

TABLE DE MATIÈRES

Introduction	3
Chapitre 1 : Modèle support	4
Chapitre 2 : Modèle du trafic et routage des connexions	6
2.1- MODÈLE DE GÉNÉRATION DU TRAFIC	6
2.2- ROUTAGE DES DEMANDES DE CONNEXION	7
2.3- IMPACT DU CHOIX D'UNE FONCTION DE COÛT	10
2.3.1- Fonctions de coût proposés	10
2.3.2- Simulation et résultats	11
2.3.3- Conclusions	13
Chapitre 3 : Problème de reconfiguration	14
3.1- MOTIVATION POUR LE RE-ROUTAGE DES CONNEXIONS	14
3.2- CLASSES DE RECONFIGURATION	14
3.3- SOLUTIONS DE BASE	15
3.3.1- 1 ^{er} classe d'heuristiques de reconfiguration	16
3.3.1.1- 1 ^{er} heuristique	16
A- Algorithme formel	16
B- Simulation et résultats	17
3.3.1.2- 2 ^{eme} heuristique	23
A- Algorithme formel	24
B- Simulation et résultats	25
3.3.2- 2 ^{eme} classe d'heuristiques de reconfiguration	28
3.3.2.1- 1 ^{er} heuristique	29
A- Simulation et résultats	30
B- Améliorations proposés	32
C- Simulation et résultats	33
3.3.2.2- 1 ^{er} heuristique	36
Chapitre 4 : Problème de protection	37
4.1- PROBLÈME DE PROTECTION DES CONNEXIONS	37
4.2- PROTECTION À BASE DE RESSOURCES DÉDIÉES	38
4.2.1- Protection dédiée des chemins	39
4.2.2- Protection partagée des chemins	40
4.2.3- Simulation et résultats	41
4.2.4- Protection dédiée et partagée des liens	41
4.2.5- Simulation et résultats	43
4.2.6- Conclusions	46
Chapitre 8 : Conclusions et perspectives	47
Chapitre 9 : Bibliographie	49

TABLE DES FIGURES :

Figure 1.1 : Exemple d'un réseau optique	4
Figure 1.2 : Représentation d'un modèle en couches d'un réseau optique	5
Figure 1.3 : Représentation d'une couche.....	5
Figure 2.1 : Exemple d'un réseau avec des poids sur les nœuds	6
Figure 2.2 : Allure de l'évolution de la charge totale appliquée sur le réseau au cours du temps.....	7
Figure 2.3 : Réseau Cost239	11
Figure 2.4 : Charge appliquée sur le réseau pour les fonctions de coût.....	11
Figure 2.5 : Longueur moyenne des chemins empruntés pour les fonctions de coût.....	12
Figure 2.6 : Charge appliquée sur le lien le plus congestionné pour les fonctions de coût.....	12
Figure 3.1 : la charge appliquée sur le réseau avec une fonction de coût Diamètre.....	18
Figure 3.2 : la charge sur le lien le plus congestionné avec une fonction de coût Diamètre.....	18
Figure 3.3: Longueur moyenne des chemins avec une fonction de coût Diamètre.....	18
Figure 3.4 : Nombre de reconfigurations enregistrées tout au long de la simulation.....	19
Figure 3.5 : Charge appliquée sur le réseau avec un fonction de coût Parabole.....	19
Figure 3.6 : Longueur moyenne des chemins avec une fonction de coût Parabole.....	20
Figure 3.7 : Charge appliquée sur le réseau avec une fonction de coût Exponentiel.....	20
Figure 3.8 : La charge sur le lien le plus congestionné avec une fonction de coût Exponentiel.....	21
Figure 3.9 : Longueur moyenne des chemins avec une fonction de coût Exponentiel.....	21
Figure 3.10 : Charge globale appliquée sur le réseau.....	22
Figure 3.11 : Longueur moyenne des chemins empruntés.....	22
Figure 3.12 : Charge appliquée sur le lien le plus congestionné.....	23
Figure 3.13 : Charge globale appliquée sur le réseau avec ou sans reconfiguration.....	25
Figure 3.14 : Longueur moyenne des chemins empruntés avec ou sans reconfiguration.....	26
Figure 3.15 : Charge appliquée sur le lien le plus congestionné avec ou sans reconfiguration.....	26
Figure 3.16 : Exemple de choix de chemin avec une fonction parabolique.....	27
Figure 3.17 : Inconvénient de la première heuristique.....	27
Figure 3.18 : Nombre de reconfigurations effectuées par H1 et H2.....	28
Figure 3.19 : exemple illustratif de l'avantage de reconfiguration par plusieurs connexions (1).....	29
Figure 3.20 : exemple illustratif de l'avantage de reconfiguration par plusieurs connexions (2).....	29
Figure 3.21 : Charge appliquée sur le réseau.....	30
Figure 3.22 : longueur moyenne des chemins empruntés.....	30
Figure 3.23 : Charge appliquée sur le lien le plus congestionné.....	31
Figure 3.24 : nombre de reconfigurations cumulés tout au long de la simulation.....	31
Figure 3.25 : nombre de reconfigurations cumulés tout au long de la simulation pour H1.....	32
Figure 3.26 : cas de reconfiguration validée.....	33
Figure 3.27 : Evolution de la charge appliquée sur le réseau.....	34
Figure 3.28 : nombre de reconfigurations cumulés tout au long de la simulation pour H1.....	35
Figure 3.29 : reconfigurations cumulés et charge moyenne appliquée en fonction du seuil.....	35
Figure 3.30 : charge totale appliquée sur le réseau pour plusieurs reconfigurations par étape	36
Figure 4.1 : exemple illustratif de faisabilité des techniques de protection.....	38
Figure 4.2 : Evolution de la charge appliquée sur le réseau.....	41
Figure 4.3 : protection de chemin à travers la protection de ses liens.....	41
Figure 4.4 : Evolution de la charge appliquée sur le réseau.....	43
Figure 4.5 : charge appliquée sur le lien le plus congestionné.....	44
Figure 4.6 : Cumul du nombre de demandes rejetés.....	44
Figure 4.7 : Comparaison de la charge appliquée pour la protection partagée liens et chemins.....	45
Figure 4.8 : charge appliquée sur réseau sans et avec reconfiguration.....	45
Figure 4.9 : charge appliquée sur le lien le plus congestionné du réseau sans et avec reconfiguration.....	46

INTRODUCTION

Depuis une vingtaine d'années, et encore actuellement, les opérateurs de réseaux d'infrastructures intercontinentaux, trans-continentaux, voire métropolitains privilégient les systèmes de communications optiques utilisant les technologies de multiplexage en longueurs d'onde (**WDM** pour **Wavelength Division Multiplexing**) pour leur bande passante sans équivalent. Pourtant, du point de vue d'un opérateur les investissements nécessaires sont tels que le déploiement de ce type de réseaux ne se fera pas sans garantie d'efficacité suffisante.

En effet, l'optimisation des réseaux optiques **WDM** est essentielle à son développement ; les coûts de déploiement d'un réseau optique d'infrastructure sont gigantesques et les opérateurs sont intéressés par l'optimisation de la taille des liens installés et de la complexité des routeurs présents dans les noeuds du réseau.

Dans ce contexte, l'objectif de mon stage de master est d'étudier deux problèmes reconnus si importants et intéressants à analyser : problème de reconfiguration et problème de protection. Le travail consistera à distinguer des solutions existantes, les adapter au modèle support de notre travail et les évaluer à base de critères de performances à définir.

Je commencerais par introduire le contexte et le modèle support de tout le travail au niveau du premier chapitre. Ensuite, je m'intéresserais à présenter le modèle du trafic, l'algorithme de base servant au routage des connexions tout en étudiant l'impact du choix d'une fonction de coût sur l'efficacité d'utilisation des ressources disponibles sur un réseau. Dans le chapitre trois, j'exposerais le problème de reconfiguration appliqué à notre modèle support : contextes du problème, motivations, travail existant, solutions proposées, implémentation, simulations et résultats. Le chapitre quatre portera sur le problème de la protection des connexion : contextes du problème, motivations, travail existant, solutions proposées et implémentation, simulations et résultats. Et je finirais le travail par illustrer les conclusions et exposer quelques pistes de perspectives.

Chapitre 1

MODELE SUPPORT

Un réseau WDM est généralement représenté [1] comme suit :

- $G(V, E, F_s, W)$: où V est l'ensemble des nœuds du réseau, E est l'ensemble des liens bidirectionnels et W est l'ensemble des longueurs d'ondes disponibles par fibre. Supposons que l'ensemble de L.O soit le même sur chaque fibre, exemple : $\{\lambda_1, \lambda_2, \dots, \lambda_w\}$. Pour chaque lien: $ij \in E$, F_{ij} représente les fibres existantes sur ce lien. Ainsi ij est composé par $|F_{ij}|$ fibres unidirectionnelles et $F_s = \{F_{ij}\}$.

Exemple de réseau WDM :

Soit le réseau suivant : $G(V, E, F_s, W)$:

- V : $\{A, B, C, D\}$
- E : $\{A \rightarrow B, A \rightarrow C, B \rightarrow C, D \rightarrow B, C \rightarrow D\}$
- W : 3.
- F_s : $\{F_{AB}, F_{BA}, F_{BC}, F_{CB}, F_{AC}, F_{CA}, F_{BD}, F_{DB}, F_{CD}, F_{DA}\}$ on prend le cas où on a deux fibres dans chaque sens sur un lien donné.

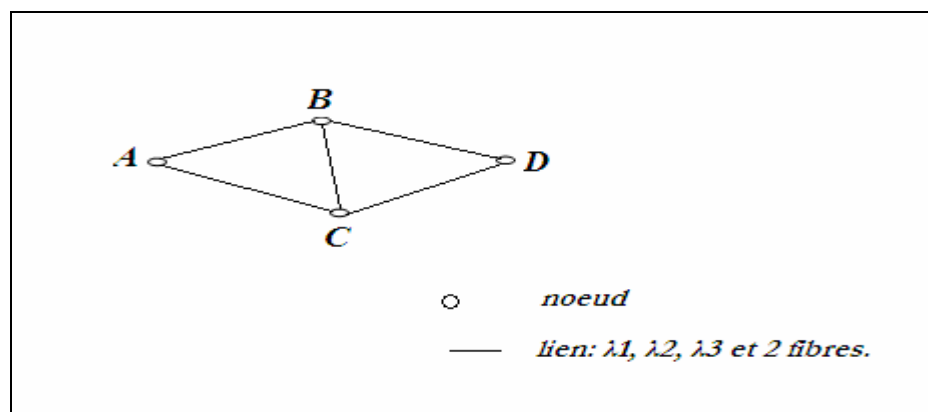


Figure 1.1: exemple d'un réseau optique

Le routage d'une demande de connexion sur un réseau WDM revient à trouver un chemin et une longueur d'onde disponible tout au long de ce chemin : **contrainte de continuité de longueur d'onde**. Ainsi le problème de routage de demande de connexion est divisé en deux sous problèmes à savoir : le routage et l'attribution de longueur d'onde exigeants tous les deux à être résolus pour mettre en place la connexion. Une autre modélisation plus simple, plus claire et qui nous permet de combiner le routage et l'attribution de longueur d'onde est la suivante : « *Modèle du graphe à niveaux* ».

Appliqué à l'exemple du réseau précédent, le **modèle du graphe à niveaux** revient à ça :

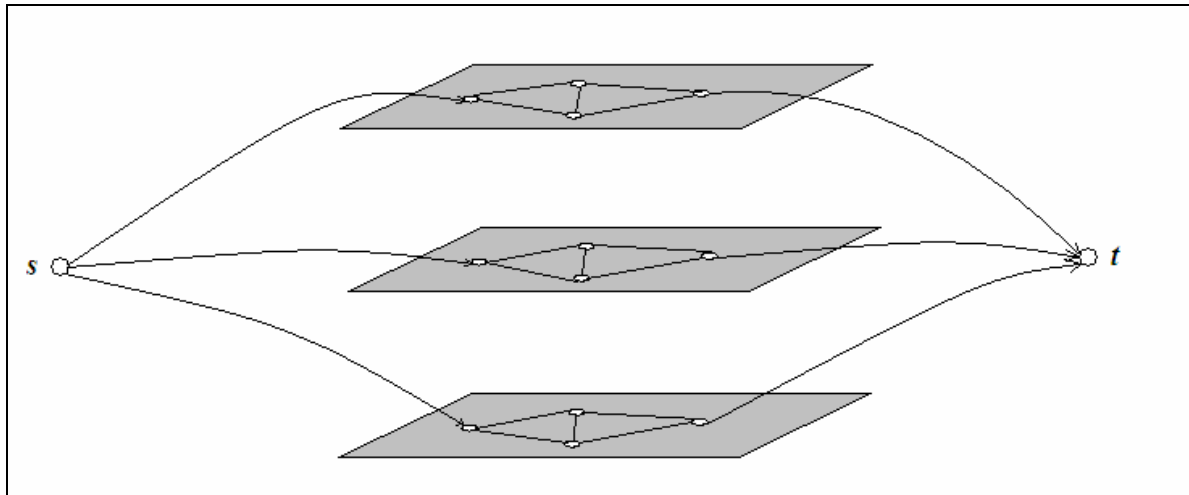


Figure 1.2 : Représentation en modèle en couches d'un réseau optique

Ce modèle est décomposé en $|W|$ couches, chacune correspond à une longueur d'onde, et au niveau de chaque couche une réplique de la topologie physique est effectuée, ou en d'autres termes un graphe avec une représentation plus simple est obtenu : en effet, ce graphe peut être représenté simplement par $G(V, E, c)$:

Avec

- V : l'ensemble des nœuds.
- E : l'ensemble des liens bidirectionnels sur le graphe.
- $c : \{c_{ij}, ij \in E\}$ est la capacité disponible sur le lien « ij » (correspond au nombre de fibres disponibles sur ce lien).

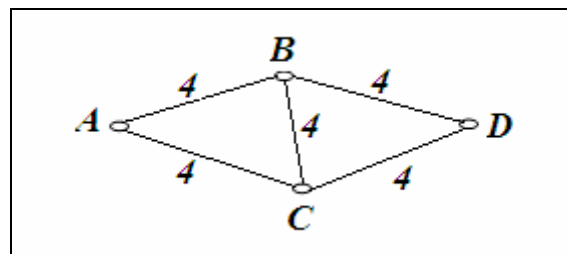


Figure 1.3 : Représentation d'une couche

Ainsi, router une demande de connexion entre deux nœuds du graphe de la figure 2.1 revient simplement à trouver un chemin, suivant une certaine politique de routage, sur l'un des niveaux du graphe tant qu'il y a bien sûr suffisamment de capacité. En un seul coup, on aura alors une route valide et une longueur d'onde (celle de la couche), mais il est à noter que le problème sera un peu plus complexe en temps de calcul suite à une taille plus grande du réseau (plusieurs couches et pour chacune un réseau).

Durant tout le travail qui va suivre, on ne va plus s'intéresser au problème d'attribution de longueur d'onde, tout notre effort portera essentiellement à assurer le routage des demandes de connexion sur l'une des couches (un graphe simple).

Chapitre 2

MODELE DU TRAFIC ET ROUTAGE DES CONNEXIONS

2.1- Modèle de génération du trafic :

Généralement, le trafic utilisé pour réaliser les simulations est un trafic aléatoire et ceux qui essayent de mettre en place des scénarios de trafic se rapprochant de la réalité se basent essentiellement sur le fait que l'évolution du trafic est cyclique [3] et [4] (réseaux de cœur). De toute façon le modèle de trafic choisi n'affectera pas beaucoup nos propositions pour le problème de reconfiguration : en effet les simulations qu'on compte réaliser seront à base des mêmes scénarios pour différentes heuristiques implémentées : ainsi on comparera la robustesse et l'efficacité des heuristiques proposées l'une par rapport à l'autre sous les mêmes charges et conditions. Toutefois, la génération du trafic sera en quelques sortes paramétrable à travers:

- Le nombre de demandes générées par cycle.
- Le nombre de demandes générées par un nœud : nous utilisons un modèle de trafic gravitationnel, en effet le nombre de demandes générés entre deux nœuds du réseau est proportionnel au produit des poids assignés à ces nœuds. *Exemple* : sur la figure suivante les trafics les plus importants seront entre Vienne/Paris et Vienne/Prague et ce à cause des poids qu'on les a attribué.
- La fréquence des changements à savoir la génération ou la sortie des demandes.

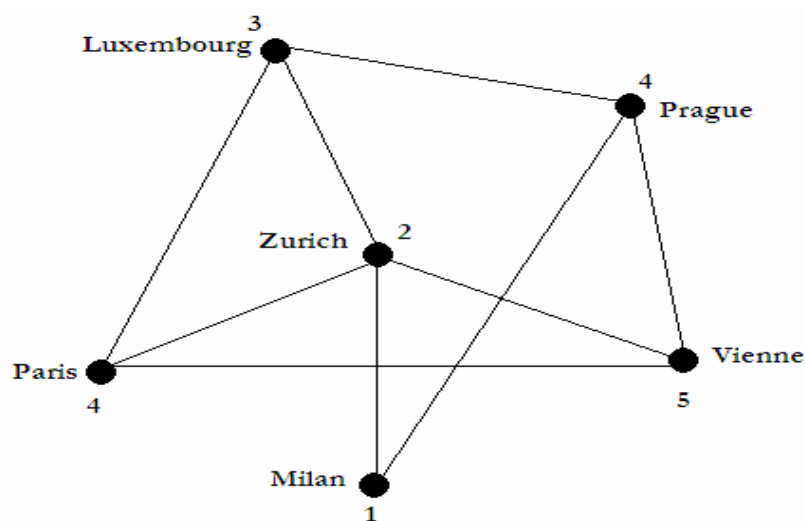


Figure 2.1 : exemple d'un réseau avec des poids sur les nœuds

L'évolution proposée de notre trafic a trois périodes distinctes à savoir :

- 1- Une augmentation à partir d'un seuil initiale (car généralement le réseau doit avoir un certain trafic en circulation permanente) vers un certain seuil, cette partie durera une période à définir du cycle.
- 2- Des fluctuations autour du seuil de trafic atteint pour une période à définir du cycle : ces fluctuations résultent nécessairement des entrées et sorties des connexions.
- 3- Une descente vers le seuil initial.

La première et la troisième contiendront impérativement des retraits des connexions et non seulement des générations de demandes de connexion.

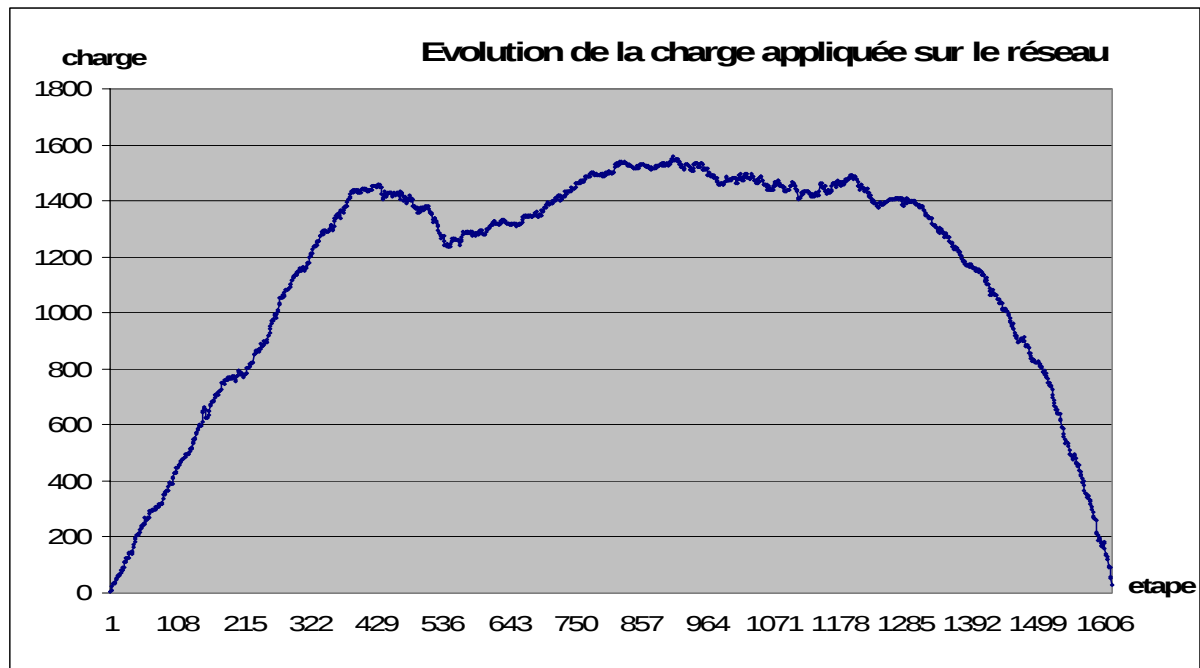


Figure 2.2 : allure de l'évolution de nombre de connexions sur le réseau au cours du temps

2.2- Routage des demandes de connexion :

Le routage d'une demande de connexion consiste à trouver un chemin ayant suffisamment d'unités de capacité pour accueillir cette demande.

Pour notre cas le routage des demandes de connexions est fait à base de la programmation linéaire. Pour un premier temps, une formulation [2] à base de chemins a été introduite afin de trouver l'ensemble de routes les plus convenable sous des conditions bien précises du réseau (coûts et capacités disponibles sur les liens).

On considère un graphe (réseau) $G: G = (V, A)$ constitué de noeuds V et d'un ensemble d'arcs A . Sur ce réseau, on considérera un **coût** et une **capacité** relative à chacun des arcs (La capacité peut varier évidemment d'un arc à un autre).

Sur un tel graphe, des scénarios de trafic peuvent être produits suivant des critères bien définis et notre objectif est d'assurer le routage des requêtes qui les composent.

Les variables utilisés pour formuler le problème sont les suivantes :

- $T = \{0, 1, 2, \dots, T_{\max}\}$: l'ensemble des étapes de la simulation.
- $R = \{R_1, R_2, R_3, \dots, R_{\max}\}$ l'ensemble des requêtes des demande de connexion du scénario, chaque demande de connexion est identifié par une source et une destination.
- $V = \{V_1, V_2, V_3, \dots, V_n\}$ l'ensemble des nœuds du graphe.
- $ch_{ij} = \{ch_1, \dots, ch_p\}$, avec p : le nombre de chemins possible entre les deux noeuds i et j .

- $CH = \{ch_{12}, ch_{13}, \dots, ch_{ij}\}$, avec $i, j = 1, 2, \dots, n$ représentant les nœuds du graphe; CH représente l'ensemble des ensembles de ch_{ij} qui définissent les chemins existants entre tous les paires de nœuds i et j (pour $i = j$ on a $ch_{ij} = \emptyset$).
- $x_{req, ch} = \{0, 1\}$; variable binaire indiquant l'utilisation ou non d'un chemin par une demande cette variable de décision permettra la distinction du chemin le moins coûteux emprunté par la demande d .
- $R_E(t)$ = l'ensemble des requêtes générées à une étape t .
- A : Ensemble des arcs du réseau.
- $ch_{ij}^a : \{ch_1, \dots, ch_k\}$, avec p : le nombre de chemins possible entre les deux nœuds i et j et passant par l'arc a .
- C^{ch} : coût relatif à l'utilisation d'un chemin « ch ».
- c^a : capacité disponible sur un arc « a ».

Nous présentons ci-dessous le programme linéaire relatif à une étape unique. Ainsi, pour résoudre notre problème d'origine ce programme doit être reproduit à chaque étape de la simulation :

$$\text{Objectif:} \quad \min \sum_{req \in R_E(t)} \sum_{ch \in ch_{ij}} x_{req, ch} * C^{ch}$$

- ij : les nœuds source et destination de la requête « req ».

Sous les contraintes :

$$\forall req \in R_E(t), \forall a \in A \quad \sum_{req \in R_E(t)} \sum_{ch \in ch_{ij}^a} x_{req, ch} * req_{size} \leq c^a \quad (1.1)$$

$$\forall req \in R_E(t) \quad \sum_{ch \in ch_{ij}} x_{req, ch} = 1 \quad (1.2)$$

$$\forall req, ch \quad x_{req, ch} = \{0, 1\}$$

Ce programme linéaire assurera l'attribution optimale des chemins aux demandes de connexion pour l'instance courante du réseau, bien évidemment une mise à jour par étape est effectuée pour chaque chemin utilisé et la génération d'un ensemble de requête entre 2 nœuds du réseau est permise.

L'objectif étant la minimisation du coût de la solution de routage globale, la contrainte n° 1.1 assure le respect de la capacité disponible sur chaque lien du réseau et la contrainte n° 1.2 assure qu'une demande ne peut être routée qu'à travers un seul des chemins possibles entre les nœuds source et destination de cette demande.

Pour mener à bien la phase du routage des demandes de connexion, il faut en premier lieu extraire tous (ou une partie) les chemins entre chaque paire de nœuds et affecter à chacun un coût initial.

En premier lieu on va travailler avec une charge aisément routable et on essaiera à être loin de la saturation complète des liens et sous de telles conditions observer les résultats.

Algorithme formel proposé :

Algorithme1 Algorithme de routage à base des chemins

Entrée : $G = (V, E)$, $Duree_simulation$, $D = \{d_1, d_2, \dots, d_m\}$ (ensemble des demandes de connexions Triées dans un ordre croissant suivant leurs temps d'arrivée), $nombre_demandes$

Sortie : Taux de réussite

Variables locales :

$connexions_finies_etape$: vecteur servant à stocker les connexion finies à une étape spécifiée.

demandes_entrantes_etape : vecteur servant à stocker les demandes de connexion générées à une étape spécifiée.

pl_cplex : Programme linéaire pour résoudre le problème.

1. **Pour** chaque « étape » allant de 1 à *Duree_simulation* **faire**
2. *connexions_finies_etape* $\leftarrow$ *trouver_requetes_sortantes(etape)*
3. **Si** (*connexions_finies_etape* est non vide) **faire**
4. **Pour** chaque « connexion » de l'ensemble *connexions_finies_etape* **faire**
5. *Mettre_à_jour_capacités_et_coûts_des_liens_du_chemin_de(connexion)*
6. *Mettre_à_jour_coût_du_chemin(connexion)*
7. **Fin Pour**
8. **Fin Si**
- 9.
10. *demandes_entrantes_etape* $\leftarrow$ *trouver_demandes_entrantes(etape)*
11. **Si** (*demandes_entrantes_etape* $\neq \emptyset$) **faire**
12. *créer_variables_cplex_des_demandes(demandes_entrantes_etape, pl_cplex)*
13. *Ajouter_contraintes_de_capacités(pl_cplex)*
14. *Ajouter_objectif(pl_cplex)*
15. *resoudre(pl_cplex)*
16. *sauvegarder_résultats(requetes_entrantes_etape)*
17. *Mettre_à_jour_capacités_et_coûts_des_liens_du_chemin_de(requetes_entrantes_etape)*
18. *Mettre_à_jour_coût_des_chemins(requetes_entrantes_etape)*
19. **Fin Si**
20. **Fin Pour**

Description détaillée de l'algorithme proposé :

Après avoir extrait le scénario des demandes de connexions, un certain nombre de traitements doivent être appliqués pour mettre en œuvre l'algorithme de routage proposé:

- ✚ Mettre à jour les capacités et les coûts sur les liens suite à la sortie des requêtes: Suite à la sortie d'une ou plusieurs requêtes, des unités de capacité vont être libérées sur les liens qu'elles empruntaient. Ainsi une opération de mise à jour de coût et de capacité est exigée sur ces liens.
- ✚ Créer les variables du modèle au niveau du programme linéaire: Pour résoudre le problème, il faut tout d'abord créer les différentes variables de décision. Les variables de décision dans notre exemple représentent l'utilisation ou non d'un chemin par une demande et il faut créer toutes les variables relatives aux requêtes et leurs chemins associés possibles.
- ✚ L'ajout des contraintes: Un ajout de contraintes est réalisé à chaque étape pour ne pas dépasser les capacités disponibles sur les liens à une étape donnée.
- ✚ Ajouter l'objectif au programme linéaire: Spécifier comme objectif du programme linéaire: la minimisation du coût global de la solution de routage.
- ✚ Résoudre le programme linéaire: A ce niveau une solution (si elle existe) est renvoyée par le **Solveur Cplex** utilisé. A partir de cette solution, les demandes de connexion peuvent avoir leurs chemins spécifiques et ils seront ainsi mis sur le réseau.

Nous allons maintenant utiliser notre programme linéaire de routage avec différentes fonctions de coût et nous allons observer l'aptitude de chacune de ces fonctions à utiliser les ressources disponibles sur le réseau et ce à travers trois critères de performance qui sont :

- La charge appliquée sur le réseau : le nombre d'unités de capacité utilisées sur le réseau à chaque étape de la simulation.
- La longueur moyenne des chemins utilisés par les connexions à chaque étape de la simulation.

- La charge sur le lien le plus congestionné du réseau à chaque étape de la simulation.

2.3 - Impact du choix d'une fonction de coût :

Dans cette section nous allons illustrer l'impact du choix de la fonction de coût sur l'utilisation des ressources d'un réseau de notre choix et avec notre modèle de trafic. Mais en premier lieu nous allons exposer quelque fonctions de coût qu'on pourra utiliser :

2.3.1- Fonctions de coût :

a- Première fonction de coût :

- Attribuer en premier lieu des coûts égaux à 1 à tous les liens tant qu'il reste encore de la capacité disponible.
- S'il ne reste plus de capacité sur un lien on lui attribue un coût égal au diamètre du réseau ou à la somme de l'ensemble de ses liens : $|E|$.

Cette fonction de coût cherche d'abord à router les demandes de connexions le long d'un plus court chemin. Lorsqu'il n'y a plus de capacité disponible le long d'un plus court chemin, alors on cherche des chemins plus long en utilisant les capacités encore disponibles. Cette solution n'est pas préventive et peut entraîner la saturation de certains liens du réseau en laissant d'autres liens sous-utilisés. Dès lors des connexions peuvent rapidement être refusées.

b- Deuxième fonction de coût :

Contrairement à la première fonction de coût, ici on ne route plus la demande par le plus court chemin en nombre de sauts mais par celui qui nous coûte le moins cher en terme de congestion : toutes les fonctions qui vont être présentées se basent sur le principe de pénalisation suite à l'utilisation de la capacité sur les liens (ce qui revient à utiliser tant que c'est possible les chemins les moins congestionnés).

$$- \text{cout}(e) = f(\text{capacité_utilisée})$$

La fonction du coût est définie comme suit :

La fonction « $f(\text{capacité_utilisée})$ » est une fonction de coût croissante dont la vitesse de croissance peut suivre une exponentielle, une parabole, ou même une discrétisation de l'intervalle de la capacité utilisée. Nous utiliserons les notations suivantes vont être utilisées :

- capacite : capacité totale sur un lien.
- capacite_restante : capacité restante sur un lien.
- capacite_utilisee : capacité utilisée sur un lien.

$$\text{Avec : } \text{capacite} = \text{capacite_restante} + \text{capacite_utilisee}.$$

Fonctions proposées :

- o Fonction à croissance exponentielle :

$$f : \text{capacite_utilisee} \rightarrow e^{\text{capacite_utilisee}}$$

- o Fonction à croissance parabolique :

$$f : \text{capacite_utilisee} \rightarrow \text{capacite_utilisee}^2$$

- o Fonction discrétisée :

$$cost(e) = \begin{cases} 1 & \text{si } congestion \leq 0,5 \\ 10 & \text{si } 0,5 < congestion \leq 0,8 \\ 1000 & \text{pour les autres cas} \end{cases}$$

Les valeurs 1, 10 et 1000 sont choisies aléatoirement et la congestion est définie comme suit :

$$congestion = capacite_utilisee / capacite \quad (\text{Varie clairement entre 0 et 1}).$$

Le réseau utilisé pour les simulations est le « **cost239** » :

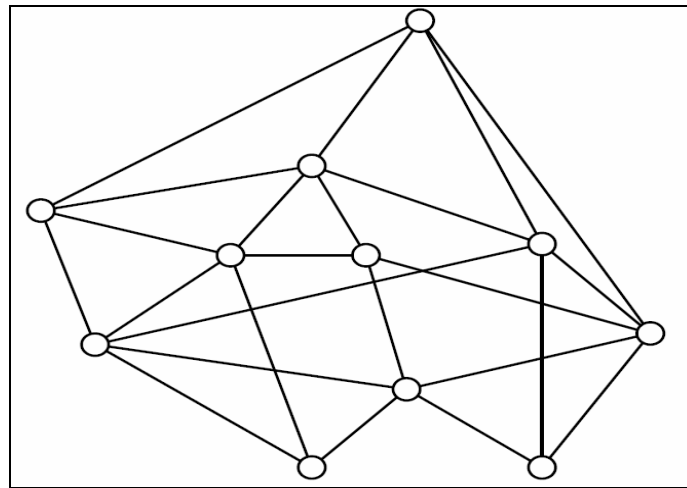


Figure 2.3 : réseau cost239

2.3.2- Simulation et résultats :

Disposant de 48 unités de capacités sur chacun des liens de ce réseau on a obtenu les résultats suivants pour nos simulations.

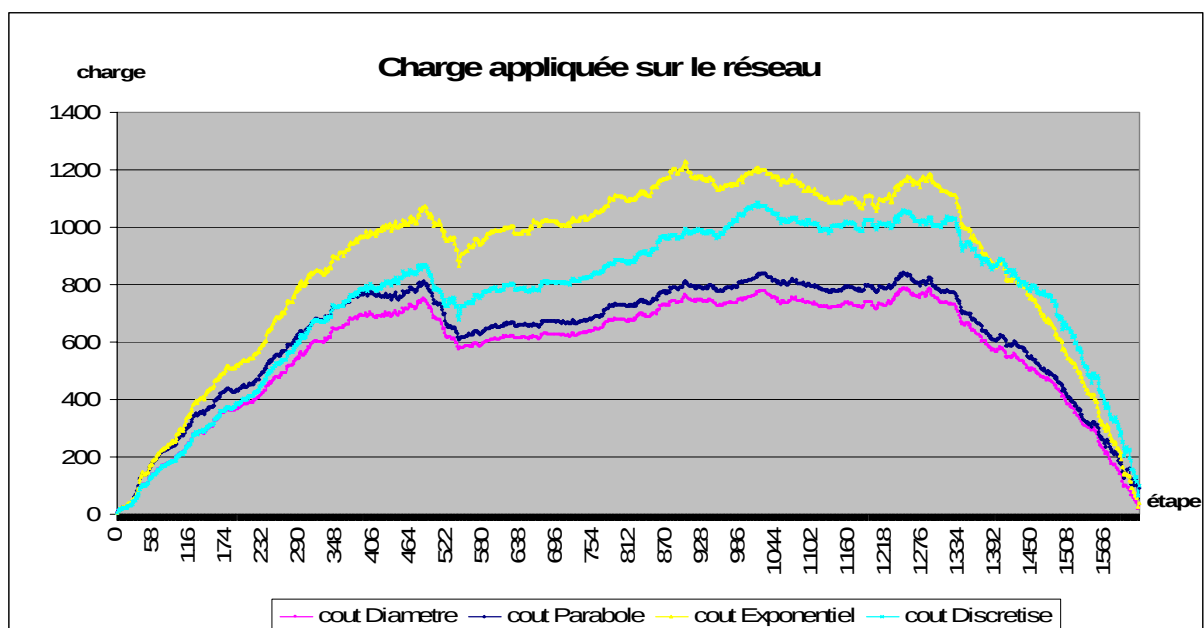


Figure 2.4 : charge appliquée sur le réseau pour les fonctions de coût

D'après la *figure 2.4*, il est bien clair que la fonction de « *coût Diamètre* » applique le minimum de charge totale sur le réseau car c'est une fonction qui tend à concentrer le trafic sur certains liens du réseau seulement (les chemins les plus courts en termes de sauts). A l'opposé, la fonction de coût exponentielle augmente la charge totale et diminue la charge max d'un lien. Cette fonction distribue le trafic sur l'ensemble des liens du réseau et préserve de la capacité sur le plus de liens possibles (maximise la capacité restante). La fonction parabolique et la fonction discrétisé semblent être de bons compromis.

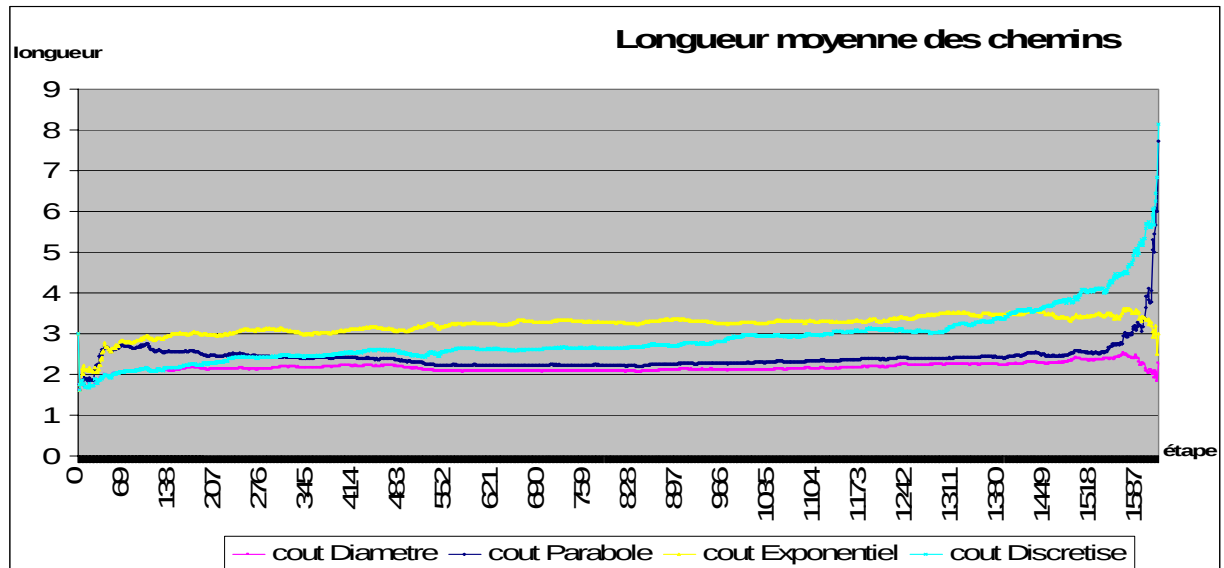


Figure 2.5 : longueur moyenne des chemins choisis sur le réseau pour les fonctions de coût

De même sur la *figure 2.5*, on remarque que la fonction de « *coût Diamètre* » dépasse les autres fonctions en performances : en effet les chemins empruntés par cette fonction admettent les plus petites valeurs moyennes. Tandis qu'on emprunte les plus longs chemins en utilisant la fonction de « *coût exponentiel* », une fonction qui essaye toujours de disperser le trafic sur le réseau pour assurer un certain équilibre au niveau de l'utilisation des ressources du réseau.

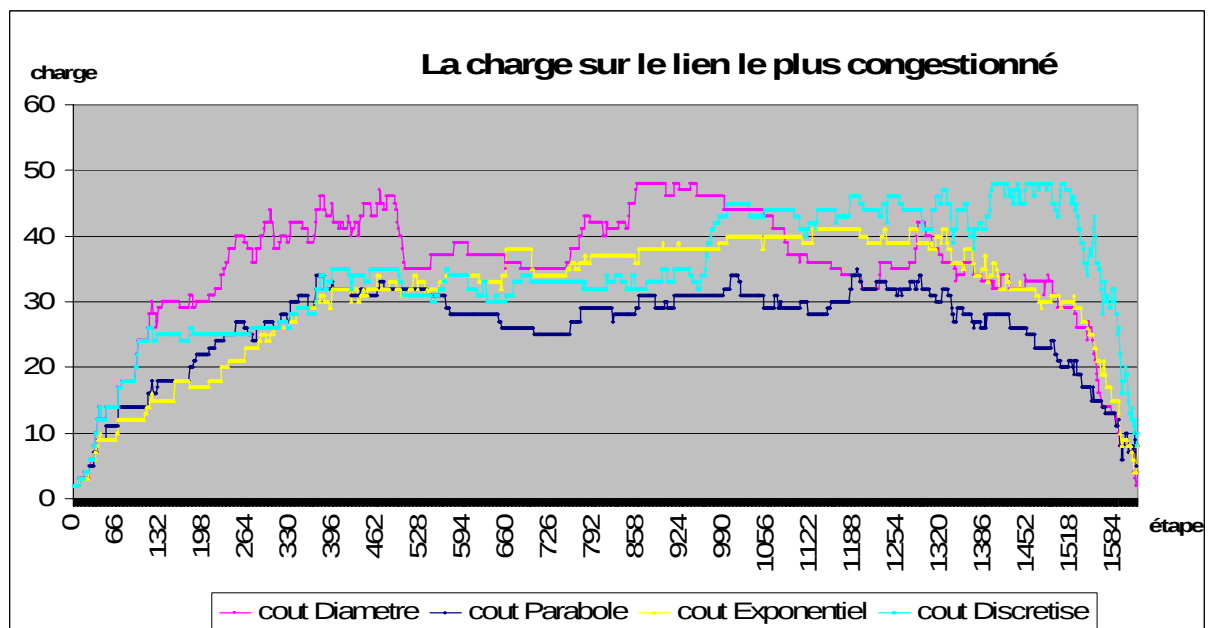


Figure 3.6 : charge appliquée sur le lien le plus congestionné du réseau pour les fonctions de coût

Notre interprétation sur la *figure 2.6* peut être décomposé en deux parties :

- Une avant l'étape n° 550 : pendant cette période, on remarque l'efficacité des fonctions les plus pénalisantes : en effet, les plus basses valeurs sur le lien le plus congestionné sont observés avec les fonctions de « *coût Exponentiel* » et de « *coût Parabole* », ce qui confirme la disponibilité des ressources et ainsi plus de survie pour le réseau. Pour cette première phase, la fonction de discrétisation semble en premier lieu utiliser excessivement un certain nombre de liens mais ensuite elle se récupère et ce à cause du premier seuil 0.5 (premier intervalle : $0 \rightarrow 0.5$) duquel elle se distingue de la fonction « *coût Diamètre* » qui admet les plus grandes valeurs enregistrées.
- L'autre après l'étape n°550 : la fonction de « *coût Parabole* » confirme son efficacité, tandis que la fonction de « *coût Exponentiel* » n'arrive pas à l'assurer. Elle passe même parfois au dessus des fonctions de coût « *Diamètre* » et « *Discrétisé* » : une explication possible est que vu la dispersion des demandes de connexions sur la totalité des liens, de longs chemins ont été attribué et non libérés ce qui a causé la dégradation de la qualité de la solution offerte par cette fonction.

Tout au long de la simulation, presque la fonction de coût « *Diamètre* » a enregistré les valeurs les plus élevées sur le lien le plus congestionné. Mais ce qui peut être remarqué aussi est que la fonction de coût « *Discrétisé* » admet trois paliers au niveau des valeurs de la charge appliquée sur le lien le plus congestionné : résultat du aux trois intervalles fixés pour l'attribution des coûts par rapport au taux d'occupation des liens.

2.3.3- Conclusion générale :

Les simulations faites montrent clairement qu'avec un algorithme de routage comme celui qu'on a utilisé et sous un trafic complètement dynamique, une fonction très pénalisante (pour l'utilisation des capacités sur les liens) se montre la moins efficace. Une fonction moins pénalisante s'est montré plus robuste pour les changements de trafic (entrées et sorties des connexions), en effet la fonction de coût à augmentation parabolique par rapport à la capacité utilisée a assuré globalement les meilleurs résultats. Un comportement acceptable globalement a été observé pour les autres fonctions de coût à exception de leurs tendances à utiliser toujours les plus courts chemins à base de nombre de sauts (d'une façon meilleure pour la fonction discrétisé).

Ainsi, il est clair qu'un choix judicieux de la fonction de coût est exigé pour assurer la meilleure utilisation des ressources sur le réseau.

Chapitre 3

PROBLEME DE RECONFIGURATION

Dans un réseau optique WDM *reconfigurable*, il est possible de déplacer une connexion (canal de communication optique entre 2 noeuds du réseau, généralement à plus de 2.5Gbit/s) vers une nouvelle route. L'évolution du trafic (ajout/suppression de connexions) au court du temps entraîne une mauvaise utilisation des ressources du réseau. Une reconfiguration du réseau consiste alors à déplacer des connexions pour retrouver une utilisation optimale des ressources. Toutefois, le déplacement d'une connexion comporte des risques: arrêt du trafic, perte d'informations, ...

Dans cette partie, l'objectif est d'extraire des solutions assurant la résolution du problème de la reconfiguration. Nous travaillons dans un cadre général, et presque toutes les solutions existantes dans la littérature sont relatives aux réseaux WDM où le problème est important. Ainsi, chaque solution extraite doit être adaptée à nos cas de figures, évalué et amélioré tant que c'est possible.

3.1 - Motivation pour le re-routage des connexions :

Pour un routage statique [5], les routes pour n'importe quelle connexion sont prédéterminées et ne changent pratiquement pas, tandis que pour un routage dynamique un chemin attribué pour une demande de connexion dépendra de l'étape de sa génération donc de l'état d'occupation des ressources du réseau. L'état du réseau changera ainsi suivant l'évolution du trafic, et dans un cas où une demande de connexion est rejetée et ce faute de disponibilité de ressources, une reconfiguration peut être utile afin de permettre l'accommodation de cette connexion sur le réseau.

Un autre cas où la reconfiguration peut se distinguer intéressante est le cas de panne d'un lien ou d'un nœud. Cette situation exige le re-routage de la connexion sur un chemin alternatif mais ce dernier peut ne pas être toujours disponible. Alors, une action de reconfiguration d'une ou quelques connexions existantes peut libérer de la capacité sur ce chemin pour accommoder la connexion rejetée.

La reconfiguration peut être utile dans un cadre préventif : l'évolution du trafic au cours du temps (entrée ou sortie de connexions) rendent nécessairement l'état courant d'utilisation des ressources non optimal et même gênant surtout dans le cas où les liens du réseau sont en état de congestion, d'où une action périodique de reconfiguration peut assurer la survie et le fonctionnement acceptable du réseau.

La reconfiguration admet en plus de ses avantages des inconvénients non négligeables qui posent impérativement des limites au niveau des solutions proposées pour ce sujet : la reconfiguration entraîne une interruption de service qui n'est pas appréciée par les opérateurs des réseaux. Dès lors, une reconfiguration totale n'est pas envisageable pour ce problème et on préférera généralement fixer un seuil à ne pas dépasser pour le nombre de changement. Ainsi l'objectif se distingue comme étant un compromis entre le nombre de changement à apporter à une instance courante et l'efficacité résultante suite à cette reconfiguration.

3.2 – Classes de reconfigurations:

Il existe 3 types de reconfigurations à savoir :

- Reconfiguration séquentielle : consiste à re-router les connexions à reconfigurer l'une après l'autre. Une technique assez simple à mettre en place et ne dérange que les connexions qui sont en cours de re-routage. L'inconvénient d'une telle technique est le temps de traitement de la reconfiguration qui est assez long et ce qui est intéressant à résoudre dans ce problème est la détermination du meilleur ordre du re-routage des connexions et ce afin d'assurer le minimum d'interruption de service possible.
- Reconfiguration en parallèle : toutes les connexions qui ont besoin d'être re-router sont interrompues en même temps et rétablies sur leurs nouveaux chemins. Cette technique est assez complexe à mettre en place surtout au niveau des grands réseaux (nécessité de synchronisation) mais cette technique a comme avantage l'élimination des dépendances existantes dans la reconfiguration séquentielle.
- Reconfiguration hybride : combinaison des deux précédentes. En cas de dépendance utiliser la technique séquentielle et en cas de non dépendance utiliser la technique parallèle.

Une autre classification, selon le nombre de changement peut être introduite d'après [4] :

- Reconfiguration totale : la plus part des connexions déjà établies sont re-routées. Ce type de reconfiguration est à éviter pour les énormes pertes qu'il cause.
- Reconfiguration incrémentale : une connexion est re-routée à chaque étape. Cette technique marche bien pour les réseaux dont le trafic offert ne change pas trop (fréquence de changement assez réduite : Pour les réseaux WDM).
- Reconfiguration hybride (à base des deux premières) : exécutée en deux phases : la première phase consiste à exécuter un ensemble de reconfigurations en même temps tandis que la deuxième phase consiste à modifier juste un chemin par étape.

3.3- Solutions de base :

La première heuristique apparue pour ce genre de problème a été introduite par [3], il existe toujours une formulation possible en programme linéaire pour le problème de reconfiguration mais une telle formulation ne sera applicable que pour des réseaux de petite tailles, ainsi les heuristiques sont ici incontournables. L'heuristique proposée est la suivante : les charges appliquées sur les liens sont considérées comme déclencheurs de tout acte de reconfiguration mais une seule de reconfiguration est permise à chaque étape. Etant donnée que la solution a été introduite dans le cadre d'un réseau WDM, une reconfiguration peut être un ajout d'un nouveau chemin optique ou un retrait d'un chemin optique faiblement exploité :

- *Un ajout* : si la charge appliquée à un lien est supérieure à un certain seuil alors il faut chercher parmi tous les trafics empruntant ce lien celui transportant la plus grande quantité de données et essayer de le basculer sur un autre chemin totalement ou partiellement disjoints.
- *Une suppression* : pour les liens sous utilisés ou mal exploité, la reconfiguration consiste à basculer le trafic sur un autre lien ou chemin (suivant la disponibilité) mieux occupé. L'inconvénient d'une telle solution est que si on ne fait pas attention au choix de la valeur du *seuil minimal*, nous ferons face à un problème d'oscillation. Cette solution issue de l'environnement des réseaux WDM a pour objectif d'éliminer l'utilisation inefficace des ressources des réseaux et d'assurer la disponibilité des longueurs d'ondes à fin de permettre la mise en place des chemins optiques en cas de besoin dans le futur

Dans le cas d'un conflit entre une action d'ajout ou de suppression, un ajout sera mis en place et non pas un retrait.

Dans notre cas de figure, on travaille sur des graphes dont chacun des liens composants admet une certaine quantité de capacité disponible et un coût relatif. La solution précédente peut être implémentée et évaluée, mais une autre proposition peut bien être introduite : en effet, on peut se baser sur le gain en terme de coût au lieu de déclencher une reconfiguration à base du seuil représentant le taux d'occupation du lien.

La prochaine partie de notre travail portera en premier lieu sur les heuristiques proposées pour exécuter une reconfiguration par étape et en deuxième lieu sur les propositions pour exécuter plusieurs reconfigurations à une étape donnée tout en identifiant la nécessité ou non de telle proposition par des simulations.

3.3.1 – 1^{re} classe d'heuristiques de reconfigurations :

La première classe des heuristiques de reconfiguration consiste à exécuter un seul changement à savoir un ajout ou une suppression de connexion suite au déclenchement de la procédure de reconfiguration.

On se propose d'étudier et d'analyser deux modèles de raisonnement, un se basant sur les coûts des chemins et un autre se basant sur la congestion ou l'occupation des liens.

Pour cette phase, la politique de reconfiguration peut consister à router juste les demandes de connexion générées tant qu'on n'a pas dépassé un seuil d'occupation sur le lien le plus congestionné (à définir). Après dépassement, une reconfiguration est exécutée à chaque étape (s'il y a nécessité bien sûr). Une autre politique beaucoup plus simple consistera à démarrer les actions des reconfigurations dès le début de la simulation.

Pour le processus de reconfiguration, il est extrêmement simple du fait qu'on ne déplace qu'une seule connexion.

3.3.1.1- 1^{re} Heuristique :

Voici l'heuristique proposée à base des coûts des chemins, cette heuristique est relative à une seule action de re-routage de connexion et décrit comment choisir la connexion à déplacer :

A- Algorithme formel :

Algorithme1	1 ^{er} Heuristique de reconfiguration
-------------	--

Entrée : $G = (V, E)$, $Duree_simulation$, $D = \{d_1, d_2, ..., d_m\}$ (ensemble des demandes de connexions actuellement sur le réseau), l'ensemble des chemins relatifs à chacune des connexions et leurs coûts, les données relatives aux capacités disponibles et coûts des liens.

Sortie : connexion re-routée

Variables locales :

$k_meilleurs_chemins$: vecteur servant à stocker les « k » plus courts chemins pour une connexion.
 $connexion_a_deplacer$: structure pour stocker les données relatives à la connexion choisie pour être re-router à savoir son chemin actuel, son nouveau chemin possible et le gain résultant.
 $Chemin_actuel$, $coût_chemin_actuel$, $nouveau_chemin$, $chaux_réussite$, $connex$, $gain$ et $gain_max$: ensemble des variables intermédiaires.

1. $gain \leftarrow 0$; $gain_max \leftarrow 0$
2. **Pour** chaque « $connex$ » **de** $liste_de_connexions$ **faire**
3. $chemin_actuel \leftarrow DonnerChemin.Actuel(connex)$
4. $coût_chemin_actuel \leftarrow eliminerConnexion(connex, chemin_actuel)$
5. $k_meilleurs_chemins \leftarrow DonnerKMeilleursCheminsPour(connex)$
6. **Pour** i allant **de** 0 **à** $(k - 1)$ **faire**
7. **Si** $((k_meilleurs_chemins[i] = chemin_actuel) \text{ ou } (cout \text{ de } k_meilleurs_chemins[i] \geq cout_chmin_actuel))$ **alors**
8. $mettreAJour(connex, chemin_actuel)$
9. **Sortir**
10. **Sinon Si** $((cheminRealisable(connex, k_meilleurs_chemins[i]))$ **alors**
11. $gain \leftarrow (DonnerCoutChemin(k_meilleurs_chemins[i])) - coût_chemin_actuel$
12. **Si** $(gain_max < gain)$ **alors**

```

13.          nouveau_chemin ← k_meilleurs_chemins[i]
14.          connexion_a_deplacer ← nouvelle_connexion(connex, chemin_actuel, nouveau_chemin, gain)
15.          mettreAJour(connex, chemin_actuel)
16.          Fin Si
16.          Sortir
17.      Fin Si
18.  Fin Pour
19. Fin Pour
20. eliminerConnexion(connexion_a_deplacer)
21. mettreAJourConnexion(connexion_a_deplacer)

```

- Description détaillée de l'algorithme proposé :

A une étape donnée, l'algorithme de reconfiguration a été déclenché afin de trouver la connexion à re-router (rapportant le plus de gain).

- ✚ Extraire le chemin rapportant plus de gain et réalisable pour chaque connexion : Pour chacune des connexions déjà établies sur le réseau, extraire et éliminer le chemin actuel (mettre à jour les capacités disponibles sur les liens concernées ainsi que leurs coûts). Ensuite, chercher les « *k* » meilleurs chemins possibles pour cette connexion. Enfin, trouver s'il existe un chemin moins coûteux que le chemin actuel et pouvant accueillir la connexion. Si ce chemin existe alors passer à l'étape suivante sinon remettre à cette connexion son chemin actuel et sortir.
- ✚ Ajouter une connexion à l'ensemble des connexions concurrentes : Une connexion pour laquelle on a trouvé un nouveau chemin moins coûteux et valide, son gain relatif doit être comparé au gain maximal courant enregistré : si ce gain est supérieur au gain maximal alors cette connexion sera enregistrée comme étant la connexion à déplacer sinon remettre pour la connexion son chemin actuel et sortir.
- ✚ Éliminer et mettre à jour la connexion à déplacer : Après avoir choisie la connexion à déplacer, il faut juste enlever cette connexion de son chemin actuel (mettre à jour les coûts et les capacités disponibles sur les liens) et ensuite la remettre sur son nouveau chemin (mettre à jour les coûts et les capacités disponibles sur les liens).

B- Simulation et résultats :

Un scénario de trafic a été généré avec les paramètres suivants :

- *Nombres de requêtes* : 800 requêtes.
- *Intervalle des arrivées* : 1618 étapes.
- *Réseau utilisé* : cost239 avec 48 unités de capacité libres sur les liens.

En premier lieu on va voir l'impact de notre première heuristique sur le scénario de trafic pour chacune les fonctions de coût déjà proposées.

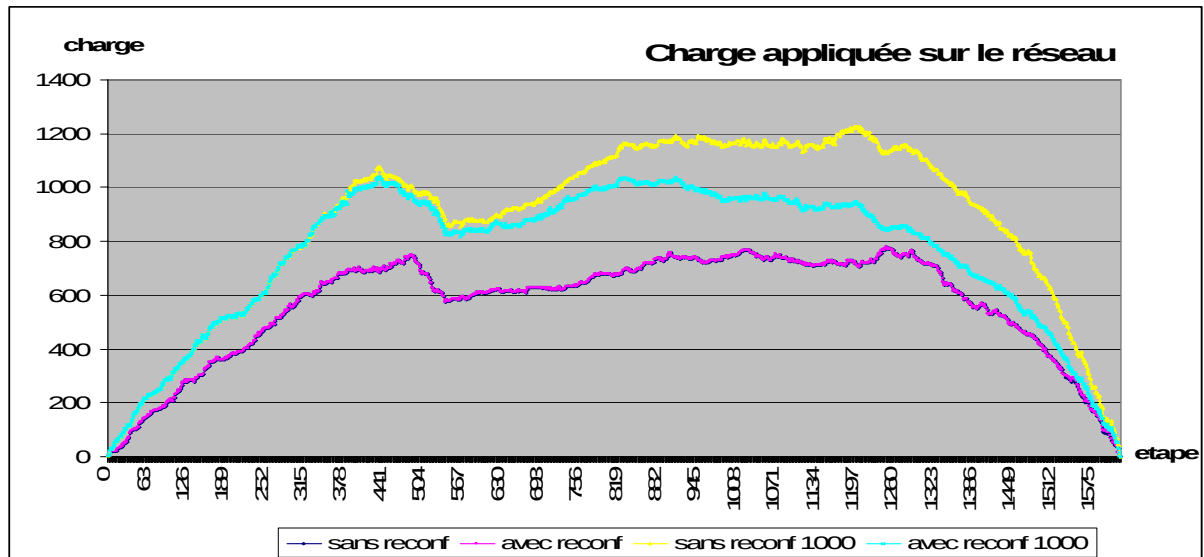


Figure 3.1 : la charge appliquée sur le réseau avec une fonction de coût Diamètre

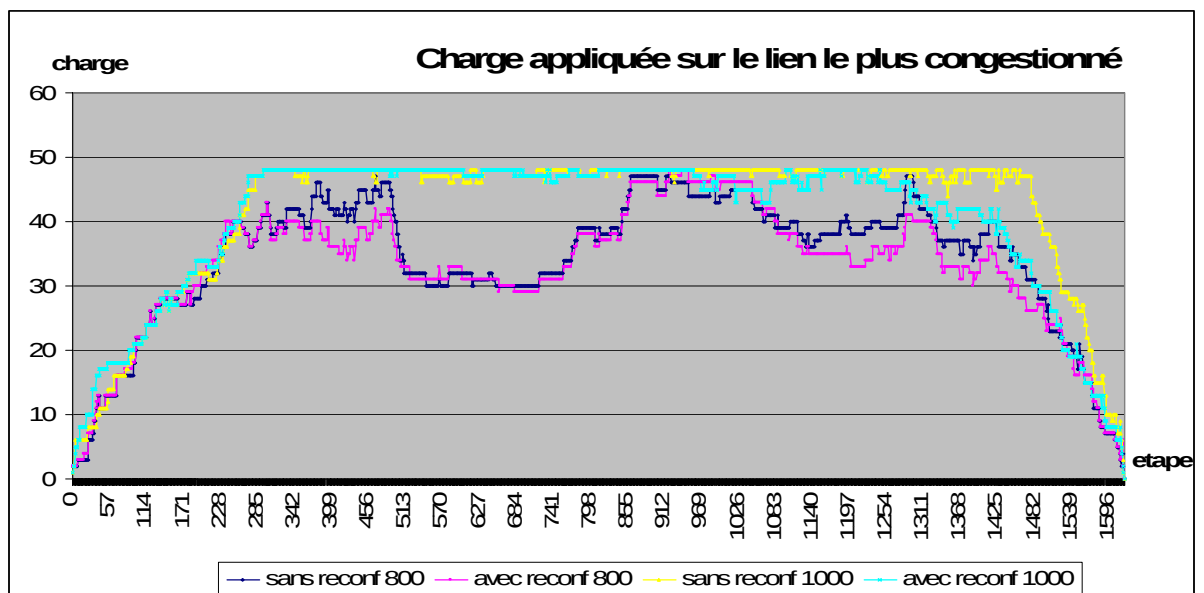


Figure 3.2 : la charge sur le lien le plus congestionné avec une fonction de coût Diamètre

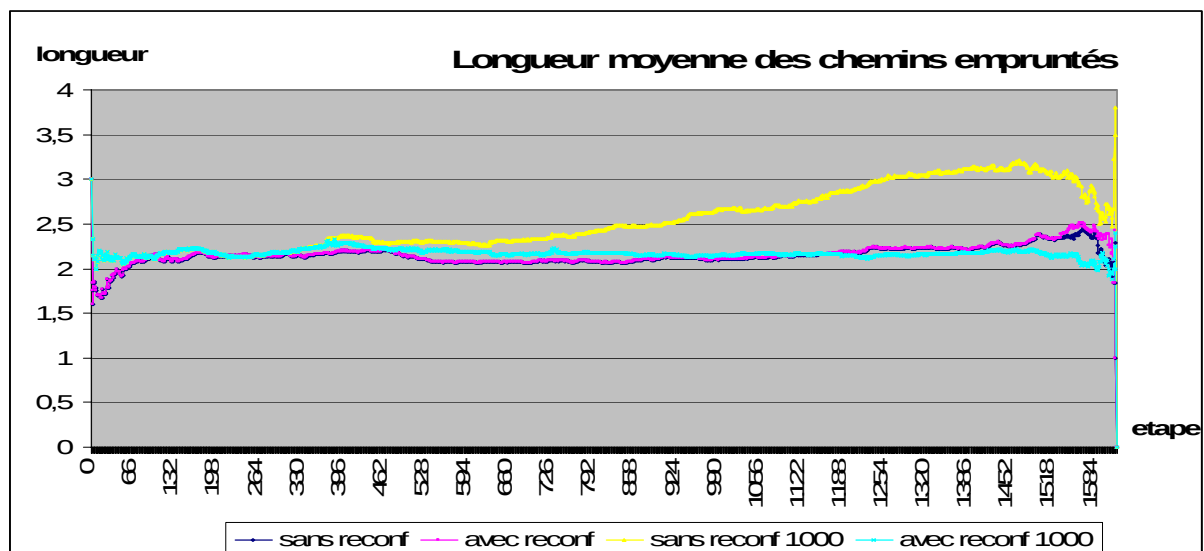


Figure 3.3 : Longueur moyenne des chemins avec une fonction de coût Diamètre

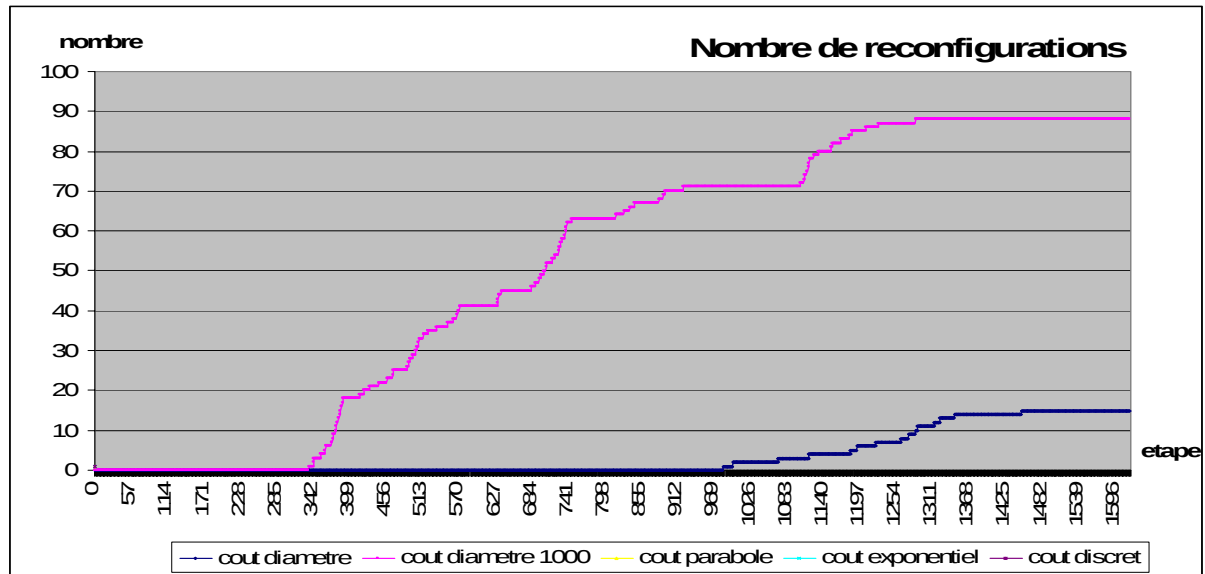


Figure 3.4 : Nombre de reconfigurations enregistrées tout au long de la simulation

Avec la simulation à base 800 demandes de connexion, on ne remarque aucun changement et ça revient à la nature de la fonction de coût utilisée qui ne permet le déclenchement du processus de reconfiguration qu'à l'épuisement des capacités disponibles sur les liens du réseau. La *figure3.4* nous affirme cette logique par le nombre de reconfigurations très petit enregistré tout au long de la simulation. Mais pour voir l'impact réel de notre heuristique, on a augmenté un peu le nombre de demandes générées et on a remarqué sur la *figure3.1*, une diminution de la charge globale appliqué sur le réseau et aussi sur la *figure3.3* une diminution de la longueur moyenne des chemins empruntés tout au long de la simulation. La fonction de coût « Diamètre » sélectionne pour chaque demande de connexion le plus court chemin en terme de sauts, et si on regarde sur la *figure3.2*, on remarque qu'à partir de l'étape n° 250 le lien le plus congestionné n'a plus de capacité disponible : ainsi avec l'algorithme de routage sans reconfiguration on est obligé à chercher des chemins n'empruntant pas ce lien et qui peuvent être long (chemins plus longs donc charge totale plus grande). Cependant, avec l'heuristique de reconfiguration, le travail consistera à amener la charge utilisée sur le lien le plus congestionné au dessous de la capacité totale disponible et ainsi de rendre ce lien utilisable par tous les chemins.

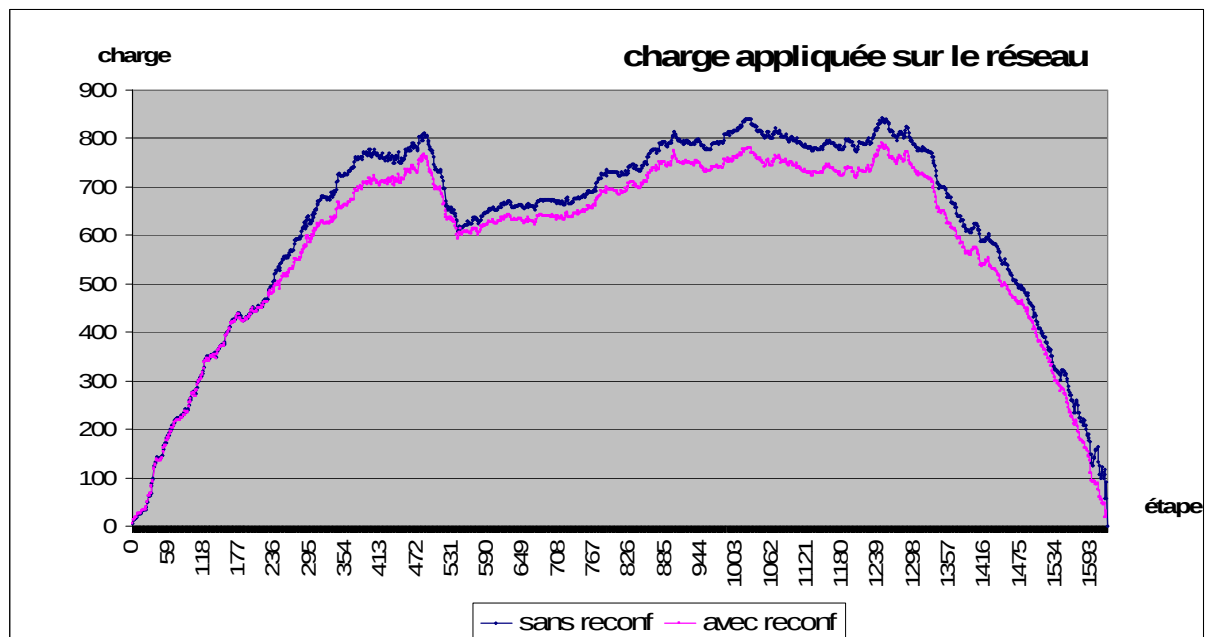


Figure 3.5 : Charge appliquée sur le réseau avec un fonction de coût Parabolique

Pour le cas d'une fonction de coût parabolique, les actions de reconfiguration ont amélioré considérablement la performance de mise en place des connexions. En effet, la *figure 3.5* représentant l'évolution de la charge globale que subit le réseau montre une diminution considérable de charge à partir du début du processus de reconfiguration : cette amélioration est confirmée par la *figure 3.6* visualisant la longueur moyenne des chemins attribués aux connexions sur le réseau. Le processus de reconfiguration a régulé en quelque sorte la longueur moyenne des chemins pour rester toujours à une valeur de 2.5.

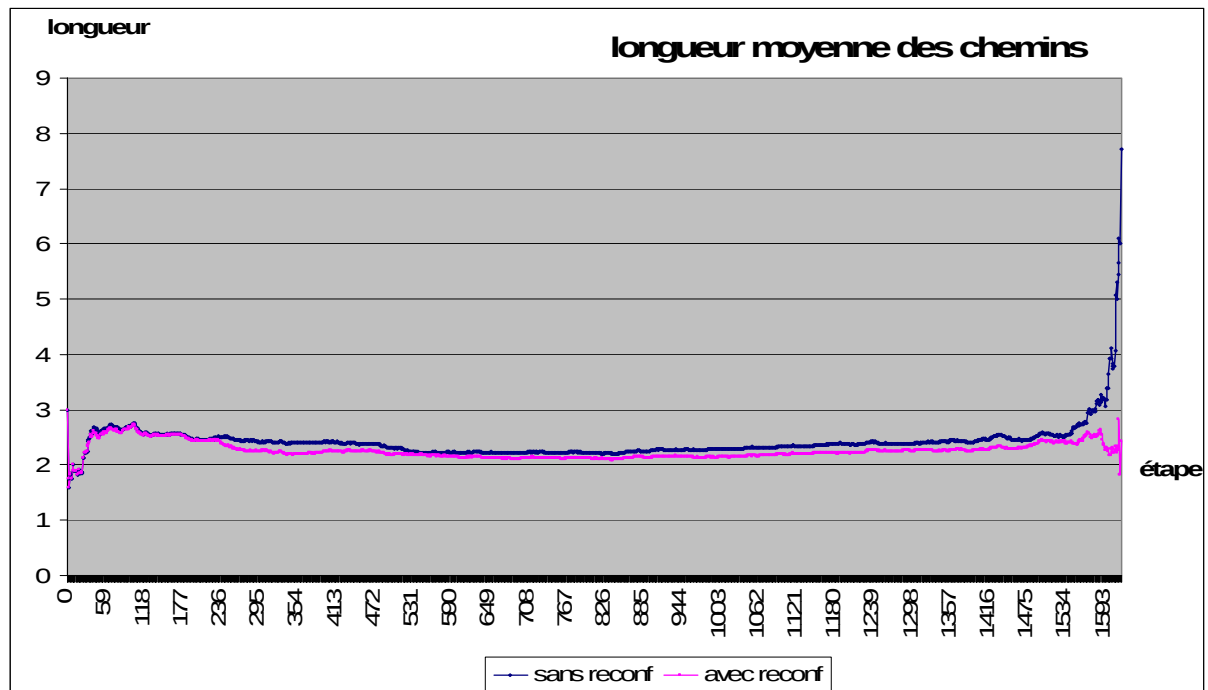


Figure 3.6 : Longueur moyenne des chemins avec une fonction de coût Parabolique

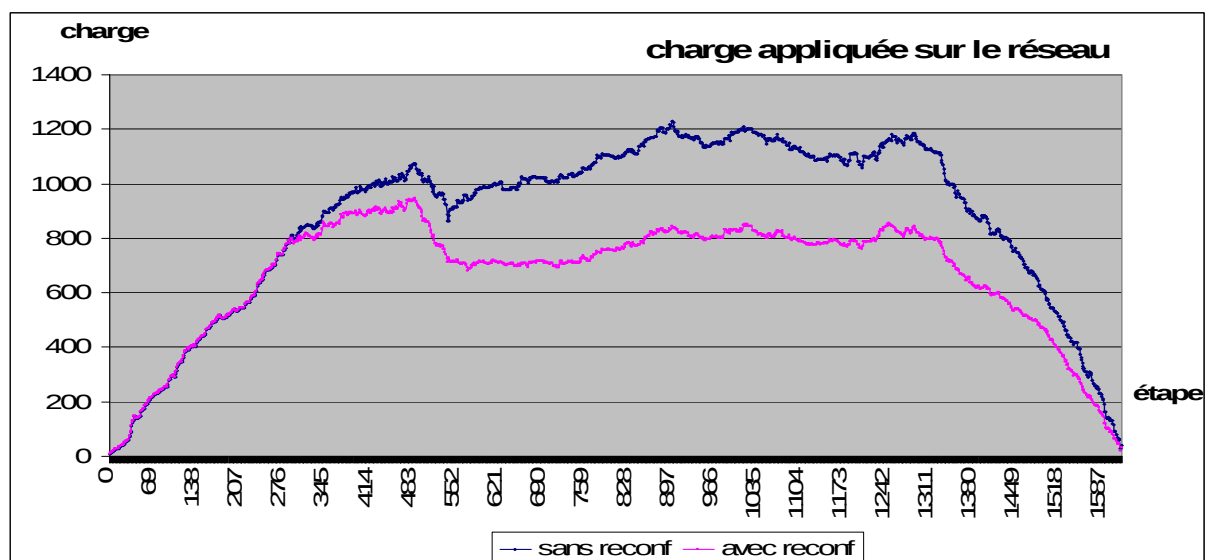


Figure 3.7 : Charge appliquée sur le réseau avec une fonction de coût Exponentiel

Pour une fonction à coût exponentiel, l'impact de la reconfiguration se révèle extrêmement important. En effet, cette fonction essaye d'utiliser les ressources du réseau d'une manière équilibrée. Ainsi des chemins assez long sont généralement choisis pour éviter l'utilisation des liens congestionnés qui coûtent chers. La durée de détention de ces ressources peut être longue ce qui rend à certain moment l'utilisation courante des ressources du réseau non optimale (après fin de certaines connexions et génération de nouvelles). Les actions de reconfiguration pour une fonction de coût à croissance

exponentielle sont indispensables. On remarque une amélioration au niveau des trois paramètres observés : une diminution de la charge globale appliquée sur le réseau, une régulation de la longueur moyenne des chemins empruntés ainsi qu'une diminution assez remarquable de la charge observée sur le lien le plus congestionné. Ce dernier point est important car il est faut préserver des capacités sur l'ensemble des liens (une telle situation nous permet d'accueillir n'importe quelle demande de connexion).

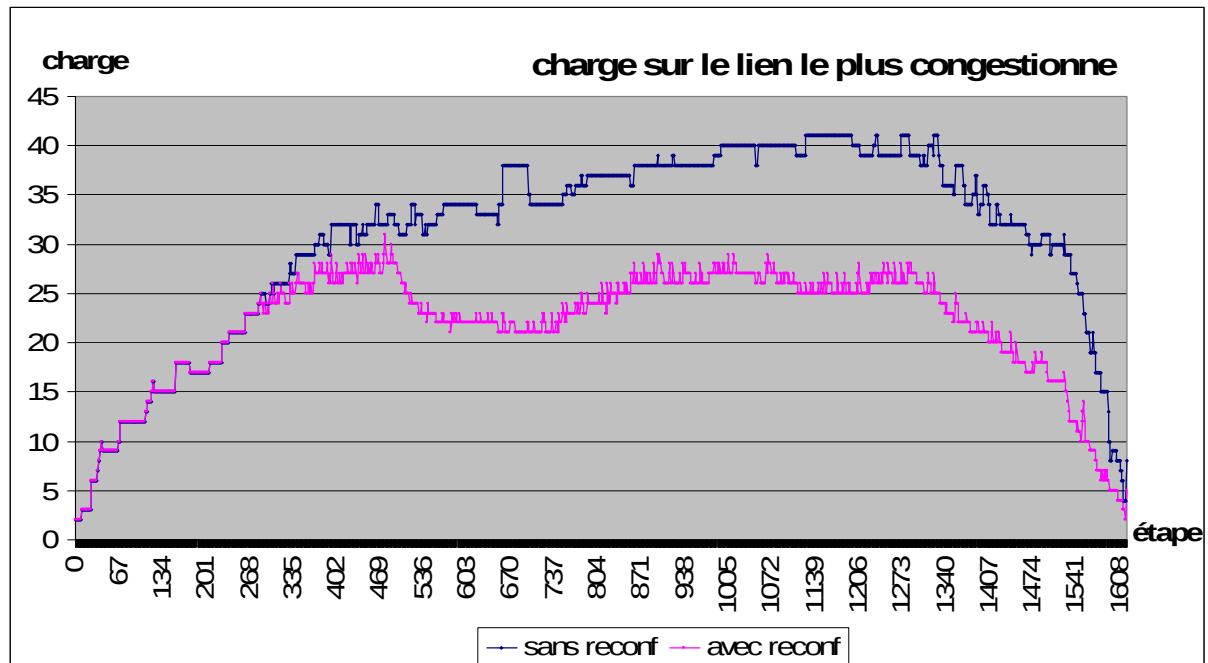


Figure3.8 : la charge sur le lien le plus congestionné avec une fonction de coût Exponentiel

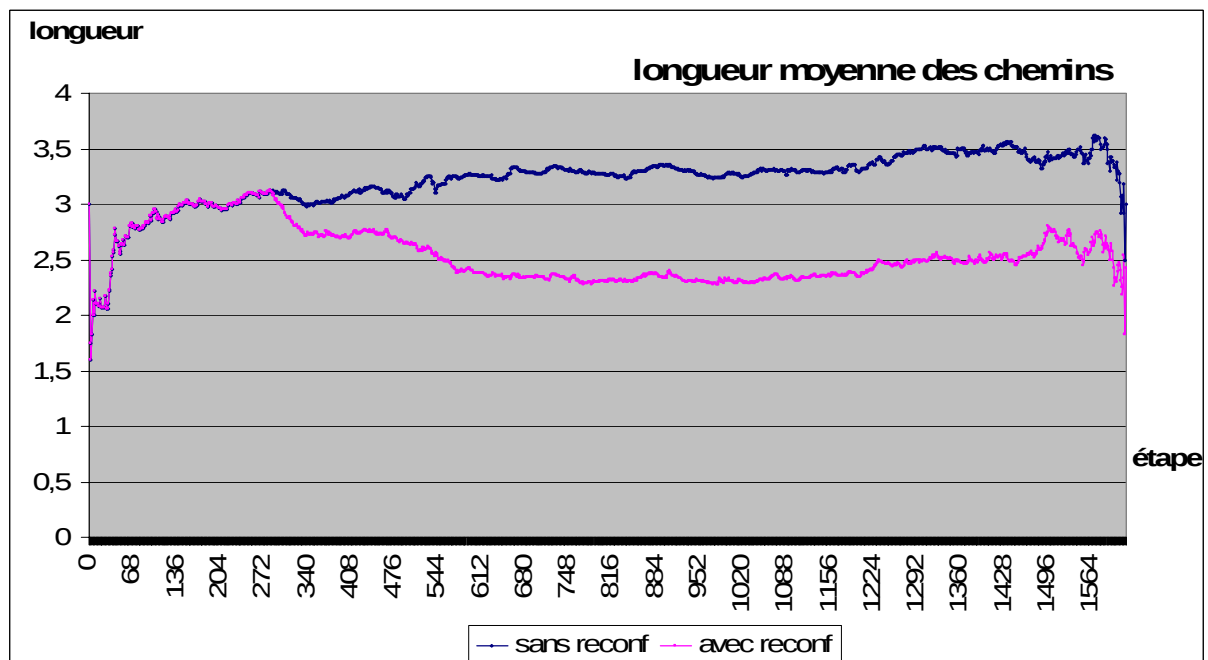


Figure3.9 : Longueur moyenne des chemins avec une fonction de coût Exponentiel

Maintenant on va voir, après application de la première heuristique de reconfiguration, une comparaison au niveau d'utilisation des ressources pour chacune des fonctions de coût.

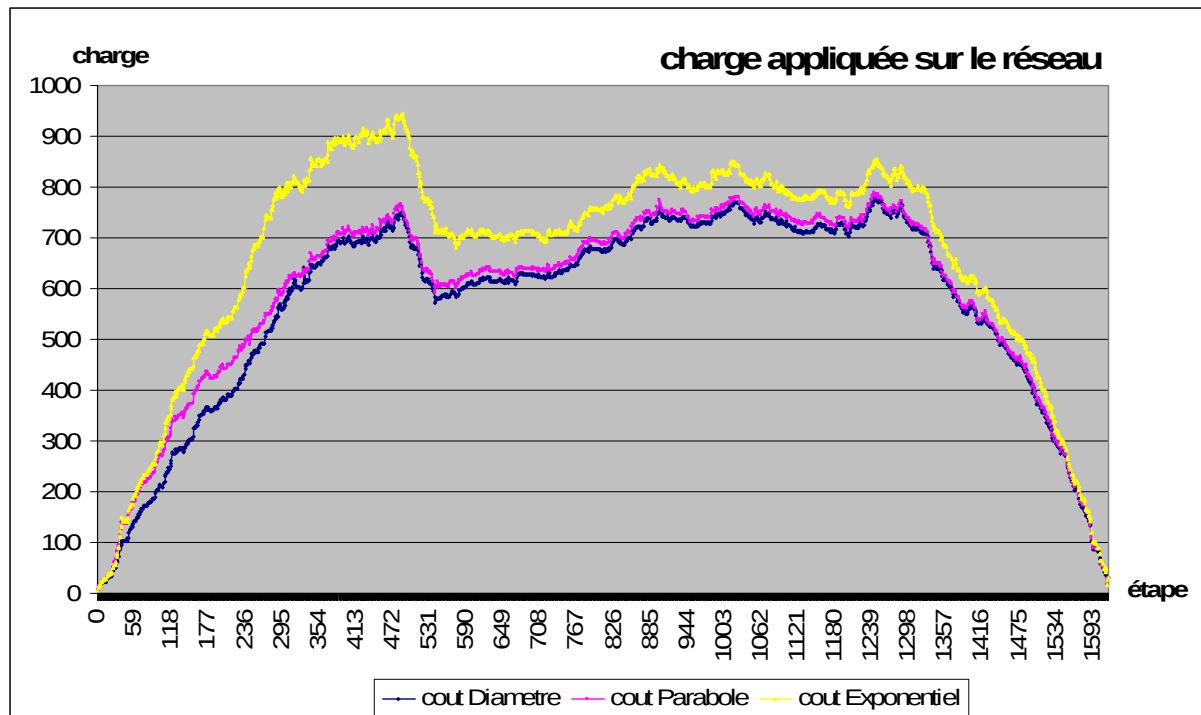


Figure3.10 : charge globale appliquée sur le réseau

Avec les actions de reconfiguration, la fonction de coût à croissance exponentiel reprend son avantage qui est la répartition de charge sur le réseau : elle assure les valeurs minimales de la charge appliquée sur le lien le plus congestionné. En contre partie, la charge globale appliquée est plus grande.

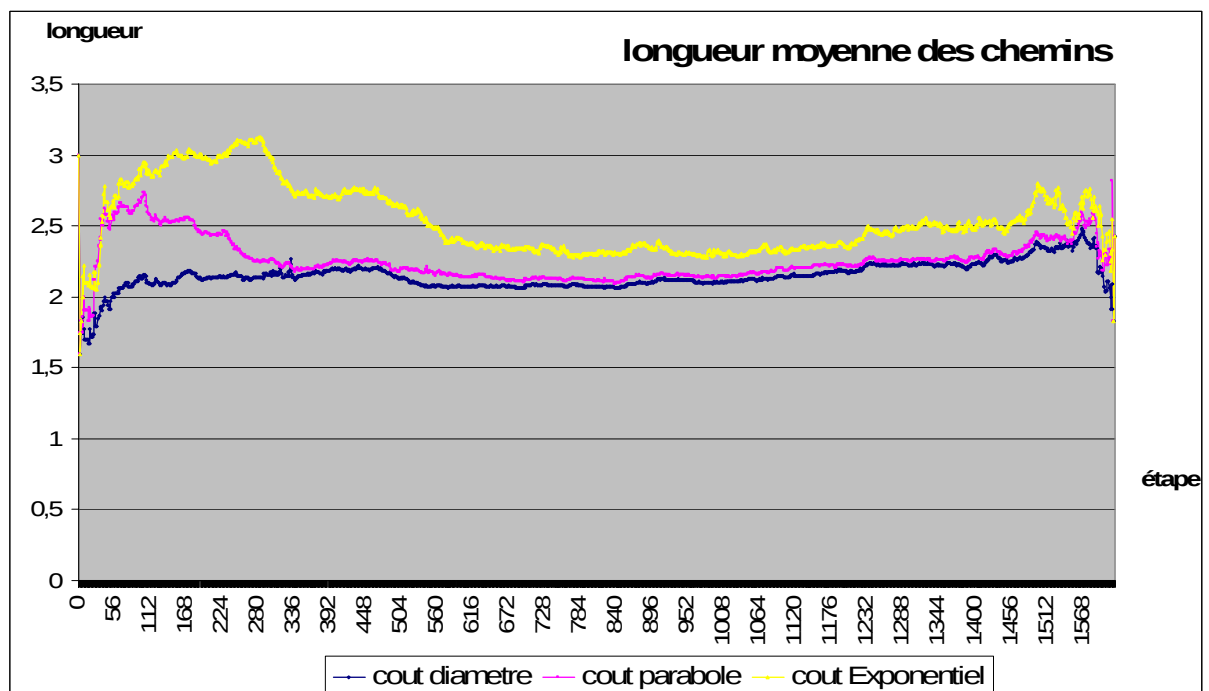


Figure3.11 : Longueur moyenne des chemins empruntés

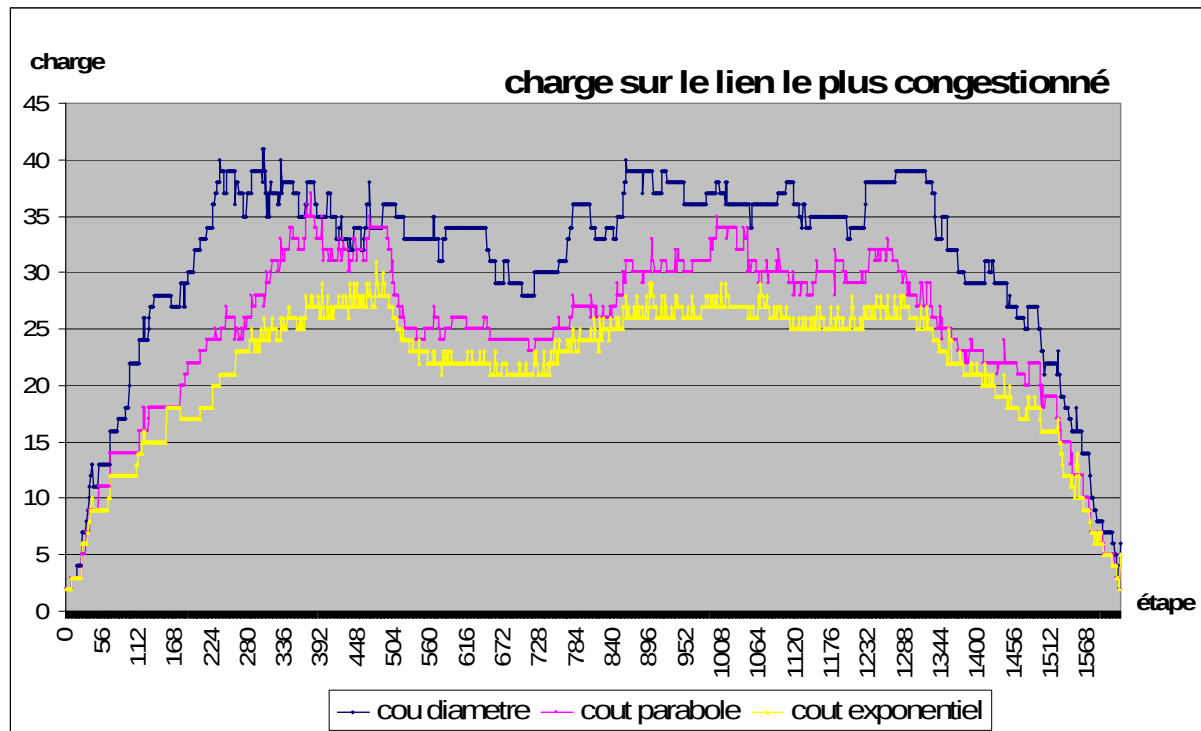


Figure 3.12 : charge appliquée sur le lien le plus congestionné

Dans la partie précédente, on a remarqué que la fonction de coût à croissance « *exponentielle* » était la moins performante et on a même remarqué des rejets de demandes de connexion durant la simulation. Mais en procédant parfois à des actions de reconfiguration on remarque que cette fonction de coût admet de bonnes performances sur la répartition équilibrée des charges.

Pour tout le travail qui suivra, on n'utilisera que la fonction de coût parabolique qui a affirmé ses performances et ses qualités avec ou sans actions de reconfiguration. Cette fonction est une fonction qui n'est pas très pénalisante mais elle est raisonnablement sensible aux changements que subit sur le réseau.

3.3.1.2- 2^{ème} Heuristique :

La deuxième heuristique basée sur le dépassement d'un seuil [3] d'occupation sur un des liens, consiste à trouver la plus grande charge relative à une connexion empruntant ce lien. La connexion étant trouvée, le travail de reconfiguration revient alors à chercher, si c'est possible, à déplacer complètement la connexion de son chemin actuel et dans le cas échéant la déplacer partiellement, parfois juste du lien le plus congestionné seulement.

Cette solution, extraite du monde des réseaux WDM nécessite évidemment quelques adaptations pour être appliquée à nos cas de figure. L'avantage de la solution précédente par rapport à cette solution est qu'elle n'est pas dépendante d'un choix de seuil d'occupation maximal, tandis que [3] exige un choix judicieux. Ce choix concerne la valeur du seuil maximal pour établir un nouveau chemin optique et la valeur du seuil minimal pour supprimer un chemin optique déjà existant et sous utilisé.

L'heuristique adaptée proposée, ne se base sur aucun choix de seuil. En effet, il y'a juste un seuil facultatif à fixer pour débiter les actions de reconfiguration. Ensuite, une mesure à chaque étape de la plus grande valeur d'occupation enregistrée sur le réseau est faite « v_{max} ». L'objectif du processus de reconfiguration à ce stade sera :

- Trouver la connexion « C » injectant le plus de charge sur un lien « l » ayant le taux d'occupation « v_{max} ».

- Essayer de trouver pour cette connexion un nouveau chemin, à condition que la charge sur chacun des liens de ce nouveau chemin ne dépassera ou ne soit égale à « v_{max} » après la reconfiguration.
- Si le changement complet de chemin est impossible alors changer partiellement le chemin ou juste pour seulement le lien « l » tout en assurant une valeur moindre que « v_{max} » pour le taux d'occupation.

A- Algorithme formel :

Voici l'heuristique proposée à base de dépassement de seuil, cette heuristique est relative à une seule modification de connexion et décrit comment choisir la connexion à re-router et la valider :

Algorithme2

2^{ème} Heuristique de reconfiguration

Entrée : $G = (V, E)$, $Duree_simulation$, $D = \{d_1, d_2, ..., d_m\}$ (ensemble des demandes de connexions actuellement sur le réseau), l'ensemble des chemins relatifs à chacune des connexions et leurs coûts, les données relatives aux capacités disponibles et coûts des liens.

Sortie : connexion re-routée

Variables locales :

$K_meilleurs_chemins$: vecteur servant à stocker les « k » plus courts chemins pour une connexion.
 $connexion_a_deplacer$: structure pour stocker les données relatives à la connexion choisie pour être re-router à savoir la connexion, le chemin actuel et la capacité requise.
 $Arc_congestionne$, et $reroutage_fait$: ensemble des variables intermédiaires.

```

1.  $seuil\_max \leftarrow extraireSeuilMax()$ 
2.  $les\_liens\_les\_plus\_congestionnes \leftarrow extraireSeuilMax(seuil\_max)$ 
3. repete
4.    $arc\_congestionne \leftarrow les\_liens\_les\_plus\_congestionnes[0]$ 
5.    $connexion\_concurrentes \leftarrow connexionsPouvantEtreDeplacees(arc\_congestionne)$ 
6.    $reroutage\_fait \leftarrow faux$ 
7.   Pour  $i$  allant de 0 à la taille de  $connexions\_concurrentes$  faire
8.      $connexion\_a\_deplacer \leftarrow connexions\_concurrentes[i]$ 
9.      $eliminerConnexion(connexion\_a\_deplacer)$ 
10.     $k\_meilleurs\_chemins \leftarrow DonnerKMeilleursCheminsPour(connexion\_a\_deplacer)$ 
11.    Pour  $i$  allant de 0 à la taille de  $k\_meilleurs\_chemins$  faire
12.      Si  $((k\_meilleurs\_chemins[i] \text{ ne contient pas } (arc\_congestionne)))$  alors
13.        Si  $((k\_meilleurs\_chemins[i] \neq \text{chemin de la connexion\_a\_deplacer})$ 
          &&  $(seuilNonDepasse(connexion\_a\_deplacer, seuil\_max, k\_meilleurs\_chemins[i])))$  alors
14.           $mettreAJour(connex, k\_meilleurs\_chemins[i])$ 
15.           $reroutage\_fait \leftarrow vrai$ 
16.          Sortir
17.        Fin Si
18.      Fin Si
19.    Fin Pour
20.    Si  $(reroutage\_fait)$  alors
21.      Sortir
22.    Fin Si
23.  Fin Pour
24.   $supprimer les\_liens\_les\_plus\_congestionnes[0]$ 
25. jusqu'à  $(reroutage\_fait \text{ ou plus d'éléments dans } les\_liens\_les\_plus\_congestionnes)$ 

```

Description détaillée de l'algorithme proposé :

A une étape donnée, l'algorithme de reconfiguration a été déclenché afin de trouver la connexion à re-router pour utiliser efficacement les ressources disponibles sur le réseau.

- ✚ Déterminer les liens les plus congestionnés : La première phase consiste à déterminer les liens enregistrant la plus grande valeur de taux d'occupation sur le réseau. L'ensemble des liens sera enregistré dans une structure de données.
- ✚ Extraire les connexions empruntant le lien congestionné : Après avoir sélectionner un des liens les plus congestionnés, il faut extraire les connexions empruntant ce lien et les trier suivant la charge qu'ils font passer sur ce lien.
- ✚ Reconfigurer : Sélectionner les connexions en commençant par celle qui a plus de charge et essayer de lui trouver un autre chemin. Il faut bien évidemment enlever la connexion du réseau. Tous les liens du nouveau chemin trouvé ne doivent pas dépasser le seuil maximal enregistré. Si le chemin est trouvé et le re-routage est réalisable alors le processus de reconfiguration est fait.

Si le processus de reconfiguration est complètement réalisé alors une action de mise à jour est requise pour tenir compte de la mise en place du nouveau chemin. Sinon, l'ancien chemin doit être remis en place. Le cas où on n'a pas de reconfiguration à faire est le cas où on ne trouve aucune solution réalisable pour toutes les demandes empruntant chacun des liens enregistrant la plus grande valeur pour le taux d'occupation.

B- Simulations et résultats :

Un scénario de trafic a été généré avec les paramètres suivants :

- *Nombres de requêtes* : 1000 requêtes.
- *Intervalle des arrivées* : 1618 étapes.
- *Réseau utilisé* : cost239 avec 48 unités de capacité libres sur les liens.
- *Fonction de coût* : parabolique.

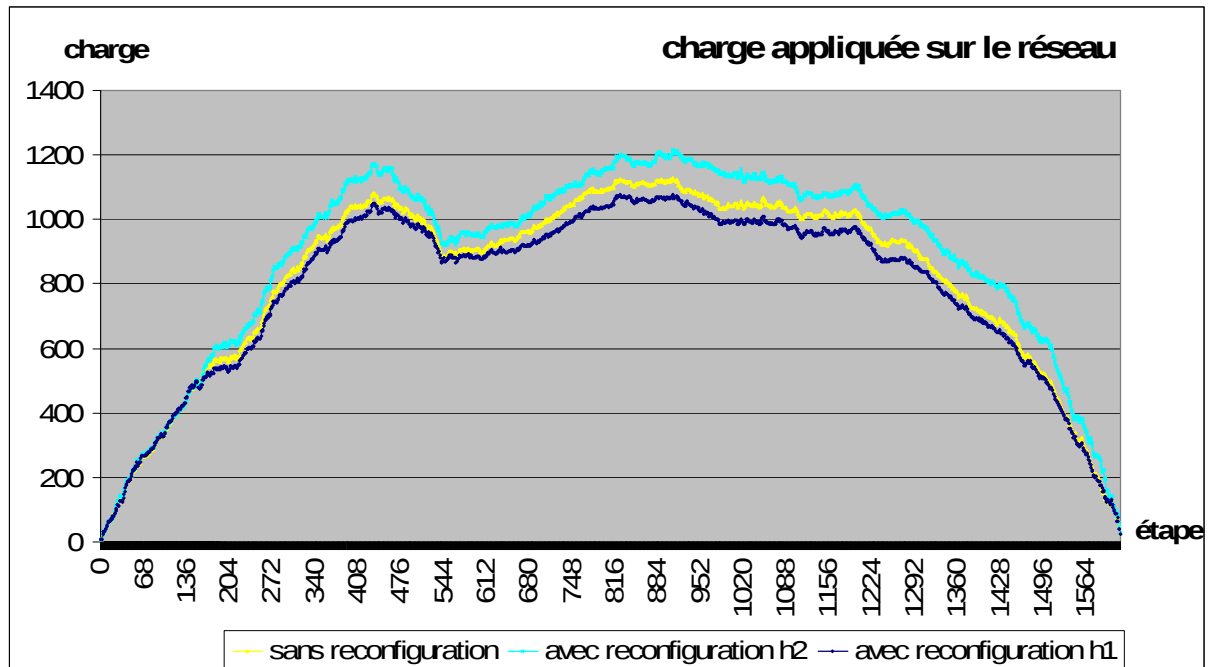


Figure3.13 : charge globale appliquée sur le réseau avec ou sans reconfiguration

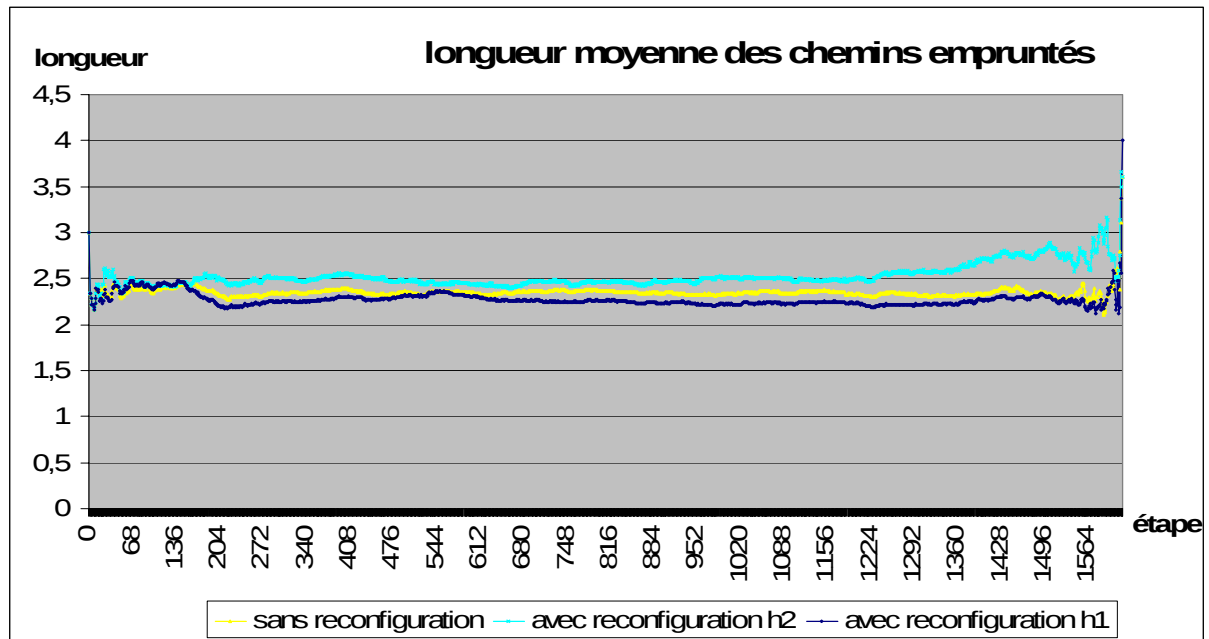


Figure3.14 : longueur moyenne des chemins empruntés avec ou sans reconfiguration

La deuxième heuristique assure une certaine amélioration bien visualisé sur la *figure3.15* et ce en vu de minimiser le plus possible la probabilité de blocage et de rejet d'une demande de connexion. J'ai mis dans les trois figures les résultats des simulations enregistrées aussi avec la première heuristique, et on remarque clairement que les deux heuristiques diffèrent sur leurs objectifs : la première heuristique assure une utilisation plus efficace des ressources en diminuant la charge inutile appliquée sur le réseau et l'autre essaye de minimiser la charge appliquée sur le lien le plus congestionné au prix d'une faible augmentation de la charge totale utilisée.

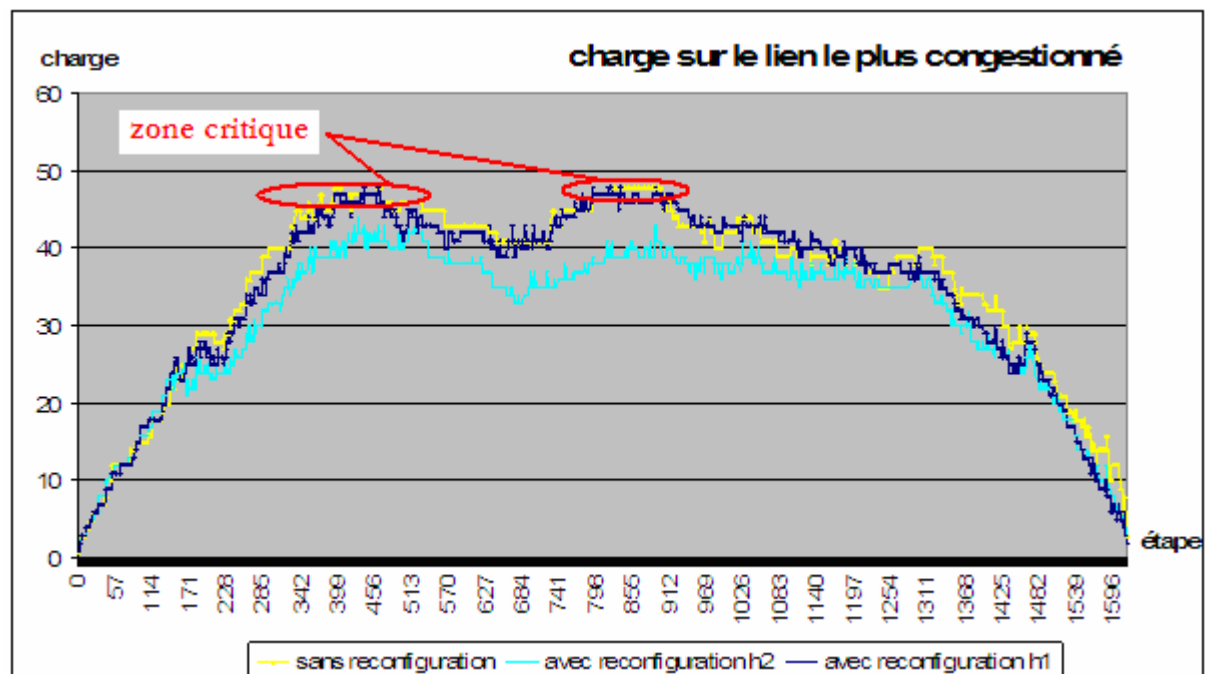


Figure3.15 : charge appliquée sur le lien le plus congestionné avec ou sans reconfiguration

La *figure3.18* montre que la première heuristique effectue plus d'actions de reconfiguration que la deuxième heuristique pour un gain très faible. Voir les figures : *figure3.13*, *figure3.14* et *figure3.15*.

La deuxième heuristique cible les liens les plus congestionnés. Son objectif est de réduire leurs taux d'occupation courant, ce qui n'est pas toujours une chose réalisable et il faut parfois attendre quelques

changements pour réussir à diminuer ce taux, alors que la première heuristique cible toutes les connexions pouvant être déplacé et rapportant une diminution de coût : un objectif moins contraignant que le précédent et plus fréquent au niveau de la faisabilité. L'apport de la première heuristique se distingue sur les *figure3.13* et *figure3.14* au niveau de la diminution de la charge globale appliquée, mais aussi sur un autre critère qu'on n'a pas observé sur les courbes et qui justifie le nombre de reconfigurations enregistrés :

- Plusieurs connexions peuvent être ciblé pour reconfiguration par notre heuristique sans qu'on remarque des changements sur les courbes dessinées ;

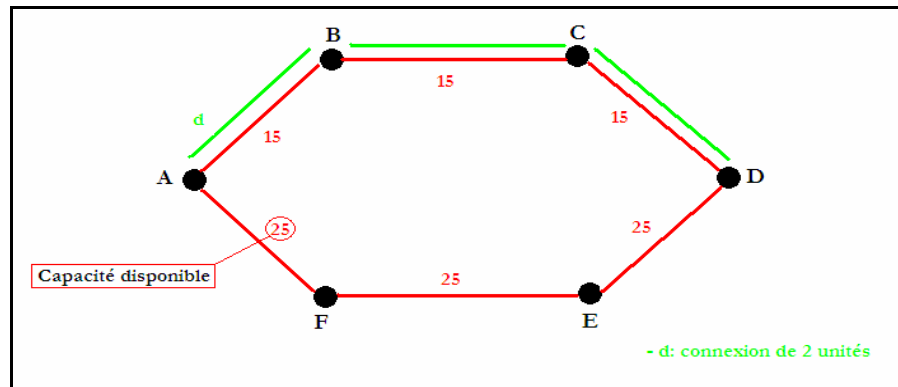


Figure3.16 : Exemple de choix de chemin avec une fonction parabolique

Exemple : une connexion de 2 unités de capacité déjà routé sur le chemin " $A \rightarrow B \rightarrow C \rightarrow D$ " donc appliquant 6 unités de charge sur le réseau peut avec notre heuristique de reconfiguration être déplacé sur le chemin " $A \rightarrow F \rightarrow E \rightarrow D$ " mais sans aucune diminution pour la charge appliquée sur le réseau et ne pouvant être visualisé sur la *figure3.15* tant que le chemin ne comporte pas le lien le plus le plus congestionné.

Avec notre fonction de coût à croissance « *parabolique* » (peu pénalisante), ce ne sont pas les liens les plus congestionnés qui sont généralement concernés par les reconfigurations ; exemple : sur la *figure3.17*, une demande de connexion entre les nœuds A et B empruntera le chemin " $A \rightarrow B$ " lui coûtant : 2209 mais en contre partie surchargeant le lien " $A \rightarrow B$ ". Un autre choix écarté par cette fonction est le chemin " $A \rightarrow C \rightarrow D \rightarrow B$ " lui coûtant : 4800 plus cher que le précédent. A présent on n'a pas présenté l'avantage, on a juste présenté l'inconvénient de la première heuristique et ainsi justifié pourquoi on ne remarque aucune amélioration sur la *figure3.15* pour cette heuristique.

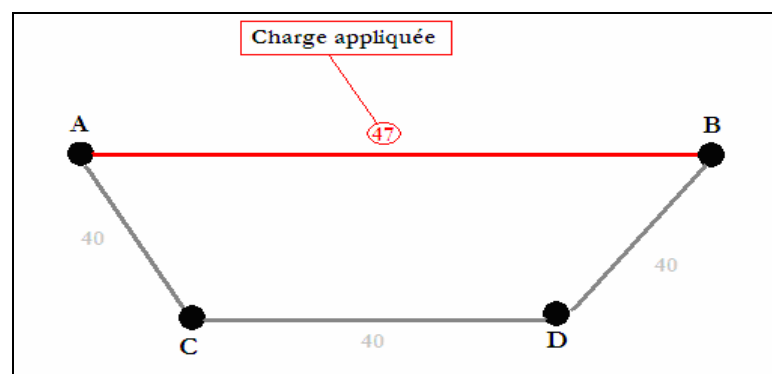


Figure 3.17 : inconvénient de la première heuristique

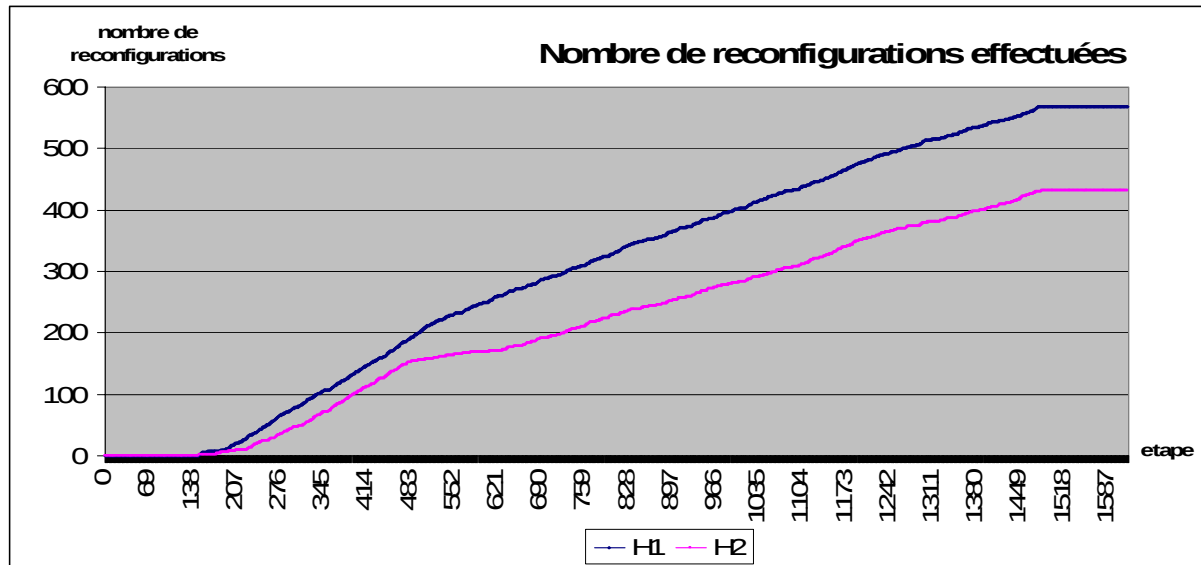


Figure3.18 : Nombre de reconfigurations effectuées par H1 et H2

A ce stade et à mon avis la deuxième heuristique est beaucoup plus intéressante si on est face à un trafic dense amenant l'ensemble des liens du réseau à se rapprocher de la zone de congestion (voir en rouge sur la figure 3.15) et la première plus efficace en cas d'un trafic peu dense appliqué sur le réseau.

3.3.2 – 2^{ème} classe d'heuristiques de reconfigurations :

Comme on a déjà présenté précédemment, la deuxième classe d'heuristiques de reconfiguration est la classe où on permettra non plus une seule action de reconfiguration par étape mais plusieurs en même temps. L'objectif est de vérifier l'efficacité ainsi que la nécessité d'un tel choix au prix de nombre de reconfigurations à réaliser.

Dans ce contexte [2] propose une première solution à base d'une technique incrémentale assurant une solution acceptable suite à un certain nombre d'actions de reconfiguration. La solution consiste à prendre pour chaque paire de connexions déjà établis sur le réseau l'ensemble des chemins possibles qu'ils peuvent emprunter sur le réseau et essayer de trouver dans cet ensemble une paire de chemins assurant une solution meilleure que la solution actuelle. Si une solution a été trouvée alors une reconfiguration est requise. Sinon, on passe à une autre paire de demandes. La condition d'arrêt est le fait de traiter toutes les demandes sans trouver aucune amélioration.

Les études de [2] parlent d'une convergence rapide vers la solution optimale : leur solution optimale est celle donnée par le programme linéaire qu'on utilise pour le routage des demandes de connexion. Pour leurs cas, les coûts des chemins sont fixes et ils essaient à partir d'une instance de trafic mal routé de trouver avec le minimum de déplacements la solution optimale.

Notre cas de figure est loin de coïncider avec leurs cas : en effet, vu que les coûts de nos chemins sont modifiables suivant l'occupation des liens qui les constituent, on se retrouve avec un grand nombre de changements sans pouvoir affirmer qu'on a la solution optimale.

L'avantage de cette solution est qu'elle permet à chacune des connexions (à déplacer) d'utiliser les unités des capacités libérées temporairement par les autres connexions (à déplacer). En effet, si on a notre réseau avec son état actuel : occupation des liens et leurs coûts. Si on enlève une connexion « c1 » alors cette dernière ne peut profiter que des capacités disponibles, tandis que si on enlève une autre connexion « c2 » au même temps un meilleur arrangement peut être envisageable.

Voici un exemple illustratif permettant de visualiser l'avantage d'élimination plus de connexion pour trouver un meilleur arrangement des connexions. La fonction de coût utilisée est une fonction à croissance parabolique.

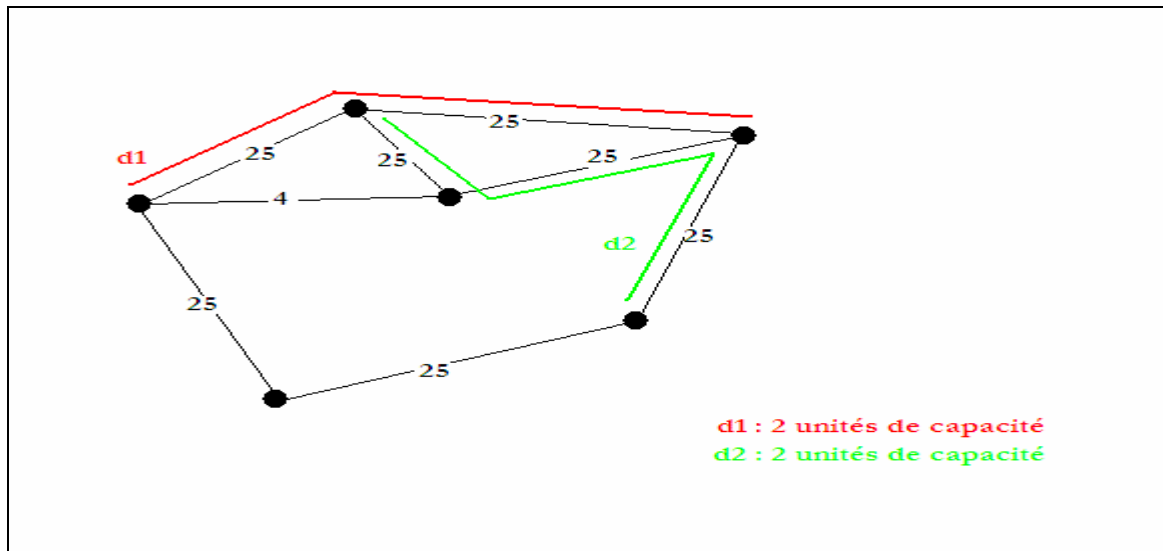


Figure 3.19 : exemple illustratif de l'avantage de reconfiguration par plusieurs connexions (1)

Si on enlève la connexion « *d1* » alors le coût de chemin qu'elle emprunte sera égal à 18 et sera le coût le moins cher. Même chose si on élimine la connexion « *d2* », le coût du chemin qu'elle emprunte sera égal à 27 et sera le coût le moins cher aussi.

Tandis que si élimine les deux connexions en même temps, un réarrangement possible se distingue comme suit :

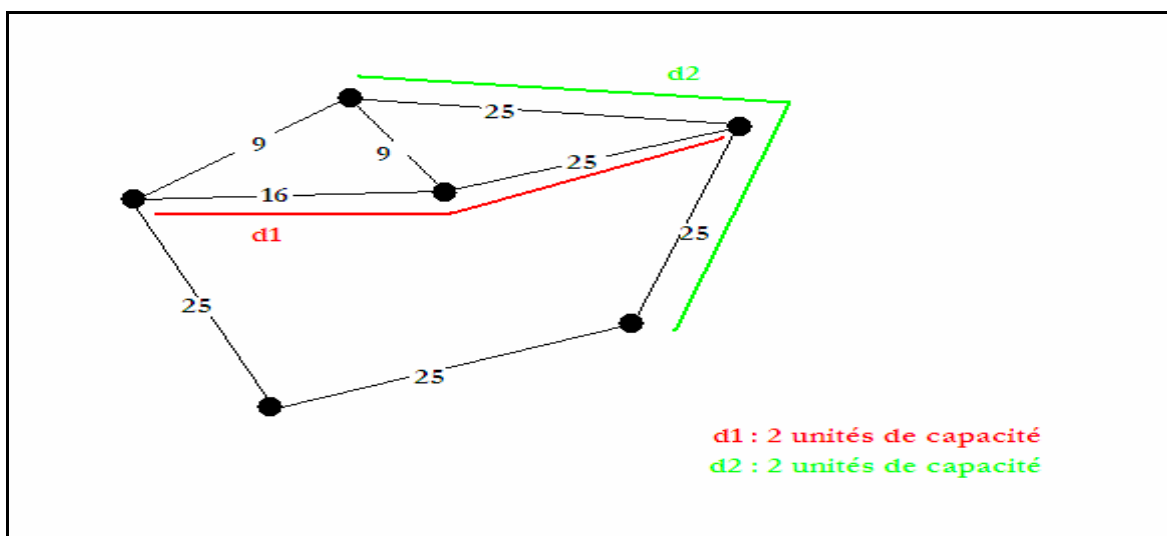


Figure 3.20 : exemple illustratif de l'avantage de reconfiguration par plusieurs connexions

Avec ce réarrangement, pour la connexion « *d1* » un chemin de coût 13 a été attribué et pour la connexion « *d2* » un chemin de coût 18 a été attribué. Donc une meilleure solution a été trouvé.

Dans cette partie on va s'intéresser à appliquer deux idées de reconfiguration basées sur plusieurs actions de re-routage de connexions par étape, si on a besoin évidemment.

3.3.2.1- 1^{er} Heuristique :

Dans la première partie on a mis en place deux heuristiques appliquant juste une seule action de reconfiguration par étape. Toutefois il est possible d'appliquer plusieurs déplacements à la même étape. Ainsi dans cette première heuristique on s'intéressera à mettre en place tous les changements possibles qu'on trouvera à une étape donnée. Les connexions qu'on re-routera seront extraites séquentiellement en se servant de la première et de la deuxième heuristique déjà élaborées au niveau de la section intitulé « Première classe d'heuristiques ».

A- Simulation et résultats :

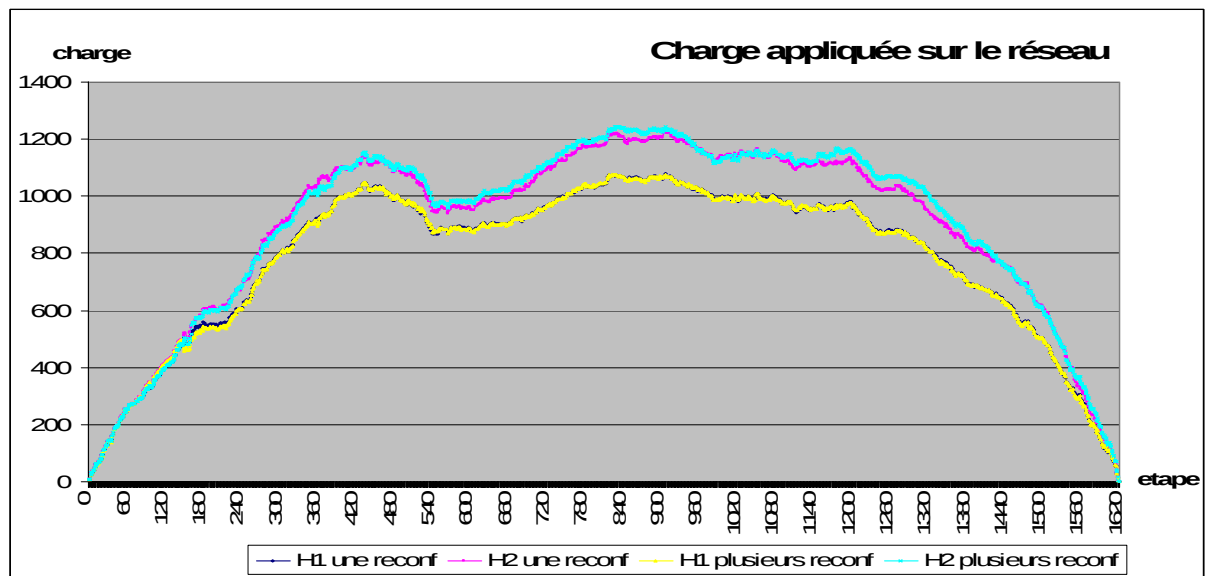


Figure3.21 : Charge appliquée sur le réseau

L'évolution de la charge appliquée sur le réseau ne marque aucun changement pour la première heuristique, et une petite augmentation pour la deuxième heuristique : cette petite augmentation est bien visible entre l'étape 540 et l'étape 1440.

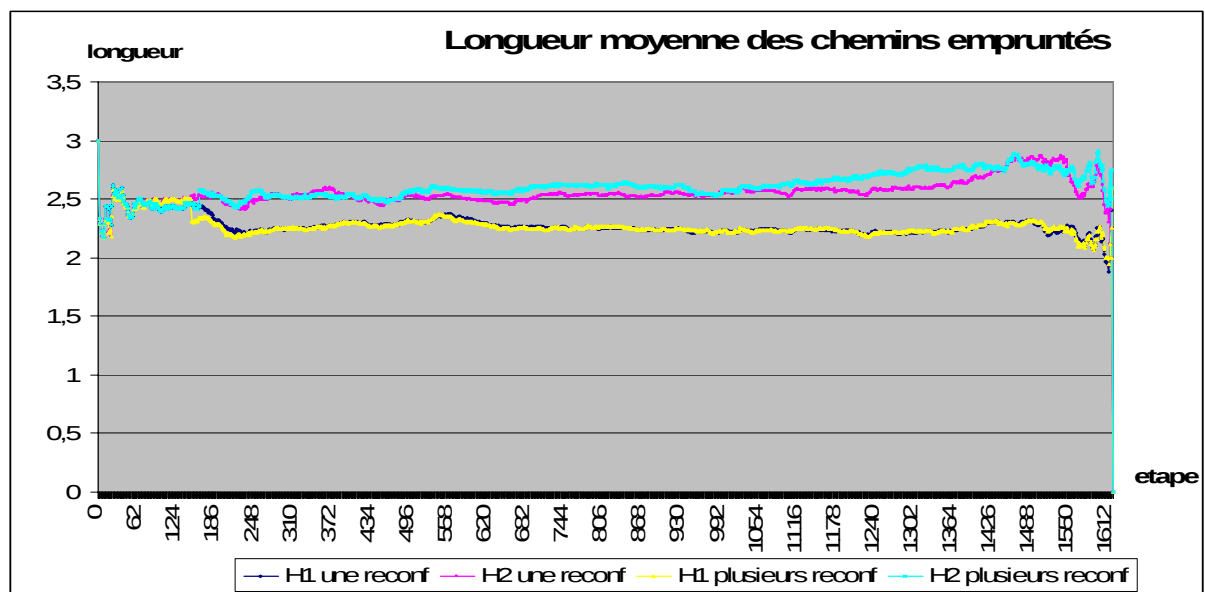


Figure3.22 : longueur moyenne des chemins empruntés

Les courbes figurantes sur le schéma précédent confirment les résultats visualisés sur la figure représentant l'évolution de charge globale appliquée. En effet, la longueur moyenne des chemins empruntés pour la première heuristique ne marque aucun changement par rapport à l'exécution d'une seule action de reconfiguration, tandis que la deuxième heuristique admet une augmentation si claire pour les chemins empruntés.

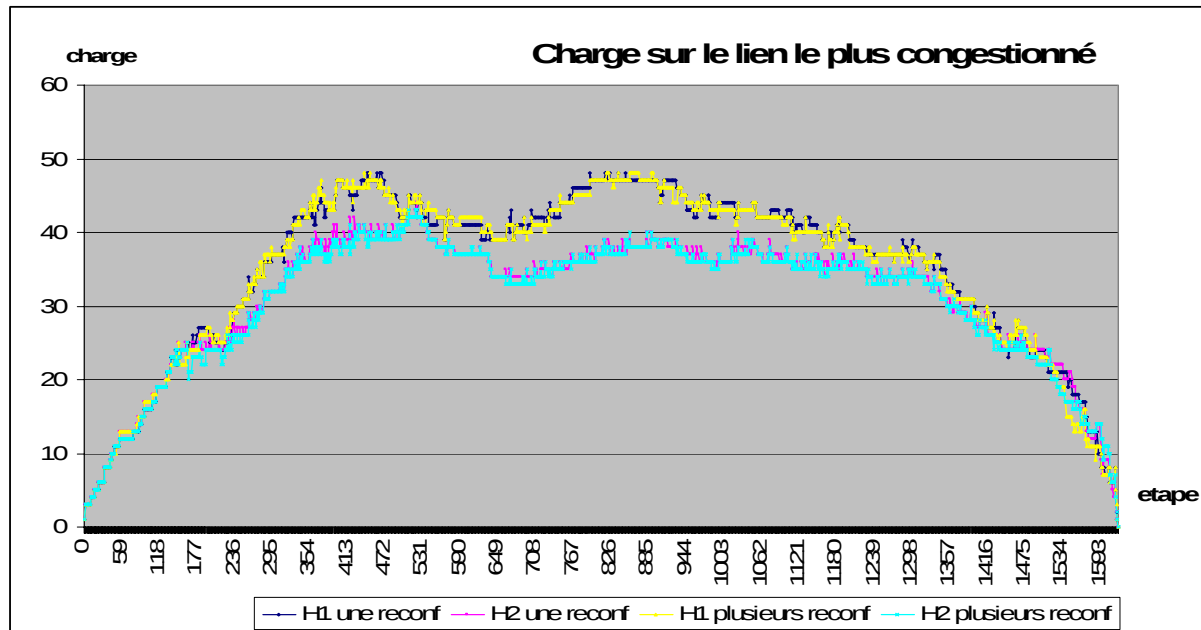


Figure3.23 : Charge appliquée sur le lien le plus congestionné

Confirmant les résultats illustrés précédemment, la *figure3.23* marque une légère amélioration au niveau de la deuxième heuristique et presque une stabilité pour la première heuristique. La question qui se pose maintenant est : l'utilisation de la première heuristique en effectuant une seule reconfiguration ou plusieurs en même étape donne les mêmes performances. Mais est ce que ces deux modes d'utilisation de cette première heuristique coûtent le même prix ?

Pour répondre à cette question on a essayé de se focaliser un peu sur le nombre de reconfigurations cumulés tout au long de la simulation et le résultat était le suivant :

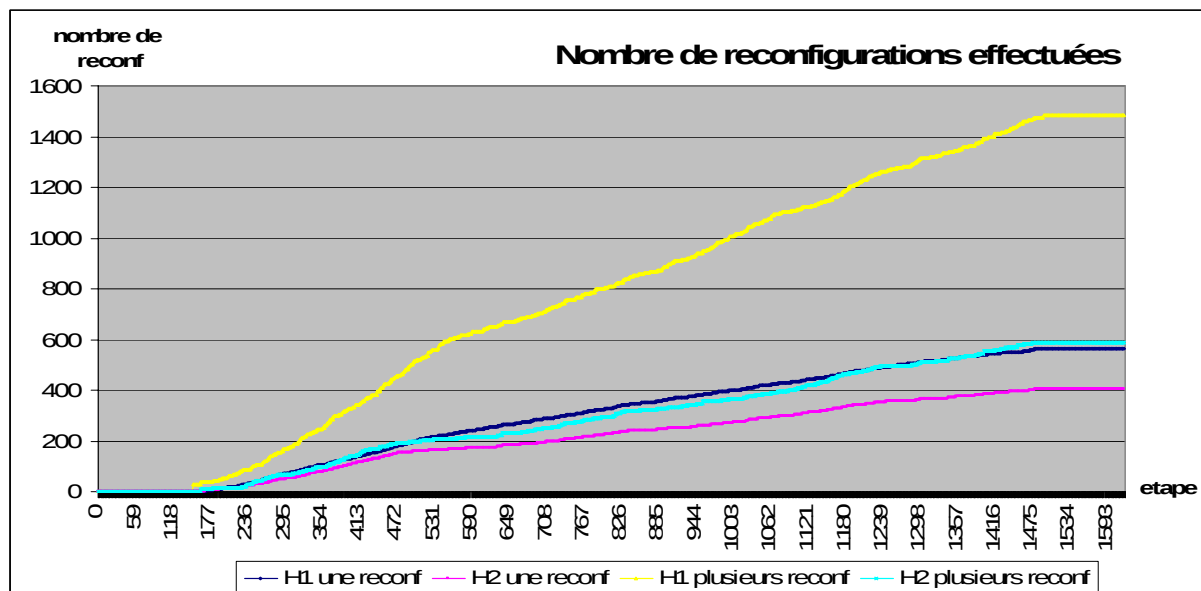


Figure3.24 : nombre de reconfigurations cumulés tout au long de la simulation

Pour la deuxième heuristique, la légère augmentation des charges appliqués sur le réseau, la longueur moyenne des chemins empruntés et la diminution de la charge appliquée sur le lien le plus congestionné sont justifiés par une augmentation du nombre de reconfigurations total.

A l'inverse, la première heuristique marque une augmentation importante du nombre de reconfigurations effectuées. Une explication possible est que l'amélioration apportée a touché des liens qui sont parfois loin de la zone de congestion et des connexions dont les anciens chemins et les nouveaux chemins ont la

même longueur. Pour vérifier cette hypothèse, j'ai représenté l'évolution du nombre de connexions déplacées de leurs chemins originaux pour de nouveaux chemins ayant la même longueur.

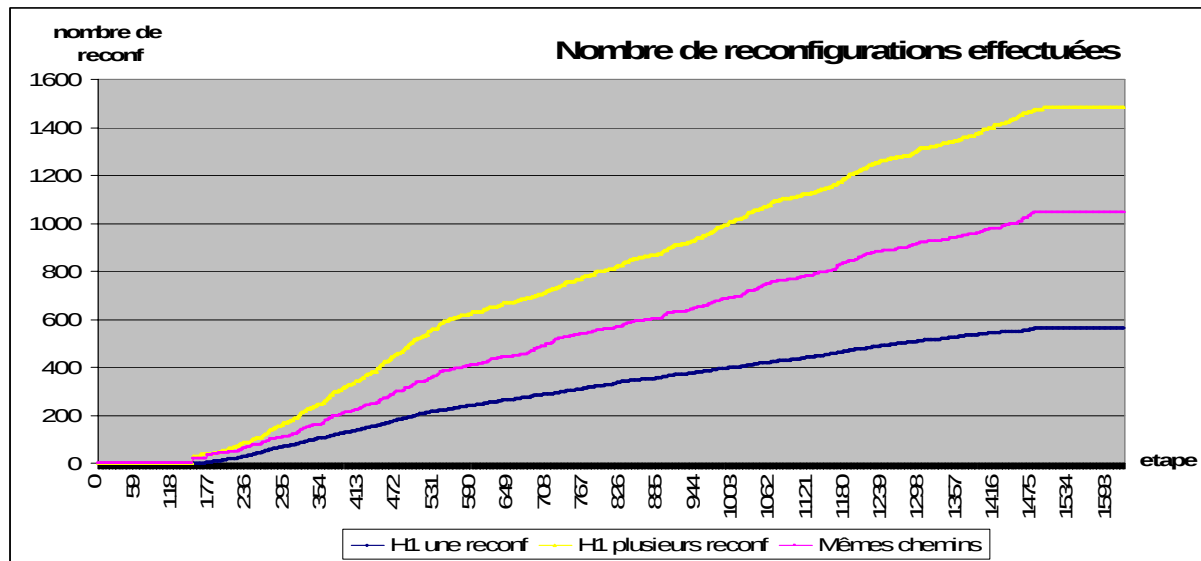


Figure 3.25 : nombre de reconfigurations cumulées tout au long de la simulation pour H1

Le nombre très grand des reconfigurations portant sur des connexions déplacés vers des chemins de même longueur que leurs chemins originaux confirme l'hypothèse proposée. Ainsi la première heuristique est une heuristique dont l'objectif est d'établir un certain équilibre général et les améliorations touchent tous les liens sans exception, ne qualifie intéressante mais pour un coût non négligeable à savoir un nombre important de reconfigurations.

Généralement ces types de reconfigurations ne sont pas très appréciés tant qu'on n'arrive pas à en tirer profit.

La question qui se pose maintenant, vue le nombre élevé de reconfigurations enregistré par rapport à la qualité et les performances remarquables de la solution : y'a-t-il des modifications à apporter à la solution initiale pour assurer un rapport acceptable (*nombre de reconfigurations, qualité de la solution*).

B- Améliorations apportées :

- Utiliser une fonction de coût moins sensible aux changements que subissent les ressources du réseau. Exemple : fonction de coût *discretisé*.
- Ajouter quelques contraintes lors de la prise de décision des actions de reconfigurations afin de ne permettre que les changements les plus significatifs.

La première proposition a été écartée et ce pour la raison suivante : une fonction discrétisée nécessite un choix judicieux pour ses bornes limites des intervalles de classification de coût, un choix non évident et variant suivant la densité du trafic. De plus cette fonction n'a pas pu assurer un routage acceptable et efficace lors de la première partie (comparaison des comportements des différentes fonctions de coût face à un trafic dynamique). Aussi, un changement de la fonction de coût à cette étape marquera une amélioration (conditionné par le choix des intervalles) mais dégradera la qualité de la solution en terme de performance.

La deuxième proposition a été retenue et dans la partie qui suivra, la classification suivante à base des longueurs chemins reconfigurés sera utilisée :

Une connexion « *c1* » établie sur un chemin « *ch1* » peut être re-router sur un chemin :

- « *ch2* » de même longueur que « *ch1* ».

- « *ch3* » de longueur inférieure que « *ch1* ».
- « *ch4* » de longueur supérieure que « *ch1* ».

Les contraintes ajoutées sont les suivantes :

1- Pour un chemin moins long « *ch3* », la contrainte ajoutée pour valider la reconfiguration est : si un seuil « *s_max* » est dépassé sur l'un des liens du nouveau chemin alors il doit également être dépassé sur l'ancien chemin pour que la reconfiguration soit validée. Dans les autres cas la reconfiguration est validée sans contrainte. Cette contrainte limite la minimisation de la charge globale en vue d'assurer plus d'équilibre sur les liens.

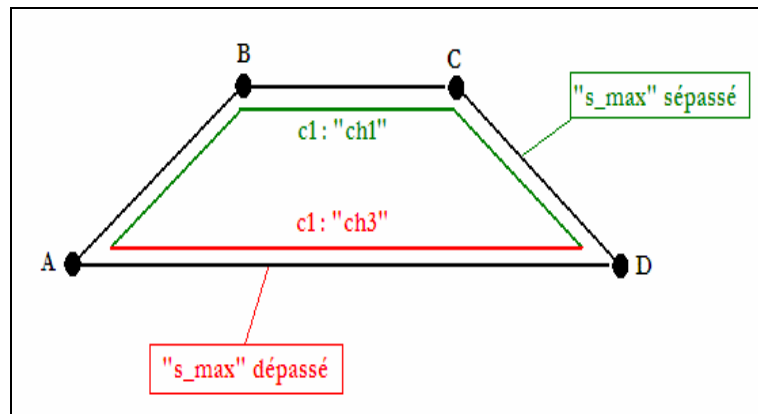


Figure3.26 : cas de reconfiguration validée

2- Pour un chemin plus long « *ch4* », la contrainte ajoutée pour valider la reconfiguration est : il suffit de mesurer les deux seuils d'occupation maximaux enregistrés sur les deux chemins et si « *seuil_max_ch1* » est inférieur à « *seuil_max_ch4* » alors la reconfiguration est annulée et pour l'autre cas elle est validée. Mais toute action de reconfiguration dans ce contexte n'est déclenchée que si sur l'un des liens du chemin « *ch1* » le seuil « *s_max* » a été dépassé.

3- Pour qu'une reconfiguration de type « *ch1* » vers « *ch2* » (chemin de même longueur) ait lieu, les deux chemins doivent satisfaire :

- Le seuil « *s_max* » est dépassé sur au moins un des liens « *ch1* ».
- le seuil maximal enregistré sur le chemin « *ch1* » est inférieur au seuil maximal enregistré sur le chemin « *ch2* ».

En fonction des valeurs du seuil « *s_max* » fixé on trace maintenant l'évolution de la charge globale appliquée ainsi que le nombre de reconfigurations cumulées.

A- Simulation et résultats :

Voici la figure représentant l'évolution de la charge globale appliquée sur les réseaux pour chacun des valeurs des seuils « *s_max* » choisis.

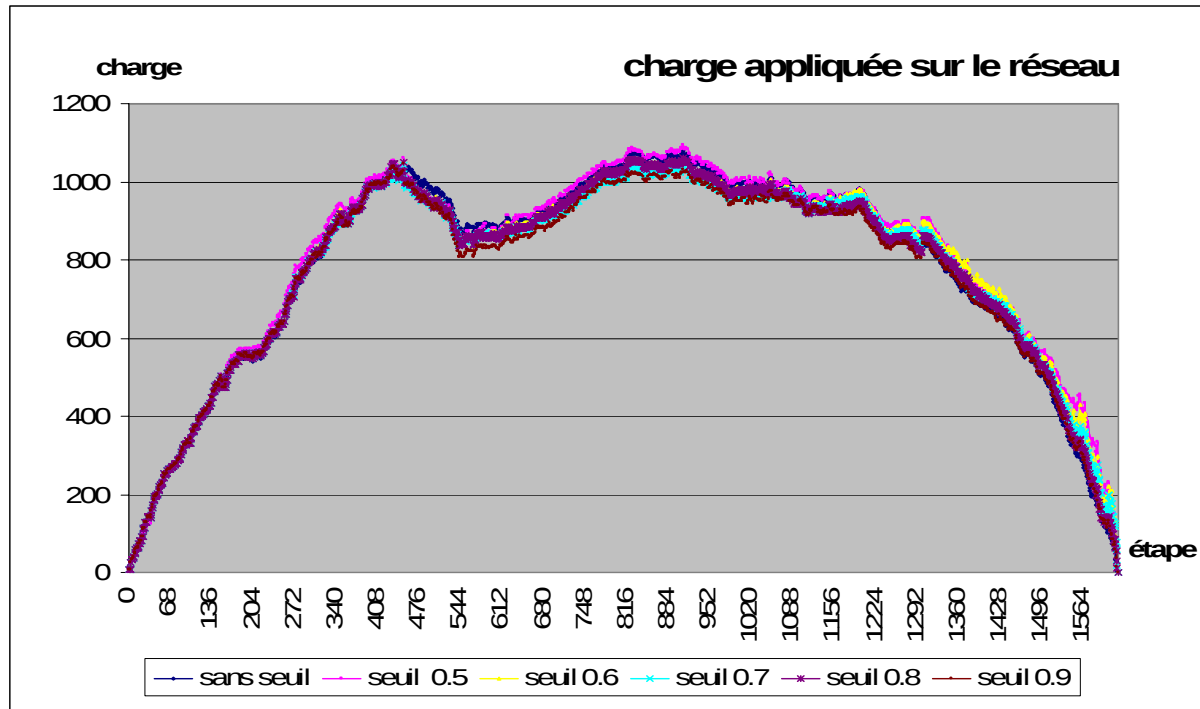


Figure 3.27 : Evolution de la charge appliquée sur le réseau

En regardant cette figure, on n'est certain que la performance et la qualité de la solution proposée ne sont pas dégradées avec nos contraintes intégrées et au contraire on peut même affirmer qu'ils ont été améliorés. Vu le très grand rapprochement des valeurs enregistrées, on remarque à peine que plus on augmente le seuil « s_{max} » plus on diminue la charge globale appliquée sur le réseau. Mais on peut voir ça plus clairement dans le tableau suivant :

Seuil	0	0.5	0.6	0.7	0.8	0.9
Charge moyenne	832,5	829	814	803	802	788

On remarque clairement que les valeurs des charges moyennes appliquées sur le réseau décroissent lorsqu'on augmente le seuil « s_{max} ». Ce résultat était prévisible, puisqu'en augmentant « s_{max} » la validation de quelques types de reconfigurations devient plus difficile à savoir : basculer une connexion d'un chemin vers un autre chemin de même longueur ou de longueur plus grande. Ce raisonnement est vérifié à travers le tableau suivant :

Seuil	0	0.5	0.6	0.7	0.8	0.9
Même longueurs	540	228	178	72	34	13
Plus court	222	369	352	207	142	114
Plus long	186	113	116	76	77	17

Dans un premier temps on remarque qu'on gagne considérablement en terme de charge appliquée sur le réseau en augmentant le seuil « s_{max} » mais on n'a pas encore exposé l'impact de telle supposition sur l'évolution de nombre de reconfigurations effectuées.

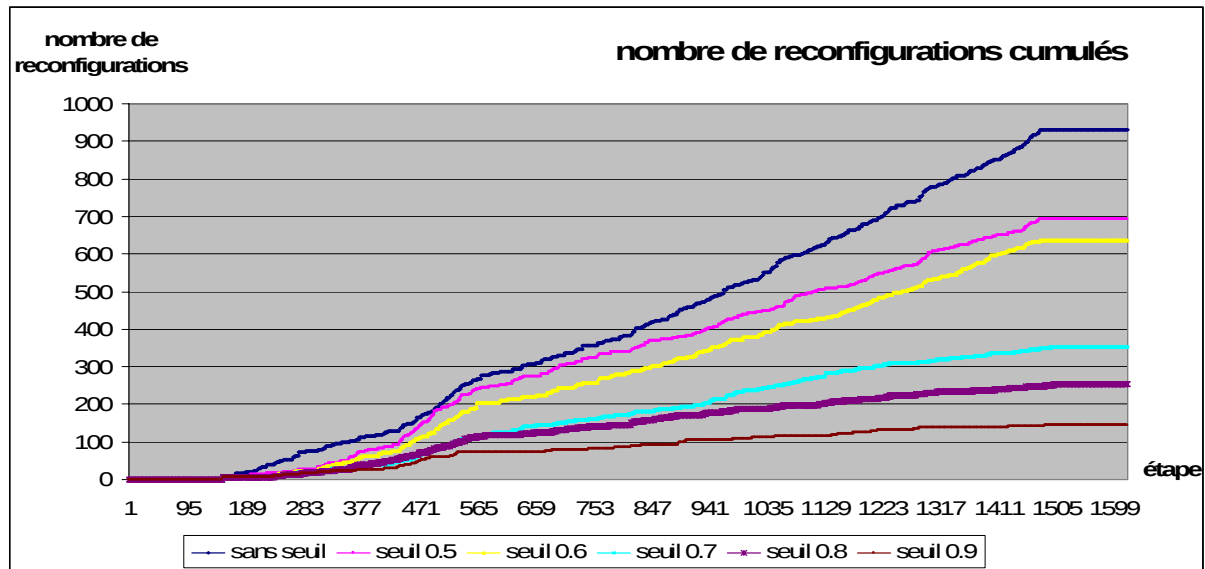


Figure 3.28 : nombre de reconfigurations cumulés tout au long de la simulation pour H1

On remarque clairement que d'une manière cohérente on arrive à diminuer le nombre de reconfigurations en augmentant le seuil « s_{max} ».

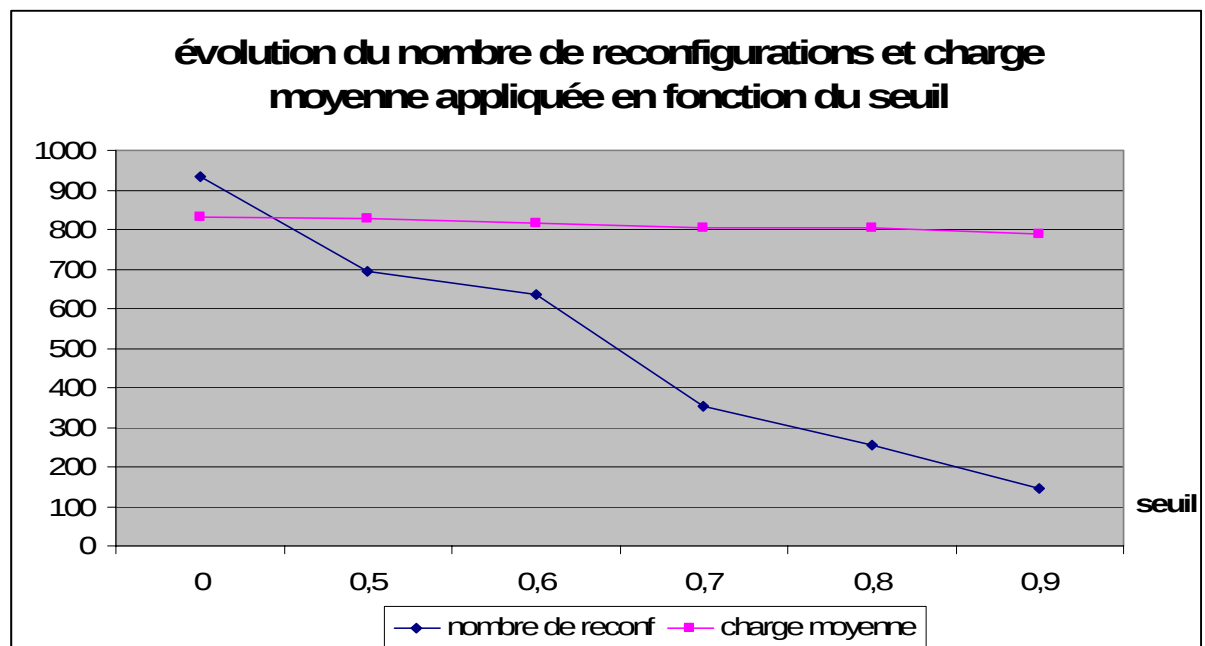


Figure 3.29 : reconfigurations cumulés et charge moyenne appliquée en fonction du seuil

La figure 3.29 montre clairement l'apport positif des contraintes intégrées avec des seuils élevés soit sur la charge moyenne globale appliquée soit sur le nombre de reconfigurations effectués. Ainsi on peut conclure qu'il sera toujours plus avantageux dans notre cas de figure d'augmenter les seuils puisqu'ils nous éviteront d'effectuer de nombreuses reconfigurations inutiles et tout en assurant de bonnes performances pour la charge moyenne appliquée.

En guise de conclusion pour cette première partie de la deuxième classe de reconfiguration, on a trouvé que pour notre cas de figure (graphe plus modèle du trafic) :

- Il n'est pas nécessaire d'effectuer plusieurs reconfigurations à une même étape puisque la qualité de la solution résultante est presque similaire.

- La première heuristique entraîne un très grand nombre de reconfigurations qu'on a jugé non nécessaire. Ainsi, on a intégré quelques contraintes à fin de n'accepter que les reconfigurations les plus significatives. De telles contraintes assureront sûrement plus d'efficacité pour un cas de figure d'un trafic plus dense où une seule reconfiguration par étape ne pourra plus suffire pour garantir la survie et le bon fonctionnement du réseau.

3.3.2.2- 2^{ème} Heuristique :

Pour la deuxième heuristique de cette partie, on a prévu faire face au même problème rencontré précédemment (nombre élevé de reconfigurations effectuées) et ce vu que les deux heuristiques se basent sur l'amélioration apportée à partir des gains en coût des chemins. Ainsi, pour assurer un meilleur compromis entre la qualité des résultats et le nombre de reconfigurations effectués, on a choisi d'intégrer, de même que précédemment, nos contraintes avec un seuil fixé à une valeur de 0.9.

A partir des simulations on a remarqué clairement que :

- Cette technique est très coûteuse en temps de calcul.
- Cette technique entraîne plus d'actions de reconfigurations que la précédente du fait qu'elle procède à la reconfiguration d'une manière incrémentale. 254 reconfigurations ont été enregistrés dont 180 concerne des connexions qu'on a basculé vers des chemins plus courts que leurs chemins originaux.

La figure suivante visualise l'évolution de la charge totale appliquée pour un scénario de connexions établies sans reconfiguration, avec une seule reconfiguration par étape et avec plusieurs par étape en utilisant notre heuristique:

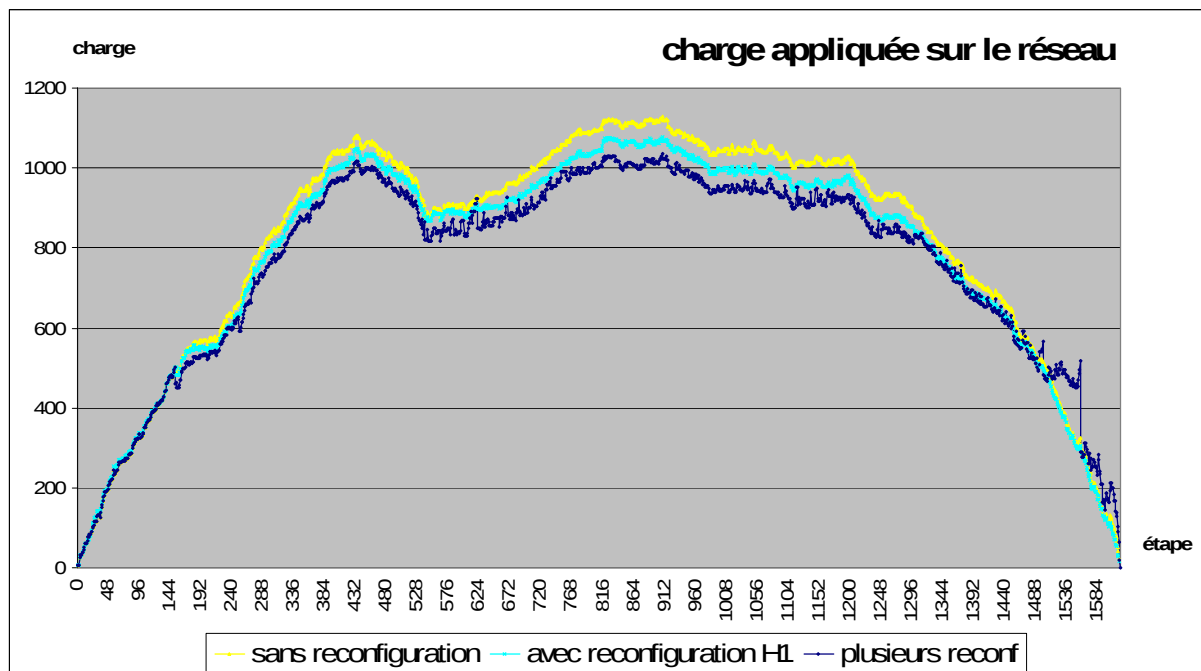


Figure 3.30 : charge totale appliquée sur le réseau pour plusieurs reconfigurations par étape

La technique incrémentale a réussi à diminuer légèrement la valeur de la charge moyenne globale enregistrée par rapport à l'heuristique précédente. En effet, une valeur de 785 unités de capacité en moyenne a été observée. Cette dernière partie, vient confirmer l'efficacité de la politique d'une seule reconfiguration par étape pour le contexte de réseaux et de trafic utilisé.

Dans la partie suivante, on exposera la problématique de protection des connexions sur un réseau et on retournera en fin à la reconfiguration à travers l'application des deux heuristiques de reconfiguration déjà élaborés sur un scénario de connexions établies et protégés sur un réseau.

Chapitre 4

PROBLEME DE PROTECTION

Dans un réseau WDM une panne, soit d'un « OXC » (nœud) ou d'une ligne de communication, cause une très grande perte de données. Pour éviter une telle perte, une protection des chemins utilisés par les connexions se distingue pertinente.

Le problème de la protection des chemins est l'un des problèmes les plus intéressants à résoudre et surtout dans le contexte des réseaux WDM où une panne quelconque cause des pertes de données de l'ordre des Gbits. L'objectif de la résolution d'un tel problème est d'assurer la continuation de fonctionnement efficace d'un réseau même en cas de pannes (situations bien définies). L'objectif de la protection peut s'étendre à assurer la non congestion du réseau suite à une panne et ce afin de pouvoir accueillir normalement de nouvelles demandes de connexion.

Différentes variantes de solutions ont été introduites dans la littérature pour résoudre le problème de protection. En premier lieu on va les exposer et après le travail consistera à extraire et à appliquer les solutions qu'on peut adapter à nos cas de figure.

Assurer la protection des chemins empruntés par les connexions est un problème composé de deux sous problèmes [6], le premier concerne la protection des chemins de base et c'est auquel on va s'intéresser tout au long de cette partie et le deuxième concerne la restauration des chemins : tous les éléments relatifs à la découverte de la panne par les deux nœuds source et destination tel que le protocole de communication et le temps mis pour basculer du chemin/lien vers les solutions de secours.

4.1- Problème de protection des connexions :

La résolution du problème de protection des chemins peut être faite soit avec les ressources dédiées, soit avec une restauration dynamique.

- 1- *Protection d'un chemin avec des ressources dédiées* : Dans cette classe de protection, des ressources déjà existantes et qui ont été réservés sont dédiés pour différents scénarios de panne. Cette classe, inclut évidemment les sous classes de protection à savoir dédiées ou partagées qu'on détaillera dans de la prochaine section.
- 2- *Protection d'un chemin avec restauration dynamique* : l'idée de base consiste à restaurer des connexions qui ont subi une interruption de service suite à une panne en utilisant les ressources disponibles dans le réseau au moment de la panne.

La différence principale entre ces deux classes est que dans la première la restauration est plus rapide vu la réservation établie au préalable pour tout type de panne sur les chemins empruntés par les connexions tandis qu dans la deuxième réserve juste une quantité de ressources pour être utiliser en cas de panne mais non dédié pour des cas précis de panne. Ainsi, une opération de recherche et de calcul est exigée pour restaurer les connexions interrompues.

Dans tout ce qui va suivre on ne va s'intéresser qu'au problème de protection et plus précisément, on portera tout notre effort sur la classe de protection des chemins avec des ressources dédiées.

4.2- Protection à base de ressources dédiées :

La protection à base de ressources dédiées est divisée en deux sections à savoir :

- La protection dédiée : pour chaque chemin « $ch1$ » choisi pour une connexion « $d1$ » il faut réserver un chemin « $ch2$ » permettant également d'accommoder « $d1$ » et tel que $ch1 \cap ch2 = \Phi$. Ce chemin devra rester réservé pour remplacer le chemin « $ch1$ » en cas d'une panne détectée sur l'un des liens de ce dernier. Ainsi les ressources de ce chemin de réserve ne peuvent être libérées qu'après la terminaison de la connexion et en aucun cas ne peuvent être partagé avec aucun autre chemin de réserve.
- La protection partagée : pour chaque chemin « $ch1$ » choisi pour une connexion « $d1$ » il faut réserver un chemin « $ch2$ » permettant également d'accommoder « $d1$ » et tel que $ch1 \cap ch2 = \Phi$. Mais la différence avec le cas de la protection dédiée réside dans le fait que les unités de capacité réservée sur un chemin de protection peuvent être partagée avec d'autres chemins de protection dont les chemins primaires ou de base n'admettent pas des liens en commun entre eux

Dans la définition des classes de protection, on a parlé que de chemins mais les deux classes peuvent être appliquées aussi aux liens : en effet, [6] parle aussi de la protection dédiée des liens et de la protection partagée des liens et leurs définitions sont pratiquement similaire à celles des cas des chemins.

Une autre technique de protection connue sous le nom de « *protection partielle de chemin* » [7] consiste à réserver des chemins de protection autres que le chemin primaire lors de la mise en place d'une connexion. La différence entre cette technique et la technique de protection de chemin est qu'ici :

- un chemin de réserve est choisi pour chaque cas de panne d'un des liens du chemin primaire.
- Les chemins de réserve peuvent être partiellement disjoints du chemin de base.

La différence entre cette technique et la technique de protection de lien est qu'ici le chemin de réserve pour chacun des liens peut ne pas traverser les nœuds extrêmes du lien en panne tandis que dans le cas de protection de liens ce passage est exigé.

Chacune de ces techniques a des avantages et des inconvénients l'une par rapport à l'autre. Généralement les critères de performance sont les suivants :

- la charge appliquée pour assurer la protection : le critère le plus important et le plus coûteux en terme de ressources.
- La faisabilité des techniques : en effet des techniques de protection sont plus élastiques et souple que d'autres. En effet, voici un exemple illustratif où en utilisant la technique de protection de chemin on n'arrive pas à trouver une solution et en utilisant la technique de protection partielle de chemin on arrive à assurer une certaine protection pour le chemin primaire.

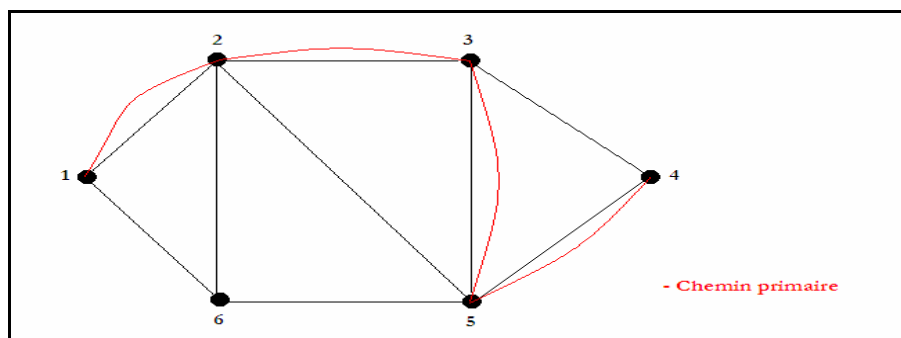


Figure 4.1 : exemple illustratif de faisabilité des techniques de protection

Dans l'exemple de la *figure 4.1*, une connexion est établie entre le nœud 1 et le nœud 4 dont le chemin primaire est affiché en rouge. Avec la technique de protection de chemin qui exige le choix d'un chemin de réserve complètement disjoint du chemin primaire, on n'arrive à trouver aucune solution satisfaisant cette contrainte.

Avec la technique de protection partielle de chemin, une des solutions possibles est la suivante :

Lien en panne sur le chemin primaire	Chemin de réserve relatif
1-2	1-6-2-3-5-4
2-3	1-2-5-4
3-5	1-2-5-4
5-4	1-2-3-4

Dans cet exemple la technique de protection partielle des chemins est plus souple et performante que la solution de protection de chemin avec laquelle on n'est pas arrivé à assurer un chemin de protection.

[8] évoque une comparaison entre les techniques de protection des connexions par chemins et par liens avec une technique de protection des connexions par segments. Cette technique consiste, comme son nom l'indique, à assurer la protection du chemin de base en procédant à le sécuriser segment par segment. Les longueurs des segments sont des variables à choisir, et ainsi les deux cas de protection des connexions par chemins ou par liens ne sont que des cas figures spéciales de la protection des connexions par segments. L'étude expose l'apport observé par cette dernière technique soit en terme d'utilisation de ressources soit en terme de temps mis pour la restauration.

Dans une première partie on s'intéressera à tout mettre en place le programme linéaire pour la protection dédiée.

4.2.1- Protection dédiée des chemins :

Le programme linéaire est le suivant à partir d'une formulation extraire de [2] :

- $T = \{0, 1, 2, \dots, T_{\max}\}$: l'ensemble des étapes de la simulation.
- $R = \{R_1, R_2, R_3, \dots, R_{\max}\}$ l'ensemble des requêtes de demande de connexion du scénario, chaque demande de connexion est identifié par une source et une destination.
- $V = \{V_1, V_2, V_3, \dots, V_n\}$ l'ensemble des nœuds constituant le graphe.
- $ch_{ij} = \{ch_1, \dots, ch_p\}$, avec p : le nombre de chemins possible entre les deux noeuds i et j .
- $CH = \{ch_{12}, ch_{13}, \dots, ch_{ij}\}$, avec $i, j = 1, 2, \dots, n$ représentant les nœuds du graphe; CH représente l'ensemble des ensembles de ch_{ij} qui définissent les chemins existants entre tous les paires de nœuds i et j (pour $i = j$ on a $ch_{ij} = \emptyset$).
- $H = \{(p, q), \dots\}$, avec (p, q) : représentent l'ensemble des paires de chemins ayant au moins un lien en commun ; $p \cap q \neq \emptyset$.
- $x_{req, ch} = \{0, 1\}$; variable binaire indiquant l'utilisation ou non d'un chemin par une demande. Cette variable de décision permettra la distinction du chemin le moins coûteux emprunté par la demande d .
- $y_{req, ch} = \{0, 1\}$; variable binaire indiquant l'utilisation ou non d'un chemin par une demande comme étant un chemin de sécurité.
- $R_E(t)$ = l'ensemble des requêtes générées à une étape t .
- V : ensemble de nœuds du réseau.
- A : Ensemble des arcs du réseau.
- $ch_{ij}^a : \{ch_1, \dots, ch_k\}$, avec p : le nombre de chemins possibles entre les deux noeuds i et j et passant par l'arc a .
- C^{ch} : coût relatif à l'utilisation d'un chemin « ch ».
- $c^a(t)$: capacité disponible sur un arc « a » à l'étape t .

Nous présentons ci-dessous le programme linéaire relatif à une étape unique. Ainsi, pour résoudre notre problème d'origine ce programme doit être reproduit sur chacune des étapes de la simulation :

$$\text{Objectif:} \quad \min \sum_{req \in R_E(t)} \sum_{ch \in ch_{ij}} C^{ch} * (x_{req,ch} + y_{req,ch})$$

- ij : les nœuds source et destination de la requête « req ».

Sous les contraintes :

$$\forall req \in R_E(t), \forall a \in A, \quad \sum_{req \in R_E(t)} \sum_{ch \in ch_{ij}} req_{size} * (x_{req,ch} + y_{req,ch}) \leq c^a(t) \quad (1.1)$$

$$\forall (ch1, ch2) \in H, \forall req \in R_E(t), \quad x_{req,ch1} + y_{req,ch2} \leq 1 \quad (1.2)$$

$$\forall req \in R_E(t), \quad \sum_{ch \in ch_{ij}} x_{req,ch} = 1 \quad (1.3)$$

$$\forall req \in R_E(t), \quad \sum_{ch \in ch_{ij}} y_{req,ch} = 1 \quad (1.4)$$

$$\forall req, ch, \quad x_{req,ch} = \{0,1\} \quad , \quad y_{req,ch} = \{0,1\}$$

Ce programme linéaire assurera en plus de l'attribution optimale des chemins aux demandes de connexion sous l'état courant du réseau, la protection des chemins de base de ces demandes de connexion par la réservation d'un ensemble de chemins de sécurité complètement disjoints des chemins de base.

L'objectif étant la minimisation du coût de la solution de routage et de protection, la contrainte n° 1.1 assure le respect de la capacité disponible sur chaque lien du réseau. La contrainte n° 1.2 assure que le chemin de base emprunté par la connexion et le chemin de protection réservé soient complètement disjoints. La contrainte 1.3 assure qu'une demande ne peut être routé qu'à travers un et un seul des chemins possibles entre les nœuds source et destination de cette demande. La contrainte 1.4 assure que pour une demande de connexion un seul des chemins possibles entre les nœuds source et destination de cette demande est réservé.

Pour mener à bien la phase du routage des demandes de connexion, il faut :

- Extraire tous (ou une partie) des chemins entre chaque paire des nœuds et affecter à chacun son prix initial.
- Classifier l'ensemble des chemins par lien. (pour tel lien il existe tels chemins).
- Classifier les chemins en des paires ayant des liens en commun entre eux.

Tout action de routage ou de réservation est suivie systématiquement d'une opération de mise à jour des capacités et des coûts des liens empruntés ainsi que les coûts courants des chemins passants par chaque lien dont les paramètres ont subi des modifications.

Il est très rare [2] [6] [7] que plus qu'une panne touche un réseau en même temps. De ce fait, on pourra se servir efficacement de la protection partagée des chemins afin d'assurer d'une part une meilleure utilisation des ressources disponibles et d'autre part la sécurité des différentes connexions établies.

4.2.2- Protection partagée des chemins :

Dans cette partie, le même programme linéaire utilisé précédemment sera réutilisé mais la seule différence entre les deux techniques est la suivante : Pour chaque demande de connexion « d » dont le chemin principal est « p » et le chemin de protection est « q ». Le chemin principal est mis en place, et le chemin de

protection peut être réservé totalement, partiellement ou non réservé du tout. En effet, on va chercher à ce niveau s'il existe déjà un ou plusieurs chemins de protections avec lesquels on peut partager des liens sans qu'aucun conflit ne soit engendré.

4.2.3- Simulation et résultats :

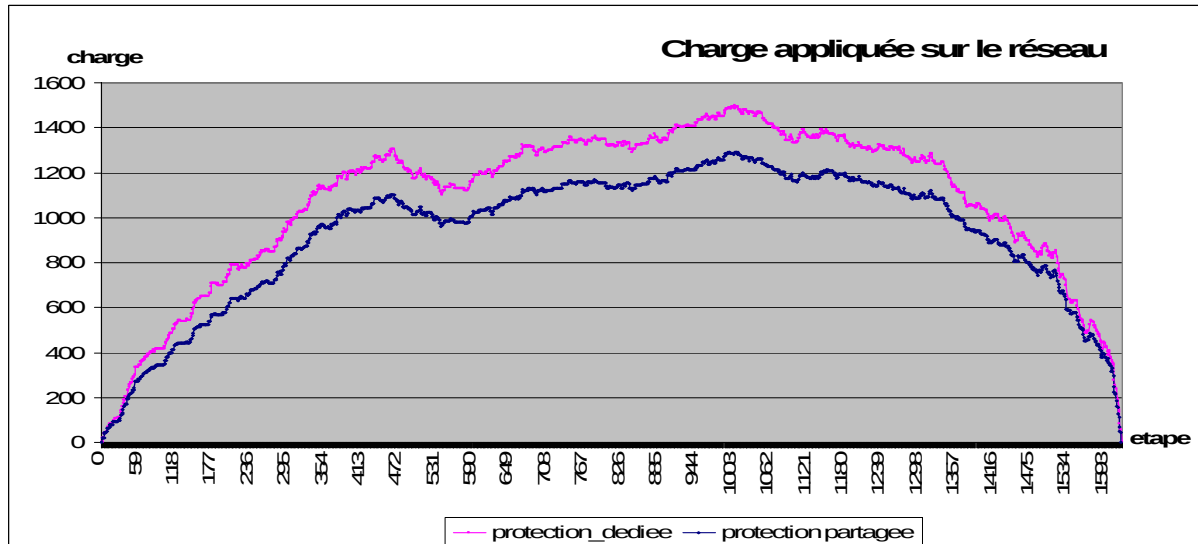


Figure 4.2 : Evolution de la charge appliquée sur le réseau

Comme prévu, la *figure 5.2* montre l'intérêt de la protection partagée en terme d'utilisation de ressources : en effet de l'étape 472 à l'étape 1100 on remarque une marge de différence égale à 200 unités de capacités entre les deux techniques de protection.

4.2.4- Protection dédiée et partagée des liens :

Contrairement à la partie précédente, on s'intéresse maintenant à mettre en place une technique de protection des connexions à base de la protection des liens composant le chemin de base. En premier lieu, on propose une formulation du problème à travers un programme linéaire à base des flots (modèle *arc-somme*). Ce programme linéaire servira dans toute la partie qui suivra à assurer le routage des demandes de connexion et la sélection des meilleurs chemins de protection sous les conditions courantes du réseau.

Protéger un chemin dans cette section revient à trouver pour chaque lien composant le chemin primaire choisi un chemin remplaçant :

- Disjoint de tout lien du chemin de base
- Les deux nœuds aux extrémités de ce remplaçant sont les mêmes nœuds d'extrémité du lien protégé.

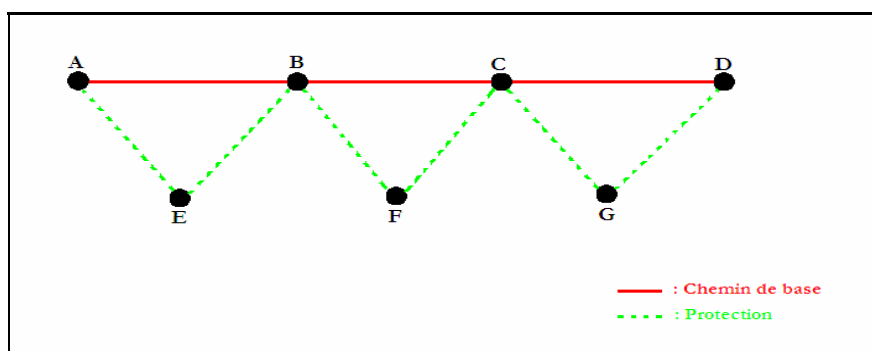


Figure 4.3 : protection de chemin à travers la protection de ses liens

Sur cet exemple, la demande de connexion à router est « A, D » : le chemin de base choisi est : " $A \rightarrow B \rightarrow C \rightarrow D$ " et les chemins de protection par lien sont mentionnés dans le tableau suivant :

Lien protégé	Chemin de protection
" $A \rightarrow B$ "	" $A \rightarrow E \rightarrow B$ "
" $B \rightarrow C$ "	" $B \rightarrow F \rightarrow C$ "
" $C \rightarrow D$ "	" $C \rightarrow G \rightarrow D$ "

Le modèle utilisé pour formuler le problème est appelé modèle « **sommet-arc** » et il nécessite une variable de flot par arc et par requête :

- $T = \{0, 1, 2, \dots, T_{\max}\}$: l'ensemble des étapes de la simulation.
- R : l'ensemble des requêtes de demande de connexion du scénario.
- $R_E(t)$: l'ensemble des requêtes générées à une étape t .
- V : ensemble de nœuds du réseau.
- A : Ensemble des arcs du réseau.
- x_{req}^a : la variable de décision binaire indiquant si les données de la requête « req » traversent l'arc « a » ou non. Si oui alors sa valeur sera 1 sinon elle prendra la valeur 0.
- $y_{req}^{a,b}$: la variable de décision binaire indiquant si des unités de capacité vont être réserver sur le lien « b » afin de protéger les données de la requête « req » traversent l'arc « a ». Si oui alors sa valeur sera 1 sinon elle prendra la valeur 0.
- C^a : coût relatif à l'utilisation de l'arc « a ».
- c^a : capacité disponible sur un arc « a ».
- $size_{req}$: les unités de capacité requise par une demande de connexion « req ».

Nous présentons ci-dessous le programme linéaire relatif à une étape unique. Ainsi, pour résoudre notre problème d'origine ce programme doit être reproduit pour chacune des étapes de la simulation :

$$\text{Objectif: } \min \sum_{req \in R_E(t)} \sum_{a \in A} x_{req}^a * C^a * size_{req} + \sum_{req \in R_E(t)} \sum_{a \in A} \sum_{b \in A} y_{req}^{a,b} * C^b * size_{req}$$

Sous les contraintes :

- Pour chaque requête « req » : ($s \rightarrow d$) :

$$\left\{ \begin{array}{l} \sum_{(i,v) \in A} x_{req}^{(i,v)} - \sum_{(v,j) \in A} x_{req}^{(v,j)} = 0 \\ \sum_{(i,s) \in A} x_{req}^{(i,s)} - \sum_{(s,j) \in A} x_{req}^{(s,j)} = -1 \\ \sum_{(i,d) \in A} x_{req}^{(i,d)} - \sum_{(d,j) \in A} x_{req}^{(d,j)} = 1 \end{array} \right. \quad \text{Si } v \neq s \text{ et } d. \quad (1)$$

$$\left\{ \begin{array}{l} \sum_{(l,v) \in A} y_{req}^{(i,j),(l,v)} - \sum_{(v,k) \in A} y_{req}^{(i,j),(v,k)} = 0 \\ \sum_{(i,k) \in A} y_{req}^{(i,j),(i,k)} - \sum_{(l,i) \in A} y_{req}^{(i,j),(l,i)} \geq x_{req}^{(i,j)} \\ \sum_{(l,j) \in A} y_{req}^{(i,j),(l,j)} - \sum_{(j,k) \in A} y_{req}^{(i,j),(j,k)} \geq x_{req}^{(i,j)} \end{array} \right. \quad (2)$$

$$- \forall a \in A \quad \sum_{req \in R_E(t)} x_{req}^a + \sum_{req \in R_E(t)} \sum_{(l,k) \in A} y_{req}^{a,(l,k)} \leq c^a \quad (3)$$

$$- \forall a \in A, req \in R_E(t) \quad y_{req}^{a,(l,k)} \leq x_{req}^a \quad (4)$$

$$- \forall a \in A, req \in R_E(t) \quad x_{req}^a + \sum_{(l,k) \in A} y_{req}^{(l,k),a} \leq 1 \quad (5)$$

$$- req \in R_E(t), \forall a \in A, \forall b \in A \quad x_{req}^a = \{0,1\} ; y_{req}^{a,b} = \{0,1\}$$

Notre objectif étant la minimisation du coût total du routage des demandes de connexion ainsi que leurs protection, la contrainte 1 assure la conservation du flot tout au long du chemin de base tandis que la contrainte 2 assure la conservation du flot tout au long du chemin de protection. La contrainte 3 exige que le flot circulant sur un Arc « a » ne dépasse pas la capacité disponible sur cet arc. Un lien « l,k » ne peut être réservé pour protéger un line « a » que si ce dernier fait partie du chemin de base : cette condition est exprimé à travers la contrainte 4. La dernière contrainte confirme la disjonction du chemin de base et de toute solution de protection pour une même requête.

4.2.5- Simulations et Résultats :

En terme de charge totale appliquée sur le réseau, la protection partagée par liens marque plus de performance : en effet, presque tout au long de la simulation on a remarqué que la charge appliquée pour la protection dédiée est plus grande que celle appliquée pour la protection partagée. Mais la chose qui est un peu bizarre, c'est qu'un certain moment : étape 1020 les valeurs des charges appliquées se coïncident. Alors on a essayé de voir l'évolution de la charge appliquée sur le lien le plus congestionné et ce pour voir si le réseau avait toujours assez de ressources pour accueillir les demandes de connexions générées.

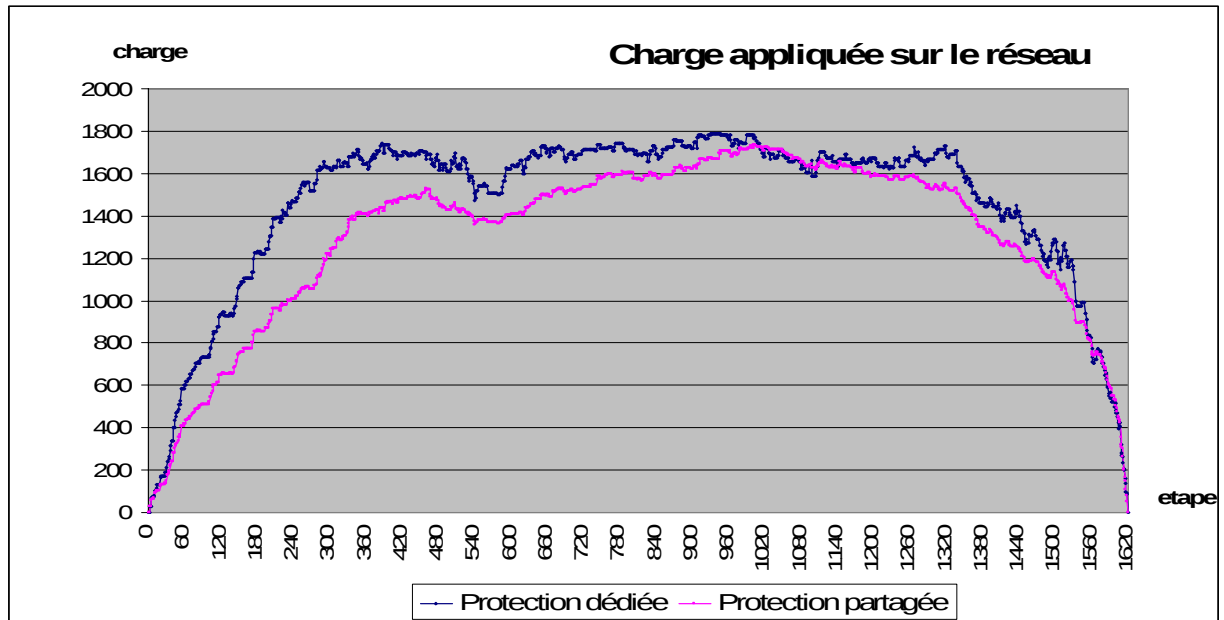


Figure 4.4 : Evolution de la charge appliquée sur le réseau

La figure 4.5 montre clairement que le lien le plus congestionné était presque toujours saturé. Alors une interprétation logique pour la coïncidence des valeurs des charges appliquées sur le réseau pour la protection partagée et dédiée est que le réseau n'a plus suffisamment de capacité et a du rejeté des demandes, ainsi la différence entre les charges appliquées se réduit de plus en plus puisque avec la protection partagée on peut assurer plus de capacité disponible. Et pour valider cet hypothèse, on a tracé sur la figure 4.6 l'évolution du nombre de demandes rejetés à travers le temps.

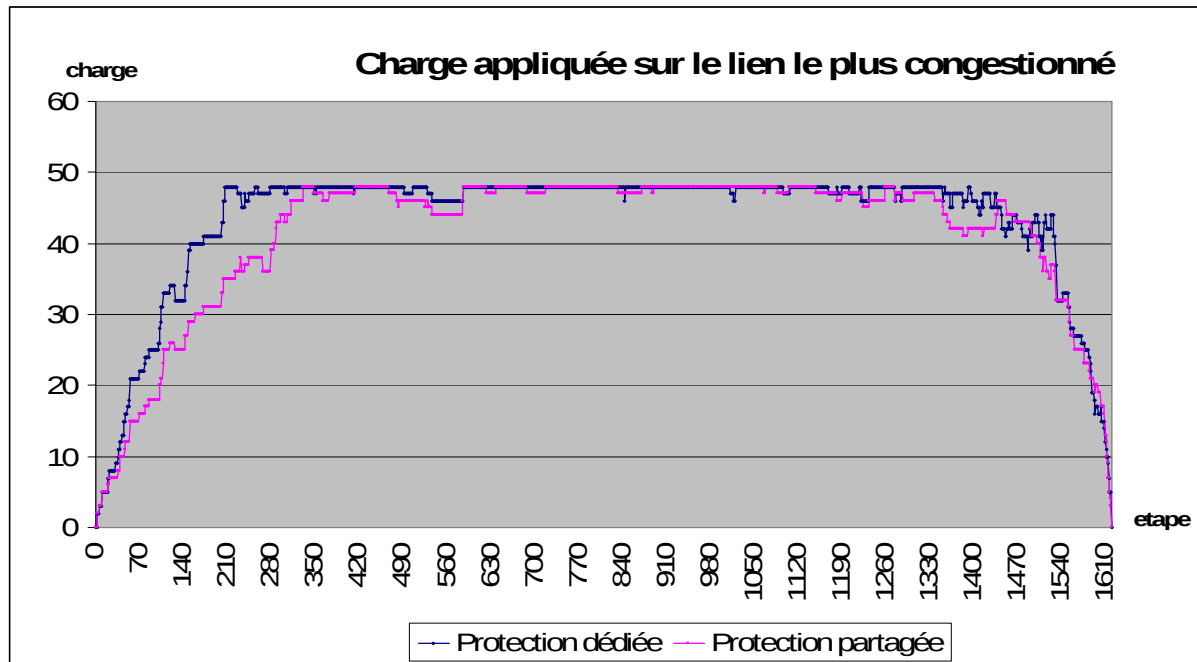


Figure 4.5 : charge appliquée sur le lien le plus congestionné

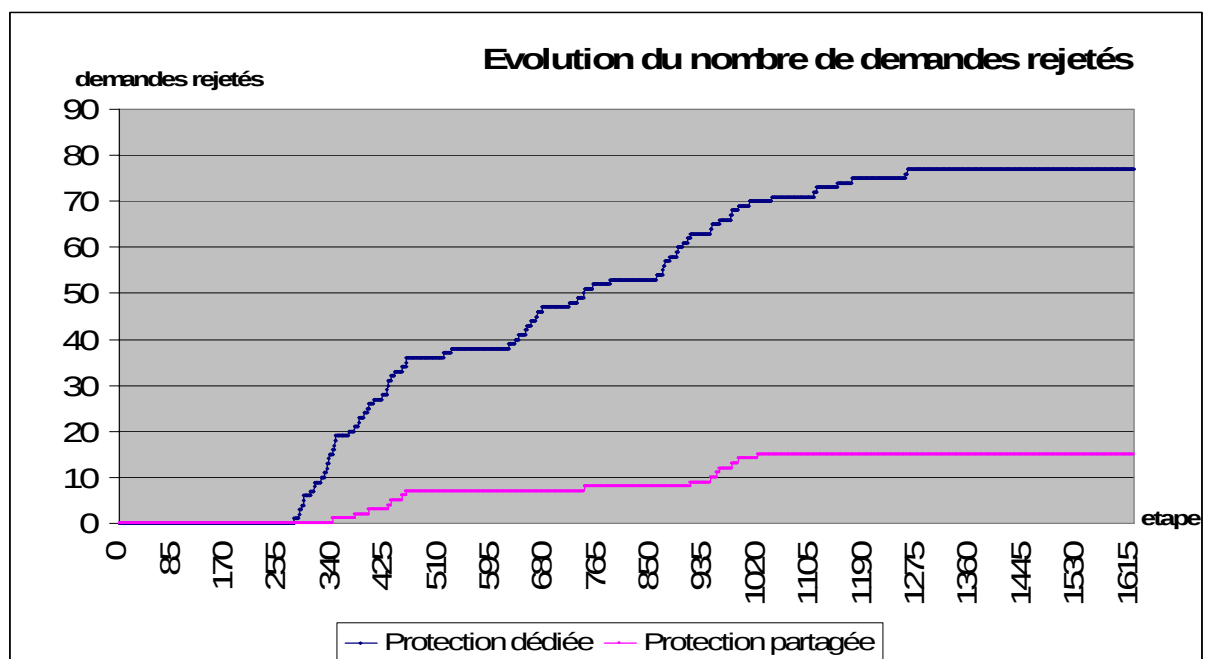


Figure 4.6 : Cumul du nombre de demandes rejetés

Maintenant, on va comparer les deux techniques de protection par chemins et liens et ce en terme de ressources utilisée : sur la *figure 4.7*, il est clair que la technique de protection par chemins est beaucoup moins exigeante et on remarque que la différence est presque égale à 400 unités de capacités de l'étape 414 et l'étape 1450. Une différence qui n'est pas négligeable et même très importante, mais en contre partie l'avantage de la protection par liens réside dans le temps de restauration puisque c'est plus facile et rapide de basculer un trafic juste d'un lien à un chemin dont les extrémités sont les mêmes que de le basculer sur un autre chemin complètement différent de l'original.

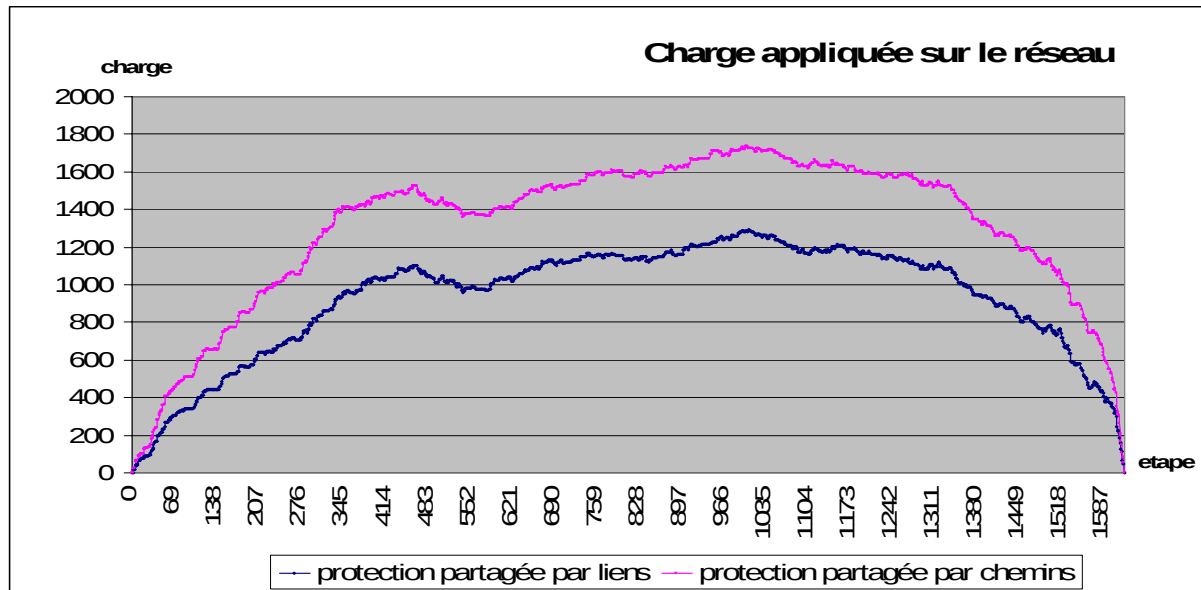


Figure 4.7 : Comparaison charge appliquée sur réseau entre protection partagée liens et chemins

Ainsi, deux techniques de protection seulement ont été étudiées, mis en place et évaluées sur notre modèle de trafic et notre modèle support de réseau et ce à cause des contraintes relatives à la durée du stage. Toutefois, on a voulu finir cette partie par appliquer, au dessus de la technique de protection des connexions par chemins, les deux heuristiques de reconfigurations déjà élaboré dans le chapitre précédent. Les deux heuristiques seront implémentées sur un trafic qu'on lui assurera une protection dédiée.

En appliquant l'heuristique basée sur le gain en coût, les résultats visualisés sur la charge globale appliquée étaient :

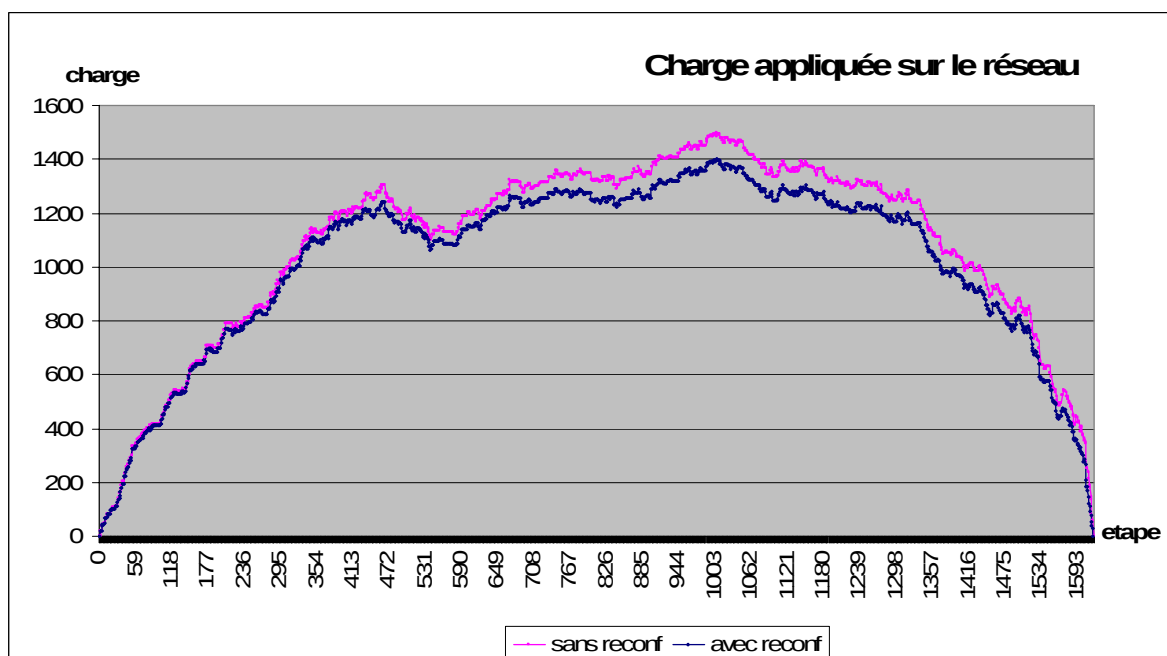


Figure 4.8 : charge appliquée sur réseau sans et avec reconfiguration

Pour cette heuristique, on s'est intéressé qu'à observer l'évolution de la charge totale appliquée car c'est le plus important critère que cible améliorer cette heuristique. Comme il a été prévu on a arrive visiblement à diminuer cette charge même pour un trafic qui n'est pas très dense.

En appliquant l'heuristique basée sur le dépassement d'un certain seuil d'occupation des liens, les résultats enregistrés pour la charge appliquée sur le lien le plus congestionné étaient :

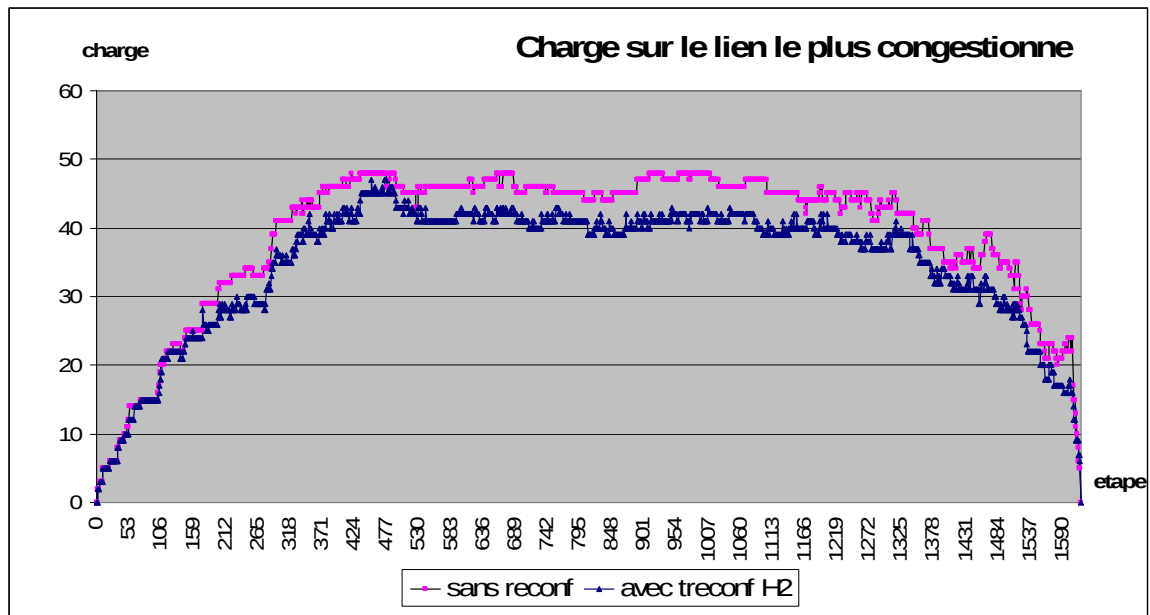


Figure 4.9 : charge appliquée sur le lien le plus congestionné sans et avec reconfiguration (H2)

Pour cette heuristique, on s'est intéressé qu'à observer l'évolution de la charge appliquée sur le lien le plus congestionnée car c'est le plus important critère que cible améliorer cette heuristique. D'après la *figure 4.9* on s'est éloigné clairement de la saturation complète des liens : une telle situation permet aisément l'accommodation de toute connexion sans avoir à se soucier de disponibilité de capacité.

4.2.6- Conclusions :

A travers les deux techniques de protection des connexions qu'on a étudié à savoir la protection des connexions par chemins et la protection par liens, on peut conclure que :

- Dans ces deux cas de figure, une protection partagée est considérablement plus efficace et performante en terme de ressources utilisées.
- La protection par liens est une technique qui nécessite, pour être mise convenablement en place, une quantité de ressources beaucoup plus grande que celle exigée par la technique de protection des connexions par chemins.
- La reconfiguration, appliquée à travers les deux heuristiques élaborés dans le chapitre précédent sur un scénario de connexions établies et protégées via la technique de protection dédiée par chemins, confirme une autre fois son importance ainsi que son efficacité.

CONCLUSIONS ET PERSPECTIVES

Ce projet m'a permis d'étudier et d'analyser deux problèmes importants pour les réseaux WDM à savoir : le problème de reconfiguration et le problème de protection des connexions. A partir d'un modèle support de notre choix, on a étudié dans un premier temps l'impact du choix d'une fonction de coût des liens sur la mise en place des demandes de connexion sur le réseau. Quatre fonctions de coût ont été mises à l'épreuve mais seule la fonction de coût à croissance parabolique s'est montrée la plus performante face à d'autres fonctions de coût qui ont parfois amené le réseau à se placer dans des situations de saturation et ainsi à rejeter des demandes de connexions.

Face à un trafic dynamique, l'utilisation des ressources disponibles sur le réseau diverge petit à petit d'une situation initialement optimale. A ce niveau, la reconfiguration intervient pour assurer une solution pouvant, suivant les coûts sacrifiés, varier d'une solution acceptable à une solution optimale. A travers deux heuristiques mises en place, une (**H1**) basée sur le gain en terme de coûts et l'autre (**H2**) sur le dépassement d'un certain seuil d'occupation des liens, on a remarqué que :

- Ces deux heuristiques diffèrent complètement d'objectifs d'où des résultats obtenus.
- L'heuristique H2 minimisant le plus grand taux d'occupation enregistré assure plus de survie au réseau au détriment d'une augmentation de charge totale appliquée.
- L'heuristique H1 applique des reconfigurations à toutes les connexions en ciblant tous les liens sans exception et ce afin d'assurer une meilleure utilisation des ressources disponibles sur le réseau.

Chacune des deux heuristiques cherche à améliorer des critères différents, alors on a essayé de les évaluer à travers le nombre de reconfigurations effectués. Pour H2, le nombre de reconfigurations était justifié et acceptable. Tandis que pour H1, le nombre de reconfigurations était très grand pour une diminution très faible de la charge totale appliquée. Les améliorations apportées ciblaient les reconfigurations appliquées à des connexions qu'on basculaient soit d'un chemin vers un autre de même longueur ou un autre de longueur plus grande. Les résultats étaient satisfaisants et on a pu affirmer que :

- Cette heuristique appliquait des reconfigurations inutiles pour améliorer l'utilisation des ressources.
- Cette heuristique nécessite un mécanisme de filtrage pour écarter tout type de reconfiguration non nécessaire ; dans notre cas on a travaillé avec une technique de seuils et on a remarqué que plus la valeur ce seuil augmentait plus les performances s'amélioraient.

Pour un réseau WDM, appliquer une reconfiguration par étape est largement suffisant. Effectuer plusieurs reconfigurations à une étape ne me semble pas améliorer significativement les performances du réseau. Par contre, cette technique (une par étape) ne peut résister face à trafic très dense où la fréquence des changements (entrée et sortie de connexions) est assez élevée.

Le problème de protection, étudié partiellement dans notre projet, admet différentes solutions soit à base de chemins ou de liens. On a remarqué en premier lieu que dans les deux cas de figures la protection partagée minimise potentiellement la quantité de ressources utilisée et que la protection par chemins applique globalement moins de charge que celle appliquée par la technique de protection par liens.

A ce stade, il est très clair qu'il reste beaucoup de choses qu'on peut aborder dans le futur soit sur le problème de la reconfiguration soit sur le problème de la protection. Quelques pistes de perspectives peuvent être définies comme suit:

- Chercher à trouver une certaine fonction *auto-adaptative* permettant, à partir de quelques paramètres (charge appliquée sur les liens, longueur de chemins, ...), de valider ou non une action de reconfiguration.
- Etudier le problème de reconfiguration pour un autre cas de figure complètement différent où d'autres types de problèmes nécessitent d'être résolus plus prioritairement. On peut citer par exemple le problème de dépendance entre les actions de reconfigurations dans le cas d'un réseau avec peu de ressources et où on aura besoin impérativement de trouver le meilleur des ordres permettant la minimisation des interruptions.
- Définir exactement ce qu'est une fonction efficace et performante en rapport nombre de reconfigurations effectuées et utilisation des ressources sur le réseau.
- Etudier d'autres techniques de protection et comparer leurs performances par rapport à celles des deux solutions de base à savoir la protection par chemins et la protection par liens.
- Appliquer les heuristiques de reconfigurations élaborés et analysés aux cas de scénarios de connexions établies et protégés via les techniques de protection partagée où le problème se montre plus complexe. En effet, la reconfiguration dans ce rapport a été appliqué seulement sur un scénario de connexion à protection dédiée où le travail consistait à basculer seulement les chemins de bases vers d'autres chemins moins coûteux (opération n'entraînant aucun conflit).

Bibliographie

- [1] S. Xu, L. Li and S. Wang, « Dynamic Routing and Assignment of Wavelength algorithms in Multi-fiber WDM Networks,” IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 18, N°. 10, OCTOBER 2000.
- [2] D. Allen, I. Ismail, J. Kennington and E. Olinick, “An Incremental Procedure for Improving Path Assignment in a Telecommunications Network”. Technical Report 01-EMIS-04, Revised September 5, 2003.
- [3] A. Gençata and B. Mukherjee, “Virtual-Topology Adaptation for WDM Mesh Networks Under Dynamic Traffic”. IEEE/ACM 2003 Transactions on Networking; vol. 11, issue 2, pp. 236-247, April 2003
- [4] B. Gillani, R. D. Kent, and A. K. Aggarwal “Topology Reconfiguration Mechanism for Traffic Engineering in WDM Optical Network”. IEEE 2005.
- [5] N. CJose and A. K. Somani, “Connection Rerouting/Network Reconfiguration”. DRCN , Banff, Alberta, Canada, October 19-22, 2003.
- [6] S. Rammamurthy, Laxman Sahasrabuddhe and Biswanath Mukherjee, “Survivable WDM Mesh Networks” Journal Of Lightwave Technology, Vol. 21, NO. 4, APRIL 2003.
- [7] H. Wang, E. Modiano and M. Médard, “Partial Path Protection for WDM Networks: End-to-End Recovery Using Local Failure Information” Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium page(s): 719- 725.
- [8] Vishal, Anand, Sunit Chauhan, Chuning Qiao, “Sub-Path Protection: A New Framework for Optical Layer Survivability and its Quantitative Evaluation,” CSE Departmental Technical Reports, January 2002.