développement des équipements optiques. L'un des points le plus souvent abordés par les chercheurs et qui sera traité dans ce mémoire est le routage optique. Dans ce chapitre, nous présenterons la problématique relative à ce sujet après avoir défini les concepts de base. Ensuite, nous préciserons nos objectifs de recherche et notre approche méthodologique pour les réaliser, puis nous finirons par une esquisse des grandes lignes du mémoire.

1.1 Définitions et concepts de base

Les *réseaux optiques* sont des réseaux de télécommunications de haute capacité basés sur la technologie optique qui permettent le transfert de données. Ils sont constitués de nœuds interconnectés par des liens de fibres optiques. L'ensemble des nœuds d'un réseau optique ainsi que les liens qui les lient défini la *topologie physique* du réseau. Il peut être représenté par un graphe dont les sommets correspondent aux nœuds du réseau. La Figure 1.1 illustre un exemple de réseau optique comportant cinq nœuds et sept liens de fibre optique. Un réseau est dit *connexe* si le graphe qui lui est associé est *connexe.* C'est-à-dire si pour tout couple de sommets i et j, il existe une *chaîne* joignant i et j. Une *chaîne* étant une séquence d'arcs. De la même manière, un réseau est *k-connexe au sens des nœuds* si pour toute paire de sommets il existe k *chaînes disjointes de nœuds* les reliant. C'est-à-dire k chaînes n'ayant aucun sommet en commun à part les extrémités.

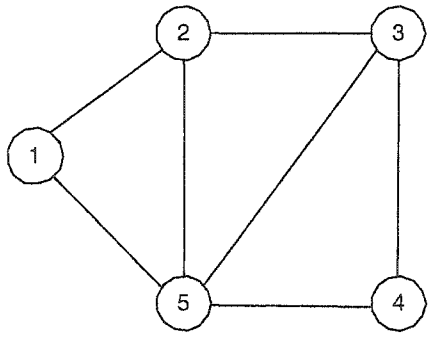


Figure 1.1 Exemple de réseau optique

La *fibre optique* est une mince fibre de verre qui offre la possibilité de transporter des informations sous forme d'ondes lumineuses. Elle présente plusieurs aspects intéressants parmi lesquels : l'atténuation du signal qui est faible de l'ordre de 0.2 dB/Km, et la bande passante disponible, supérieure à celle que l'on peut obtenir avec les câbles électriques.

Il existe deux types de fibre :

- la *fibre monomode* qui permet de couvrir des réseaux étendus et est principalement utilisée dans les WAN (*Wide Area Network*) et pour des applications nécessitant beaucoup de bande passante;
- la *fibre multimode* qui est surtout utilisée dans des LAN et des MAN. Elle est très répandue. Ce type de fibre se divise en deux sous catégories : les fibres dites à gradient d'indice qui sont utilisées surtout en Europe et les fibres à saut d'indice qui se retrouvent en Amérique du Nord.

Afin d'exploiter au maximum la bande passante de la fibre optique, le *multiplexage en longueur d'onde* a été introduit dans les réseaux optiques. Le *multiplexage* est une méthode qui permet de transmettre plusieurs messages par un même canal de transmission. Le *multiplexage en longueur d'onde* consiste à véhiculer sur une même fibre plusieurs signaux à des longueurs d'onde différentes. Dans les *réseaux optiques*, lorsque deux nœuds veulent entrer en communication, un chemin, soit une suite de liens allant de l'émetteur au récepteur, doit être assigné à travers le réseau [6]. Pour

que la communication soit établie, une longueur d'onde lui est aussi assignée et elle est acheminée sur les liens de fibres du réseau. La recherche de chemins pour établir la communication et l'affectation de longueur d'onde constituent le problème de *routage optique*, rencontré dans la littérature comme étant le problème *RWA* (*Routing and Wavelength Assigment*), qui veut dire routage et affectation de longueur d'onde.

Un ensemble de liens traversés par des données entre une source et une destination forme un chemin auquel est attribuée une longueur d'onde et est appelé un *chemin optique*. Un *chemin optique* peut être de deux types, *permanent* ou *robuste*. La différence entre les deux types vient de la façon de faire face à une panne. En effet, en cas de panne, le trafic correspondant à un *chemin robuste* pourra être routé suivant un autre chemin si possible, tandis que pour un *chemin permanent*, il sera tout simplement bloqué. Deux *chemins optiques* ne peuvent avoir la même longueur d'onde sur un lien. Si jamais une telle situation se présentait, il y aurait brouillage des signaux et les informations à transmettre seraient perdues. Pour pallier ce problème, certains équipements optiques permettent de faire une conversion de longueur d'onde entre l'entrée et la sortie au niveau d'un nœud, ce sont des convertisseurs de longueur d'onde. En l'absence de tels convertisseurs, un chemin optique occupera la même longueur d'onde entre l'origine et la destination, il s'agit de la contrainte de *continuité de longueur d'onde* [6][7]. Pour un chemin optique donné, la *conversion de longueur* d'onde à un nœud quelconque est le fait de changer la longueur d'onde entre l'entrée et la sortie du nœud. Il y a conversion de longueur d'onde à un nœud si deux ou plusieurs chemins optiques doivent emprunter un même lien partant de ce nœud et ont la même longueur d'onde. Il existe deux types de conversion de longueur d'onde : la *conversion totale* dans laquelle une longueur d'onde peut être remplacée par n'importe quelle autre, et la *conversion partielle* où les longueurs d'ondes intermédiaires sont choisies dans un ensemble dépendant de la longueur d'onde de départ. Avec la contrainte de continuité de longueurs d'onde, seuls deux chemins disjoints d'arcs ou disjoints de nœuds peuvent avoir la même longueur d'onde.

Lorsque deux nœuds du réseau communiquent, il y a des informations qui sont échangées, on dit qu'il y a un *trafic* entre ces nœuds. Un *trafic* entre deux nœuds

correspond à une demande de communication. Le nœud d'où part la demande est appelé *origine* et celui à qui les informations sont destinées est le nœud *destination*. Dans les réseaux, il existe deux modèles pour le trafic : *statique* ou *dynamique*. Dans les modèles *statiques*, l'ensemble des demandes à satisfaire est connu à l'avance et ne varie pas. Cet ensemble est appelé *matrice de trafic* et contient les paires origine-destination des demandes. Tandis que dans les cas dynamiques, les demandes entre deux nœuds source et destination arrivent dans le réseau une à la fois. Dans les modèles statiques à partir de la matrice de trafic, on définit la *topologie logique* du réseau. Elle est représentée par un graphe dont les sommets correspondent aux nœuds du réseau, deux sommets sont reliés s'ils constituent une paire origine-destination pour une demande de trafic [5] [15]. La Figure 1.2 représente la topologie logique associée au réseau illustré à la Figure 1.1 dans le cas où la matrice de trafic est formée des paires *(1,3), (2,1), (2,5), (3,4) et (5,4)*. Pour construire la topologie logique, on a relié tous les nœuds du réseau qui forment une paire comprise dans la matrice de trafic. Par exemple, les sommets 1 et 3 ont été reliés car il existe une demande allant de 1 à 3.

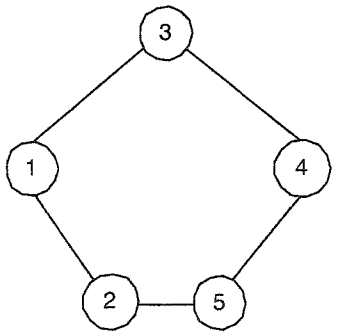


Figure 1.2 Exemple de topologie logique

Une grande partie des définitions présentées prennent en compte le fait que les liens du réseau soient bidirectionnels. Pour les réseaux optiques, les liens bidirectionnels consistent en fait en une paire de liens unidirectionnels. Pour bien comprendre certaines notions qui seront présentées dans la suite du document, il est important de bien définir la notion *d'adjacence de chemins*. En effet, deux chemins sont dits adjacents s'ils ont au moins un arc en commun. La Figure 1.3 illustre la notion d'adjacence des chemins. Trois

trafics sont représentés, et ils passent tous par le lien 3-4. Cependant, seuls les trafics de 1 à 6 et de 2 à 4 (vert et orange) sont adjacents, conformément à la définition que nous avons donnée ci-haut.

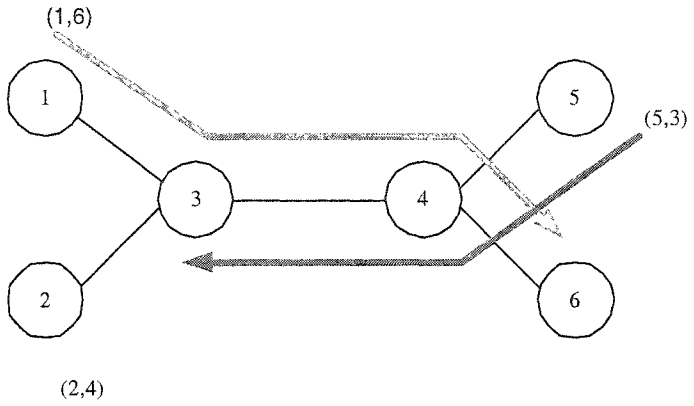


Figure 1.3 Adjacence des chemins

1.2 Éléments de problématique

La mise sur pied de réseaux de communications peut se révéler très complexe, étant donné le nombre de paramètres dont il faut tenir compte. En effet, en plus de l'étape de routage, les planificateurs de réseaux doivent penser à d'autres éléments tels que la conception topologique par exemple. Dans ce mémoire, nous nous intéresserons particulièrement au problème de routage dans les réseaux optiques. Contrairement aux réseaux classiques, le problème de routage optique ne correspond pas simplement à l'assignation de tel ou tel chemin pour les trafics. En effet, avec l'introduction du multiplexage en longueur d'onde dans les réseaux à fibres optiques, en plus du routage de chemins, il faut résoudre le problème d'affectation des longueurs d'onde aux trafics.

Tel que mentionné précédemment, le routage optique est appelé RWA dans la littérature. Le problème RWA a été démontré NP-complet, de ce fait, il est possible de trouver une solution de routage optimale en explorant l'ensemble des solutions possibles. Tandis que pour les réseaux actuels, cette approche est très complexe. En effet, pour des réseaux de grande taille, tels que les réseaux optiques déployés, l'ensemble des solutions possible est très vaste et le définir n'est pas aisé et pourrait prendre des temps de calculs

élevés. C'est ainsi que le problème de RWA est très souvent scindé en deux parties afin d'obtenir une solution satisfaisante [1,3,7,17]. De cette manière, des méthodes sont trouvées d'un côté pour le routage et de l'autre pour l'affectation de longueur d'onde. le routage consiste alors à trouver un chemin dans le réseau pou acheminer un trafic. Tout comme le modèle de trafic, une méthode de routage de chemins peut se faire de manière statique, auquel cas les chemins pour chaque trafic sont trouvés à l'avance en fonction de la matrice de trafics, ou de manière dynamique, auquel cas, la méthode de routage tient compte de l'état du réseau pour trouver les chemins. Quelle que soit la méthode de routage considérée, un aspect important à considérer est la connexité du réseau. En effet, pour un réseau non connexe, il pourrait avoir des demandes impossibles à satisfaire u fait qu'il n'existe aucun chemin reliant la source à la destination. L'affectation des longueurs d'onde aux trafics permet de choisir une longueur d'onde pour chaque trafic tout en évitant que deux trafics dont les chemins ont au moins un lien en commun aient la même longueur d'onde.

Il est possible de placer des convertisseurs de longueurs d'onde dans le réseau de telle sorte qu'en cas de conflit, un trafic puisse arriver à un nœud avec une certaine longueur d'onde et en ressortir avec une autre. Plusieurs stratégies de routage et d'affectation de longueurs d'onde offrant des possibilités de conversion ont été proposées [19,24,25,40,41]. Cependant, le coût associé au convertisseurs est très élevé, c'est pour cette raison que la contrainte de continuité de longueurs d'onde est très souvent appliquée [1,16,30]. Le principal inconvénient relié à l'application de cette contrainte est le taux de blocage du trafic qui est plus élevé que dans le cas où il y a la présence de convertisseurs.

Dans une approche statique du problème de RWA, l'objectif est souvent de maximiser le nombre de trafics acheminés étant donné un nombre de longueurs d'onde disponibles sur chaque fibre ou, de minimiser le nombre de longueurs d'onde nécessaires pour satisfaire un ensemble de demandes de connexions. Une des principales préoccupations des planificateurs de réseaux est la gestion des pannes. Il faudrait développer des méthodes de routage permettant de faire face aux pannes éventuelles en

utilisant les ressources encore fonctionnelles du réseau, tout en satisfaisant un maximum de demandes.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de proposer une solution au problème de RWA statique, qui permettrait de maximiser le nombre de demandes satisfaites. Plus précisément, nous développerons des méthodes pour le routage d'une part et l'affectation de longueur d'onde d'autre part, en tenant compte de la contrainte de continuité de longueur d'onde et du nombre de longueurs d'onde disponibles dans le réseau. Les méthodes proposées seront applicables à des réseaux divers et pour différentes matrices de trafics. Nous allons considérer spécifiquement le routage et l'affectation de longueurs d'onde dans les cas de panne sur un seul lien du réseau. En effet, pour chaque panne de lien, notre objectif sera de satisfaire un maximum de demandes en tenant compte de demandes robustes et permanentes, afin de minimiser le nombre total de demandes bloquées. Plusieurs travaux visant la maximisation du nombre de demandes satisfaites dans le modèle RWA statique ont déjà été effectués [1][15][30], ils nous serviront d'élément de comparaison pour notre solution. Nous mettrons beaucoup plus l'accent sur les résultats fournis par l'algorithme H+ développé par Oulaï [16], parce qu'il considère également le cas où il y a des demandes permanentes et des demandes robustes.

1.4 Esquisse méthodologique

Afin d'atteindre les objectifs mentionnés ci-haut, nous présenterons dans un premier temps un modèle mathématique du problème basé sur les travaux développés par Oulaï [16]. Par la suite nous présenterons plusieurs méthodes de routage et d'affectation de longueur d'onde en utilisant le parallèle entre le problème de RWA et la théorie des graphes, en particulier le coloriage de graphes. Nous partirons du modèle proposé par Oulaï pour arriver à d'autres solutions, ceci en utilisant une ou des méthodes de recherche locale telle que la recherche taboue par exemple, pour l'affectation de longueur d'onde. Pour l'implémentation de notre heuristique, nous utiliserons la même architecture de

réseau qu'Oulaï ainsi que les mêmes conditions expérimentales, tout ceci afin d'obtenir une comparaison des résultats plus fiable et plus adéquate.

1.5 Plan du mémoire

Les travaux présentés dans ce mémoire sont répartis en cinq chapitres. Dans le prochain chapitre, nous ferons une présentation plus détaillée des réseaux optiques puis, nous présenterons quelques algorithmes et méthodes de résolution du problème RWA rencontrés dans la littérature. Dans le troisième chapitre, nous proposerons une formulation mathématique du problème ainsi que des méthodes de résolution sous la forme d'heuristiques. L'implémentation de ces heuristiques et l'analyse des résultats obtenus feront l'objet du quatrième chapitre. C'est également dans ce chapitre que nous ferons une comparaison de nos résultats avec ceux déjà existants. Le dernier chapitre fera une synthèse des travaux effectués, analysera les limites de ces travaux et donnera des indications en vue de recherche future.

CHAPITRE II
ROUTAGE ET AFFECTATION DE LONGUEURS D'ONDE DANS LES RÉSEAUX OPTIQUES

Avec les applications multimédias et les nouveaux services tels que Internet et la vidéoconférence, l'organisation en réseaux a pris beaucoup d'ampleur. L'intérêt accru des consommateurs pour ces différents services et applications a entraîné un immense besoin en bande passante. Face aux limitations des réseaux jusque là utilisés, les réseaux optiques, basés sur les développements de la fibre optique et la technologie WDM, ont pris de l'importance. En effet, le multiplexage de plusieurs longueurs d'ondes sur une même fibre rend possible la transmission de plusieurs térabits par seconde. Afin d'exploiter judicieusement les possibilités offertes par cette nouvelle technologie, plusieurs chercheurs s'attèlent à développer ou tout simplement optimiser divers aspects entrant en jeu dans la planification de réseaux optiques tels que le routage et l'affectation de longueur d'onde [1][7][30][31]. Dans ce chapitre, nous présenterons d'abord les réseaux optiques en mettant l'accent sur la technologie WDM, le routage optique et la gestion des pannes. Par la suite, nous présenterons les principales techniques utilisées pour résoudre le problème de RWA.

2.1 Caractéristiques de base des réseaux optiques

Trois aspects principaux caractérisent les réseaux optiques : le multiplexage en longueur d'onde (WDM), le routage optique et la gestion des pannes. Dans cette section, nous allons passer en revue ces trois aspects.

2.1.1 La technologie WDM

Avec l'arrivée de la fibre optique, la bande passante disponible est énorme mais la technologie de multiplexage temporel jusque là utilisée ne permet pas de l'exploiter de manière satisfaisante. C'est pour cette raison que l'utilisation de la technologie WDM

s'est répandue. Apparu au début des années 1990, le WDM divise la largeur de bande d'une fibre en canaux. Chaque canal peut opérer indépendamment des autres. Cette technologie est née de l'idée de juxtaposer dans la même fibre optique plusieurs trains de signaux numériques mais chacun à une longueur d'onde distincte. La vitesse de modulation des ondes est la même pour tous les signaux. Le nombre de longueurs d'onde pouvant être transporté par une fibre a augmenté au cours de ces dernières années. Aujourd'hui, les systèmes WDM comportent jusqu'à 80 canaux optiques. La Figure 2.1 représente une des nombreuses illustrations de cette technologie.

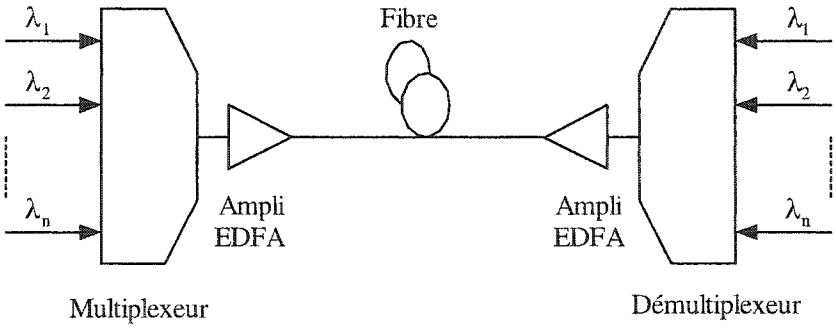


Figure 2.1 Principe d'une liaison WDM

L'amplificateur de fibre dopée à l'erbium (*Erbium Doped Fiber Amplifier : EDFA*) est l'un des éléments clés du WDM; il permet de compenser les pertes d'insertion causées par le multiplexage et le démultiplexage des longueurs d'ondes. Deux autres éléments importants sont les multiplexeurs à insertion/extraction optiques (*Optical Add Drop Multiplexing : OADM*) et les brasseurs optiques (*Optical Cross-Connect : OXC*). L'OADM est un dispositif servant à insérer ou à extraire sélectivement l'un des canaux de longueur d'onde d'un signal WDM circulant sur une fibre sans recourir à une conversion optoélectronique. Ces éléments permettent non seulement de mieux exploiter la capacité des fibres optiques, mais offrent également des possibilités de restauration et de protection élevées.

2.1.2 Le routage optique

Comme nous l'avons mentionné au chapitre précédent, le routage optique consiste en la recherche d'un chemin pour le trafic ainsi qu'à l'affectation de longueur d'onde. Le problème de routage optique est de savoir s'il est possible d'établir un ensemble de demandes de communications, en utilisant un certain nombre de fibres par liens et de longueurs d'onde. En effet, les réseaux optiques peuvent être *multifibres*, auquel cas les liens sont constitués de k fibres optiques. De tels réseaux sont aussi appelés *réseaux k-fibres*. Dans ce mémoire, nous ne considérerons que les réseaux *monofibres*, c'est-à-dire comportant une seule fibre optique par lien.

Le problème de RWA pris séparément est moins complexe. Dans ce cas, une bonne solution devrait fournir un ensemble de chemins pour l'acheminement du trafic et permettre une allocation judicieuse des longueurs d'onde. Il existe deux approches de routages, statique et dynamique. Les travaux recensés dans la littérature traitent autant du routage statique que du routage dynamique. Dans une considération dynamique [12][20], contrairement au cas statique, les ressources du réseau sont utilisées judicieusement, la recherche de chemins s'effectuant en fonction de l'état du réseau. Les méthodes de routage statiques sont cependant moins complexes à développer, ce qui constitue leur attrait principal.

2.1.3 La gestion des pannes

Les réseaux optiques actuels sont très étendus et constituent la base de la communication à travers la terre. La protection des différents éléments du réseau est essentielle afin d'empêcher la perte éventuelle d'informations. En effet, les catastrophes naturelles, la surcharge de réseau ou les erreurs humaines, peuvent affecter plusieurs instances de communication entraînant ainsi la perte de quantités importantes d'informations. Dans la littérature, il existe de nombreux travaux qui traitent du problème de la survivabilité des réseaux [5][9][33][35][36]. La fibre optique ainsi que les autres éléments optiques ne sont pas infaillibles et peuvent à tout moment devenir non fonctionnels. Le besoin de faire face aux éventuels scénarios de panne se fait sentir. Il

existe deux types de pannes : les pannes de liens et les pannes de nœuds. Les pannes de liens sont généralement causées par le bris d'une fibre qui constituent la plus grande source de panne dans les réseaux optiques. La probabilité d'avoir simultanément plusieurs bris de fibres étant très faible, les modèles de solution proposés traitent surtout des cas de pannes simples (*single link failure*). Les pannes au niveau d'un nœud, quant à elles, peuvent être dues à une multitude de raisons. Ce type de panne est beaucoup plus complexe car il met hors d'usage les différents liens qui sont reliés à ce nœud. Il existe deux méthodes pour faire face à une panne du réseau : la méthode proactive et la méthode réactive.

La méthode proactive

Aussi appelée protection, elle consiste à réserver des ressources de sauvegarde pour chaque connexion qui permettront de définir un chemin de secours. Les différentes approches de protection sont [16] : la protection 1+1, 1:1 et 1:*N*. Dans le cas de la protection 1+1, les données sont envoyées simultanément sur le chemin principal et le chemin de secours. En cas de panne, le récepteur se syntonise sur le chemin de secours. Cette situation est illustrée à la Figure 2.2.

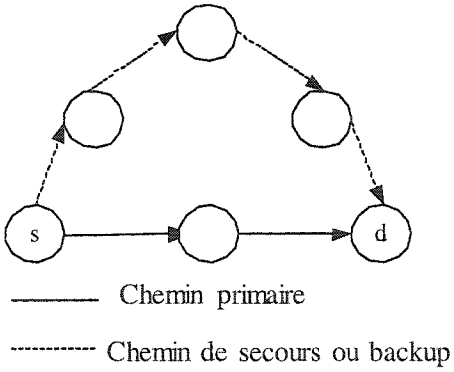
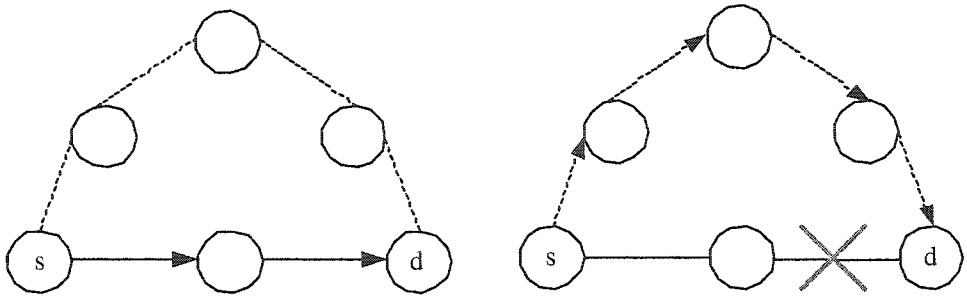


Figure 2.2 Protection 1+1

Pour la protection 1 :1 par contre, un chemin de secours disjoint d'arc est réservé pour chaque chemin principal. Contrairement au cas précédent, les informations ne sont

pas envoyées simultanément sur les deux canaux. On n'émet sur le chemin secondaire que lorsqu'une panne est déclarée sur le chemin principal. La Figure 2.3 présente cette situation. Avec la protection 1 :*N*, *N* chemins principaux partagent le même canal de protection. En cas de défaillance touchant plusieurs chemins principaux, seul le trafic de plus haute priorité est rerouté.



(a) Cas sans panne (b) Cas présentant le bris d'un lien

——— Chemin primaire

---------- Chemin de secours ou backup

Figure 2.3 Protection 1 :*N*

La méthode réactive

C'est la méthode de restauration dynamique. Lorsqu'une panne survient, les chemins secondaires sont déterminés en fonction des ressources disponibles dans le réseau. Il existe la restauration de lien et la restauration de chemin. Dans l'approche de restauration de lien, un autre chemin est déterminé entre les nœuds qui forment les extrémités de la liaison défectueuse, tandis qu'avec la restauration de chemins, un chemin alternatif est trouvé entre l'origine et la destination du trafic affecté par la panne.

Avec l'approche proactive, des ressources sont réservées aux chemins alternatifs. De cette manière, le reroutage du trafic en cas de panne est rapide. Cette méthode est très efficace mais elle entraîne une mauvaise utilisation de la largeur de bande disponible dans le réseau causée par les chemins de secours non utilisés. Ce problème est réglé avec

l'approche réactive qui permet une utilisation plus judicieuse de la largeur de bande. Toutefois, dans ce cas, le temps nécessaire pour contourner la panne peut être très long.

2.2 Le coloriage de graphes

Le problème d'affectation de longueurs d'ondes est très souvent résolu en utilisant les techniques de coloriage des sommets d'un graphe. Dans la littérature, nous retrouvons plusieurs travaux qui utilisent ces techniques [1][16][39]. Dans cette section, nous allons faire une présentation générale du problème de coloriage de graphes. Après quelques définitions de base, nous présenterons les principales techniques rencontrées dans la littérature pour ce problème. Puis, nous ferons le parallèle entre le coloriage de graphes et l'affectation de longueurs d'onde.

2.2.1 Définitions

Le coloriage des sommets d'un graphe G est l'affectation d'une couleur à chacun d'entre eux de telle sorte que deux sommets adjacents n'aient pas la même couleur. Soit un graphe non orienté $G = (V,E)$ comportant un ensemble de sommets V et un ensemble d'arêtes E. Un sous-ensemble I de V est dit *stable* ou *indépendant* s'il ne comprend aucune paire de sommets adjacents : $\forall i \in I,\ \forall j \in I,\ (i, j) \notin E$. Étant donné un entier k, le problème de k-coloriage consiste à colorier un graphe G en k couleurs. Un graphe est dit *k-chromatique* s'il admet un coloriage de ses sommets en k couleurs. On appelle *nombre chromatique* le nombre optimal de couleurs, c'est-à-dire le nombre minimal de couleurs distinctes pour effectuer un coloriage de sommets : il est noté $\chi(G)$. Le problème de coloriage de graphes est NP-difficile [15]. C'est pour cette raison que des heuristiques ont été développées pour obtenir de bonnes solutions pas forcément optimales, mais dans un délai de calcul raisonnable. Les heuristiques dites gloutonnes ont été les premières à être utilisées, puis ont suivi les méthodes de recherche locale.

2.2.2 Les heuristiques gloutonnes

Ce sont les méthodes de résolution qui ne permettent pas de retour arrière. Les sommets sont coloriés un à un de manière statique ou dynamique. Dans le cas statique, les sommets sont coloriés après avoir été rangés dans un certain ordre, tandis que dans le cas dynamique, le nouveau sommet à colorier est choisi lorsque les sommets précédents ont été coloriés [15]. Parmi les heuristiques gloutonnes, mentionnons l'heuristique séquentielle et l'heuristique de saturation.

- **L'heuristique séquentielle**

C'est le cas le plus simple des heuristiques gloutonnes. Les sommets du graphe sont classés selon un ordre donné. À chaque itération, le sommet défini par cet ordre est colorié avec la plus petite couleur non encore présente parmi ses voisins déjà coloriés. En général, un tel coloriage n'est pas optimale.

- **L'heuristique de saturation**

C'est une heuristique dynamique basée sur la saturation des sommets. La saturation d'un sommet non colorié correspond au nombre de couleurs utilisées par ses voisins déjà coloriés. Le nouveau sommet à traiter à chaque fois est celui qui a la saturation maximale, et il se voit attribué la plus petite couleur autorisée. En cas d'égalité entre deux sommets, le sommet de plus grand degré sera choisi.

2.2.3 Les méthodes de recherche locale

La recherche locale est une classe de métaheuristiques basée sur la notion de voisinage [15/109,25,121]. Les méthodes de recherche locale ont été utilisées entre autres pour résoudre le problème du voyageur de commerce. Le principe de ce type de méthode est simple. Étant donné un problème d'optimisation (S,f) défini par l'ensemble S des configurations et la fonction coût f, on définit une fonction de voisinage et un mécanisme de parcours du voisinage. Pour commencer, un algorithme de recherche locale engendre tout d'abord une configuration initiale s_0, puis il effectue une suite d'itérations. À chaque

itération, une configuration *s'* est choisie dans le voisinage de la configuration courante, en fonction de son coût. Le processus s'arrête et retourne la meilleure configuration trouvée quand le critère d'arrêt est atteint.

Les algorithmes basés sur la recherche locale sont en général des métaheuristiques, les principales étant la recherche taboue et le recuit simulé. Plusieurs stratégies, telles que l'approche de pénalisation, l'approche de réparation et l'approche constructive [15], ont été définies afin de résoudre les problèmes de coloriage et de *k*-coloriage.

- **L'approche de réparation**

Elle permet de traiter le problème de *k*-coloriage. Dans ce contexte, il est très important de bien définir l'espace de recherche, la configuration et la fonction d'évaluation ou fonction de coût. L'espace de recherche est l'ensemble des *k*-coloriages impropres, c'est-à-dire des k-coloriages incluant des sous-ensembles instables. La fonction de coût *f* correspond au nombre d'arêtes violées dans une solution *S*. Dans ces conditions, le problème d'optimisation est alors le suivant : Étant donné $S, F : S \rightarrow \Re$, on veut trouver $s*; f(s*) = \min_{s \in S} f(s)$. Le voisinage d'une solution est défini par la fonction *N* telle que : $N : S \underset{s \mapsto N(s) \subseteq S}{\rightarrow} 2^S$, avec 2^S qui est l'ensemble des parties de S. Pour passer de *S* à *S'*, *S'* appartenant à *N(S)*, il faut faire un mouvement qui consiste à changer la couleur d'un sommet. Avec cette approche, la configuration de départ est un coloriage complète impropre dans laquelle les couleurs ont été attribuées aléatoirement.

Dans l'exemple qui suit, nous présentons une illustration de cette méthode. Le graphe considéré comporte cinq sommets et nous disposons de deux couleurs pour le colorier, les couleurs rouge et bleu. Pour commencer les couleurs sont attribuées aléatoirement, tel que présenté à la Figure 2.4. Avec cette configuration, nous avons trois arêtes violées, soit 4, 5 et 7, la fonction coût *f* vaut donc 3.

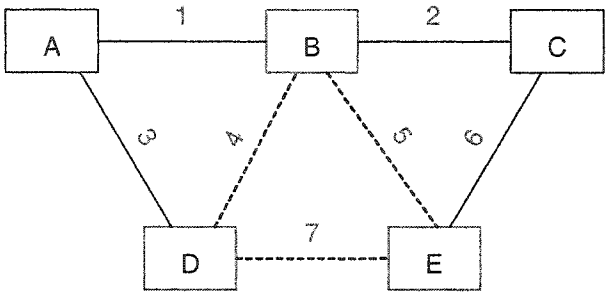


Figure 2.4 Illustration de l'approche de réparation : initialisation aléatoire

À partir de cet état, la couleur du sommet D est modifiée, ce qui permet d'obtenir la configuration présentée à la Figure 2.5. La fonction coût vaut maintenant 2.

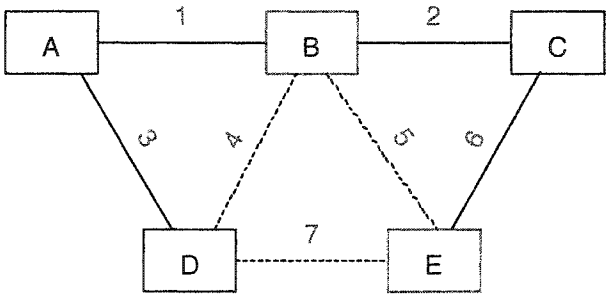


Figure 2.5 Illustration de l'approche de réparation : état après un mouvement

L'approche de réparation peut être utilisée avec des métaheuristiques de recherche locale telle que le recuit simulé ou la recherche taboue [15].

- **L'approche constructive**

Elle permet également de résoudre le problème de *k*-coloriage [15]. À partir d'un coloriage partiel vide, les sommets sont coloriés successivement pour obtenir éventuellement un coloriage complet si possible optimale. Contrairement au cas précédent, nous n'avons pas le droit de violer des sommets; l'espace des solutions est donc l'ensemble des coloriages partielles propres. Un mouvement se fait en coloriant un sommet x non colorié avec une couleur v et en décolorant tous les sommets de x colorié avec v. Le problème avec cette approche est le risque élevé de tourner en rond. Plusieurs

stratégies, comme par exemple le bruitage et l'utilisation de listes taboues, peuvent être adoptées pour faire face à cette situation.

Dans la suite de ce mémoire, c'est cette approche qui sera utilisée en vue de proposer une heuristique pour le problème de RWA, plus particulièrement pour l'affectation des longueurs d'onde.

2.2.4 Le coloriage de graphes et l'affectation de longueurs d'ondes

Dans la littérature, les méthodes de coloriage de graphes sont très souvent utilisées pour l'affectation de longueurs d'onde [1][3][16]. Les chemins optiques correspondent aux sommets du graphe à colorier. Deux sommets sont reliés si et seulement si les chemins qu'ils représentent ont au moins un lien en commun. De plus, une couleur représente une longueur d'onde. La plupart des travaux présentés jusqu'à présent sont basés sur des heuristiques séquentielles. C'est le cas de l'algorithme H+ qui sera présenté en détail dans la prochaine section.

2.3 L'algorithme H+

Oulaï [16] propose un algorithme basé sur le routage de plus court chemin et une méthode de coloriage séquentielle pour l'affectation de longueurs d'onde. Le modèle qu'il présente est appelé RWAP pour *Routing and Wavelength Assignment Problem*. Son objectif est de minimiser la quantité de trafic bloqué sur l'ensemble des scénarios de pannes simples de liens possibles. Un scénario de panne simple est associé au graphe du réseau sans le lien en panne. Donc, pour un scénario donné, il faut essayer d'acheminer un maximum de trafics en tenant compte du lien en panne. Le scénario de base ou scénario sans panne correspond au réseau avec tous les liens fonctionnels. Il se place dans le cas statique et considère des demandes permanentes et robustes. Il propose un modèle de programmation mixte basé sur les hypothèses suivantes : la continuité de longueurs d'onde, le routage selon une méthode de plus court chemin, un réseau bi-connexe et des liens bidirectionnels. Voici les ensembles et les principales variables qu'il a utilisés :

- N, l'ensemble des nœuds du réseau ;
- M, l'ensemble des liens unidirectionnels du réseau ;
- K, l'ensemble des chemins tels que $K=K^P U K^R$ où K^P représente l'ensemble des chemins permanents et K^R, l'ensemble des chemins robustes ;
- $O(k)$ et $D(k)$, l'origine et la destination du chemin $k \in K$;
- Ω, l'ensemble des longueurs d'onde disponibles sur tout le réseau ;
- Ω_{ij}, l'ensemble des longueurs d'onde disponibles sur le lien (i,j);
- T, l'ensemble des états du réseau considéré. Le scénario sans faute est noté t_0.;

Dans un premier temps, il propose un algorithme appelé H, qui permet d'affecter les chemins et les longueurs d'onde pour tous les scénarios possibles. Par la suite, il propose un critère pour améliorer l'algorithme H et obtient l'algorithme H+. Le déroulement de l'algorithme H est présenté à la Figure 2.6. L'algorithme commence par traiter le scénario de base. Pour ce faire, les plus courts chemins sont calculés pour chaque trafic en fonction d'une métrique de routage qu peut être soit le coût des liens ou le nombre de sauts. Par la suite un chemin est choisi arbitrairement pour chaque trafic et l'algorithme procède à l'affectation des longueurs. Les longueurs d'onde sont affectées en utilisant une heuristique de coloriage de graphes séquentielle. Le graphe à colorier est construit en fonction des trafics à router. Chaque sommet du graphe représente un chemin associé à un trafic. Deux chemins partageant au moins un lien sont représentés par deux sommets adjacents. Pour chaque scénario de panne, tous les trafics bloqués dans le scénario de base sont automatiquement bloqués. Il en est de même pour les trafics permanents touchés par la panne correspondante. Les trafics permanents non touchés par la panne dans un scénario seront routés sur le même chemin que dans le scénario de base. En ce qui concerne les trafics robustes, s'ils sont touchés par la panne, l'algorithme cherche un autre plus court chemin pour les reroutés. S'il n'y en a pas d'autres, ils sont bloqués dans ce scénario.

POUR le scénario sans panne
Trouver les plus courts chemins pour chaque paire $i, j \in N$, en considérant la métrique de routage
Router les trafics selon un plus court chemin
Affecter les longueurs d'onde notées $\lambda(k)$, aux trafics en tenant compte de la continuité de longueur d'onde
SI une longueur d'onde est trouvée
Acheminer le trafic en utilisant cette longueur d'onde
SINON
SI il existe un autre plus court chemin pour le trafic,
Essayer ce chemin
SINON, pour tout $t \in T$, le trafic est bloqué

POUR tous les scénarios de panne
POUR tous les trafics acheminés dans le scénario sans panne
POUR chaque trafic $k \in K^P$
SI le chemin k utilise un élément défectueux, il est bloqué
SINON il est routé sur le même chemin que dans le scénario sans panne

POUR chaque trafic $k \in K^R$
SI k n'est pas affecté par la panne
Router sur le même chemin que dans le scénario sans panne
Attribuer la même longueur
SINON SI k est affecté par la panne
Trouver un autre plus court chemin de l'origine à la destination

POUR tous les trafics reroutés
Affecter une longueur d'onde en tenant compte des longueurs d'onde déjà affectées.
SI une longueur d'onde est trouvée
L'Attribuer au trafic
SINON
SI il existe un autre plus court chemin k
Essayer ce chemin
SINON
Bloquer le trafic

Figure 2.6 Algorithme H

À la fin de l'algorithme la fonction objectif est évaluée. Pour améliorer cette proposition, Oulai définit un indice qui permet de déterminer si une connexion doit être désactivée ou non. Cet indice est un estimateur du nombre de longueurs d'onde nécessaires à l'établissement d'une connexion k, il est noté n_k. Il part du principe que, l'affectation d'une connexion dont l'établissement ainsi que l'établissement de toutes les connexions adjacentes nécessite un grand nombre de longueurs d'onde, est pénalisante. Pour trouver n_k, il considère le trafic k et tous les trafics avec lesquels k partagent au moins un lien. De cette manière, il définit : $n_k = \sum d$ avec $d(o,d)$ la distance en nombre de sauts optiques entre une origine o et une destination d. L'algorithme H+ obtenu est présenté à la Figure 2.7.

Faire une affectation initiale avec l'algorithme H et calculer b
Désactiver $b/2$ connexions en désactivant à chaque fois la connexion qui a la plus grande valeur de n_k.

FAIRE
    Désactiver un nombre de connexions égal à un pas donné
    Affecter les longueurs d'onde
TANT QUE b connexions ne sont pas désactivées

Retenir le nombre n de connexions supprimées au départ qui donne la meilleure solution
Définir un espace de recherche centré autour de n
POUR tout l'espace de recherche défini
    Désactiver une connexion à la fois
    Évaluer la solution avec l'algorithme d'affectation de longueurs d'onde
FIN POUR

Retenir la meilleure solution.
Réactiver tous les trafics supprimés au départ
Évaluer la solution avec l'algorithme d'affectation de longueurs d'onde

Figure 2.7 Algorithme H+

2.4 Le problème max-RWA

En général, dans une approche statique du problème RWA, le but est de maximiser le nombre de demandes pouvant être établies, étant donné une matrice de connexions et le nombre de longueurs d'ondes disponibles sur chaque fibre. C'est ce qu'on appelle le problème de max-RWA. Il a été traité entre autres par Krishnaswamy et Sivarajan [30] . Ils proposent des algorithmes basés sur des solutions de relaxation de programmes linéaires. Contrairement aux programmes linéaires en nombres entiers, ILPs, qui avaient jusque là été présenté, dans lesquelles les chemins qu'une paire source-destination sont autorisés à emprunter sont définis avant la résolution de l'ILP, ils proposent une formulation qui permet de choisir n'importe quel chemin possible et n'importe quelle longueur d'onde au cours de la résolution de l'ILP. Cette approche est appelée *link based ILP formulation*, c'est-à-dire que les contraintes sont placées sur les liens du réseau. Ils utilisent principalement des contraintes de continuité de longueur d'onde. Dans le cadre de ce travail, deux ILPs sont proposés, nous allons détailler le premier.

Soit $\rho(i,j)$ le nombre de connections devant être établies entre le nœud i et le nœud j. Une route et une longueur d'onde doivent être assignées à chaque connexion de telle sorte que deux chemins optiques traversant le même lien n'aient pas la même longueur d'onde. Les paramètres dont on dispose sont :

- N, le nombre de nœuds dans le réseau ;
- ρ, la matrice de trafics ;
- F, le nombre de longueurs d'onde sur une fibre ;
- P_{lm} qui dénote l'existence d'un lien physique entre le nœuds l et m vaut 1 s'il y a un lien entre l et m et vaut 0 sinon.

Les notations suivantes sont utilisées :

- i et j représentent les nœuds source et destination d'un chemin optique ;
- l et m dénotent les extrémités d'un lien physique ;
- k, utilisé en exposant, correspond au numéro de la longueur d'onde, $k \in \{0,1,..., F-1\}$;

- q, utilisé en indice ou exposant, correspond au $q^{ème}$ chemin optique. Pour *(i,j)* donné, $q \in \{1,2,..., \rho(i,j)\}$.

Pour présenter le modèle proposé, les variables suivantes sont également définies :

- $b_q(i,j) = 1$, s'il existe un $q^{ème}$ chemin optique entre i et j, sinon $b_q(i,j) = 0$;
- $C^{(k,q)}(i,j) = 1$ si le $q^{ème}$ chemin optique entre i et j utilise la longueur d'onde k, $C^{(k,q)}(i,j) = 0$ sinon;
- $C_{l,m}^{(k,q)}(i,j) = 1$ si le $q^{ème}$ chemin optique entre i et j utilise la longueur d'onde k et est routé sur le lien physique *(l,m)*, sinon $C_{l,m}^{(k,q)}(i,j) = 0$.

L'objectif est alors de :

$$\max \sum_{i,j} \sum_{q} b_q(i,j) \tag{2.1}$$

sujet à

$$\sum_{q} b_q(i,j) \leq \rho(i,j), \ avec\, b_q(i,j) \in \{0,1\} \tag{2.2}$$

$$\sum_{k=0}^{F-1} C^{(k,q)}(i,j) = b_q(i,j) \ , \text{pour tout } q \text{ et } (i,j) \tag{2.3}$$

$$C_{l,m}^{(k,q)}(i,j) \leq C^{(k,q)}(i,j) \text{, pour tout } q, (i,j), (l,m) \text{ et } k \tag{2.4}$$

$$\sum_{q} \sum_{(i,j)} C_{l,m}^{(k,q)}(i,j) \leq 1 \text{, pour tout } k \text{ et } (l,m) \tag{2.5}$$

$$\sum_{l} C_{l,m}^{(k,q)}(i,j) P_{l,m} - \sum_{l} C_{l,m}^{(k,q)}(i,j) P_{m,l}$$

$$= \begin{cases} C^{(k,q)}(i,j), & \text{si } m = i \\ -C^{(k,q)}(i,j), & \text{si } m = j; \text{ pour tout } (i,j), k, q \text{ et } m \\ 0, & \text{si } m \neq i \text{ et } m \neq j \end{cases} \tag{2.6}$$

La relation (2.1) correspond à la fonction objectif qui est de maximiser le nombre de connexions. Les relations (2.2) à (2.6) ont les contraintes dont il faut tenir compte lors de la résolution. La relation 2.2 signifie que, pour une paire origine destination *(i,j)*, le nombre de connexions établies est au plus égal à $\rho(i,j)$. Les contraintes qui suivent sont

les contraints de continuité de longueur d'onde. La première, (2.3), assure que si un chemin optique $b_q(i, j)$ existe, alors il y a une longueur d'onde k qui lui est attribuée. La relation (2.4) signifie que la variable $C_{l,m}^{(k,q)}(i, j)$ ne peut être non nulle que si $C^{(k,q)}(i, j)$ est non nulle. La contrainte qui veut que deux chemins optiques traversant le même lien n'ait pas la même longueur d'onde est exprimée par la relation (2.5). Enfin, la relation (2.6) spécifie que un chemin optique aura la même longueur d'onde sur tous les liens qui forment sa route.

Les deux ILPs développés servent de point de départ pour deux algorithmes de résolution du problème max-RWA. Le premier algorithme arrondi les variables obtenues à la fin de la résolution du premier ILP et s'arrête lorsqu'il ne peut plu faire d'arrondi sans violer les contraintes de continuité de longueurs d'onde. Le deuxième algorithme quant à lui se déroule en deux étapes. Dans la première, les valeurs obtenues par la résolution de l'un ou l'autre des ILPs sont arrondies. De cette manière, on obtient les routes pour toutes les connexions présentes dans la matrice de trafics. À partir de ces routes, un graphe des chemins appelé *path-graph* est créé. La deuxième étape de l'algorithme utilise une heuristique de coloriage des sommets d'un graphe pour trouver un ensemble maximal de sommets pouvant être coloriés avec F couleurs. F étant le nombre de longueurs d'onde par fibre.

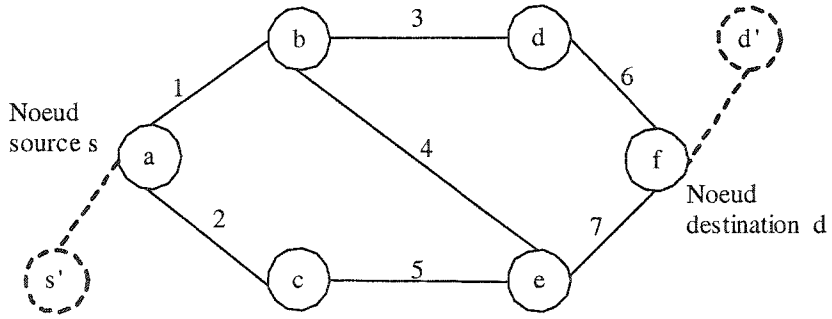
2.5 L'algorithme WCA

Zhang et al. [7] proposent un algorithme qui détermine les chemins optiques de manière dynamique en utilisant des convertisseurs de longueurs d'onde, afin de minimiser le taux de blocage dans le réseau. Ils l'appellent l'algorithme WCA (*Wavelength and Converter-Aware*). Cet algorithme divise le problème RWA en deux sous-problèmes. Dans un premier temps, le problème de routage est résolu, puis celui de l'affectation de longueurs d'onde. Avec cette approche, ils prennent en compte le nombre de longueurs d'onde disponible à chaque nœud ainsi que le nombre de convertisseurs et leur coût. Pour développer une solution, le réseau est modélisé par un graphe orienté

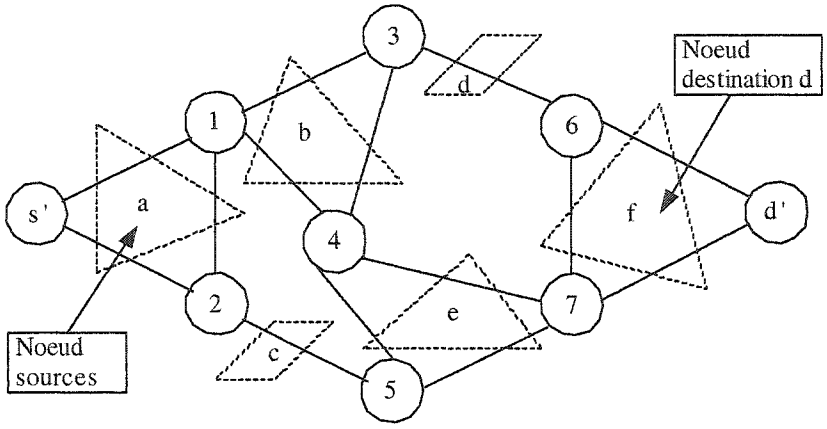
$G = (N, L)$, où N et L représentent respectivement l'ensemble des nœuds et des liens. La largeur de bande de chaque lien de fibre optique est divisée en W longueurs d'onde et la conversion de longueur d'onde est totale. Le routage et l'affectation de longueurs d'onde sont alors résolus en utilisant un algorithme de routage de plus court chemin. Le graphe de départ est modifié dans les deux cas. La solution proposée est la réunion de deux algorithmes, l'algorithme de routage et l'algorithme d'affectation de longueurs d'onde.

- **L'algorithme de routage**

Pour trouver le meilleur chemin entre deux nœuds *s* et *d*, un graphe auxiliaire est d'abord créé à partir du réseau original, tel qu'illustré à la Figure 2.8. Pour ce faire, deux pseudo-nœuds sont ajoutés au réseau original. Les sommets du graphe auxiliaire représentent les liens du graphe de départ; les pseudo-nœuds sont également représentés par des sommets. Deux sommets sont alors reliés par une arête si, dans le réseau de départ, les liens qu'ils représentent sont reliés par un nœud.



(a) réseau original avec les pseudo-nœuds



(b) graphe auxiliaire

Figure 2.6 Illustration de la transformation de graphe pour l'algorithme WCA

Après avoir obtenu le graphe auxiliaire, la métrique de routage (le coût sur les arêtes) est obtenue en tenant compte du nombre de chemins passant par cette arête. Puis, l'algorithme de Dijkstra est utilisé pour déterminer le plus court chemin entre les pseudos origine et destination. Cependant, il est interdit que le chemin trouvé passe successivement par deux arêtes correspondant au même nœud dans le réseau original.

- **L'algorithme d'affectation de longueur d'onde**

Zhang et al. [7] proposent trois processus pour l'affectation de longueur d'onde :

- L'algorithme FFW (First Fit Wavelength First) qui est une extension du First-Fit utilisé par Mokhtar et Azizoglu [12] pour résoudre le problème d'affectation de longueurs d'onde. Le nœud source doit trouver la première longueur d'onde de libre sur le chemin déterminé par l'algorithme de routage précédent. Une longueur d'onde libre est cherchée sur un lien selon un ordre prédéfini et, si possible, une conversion est effectuée en cas de besoin. Si une longueur d'onde est trouvée, le processus se poursuit au nœud suivant. Sinon, le nœud source choisit une autre longueur et le processus se répète.

- L'algorithme LEC (Least Converter First) : dans ce cas, l'affectation de longueur d'onde est traitée comme un problème de routage et utilise l'algorithme de Dijkstra pour trouver une solution, le but de cette méthode étant de déterminer un chemin optique utilisant le plus petit nombre de convertisseurs.

- L'algorithme LCC (Least Conversion Cost First) : cet algorithme est semblable au LEC. La différence se situe au niveau du calcul de la fonction de coût pour la recherche du plus court chemin.

Les résultats obtenus grâce à l'algorithme WCA sont comparés, entre autres, à ceux des algorithmes HW (Hop-based) et TAW (Total Wavelength and Available Wavelength) présentés par Bhide et al. [7]. Les simulations sur un modèle de réseau NSFNET montrent que le WCA est plus performant que ces algorithmes.

2.6 L'Approche quasi-statique

Dans un article relatif au routage et à l'affectation de longueurs d'onde, Ozdaglar et Bertzekas [31] proposent plusieurs formulations pour l'optimisation du problème RWA qui, offrent de sérieuses améliorations comparativement aux méthodes déjà existantes. Ils adoptent une vue quasi-statique du problème; quasi-statique car, pour un réseau optique donné, ils considèrent tout d'abord un ensemble fixe de demandes de connexions. Par la suite, des demandes additionnelles arrivent aléatoirement dans le réseau, une à la fois. Le but est alors d'affecter une route et une longueur d'onde aux nouvelles demandes sans toutefois rerouter celles déjà existantes. Partant du principe qu'il existe un coût associé à toute demande bloquée, l'objectif est de minimiser la somme des coûts de blocage.

Cette approche est une formulation optimale pour des problèmes impliquant plusieurs types de flots. Généralement, ce genre de problème s'exprime de la manière suivante : $\min \sum_{l \in L} D_l(f_l)$, sujet à des contraintes de conservation de flot et toute autre contrainte supplémentaire possible. f_l est le flot total sur un lien l de L, et D_l est la

fonction de coût. Pour les réseaux optiques, le flot sur un lien est le nombre de chemins optiques qui passent par ce lien. Les ILPs proposées correspondent à des réseaux avec différentes capacités de conversion. En effet, elles peuvent être utilisées pour des réseaux sans convertisseurs de longueurs d'onde et peuvent être étendues à des réseaux avec présence de convertisseurs. Ces ILPs tendent à fournir une solution optimale entière, même si les contraintes d'intégralité ont été relaxées. Ainsi, le problème peut être résolu de manière optimale par des méthodes de programmation linéaire. La preuve de l'optimalité des solution proposées est faite pour des topologies spécifiques, comme les réseaux en anneau par exemple.

2.7 Autres méthodes de résolution

Le problème de maximisation des chemins disjoints d'arc est adapté au problème de RWA par Manohar et al. [3] Partant du principe que deux chemins optiques de même longueur d'onde ne traversent pas un même lien, ils sont disjoints d'arcs. En théorie des graphes, il est possible de trouver l'ensemble maximal des chemins disjoints d'arcs. Soit G=*(V,E)* le graphe correspondant à une topologie de réseau. Soit (s_i,t_i) l'ensemble des paires origine/destination, et τ l'ensemble des connexions qui doivent être disjointes d'arcs. τ est dit réalisable dans G s'il existe des chemins P_i mutuellement disjoints d'arcs pour chaque paire origine-destination. Le problème de chemins disjoints d'arcs maximum revient à trouver le sous-ensemble de τ réalisable de taille maximale. Ce problème est bien connu comme étant NP-difficile. Afin de résoudre le problème de RWA, ils définissent une borne supérieure d pour le nombre de liens dans un chemin qui seront considérés par l'algorithme comme faisant partie de la solution. Une valeur possible de d est donnée par la relation suivante : d = max (diamètre(G),$|E|^{1/2}$). Les données de l'algorithme sont le graphe représentant la topologie du réseau, l'ensemble τ et la borne d. L'algorithme se déroule comme suit :

- Un chemin optique est sélectionné aléatoirement dans l'ensemble τ ;
- Le plus court chemin est trouvé pour établir la connexion ;

- Si la longueur du plus court chemin ne dépasse pas *d*, la connexion est routée et tous les liens utilisés par le chemin sont effacés. Si la longueur dépasse *d*, la connexion n'est pas routée ;
- Ces étapes sont répétées jusqu'à ce que tous les chemins optiques aient été traités.

À la fin, les chemins optiques qui auront la même longueur d'onde sont obtenus.

Modiano et Narula-Tam [5] considèrent le problème du routage des chemins optiques sur la topologie du réseau, tout en gardant la topologie logique connectée en cas de panne simple de lien. Ils partent du principe que, si la route principale et la route de protection d'un chemin optique passent par le même lien physique, il sera bloqué en cas de bris de ce lien. Afin de développer une solution, ils formulent le problème sous forme de ILP (*Integer Linear Program*) dont le but est de minimiser le nombre total de liens et de longueurs d'onde utilisées. Les résultats montrent que cette formulation offre un plus grand degré de protection en comparaison de l'utilisation du routage de plus court chemin pour les chemins optiques. Cet aspect du problème a également été envisagé par Shifeng et al. [33]. Ils introduisent le problème différemment, mais l'objectif reste le même : router les chemins optiques de telle sorte que le réseau reste connecté en cas de panne simple.

2.8 Algorithmes d'affectation de longueurs d'onde

Dans [20] et [12] Mokhtar et Azigoglu développent ce qu'ils appellent le routage adaptatif sans contraintes (Adaptative Unconstrained Routing, AUR). Un modèle de routage sans contraintes considère tous les chemins entre la source et la destination pour la décision de routage. Dans ce modèle, ils proposent cinq algorithmes adaptatifs pour l'affectation de longueurs d'onde :

- PACK : cet algorithme essaie d'acheminer le trafic en premier sur la longueur d'onde la plus utilisée. Pour ce faire, les longueurs d'onde sont rangées par ordre décroissant d'utilisation. L'utilisation d'une longueur d'onde est définie par le

nombre total de liens sur lesquels elle est utilisée. Avec cette approche, l'utilisation des longueurs d'onde disponibles est maximisée.

- SPREAD : dans ce cas, le trafic est acheminé en premier sur la longueur d'onde la moins utilisée afin de répartir la charge sur l'ensemble des longueurs d'onde de manière quasi uniforme. Les longueurs d'onde sont classées par ordre croissant d'utilisation.
- RANDOM : la recherche de longueur d'onde est faite dans un ordre aléatoire. Toutes les permutations de l'ensemble des longueurs d'onde sont équiprobables.
- EXHAUSTIVE : toutes les longueurs d'onde sont traitées pour la recherche d'un plus court chemin et le chemin le plus court parmi ceux ainsi trouvés sera retenu.
- FIXED : l'ordre de recherche est fixé dès le départ et l'algorithme route le trafic sur la première longueur d'onde disponible.

CHAPITRE III
MÉTHODES DE ROUTAGE PROPOSÉES

Dans ce chapitre, nous présentons les méthodes de résolution que nous proposons pour le problème RWAP. Le problème RWAP est un problème de routage et d'affectation de longueur d'ondes présentant les caractéristiques suivantes : Un modèle de trafic statique comportant des demandes robustes et permanentes, l'absence de convertisseurs de longueurs d'onde et par conséquent l'application de la contrainte de continuité de longueurs d'onde, un ensemble de scénarios pour lesquels il faut maximiser le nombre de trafics acheminés, la bi-connexité du réseau qui permet d'assurer qu'en cas de panne, il existe au moins un autre chemin permettant de router les trafics affectés et des liens bidirectionnels. Il faut noter que le problème max_RWA qui a été classé NP-difficile [1][30], est un cas particulier du problème RWAP. Les méthodes que nous proposons sont basées sur l'utilisation d'heuristiques qui permettent d'obtenir des solutions réalisables en des temps de calcul raisonnables.

Dans un premier temps, nous allons définir le problème plus précisément. Par la suite, nous présenterons le schéma de résolution que nous proposons puis, nous ferons l'analyse de complexité de notre heuristique.

3.1 Définition du problème

Un exemplaire du problème RWAP est défini par :

- Le réseau, composé de l'ensemble des stations (les nœuds) et des liens de fibre optique. Il est représenté par un graphe noté G_R tel que $G_R = (V_R, E_R)$. Ce graphe comprend :
  - V_R, l'ensemble des sommets du graphe G_R représentant les nœuds du réseau ;
  - E_R, l'ensemble des arêtes du graphe G_R représentant les liens du réseau.

- D, l'ensemble des demandes ou trafics. Un trafic d est représenté par une demande de communication entre une paire de stations ou nœuds du réseau. L'origine est notée $o(d)$ et la destination $d(d)$. $D=D^P \cup D^R$, où D^P représente l'ensemble des demandes permanentes et D^R celui des demandes robustes. Les demandes robustes sont celles qui peuvent être reroutées en cas de panne sur le chemin sur lequel elles sont acheminées. Les demandes permanentes ne bénéficiant pas de cette opportunité sont bloquées en cas de panne ;
- T, l'ensemble des scénarios ou états possibles du réseau. À un scénario t de T est associée l'ensemble $E_t \subseteq E_R$. E_t est l'ensemble des arêtes valides dans le réseau ;
- k, le nombre de longueurs d'onde disponibles dans le réseau.

Pour chaque scénario probable, en considérant les données présentées ci-haut, nous cherchons à affecter un chemin et une longueur d'onde à un maximum de trafic et par conséquent à minimiser le nombre de demandes bloquées. Nous ne considérerons que les scénarios associés à des pannes de liens simples. Une solution au problème consiste, pour un scénario donné, à trouver un chemin et une longueur d'onde pour un ensemble de trafics. Le triplet $\left(D_t^{'}, \mu_t, \lambda_t\right)_{t \in T}$ représente une solution possible au problème RWAP, où :

- $D'_t \subseteq D$, représente l'ensemble des trafics acheminés dans le scénario t;
- $\begin{array}{l} \mu_t : D_t^{'} \rightarrow 2^{E_t} \\ \quad d \mapsto \mu_t(d) \end{array}$, $\mu_t(d)$ représente le chemin emprunté par le trafic d dans le scénario t. $\mu_t(d)$ est inclus dans l'ensemble des chemins possibles entre l'origine et la destination associés au trafic d et pour E_t donné. Cet ensemble est appelé PATH(E_t , $o(d)$, $d(d)$). De plus, on a $\mu_t(d) \subseteq E_t$;
- $\begin{array}{l} \lambda_t : D_t^{'} \rightarrow \{1,...,k\} \\ \quad d \mapsto \lambda_t(d) \end{array}$, $\lambda_t(d)$ représente la longueur d'onde affectée au trafic d dans le scénario t.

Le triplet $\left(D_t^{'}, \mu_t, \lambda_t\right)_{t \in T}$ doit vérifier les contraintes suivantes :

$$\forall d_1, d_2 \in D,\ \forall t, t_1, t_2 \in T,\ \forall (i,j) \in \mu_t(d),$$

$$\lambda_t(d_1) = \lambda_t(d_2) \Rightarrow \mu_t(d_1) \cap \mu_t(d_2) = \varnothing \qquad (3.1)$$

$$d \in D'_{t_1} \cap D'_{t_2} \cap D^P \Rightarrow \mu_{t_1}(d) = \mu_{t_2}(d) \qquad (3.2)$$

$$\forall t \in T, D_t \subseteq D_{t_0} \qquad (3.3)$$

La relation (3.1) implique que deux trafics partageant au moins un arc ne peuvent avoir la même longueur d'onde. Cette contrainte est essentielle avec l'utilisation de la technologie WDM dans les réseaux optiques actuels. La relation (3.2), quant à elle, découle de la définition d'un chemin permanent, qui ne peut être rerouté en cas de panne. Elle précise que si une demande permanente est satisfaite dans deux scénarios quelconques, elle sera routée sur le même chemin dans les deux cas. Par contre, cette contrainte ne concerne pas les longueurs d'onde. En effet, une demande permanente peut se voir attribuer deux longueurs différentes dans deux scénarios distincts. La relation (3.3) spécifie qu'une demande non satisfaite dans le scénario de base t_0, ne sera satisfaite dans aucun autre scénario.

Un routage optimal est un routage qui maximise le nombre de demandes satisfaites sur l'ensemble des scénarios en respectant les contraintes présentées ci-haut. L'objectif est donc le suivant :

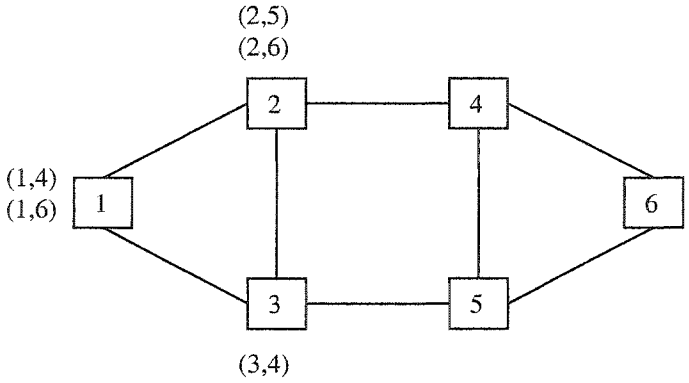
$$\max_{t \in T} \sum_{t \in T} \left| D'_t \right| \qquad (3.4)$$

Le but du problème est de trouver un routage optimal. La relation (3.4) exprime la minimisation du taux de blocage qui est notre objectif principal. En effet, en maximisant le nombre de trafics acheminés, nous diminuons le nombre de trafics bloqués, donc le taux de blocage.

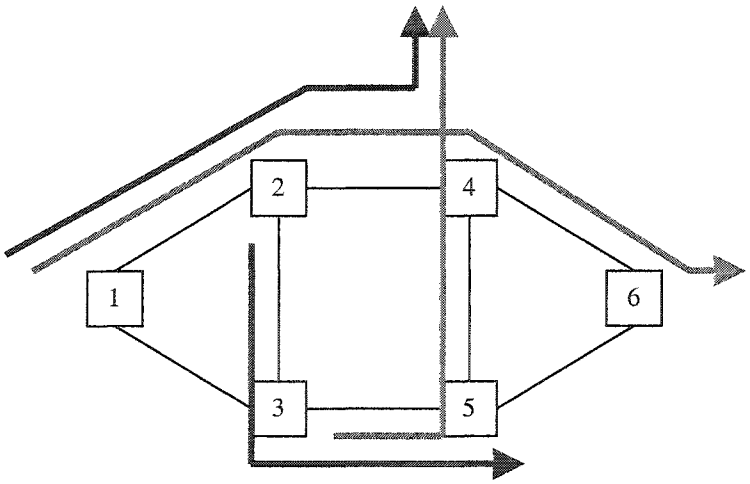
3.1.1 Exemple illustratif

La Figure 3.1 est une illustration du problème de RWAP pour le scénario sans panne. La topologie du réseau est représentée à la Figure 3.1a. L'ensemble D des trafics à acheminer est composé des paires (1,4), (1,6), (2,5), (2,6) et (3,4). On dispose de deux longueurs d'ondes λ_1 et λ_2 représentées respectivement par les couleurs bleu et rouge. La

Figure 3.1b présente une solution possible de routage. À chaque trafic, une longueur d'onde est attribuée de telle sorte que, sur un lien donné, deux trafics n'aient pas la même longueur d'onde. Dans la solution proposée, le trafic *(2,6)* n'a pu être acheminé. En effet, avec la solution de routage choisie, on se rend compte que, quel que soit le chemin choisi entre les nœuds *2* et *6*, il n'y a pas de longueur d'onde disponible, à moins de bloquer un autre trafic.



(a) topologie du réseau



(b) Solution potentielle

Figure 3.1 Illustration du problème de RWA

Dans la section qui suit, nous présenterons les méthodes que nous proposons pour la résolution du problème RWAP. Les méthodes de résolution exactes étant très difficiles à développer pour des réseaux de taille réelle, et l'utilisation de programmes linéaires étant très souvent coûteuse en temps et en ressources, nous proposerons des heuristiques permettant d'atteindre nos objectifs.

3.2 Heuristiques proposées

Les solutions proposées sont basées sur une combinaison des méthodes de sélection de chemins pour le routage et de coloriage de graphes pour l'affectation de longueurs d'onde. Dans cette section, nous définissons l'approche générale que nous préconisons pour résoudre le problème RWAP ainsi que les différents algorithmes de mise en œuvre.

3.2.1 Schéma général

Comme nous l'avons mentionné, nous adoptons une approche de décomposition du problème. Dans un premier temps, nous faisons la recherche et la sélection de routes pour acheminer les trafics. Ainsi, pour une paire origine-destination donnée, nous considérons tous les chemins possibles. Par la suite, nous construisons un graphe auxiliaire à colorier pour l'affectation de longueurs d'onde. Ce graphe est noté G_t et est construit pour un scénario t de T. Le schéma général est présenté à la Figure 3.2.

Pour le scénario de base t_0
- Routage : sélection des chemins et construction du graphe auxiliaire G_{t0}
- Affectation de longueurs d'onde : coloriage du graphe G_{t0}

Pour les scénarios de panne t
- Routage : sélection des chemins et construction du graphe auxiliaire G_t
- Affectation de longueurs d'onde : coloriage du graphe G_t

Figure 3.2 Schéma général de résolution

Le graphe auxiliaire, G_t, est construit pour un scénario t de T, après la recherche des chemins. À chaque chemin trouvé, est associé un sommet dans le graphe G_t. Les sommets sont regroupés en fonction du trafic auquel ils correspondent. Un graphe G_t comporte ainsi autant de groupes de sommets qu'il y a de demandes à acheminer. Deux sommets sont reliés par une arête si et seulement s'ils n'appartiennent pas au même groupe et si les chemins qu'ils représentent sont adjacents, c'est-à-dire qu'ils partagent au moins un lien. Pour construire le graphe auxiliaire, seuls les chemins dits *légaux* sont considérés pour chaque trafic. Un *chemin légal* est défini pour chaque scénario, à partir des contraintes énoncées précédemment. Nous rappelons que les règles à suivre sont:

- Une demande d doit être satisfaite par un chemin de $o(d)$ à $d(d)$;
- Une demande non acheminée dans le scénario de base ne le sera pas dans les autres scénarios ;
- Une demande permanente acheminée sur une route dans le scénario de base devra être acheminée sur cette même route dans les scénarios de panne.

Ainsi, pour le scénario de base, pour toute demande d, tout chemin de $o(d)$ à $d(d)$ est *légal*. Pour les scénarios de panne par contre, nous avons les cas de figure suivants :

- Pour toute demande non routée dans le scénario de base, aucun chemin n'est *légal* (relation (3.3));
- Pour toute demande permanente routée dans le scénario de base, il y a un seul *chemin légal* (relation (3.2));
- Pour toute demande robuste routée dans le scénario de base, tout chemin non interrompu est *légal*. C'est-à-dire que tout chemin non touché par la panne associée au scénario dans lequel on se trouve est *légal*.

Pour un réseau donné, le nombre de chemins possibles pour un trafic est exponentiel, de ce fait, en considérant tous les chemins pour chaque trafic, nous risquons fort de nous retrouver avec trop de sommets dans le graphe auxiliaire. Pour cette raison, nous allons choisir les chemins à insérer dans le graphe auxiliaire en fonction d'un critère donné. Une fois le critère de sélection déterminé, nous pouvons choisir d'insérer dans le

graphe auxiliaire tous les chemins qui satisferont nos exigences, ou juste un sous-ensemble de chemins. Afin de procéder au coloriage du graphe auxiliaire, nous disposons de plusieurs méthodes : des méthodes exactes, gloutonnes ou de recherche locale. Cependant, les méthodes exactes sont difficiles à développer, et les méthodes gloutonnes donnent souvent des solutions loin de l'optimalité. Un trafic dont aucun sommet du graphe ne sera colorié sera bloqué. Le fait de regrouper les sommets dans le graphe auxiliaire introduit une contrainte supplémentaire : *Deux sommets correspondant au même trafic ne pourront pas être coloriés en même temps*. Dépendamment du mode de coloriage choisi, il va sans dire que la solution de routage proposée sera plus ou moins bonne.

Nous proposons de faire la sélection des chemins, pour chaque trafic, en se basant sur les plus courts chemins en terme de saut, et d'utiliser une méthode de recherche locale utilisant les listes taboues pour l'affectation de longueurs d'onde. Nous appelons cette méthode *l'algorithme TabouP*. Pour construire le graphe auxiliaire, nous ne prenons en compte qu'un maximum P de chemins, pour chaque trafic. Pour chaque scénario et chaque demande, nous choisissons alors :

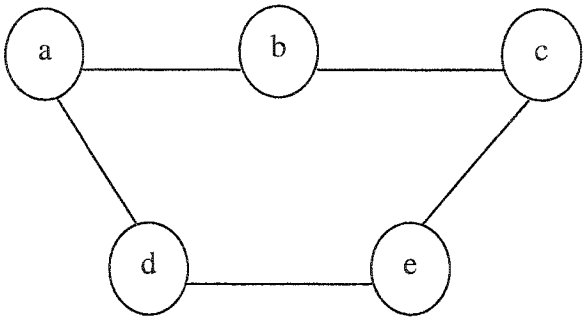
- tous les chemins légaux optimaux (les plus courts) s'il y a P chemins légaux optimaux ou moins ;
- un échantillon aléatoire de P chemins optimaux s'il y en a plus que P.

Pour commencer, nous cherchons tous les plus courts chemins possibles pour router les trafics et nous n'en considérons que P au plus. Par la suite, l'algorithme se déroule en deux étapes précises : d'abord le traitement du scénario de base t_0 et ensuite les traitements des scénarios de panne. Les plus courts chemins sont trouvés en utilisant une version de l'algorithme de Dijkstra qui permet, pour toute paire de sommets d'un graphe donné, de trouver tous les plus courts chemins. L'étape d'affectation de longueurs d'onde correspond au coloriage du graphe auxiliaire. Il est important de noter que cette étape permet non seulement de trouver une longueur d'onde mais aussi de trouver le chemin sur lequel le trafic sera routé. En effet, pour obtenir le graphe à colorier, on considère un ensemble possible de chemins, mais aucune sélection n'est faite.

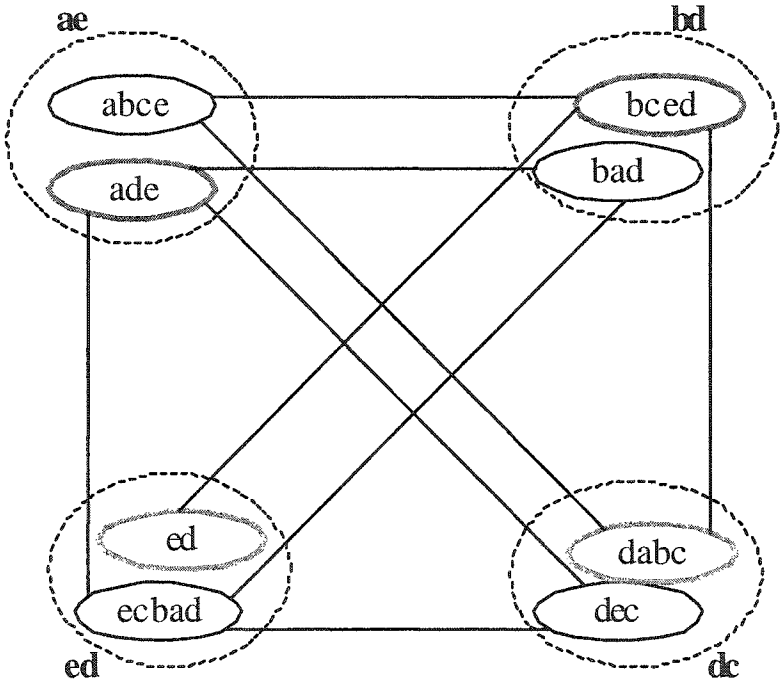
L'exemple présenté à la Figure 3.3 illustre le déroulement global de la méthode proposée ci-haut. Nous allons illustrer l'approche de résolution présentée ci-haut pour la topologie représentée à la figure 3.3a. Dans ce cas, nous considérons tous les chemins possibles pour chaque trafic. Pour un ensemble de demandes, nous mettons l'accent sur la construction du graphe auxiliaire. Nous présentons également l'obtention de la solution de routage et d'affectation de longueurs d'onde à partir du coloriage du graphe auxiliaire. Dans cet exemple, nous ne présenterons que le scénario de base noté t_0. On a :

- $V_R = \{a,b,c,d,e\}$;
- $E_R = \{(a,b),(a,d),(b,e),(d,e),(e,c)\}$;
- $D = \{(a,e),(b,d),(e,d),(d,c)\}$;
- k=2, les deux longueurs d'onde étant représentées par les couleurs bleu et rouge lors de l'étape de coloriage.

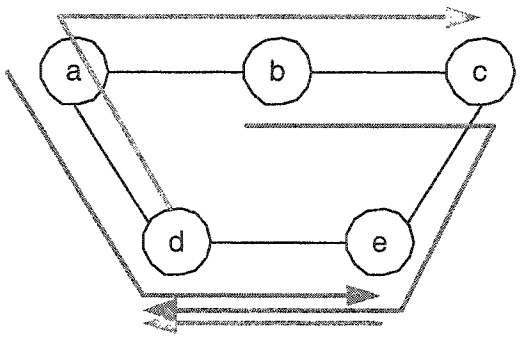
À partir du graphe G_R, nous trouvons tous les chemins pour chaque trafic. Pour le trafic de *a* à *e* par exemple, il y a en tout deux chemins possibles : ***a-d-e***, et ***a-b-c-e***. Par la suite, le graphe G_{t0} (scénario sans panne) est construit en regroupant tous les sommets correpondant au même trafic, tel que montré à la Figure 3.3b. Afin de faciliter la compréhension du graphe, les sommets sont nommés avec le chemin qu'ils représentent. Les arêtes sont placées en fonction des adjacences. Nous sommes dans le scénario de base donc, tous les chemins possibles pour chaque trafic sont légaux. On remarquera que certains sommets, comme par exemple *ade* et *ed*, ne sont pas reliés, ceci tout simplement parce qu'ils n'ont pas d'arcs en commun. C'est la particularité de la notion d'adjacence que nous avons mentionné dans le premier chapitre. Une fois le graphe auxiliaire construit, il est colorié selon la technique choisie. La Figure 3.3b présente également une configuration possible de coloriage. En utilisant cette configuration, on obtient la solution de routage présentée à la Figure 3.3c. Pour chaque trafic, on regarde le groupe de sommet qui lui est associé dans le graphe auxiliare, et il est routé sur le chemin correspondant au sommet colorié du groupe. Si aucun sommet du groupe n'est colorié il est bloqué.



(a) graphe G_R du réseau



(b) construction du graphe G_{t0} : exemple de coloriage



(c) Solution trouvée

Figure 3.3 Illustration de l'approche de résolution pour le scénario sans panne t_0

3.2.2 Routage et construction du graphe auxiliaire dans l'algorithme TabouP

Pour chaque scénario, la sélection des chemins et la construction du graphe sont les premières étapes de résolution. Les données utilisées sont : le graphe $G_R = (V_R, E_R), D = D^P \cup D^R$ et l'ensemble des arêtes valides dans un scénario t, E_t. La Figure 3.4 montre le détail de la sélection des chemins et de la construction du graphe auxiliaire. Pour cela, nous introduisons les données et variables suivantes :

- Le graphe auxiliaire $G_t = (V_t, E_t)$, qui est construit à partir des adjacences entre les chemins pour chaque scénario $t \in T$. Nous utiliserons la lettre μ pour désigner un chemin ;
  - V_g l'ensemble des sommets du graphe G_t représentant les trafics ;
  - E_g, l'ensemble des arêtes du graphe G_t. Pour deux chemins μ_1 et μ_2, $(v_{\mu_1}, v_{\mu_2}) \in E_g \Leftrightarrow \mu_1 \cap \mu_2 \neq \varnothing$, et μ_1 et μ_2 ne correspondent pas au même trafic ;
  - C_t, l'ensemble des groupes de sommets. $C_t = \{c_1, c_2, ..., c_{|D|}\}$ et $|C_t| = |D|$;
- P le nombre maximum de chemins à considérer pour chaque trafic ;
- M_d l'ensemble des plus courts chemins pour un trafic $d \in D$. $M_d \subseteq PATH(E_t, o(d), d(d))$ qui, on le rappelle, est l'ensemble de tous les chemins possibles entre la paire origine-destination du trafic d, pour E_t, l'ensemble des arêtes valides dans le réseau.

Pour commencer, tous les plus courts chemins sont trouvés pour chaque trafic. Nous construisons ainsi les ensembles M_d pour chaque trafic. Ensuite, l'algorithme se déroule en deux parties. La première est relative au scénario de base t_0 et la seconde aux scénarios de pannes. Pour le scénario de base, un graphe vide est créé. Les groupes de sommets sont insérés par la suite. Dans chacun d'eux, nous plaçons au maximum P chemins choisis aléatoirement dans M_d. Ensuite, les chemins sont comparés deux à deux afin d'insérer les arêtes dans le graphe. Les arêtes reliant deux sommets de deux groupes différents dont les chemins sont adjacents. Une fois le graphe complété, il est colorié et la solution de routage pour le scénario de base est obtenue. Cette solution est représentée par un triplet $\left(D_t^{'}, \mu_t, \lambda_t\right)$. Pour tous les autres scénarios, le graphe auxiliaire est créé vide au départ tout comme pour le scénario de base t_0. Conformément à la contrainte (3.3), nous ne considérons que les trafics acheminés dans le scénario de base pour construire le graphe auxiliaire dans un scénario de panne. Car, tous les trafics bloqués dans le scénario de base le sont également dans tous les scénarios de panne. Ainsi, pour chaque demande d routée dans le scénario de base, $d \in D_{t0}^{'}$, le groupe de sommet c_d est ajouté dans le graphe. À ce niveau de l'algorithme, les demandes permanentes et robustes sont traitées différemment. Pour mieux illustrer le déroulement de l'algorithme pour les demandes robustes, nous introduisons une variable notée E, qui est un ensemble dans lequel tous les chemins valides pour une demande robuste sont stockés. Un chemin valide étant un chemin non interrompu, c'est-à-dire non touché par la panne associée au scénario traité. Ainsi, pour chaque trafic robuste d, nous insérons tous les chemins valides de M_d dans E. Pour un trafic d, si l'ensemble E est vide, c'est-à-dire qu'aucun chemin de M_d n'est valide dans ce scénario, les plus courts chemins sont recalculés en fonction de la panne du réseau et introduits dans E. Cette opération est réalisée en mettant dans E tous les éléments de $MINPATH(E_t, o(d), d(d))$ qui l'ensemble des chemins minimaux entre l'origine et la destination d'un trafic d donné. Parmi les chemins présents dans l'ensemble E, nous choisissons au maximum P que nous introduisons dans le groupe de sommets correspondant au trafic considéré. Pour chaque demande permanente d, un seul sommet est introduit dans le groupe c_d. En effet, selon la contrainte (3.2), toute demande

permanente est routée sur un chemin dans le scénario de base est routée sur le même chemin dans tous les scénarios de panne. De cette manière, le sommet introduit dans le groupe c_d pour une demande permanente d correspond à $\mu_{t0}(d)$ qui est obtenu de la solution $\left(D'_{t0}, \mu_{t0}, \lambda_{t0}\right)$. À la fin, le graphe obtenu est colorié pour trouver la solution $\left(D'_t, \mu_t, \lambda_t\right)$.

POUR toute demande $d \in D$
    Construire l'ensemble M_d des plus cours chemins entre o(d) et d(d).
/* **POUR** le scénario sans panne t_0 */
    Construire le graphe auxiliaire G_{t0} vide.
    POUR toute demande $d \in D$ **FAIRE**
        Insérer le groupe c_d dans G_{t0}.
        Choisir aléatoirement au maximum P chemins de M_d et insérer dans c_d les sommets correspondants (le sommet associé au chemin μ est noté v_μ).
    POUR toute paire de sommets $(v_{\mu 2}, v_{\mu 1})$ dans G_{t0} **FAIRE**
        SI $v_{\mu 2}$ et $v_{\mu 1}$ n'appartiennent pas au même groupe **ET** μ_2 et μ_1 adjacents
            Insérer l'arête $(v_{\mu 2}, v_{\mu 1})$ dans G_{t0}.
Colorier le graphe G_{t0} : on obtient la solution S = $\left(D'_{t0}, \mu_{t0}, \lambda_{t0}\right)$.
POUR tout scénario de panne $t \in T - \{t_0\}$ **FAIRE**
    Construire le graphe auxiliaire G_t vide.
    POUR toute demande $d \in D$ **FAIRE**
        SI d est routé dans t_0 ($d \in D'_t$)
            Insérer le groupe c_d dans G_t
            SI d est robuste ($d \in D^R$)
                Affecter $E \leftarrow \{\}$
                POUR tout $\mu \in M_d$ **FAIRE**
                    SI μ n'est pas interrompu
                        Insérer μ dans E
                    SI E vide
                        Faire $E \leftarrow MINPATH(E_t, o(d), d(d))$
                        Choisir aléatoirement au maximum P chemins de E et insérer dans c_d les sommets correspondants.
            SINON SI d est une demande permanente ($d \in D^P$)
                Insérer $v_{\mu 0(d)}$ dans c_d
    POUR toute paire de sommets $(v_{\mu 2}, v_{\mu 1})$ dans G_t **FAIRE**
        SI μ_2 ne correspond au même trafic que μ_1 **ET** μ_2 et μ_1 adjacents
            Insérer l'arête $(v_{\mu 2}, v_{\mu 1})$ dans G_{t0}.
Colorier le graphe G_{t0} : on obtient la solution S = $\left(D'_t, \mu_t, \lambda_t\right)$.

Figure 3.4 Détail des étapes de routage et de construction du graphe auxiliaire pour *TabouP*

3.2.3 Heuristique pour l'affectation des longueurs d'onde

Pour chaque scénario nous construisons un graphe auxiliaire que nous devons ensuite colorier. Comme nous l'avons mentionné, l'heuristique de coloriage est basée sur une méthode constructive de coloriage de graphes qui utilise la recherche taboue. Une solution potentielle (coloriage) est définie par l'ensemble des sommets ainsi que les couleurs qui leurs sont associées. Dans la suite, nous présenterons une solution par un tableau *C*, dans lequel seront placées les informations sur la couleur de chaque sommet. Ainsi, un élément *C[i]* du tableau indiquera la couleur affectée au sommet *i*. Le symbole • correspondra à un sommet non colorié. Dans chacune des classes de sommets $C_1 \ldots C_{|D|}$, au plus un sommet devra être colorié. Pour être valide, une solution devra vérifier les contraintes présentées dans les sections précédentes.

L'ensemble des coloriages ou configurations est noté *S*. La fonction d'évaluation *f*, $f : S \rightarrow R$, indique la valeur attribuée à chaque solution. Elle est aussi appelée fonction coût. Pour toute configuration *s* de *S*, *f(s)* est défini comme étant le nombre de sommets coloriés dans *s*. Compte tenu de la contrainte liée aux graphes auxiliaires, nous avons $f(s) \leq |D|$. Ceci veut dire que, pour une configuration *s*, le nombre de sommets coloriés est toujours inférieur ou égal au nombre de classes. En effet, deux sommets d'une classe ne pouvant être coloriés en même temps, nous aurons au maximum autant de sommets coloriés qu'il y a de classes ou groupes de sommets. Pour trouver une solution, nous proposons une méthode de recherche locale. Pour ce faire, partant d'une configuration *s* de *S*, nous choisissons une configuration *s'* de *S* appartenant au voisinage de *s*. Le rôle de la procédure taboue est de trouver une configuration avec une évaluation aussi élevée que possible.

Typiquement, une procédure taboue commence avec une configuration initiale s de l'ensemble des configurations *S*. Ensuite, le processus se déroule itérativement en analysant à chaque fois le voisinage de la configuration courante. À chaque itération, un mouvement est appliqué à la configuration courante s pour passer à une configuration *s'*, ceci, même si la fonction coût n'est pas améliorée de *s* à *s'*. Pour éviter que la procédure taboue ne tourne en rond ou qu'elle ne se retrouve coincée dans des optimum locaux, la

notion de *liste taboue* est introduite. La liste taboue contient un historique des configurations rencontrées dans les itérations précédentes. De cette manière, les configurations présentes dans la liste taboue ne seront pas considérées pendant un nombre donné d'itérations. L'espace de recherche (l'ensemble des configurations) et la fonction coût ont été présentés précédemment. Nous définissons ci-après les autres composantes de la procédure taboue.

À chaque itération, la procédure taboue effectue un mouvement à partir de la configuration courante. Un mouvement consiste à colorier un sommet x du graphe avec une couleur v et à décolorier tous les voisins de x coloriés avec v. Il est noté « x,v ». nous définissons également un mouvement élémentaire, il n'implique pas de coloriage et est noté <x,v>. Pour une configuration s, le fait d'appliquer un mouvement « x,v » permet d'obtenir une configuration s' voisine de s. À chaque itération, le mouvement ayant la meilleure performance est choisi. Pour calculer la performance d'un mouvement « x,v », il faut analyser tous les sommets voisins de x. Nous introduisons la structure *gamma* notée γ qui, pour un sommet x et une couleur v est telle que $\gamma(x,v)$ représente le nombre de voisins de x coloriés avec v dans une configuration s. De cette manière, le mouvement de performance la plus élevée est le mouvement « x,v » pour lequel $\gamma(x,v)$ a la plus petite valeur.

Lorsqu'un mouvement « x,v » est effectué, il est rendu tabou en l'introduisant dans la liste taboue. Nous introduisons à ce niveau la variable *TabuTenure* qui est telle que pour un couple (x,v), *TabuTenure*(x,v) est l'itération à partir de laquelle le mouvement « x,v » ne sera plus tabou. Pour déterminer la valeur de TabuTenure pour un couple (x,v) donné, nous calculons le nombre d'itérations pendant lesquelles « x,v » sera tabou en fonction de la longueur de la liste taboue qui est une variable notée *lgtl*. Au cours des itérations pendant lesquelles un mouvement « x,v » est tabou, nous ne pouvons pas re-assigner la couleur v au sommet x. Cependant, un mouvement tabou dont l'exécution permettrait d'obtenir une configuration de meilleur coût que la configuration courante peut être choisi. C'est le *critère d'aspiration*. Il permet d'effectuer un mouvement tabou pour aller chercher une meilleure solution. Dans la suite de notre

travail, nous avons choisi de ne pas appliquer ce critère afin de simplifier le déroulement de notre heuristique.

La figure 3.5 présente le déroulement global de la procédure taboue appliquée au coloriage du graphe auxiliaire. Nous rappelons que le graphe auxiliaire est constitué des éléments suivants :

- Un graphe non orienté $G=(V,E)$, avec l'ensemble V des sommets correspondant aux chemins présélectionnés pour chaque trafic. On définit $n = |V|$;
- Une partition $C_1, C_2,\ldots, C_{|D|}$, de l'ensemble V ;
- Un nombre k de couleurs correspondant au nombre de longueurs d'onde disponibles dans le réseau. Les couleurs seront notées 1, 2,…,k.

Cette procédure correspond à l'heuristique d'affectation des longueurs d'onde. Pour commencer, le graphe à colorier est initialisé. Puis, pour chaque itération, le meilleur mouvement trouvé est effectué. Si le coût de la configuration obtenue après le mouvement est meilleur que celui de la meilleure solution jusque là trouvée, la configuration courante devient la meilleure configuration. Nous avons deux critères d'arrêt pour les itérations : Atteindre le nombre maximum d'itérations appelé *It_max* et réussir à colorier tous les groupes ou partitions, sachant qu'un groupe est colorié si une couleur a été trouvée pour au plus un de ses sommets.

```
Initialiser le graphe pour obtenir une configuration de départ s
Faire s*←s, s* étant la meilleure solution trouvée
Faire It←0, initialiser les itérations à 0
TANT QUE It < It_Max ET tous les groupes ne sont pas coloriés
        Choisir le meilleur mouvement
        Effectuer ce mouvement
        Mettre à jour la liste taboue
        SI f(s) > f(s*)
                f(s*) ←f(s), s*←s
        Incrémenter It : It←It + 1
```

Figure 3.5 Heuristique d'affectation de longueurs d'onde

L'initialisation du graphe correspond à une configuration vide, c'est-à-dire qu'aucun sommet n'est colorié. Ainsi, la liste taboue est vide et tous les $\gamma(x,v)$ valent zéro. $\gamma(x,v)$ étant le nombre de sommets voisins de x coloriés avec v. La figure 3.6 détaille le choix du mouvement et les opérations qui en découlent. Un mouvement est choisi parmi tous ceux qui sont valides. Un mouvement « x,v » est valide si le groupe C auquel x appartient n'est pas encore colorié, c'est-à-dire qu'aucun sommet de C n'est colorié. À ce niveau, chaque mouvement non tabou est analysé et sa performance calculée. Si toutefois tous les mouvements valides sont tabous, nous en choisissons un aléatoirement.

POUR tout sommet $x \in V_t$
    SI x est candidat (le groupe de x n'est pas colorié)
        POUR toute couleur $v \in \{1,\ldots,k\}$
            SI « x,v » n'est pas tabou, $It \geq TabuTenure(x,v)$
                Calculer $\gamma(x,v)$

SI tous les mouvements « x,v », avec x candidat, sont tabous
    Choisir un mouvement « x,v » aléatoirement, pour x candidat

SINON
    SI plus d'un mouvement a la plus petite valeur de γ
        Choisir un aléatoirement
    SINON
        Choisir le mouvement qui a la plus petite valeur de γ

Mettre *TabuTenure* et γ à jour

Calculer le coût

Figure 3.6 Sélection et exécution d'un mouvement

Lorsque le mouvement à effectuer est choisi, la liste taboue est mise à jour ainsi que la structure γ. La mise à jour associée à un mouvement « x,v » comprend les points suivants :

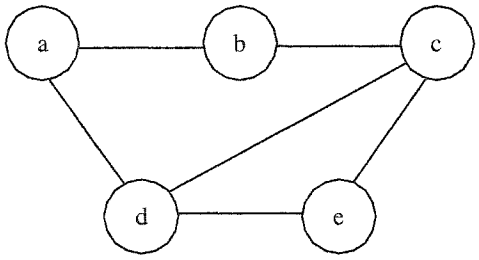
- attribuer la couleur v au sommet x ;

- décolorier tous les voisins y de x coloriés avec v et recalculer γ pour tous les voisins des sommets y voisins de x ;
- rendre « x,v » tabou en recalculant *TabuTenure(x,v)* à partir de l'itération courante et de la longueur de la liste taboue *lgtl*.

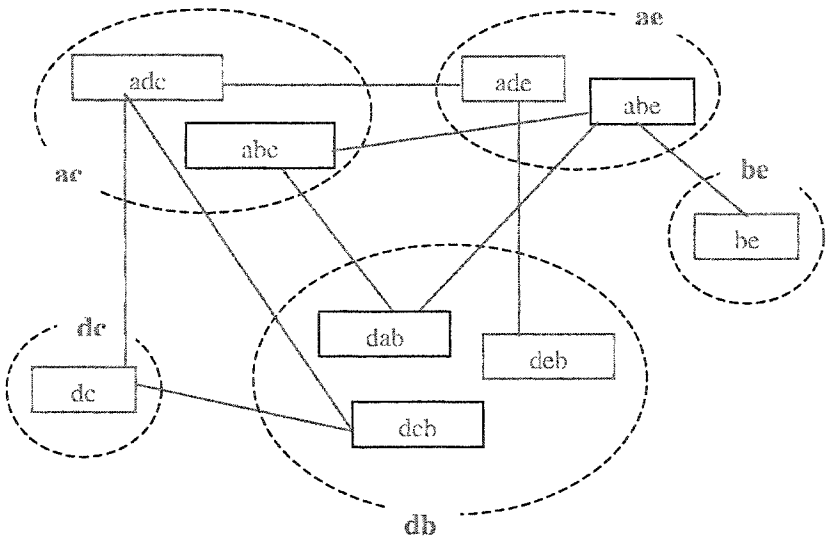
Pour tous les scénarios, nous commençons avec un graphe dont tous les sommets sont non coloriés. Pour les cas de pannes, la solution trouvée pour le scénario sans panne est recyclée pour avoir une configuration de départ non vide. Pour ce faire, nous assignons aux sommets de G, présents également dans le graphe du scénario de base t_0, la couleur qu'ils avaient dans la solution de coloriage obtenue à la fin du scénario de base. Ensuite, il faut procéder à une réinitialisation de la structure γ en tenant compte des sommets coloriés. La liste taboue quant à elle est laissée vide. Pour terminer, nous calculons la fonction coût de cette configuration en comptant le nombre de groupes coloriés.

3.2.4 Illustration de l'heuristique d'affectation de longueurs d'onde

Dans cet exemple, nous considérons une version de l'algorithme *TabouP* appelée *TabouN* dans laquelle, nous prenons en compte tous les plus courts chemins possibles pour chaque trafic. La Figure 3.7b représente le graphe auxiliaire obtenu pour le scénario de base. Ce graphe correspond au réseau présenté à la Figure 3.7a. Nous avons cinq trafics à acheminer : (a,e), (a,e), (d,c), (d,b) et (b,e), avec (a,c) et (d,b) permanents. Nous disposons de deux couleurs pour colorier le graphe, rouge et bleu. Afin de faciliter la compréhension de cet exemple, nous avons omis de présenter à chaque étape l'état de la liste taboue.



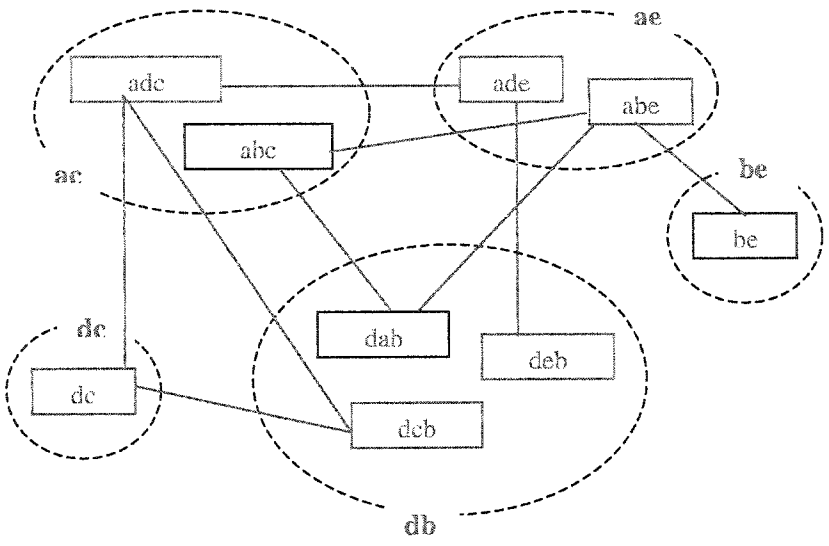
(a) topologie du réseau



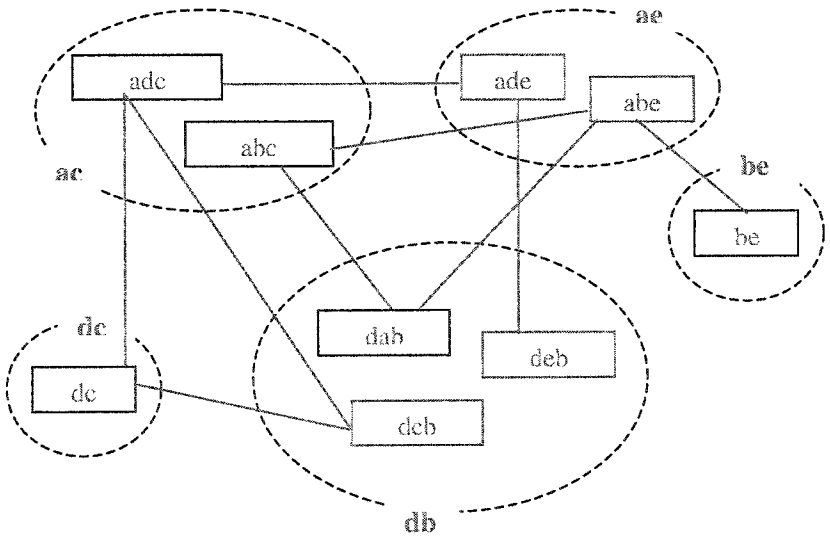
(b) Graphe Gt0 : configuration initiale

Figure 3.7 Configurations de départ pour l'affectation de longueurs d'onde : scénario sans panne

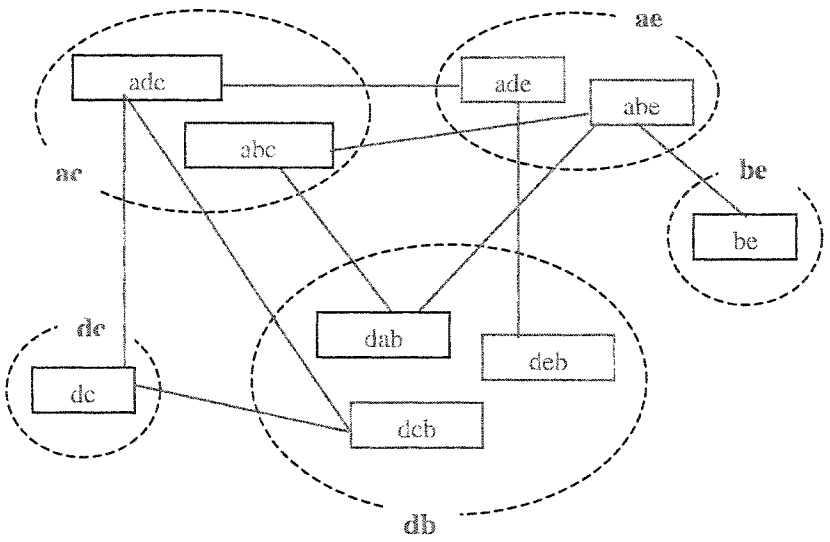
Lorsque l'affectation des longueurs d'onde débute, tous les sommets sont non coloriés. Ensuite, le mouvement de meilleure performance est choisi. La performance d'un mouvement « *x,v* » est fonction du nombre de sommets voisins de *x* coloriés avec *v*. À la première itération, tous les mouvements ont la même performance, on en choisit un aléatoirement. La Figure 3.8 illustre trois itérations de l'heuristique. La Figure 3.8a correspond à une itération *i* quelconque. Nous avons trois sommets coloriés. À l'itération *i+1*, nous avons deux mouvements possibles de même coût : « *dc, bleu* » et « *dc, rouge* ». Nous choisissons le premier, ce qui entraîne le décoloriage du sommet *adc* qui était colorié en bleu à la configuration précédente. À l'étape *i+2*, le mouvement de plus petite performance est « *abc, bleu* ». Après avoir exécuté ce mouvement, l'heuristique s'achève parce que tous les groupes de sommets ont été coloriés. Ceci veut dire qu'une longueur d'onde a pu être affectée à chaque trafic.



(a) État après une itération *i*



(b) État après l'itération i+1



(c) Solution finale

Figure 3.8 Illustration des itérations pour l'affectation de longueurs d'onde : scénario sans panne

Plaçons nous maintenant dans un scénario de panne, une panne du lien d-c par exemple. La Figure 3.9 montre le graphe auxiliaire correspondant à ce scénario après la mise à jour à partir de la solution de coloriage obtenue pour G_{t0}. Nous remarquons que cette panne bloque automatiquement le trafic *dc*. Il faut donc trouver les plus courts chemins pour router *dc* en fonction de l'état actuel du réseau. Il y a un seul chemin de coût minimal, *dec*. C'est pour cette raison que quand on compare le graphe G_t représenté à la Figure 3.8 au graphe G_{t0} précédent, on remarque que le sommet *dc* a disparu et que le *dec* a été introduit. Les sommets *adc* et *dcb* sont également ôtés du graphe, parce que les chemins qu'ils représentent sont touchés par la panne du lien d-c. Comme nous l'avons mentionné précédemment, le trafic *db* est permanent, ce qui veut dire qu'il doit être routé sur le même chemin dans tous les scénarios. Il doit donc être routé sur le chemin *dcb* dans tous les scénarios. Or, dans ce cas, ce chemin est touché par la panne. Pour cette raison, le trafic *db* sera automatiquement bloqué et tous les autres chemins faisant partie de ce

groupe sont retirés du graphe. Il s'agit de *dab* et *deb*. Tous les sommets qui ont été retirés sont représentés à la Figure 3.9 par des traits interrompus plus fins que ceux des autres sommets. Une fois les sommets retirés ou rajoutés, on se sert de la solution de coloriage de G_{t0} pour avoir une configuration de départ pour *Gt*. Dans ce cas, on attribue à tous les sommets de G_t présents également dans G_{t0} la même couleur que dans la solution finale de G_{t0}. C'est pour cette raison que quand on regarde la Figure 3.8, les sommets *abc* et *be* sont coloriés en bleu et *abe* est colorié en rouge. Tous les autres sommets sont non coloriés.

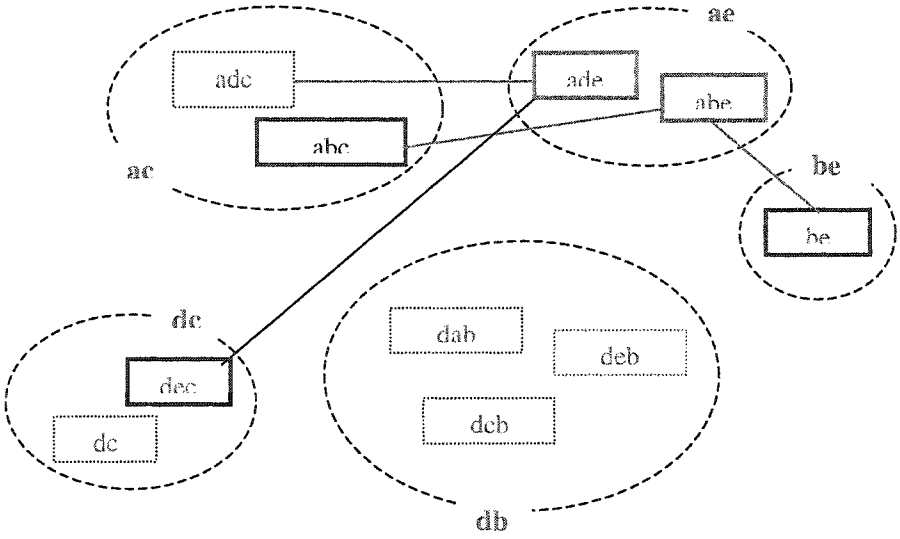


Figure 3.9 Configuration de départ pour G_t : Panne du lien d-c

Une fois la configuration initiale établie, l'heuristique se déroule de la même manière que pour le scénario de base. À chaque itération, un mouvement est choisi en fonction de sa performance.

3.3 Analyse de complexité des heuristiques proposées

La complexité des heuristiques proposées est évaluée en calculant la complexité des étapes principales. Dans un premier temps, nous allons analyser les étapes de routage

et de construction du graphe auxiliaire, puis nous étudierons l'heuristique d'affectation de longueurs d'onde.

3.3.1 Le routage et la construction du graphe auxiliaire

Avant de commencer, nous définissons :

- $d=|D|$, le nombre total de trafics à acheminer ;
- $n = |V_t|$, le nombre de sommets du réseau ;
- m, le nombre de liens du réseau, $2m=|E_t|$;
- $t = |T|$, le nombre de scénarios à traiter, $t = m+1$;
- M, le nombre d'itérations maximales pour l'affectation de longueurs d'onde ;
- k, le nombre de longueurs d'onde.

L'algorithme commence par la recherche de tous les plus courts chemins pour chaque trafic. La complexité de l'algorithme de Dijkstra utilisé est $O(m\ ln(n))$. Cet algorithme permet de calculer, pour un sommet d'un graphe, tous les plus courts chemins à tous les autres sommets du graphe. Donc, pour tous les trafics, la complexité totale est : $O(dm\ ln(n))$. Ensuite, pour chaque trafic, on sélectionne au maximum p chemins. Cette action est exécutée d fois, ainsi la complexité équivalente est $O(d)$. Ce qui fait que, pour la recherche des chemins, on a en tout :

$$O(dm \times \ln(n)) + O(d) = O(dm \times \ln(n))$$

Par la suite, pour le scénario de base, le graphe auxiliaire est construit. Pour ce faire, on compare tous les chemins retenus deux à deux en analysant les liens. Dans le pire cas, la complexité de cette partie est $O(p^2d^2(m-1)^2)=O(p^2d^2m^2)$ étant donné qu'on a d trafics ayant chacun p chemins au maximum et qu'un chemin passe au maximum par $m-1$ liens. Pour chaque scénario de panne, dans le pire cas, pour chaque trafic, il faudra appliquer Dijkstra une fois encore pour trouver de nouveaux chemins, ce qui correspond à $O(dm\ ln(n))$. Pour la construction de G_t, le pire cas correspond à l'établissement des adjacences qui, comme pour le scénario de base, a une complexité de $O(p^2d^2m^2)$. Pour tous les scénarios, on a donc une complexité de :

$$O(p^2d^2m^2)+ O(tp^2d^2m^2)+ O(tdm\ ln(n))= O(tp^2d^2m^2) + O(tdm\ ln(n))$$

Pour la recherche de chemins et la construction du graphe auxiliaire, on a ainsi une complexité de :

$$O(tp^2d^2m^2) + O(tdm\ ln(n\ )) + O(dm \text{ x } ln(n))= O(dm(tpdm + tln(n)))$$

3.3.2 L'affectation des longueurs d'onde

Pour l'affectation de longueurs d'onde, l'initialisation du graphe se fait en $O(pdk)$, car il y a au maximum p chemins par trafic et k couleurs de plus, nous parcourons chaque sommet et chaque couleur pour initialiser γ et la liste taboue. Par la suite, pour les scénarios de panne, pour la mise à jour, dans le pire cas tous les sommets sont comparés deux à deux afin d'initialiser la structure γ, ce qui fait $O(p^2d^2)$. À ce niveau, la complexité est :

$$O(tpdk) + O((t-1)\ p^2d^2)=O(tpdk + t\ p^2d^2)$$

Pour chaque itération, il faut choisir un mouvement, puis l'exécuter. Le choix d'un mouvement se fait en $O(pdk)$, parce que dans le pire cas, on parcourt chaque sommet et chaque couleur. Ensuite, exécuter un mouvement « x,v » implique d'analyser tous les sommets voisins de x. Chaque sommet ayant au maximum $pd-1$ voisins, tout ceci se fait en $O((pd-1)^2)$. La complexité équivalente aux itérations est donc :

$$O(Mtpdk)+O(Mt(pd-1)^2)=O(Mtpdk+Mtp^2d^2)$$

Ainsi, on peut dire que l'affectation de longueurs d'onde se fait en :

$$O(tpdk + t\ p^2d^2)+ O(Mtpdk+Mtp^2d^2)= O(Mtpdk+Mtp^2d^2)=O(Mtpd(k+pd))$$

La complexité globale de l'algorithme *TabouP* est la somme des complexités de l'affectation de longueurs d'onde et de la recherche de chemins et de la construction du graphe auxiliaire. Ce qui fait :

$O(Mtpd(k+pd))+O(dm(tpdm + tln(n)))= O(M(m+1)pd(k+pd))+ O(dm((m+1)pdm + tln(n)))$, car t=m+1. On a alors en tout :

$$O(pdm(M(k+pd)+dm^2)+ dmt \text{ x } ln(n)\ )$$

CHAPITRE IV
IMPLÉMENTATION ET RÉSULTATS

Dans ce chapitre, nous allons procéder à l'évaluation de performance de deux versions de l'algorithme *TabouP* que nous avons présenté dans le chapitre précédent : l'algorithme *TabouN* et l'algorithme *Tabou1*. pour *TabouN*, nous considérons tous les plus courts chemins pour la construction du graphe auxiliaire tandis que, pour *Tabou1*, nous ne considérons qu'un seul chemin par trafic. Cette évaluation sera réalisée à partir de résultats de diverses simulations. Dans un premier temps, nous allons décrire les détails de l'implémentation de ces algorithmes, à savoir les structures de données utilisées ainsi que l'environnement de programmation. Par la suite, nous présenterons le plan d'expérience ainsi que les réseaux sur lesquels nous avons effectué les simulations. Pour finir, nous analyserons les résultats obtenus.

4.1 Structures de données utilisées

Les algorithmes ont été implémentés en C++ avec le logiciel Visual C++ 6.0. Nous avons privilégié une structure de programmation orientée objet à cause de l'interaction existant entre les éléments à programmer. De plus, l'utilisation d'objets facilite la compréhension du code.

4.1.1 La classe CGraphe

Elle représente la topologie physique du réseau à traiter. Ses attributs principaux sont :

- *taille* : la taille du graphe qui est le nombre de sommets ;
- *NbLiens* : le nombre de liens du graphe ;
- *TabAdjacences* : la matrice des adjacence qui est représentée par un tableau à deux dimensions. Soit deux sommets i et j du graphe, *TabAdjacences[i][j]* est égal à 1 si les sommets i et j sont reliés par une arête; il est égal à 0 sinon.
- *LesChemins* : attribut contenant tous les plus courts chemins entre les paires de sommets du graphe. Un chemin est sauvegardé à l'aide d'une liste formée par les

sommets qui le composent. Nous avons utilisé le type *vector* de la librairie *vector* pour représenter les listes de sommets.

Outre l'implémentation de la topologie, le calcul de tous les plus courts chemins, la lecture des données du graphe à partir d'un fichier texte, les fonctions les plus importantes sont les suivantes :

- *CaculPcc* : elle permet de calculer tous les plus courts chemins entre un sommet du graphe et tous les autres sommets.
- *ChargerGraphe* : cette fonction permet de lire les données du graphe dans un fichier de topologie. Les fichiers à lire ont la forme suivante :

  Nombre de nœuds Nombre de liens

  Sommet1 Sommet2 Coût du lien

  Sommet1 Sommet2 Coût du lien...

  Tous nos liens ont des coûts unitaires parce que nous calculons les plus courts chemins en terme de sauts.
- *AffichePccs* : cette fonction permet d'afficher tous les plus courts chemins trouvés. L'affichage peut se faire à l'écran ou alors dans un fichier, dépendamment du paramètre d'entrée.

4.1.2 La classe CReseau

Dans cette classe, une fois la topologie physique enregistrée, le fichier des demandes est à son tour lu à partir d'un fichier. Les principaux attributs de cette classe sont :

- *RGraphe*: c'est un pointeur vers un objet de type CGraphe représentant le graphe associé au réseau.
- *Trafics* : c'est un vecteur de vecteurs contenant tous les trafics. Les trafics sont représentés à l'aide de vecteurs d'entiers formés de trois éléments qui sont, dans l'ordre : le type du trafic (permanent ou robuste), l'origine et la destination. Le type du trafic est soit 0 ou 1, selon qu'il soit respectivement robuste ou permanent.

- *LesPccs* : c'est un vecteur de chemins. Il représente la liste de tous les chemins retenus pour chaque trafic. Nous avons défini une structure *Chemins*, dont les attributs sont : la liste des sommets formant le chemin, l'état du chemin représenté par une variable booléenne, le numéro du trafic auquel le chemin est associé et le type du trafic. L'état d'un chemin représente sa validité, par exemple, un chemin touché par une panne ne sera pas valide. Le numéro d'un trafic est la position qu'il occupe dans la liste des trafics.

Les fonctions les plus importantes sont :

- *LireTrafic* : cette fonction permet de charger les données relatives au trafic à partir d'un fichier texte. Un exemple de fichier de trafic est présenté à la Figure 4.1. Sur la première ligne, on a le nombre total de demandes. Chaque ligne qui suit comporte les caractéristiques d'un trafic, à savoir : l'origine, la destination et le type dans cet ordre.
- *AffecterTrafic* : pour chaque trafic, cette fonction va retourner un certain nombre de chemins, et elle va mettre à jour la liste des chemins (*LesPccs*).
- *MiseAJour* : comme son nom l'indique, cette fonction permet d'effectuer la mise à jour de l'état des chemins dans les scénarios de panne.
- *TabAdj* : cette fonction retourne un tableau qui contient les adjacences entre les chemins. Ce tableau sera nécessaire pour la création du graphe auxiliaire.

```
11
1 0 1
1 0 0
1 4 0
2 3 1
2 4 0
3 0 0
3 2 1
3 2 0
3 4 1
4 1 1
4 1 0
```

Figure 4.1 Exemple de fichier de trafic

4.1.3 La classe CGrapheAux

Cette classe représente le graphe auxiliaire pour un scénario donné. Ces attributs sont :

- *NbSommets* : le nombre de sommets, de chemins.
- *NbGroupes* : le nombre de groupes de sommets, de trafics.
- *TabAdjacent* : un tableau à deux dimensions représentant les adjacences entre les chemins.
- *Groupes* : un tableau à une dimension qui, pour un sommet donné, donne le numéro de groupe auquel il appartient.

4.1.4 Les classes de coloriage

Nous avons trois classes qui permettent d'implémenter l'algorithme de coloriage du graphe auxiliaire : CColouring, CvertColoring et CRwaColoring.

➢ La classe *CColouring*

Cette classe permet de représenter une configuration de coloriage de graphes. Parmi ses attributs, on distingue le nombre de sommets du graphe à colorier, le tableau de couleur des sommets et le coût de la configuration.

➢ La classe *CVertColoring*

C'est la classe principale pour le coloriage de notre graphe auxiliaire. Elle permet d'introduire l'utilisation de listes taboues. Les attributs les plus importants sont :

- *max_vtx* : le nombre de sommets du graphe à colorier ;
- *max_col* : le nombre de couleurs disponibles ;
- *f* et *f_best* : respectivement, le coût de la configuration actuelle et de la meilleure configuration trouvée ;
- *pt_Colour* : pointeur vers un objet de type *CColouring* représentant la configuration courante ;
- *pt_BestColour* : pointeur de type *CColouring* représentant la meilleure solution trouvée ;

- *color* : tableau contenant les couleurs des sommets correspondant à l'attribut *pt_Colour* ;
- *gamma* : tableau d'entiers de deux dimensions correspondant à la structure de données contenant le nombre de voisins d'un sommet x colorié avec une couleur v ;
- *edge* : matrice à deux dimensions montrant les adjacences entre les sommets. Soit deux sommets x et y, on a *edge[x][y]* qui vaut 1 s'ils sont reliés et 0 sinon ;
- *iter* : itération courante ;
- *max_iter* : nombre maximal d'itérations ;
- *lgtl* : longueur de la liste taboue ;
- *tabuTenure* : matrice à deux dimensions contenant les mouvements tabous. Soit x un sommet et v une couleur, *tabuTenure[x][v]* représente l'itération jusqu'à laquelle le mouvement « x,v » sera tabou. On rappelle que le mouvement « x,v » veut dire que le sommet x est colorié avec v.

L'affectation des longueurs d'ondes se fait grâce aux fonctions de cette classe, de la manière suivante :

- La fonction *run* gère le processus de coloriage. En effet, c'est ici que le processus est initialisé, de plus elle contrôle les itérations et sauvegarde la meilleure solution ;
- La fonction *init* permet d'initialiser tous les paramètres, tels que la liste taboue, la couleur des sommets et la structure *gamma* ;
- La fonction *iteration* permet d'implémenter le déroulement d'une itération. C'est ici qu'on va chercher le meilleur mouvement à faire et qu'il va être exécuté ;
- La fonction *exec_move* permet d'exécuter un mouvement. Elle prend en paramètre le numéro d'un sommet x et d'une couleur v et décolore les voisins de x coloriés avec v, sans oublier la mise à jour de la liste taboue ;

- La fonction *update*, réalisée uniquement pour les scénarios de panne, permet de mettre à jour la solution obtenue pour le scénario de base afin de l'utiliser comme point de départ pour les itérations.

➢ La classe *CRwaColoring*

Dans le graphe auxiliaire, les sommets étant organisés en groupes, il a fallu définir une structure de données qui permet d'adapter la notion de « coloriage de groupes de sommets » à celle de coloriage de sommets. Nous avons défini la classe *CRwaColoring* qui hérite de la classe *CvertColoring*. Ses attributs sont :

- *max_clust* : le nombre de groupes de sommets dans le graphe auxiliaire ;
- *cluster*: C'est un tableau d'entiers. Soit x appartenant à l'intervalle [0, *max_vtx*[, un sommet du graphe, *cluster[x]* représente le groupe auquel x appartient ;
- *is_ClusterColored* : C'est un tableau de variables booléennes tel que pour un groupe c, *is_ClusterColored[c]* vaut « vrai » si le groupe c est colorié et « faux » sinon, c étant pris dans l'intervalle [0, *max_clust*[.

En plus des fonctions de la classe mère auxquelles on a accès, cette classe permet de :

- Initialiser tous les groupes du graphe grâce à la fonction *init*. Elle va mettre tous les champs du tableau *is_ClusterColored* à « faux » ;
- Faire une sélection, à chaque itération, des sommets à colorier grâce à la fonction *isVertexCandidate*. Elle prend en paramètre un sommet x et vérifie qu'aucun sommet du même groupe que x n'est déjà colorié. Cette vérification permet d'éviter d'avoir deux sommets du même groupe coloriés en même temps.

4.2 Données utilisées pour les tests

Notre travail étant basé sur la formulation du problème proposée par Oulaï [16], nous avons dans un premier temps, choisi des données parmi celles qu'il a présentées.

4.2.1 Les fichiers de topologie

Afin de réaliser les simulations, nous avons retenu trois réseaux. Le Tableau 4.1 résume les tailles de ces réseaux.

Tableau 4.1 Taille des réseaux utilisés

réseaux	Nombre de nœuds	Nombre de liens
Aléatoire	20	25
ARPANET	20	31
NSFNET	14	21

Le premier est présenté à la Figure 4.2. Il comporte 20 nœuds et 25 liens. C'est un des réseaux qui a été utilisé par Oulaï [16] pour présenter ses résultats.

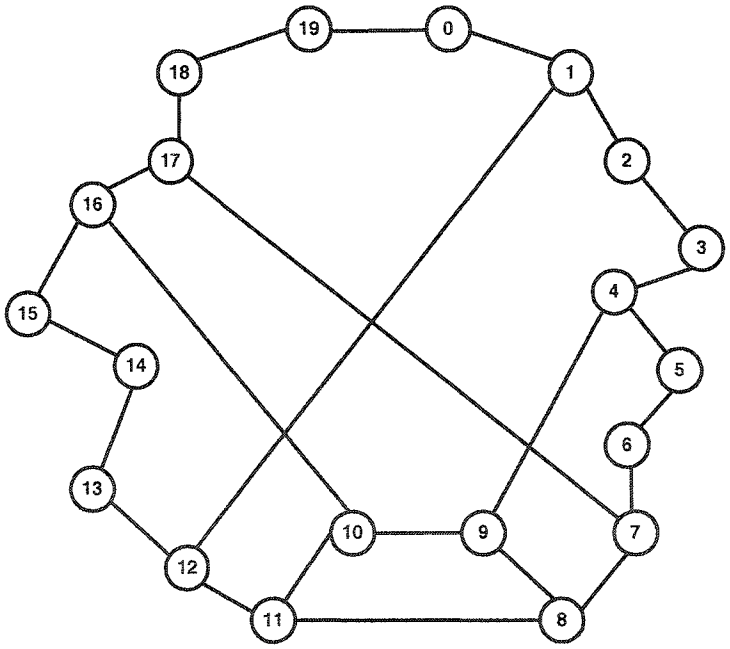


Figure 4.2 Topologie du réseau de 20 nœuds et 25 liens

Il a été généré aléatoirement à partir d'un programme ayant comme variables d'entrées le nombre de nœuds n et le nombre de liens l, l étant supérieur à n. Afin d'assurer la bi-connexité, les sommets sont d'abord reliés de manière à créer une boucle

passant une seule fois par chacun d'eux. En procédant ainsi, le réseau comporte un nombre de liens égal au nombre de nœuds. Les liens restant sont alors placés aléatoirement entre les paires de sommets du réseau non encore reliés.

Les deux autres réseaux utilisés sont ARPANET et NSFNET. Nous avons utilisé la topologie du réseau ARPANET comportant 20 nœuds et 31 liens. Elle est représentée à la Figure 4.3. Quant à NSFNET, montré à la Figure 4.4, nous avons 14 nœuds et 21 liens. Nous avons choisi ces deux réseaux parce qu'ils sont deux des réseaux les plus utilisés pour présenter des résultats dans la littérature.

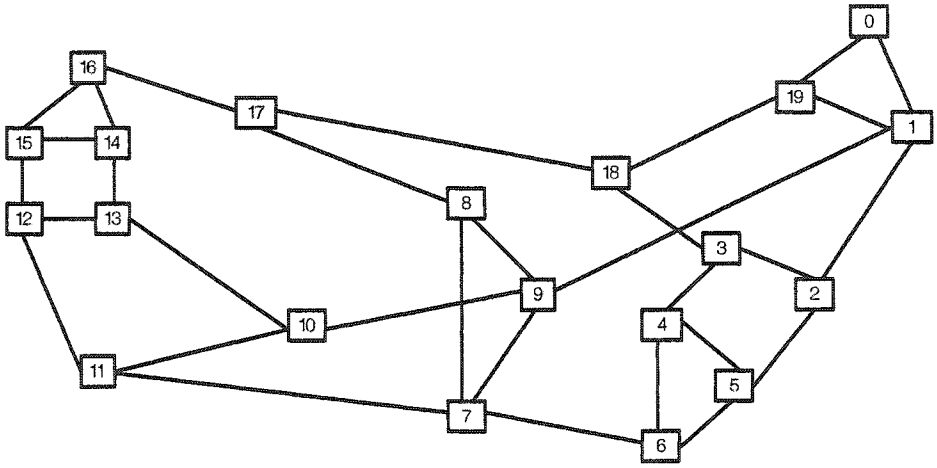


Figure 4.3 Topologie du réseau ARPANET

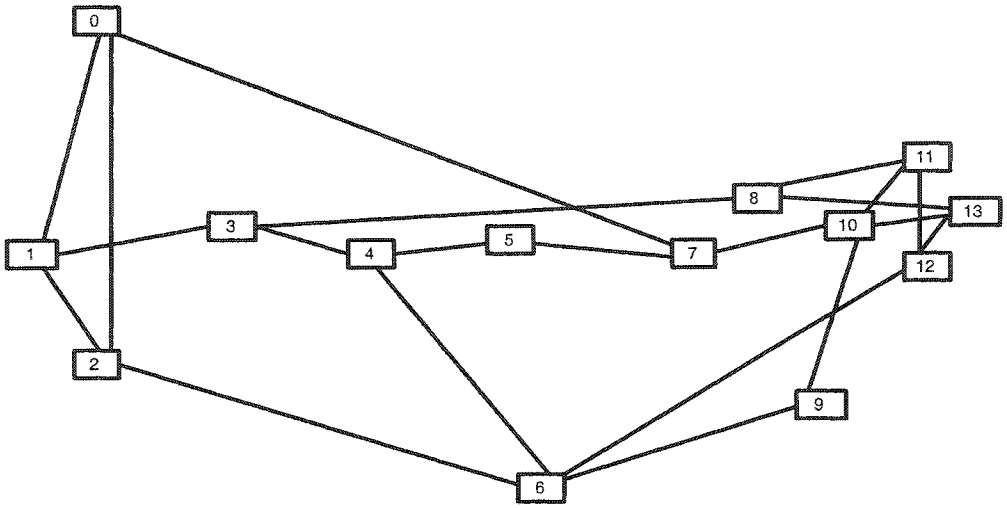


Figure 4.4 Topologie du réseau NSFNET

4.2.2 Les fichiers de trafic

Tout comme le réseau aléatoire présenté ci-haut, les fichiers de trafic ont été générés par un programme en fonction du nombre de nœuds des réseaux. Nous avons utilisé deux fichiers pour le trafic. Le premier, adapté à des réseaux de 20 nœuds, a été utilisé sur les réseaux aléatoires et ARPANET. Le second a été généré pour des réseaux comportant 14 nœuds. Le Tableau 4.2 présente la structure de ces deux fichiers, c'est-à-dire le nombre total de trafics ainsi que le nombre de trafics permanents et robustes.

Tableau 4.2 Répartition du trafic dans les fichiers utilisés

Nombre de nœuds	Nombre total de trafics	Nombre de trafics robustes	Nombre de trafics permanents
20	152	68	84
14	131	70	61

4.3 Plan d'expérience

Dans cette section, nous allons définir les techniques que nous utiliserons pour évaluer la performance de notre système, à savoir l'algorithme de routage et d'affectation de longueurs d'onde proposé dans le chapitre précédent. Nous identifierons les facteurs pertinents pour l'évaluation de performance qui sera réalisée à partir des résultats de simulations pratiquées sur un ensemble de réseau et pour diverses demandes de trafics.

4.3.1 Les facteurs de simulations

Avant de réaliser les simulations, nous devons identifier les paramètres susceptibles d'influencer appréciablement les résultats. Notre algorithme comprend deux parties liées, la recherche des chemins et le coloriage du graphe auxiliaire. Nous ne mettrons cependant l'accent que sur les facteurs pouvant influencer l'algorithme

d'affectation des longueurs d'onde (comprenant la construction du graphe auxiliaire et son coloriage). En effet, les paramètres qui pourraient influencer de quelque manière que ce soit la recherche et l'attribution des chemins sont la taille du réseau et la matrice de trafics. Or, ces deux données sont fixes. Les données qui entrent en compte dans l'affectation des longueurs d'onde sont :

- Le nombre de longueurs d'onde disponibles sur chaque fibre. C'est également le nombre de couleurs utilisées pour colorier le graphe auxiliaire ;
- La longueur de la liste taboue ;
- Le nombre d'itérations maximal à faire au cours du coloriage du graphe auxiliaire.

4.3.2 Choix des niveaux des facteurs

Le nombre de longueurs d'onde est le facteur le plus important, car c'est lui qui affecte directement le nombre de trafics bloqués. Théoriquement, l'augmentation du nombre de longueurs d'onde devrait entraîner une diminution du nombre de trafics bloqués. La longueur de la liste taboue, quant à elle, tout comme le nombre maximum d'itérations, devrait également influencer le temps d'exécution de l'algorithme ainsi que la fonction objectif. Ils pourraient également avoir une influence sur le nombre de trafics bloqués.

L'impact des paramètres que nous avons présentés peut être positif ou négatif, dépendamment des valeurs qui leurs sont attribuées. C'est pour cette raison que nous définissons des niveaux pour chacun d'eux, qui vont nous permettre de cerner les effets de chaque paramètre sur le système. Dans notre cas, le niveau des facteurs correspond à des valeurs qu'ils peuvent prendre parmi un ensemble possible relatif à chaque facteur. Le Tableau 4.3 résume les niveaux choisis. Pour chaque facteur, nous avons choisi un symbole afin de faciliter la présentation des résultats. Ainsi, le nombre de longueurs d'onde sera noté λ, la longueur de la liste taboue LT et le nombre d'itérations M.

Tableau 4.3 Niveaux des facteurs

Facteurs		Niveaux					
Noms	Symbole						
Nombre de longueurs d'onde	λ	2	4	8	16	32	64
Longueur de la liste taboue	LT	0	5	10	15	20	25
Nombre maximum d'itérations	M	100	1000	10000	100000	-	-

Le choix des niveaux s'est fait de telle sorte qu'ils couvrent adéquatement l'étendue de variation des facteurs. Nous avons retenu six valeurs pour λ : 2, 4, 8, 16, 32 et 64. En ce qui concerne le paramètre LT, nous avons décidé de commencer par 0 et d'incrémenter à chaque fois de cinq. La valeur 0 affectée à LT, voulant dire que la liste taboue sera toujours vide. Six valeurs ont ainsi été déterminées : 0, 5, 10, 15, 20, 25. Pour M, les valeurs vont de 100 à 100000 en multipliant à chaque fois par 10. Nous considérons que 100 est une bonne valeur de départ et au bout de 100000 itérations, nous devrions obtenir assez d'éléments de comparaison. Le choix de 10 comme facteur multiplicatif permet juste d'uniformiser les valeurs.

4.3.3 Types d'expérience choisies

Une fois le niveau de chaque facteur spécifié, il faut déterminer des combinaisons de ces niveaux afin de réaliser les expériences. Une instance de valeurs de niveaux du triplet (λ, LT, M) correspond à une session du plan d'expérience. Une instance possible serait par exemple (2, 10, 1000), soit 2 longueurs d'onde disponibles par lien, une liste taboue de longueur 10 et un maximum de 1000 itérations. Afin d'étudier l'effet de chaque facteur, nous allons commencer par une expérience « un facteur à la fois ». Dans ce type d'expérience, on fait varier un facteur à la fois tandis que les autres restent constants. De cette manière, on peut isoler l'effet d'un facteur en fonction des résultats obtenus. Dans un premier temps, nous n'analyserons que le nombre de trafics bloqués

dans le scénario de base. Ces tests ne sont faits que sur le réseau pseudo-aléatoire de 20 nœuds et 25 liens. Nous aurons un total de 16 sessions correspondant à la somme du nombre de niveaux de chaque facteur. En effet, si un seul facteur varie, les autres étant maintenus constants, pour chaque facteur nous aurons autant de sessions qu'il y a de niveaux. Cette première étape va donc nous permettre de diminuer le nombre de sessions de tests à faire. Par la suite, nous choisirons d'autres combinaisons de niveaux et nous analyserons le nombre de trafics bloqués au total ainsi que le temps d'exécution des algorithmes ceci, pour tous les réseaux présentés plus haut.

4.3.4 Le comportement aléatoire de l'algorithme

Au cours de l'algorithme, des choix aléatoires sont effectués à plusieurs niveaux. Lors de l'étape de coloriage par exemple, si on a plusieurs mouvements possibles de même coût, l'un d'eux est choisi aléatoirement. Pour trancher dans ces cas là, la fonction *rand()*, qui renvoie un entier pseudo-aléatoire, est utilisée dans le programme. Le choix du mouvement dépend alors du résultat de *rand()*. Pour éviter de faire les mêmes choix à chaque exécution du programme, donc d'avoir la même séquence de chiffres générés par *rand()*, on utilise la fonction *srand()*. Cette fonction a comme argument une « graine » qui permet de générer une nouvelle séquence de nombres à chaque exécution du programme. L'appel à *srand* est fait une fois en début du programme. Il faut noter que toute séquence de nombres est reproductible en donnant la même graine à *srand*.

À cause du fonctionnement de ces deux fonctions, nous devons envisager la possibilité que les résultats obtenus au cours des expériences varient en fonction de la graine aléatoire. Pour cette raison, nous allons faire une série d'exécutions afin d'analyser l'impact de la graine aléatoire sur la performance de l'algorithme. Nous allons tout d'abord analyser l'effet des facteurs présentés plus haut. Ensuite, nous ferons une série de tests sur le scénario de base et sur l'ensemble des scénarios, ceci, pour diverses valeurs de la graine et pour un triplet (λ, LT, M) donné. Nous avons choisi 30 valeurs allant de 1 à 30.

4.4 Expérimentation et tests

Dans cette section, nous allons présenter et analyser les résultats que nous avons obtenus. Dans un premier temps, nous présenterons l'expérience « un facteur à la fois » puis, nous présenterons les autres tests que nous avons effectués.

4.4.1 Expériences « un facteur à la fois »

Pour ces expériences, nous avons fourni la valeur 1 comme graine aléatoire. À ce niveau, nous avons juste implémenté l'algorithme TabouN. Nous avons d'abord fait varier le facteur M avec $\lambda = 2$ et LT = 10. Les Figures 4.5 et 4.6 montrent respectivement le nombre de trafics bloqués pour le scénario sans panne et pour tous les scénarios. Dans les deux cas, en augmentant le nombre d'itérations, le nombre de trafics bloqués diminue. Cependant, à la Figure 4.6, on remarque que sur l'ensemble des scénarios, quand on passe de 10000 à 100000 itérations, le nombre de trafics bloqués passe de 2870 à 2871. Étant donné la grandeur du nombre, on peut considérer ces résultats comme semblables. Dans l'ensemble, le fait de varier M affecte le nombre de trafics bloqués. Mais, les variations observées ne sont pas très importantes.

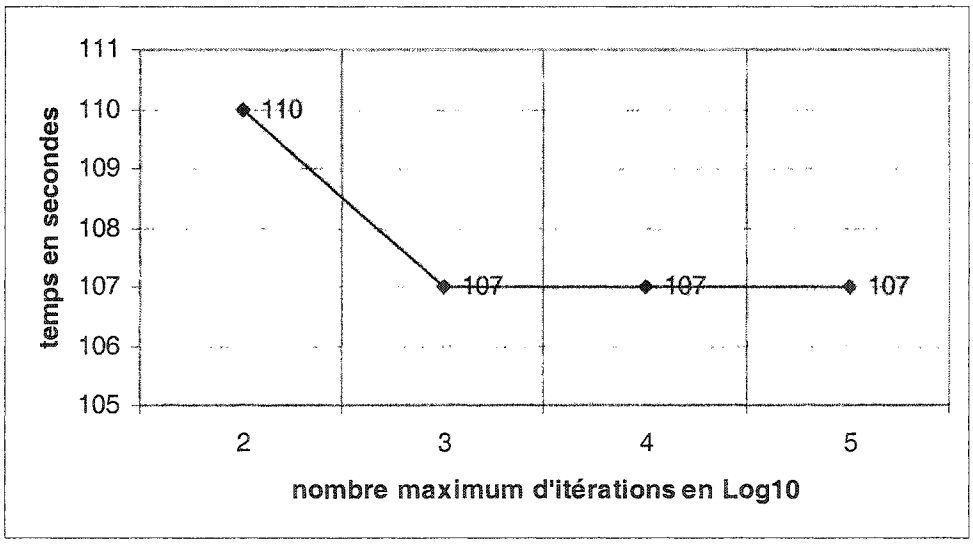


Figure 4.5 Trafics bloqués en fonction du nombre d'itérations pour le scénario de base λ=2, LT = 10 (réseau pseudo-aléatoire)

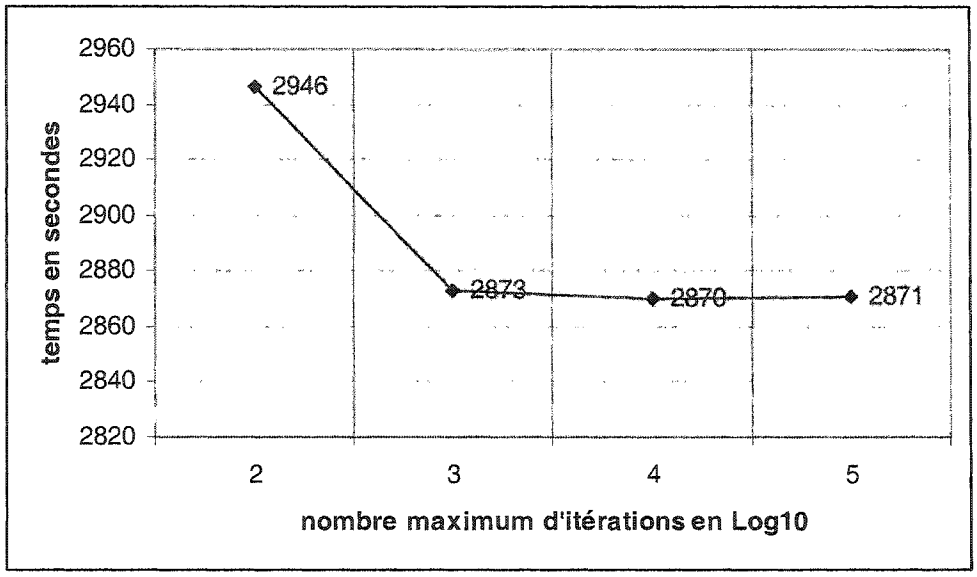


Figure 4.6 Trafics bloqués en fonction du nombre d'itérations pour tous les scénarios λ=2, LT = 10 (réseau pseudo-aléatoire)

Les Figures 4.7 et 4.8 présentent les résultats obtenus en faisant varier le nombre de longueurs d'onde disponibles sur chaque fibre. Cette fois, l'augmentation de λ diminue considérablement le nombre de demandes bloquées.

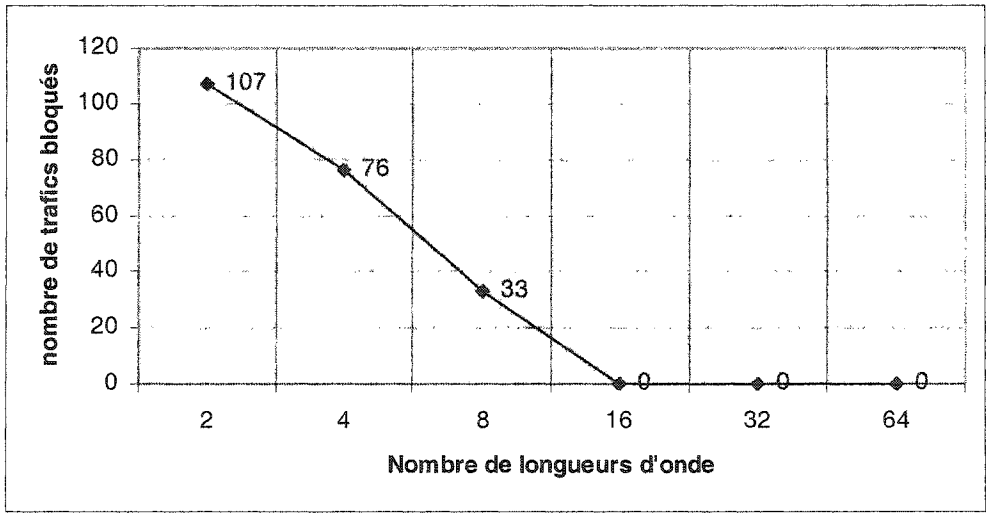


Figure 4.7 Trafics bloqués en fonction de λ pour le scénario sans panne LT=10, M=1000 (réseau pseudo-aléatoire)

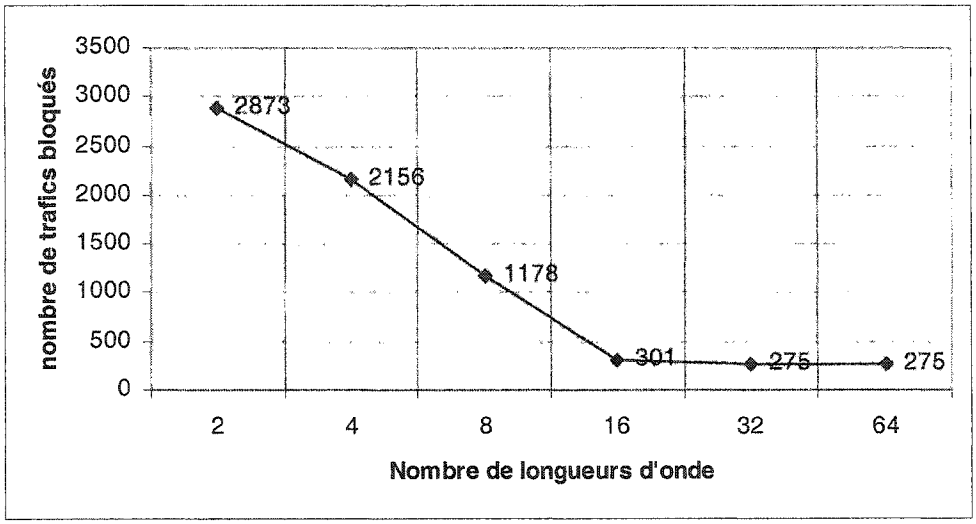


Figure 4.8 Trafics bloqués en fonction de λ pour tous les scénarios LT=10, M=1000 (réseau pseudo-aléatoire)

Lorsque la longueur de la liste taboue varie, le taux de blocage est affecté tout comme dans les cas précédents. Toutefois, en analysant la Figure 4.9, on remarque que, quand LT augmente, le nombre de trafics bloqués ne diminue pas toujours. Ceci, que ce soit pour le scénario de base ou pour le nombre total de trafics bloqués représentés à la Figure 4.10.

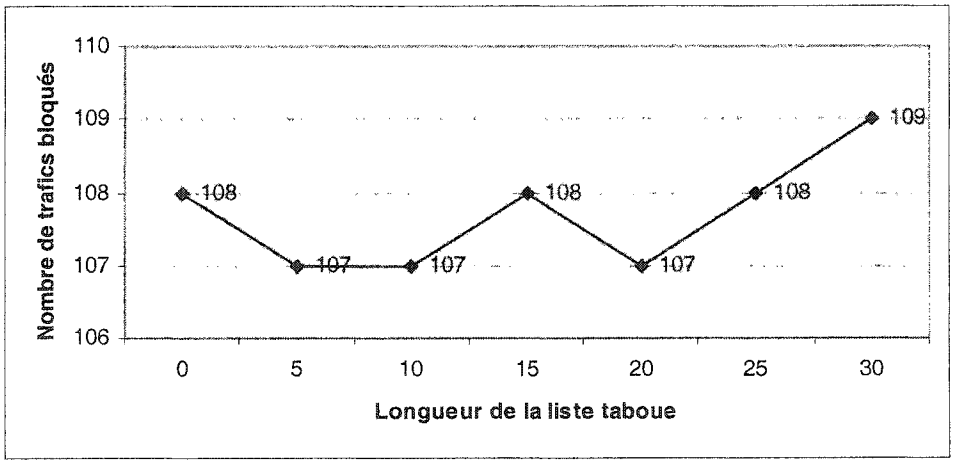


Figure 4.9 Trafics bloqués en fonction de LT pour tous les scénarios λ=2, M=1000 (réseau pseudo-aléatoire)

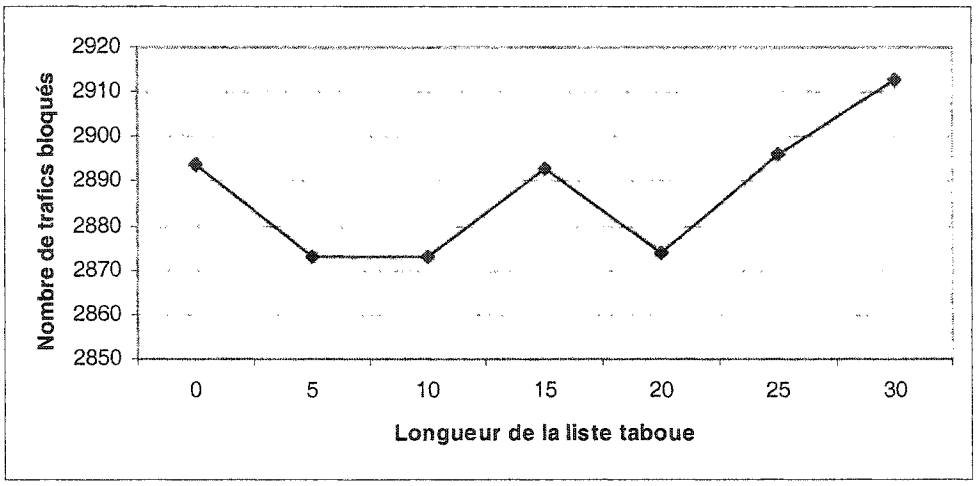


Figure 4.10 Trafics bloqués en fonction de LT pour tous les scénarios : λ=2, M=1000 (réseau pseudo-aléatoire)

Le fait d'augmenter le nombre de longueurs d'onde ou le nombre maximum d'itérations a pour effet de réduire le nombre de trafics bloqués. Cependant, nous avons remarqué que le comportement de l'algorithme TabouN face aux variations de la longueur de la liste taboue n'était pas le même. Pour des expériences, nous allons mettre l'accent sur la variation du nombre de longueurs d'onde car, elle a un effet net sur le taux de blocage. Nous allons également analyser l'effet du nombre d'itérations sur le temps d'exécution. Pour la suite, nous avons décidé de choisir une valeur pour LT et de l'utiliser comme paramètre fixe pour les tests futurs.

Afin de fixer la longueur de la liste taboue, nous avons étudié le nombre total de trafics bloqués pour les valeurs de LT, en faisant varier le nombre d'itérations et pour λ=2. Les résultats sont présentés au Tableau 4.4. En les analysant, on constate que les taux de blocage les plus faibles sont obtenus pour LT = 25 et LT = 15. On remarque que, dans l'ensemble, pour une valeur de LT donnée, plus on augmente le nombre d'itérations, plus le nombre de trafics bloqués diminue. Cette diminution est plus nette pour la valeur 15, c'est pour cette raison que nous avons décidé de garder LT = 15 pour la suite des tests. Cependant, il faut noter que tout ceci ne veut pas dire que 15 soit la valeur optimale pour LT. Il aurait fallu faire des tests plus poussés, faire varier le nombre de longueurs

d'ondes également. Notre but à ce niveau est de réduire le nombre de sessions d'expérience à réaliser dans la suite de notre travail, c'est pour cela que nous avons décidé de fixer LT à une valeur donnée.

Tableau 4.4 Nombre total de trafics bloqués pour LT et M variables : réseau pseudo-aléatoire

Longueur de la liste taboue	Nombre maximum d'itérations			
-	100	1000	10000	100000
5	2923	2873	2871	2874
10	2946	2873	2870	2871
15	2924	2893	2870	2869
20	2921	2874	2874	2870
25	2890	2896	2869	2868

4.4.2 Le temps d'exécution des itérations pour le coloriage

Avant d'étudier le temps d'exécution de l'algorithme et de pousser les tests pour comparer nos résultats, nous avons fait des tests pour analyser le temps d'exécution des itérations relatives au coloriage du graphe auxiliaire. Une fois encore, nous avons fait des sessions d'expérience sur le graphe pseudo-aléatoire de 20 nœuds et 25 liens. Nous avons fait varier le nombre de longueurs d'onde ainsi que le nombre d'itérations avec LT=15. Cette fois, en plus de TabouN, nous avons exécuté Tabou1 afin de comparer les temps de calcul et de voir l'effet du nombre de sommets du graphe auxiliaire sur le temps. Nous avons mesuré le temps écoulé entre le début et la fin de la boucle d'itérations. Il n'inclut ni la construction du graphe auxiliaire ni son initialisation. On serait porté à penser que le temps mesuré sera proportionnel aux itérations. Nous n'avons analysé que le scénario de base qui est le scénario sans panne.

En observant les résultats présentés aux Tableaux 4.5 et 4.6, on remarque que dans l'ensemble, le temps augmente avec le nombre maximum d'itérations. Pour λ=2, 4 et 8, lorsqu'on passe de 100 à 1000 itérations, le temps est multiplié par 3, 4 et 5. Toujours pour ces trois valeurs de λ, de 1000 à 10000 et de 10000 à 100000 itérations, il est multiplié en moyenne par 10, résultat auquel on n'aurait pas pu s'attendre.

Cependant, pour λ=16, 32 et 64, le temps est quasiment le même, quel que soit le nombre d'itérations. Ceci s'explique par le fait que la boucle dont nous mesurons le temps a deux critères d'arrêt : le nombre maximum d'itérations et la fonction coût. Comme nous l'avons mentionné dans le chapitre précédent, les itérations vont s'arrêter soit quand le nombre maximum d'itérations sera atteint, soit quand tous les sommets seront coloriés. Pour le scénario de base, avec TabouN, pour λ=16, 32 ou 64, toutes les demandes sont satisfaites. C'est pour cette raison que, quel que soit le nombre maximum d'itérations, le temps est sensiblement le même.

Tableau 4.5 Temps des itérations pour TabouN

	λ=2	λ=4	λ=8	λ=16	λ=32	λ=64
M=100	0.00	0.01	0.01	0.02	0.04	0.08
M=1000	0.03	0.04	0.05	0.02	0.04	0.09
M=10000	0.34	0.41	0.44	0.02	0.04	0.09
M=100000	3.35	4.11	4.37	0.02	0.04	0.09

Le Tableau 4.6 contient les résultats relatifs à Tabou1. Quand on compare les résultats des Tableaux 4.5 et 4.6, on remarque que les deux algorithmes se comportent de la même manière. Ici également, pour λ=2, 4 et 8, de 1000 à 100000 itérations, le temps est en moyenne multiplié par 10 à chaque étape. Dans l'ensemble, les temps obtenus avec Tabou1 sont inférieurs à ceux obtenus avec TabouN. Ce qui s'explique par la taille du graphe à colorier. En effet, pour le réseau que nous avons testé, le graphe auxiliaire a 194 sommets avec TabouN, tandis que, avec Tabou1, nous construisons un graphe de 152 sommets.

Tableau 4.6 Temps des itérations pour Tabou1

	λ=2	λ=4	λ=8	λ=16	λ=32	λ=64
M=100	0.00	0.00	0.00	0.01	0.04	0.06
M=1000	0.03	0.03	0.04	0.04	0.04	0.08
M=10000	0.25	0.33	0.37	0.27	0.04	0.08
M=100000	2.58	3.24	3.67	2.51	0.04	0.08

4.4.3 Comparaison entre TabouN, Tabou1 et H+

Dans cette section, nous allons comparer les résultats obtenus avec TabouN, Tabou1 et H+. Cette fois ci, en plus du réseau pseudo-aléatoire, nous allons faire les tests sur les réseaux ARPANET et NSFNET présentés aux Figures 4.3 et 4.4. Nous avons utilisé les fichiers de trafics présentés au Tableau 4.2. Nous avons fait varier le nombre de longueurs d'ondes tout en gardant le nombre maximum d'itérations M à 10000 et la longueur LT à 15. Nous avons choisi 10000 itérations parce que cela semble suffisant pour observer l'impact du nombre de longueurs d'ondes. Les Tableaux 4.7 à 4.9 contiennent les résultats obtenus pour ces trois réseaux. À chaque fois, nous avons mesuré le nombre total de trafics bloqués. Le Tableau 4.7 contient les résultats pour le réseau pseudo-aléatoire. Le graphe auxiliaire comporte 194 sommets pour TabouN, contre 152 pour Tabou1. Les deux méthodes que nous proposons donnent de meilleurs résultats que l'algorithme H+. Pour de grandes valeurs de λ, les trois algorithmes donnent les même résultats et le fait de passer de 32 longueurs d'ondes à 64 ne changent pas le nombre total de trafics bloqués. Ceci est dû aux demandes permanentes. En effet, par définition, les demandes permanentes ne peuvent être reroutées en cas de panne, elles sont donc automatiquement bloquées si elles sont touchées par la panne. Quand on regarde le Tableau 4.7, pour λ=32 et λ=64, on a 275 trafics bloqués au total. Ce nombre correspond à la somme, sur tous les scénarios0, des demandes permanentes qui sont automatiquement bloquées dans un scénario.

Tableau 4.7 Nombre total de trafics bloqués pour H+, Tabou1, TabouN (réseau 20 nœuds, 25 liens)

	H+	Tabou1	TabouN
λ= 2	2988	2893	2894
λ= 4	2354	2190	2155
λ= 8	1465	1282	1183
λ= 16	345	337	303
λ= 32	275	275	275
λ= 64	275	275	275

La Figure 4.11 montre les gains de Tabou1 par rapport à H+ et de TabouN par rapport à H+ et Tabou1. On remarque qu'en considérant un seul chemin par trafic et en appliquant une méthode taboue au coloriage de graphes, on gagne de 2 à 12% par rapport à H+ (courbe bleue de la Figure 4.11). Quand on augmente en plus la taille du graphe auxiliaire, TabouN par rapport à Tabou1, on a un gain additionnel qui va jusqu'à 10% pour λ=8.

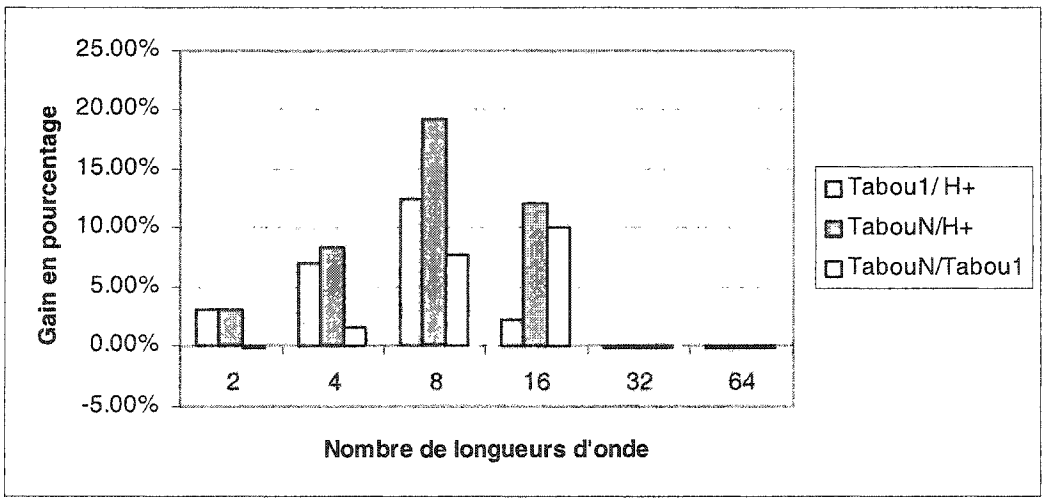


Figure 4.11 Gains pour le réseau pseudo-aléatoire

Le Tableau 4.8 présente les résultats pour le réseau ARPANET. Dans ce cas aussi, Tabou1 et TabouN donnent de meilleurs résultats que H+. Ici, les gains sont nettement

supérieurs à ceux observés précédemment quand on passe de Tabou1 à TabouN. Pour λ=8 par exemple, on gagne près de 28%. Par contre, les gains de Tabou1 par rapport à H+ sont moins importants, de l'ordre de 7%.

Tableau 4.8 Nombre total de trafics bloqués pour H+, Tabou1, TabouN (ARPANET)

	H+	Tabou1	TabouN
λ= 2	3355	3255	3201
λ= 4	2438	2357	2217
λ= 8	1475	1360	1337
λ= 16	384	359	257
λ= 32	237	237	237
λ= 64	237	237	237

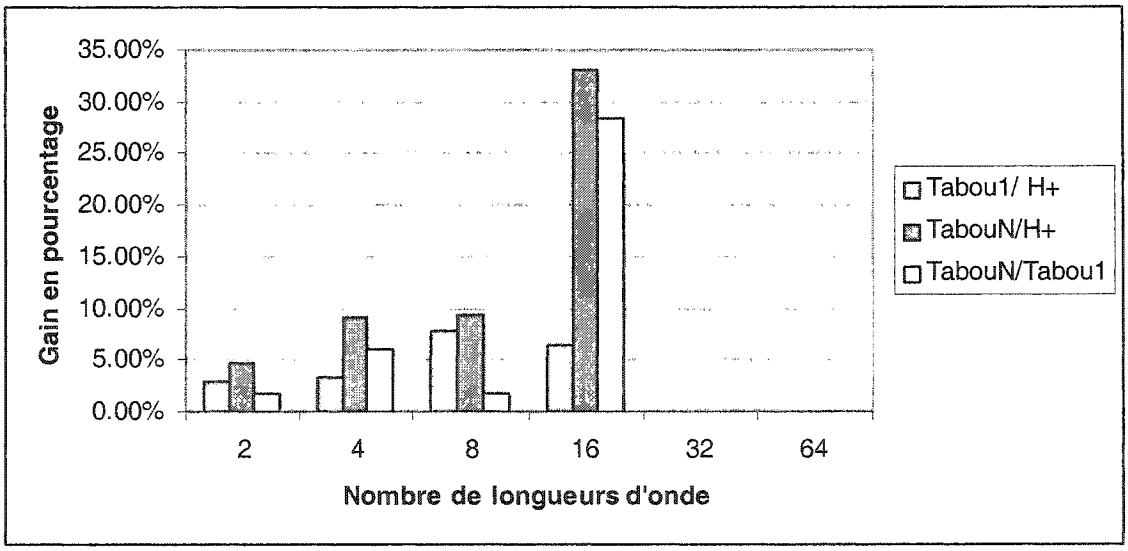


Figure 4.12 Gains pour ARPANET

Pour le réseau NSFNET, les résultats vont dans le même sens que pour les deux autres réseaux. C'est dans ce cas que nous avons les plus gros gains. En effet, pour λ=8, entre H+ et Tabou1 il y a un gain de près de 22% en faveur de Tabou1; pour TabouN par

rapport à Tabou1, on a 20% de gain additionnel, ce qui fait un total de 37% de TabouN par rapport à H+.

Tableau 4.8 Nombre total de trafics bloqués pour H+, Tabou1, TabouN (NSFNET)

	H+	Tabou1	TabouN
λ= 2	1812	1789	1753
λ= 4	1291	1174	1134
λ= 8	652	512	409
λ= 16	133	131	132
λ= 32	131	131	131
λ= 64	131	131	131

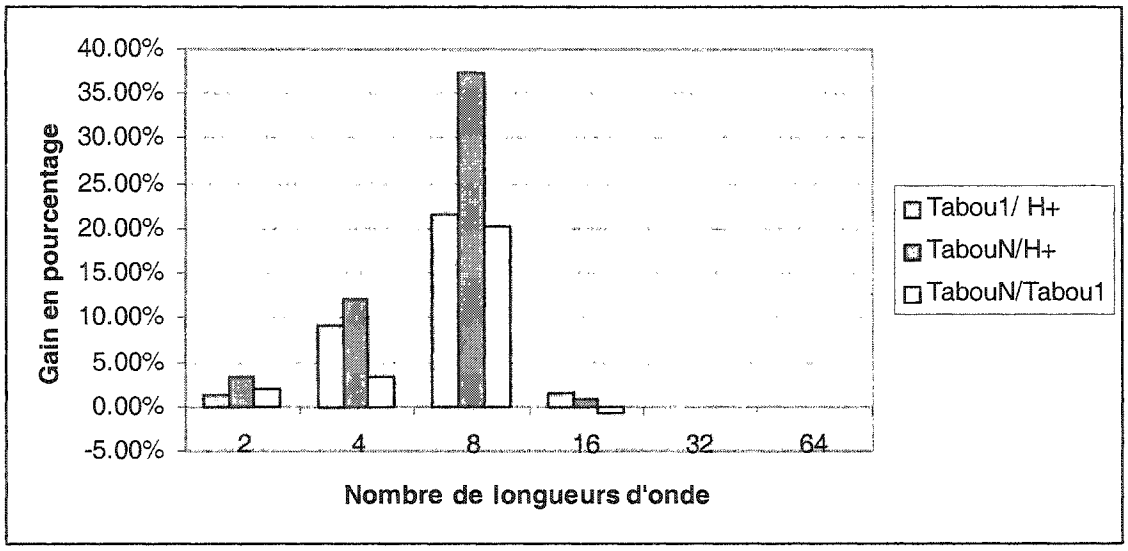


Figure 4.12 Gains pour NSFNET

Pour tous les réseaux que nous avons testés, et pour toutes les matrices de trafics que nous avons utilisées, nous avons pu observer de nouveau que l'augmentation du nombre de longueurs d'onde diminue considérablement le nombre de trafics bloqués.

4.4.4 Temps d'exécution des algorithmes

Un des aspects importants de l'évaluation de performance d'un algorithme est le temps d'exécution. Plus haut dans ce chapitre, nous avons présenté les temps d'exécution

de la section « taboue » de *TabouN*, soit le temps de coloriage du graphe auxiliaire. Dans cette section, nous allons présenter le temps total de *TabouN* et *Tabou1* en fonction du nombre d'itérations maximum et pour plusieurs valeurs de λ. Nous avons utilisé le réseau pseudo-aléatoire et nous avons fixé la longueur de la liste taboue à 15. Les résultats obtenus sont présentés dans les Tableaux 4.9 et 4.10. Dans l'ensemble, on remarque que *TabouN* est plus long que Tabou1. Ce qui s'explique encore une fois par la taille du graphe auxiliaire. Le temps d'exécution augmente non seulement avec le nombre d'itérations mais aussi avec le nombre de longueurs d'ondes. Dans le pire cas, M=10000 et λ=64, pour *TabouN*, on a un temps de 18.22 secondes. C'est un temps très satisfaisant quand on considère la taille du graphe et le nombre de demandes à satisfaire.

Tableau 4.9 Temps d'exécution de TabouN

	λ=2	λ=4	λ=8	λ=16	λ=32	λ=64
100	0.68	1.34	2.22	2.37	2.42	2.60
1000	1.43	1.93	3.80	5.55	5.67	6.24
10000	8.56	7.28	10.32	11.92	13.40	18.22

Tableau 4.10 Temps d'exécution de Tabou1

	λ=2	λ=4	λ=8	λ=16	λ=32	λ=64
100	0.63	1.13	2.25	3.80	3.80	1.96
1000	1.22	1.54	2.85	4.37	4.43	4.93
10000	7.48	5.49	8.18	10.14	10.75	14.13

Le Tableau 4.11 contient le temps d'exécution, en secondes, de l'algorithme H+. Comparé aux résultats présentés plus haut, H+ est beaucoup plus rapide. La Figure 4.13 présente une comparaison entre les temps de calcul de H+, Tabou1 et TabouN. Les résultats de H+ ne dépendent que du nombre de longueurs d'onde. C'est pour cette raison que, pour la Figure 4.13, nous avons donné la même valeur au temps pour H+, quel que

soit le nombre d'itérations. Nous avons comparé les temps de calcul pour λ=4. Dans le pire cas, c'est-à-dire pour *TabouN* et 10000 itérations, la différence de temps est de 6.85 secondes, ce qui n'est pas énorme. De plus, quand on regarde la Figure 4.11, représentant les gains de *TabouN* par rapport à H+, on remarque que pour cette valeur de λ, on a un gain de près de 9% sur le nombre de trafics bloqués au total.

Tableau 4.11 Temps d'exécution de H+

λ=2	λ=4	λ=8	λ=16	λ=32	λ=64
0.18	0.43	0.31	0.19	0.17	0.17

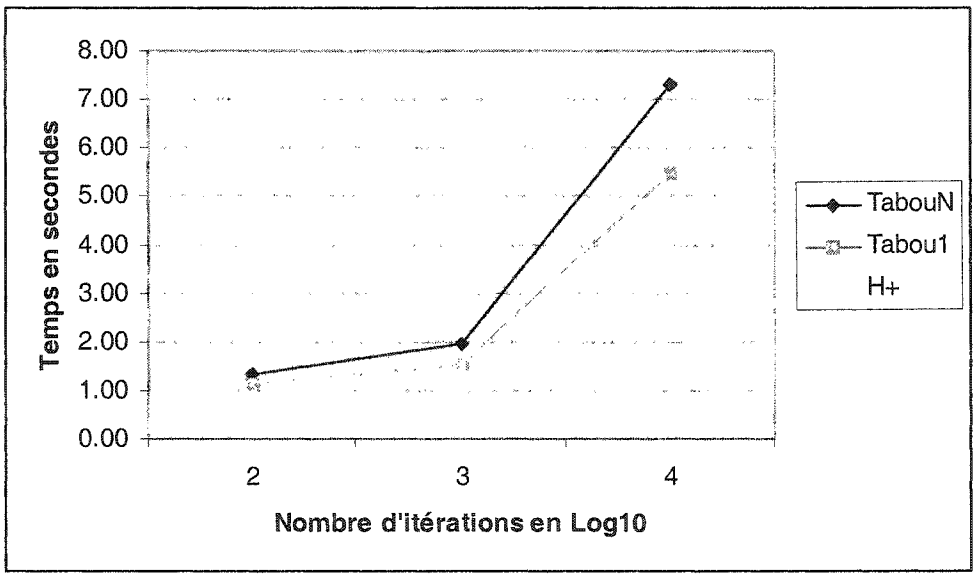


Figure 4.13 Temps de calcul en fonction des itérations pour H+, Tabou1 et TabouN : λ=4, LT=15 (réseau pseudo-aléatoire)

4.4.5 Effet de la variation de la graine aléatoire

Nous avons analysé le comportement aléatoire de l'algorithme *TabouN*, en lançant plusieurs exécutions du programme, pour différentes valeurs de la graine aléatoire. Nous avons fait trente sessions de tests, pour trois valeurs de λ : 2, 4 et 8. À

chaque fois, nous avons relevé le nombre total de trafics bloqués et le nombre de trafics bloqués pour le scénario sans panne. Les Tableaux 4.12 et 4.13 présentent les valeurs maximales et minimales obtenues pour chaque longueur d'onde, ainsi que la moyenne et l'écart type. Quand on analyse la Figure 4.14, qui présente le nombre de trafics bloqués en fonction de la graine, on remarque que, dépendamment de la valeur de la graine, le nombre de trafics bloqués pour le scénario sans panne varie. Ceci reflète bien le caractère aléatoire de notre méthode. Les résultats présentés à cette Figure correspondent à λ=8.

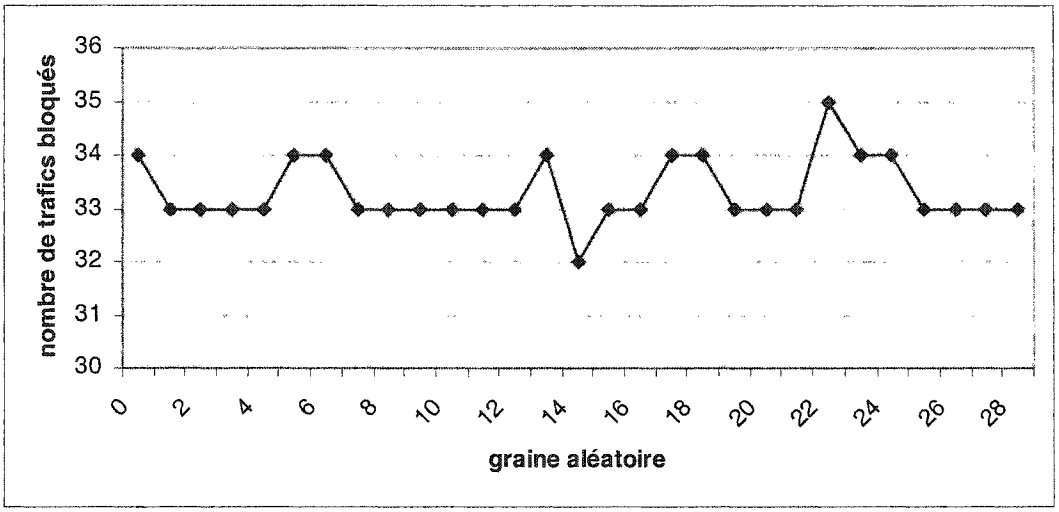


Figure 4.14 Trafics bloqués pour le scénario de base en fonction de la graine aléatoire : λ=8, LT=15, M=10000 (réseau pseudo-aléatoire)

Tableau 4.12 Moyenne et écart type (scénario de base)

	λ=2	λ=4	λ=8
Minimum	107	75	32
Maximum	108	77	35
Moyenne	107.10	75.86	33.31
Écart type	0.31	0.44	0.60

La Figure 4.15 quant à elle, correspond au nombre total de trafics bloqués, pour λ=8 également. On peut aussi y observer le comportement aléatoire de l'algorithme. Si on compare les résultats des Tableaux 4.12 et 4.13, on remarque que pour le scénario de base, les valeurs ne varient pas énormément, le minimum et le maximum sont assez proches. Ceci se reflète d'ailleurs dans la valeur de l'écart type.

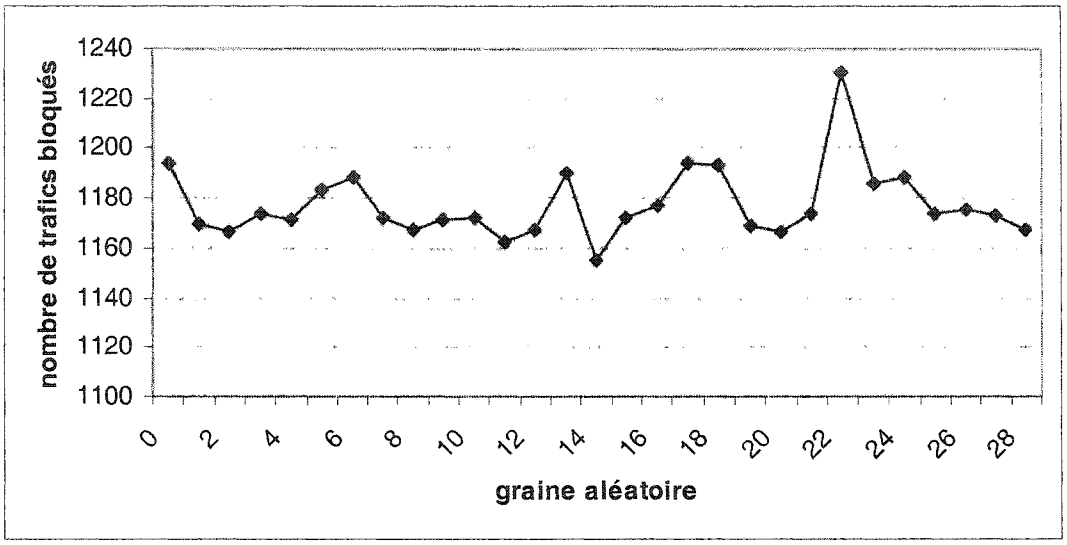


Figure 4.15 Nombre total de trafics bloqués en fonction de la graine aléatoire : λ=8, LT=15, M=10000 (réseau pseudo-aléatoire)

Tableau 4.13 Moyenne et écart type (tous les scénarios)

	λ=2	λ=4	λ=8
Minimum	2866	2131	1155
Maximum	2898	2170	1230
Moyenne	2874.28	2151.83	1177.24
Écart type	8.84	8.61	14.30

4.4.6 Comparaison avec CPLEX

Pour tester encore plus la performance de nos algorithmes, nous nous sommes intéressés à la valeur trouvée par le logiciel CPLEX comme borne inférieure de la solution optimale. Nous ne considérons que le scénario sans panne. Nous avons fait les comparaisons pour trois longueurs d'onde : 2, 4 et 8. Le Tableau 4.14 résume les résultats obtenus avec CPLEX, H+, Tabou1 et TabouN, pour le réseau aléatoire. Nous avons utilisé LT=15 et M=10000 pour Tabou1 et TabouN. Nous remarquons que pour λ=2 et λ=4, TabouN obtient la même valeur que CPLEX. Dans l'ensemble, les résultats obtenus par nos deux algorithmes ont très proches de ceux trouvés avec CPLEX.

	H+	Tabou1	TabouN	CPLEX
λ=2	112	108	107	107
λ=4	84	78	75	75
λ=8	44	37	34	32

CHAPITRE V
CONLUSION

Dans ce mémoire, nous avons traité du problème de routage et d'affectation de longueurs d'onde dans les réseaux optiques. Nous avons considéré le cas particulier de l'approche statique du problème. Nous avons proposé une méthode générique de résolution utilisant la recherche taboue. Dans ce chapitre, nous allons faire une synthèse des travaux réalisés. Par la suite, nous présenterons les limites des méthodes proposées, puis, nous donnerons un aperçu des travaux futurs.

5.1 Synthèse des travaux

Le problème de routage optique revient à trouver un chemin et une longueur d'onde pour les trafics d'un réseau en évitant que deux trafics traversant le même lien aient la même longueur d'onde. Nous avons traité de ce sujet en considérant la contrainte de continuité de longueurs d'onde dans le réseau ainsi que deux types de demandes : robustes et permanentes. Le but de notre travail était de maximiser le nombre de trafics acheminés sur l'ensemble des scénarios possibles. Un scénario est relié à l'état du réseau. En effet, on associe un scénario à chaque panne de lien simple possible dans le réseau. Le scénario de base est celui où il n'y a aucune panne dans le réseau.

Nous avons privilégié une approche séparée du problème. C'est-à-dire que nous avons traité le routage d'une part et l'affectation des longueurs d'onde d'autre part. Nous avons présenté en détail l'algorithme *TabouP* que nous proposons. Avec cette méthode, pour commencer, nous avons considéré une méthode de routage basée sur les plus courts chemins qui utilise le nombre de sauts comme métrique. Pour chaque trafic, nous avons considéré un nombre P de plus courts chemins possibles parmi lesquels le chemin sur lequel va être routé le trafic sera choisi. Par la suite, l'affectation des longueurs d'onde se fait en utilisant une heuristique de coloriage de graphes, le graphe à colorier étant un graphe auxiliaire contenant les chemins retenus pour chaque trafic.

Nous avons implémenté deux versions de la méthode proposée, la différence entre les deux provenant de la façon dont le graphe auxiliaire est construit. En effet, dans la première, *TabouN*, nous considérons tous les plus courts chemins pour la construction du graphe auxiliaire tandis que, pour la seconde, *Tabou1*, nous ne considérons qu'un seul chemin par trafic. Nous avons étudié les effets des variations de quatre facteurs sur le nombre de trafics bloqués soit : le nombre de longueurs d'onde disponibles sur chaque fibre, la longueur de la liste taboue, le nombre maximum d'itérations pour le coloriage et la graine aléatoire.

Pour évaluer la performance de *TabouN* et *Tabou1*, nous avons comparé nos résultats à ceux obtenus avec l'algorithme H+. Nous avons effectué les comparaisons sur trois réseaux. Dans tous les cas, les résultats obtenus étaient meilleurs que ceux de H+. De plus, dans la plupart des cas, pour le réseau aléatoire que nous avons considéré, nous avons trouvé des résultats identiques à ceux de la borne inférieure calculée avec CPLEX, ceci pour le scénario de base. Les temps d'exécution que nous avons relevés sont dans l'ensemble supérieurs à ceux de H+, mais demeurent toutefois raisonnables.

5.2 Limitations des travaux

Dans l'ensemble, nous avons obtenu de bons résultats. Cependant, notre travail comporte certaines limites. Tout d'abord, afin de simplifier le problème, nous n'avons pas tenu compte de la capacité des liens. En faisant cela, les méthodes proposées peuvent fournir des solutions qui routeraient plus de trafics sur un lien qu'il ne peut supporter.

Nous avons considéré des réseaux avec des liens monofibres, c'est-à-dire ne comportant qu'une seule fibre alors que, les réseaux actuels sont de plus en plus multifibres. En effet, à cause du coût associé à l'installation des liens de fibres optiques, les opérateurs privilégient le fait de mettre plusieurs fibres sur un même lien. En mettant plus de fibres sur un lien, le nombre de longueurs d'onde nécessaire pour router un maximum de trafics devrait être inférieur à ce que nous avons obtenu. Nous avons choisi de ne pas utiliser le critère d'aspiration. Ceci pourrait avoir une influence sur la performance de nos algorithmes.

5.3 Travaux futurs

Pour résoudre le problème de routage, nous avons considéré tous les plus courts chemins pour chaque trafic. Un point intéressant serait d'analyser le comportement de nos algorithmes dans le cas où nous considérerions également des chemins sous-optimaux. De cette manière, il y aurait plus de possibilité de chemins pour chaque trafic et donc plus de choix pour rerouter les trafics touchés par les pannes. Il serait également intéressant d'analyser l'effet de la proportion des demandes robustes sur la performance de l'algorithme. Il serait également intéressant d'étudier l'effet que produirait l'introduction de convertisseurs de longueurs d'onde dans les réseaux. Dans ce cas, le problème d'optimisation que nous résolvons rejoindra alors un problème de design qui est le placement des convertisseurs. Un objectif possible pourrait être de trouver les positions des convertisseurs afin de minimiser leur nombre, dans le but de maximiser toujours le nombre de demandes satisfaites étant donné un certain nombre de longueurs d'onde.

BIBLIOGRAPHIE

[1] Banerjee D., Mukherjee B., "A Practical Approach for Routing and Wavelength Assignment in Large Wavelength-Routed Optical Networks", IEEE Journal on Selected Areas in Communications, Vol. 14, No 5, June 1996.

[2] Poppe F., De Neve H., Petit G.H., "Constrained Shortest Path First Algorithm for Lambda-Switched mesh Optical Networks with Logical Overlay Och/SP Rings", IEEE Workshop on High Performance Switching and Routing, 29-31 May 2001, pp. 150 –154.

[3] Manohar P., Manjunath D., Shevgaonkar R. K., "A Practical Approach for Routing and Wavelength Assignment in Large Wavelength-Routed Optical Networks", IEEE Communications Letters, Vol. 6, No. 5, May 2002.

[4] Stern T., Krishna B., "Multiwavelength Optical Networks", Addison-Wesley, 2000.

[5] Modiano E., Narula-Tam A., "Designing Survivable Networks using Effective routing and Wavelength Assignment (RWA)", Optical Fiber Communication Conference and Exhibit, OFC 2001, Vol. 2, 2001, pp. TuG5-1 -TuG5-3.

[6] Baskiotis N., Perennes S., Rivano H., "Dimensionnement Heuristique des Réseaux Optiques WDM Multifibres par Arrondi Aléatoire de Multiflot", Rapport de recherche Projet MASCOTTE, Université Nice Sofia Antipolis, mars 2002.

[7] Zhang Y., Taira K., Takagi H., Das S. K., "An Efficient heuristic for Routing and Wavelength Assignment in Optical WDM Networks", IEEE International on Communications, ICC 2002, Vol. 5 , 28 April-2 May 2002.

[8] Ramaswami R., Sivarajan K. N., "Routing and Wavelength Assignment in All-Optical Networks", IEEE/ACM Transactions on Networking, Vol. 3, No. 5, October 1995.

[9] Limal E., Stubkjaer K. E., "An algorithm for link restoration of wavelength routing optical networks", IEEE International on Communications, ICC 1999, Vol. 3 , 6-10 June 1999.

[10] Somani A. K., Azigoglu M., "Wavelength Assignment Algorithms for Wavelength Routed Interconnection of LANs", Journal of Ligthwave Technology, Vol. 18, No. 12, December 2000.

[11] Chen B., Wang J., "Efficient Routing and Wavelength Assignment for Multicast in WDM Networks", IEEE Journal on Selected Areas in Communications, Vol. 20, No. 1, January 2002.

[12] Mokhtar A., Azigoglu M., "Adaptive Wavelength Routing in All-Optical Networks", IEEE/ACM Transactions on Networking, Vol. 6, No. 2, April 1998.

[13] Bourouha M. A., Bataineh M., Guizani M., "Advances in Optical Switching and Networking: Past, Present, and Future.", SoutheastCon, Proceedings IEEE , 5-7 April 2002.

[14] Beauquier B., Bermond J-C., Gargano L., Hell P., Perennes S., Vacarro U., "Graph Problems Arising from Wavelength-Routing in All-Optical Networks", Rapport de recherché de l'INRIA-Sofia Antipolis, Mai 1997.

[15] Prins C., "Algorithmes de graphes", Eyrolles, 1994.

[16] Oulaï D., "Routage et Affectation de Longueurs D'onde dans les Réseaux Optiques WDM", Mémoire de maîtrise, École Polytechnique de Montréal, Décembre 2002.

[17] Mewanou S. R., "Algorithmes de routage dynamique dans les Réseaux Optiques WDM", Mémoire de maîtrise, École Polytechnique de Montréal, Janvier 2003.

[18] Inkret R., Mikac B., Podnar I., "A Heuristic Approach to Wavelength Assignment in All-Optical Networks", Electrotechnical Conference, MELECON 98., 9th Mediterranean, Vol.2 , 18-20 May 1998

[19] Zhemin D., Hamdi M., "A Simple Routing and Wavelength Assignment Algorithm using the Blocking Island Technique for All-Optical Networks", IEEE International Conference on Communications, ICC 2002, Vol.5 , 28 April-2 May 2002.

[20] Mokhtar A., Azigoglu M., "Adaptive Technique for Routing and Wavelength Assignment in All-Optical WANs", IEEE 39th Midwest symposium on Circuits and Systems, Vol.3 , 18-21 August 1996.

[21] Zhang Z., Acampora A. S., "A Heuristic Wavelength Assignment Algorithm for Multihop WDM Networks with Wavelength Routing and Wavelength Re-Use ", IEEE/ACM Transactions on Networking, Vol. 3, No, 3, June 1995.

[22] Narula-Tam A., Lin P. J., Modiano E., "Efficient Routing and Wavelength Assignment for Reconfigurable WDM Networks", IEEE Journal on Selected Areas in Communications, Vol. 20, No. 1, January 2002.

[23] Peng W., Wei C., "Distributed Wavelength Assignment Protocols with Priority for WDM All-Optical Networks", Ninth International Conference on Computer Communications and Networks, 2000. Proceedings., 16-18 October 2000.

[24] Thiagarajan S., Somani A. K., "An Efficient Algorithm for Optimal wavelength-Routed Networks with Arbitrary Topologies", INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings. IEEE , Vol.2 , 21-25 March 1999, pp. 916-923.

[25] Qin H., Liu Z., Zhang S., Wen A., "Routing and Wavelength Assignment Based on Genetic Algorithm", IEEE Communications Letters, Vol. 6, No. 10, October 2002, pp. 455-457.

[26] Zhang X., Qiao C., "Wavelength Assignment for Dynamic Traffic in Multi-fiber WDM Networks", 7th International Conference on Computer Communications and Networks, 1998. Proceedings., 12-15 Oct. 1998, pp. 479 –485.

[27] Zhou B., Mouftah H. T., "Adaptive Least Loaded Routing for Multi-fiber WDM Networks Using Approximate Congestions Information", IEEE International Conference on Communications, ICC 2002., Vol. 5 , 28 April-2 May 2002, pp. 2745 –2749.

[28] Pierre S., Couture M., "Télécommunications et Transmission de Données", Eyrolles, 1992.

[29] Gondran M., Minoux M., "Graphes et Algorithmes", Eyrolles, 1995.

[30] Krishnaswamy R. M., Sivarajan K. N., "Algorithms for Routing and Wavelength Assignment Based on Solutions of LP-Relaxations", IEEE Communications Letters, Vol. 5, No. 10, October 2001, pp. 435-437.

[31] Ozdaglar A. E., Bertsekas D. P., "Routing and Wavelength Assignment in Optical Networks", IEEE/ACM Transactions on Networking, Vol. 11, No. 2, April 2003, pp. 259-272.

[32] Rouskas G. N., "Routing and Wavelength Assignment in Optical WDM Networks", Wiley Encyclopedia of Telecommunications, John Wiley & Sons, 2001.

[33] Shifeng L., Jun T., Guanqun G., "Routing and Wavelength Assignment in all Optical Networks to Establish Survivable Lightpaths", TENCON '02. Proceedings. 2002, IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering , Vol.2 , 28-31 October 2002, pp. 1193 –1196.

[34] Chu X., Li B., Sohraby K., Zhang Z., "Routing and Wavelength Assignment Issues in the Presence of Wavelength Conversion for All-Optical Networks", IEEE Global Telecommunications Conference, GLOBECOM '02. , Vol.3 , 17-21 November 2002, pp. 2787 –2791.

[35] Ramamurthy S., Mukherjee B., "Survivable WDM Mesh Networks, Part I-Protection", Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '99, Proceedings. IEEE, Vol. 2, 21-25 March 1999, pp. 744 –751.

[36] Ramamurthy S., Sahasrabuddhe L., Mukherjee B., "Survivable WDM Mesh Networks", Journal of Lightwave Technology, Vol. 21, No. 4, April 2003, pp. 870-883.

[37] Modiano E., Narula-Tam A., "Survivable Lightpath Routing: A New Approach to the Design of WDM-Based Networks", IEEE Journal on Selected Areas I Communications, Vol. 20., No. 4, May 2002, pp. 800-809.

[38] Lee T., Lee K., Park S., "Optimal Routing and Wavelength Assignment in WDM Ring Networks", IEEE Journal on Selected Areas I Communications, Vol. 18, No. 10, October 2000, pp. 2146-2154

[39] Zang H., Ou C. S., Mukherjee B., "Path-Protection Routing and Wavelength Assignment (RWA) in WDM Mesh Networks Under Duct-Layer Constraints", IEEE/ACM Transactions on Networking, Vol. 11, No. 2, April 2003, pp. 248-258.

[40] Li B., Chu X., Sohraby K., "Routing and Wavelength Assignment vs. Wavelength Converter Placement in All-Optical Networks", IEEE Communications Magazine, Vol. 41, Issue: 8 , August 2003, pp. S22 -S28.

[41] Coudert D., Rivano H., "Routage optique dans les réseaux WDM multifibres avec conversion partielle", *AlgoTel'02*, Mèze, France, Mai 2002, pp. 17-24.

[42] Kuchar A., "All-Optical Routing – Progress and Challenges", . Proceedings of the 2002 4th International Conference on Transparent Optical Networks, 2002 , Vol. 1, 21-25 April 2002, pp. 49 –50.

[43] Choi J. S., Golmie N., Lapeyrere F., Mouveaux F., Su D., "A Functional Classification of Routing and Wavelength Schemes in DWDM Networks: Static Case", NRC 2000, New Jersey, USA, 14-15 April 2000